Open/Free SoC Graphics Update

2013 Sept - Rob Clark

History

- a couple years ago, it looked hopeless
- but then libv drew some triangles
- we have come a long ways in the last year!

- etnaviv
  - gallium driver for vivante

- grate
  - gallium driver for tegra

- lima
  - classic/dri driver for mali 200/400

- freedreno
  - gallium driver for adreno 2xx/3xx
  - plus xf86-video-freedreno and msm drm/kms

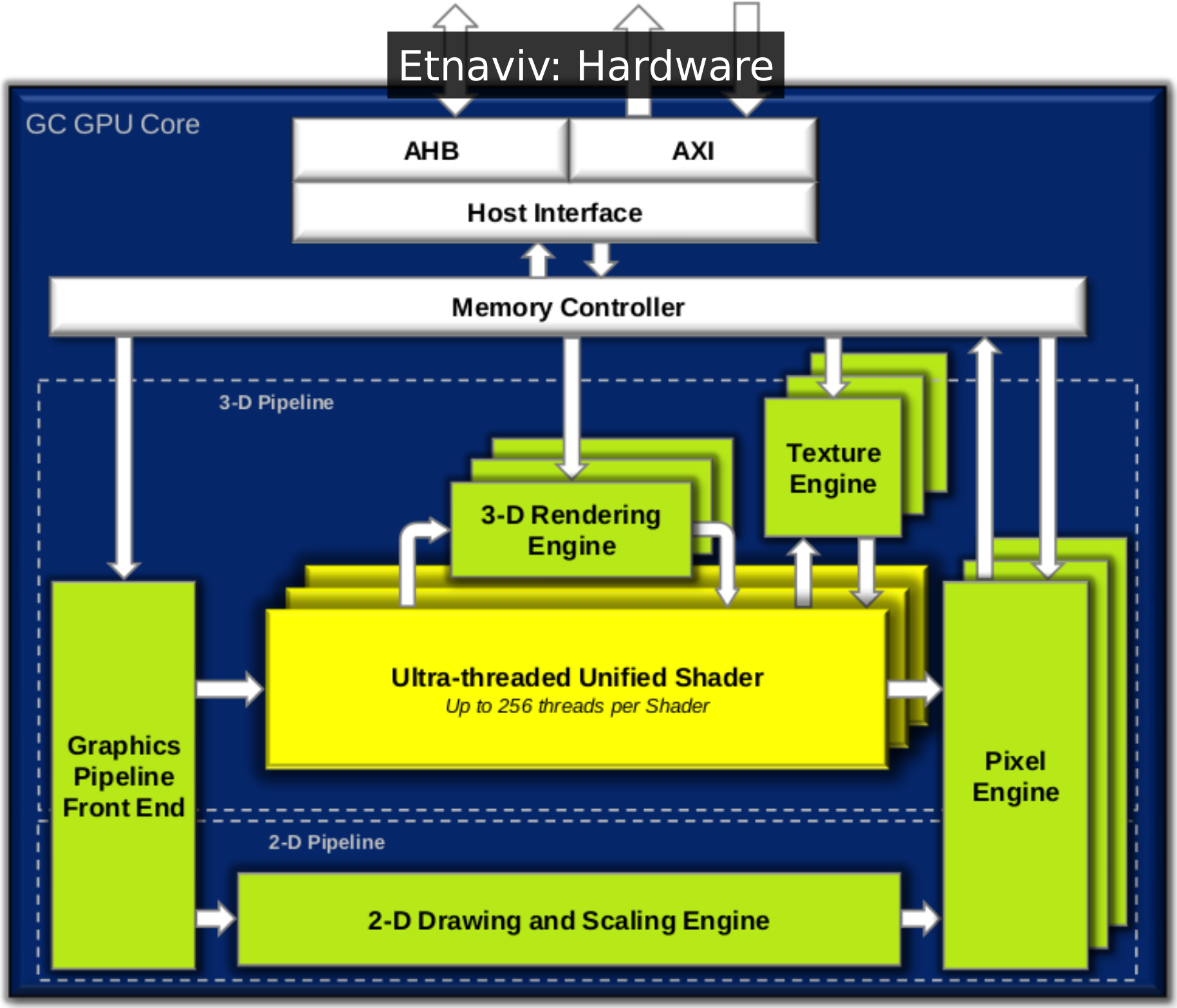
Etnaviv: Vivante

- OpenGLES 2.0
  - GC2000+: OpenGLES 3.0 and OpenCL
- Unified shader ISA
  - Vertex texture fetch
- 2x/4x MSAA
- IMR (not tiler)
- Formats
  - Textures: 2D, cubemap
  - Texture compression: DXT1-5, ETC
  - Depth: 16b or 24b
  - Stencil: 8b
  - Index: 8b, 16b, or 32b
- Modular
  - 3D, 2D, compositing, and VG engines, each optional
  - But mostly talking about 3D and 2D

Etnaviv: Devices

- SolidRun CuBox (GC800)
  - Marvell Armada 510 SoC
  - 800MHz dual-issue ARM PJ4
  - 1GiB DDR3
  - 1080p Video Decode Engine
  - HDMI, gigabit ethernet, eSata, etc
- GK802 HDMI dongle (GC2000)
- GCW Zero (GC860)
  - Ingenic JZ4770 1GHz MIPS processor
  - 3.5" LCD (320x240)
  - 512MiB DDR2
  - mini-HDMI, A/V port, 802.11b/g/n
- Utilite (imx6)

Etnaviv: Hardware



Etnaviv: Shader ISA

- Unified Shader
  - vec4 instructions
    - scalar integer instructions on GC2000
  - 128b instruction encoding

- Precision: FP32

```
; gl_Position = mvpMatrix * in_position
MUL t4, u0, t0.xxxx, void
MAD t4, u1, t0.yyyy, t4
MAD t4, u2, t0.zzzz, t4
MAD t4, u3, t0.wwww, t4
```

Etnaviv: Status

- Working gallium driver
  - But using fbdev backend only
  - Needs help for xorg DDX, DRM/DRI2 support, etc

- Very fast progress
  - r/e work started late 2012


2013-07-21
18 0 0 0
100 18
19 0 0 0
100 19
19 0 0 0
100 19
25 0 0 0
100 25


2013-07-27


2013-08-16


2013-08-28

Grate: Tegra

- OpenGLES 2.0
- Separate Vertex and Fragment shaders
  - very minimalist: no loops, etc
  - but good performance through massive # of cores
- 2x/4x MSAA (T40)
- IMR (not tiler)
- Formats
  - Textures: 2D, cubemap
  - Depth: 20b (T30) or 24b (T40)
  - Stencil: 8b
  - Index: 8b, 16b

Grate: Devices

- Tegra2
  - AC-100
  - Trimslice
- Tegra3
  - Nexus7 (original)
- Tegra4
  - Shield

Grate: Vertex Shader ISA

- VLIW vec4 (ALU) + scalar (SFU) co-dispatch
  - ALU: MOV, MUL, ADD, DP3, DP4, etc
  - SFU: SIN, COS, RCP, RSQ, LG2, EX2

- Precision: FP32

```
; gl_Position = mvpMatrix * in_position
mul r1.xyzw, v0.xxxx, c0.xyzw
mad r1.xyzw, v0.yyyy, c1.xyzw, r1.xyzw
mad r1.xyzw, v0.zzzz, c2.xyzw, r1.xyzw
mad o0.xyzw, v0.wwww, c3.xyzw, r1.xyzw
```

Grate: Fragment Shader ISA

- More weird, three different instruction streams
  - VAR/SFU - varying interpolate and special function unit
  - TEX - texture lookup
  - ALU - arithmetic logic unit

- ALU - packets of 3 or 4 scalar instructions
  - 3x 64b instructions + embedded constant
  - or 4x 64b instructions
  - only four opcodes:
    - MAD: rD = rA * rB + rC
    - MIN: rD = min(rA * rB, rC)
    - MAX: rD = max(rA * rB, rC)
    - CSEL: conditional select

- Precision: FP20

```
; gl_FragColor = texture2D(tex, vec2(0.0));
; gl_FragColor.r += gl_FragColor.a > 0.5 ? gl_FragColor.g : gl_FragColor.b;
ALU:002 mad r2.hl, #0, #1, #0
ALU:002 mad r3.hl, #0, #1, #0
TEX:002 tex s0
ALU:003 mad_lt r0.__, -r3.h, #1, ec0
ALU:004 cnd r1.hl, -x0_half, r3.l, r2.h
ALU:004 mad r2._l, d0.hl, #1, r2.l
EXP:004 export alu
```

Grate: Status

- Early research stage
  - command-stream capture and replay
  - basic GL state understood
  - vertex shader ISA understood
  - main work is on fragment shader ISA currently
  - basic gallium driver (clears)

Lima: Mali 200/400/t6xx

- Mali 200/400
  - OpenGLES 2.0
  - Separate Vertex (GP) and Fragment (PP) shaders
  - Mali 400 available with 1-4x PP
  - Tile based IMR (16x16)
  - Textures: 2D, cubemap

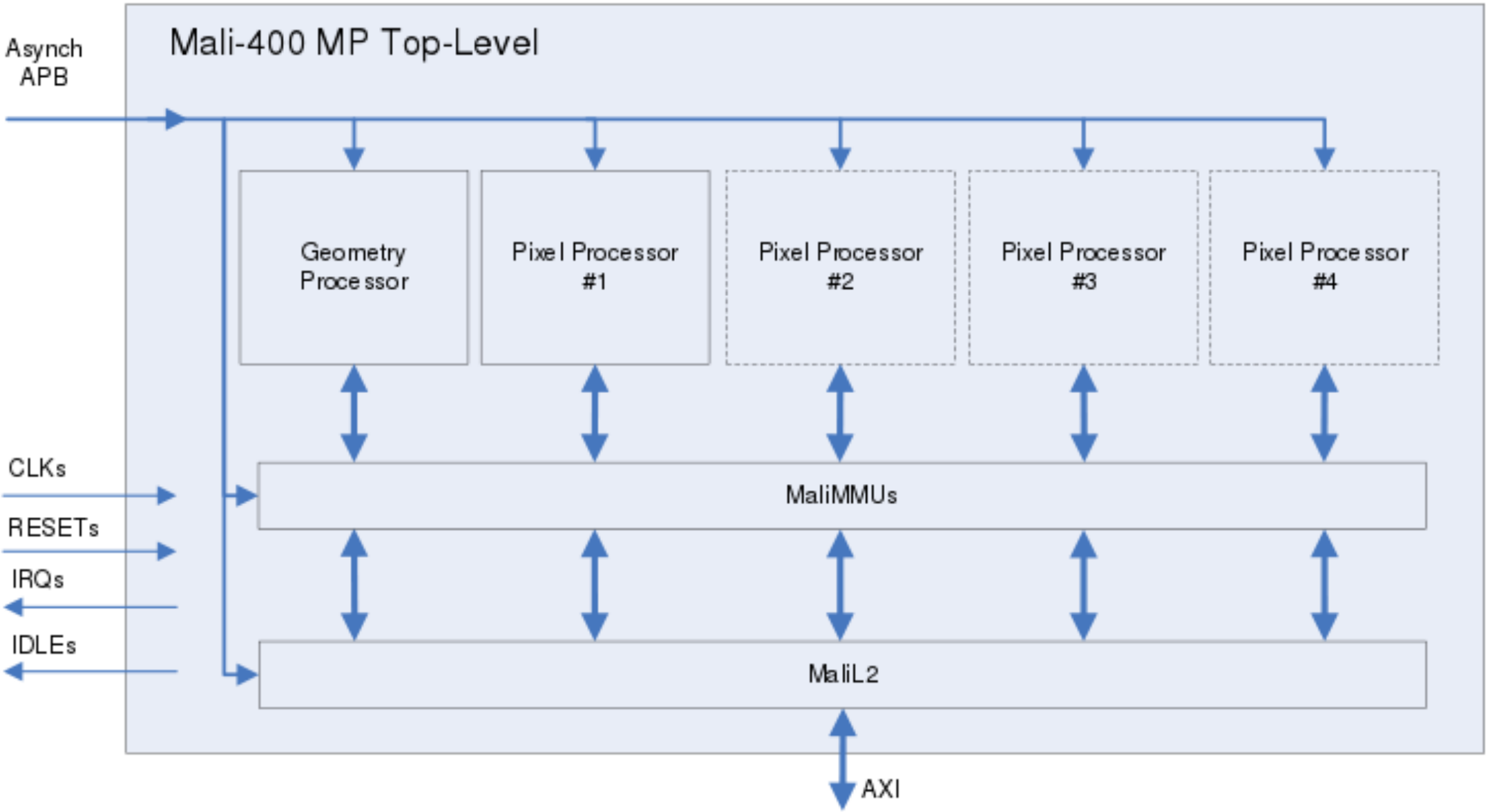
- Mali t6xx
  - OpenGLES 2.0 / 3.0
  - OpenCL 1.1
  - Unified Shader ISA
  - Various # of shader cores and ALU widths
  - Tile base IMR
  - Textures: 2D, cubemap, 3D

Lima: Devices

- Mali-t6xx
  - Samsung Exynos5 (Chromebook, Nexus10)

- Mali-200/400
  - AMLogic 8726-M (Zenithink C71)
  - Allwinner A10 (Mele A1000, MK802)
  - Samsung Exynos4 (Galaxy S2/S3/Tab/Note)
  - Telechips 8902, 8803

Lima: Hardware



Lima: 200/400 Vertex Shader

- Single-threaded but deeply pipelined
- Each fixed length VLIW instruction has fields for:
  - 2 addition ALU's
  - 2 multiplication ALU's
  - 1 complex ALU
  - 1 pass-through ALU
  - 1 attribute load
  - 1 register load
  - 1 uniform/temp load
  - 1 varying/register store
- No explicit output registers
  - Outputs from ALU of previous instruction routed directly to ALU in current instruction
  - 16 temporary registers to use when compiler schduling cannot route directly (1 load/store per instr)
- Precision: FP32

```
; gl_Position = mvpMatrix * in_position
uniform.load(1), attribute.load(0), mul[0].mul(uniform.z, attribute.y),
                    mul[1].mul(uniform.y, attribute.y), store[0].register(0, mul[1].out, mul[0].out);
uniform.load(1), mul[0].mul(uniform.x, attrib.y[1]), mul[1].mul(uniform.w, attrib.y[1]),
                    store[1].register(0, mul[0].out, unused);
uniform.load(0), attribute.load(0), mul[0].mul(uniform.z, attribute.x), mul[1].mul(uniform.y, attribute.x);
uniform.load(0), attribute.load(0), mul[0].mul(uniform.x, attrib.x[1]), mul[1].mul(uniform.w, attrib.x[1]),
                    acc[0].pass(mul[0].out[1]), acc[1].pass(mul[1].out[2]), pass.pass(mul[1].out[1]),
                    store[1].register(0, unused, mul[0].out);
uniform.load(2), mul[0].mul(uniform.z, attrib.z[1]), mul[1].mul(uniform.w, attrib.z[1]),
                    acc[0].pass(attrib.z[1]), acc[1].add(mul[1].out[1], acc[1].out[1]), complex.pass(uniform.y),
                    pass.pass(uniform.x), store[0].register(4, unused, mul[0].out);
uniform.load(3), attribute.load(0), register[1].load(0), mul[0].mul(complex.out[1], acc[0].out[1]),
                    mul[1].mul(uniform.w, attribute.w), acc[0].add(acc[0].out[2], register[1].y),
                    acc[1].add(acc[1].out[1], mul[1].out[1]), complex.pass(pass.out[2]),
                    store[1].register(4, mul[0].out, unused);
uniform.load(3), register[0].load(0), register[1].load(0), mul[0].mul(pass.out[2], acc[0].out[2]),
                    mul[1].mul(uniform.z, attrib.w[1]), acc[0].add(complex.out[1], register[1].x),
                    acc[1].add(acc[1].out[1], mul[1].out[1]), complex.pass(attrib.w[1]),
                    pass.pass(register[1].w), store[0].register(3, mul[1].out, unused);
mul[0].pass(acc[0].out[1]), mul[1].pass(mul[0].out[1]), acc[0].pass(complex.out[1]),
                    acc[1].pass(mul[0].out[2]), complex.pass(acc[0].out[2]);
uniform.load(3), register[0].load(4), register[1].load(0), mul[0].complex2(acc[1].out[2], acc[1].out[2]),
                    mul[1].mul(uniform.y, acc[0].out[1]), acc[0].add(pass.out[2], register[1].z),
                    acc[1].add(complex.out[1], register[0].y), complex.rcp(acc[1].out[2]), pass.pass(acc[1].out[2]);
uniform.load(3), mul.complex1(complex.out[1], mul[0].out[1], complex.out[1], pass.out[1]),
                    acc[0].add(mul[0].out[2], acc[1].out[2]), acc[1].add(acc[0].out[1], mul[1].out[2]),
                    complex.pass(uniform.x), pass.pass(acc[0].out[2]);
uniform.load(4), register[0].load(0), register[1].load(3), mul[0].pass(uniform.y),
                    mul[1].mul(complex.out[1], pass.out[1]), acc[0].add(acc[1].out[2], register[1].x),
                    acc[1].add(acc[0].out[1], mul[1].out[2]), pass.pass(uniform.z);
uniform.load(6), mul[0].mul(acc[0].out[1], pass.out[1]), mul[1].mul(acc[1].out[1], mul[0].out[1]),
                    acc[1].add(acc[1].out[2], mul[1].out[1]), pass.clamp(mul[0].out[2]);
uniform.load(4), mul[0].mul(acc[1].out[1], uniform.x), mul[1].mul(mul[0].out[1], pass.out[1]);
uniform.load(5), mul[0].mul(mul[1].out[2], pass.out[2]), mul[1].mul(mul[0].out[1], pass.out[2]),
                    acc[0].pass(pass.out[2]), acc[1].add(mul[1].out[1], uniform.z);
uniform.load(5), mul[1].pass(acc[0].out[1]), acc[0].add(mul[0].out[1], uniform.y),
                    acc[1].add(mul[1].out[1], uniform.x), complex.pass(acc[1].out[1]),
                    store[0].varying(0, acc[1].out, acc[0].out), store[1].varying(0, complex.out, mul[1].out);
```

Lima: 200/400 Fragment Shader

- 128 thread barrel processor
- VLIW, variable length instr (32b aligned, up to 576b)
- 6 vec4 instr (encoding supports up to 12) plus 4 special:
  - const 0, const1, tex sample result, unform fetch result
  - but pipeline registers - direct connection between two units in pipeline
- Precision: FP16

```
; gl_FragColor = clamp(
;     vColor * texture2D(uTexture0, vTexCoord0),
;     0.0, 1.0);
$0 = varying[0];

varying[1].xy,
^texture = sampler2D(0),
$0 = ^vmul = clamp($0 * ^texture, 0.0, 1.0),
sync, stop;
```

Lima: Status

- Main focus so far: Mali 200/400
  - Mesa classic/dri driver starting to work
    - es2gears, textured cube, etc
  - Need to hook up cwabbott's compiler backend

- Some preliminary investigations on Mali t6xx compiler

Freedreno: Adreno 2xx/3xx

- Adreno 2xx
  - OpenGLES 2.0
  - Unified Shader ISA
    - VLIW vec4 + scalar co-dispatch

- Adreno 3xx
  - OpenGLES 2.0 / 3.0
  - OpenCL 1.1 (embedded profile, no double)
  - Unified Shader ISA
    - explicitly pipelined scalar

- Common
  - Textures: 2D, cubemap, 3D
  - 2x/4x MSAA
  - Tile Based IMR: 256KiB-1MiB GMEM/OCMEM
    - driver explicitly handles tiling (incl. restore/resolve)
    - hw binning pass to avoid duplicated vertex processing

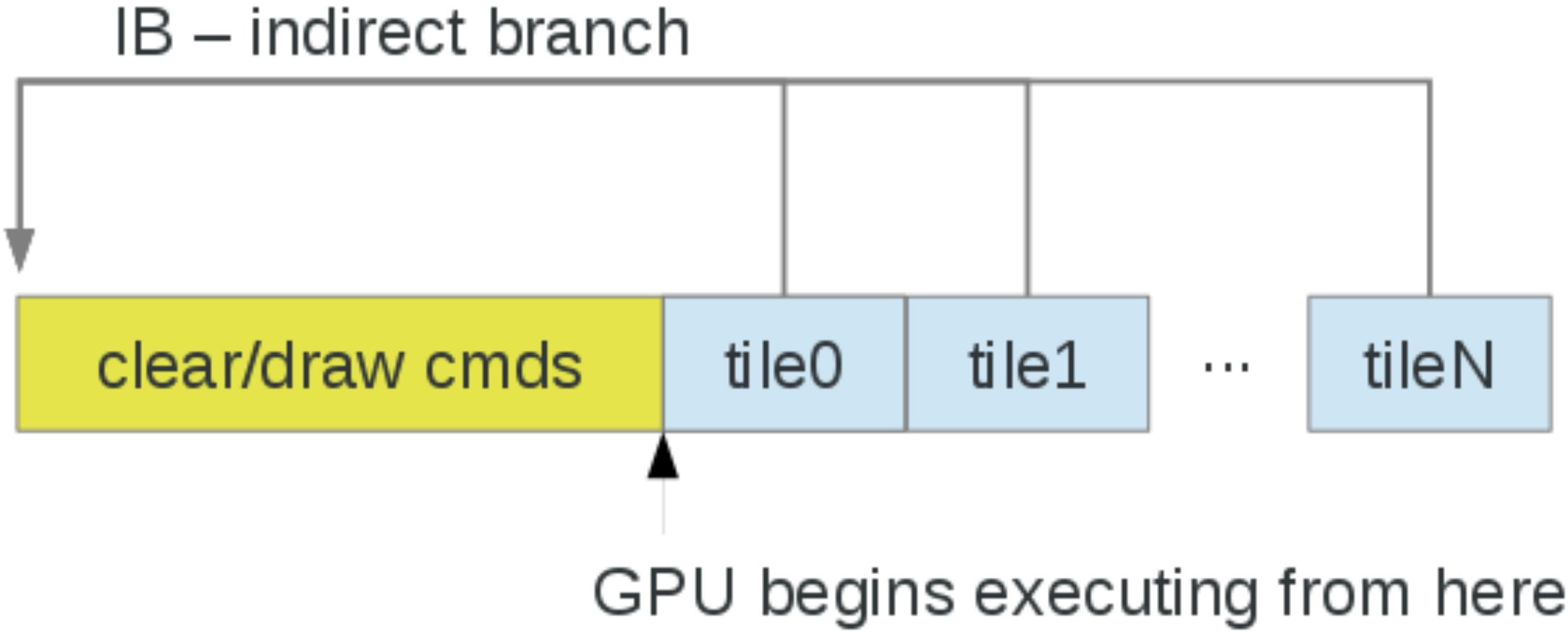
Freedreno: Devices

- Adreno 2xx
  - Snapdragon S3 (HP TouchPad)
  - Freescale iMX5 (Quickstart, Efika-MX)

- Adreno 3xx
  - Snapdragon S4 Pro (Nexus4, ifc6410)
  - Snapdragon-600 (Nexus7, Samsung Galaxy S4)
  - Snapdragon-800 (Nexus5, LG G2, Sony Xperia Z Ultra)

Freedreno: Tiling/GMEM

- "large", flexible tile buffer
- driver explicitly handles tiling
- draw/clear cmds built up normally (like IMR)
- flush: build restore/IB/resolve cmds



Freedreno: a2xx ISA

- 96 bit encoding FETCH/ALU
- 48 bit encoding CF
  - like r600, all CF instructions first
- Precision: FP32

```
EXEC ADDR(0x3) CNT(0x4)
     FETCH: VERTEX R1.xyzw = R0.x FMT_1_REVERSE UNSIGNED
                    NORMALIZED STRIDE(0) CONST(1, 1)
   (S)ALU:   MULv        R0 = R1.wwww, C3
     ALU:   MULADDv     R0 = R0, R1.zzzz, C2
     ALU:   MULADDv     R0 = R0, R1.yyyy, C1
ALLOC POSITION SIZE(0x0)
EXEC ADDR(0x7) CNT(0x1)
     ALU:   MULADDv     export62 = R0, R1.xxxx, C0  ; gl_Position
ALLOC PARAM/PIXEL SIZE(0x0)
EXEC_END ADDR(0x8) CNT(0x0)
NOP
```

Freedreno: a3xx ISA

- 64 bit encoding, 7 instruction categories
  - cat0: flow control, kill
  - cat1: mov/convert
  - cat2: ALU (add, mul..)
  - cat3: 3src ALU (mad, sel)
  - cat4: complex (rcp, rsq..)
  - cat5: tex sample
  - cat6: load/store/atomic
- "repeat" field to increase instr density
- Compiler responsible for pipelining
  - cat1-3: results avail 3 instr slots later (-1 for mad 3rd src)
  - cat4-5: special ss/sy sync bits for dependent instr
- Precision: FP32/FP16/U32/U16/S32/S16

```
; gl_FragColor.x = dot(v1, v2);
mul.f hr0.x, hr0.x, hr1.x
nop
mad.f16 hr0.x, hr0.y, hr1.y, hr0.x
nop
mad.f16 hr0.x, hr0.z, hr1.z, hr0.x
nop
mad f16 hr0.x, hr0.w, hr1.w, hr0.x
end

; gl_FragColor = v1;
(rpt3)mov.f16f16 hr0.x, (r)hr1.x
```

Freedreno: Status

- Initial msm drm/kms driver in v3.12

- Working gallium driver
  - mesa 9.2 and master (but use master)
  - supports a2xx and a3xx
  - supports either msm drm/kms or android kgsl/fbdev

- Xorg - xf86-video-freedreno
  - uses z180 2d core on devices which have it
  - work-in-progress (but issues) XA state tracker
  - supports either msm drm/kms or android kgsl/fbdev

- Wayland/Weston support
  - msm drm/kms only

Freedreno: Status (cont)

- Supported
  - OpenGL 1.4 - on best-effort basis
  - OpenGL ES 1.0/2.0
  - Textures: 2D, cubemap, 3D (incl mipmap)

- TODO
  - MSAA
  - hw binning pass (game performance!)
  - compiler could be a lot better

- Known to be working
  - gnome-shell, xbmc, xonotic, openarena, etc

- Working but minor issues
  - etuxracer, supertuxkart (MSAA issue)

Resources

- etnaviv
  - Main Developer: Wladimir J. Van Der Laan (wumpus)
  - IRC: #etnaviv (freenode)
  - https://blog.visucore.com/
  - https://github.com/laanwj/etna_viv
- grate
  - Main Developer: Erik Faye-Lund (kusma)
  - IRC: #lima (on freenode)
  - https://github.com/grate-driver/
- lima
  - Developers: Luc Verhaegen (libv) and Connor Abbott (cwabbott)
  - IRC: #lima (on freenode)
  - http://limadriver.org/
- freedreno
  - Main Developer: Rob Clark (robclark)
  - IRC: #freedreno (freenode)
  - http://bloggingthemonkey.blogspot.com/
  - http://freedreno.github.io/