# FOSDEM 2013

Freenode: #freedreno Web: http://freedreno.github.com/

## Motivation: Lack of opensrc gfx on ARM

- Open Source is about freedom
  - If you have the src and the will, you have a way
    - New widget, new feature, new distro...
- For modern UI the GPU becomes more important
  - If you don't have the src, then you are limited by the blob
- Android is dominant because of the blob
  - Gives SoC vendors a single platform to support
  - Doesn't really care that platform drivers work in a clean/sane way or reusability outside of android
  - Either use android or unaccelerated
- As a result  $\rightarrow$  hacks
  - Boot to Gecko using android HALs
  - libhybris dynamic loader hacks to reuse blobs
  - But will just be all sorts of glue / duct tape
- But lima/mali gave some hope that things can change

## History

- 2d z180
  - Started working on intercepting/parsing 2d cmds in march 2012
  - Basic EXA (fill/solid/composite) working in Apr
  - After that, mostly sidetracked on 3d
  - Batching working in Oct
  - Still a bit in need of some love and debugging
- 3d a220
  - Intercepting and initial parsing 3d cmds in Apr
  - First renders with fdre end of Jun
    - Using hard-coded, pre-compiled shaders
  - Start on shader disassembler in early Jul
  - Shader assembler for fdre and of Jul
  - Gallium driver started Nov

#### Adreno Overview

- 3d core a2xx, a3xx
  - Origin: ATI/AMD Imageon
    - Similar heritage as r300/r600
  - Psuedo-TBDR
    - Hidden surface removal
    - Memory bandwidth reduction in common cases
    - GMEM macro-tile: 256KiB or 512KiB vs 16x16 or 32x32
    - Starting with a330, OCMEM (on-chip mem) instead of GMEM.. seems to be shared w/ other accelerators like video codecs
  - I suspect similar to xbox360 / Xenos
- 2d core z1xx
  - Origin: bitboys (I think)
  - OpenVG core... but focusing on what is needed for EXA
  - Not really any similarity to 3d core, different CP format, no GMEM, etc
  - Different adreno versions have zero, one, or two 2d cores

## Tools of the Trade...

- libwrap.so intercept ioctls, dump gpu buffers and cmdstream
- redump cmdstream parser / diff-tool for 2d
- cffdump cmdstream parser for 3d
  - Follows gpu ptrs (IB's, vertices, consts)
  - Shader disassembler
  - Some register bitfield and PM4 opc parsing
- pgmdump
  - Shader program binaries dumped via GL\_OES\_get\_program\_binary extension implemented in blob driver
  - Shader disassembler
  - Used in shader ISA r/e to compare output of similar shaders, to find instruction opcodes, etc
- fdre

- Simple GL-like API
- an easy way to exercise the GPU
- Shader assembler
- Depth/stencil/textures working
- Used before gallium driver, and now to have simple way to experiment and test theories

#### Tools of the Trade...

	peniio.exe.i	estor	e lesoure dep		~
File Edit Changes View Tabs Help					
🍄 🛃 Save 🦏 Undo 🎻 🚸 🐇					
restore-resolve-depdepth24stencil8.txt 🗶					
/home/robclark/src/blob/restore-resolve/restore-resolve-depth16.txt	Browse		/home/robo	clark/src/blob/restore-resolve/restore-resolve-depth24stencil8.txt 🛛 🗸 Browse	A
660ae234: c0012d00 00040209 00000000		-	+ 660at644:	: c0012d00 00040209 00000000	_
t3 opcode: SET CONSTANT (2d) (3 dwords)				anada, SET CONSTANT (2d) (2 duarda)	
SQ INTERPOLATOR CNTL: ffffffff (-1)				SQ INTERPOLATOR CNTL: ffffffff (-1)	
660ae240: c0012d00 00040182 fffffff		→	← 660af650:	: c0012d00 00040182 fffffff	
t3 opcode: SET_CONSTANT (2d) (3 dwords)			t3 c		
PA_SU_VTX_CNTL: 00000001 (1) 660ae24c: c0012d00 00040302 00000001 t3 opcode: DRAW_INDX (22) (4 dwords) prim type: RECTLIST (8)				PA_SU_VTX_CNTL: 00000001 (1)	
660a <mark>e24</mark> c: c0012d00 00040302 00000001		→	← 660af65c:	: c0012d00 00040302 00000001	
t3 opcode: DRAW_INDX (22) (4 dwords)				opcode: DRAW_INDX (22) (4 dwords)	
				prim_type: RECTLIST (8)	
source_select: DI_SRC_SEL_AUTO_INDEX (2) num_indices: 3 current register values				source_select: DI_SRC_SEL_AUTO_INDEX (2) num indices: 3	
current register values				current register values	
A220 VSC BIN SIZE: 224x384 (00000187)		→		A220 VSC BIN SIZE: 256x256 (00000108)	
A220 VSC PIPE DATA LENGTH 7: 00040000 (262144)			•	A220 VSC PIPE DATA LENGTH 7: 00040000 (262144)	_
SQ GPR MANAGEMENT: 00020600 (132608)				SQ GPR MANAGEMENT: 00020600 (132608)	
SQ_INST_STORE_MANAGMENT: 00000180 (384)				CO THET STORE MANAGEMENT, 00000100 (204)	=
TC_CNTL_STATUS: 00000001 (1)				TC_CNTL_STATUS: 00000001 (1)	
TPO CHICKEN: 00000002 (2)				TPO CHICKEN: 00000002 (2)	-
RB_SURFACE_INF0: 000000e0 (224)		→	←	RB_SURFACE_INFO: 00000100 (256)	
				RB_COLOR_INFO: 00000205 (COLORX_8_8_8_8)	
RB_DEPTH_INFO: 00054000 (format=DEPTHX_16, base=84) PA_SC_SCREEN_SCISSOR_BR: 128,384 (01800080)		→	÷	RB_DEPTH_INFO: 00040001 (format=DEPTHX 24 8, base=64)	
PA_SC_SCREEN_SCISSOR_BR: 128,384 (01800080)				PA_SC_SCREEN_SCISSOR_BR: 256,256 (01000100)	
PA_SC_WINDOW_OFFSET: -672, -384 (7e807d60)				PA_SC_WINDOW_OFFSET: -512,-512 (7e007e00)	
PA SC WINDOW OFFSET: -672, -384 (7e807d60) PA SC WINDOW SCISSOR TL: 0,0 (WINDOW OFFSET DISABLE) (80000000) PA SC WINDOW SCISSOR BR: 128,384 (01800080)	)			PA_SC_WINDOW_SCISSOR_TL: 0,0 (WINDOW_OFFSET_DISABLE) (80000000)	
PA SC WINDOW SCISSOR BR: 128,384 (01800080) A220 PC MAX VTX INDX: ffffffff (-1)		<b>→</b> _	- 🗲	PA SC WINDOW SCISSOR BR: 256,256 (01000100) A220 PC MAX VTX INDX: ffffffff (-1)	=
RB_COLOR_MASK: 0000ffff (65535)				RB COLOR MASK: 0000ffff (65535)	
RB BLEND ALPHA: 000000ff (255)				RB BLEND ALPHA: 000000ff (255)	
PA_CL_VPORT_XSCALE: 64.000000 (42800000)		<b>→</b>	←	RB STENCILREFMASK BF: ffff0000 (-65536)	
PA_CL_VPORT_X0FFSET: 64.000000 (42800000)		`		RB STENCILREFMASK: ffff0000 (-65536)	
PA_CL_VPORT_YSCALE: -192.000000 (c3400000)				PA_CL_VPORT_XSCALE: 128.000000 (43000000)	-
				PA_CL_VPORT_XOFFSET: 128.000000 (43000000)	
PA_CL_VPORT_ZSCALE: 0.499991 (3efffee0)				PA_CL_VPORT_YSCALE: -128.000000 (c3000000)	
				PA_CL_VPORT_YOFFSET: 128.000000 (43000000)	
SQ_PROGRAM_CNTL: 10030002 (vs-regs=3, ps-regs=1, vs-export=1 pc SQ_CONTEXT_MISC: 00000004 (4) SQ_INTERPOLATOR_CNTL: ffffffff (-1) BR_DEDTLCONTENT_00000004 (openly_zzfunc=CL_NEVERsfunc=CL_NE	osition-1-	1		PA_CL_VPORT_ZSCALE: 0.499991 (3efffee0)	
SQ_CONTEXT_MISC: 00000004 (4)				PA_CL_VPORT_ZOFFSET: 0.499991 (3efffee0)	
SQ_INTERPOLATOR_CNTL: ffffffff (-1)		,		SQ_PROGRAM_CNTL: 10030002 (vs-regs=3, ps-regs=1, vs-export=1 position-1-v	
RE_DEFINCONTROL: 00000008 (early-2, ziunc=0L_Never, siunc=0L_Ne	EVER, STAL			SQ_CONTEXT_MISC: 00000004 (4)	2
RB_BLEND_CONTROL: 00010001 (color-src-blend=RB_BLEND_ONE, color RB_COLORCONTROL: 00000027 (func=GL_ALWAYS, rop=12, dither-mode=				SQ_INTERPOLATOR_CNTL: fffffff (-1)	
RB_COLORCONTROL: 00000c27 (func=GL_ALWAYS, rop=12, dither-mode= PA_SU_SC_MODE_CNTL: 00080240 (front-ptype=triangles, back-ptype				RB_DEPTHCONTROL: 00000008 (early-z, zfunc=GL_NEVER, sfunc=GL_NEVER, sfail = RB_BLEND CONTROL: 00010001 (color-src-blend=RB_BLEND_ONE, color-func=DST	_
PA_CL_VTE_CNTL: 0000030f (783)	e=criangce.	-		RB COLORCONTROL: 000000c27 (func=GL ALWAYS, rop=12, dither-mode=DITHER DIS	
RB MODECONTROL: 00000004 (4)				PA SU SC MODE CNTL: 00080240 (front-ptype=triangles, back-ptype=triangles	
RB_SAMPLE_POS: 888888888 (-2004318072)				PA CL VTE CNTL: 0000030f (783)	
CLEAR COLOR: 0.309804 0.309804 0.309804 1.000000				RB_MODECONTROL: 00000004 (4)	=
PA SU POINT SIZE: 00080008 (524296)				RB <sup>-</sup> SAMPLE POS: 888888888 (-2004318072)	
PA_SU_POINT_MINMAX: 04000008 (67108872)				CLEAR_COLOR: 0.309804 0.309804 0.309804 1.000000	
PA_SU_LINE_CNTL: 00000008 (8)				PA_SU_POINT_SIZE: 00080008 (524296)	
PA_SU_VTX_CNTL: 00000001 (1)				PA_SU_POINT_MINMAX: 04000008 (67108872)	-
			-		
				Ln 142, Col 16 INS	

restore-resolve-depth16.txt : restore-resolve-depth24stencil8.txt - Meld

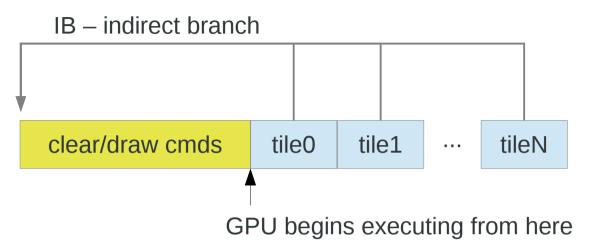
## 3d: Tiling

- Color buffer + Depth + Z must fit in GMEM
  - Side by side
  - 16bit Z or 24bit Z + 8bit stencil (optional)
- Rendering done in passes
  - GMEM is 512KiB on a220, 256KiB on a200
  - Without using hw binning/tiling:
    - Set scissor, IB to buffer w/ draw cmds
  - With hw binning (I think, not implemented yet):
    - Simple vertex shader pass to figure out which vertices in which bin (to avoid running VS many times)

## 3d: commandstream

- Command Parser
  - Same as r300/r600 PM4 type0/3
- Registers
  - Few similar registers (but different offset)
  - Mostly different
- Opcodes different
- "amd-gpu" kernel driver \o/
  - Recently found kernel driver from freescale kernel
  - Has pretty much all regs/bitfields as of a200
  - Opcode names/id's but not format

## 3d: commandstream



- Rendering within each tile works like traditional IMR
- The per-tile commands:
  - "restore" (optional) mem2gmem() transfer current contents from system memory to GMEM (tile buffer, color + depth/scissor)
  - Setup window-offset and screen scissor
  - IB to clear/draw cmds
  - "resolve" gmem2mem() transfer GMEM contents back to system memory
- Notes:
  - Not yet using "hw binning" looks like that should reduce vertex processing load for vertices not related to the current tile
  - The order of cmdstream building is not the same as order that GPU executes, and restore/resolve steps dirty some state used in clear/draw calls, so some care must be taken

#### 3d: ISA

- Unified shader ISA
- Separation of CF and ALU/FETCH
  - 48bit CF instructions in pairs
    - Control flow instructions reference offset of ALU instructions in 3\*dword (96bit)
  - 96bit ALU instructions
    - Co-dispatch of vec4+scalar

A	dreno	225 Sh	ader Ar	ray
ALU	ALU	ALU	ALU	ALU
ALU	ALU	ALU	ALU	ALU
ALU	ALU	ALU	ALU	ALU
ALU	ALU	ALU	ALU	ALU
ALU	ALU	ALU	ALU	ALU
ALU	ALU	ALU	ALU	ALU
ALU	ALU	ALU	ALU	ALU
ALU	ALU	ALU	ALU	ALU

Adreno 205 Shader Array						
ALU	ALU	ALU	ALU	ALU		
ALU	ALU	ALU	ALU	ALU		
ALU	ALU	ALU	ALU	ALU		
ALU	ALU	ALU	ALU	ALU		

#### 3d: ISA

```
uniform sampler2D g_NormalMap;
uniform float foo;
varying vec2 vTexCoord0;
void main()
{
  vec3 vNormal = vec3(2.0, 2.0, 0.0) * texture2D(g_NormalMap, vTexCoord0).xyz;
  vNormal.z = foo * -dot(vNormal, vNormal);
  gl_FragColor = vec4(vNormal, 1.0);
}
```

```
EXEC ADDR(0x2) CNT(0x3)
     FETCH:
               SAMPLE R0.xyz = R0.xyx CONST(0) LOCATION(CENTER)
   (S)ALU:
               MULV
                       R0.xvz = R0, C1.xxzw
                                                       EXEC
                                                                   ALLOC
     ALU:
               DOT3v
                       R1.x_{=} = R0, R0
ALLOC PARAM/PIXEL SIZE(0x0)
                                                       EXEC END NOP
EXEC_END ADDR(0x5) CNT(0x2)
                                                       FETCH
     ALU:
               MAXv export0.xy_w = R0, R0
               MAXs export0. w = R0
                                                       MULV
     ALU:
               MULv export0. z_ = -R1.xyxw, C0.xyxw
NOP
                                                       DOT<sub>3</sub>v
                                                       MAXV + MAXS
                                                       MULV
```

#### Status

- Hardware:
  - So far, just a220/z180
  - Snapdragon S3 (APQ8060, MSM8260, MSM8660)
    - eg. HP touchpad, dragonboard
  - a200/z160 looks like it should be pretty similar, not sure about others
  - nexux-4 with a320 on order, so we shall soon see :-)
- EXA/2d support:
  - Basics work, some bugs
  - Composite blits w/ mask surface not implemented yet
  - Enough registers understood, so just need time to implement
- Gallium/3d support:
  - Basics work, some bugs
    - >50% of glmark2, xbmc, compiz, q3a
  - Still needed
    - cmdstream: MSAA, mipmap textures
    - compiler: loops, optimizing
    - hw binning