

Freedreno Update

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 → 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...

The screenshot displays a diff tool window titled "restore-resolve-depth16.txt : restore-resolve-depth24stencil8.txt - Meld". The interface shows two side-by-side text editors with a central diff view. The left editor shows the original file "restore-resolve-depth16.txt" and the right editor shows the modified file "restore-resolve-depth24stencil8.txt". The diff highlights several changes in the shader code, including register values and control flags. Key differences include:

- RB SURFACE INFO:** Changed from 000000e0 (224) to 00000100 (256).
- RB COLOR INFO:** Changed from 00000205 (COLORX 8 8 8 8) to 00000205 (COLORX 8 8 8 8).
- RB DEPTH INFO:** Changed from 00054000 (format=DEPTHX 16, base=84) to 00040001 (format=DEPTHX 24 8, base=64).
- PA SC SCREEN SCISSOR BR:** Changed from 128,384 (01800080) to 256,256 (01000100).
- PA SC WINDOW OFFSET:** Changed from -672,-384 (7e807d60) to -512,-512 (7e007e00).
- PA SC WINDOW SCISSOR BR:** Changed from 128,384 (01800080) to 256,256 (01000100).
- PA CL VPORT XSCALE:** Changed from 64.000000 (42800000) to 128.000000 (43000000).
- PA CL VPORT XOFFSET:** Changed from 64.000000 (42800000) to 128.000000 (43000000).
- PA CL VPORT YSCALE:** Changed from -192.000000 (c3400000) to -128.000000 (c3000000).
- PA CL VPORT YOFFSET:** Changed from 192.000000 (43400000) to 128.000000 (43000000).
- PA CL VPORT ZSCALE:** Changed from 0.499991 (3efffee0) to 0.499991 (3efffee0).
- PA CL VPORT ZOFFSET:** Changed from 0.499991 (3efffee0) to 0.499991 (3efffee0).
- RB STENCILREFMASK BF:** Added, changed from ffff0000 (-65536) to ffff0000 (-65536).

The diff tool also shows a menu bar (File, Edit, Changes, View, Tabs, Help) and a toolbar with icons for Save, Undo, and other actions. The status bar at the bottom indicates "Ln 142, Col 16" and "INS".

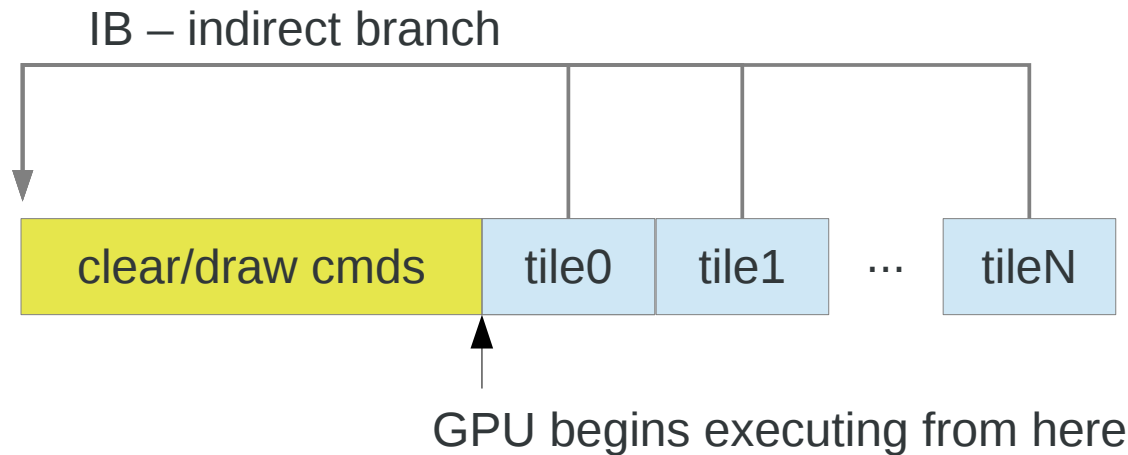
3d: Tiling

- 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



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.xyz_ = R0, C1.xxzw  
    ALU:      DOT3v   R1.x___ = R0, R0
```

ALLOC PARAM/PIXEL SIZE(0x0)

EXEC_END ADDR(0x5) CNT(0x2)

```
    ALU:      MAXv    export0.xy_w = R0, R0  
              MAXs    export0.___w = R0  
    ALU:      MULv    export0.__z_ = -R1.xyxw, C0.xyxw
```

NOP

EXEC	ALLOC
EXEC_END	NOP
FETCH	
MULv	
DOT3v	
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