

ECE 571 – Advanced Microprocessor-Based Design Lecture 18

Vince Weaver

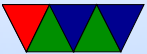
<http://www.eece.maine.edu/~vweaver>

vincent.weaver@maine.edu

11 November 2014

Project/HW Reminder

- Homework #4 comments



Stuff from Last Class

- Phase change RAM.
 - chalcogenide glass – used in CD-Rs
 - 100ns (compared to 2ns of DRAM) latency
 - heating element change from amorphous (high resistance, 0) to crystalline (low resistance, 1)
 - temp sensitive, values lost when soldering to board (unlike flash)
 - better than flash (takes .1ms to write, write whole blocks at once)



- Newer methods might involve lasers and no phase change?
- Mapping into memory? No need to copy from disk?
- But also, unlike DRAM, a limit on how many times can be written.

- Memristors

- resistors, relationship between voltage and current
- capacitors, relationship between voltage and charge
- inductors, relationship between current and magnetic flux



- memristor, relationship between charge and magnetic flux; “remembers” the current that last flowed through it
- Lot of debate about whether possible. HP working on memristor based NVRAM



Graphics and Video Cards

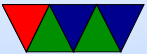


Old CRT Days

- Electron gun
- Horizontal Blank, Vertical Blank
- Atari 2600 – only enough RAM to do one scanline at a time
- Apple II – video on alternate cycles, refresh RAM for free
- Bandwidth key issue. SNES / NES, tiles. Double



buffering vs only updating during refresh



Old 2D Video Cards

- Framebuffer (possibly multi-plane), Palette
- Dual-ported RAM, RAMDAC (Digital-Analog Converter)
- Interface (on PC) various io ports and a 64kB RAM window
- Mode 13h
- Acceleration – often commands for drawing lines, rectangles, blitting sprites, mouse cursors, video overlay



Modern Graphics Cards

- Can draw a lot of power
- 2D (optional these days)
- 3D
- Video decoders



Interface

- Integrated or stand alone
- Integrated traditionally less capable, but changing. Share Memory bandwidth, take memory.



GPUs

- Display memory often broken up into tiles (improves cache locality)
- Massively parallel matrix-processing CPUs that write to the frame buffer (or can be used for calculation)
- Texture control, 3d state, vectors
- Front-buffer (written out), Back Buffer (being rendered)
Z-buffer (depth)



- Originally just did lighting and triangle calculations. Now shader languages and fully generic processing



Video RAM

- VRAM – dual ported. Could read out full 1024Bit line and latch for drawing, previously most would be discarded (cache line read)
- GDDR3/4/5 – traditional one-port RAM. More overhead, but things are fast enough these days it is worth it.
- Confusing naming, GDDR3 is equivalent of DDR2 but with some speed optimization and lower voltage (so



higher frequency)



Busses

- DDC – i2c bus connection to monitor, giving screen size, timing info, etc.
- PCIe (PCI-Express) – most common bus used in x86 systems

Original PCI and PCI-X was 32/64-bit parallel bus

PCIe is a serial bus, sends packets

Can power 25W, additional power connectors to supply can have 75W, 150W and more

Can transfer 8GT/s (giga-transfers) a second



In general PCIe is the main limiting factor to getting data to GPU.



Connectors

CRTC (CRT Controller) Can point to same part of memory (mirror) or different.

- RCA – composite/analog TV
- VGA – 15 pin, analog
- DVI – digital and/or analog. DVI-D, DVI-I, DVI-A
- HDMI – compatible with DVI (though content restrictions). Also audio. HDMI 1.0 – 165MHz, 1080p



or 1920x1200 at 60Hz. TMDS differential signalling. Packets. Audio sent during blanking.

- Display Port – similar but not the same as HDMI
- Thunderbolt – combines PCIe and DisplayPort. Intel/Apple. Originally optical, but also Copper. Can send 10W of power.
- LVDS – Low Voltage Differential Signaling – used to connect laptop LCD



LCD Displays

- Crystals twist in presence of electric field
- Asymmetric on/off times
- Passive (crossing wires) vs Active (Transistor at each pixel)
- Passive have to be refreshed constantly
- Use only 10% of power of equivalent CRT

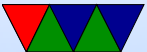


- Circuitry inside to scale image and other post-processing
- Need to be refreshed periodically to keep their image
- New “bistable” display under development, requires not power to hold state



Interfaces

- OpenGL – SGI
- DirectX – Microsoft
- For consumer grade, driven by gaming



GPGPUS

- Interfaces needed, as GPU companies do not like to reveal what their chips do at the assembly level.
 - CUDA (Nvidia)
 - OpenCL (Everyone else) – can in theory take parallel code and map to CPU, GPU, FPGA, DSP, etc



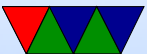
Why GPUs?

- Old example:
 - 3GHz Pentium 4, 6 GFLOPS, 6GB/sec peak
 - GeForceFX 6800: 53GFLOPS, 34GB/sec peak
- Newer example
 - Raspberry Pi, 700MHz, 0.177 GFLOPS
 - On-board GPU: Video Core IV: 24 GFLOPS



Key Idea

- using many slimmed down cores
- have single instruction stream operate across many cores (SIMD)
- avoid latency (slow textures, etc) by working on another group when one stalls



GPU Benefits

- Specialized hardware, concentrating on arithmetic. Transistors for ALUs not cache.
- Fast 32-bit floating point (16-bit?)
- Driven by commodity gaming, so much faster than would be if only HPC people using them.
- Accuracy? 64-bit floating point? 32-bit floating point? 16-bit floating point? Doesn't matter as much if color slightly off for a frame in your video game.



- highly parallel



GPU Problems

- optimized for 3d-graphics, not always ideal for other things
- Need to port code, usually can't just recompile cpu code.
- Companies secretive.
- serial code
- a lot of control flow



- lot of off-chip memory transfers



GPU Performance

- Like stream processors, need parallel. Only can operate on independent things, but can do many many at once. Stream processors are records that all need similar operations done to them. Kernels are the code applied in each processor. Vertices and fragments have shaders run on them.



Traditional GPU Setup

- CPU send list of vertices to GPU.
- Transform (vertex processor) (convert from world space to image space). 3d translation to 2d, calculate lighting. Operate on 4-wide vectors (x, y, z, w in projected space, r, g, b, a color space)
- Rasterizer – transform vertexes/vectors into a grid. Fragments. break up to pixels and anti-alias
- Shade (Fragment processor) compute color for each



pixel. Use textures if necessary (texture memory, mostly read)

- Write out to framebuffer (mostly write)

