ECE 571 – Advanced Microprocessor-Based Design Lecture 34

Vince Weaver

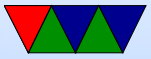
http://web.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

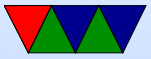
30 November 2020

Project/HW Reminder

- Homework #11 will be posted, GPU reading
- Responded to project statuses

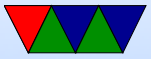


Graphics and Video Cards



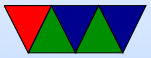
Old CRT Days

- Electron gun
- Horizontal Blank, Vertical Blank

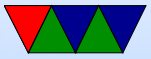


LCD Displays (sic)

- 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

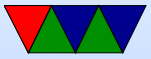


- Circuitry inside to scale image and other post-processing
- Need to be refreshed periodically to keep their image
- New “bistable” display under development, requires no power to hold state



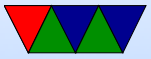
Coding for CRTs

- 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



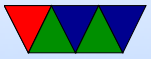
Old 2D Video Cards

- 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



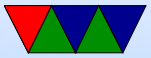
Modern Graphics Cards

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



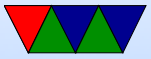
Interface

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



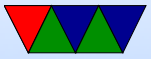
Video RAM

- 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

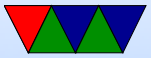
- DDC – i2c bus connection to monitor, giving screen size, timing info, etc.
- PCIe (PCI-Express) – most common bus in x86 systems
  Original PCI and PCI-X was 32/64-bit parallel bus
  PCIe is a serial bus, sends packets
  Can power 25W, additional power connectors to supply can have 75W, 150W and more
  Can transfer 8GT/s (giga-transfers) a second
  In general PCIe is limiting factor to getting data to GPU.



Connectors

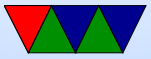
CRTC (CRT Controller) Can point to same part of memory (mirror) or different.

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



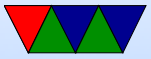
or 1920x1200 at 60Hz. TMDS differential signaling. Packets. Audio sent during blanking.

- 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



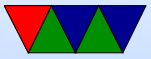
Interfaces

- OpenGL – SGI (Khronos)
- DirectX – Microsoft (Direct3d)
- Vulkan (sort of next gen OpenGL. Lower level, closer to hardware)
- Metal – from Apple



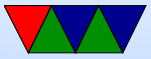
GPUs

- 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



GPGPUs

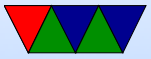
- Interfaces needed, as GPU companies do not like to reveal what their chips due at the assembly level.
  - CUDA (Nvidia)
  - OpenCL (Everyone else) – can in theory take parallel code and map to CPU, GPU, FPGA, DSP, etc
  - OpenACC?



Other Accelerator Options

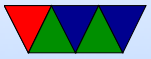
- XeonPhi – came out of the larabee design (effort to do a GPU powered by x86 chips). Large array of x86 chips(p5 class on older models, atom on newer) on PCIe card. Sort of like a plug-in mini cluster. Runs Linux, can ssh into the boards over PCIe. Benefit: can use existing x86 programming tools and knowledge.

- FPGA – can have FPGA accelerator. Only worthwhile if you don't plan to reprogram it much as time delay in reprogramming. Also requires special compiler support



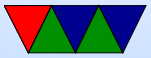
(OpenMP?)

- ASIC – can have hard-coded custom hardware for acceleration. Expensive. Found in BitCoin mining?
- DSPs – can be used as accelerators



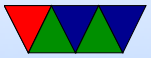
Why GPUs?

- 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



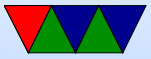
Key Idea

- 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



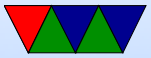
Latency vs Throughput

- CPUs = Low latency, low throughput
- GPUs = high latency, high throughput
- CPUs optimized to try to get lowest latency (caches); with no parallelism have to get memory back as soon as possible
- GPUs optimized for throughput. Best throughput for all better than low-latency for one



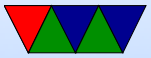
GPU Benefits

- 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



Older / Traditional GPU Pipeline

- In old days, fixed pipeline (lots of triangles).
- Modern chips much more flexible, but the old pipeline can still be implemented in software via the fancier interface.