History and Evolution of GPU Architecture

A Paper Survey

Chris McClanahan

Georgia Tech

College of Computing

chris.mcclanahan@gatech.edu

ABSTRACT

The graphics processing unit (GPU) is a specialized and highly parallel microprocessor designed to offload and accelerate 2D or 3D rendering from the central processing unit (CPU). GPUs can be found in a wide range of systems, from desktops and laptops to mobile phones and super computers [3].

This paper provides a summary of the history and evolution of GPU hardware architecture. The information in this paper, while being slightly NVIDIA biased, is presented as a set of milestones noting major architectural shifts and trends in GPU hardware.

The evolution of GPU hardware architecture has gone from a specific single core, fixed function hardware pipeline implementation made solely for graphics, to a set of highly parallel and programmable cores for more general purpose computation. The trend in GPU technology has no doubt been to keep adding more programmability and parallelism to a GPU core architecture that is ever evolving towards a general purpose more CPU-like core.

Future GPU generations will look more and more like wide-vector general purpose CPUs, and eventually both will be seamlessly combined as one.

Keywords

Hardware, Graphics, Graphics Pipeline, Multi-core

1. INTRODUCTION

A graphics processing unit (GPU) is a dedicated parallel processor optimized for accelerating graphical computations. The GPU is designed specifically to perform the many floating-point calculations essential to 3D graphics rendering. Modern GPU processors are massively parallel, and are fully programmable. The parallel floating point computing power found in a modern GPU is orders of magnitude higher than a CPU [2].

GPUs can be found in a wide range of systems, from desktops and laptops to mobile phones and super computers. With their parallel structure, GPUs implement a number of 2D and 3D graphics primitives processing in hardware, making them much faster than a general purpose CPU at these operations [3].

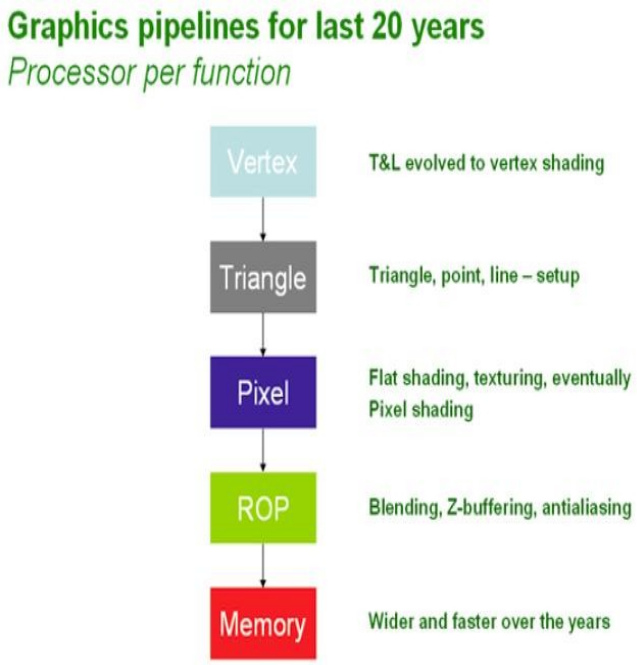

Illustration 1: The traditional fixed-function graphics pipeline

The original GPUs were modeled after the concept of a graphics pipeline. The graphics pipeline is a conceptual model of stages that graphics data is sent through, and is usually implemented via a combination of hardware (GPU cores) and CPU software (OpenGL, DirectX). The graphics pipeline design approach is fairly consistent among the major GPU manufacturers like NVIDIA, ATI, etc., and helped accelerate GPU technology adoption [1].

The pipeline simply transforms coordinates from 3D space (specified by programmer) into 2D pixel space on the screen. It is basically an "assembly line" of various stages of operations to apply to triangles/pixels, and can be generalized out to 2 stages: Geometry and Rendering.

Early GPUs implemented only the rendering stage in hardware, requiring the CPU to generate triangles for the GPU to operate on. As GPU technology progressed, more and more stages of the pipeline were implemented in hardware on the GPU, freeing up more CPU cycles [1].

2. 1980's

Leading up to the early 1980's, the "GPUs" of the time were really just integrated frame buffers. They were boards of TTL logic chips that relied on the CPU, and could only draw wire-frame shapes to raster displays [4]. The term “GPU” would not be introduced until 1999 by NVIDIA, but for consistency, it will be used throughout this paper [8].

The IBM Professional Graphics Controller (PGA) was one of the very first 2D/3D video cards for the PC [2]. The PGA used an on-board Intel 8088 microprocessor to take over processing all video related tasks, freeing up the CPU for video processing (such as drawing and coloring filled polygons). Though it was released in 1984, 10 years before hardware 2D/3D acceleration was standardized, the ~$5500 price tag and lack of compatibility with many programs and non-IBM systems at the time, made it unable to achieve mass-market success. The PGA's separate on-board processor marked an important step in GPU evolution to further the paradigm of using a separate processor for graphics computations [1].

By 1987, more features were being added to early GPUs, such as Shaded Solids, Vertex lighting, Rasterization of filled polygons, and Pixel depth buffer, and color blending. There was still much reliance on sharing computation with the CPU [4].

In the late 1980's, Silicon Graphics Inc. (SGI) emerged as a high performance computer graphics hardware and software company. With the introduction of OpenGL in 1989, SGI created and released the graphics industry's most widely used and supported, platform independent, 2D/3D application programming interface (API). OpenGL support has also become an intricate part of the design of modern graphics hardware. SGI also pioneered the concept of the graphics pipeline early on [1].

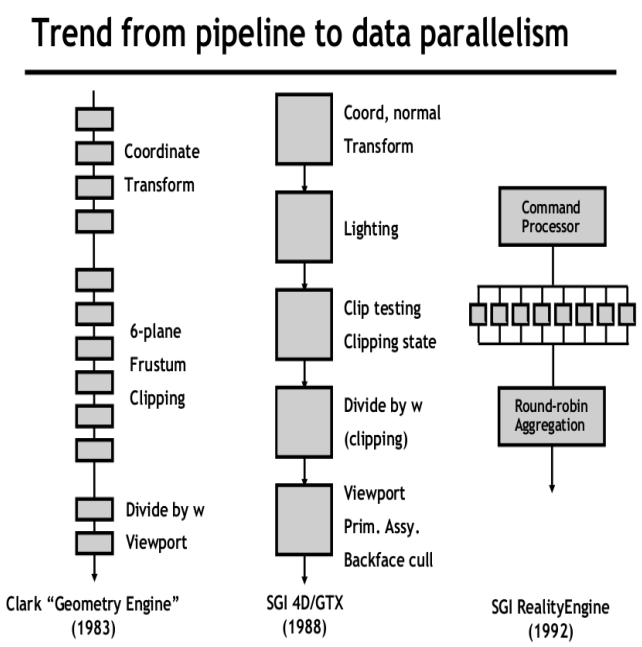

Illustration 2: Early trends in pipeline parallelism

3. 1990's

3.1 Generation 0

GPU hardware and the graphics pipeline started to really take shape in 1993, when SGI released its RealityEngine board for graphics processing [5]. There were distinct boards and logic chips for various later stages of the graphics pipeline, but still relied on the CPU for the first half. Data had a fixed flow through each stage.

By the mid 1990's SGI cards were mainly found on workstations, while 3D graphics hardware makers 3DFX (Voodoo), NVIDIA (TNT), ATI (Rage), and Matrox started providing consumer 3D graphics boards. A combination of "cheap" hardware with games such as Quake and Doom really drove the gaming industry and GPU adoption [6].

Even with deep pipelines though, early GPUs still only output one pixel per clock cycle, meaning CPUs could still send more triangles to the GPU than it could handle. This lead to adding more pipelines in parallel to the GPU (and ultimately more cores), so multiple pixels could be processed in parallel each clock cycle [1].

3.2 Generation I

The 3dfx Voodoo (1996) was considered one of the first true 3D game cards [7]. It only offered 3D acceleration, so you still needed a 2D accelerator. It operated over the PCI bus, had 1 million transistors, 4 MB of 64-bit DRAM, and the core clock operated at 50 MHz. The CPU still did vertex transformations, while the Voodoo provided texture mapping, z-buffering, and rasterization.

3.3 Generation II

In 1999, the first cards to implement the entire pipeline (now with transform and lighting calculations) in GPU hardware were released. With the introduction of NVIDIA's GeForce256 and ATI's Radeon 7500, the first true GPUs were available at the consumer level [7].

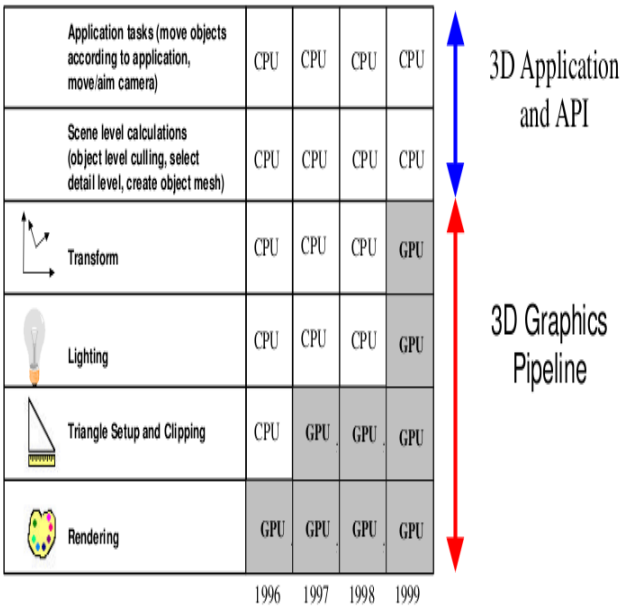

Illustration 3: Migration of functionality to GPU hardware

Up until 1999, the term "GPU" didn't actually exist, and NVIDIA coined the term during it's launch of the GeForce 256 [8]. The GeForce 256 had 23 million transistors, 32 MB of 128-bit DRAM, he core clock operated at 120MHz, and had four 64-bit pipelines for rendering [9].

This generation of cards were the first to use the new Accelerated Graphics Port (AGP) instead of the PCI bus, and offered new graphics features in hardware, such as multi-texturing, bump maps, light maps, and hardware geometry transform and lighting [7, 8].

The first pipelines now in hardware were known as a "fixed function" pipeline, because once the programmer sent graphics data into the GPU's pipeline, the data could not be modified [1].

With these cards, the GPU hardware and computer gaming market really started to take off [1]. While much faster, the main problem with the fixed function pipeline model was the inflexibility of graphical effects. Since the feature sets defined by OpenGL and DirectX APIs were implemented in hardware, as newer features were added to graphics APIs, the fixed function hardware could not take advantage of the new standards [1].

4. 2000's

4.1 Generation III

The next step in the evolution of GPU hardware was the introduction of the programmable pipeline on the GPU. In 2001, NVIDIA released the GeForce 3 which gave programmers the ability to program parts of the previously non-programmable pipeline [1]. Instead of sending all the graphics description data to the GPU and have it simply flow through the fixed pipeline, the programmer can now send this data along with vertex "programs" (called shaders) that operate on the data while in the pipeline [1].

These shader programs were small "kernels", written in assembly-like shader languages. For the first time, there was limited amount of programmability in the vertex processing stage of the pipeline. Other popular cards at this time were ATI Radeon 8500, and Microsoft's X-Box [10].

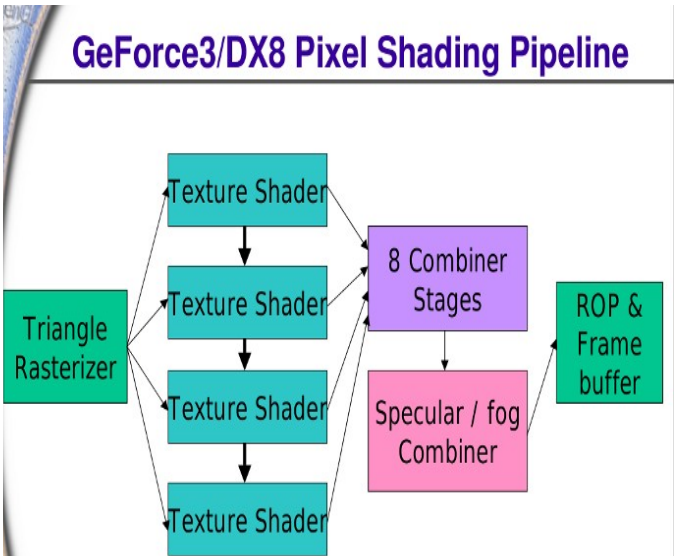

Illustration 4: GeForce 3 Architecture

The GeForce 3 had 57 million transistors, 64 MB of 128-bit DDR DRAM, and the core clock operated at 120MHz [9].

4.2 Generation IV

One year later, in 2002, the first fully programmable graphics cards hit the market: NVIDIA GeForce FX, ATI Radeon 9700. These cards allowed for per-pixel operations with programmable vertex and pixel(fragment) shaders, and allowed for limited user-specified mapping of input-output data operations [7, 10]. Separate, dedicated hardware were allocated for pixel shader and vertex shader processing.

The first GeForce FX had 80 million transistors, 128 MB of 128-bit DDR DRAM, and the core clock operated at 400MHz. [9].

In 2003, the first wave of GPU computing started to come about with the introduction of DirectX 9, by taking advantage of the programmability now in the GPU hardware, but for non graphics [8]. Full floating point support and advanced texture processing started to appear in cards.

4.3 Generation IV

By this time, the rate of GPU hardware technology was accelerating at a rate much faster than Moore's Law [7]. In 2004, the GeForce 6 and Radeon X800 were released, and were some of the first cards to use the PCI-express bus. For software, early high level GPU languages such as Brook and Sh started to appear, that offered true conditionals and loops and dynamic flow control in shader programs. On the hardware side, higher precision (64-bit double support), multiple rendering buffers, increased GPU memory and texture accesses were being introduced [7, 10].

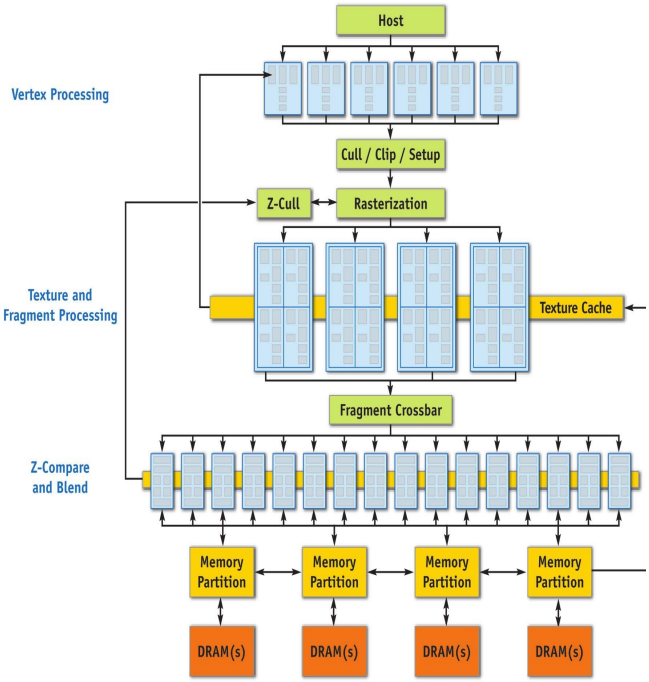

Illustration 5: GeForce 6 Architecture

The first GeForce 6 had 146 million transistors, 256 MB of 256-bit GDDR3 DRAM, and the core clock operated at 500MHz [9].

When viewed as a graphics pipeline, the a GPU contains a programmable vertex engine, a programmable fragment engine, a texture engine, and a depth-compare/blending-data write engine. When viewed alternatively as a processor "for non-graphics applications, a GPU can be seen as a large amount of programmable floating-point horsepower and memory bandwidth that can be exploited for compute-intensive applications completely unrelated to computer graphics" [11].

A trend towards GPU programmability was starting to form.

4.4 Generation VI

The introduction of NVIDIA's GeForce 8 series GPU in 2006 marked the next step in GPU evolution: exposing the GPU as massively parallel processors [4].

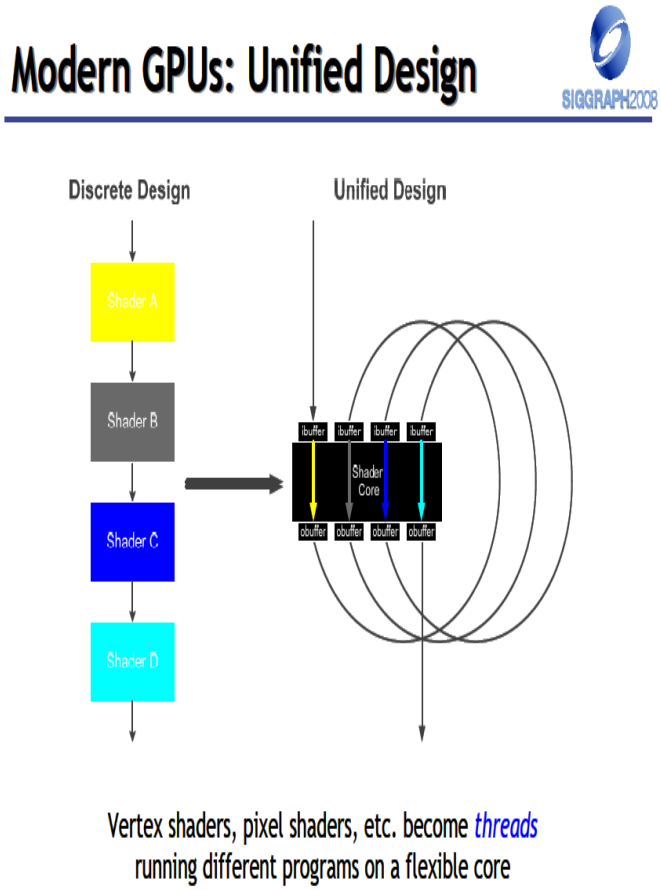

Illustration 6: Unified hardware shader design

The G80 (GeForce 8800) architecture was the first to have "unified", programmable shaders - in other words, a fully programmable unified processor called a Streaming Multiprocessor, or SM, that handled vertex, pixel, and geometry computation. Also introduced was a new geometry shader, adding more programmability when combined with the vertex shader and pixel shaders [10]. With a now unified shader hardware design, the traditional graphics pipeline model is now purely a software abstraction.

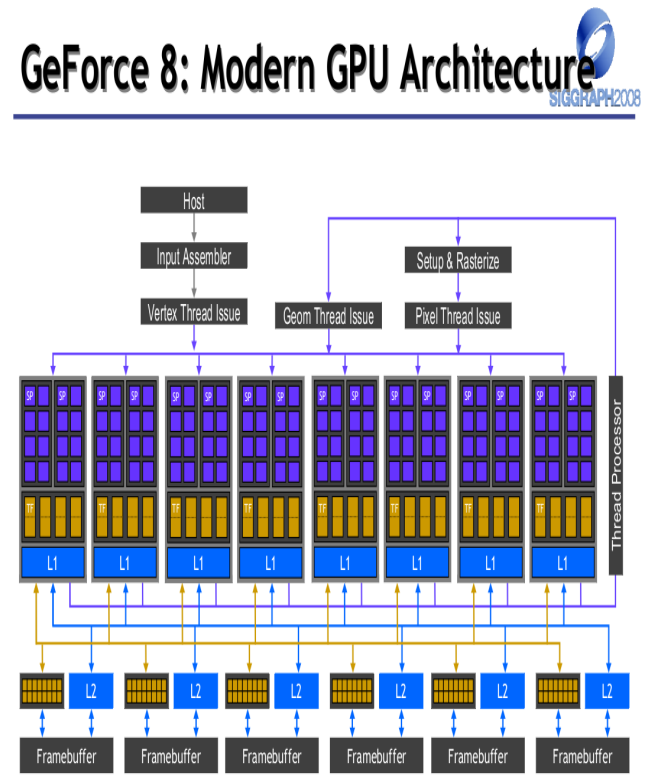

Illustration 7: GeForce 8 Architecture

The first GeForce 8 had 681 million transistors, 768 MB of 384-bit GDDR3 DRAM, and the core clock operated at 600MHz [9].

To harness all this "general purpose" GPU power was the new programming language CUDA by NVIDIA for NVIDIA only. Not much later, ATI Stream for ATI cards and DirectX 10 for either card (Microsoft Windows only though) were introduced [12].

4.5 Generation VII

The trend towards more CPU-like, programmable GPU cores continues with the introduction of NVIDIA's Fermi architecture. Announced in late 2009, but released in early 2010, the Fermi GPU was the first GPU designed for GPGPU computing, bringing features such as: true HW cache hierarchy , ECC , unified memory address space , concurrent kernel execution, better double precision performance, and dual warp schedulers [13]. The GTX480 Fermi had a total of 480 CUDA cores at launch (15 streaming multiprocessors * 32 CUDA cores each) [13].

Illustration 8: Fermi Architecture

The first Fermi cards had 3 billion transistors, 1.5 GB of 384-bit GDDR5 DRAM, and the core clock operated at 700MHz [13].

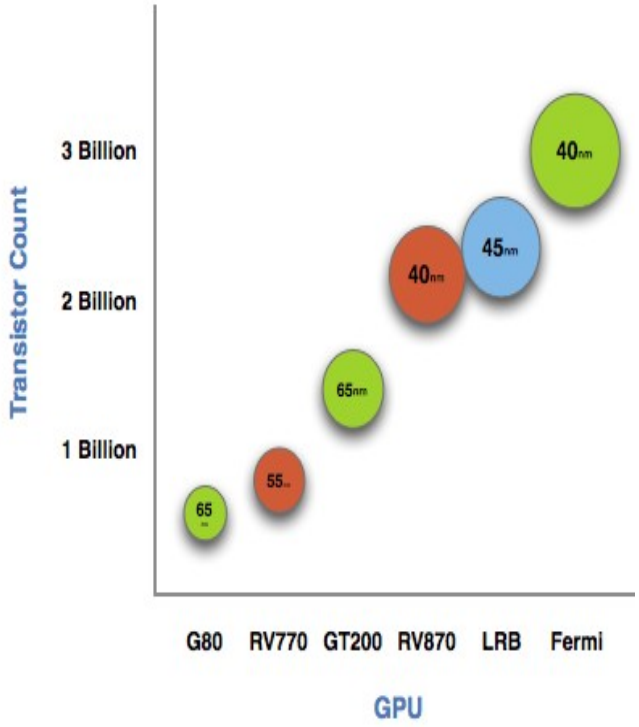

Illustration 9: Transistor count trends for GPU cores

5. 2010 and Beyond

Recently, in late 2010, NVIDIA refreshed their Fermi-based gaming card, the GTX580, adding one more SM (bringing the total CUDA core count to 512) and offering a slightly higher memory bandwidth.

The GPU hardware evolution thus far has gone from an extremely specific, single core, fixed function hardware pipeline implementation just for graphics rendering, to a set of highly parallel and programmable cores for more general purpose computation. Now, the architecture of many-core GPUs are starting to look more and more like multi-core, general purpose CPUs [14]. In that respect, Fermi can essentially be thought of as a 16-core CPU with 32-way hyper-threading per core, with a wide vector width.

AMD recently announced their Fusion line of CPU+GPUs on the same die (dubbed APU, accelerated processing unit), to be released early 2011 [14]. AMD's APUs are designed so that a standard x86 processor for scalar workloads and a DX11 GPU for vector workloads are brought together on the same die.

Intel's Larrabee processor brings many smaller x86 cores on a single die, augmented by a wide vector unit [15]. Intel's SandyBridge CPU also has an integrated GPU which both cores share the L3 cache [16].

Illustration 10: AMD Fusion Architecture

If merging CPU+GPU is the trend, then what sort of fixed-function hardware on the GPU (if any) should remain? Should there even be a distinction to the programmer between CPU vs GPU? Will future programming languages abstract the GPU into just another set of multi-core processors for applications?

"All processors aspire to be general-purpose" -Tim Van Hook, Keynote, Graphics Hardware 2001 [1].

6. CONCLUSION

The evolution of GPU hardware architecture has gone from a specific single core, fixed function hardware pipeline implementation made solely for graphics, to a set of highly parallel and programmable cores for more general purpose computation. The trend in GPU technology has no doubt been to keep adding more programmability and parallelism to a GPU core architecture that is ever evolving towards a general purpose more CPU-like core.

Future GPU generations will look more and more like wide-vector general purpose CPUs, and eventually both will be seamlessly combined as one.

7. REFERENCES

[1] Crow, Thomas. Evolution of the Graphical Processing Unit. 2004 Paper. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.142.368&rep=rep1&type=pdf

[2] Wikipedia: Video Card. Retrieved November 2010. http://en.wikipedia.org/wiki/Video_card

[3] Wikipedia: Graphics Processing Unit. Retrieved November 2010. http://en.wikipedia.org/wiki/Graphics_processing_unit

[4] Buck, Ian. The Evolution of GPUs for General Purpose Computing. GTC 2010. https://docs.google.com/viewer?url=http://www.nvidia.com/content/GTC-2010/pdfs/2275_GTC2010.pdf

[5] Luebke, David. GPU Architecture: Implications & Trends. SIGGRAPH 2008. http://s08.idav.ucdavis.edu/luebke-nvidia-gpu-architecture.pdf

[6] Franken, Yannick. Introduction to Graphics Hardware and GPUs. 2008 Lecture. https://docs.google.com/viewer?url=http://research.edm.uhasselt.be/~yfrancken/aac/aac.ppt

[7] Venkat, Suresh. Understanding the graphics pipeline. 2005 Lecture. www.cis.upenn.edu/~suvenkat/700/lectures/2/Lecture2.ppt

[8] Göddek, Dominik. GPU Computing with CUDA - Part 1: GPU Computing Evolution. 2009 Lecture. http://www.mathematik.uni-dortmund.de/~goeddeke/gpgpu/cuda-2009/1-History.pdf

[9] From Voodoo to GeForce: The Awesome History of 3D Graphics. Retrieved November 2010. http://www.maximumpc.com/print/6338

[10] Hyesoon, Kim. Design Game Consoles 09: lec_graphics.pdf. 2009 Lecture. https://docs.google.com/viewer?url=http://www.cc.gatech.edu/~hyesoon/spr09/lec_graphics.pdf

[11] GPU Gems 2. Chapters 29-30. 2005 Book. http://http.download.nvidia.com/developer/GPU_Gems_2/

[12] Houston, Mike. GPU Architecture. 2010 Lecture. https://graphics.stanford.edu/wikis/cs448s-10/FrontPage?action=AttachFile&do=get&target=CS448s-10-03.pdf

[13] Fermi Whitepaper. 2010 NVIDIA. http://www.nvidia.com/content/PDF/fermi_white_papers/NVIDIA_Fermi_Compute_Architecture_Whitepaper.pdf

[14] AMD Fusion whitepaper. 2010 AMD. http://sites.amd.com/us/Documents/48423B_fusion_whitepaper_WEB.pdf

[15] Seiler, Larry. Larrabee: a many-core x86 architecture for visual computing. http://portal.acm.org.www.library.gatech.edu:2048/ft_gateway.cfm?id=1360617&type=pdf&coll=portal&dl=ACM&CFID=17193894&CFTOKEN=81145590

[16] Wikipedia: Sandy Bridge. Retrieved December 2010. http://en.wikipedia.org/wiki/Sandy_Bridge_(microarchitecture)