“GPU Processing” Proc. IEEE 96(5), 2008

Digital Signal Processor, AAU Klagenfurt

MD. SARWAR ZAHAN & BASHIRU OTOKITI

Introduction:

A graphics processing unit (GPU) is a computer chip that performs rapid mathematical calculations, primarily for the purpose of rendering images. In the early days of computing, the central processing unit (CPU) performed these calculations. As more graphics-intensive applications developed demands put strain on the CPU and degraded performance. The GPU came about as a way to offload those tasks from the CPU, freeing up its processing power.

History of GPU:

1970s:
Arcade system boards have been using specialized graphics chips since the 1970s. Early video games hardware frame buffers were too expensive, so video chips composited data together as the display.
1980s:
The NEC µPD7220 was one of the first implementations of a graphics display controller as a single Large Scale Integration (LSI) integrated circuit chip, enabling the design of low-cost, high-performance video graphics cards such as those from Number Nine Visual Technology.
1990s:
In 1991, S3 Graphics introduced the S3 86C911, which its designers named after the Porsche 911 as an implication of the performance increase it promised.
2000 to 2010:
NVidia was first to produce a chip capable of programmable shading, the GeForce 3 (code named NV20).
2010 to present:
In 2010, NVidia began a partnership with Audi to power their cars' dashboards. These Tegra GPUs were powering the cars' dashboard, offering increased functionality to cars' navigation and entertainment systems

3D GRAPHICS RENDERING PIPELINE:

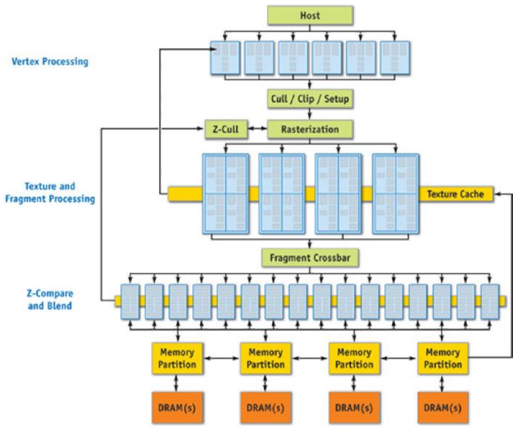
The 3D graphics rendering pipeline consists of the following main stages:

1. Vertex Processing: Process and transform individual vertices.
2. Rasterization: Convert each primitive (connected vertices) into a set of fragments. A fragment can be treated as a pixel in 3D spaces, which is aligned with the pixel grid, with attributes such as position, color, normal and texture.

3. Fragment Processing: Process individual fragments.
4. Output Merging: Combine the fragments of all primitives (in 3D space) into 2D color-pixel for the display.

In modern GPUs, the vertex processing stage and fragment processing stage are programmable.

GPU Architecture:



With six parallel vertex processors GPU receive data from the host and perform operations such as transformation and lighting .Than, the output goes into the triangle setup stage which takes care of primitive assembly, culling(cull) and clipping, and then into the rasterizer which produces the fragments. After triangle setup fragments move on to the sixteen fragment processors which operate in 4 parallel units and computes the output colors of each fragment. The fragment crossbar is a linking element that is basically responsible for directing output pixels to any available pixel engine. The 16 pixel engines are the final stage of processing, and perform operations such as alpha blending, depth tests, etc., before delivering the final pixel to the frame buffer.

GPU Programing:

CUDA:

CUDA (Compute Unified Device Architecture) is a parallel computing platform and programming model created by NVIDIA and implemented by the graphics processing units (GPUs) that they produce. CUDA has two programing environment interface:

1. Runtime: High level language (like c/C++)
2. Driver : Low level (comes with GPU driver)

Within C programs, call SIMT “kernel” routines that are executed on GPU. CUDA syntax extension to C identifies routine as a Kernel. Very easy to learn although to get highest possible

Open CL:

Is a framework for writing programs that execute across heterogeneous platforms consisting of central processing units (CPUs), graphics processing units (GPUs), digital signal processors (DSPs).
OpenCL provides a standard interface for parallel computing using task-based and data-based parallelism.

HLSL:

HLSL is the High Level Shading Language for DirectX. It help to write programmable shaders for the Direct3D pipeline.

Cg:

It's a high-level shading language developed by NVidia in close collaboration with Microsoft for programming vertex and pixel shaders.

GPU COMPUTING:

GPU computing is the use of a GPU to do general purpose scientific and engineering computing CPU and GPU together in a heterogeneous computing model. Sequential part of the application runs on the CPU and the computationally-intensive part runs on the GPU. The programmable units of the GPU follow a single program multiple-data (SPMD) programming model tasks are split up and run simultaneously on multiple processors with different input in order to obtain results faster. SPMD is the most common style of parallel programming.

GPU Application Area:

- Scientific computing
- Weather forecasting
- Climate research
- Video processing
- Hardware accelerated video decoding and post-processing
- Audio signal processing
- Audio and sound effects processing, to use a GPU for digital signal processing (DSP)
- Analog signal processing
- Speech processing

GPU Problems:

The basic difficulty in having high performance GPU is that it has ton of cores, and it tries them to all be utilized to their full potency as much as possible even in small task. Early GPU programming has dissipated with the new capabilities of these programming systems, though support for debugging and profiling on the hardware is still primitive. Programmers want a small branch granularity so each thread can be independent; architects want a large branch granularity to build more efficient.

Future Computing:

Both AMD and NVIDIA announced future support for double-precision floating-point hardware.

The more double-precision support removes one of the major obstacles for the adoption of the GPU in many scientific computing applications. Another upcoming trend is a higher bandwidth path between CPU and GPU. The PCI Express bus between CPU and GPU is a bottleneck in many applications, so future support for PCI Express 2, HyperTransport, or other high-bandwidth connections is a welcome trend.

Conclusion:

Today GPU is very, very parallel and is able to deal with more and more complex types of parallel tasks. In near future all apps will simply run incredibly fast by using high performance GPU. CPU may be a quad-core, but now days some graphics cards have over 200 cores. GPU can handle now non-graphical tasks and results are amazing. An algorithm that lends itself well to parallelism is much faster on GPU than it could ever be on a CPU.

References:

[1] J. D. Owens, M. Houston, D. Luebke, S. Green, J. E. Stone, and J. C. Phillips, "GPU computing," Proceedings of the IEEE, vol. 96, no. 5, pp. 879–899, May 2008.

[2] http://www.nvidia.com/object/what-is-gpu-computing.html

[3] https://en.wikipedia.org/wiki/Graphics_processing_unit

[4] https://www.ntu.edu.sg/home/ehchua/programming/opengl/CG_BasicsTheory.html

[5] https://summerofhpc.prace-ri.eu/if-not-task-farming-then-what/