GAZI UNIVERSITY

ENGINEERING FACULTY

COMPUTER ENGINEERING

CENG479 PARALLEL COMPUTER ARCHITECTURES AND PROGRAMMING

Gamze Aksu

171180005

CUDA Architectures

JUNE 2022

CONTENTS

Page

1. GPU

GPU stands for Graphics Processing Unit. It is a special processing unit responsible for processing graphics and images. It is located on the motherboard or video card. In the past, these processes were done with the CPU. However, as more graphics-intensive applications increased, CPU performance began to decline. That's why GPUs were developed to improve graphical processing. [1]

GPUs were formerly used for rendering 2D and 3D images, animations and video. However, its usage areas have now expanded. GPUs are used in training deep neural networks for machine learning and artificial intelligence applications. It is frequently used in image processing applications. It is also used in augmented reality (AR) and virtual reality (VR) areas. [1]

The architectures of CPUs and GPUs are actually very similar. However, GPUs are focused on data parallelism and are designed in a system based on SIMD (Single Instruction Multiple Data), that is, the application of a process to more than one data. CPUs, on the other hand, focus on task parallelism and are designed to perform different operations. [1]

GPUs work with an architecture called parallel processing, where multiple processors perform different parts of the same task. Thanks to this architecture, which allows the GPU to perform multiple calculations at the same time, it processes images faster than CPUs. By combining more than one CPU, only parallel processing can be done. However, a single CPU has not this feature. However, CPUs have higher clock speed. Therefore, it can perform a single calculation faster than GPUs. So it is more suitable for basic computing tasks. [1] [2]

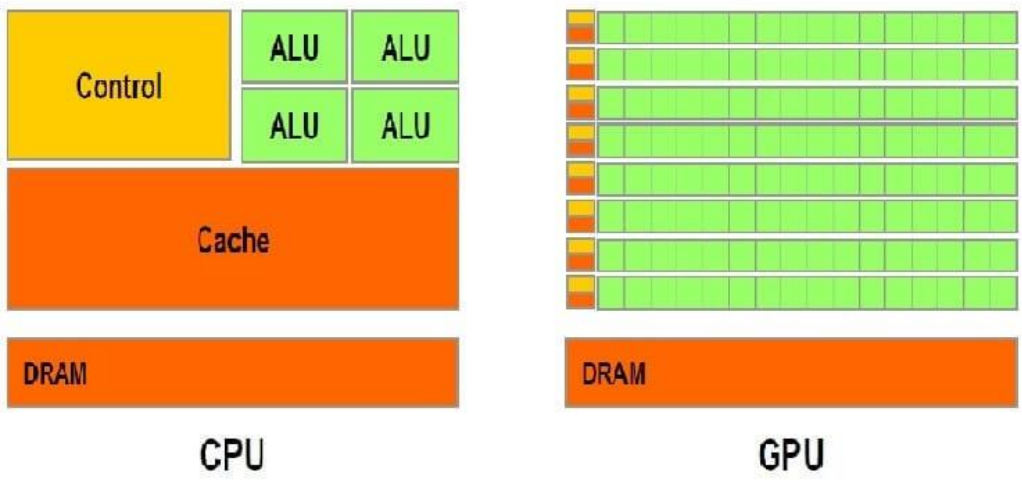

Figure 1: CPUs and GPUs design.

Figure 1 shows the design differences between CPUs and GPUs. A CPU is designed to improve sequential code performance.

ALU is a unit in which mathematical and logical operations are performed. Since this unit is much more on the GPU, GPUs have the ability to perform the same operation on many data. CPUs, on the other hand, have more advanced control units than GPUs. So it can perform more complex operations.

A GPU can be integrated with a CPU on the same electronic circuit, on a graphics card, or on the motherboard of a personal computer or server.

GPUs are more efficient than CPUs when large blocks such as cryptographic hash functions, machine learning, molecular dynamics simulations, physics engines, sort algorithms are processed in parallel.

2. CUDA

CUDA stands for Compute Unified Device Architecture. CUDA is a parallel computing platform and programming model. CUDA was developed by Nvidia. It allows developers to use GPU power. This way developers can speed up their applications. CUDA provides access to the GPU's virtual instruction set and parallel computing elements to perform computational operations. Information such as CUDA usage areas, used frameworks and applications, CUDA libraries and available platforms are shown in Figure 2. [3][4]

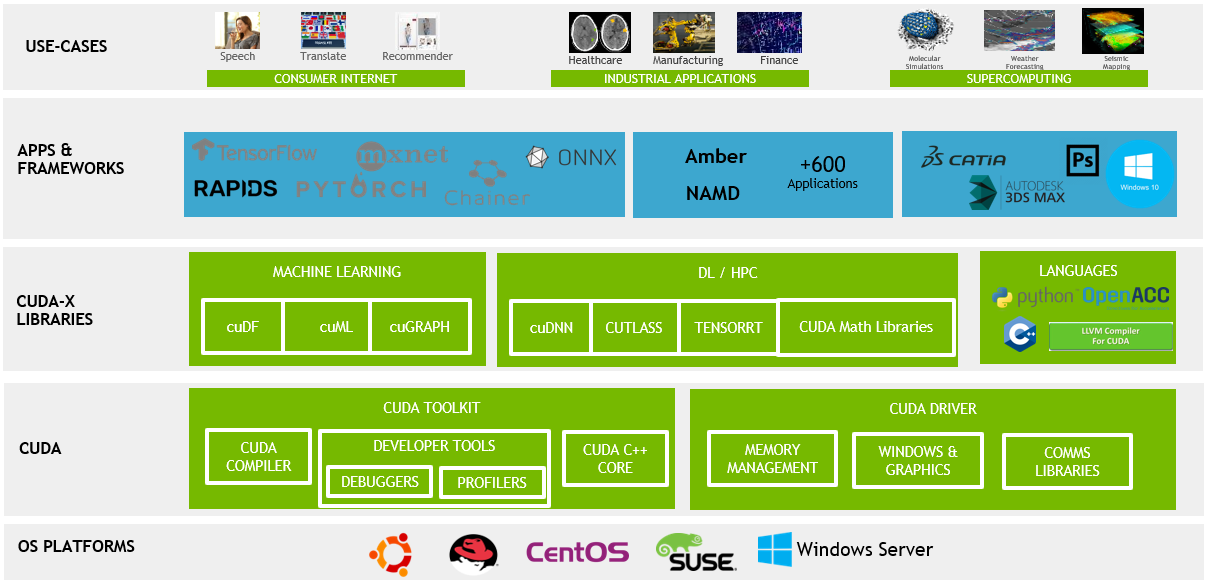

Figure 2: Brief information table about CUDA. [4]

In 2013, a team led by Ian Buck aimed to improve the C language for data parallel structures. For this came Brook, the first widely adopted programming model. Later, Ian Buck joined the Nvidia team and CUDA was released in 2006 for general purpose computing on GPUs. [3]

There are other APIs such as OpenCL to take advantage of the GPU power. There are also GPU models produced by AMD as competitors to NVIDIA. However, the combination of CUDA and NVIDIA GPUs provides the foundation for the fastest computers and dominates many application areas. Unlike its competitors, CUDA makes it easy for developers to use GPU resources. CUDA powered GPUs also support programming frameworks such as OpenMP, OpenACC, and OpenCL. [3]

OpenCL is a competitor to CUDA. OpenCL was released in 2009 by Apple and the Khronos Group. It has attracted attention because it provides generality. It is not limited to Intel/AMD CPUs with Nvidia GPUs. However, it did not perform as well as CUDA on Nvidia GPUs. [3]

CUDA is designed to work with programming languages such as C, C++, and Fortran. Additionally, the CUDA platform supports other computational interfaces, including OpenCL, DirectCompute, OpenGL Compute Shader, and C++ AMP. It also supports Python, Perl, Fortran, Java, Ruby, Lua, Common Lisp, Haskell, R, MATLAB, IDL, Julia and Mathematica. [5]

The CUDA toolkit includes libraries, debugging and optimization tools for application developers. It also includes a compiler for compiling applications and documentation for developers in the CUDA toolkit. CUDA libraries support all Nvidia GPUs. It also supports Deep learning, linear algebra, signal processing, and parallel algorithms. However, the latest generation GPUs are required for the best performance in training deep learning models. The easiest way to take advantage of GPUs in training models is to use libraries.

3. History of the GPU

Before computing with programmable GPUs like todays, GPU history was the first to develop non-programmable 3G graphics accelerators. Beginning in the 1980s, multi-chip 3D rendering engines were developed. But in the mid-1990s it became possible to integrate all the essentials on a single chip. From 1994 to 2001, it moved from simple drawing operations to full 3D pipeline operations such as transforms, lighting, rasterization, texturing, depth testing, and display. [5][6]

In 2001 Nvidia produced the GeForce 3 chip with programmable shading. The programmability of this chip was limited at first, but later additions were made to become more flexible and faster. [5][6]

With the introduction of the Nvidia GeForce 8800 in 2006, GPUs became generalized computing tools. General-purpose GPU (GPGPU) programming was developed to perform non-graphics operations on graphics-optimized architectures. GPGPU can be used in fields such as machine learning, scientific image processing, linear algebra and statistics. GPGPU at the time was the precursor to what is now called a Compute Shader such as CUDA, OpenCL, DirectCompute. [5][6]

In 2007 Nvidia introduced the CUDA platform, the first C-based development environment for GPUs. CUDA provided an easier and more efficient programming model than previous GPGPU approaches. [5][6]

In 2010, Nvidia released the Fermi architecture. Nvidia's Kepler line of graphics cards came out in 2012. Nvidia's Kepler GPU series followed, the Maxwell series produced by the same process. Pascal is the next generation of consumer graphics cards Nvidia released in 2016. GeForce 10 series cards are below this generation of graphics cards. [5][6]

4. CUDA Supported GPU Architectures

4.1. Tesla

Tesla is the codename for the GPU microarchitecture released by Nvidia in 2006. It is named after Nikola Tesla. It is Nvidia's first microarchitecture to use composite shaders. It was used with Tesla's GeForce 8 Series, GeForce 9 Series, GeForce 100 Series, GeForce 200 Series, and GeForce 300 Series GPUs. When the Tesla GeForce 7 series was introduced, it replaced the old fixed-pipeline microarchitectures.

4.1.1. Tesla Architecture

Figure 3 shows the GeForce 8800 GPU block diagram. This GPU has 128 streaming-processor (SP) cores organized as 16 streaming multiprocessors (SMs) in eight independent processing units called texture/processor clusters (TPCs). [7]

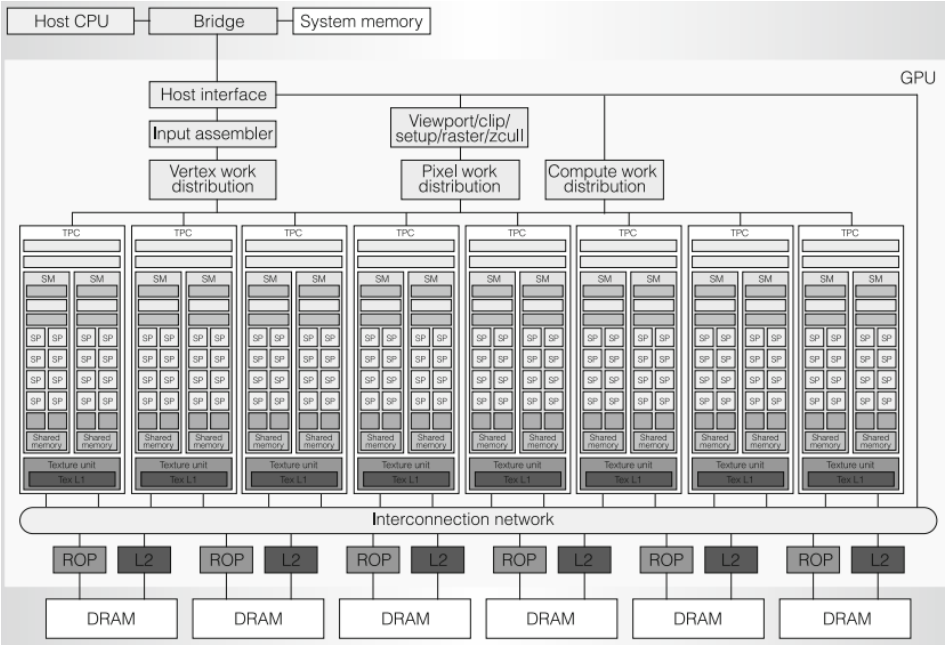

Figure 3: Tesla architecture

4.2. Fermi

Fermi is the codename for a GPU microarchitecture developed by Nvidia, introduced in April 2010. This architecture is named after the Italian physicist Ernico Fermi. It was used in the GeForce 400 series and GeForce 500 series. It was later used with Kepler in the GeForce 600 series, GeForce 700 series and GeForce 800 series. Figure 4 shows the high-level block diagram of the Fermi chip. It consists of multi-streaming multiprocessors (SM) consisting of 32 cores. Each of these cores executes one floating point and integer instruction per clock. SMs support L2 cache, host interface, GigaThread scheduler, and multiple DRAM interfaces. [6] [8]

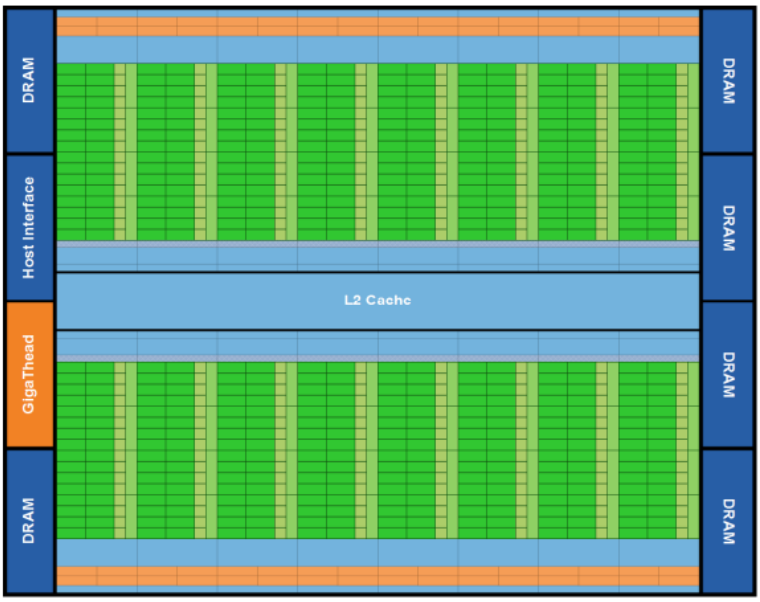

Figure 4: Fermi GPU architecture

4.2.1 The Programming Model

The complexity of the Fermi architecture is governed by a multi-level programming model that allows developers to focus on algorithm design, thereby increasing their productivity. The computational elements of algorithms that come in the framework of CUDA and Open CL are called kernels from signal processing jargon. An application and library function may consist of one or more cores. Kernels can be written in the C language. C language has been extended with keywords that express parallelism directly rather than loops. After compilation, cores consist of many threads executing in parallel for the same program. A thread can be thought of as performing a loop operation. To give an example through image processing, while a thread works on a pixel, all threads, namely the kernel, work on the whole image. [6]

4.2.2 The Streaming Multiprocessor

Fermi's Streaming Multiprocessor has 32 cores, 16 load-store units, 4 special units and 64K of slot SRAM. Each core can perform floating point and integer operations. 16 load-store units are used for memory operations. 64K local SRAM is split between cache and local memory. It also includes 32K log files and thread control logic. Figure 5 shows the Fermi SM architecture. [6]

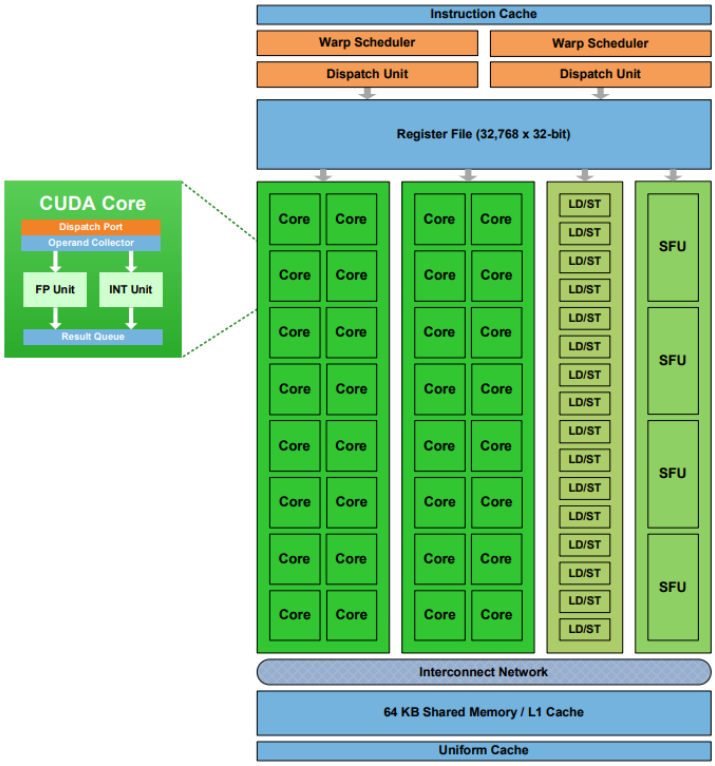

Figure 5: Fermi Streaming Multiprocessor.

4.3 Kepler

The Kepler architecture is the codename for the GPU microarchitecture released in April 2012 as a successor to the Fermi architecture. The Kepler architecture is Nvidia's first microarchitecture focused on energy efficiency. It is used in most GeForce 600 series, most GeForce 700 series and some GeForce 800M series GPUs. It was used in the GeForce 700 series and GeForce 800M series, along with the Maxwell architecture that came after Kepler. It is named after the German mathematician Johannes Kepler. [9]

4.3.1 Kepler Architecture

Figure 6 shows Kepler architecture. It includes 15 SMs and 64-bit memory controllers. Each SM contains 192 single-precision CUDA cores, 64 double-precision units, 32 SFUs, 32 LD/ST units and 16 texture units. [10]

Figure 6: Kepler architecture

4.4 Maxwell

Maxwell architecture is the codename of GPU microarchitecture released by Nvidia in February 2014. This architecture is named after James Clerk Maxwell, the founder of electromagnetic radiation theory. It is the successor of Kepler architecture. The Maxwell architecture was used

in later models of the GeForce 700 series. Maxwell uses an advanced Streaming Multiprocessor (SM) design to increase power efficiency. [11]

4.4.1 Maxwell Streaming Multiprocessor

Figure 7 shows the Maxwell Streaming Multiprocessor design. It consists of 16 SMs and four memory controllers. Each SM has been reconfigured to increase performance per watt. The SM is partitioned into four 32-CUDA core processing blocks, each with eight texture units, 8 SFUs and 8 LD/ST units. [12]

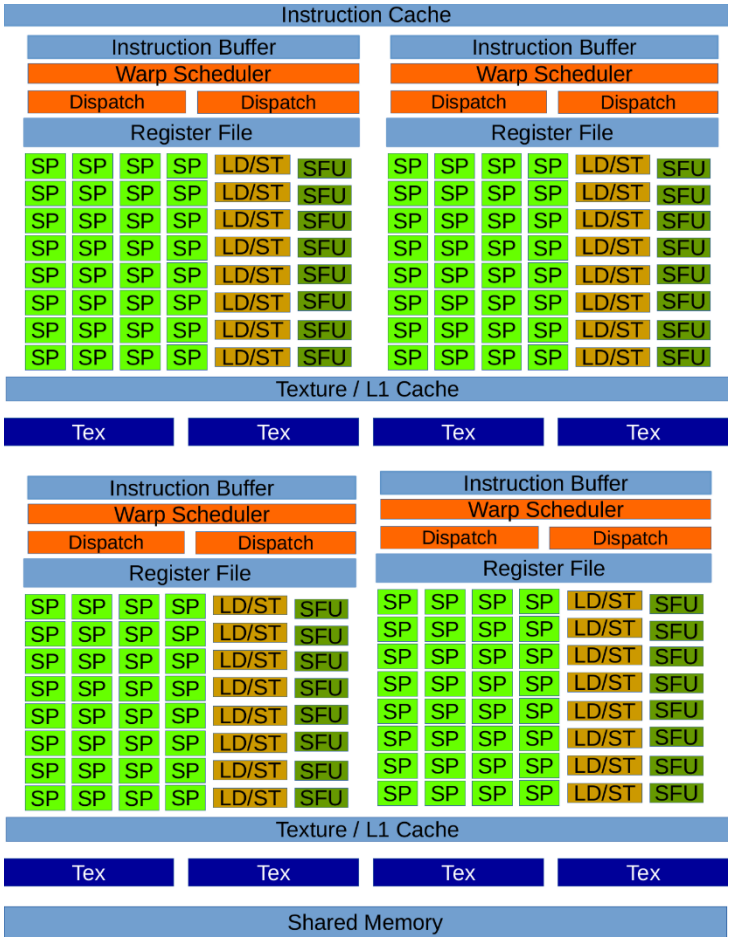

Figure 7: Maxwell Streaming Multiprocessor.

4.5 Pascal

Pascal is the codename for GPU microarchitecture released by Nvidia in April 2016. It is the successor of the Maxwell architecture. Pascal architecture is named after the French mathematician and physicist Blaise Pascal. It is primarily used in the GeForce 10 series, including the GeForce GTX 1080 and GTX 1070. [13]

4.5.1 Pascal Streaming Multiprocessor

Pascal Streaming Multiprocessor architecture is seen in Figure 8. The architecture has up to 60 SMs and 8 512-bit memory controllers. Each SM has 64 CUDA cores and four texture units. CUDA cores are able to process both 16-bit and 32-bit instructions and data. Although it has the same number of records as Maxwell and Kepler, it provides more records as it has more SMs. The shared memory bandwidth has been doubled to execute code more efficiently. [12]

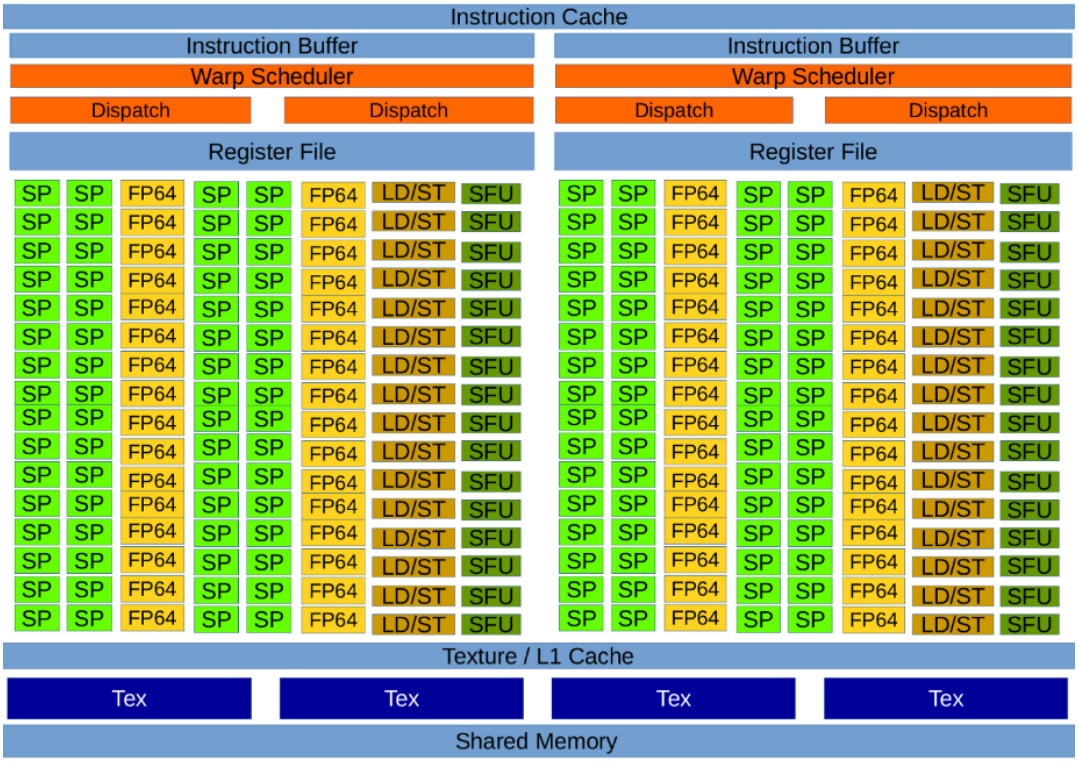

Figure 8: Pascal Streaming Multiprocessor.

4.6 Volta

Volta architecture is the codename of GPU microarchitecture released by Nvidia in December 2017. This architecture is named after the Italian chemist and physicist Alessandro Volta. This architecture was originally announced in 2013 but was released in 2017. It is Nvidia's first chip specifically designed to have deep learning performance superior to regular CUDA cores. [14]

4.6.1 Volta Streaming Multiprocessor

Figure 9 shows Volta Streaming Multiprocessor architecture. The Volta board has 84 SMs and eight 512-bit memory controllers. Each SM has 64 FP32 CUDA cores, 64 INT32 CUDA cores, 32 FP64 CUDA Cores, 8 tensor cores for deep learning matrix arithmetic, 32 LD/ST units, 16

SFUs. Each SM is divided into 4 processing blocks, each containing a new L0 instruction cache to provide higher efficiency. [12]

Figure 9: Volta Streaming Multiprocessor.

4.7 Turing

It is the codename of GPU microarchitecture released in September 2018, developed by Turing Nvidia. It is named after the mathematician Alan Turing. This architecture was first introduced on Quadro RTX cards at SIGGRAPH 2018 and a week later on consumer GeForce RTX 20 series graphics cards at Gamescom. The Turing architecture is the first consumer product capable of real-time ray tracing. This has long been a target of the graphics industry. It contains special artificial intelligence processors (Tensor cores) and special ray processing processors (RT cores). [12] [15] [16]

Turing architecture has a fundamentally different target market than Volta. Volta is the fastest HBM2 and is designed specifically for training in data centers. The Turing architecture, on the other hand, is designed for Pro Visualization, video game, and rendering using Tensor Core.

4.7.1 Turing Architecture

Figure 10 shows Turing architecture. The Turing architecture includes six Graphics Processing Clusters (GPCs), 36 Texture Processing Clusters (TPCs), and 72 Streaming Multiprocessors (SMs). Each streaming processor contains 64 CUDA Cores, eight Tensor Cores, a 256 KB register file, four texture units, and 96 KB of L1/shared memory. [16]

Figure 10: Turing Architecture

4.7.2 Turing Streaming Multiprocessor

Figure 11 shows the Turing Streaming Multiprocessor architecture. Turing SM is divided into four processing blocks. Each block contains 16 FP32 Cores, 16 INT32 Cores, two Tensor Cores, one warp scheduler, and one dispatch unit. [16]

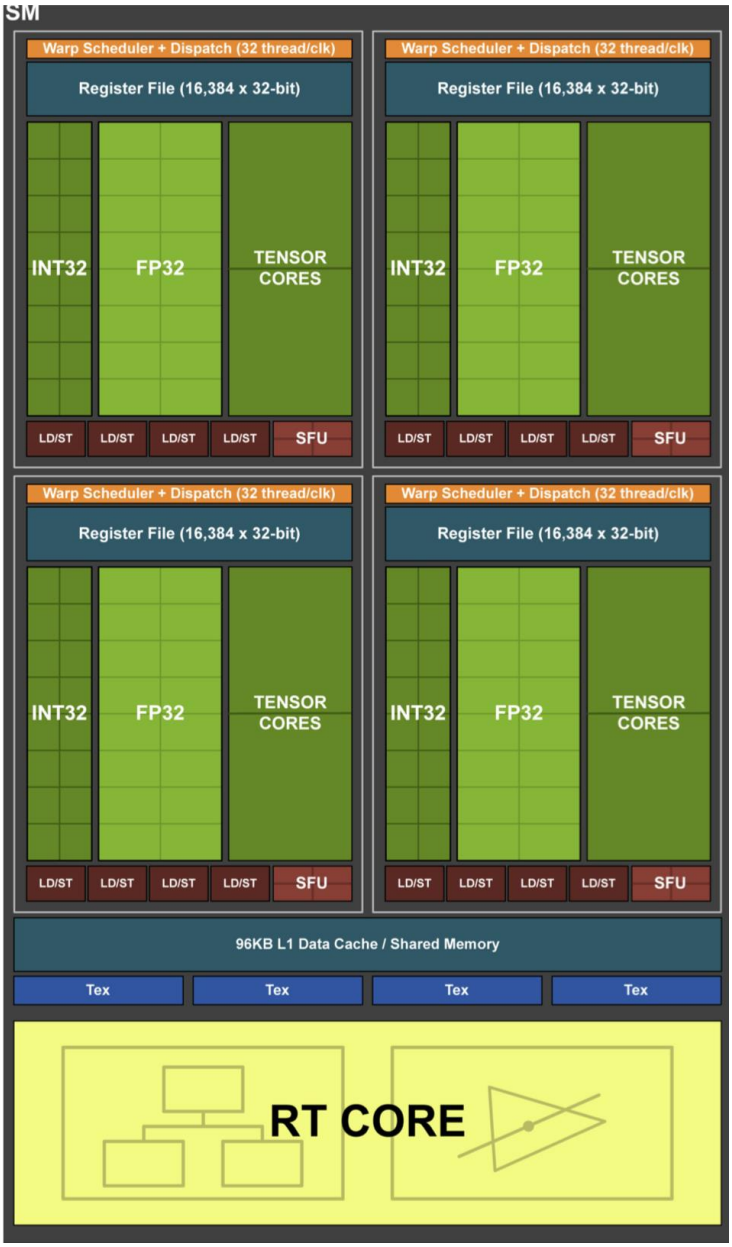

Figure 11: Turing Streaming Multiprocessor.

4.8 Ampere

Ampere architecture is the codename of GPU micro architecture introduced in May 2020, developed by Nvidia. This architecture is named after the French mathematician and physicist André-Marie Ampère. It is the successor of both Volta and Turing architectures. In May 2020, the GeForce 30 series consumer GPUs were announced. In November 2020, the A100 GPU was announced. [17]

4.8.1 Ampere Streaming Multiprocessor

Figure 12 shows the Ampere Streaming Multiprocessor (SM) architecture. Each Ampere SM contains four processing blocks with each having L0 cache for data caching, a warp scheduler, 16 INT32 CUDA cores, 16 FP32 CUDA cores, 8 FP64 CUDA cores, 8 LD/ST cores, a Tensor core for matrix multiplication, and a 16K 32-bit register file. Each SM has an 192 KB of combined shared memory and L1 data cache. It has 40MB cache to increase GPU-level performance. The L2 cache is split into two partitions to achieve higher bandwidth. [12]

Figure 12: Ampere Streaming Multiprocessor.

4.9 Latest Announced GPU Architecture: Hopper

Hopper architecture is the codename of GPU Data Center microarchitecture introduced by Nvidia. This architecture is named after American computer scientist and United States Navy Rear Admiral Grace Hopper. Nvidia GTC 2022 on March 22 has officially announced the 2022 Hopper GPU microarchitecture and the H100 GPU. [18]

The H100 GPU is designed to deliver an order of magnitude performance leap for larger scale AI and HPC. For AI and HPC models, H100 GPUs with InfiniBand interconnect have 30x the performance of A100 GPUs. [19]

It is 6x faster than the A100 GPU, thanks to the fourth-generation Tensor Cores built into each SM. It uses distributed shared memory. Communication between SMs is provided. Load and store multiple SMs into shared memory. It includes the HBM3 memory subsystem, which provides 2 times the bandwidth increase over the previous generation. The H100 SXM5 GPU is the first HBM3 memory GPU to offer 3TB/s memory bandwidth. [19]

4.9.1 Hopper Architecture

Figure 13 shows the Hopper architecture. This architecture has 8 GPCs, 72 TPCs (9 TPCs/GPC), 2 SMs/TPC, 144 SMs per full GPU. Each SM contains 128 FP32 CUDA Cores and 4 fourth generation Tensor Cores. It also includes 6 HBM3 or HBM2e stacks, 12 512-bit memory controllers and 60 MB L2 cache. [19]

Figure 13: Hopper architecture

4.9.2 Hopper Streaming Multiprocessor

With IEEE FP64 and FP32 in the Hopper streaming processor, processing 3 times faster than the A100 architecture can be performed. It contains 256 KB of combined shared memory and L1 data cache. Hopper architecture includes new asynchronous execution features. For this, it has a new Tensor Memory Accelerator (TMA) unit for efficient transfer of large blocks of data between global memory and shared memory. New thread block cluster feature exposes control of locality across multiple SMs. Figure 14 shows the Hopper streaming processor. [19]

Figure 14: Hopper streaming multiprocessor.

4.10 GPU Architectures and CUDA

The CUDA versions supported by GPU cards are listed below. [20]

- Tesla cards support CUDA SDK 1.0 to CUDA SDK 7.0.
- Fermi cards support CUDA SDK 3.2 to CUDA SDK 8.0. Deprecated from CUDA 9, support completely dropped from CUDA 10.
- Kepler cards support CUDA SDK 5.0 to CUDA SDK 10.0. Deprecated from CUDA SDK 11.
- Maxwell cards support CUDA SDK 6.0 to CUDA SDK 11. It will be removed in future versions.
- Pascal cards support CUDA SDK 8.0 and later.
- Volta cards support CUDA SDK 9.0 and later.
- Turing cards support CUDA SDK 10.0 and later.
- Ampere cards support CUDA SDK 11.1 and later.
- Hopper cards support CUDA SDK 12.0 and later.

5. CUDA Libraries

The latest version of CUDA, CUDA Toolkit 11.7.0, was released in May 2022. CUDA includes cuBLAS – CUDA Basic Linear Algebra Subroutines library, CUDART – CUDA Runtime library, cuFFT – CUDA Fast Fourier Transform library, cuRAND – CUDA Random Number Generation library, cuSOLVER – CUDA based collection of dense and sparse direct solvers, cuSPARSE – CUDA Sparse Matrix library, NPP – NVIDIA Performance Primitives library, nvGRAPH – NVIDIA Graph Analytics library, NVML – NVIDIA Management Library, NVRTC – NVIDIA Runtime Compilation library for CUDA C++, nView – NVIDIA nView Desktop Management Software, nvJPEG – Hybrid (CPU and GPU) JPEG processing, nvJPEG2000 – JPEG 2000 encoder and decoder libraries. Some of these libraries are described below. [21]

cuBLAS is CUDA Basic Linear Algebra Subroutine library. The cuBLAS library is an implementation of BLAS (Basic Linear Algebra Subprograms). Allows the user to access GPU resources. The cuBLAS library offers three APIs, the cuBLAS API, the cuBLASXt API, and the cuBLASLt. [22] [23]

The cuSOLVER library is a package based on the cuBLAS and cuSPARSE libraries. It is a high-level package that provides LAPACK-like features. It provides features such as common matrix factorization and triangulation solving routines for dense matrices. It is a GPI accelerated

library for linear system solutions and parsing for both dense and sparse matrices. It consists of two APIs, the cuSolver API and the cuSolverMG API. [22] [24]

cuSPARse is implemented in the Nvidia CUDA runtime. The cuSPARSE library is designed to be called from C and C++. contains a set of basic linear algebra subroutines used to handle sparse matrices. [22] [25]

The cuFFT library does Fast Fourier Transform (FFT). It consists of two separate libraries. These are cuFFT and cuFFTW. The cuFFT library provides high performance on GPUs while the cuFFTW library makes it use GPUs with minimal effort. FFT is a divide and conquer algorithm. FFT is used to calculate discrete Fourier transforms of complex or real-valued datasets. FFT is used for computing discrete Fourier transforms of complex or real-valued data sets. [22] [26]

The cuRAND library is a random number generation library. It can generate high-quality pseudo-random and semi-random numbers. It focuses on the simple and efficient generation of generated numbers. [22] [27]

nvJPEG library is a GPU accelerated JPEG codec library. The nvJPEG library provides GPU-accelerated JPEG encoding and decoding functionality. For image formats used in deep learning and hyperscale multimedia applications. [22] [28]

NPP library is a library created for CUDA-accelerated 2D image and signal processing. Flexibility is maximized while maintaining high applicability and performance. [22] [29]

The cuTENSOR library is a GPU-accelerated tensor linear algebra library. It provides high performance tensor contraction, reduction and elementwise operations. cuTENSOR is used to train deep learning learning models, accelerate applications in computer vision, quantum chemistry, and computational applications in physics. [22] [30]

cuSPARSELt library is a high-performance CUDA library for sparse matrix-matrix multiplication.cuSPARSELt library allows developers to leverage the computational resources of GPUs. [22] [31]

The nvJPEG2000 library handles decoding and encoding of JPEG2000 images on Nvidia GPUs. It does this in a high-performance and GPU-accelerated way. The JPEG2000 image format is used in deep learning, medical imaging, slinky sensing and digital cinema applications. Pascal and above are supported on GPUs architectures. [22] [32]

cuDNN aka CUDA Deep Neural Network library is a GPU accelerated library for open source deep learning framework. cuDNN provides highly tuned implementations for standard routines. Examples are forward and backward convolution, pooling, normalization, and activation layers. [33]

TensorRT is an SDK for high performance deep learning inference. It provides low latency and high throughput for inference applications with TensorRT. In short, it is a deep learning inference optimizer. [34]

DeepStream is a video inference library. There are many cameras and sensors around the world. The DeepStream library is used to process the data generated by these cameras and sensors. DeepStream offers a streaming analytics toolkit to understand sensor processing video, audio and image. [35]

REFERENCES

1. Gilis, A.S. (2020). graphics processing unit (GPU). Retrieved from https://www.techtarget.com/searchvirtualdesktop/definition/GPU-graphics-processing-unit
2. Kirk, D.B. Hwu, W.W. (2016). Programming Massively Parallel Processors (Third Edition). Morgan Kaufmann. Pages 1-18. ISBN 9780128119860. https://doi.org/10.1016/B978-0-12-811986-0.00001-7.
3. Heller, M. (2018). What is CUDA? Parallel programming for GPUs. Retrieved from https://www.infoworld.com/article/3299703/what-is-cuda-parallel-programming-for-gpus.html
4. Oh, F. (2012). What Is CUDA? Retrieved from https://blogs.nvidia.com/blog/2012/09/10/what-is-cuda-2/
5. Wikipedia contributors. (2022, June 16). Graphics processing unit. In *Wikipedia, The Free Encyclopedia*. Retrieved June 18, 2022, from https://en.wikipedia.org/wiki/Graphics_processing_unit
6. Glaskowsky, P.N. (2009). NVIDIA's Fermi: The First Complete GPU Computing Architecture. Retrieved from https://www.nvidia.com/content/PDF/fermi_white_papers/P.Glaskowsky_NVIDIA's_Fermi-The_First_Complete_GPU_Architecture.pdf
7. Lindholm, E., Nickolls, J., Oberman, S., & Montrym, J. (2008). NVIDIA Tesla: A Unified Graphics and Computing Architecture. IEEE Micro, 28(2), 39–55. doi:10.1109/mm.2008.31.
8. Wikipedia contributors. (2022, June 13). Fermi (microarchitecture). In *Wikipedia, The Free Encyclopedia*. Retrieved June 18, 2022, from https://en.wikipedia.org/wiki/Fermi_(microarchitecture)
9. Wikipedia contributors. (2022, June 7). Kepler (microarchitecture). In *Wikipedia, The Free Encyclopedia*. Retrieved June 19, 2022, from https://en.wikipedia.org/wiki/Kepler_(microarchitecture)
10. S, Sandor. (2014). Optimizing General Purpose Computations Using Kepler Based Graphics Accelerators.
11. Wikipedia contributors. (2022, April 20). Maxwell (microarchitecture). In *Wikipedia, The Free Encyclopedia*. Retrieved June 19, 2022, from https://en.wikipedia.org/wiki/Maxwell_(microarchitecture)

12. PD, A. (2020) How the hell are GPUs so fast? A HPC walk along Nvidia CUDA-GPU architectures. From zero to nowadays. Retrieved from https://towardsdatascience.com/how-the-hell-are-gpus-so-fast-a-e770d74a0bf
13. Wikipedia contributors. (2022, June 13). Pascal (microarchitecture). In *Wikipedia, The Free Encyclopedia*. Retrieved June 19, 2022, from https://en.wikipedia.org/wiki/Pascal_(microarchitecture)
14. Wikipedia contributors. (2022, May 3). Volta (microarchitecture). In *Wikipedia, The Free Encyclopedia*. Retrieved June 19, 2022, from https://en.wikipedia.org/wiki/Volta_(microarchitecture)
15. Wikipedia contributors. (2022, May 19). Turing (microarchitecture). In Wikipedia, The Free Encyclopedia. Retrieved June 20, 2022, from https://en.wikipedia.org/wiki/Turing_(microarchitecture)
16. Kilgariff, E. Moreton, H. Stam, N and Bell, B. (2018). NVIDIA Turing Architecture In-Depth. Retrieved from https://developer.nvidia.com/blog/nvidia-turing-architecture-in-depth/
17. Wikipedia contributors. (2022, June 16). Ampere (microarchitecture). In *Wikipedia, The Free Encyclopedia*. Retrieved June 20, 2022, from https://en.wikipedia.org/wiki/Ampere_(microarchitecture)
18. Wikipedia contributors. (2022, May 19). Hopper (microarchitecture). In *Wikipedia, The Free Encyclopedia*. Retrieved June 20, 2022, from https://en.wikipedia.org/wiki/Hopper_(microarchitecture)
19. Andersch, M. Palmer, G. Krashinsky, R. Stam, N. Mehta, V. Brito, G and Ramaswamy, S. (2022). NVIDIA Hopper Architecture In-Depth. Retrieved from https://developer.nvidia.com/blog/nvidia-hopper-architecture-in-depth/
20. Shimoni, A. (2020). Matching CUDA arch and CUDA gencode for various NVIDIA architectures. Retrieved from https://arnon.dk/matching-sm-architectures-arch-and-gencode-for-various-nvidia-cards/
21. Wikipedia contributors. (2022, June 16). CUDA. In *Wikipedia, The Free Encyclopedia*. Retrieved June 20, 2022, from https://en.wikipedia.org/wiki/CUDA
22. Nvidia Developer. CUDA Libraries Documentation. Retrieved from https://docs.nvidia.com/cuda-libraries/index.html
23. CUDA TOOLKIT DOCUMENTATION, (2022). cuBLAS. Retrieved from https://docs.nvidia.com/cuda/cublas/index.html

24. CUDA TOOLKIT DOCUMENTATION, (2022). cuSOLVER. Retrieved from https://docs.nvidia.com/cuda/cusolver/index.html
25. CUDA TOOLKIT DOCUMENTATION, (2022). cuSPARSE. Retrieved from https://docs.nvidia.com/cuda/cusparse/index.html
26. CUDA TOOLKIT DOCUMENTATION, (2022). cuFFT. Retrieved from https://docs.nvidia.com/cuda/cufft/index.html
27. CUDA TOOLKIT DOCUMENTATION, (2022). cuRAND. Retrieved from https://docs.nvidia.com/cuda/curand/index.html
28. CUDA TOOLKIT DOCUMENTATION, (2022). nvJPEG . Retrieved from https://docs.nvidia.com/cuda/nvjpeg/index.html
29. CUDA TOOLKIT DOCUMENTATION, (2022). NVIDIA 2D Image And Signal Performance Primitives (NPP). Retrieved from https://docs.nvidia.com/cuda/npp/index.html
30. CUDA TOOLKIT DOCUMENTATION, (2022). cuTENSOR: A High-Performance CUDA Library For Tensor Primitives. Retrieved from https://docs.nvidia.com/cuda/cutensor/index.html
31. CUDA TOOLKIT DOCUMENTATION, (2022). cuSPARSELt: A High-Performance CUDA Library for Sparse Matrix-Matrix Multiplication. Retrieved from https://docs.nvidia.com/cuda/cusparselt/index.html
32. CUDA TOOLKIT DOCUMENTATION, (2022). NVIDIA nvJPEG2000. Retrieved from https://docs.nvidia.com/cuda/nvjpeg2000/index.html
33. Nvidia Developer. (n.d). NVIDIA cuDNN. Retrieved from https://developer.nvidia.com/cudnn
34. Nvidia Developer. (n.d). NVIDIA TensorRT. Retrieved from https://developer.nvidia.com/tensorrt
35. Nvidia Developer. (n.d). DeepStream SDK. Retrieved from https://developer.nvidia.com/deepstream-sdk