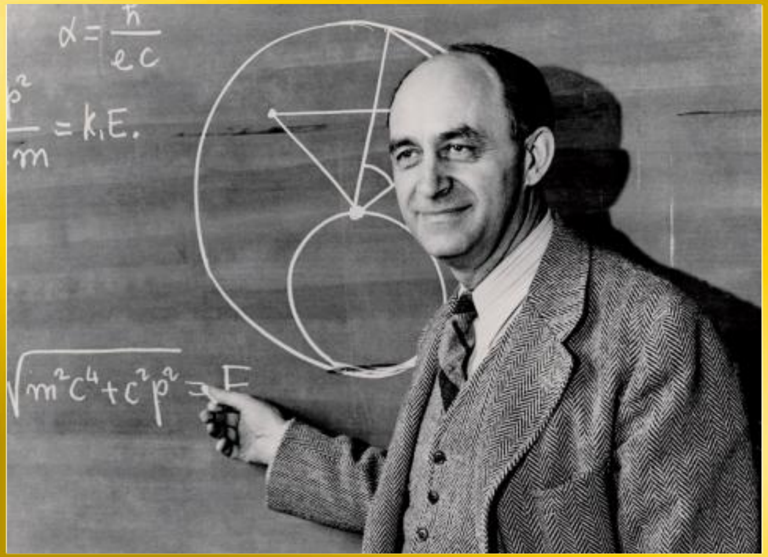
FERMI

Year 2010



Micro-architecture. Chip Size : 40nm

Card specifications:

- *Clock frequency:* **1.5 GHz (Estimated)**
- *Peak Performance:* **1.5 TFlops**
- *Number of transistors:* **3.0 Billions**
- Or
  - *Total Number of FP32 Cuda Core:* **512**
  - *Total Number of FP64 Cuda Core:* **256**
- *Global memory clock*: **4 GHz**
- *DRAM Bandwith :* **192 GB/s**
- *Max DRAM :* **6 GB**
- *DRAM Type:* **GDDR5**
- *L2 Unified Cache:* **768KB**
- *Number of SMs: **16***
- *Number of TPCs: **NA***

Streaming Multiprocessor (SM) specifications:

- *Number of CUDA Cores per SM*: **32**
- Or
  - *Number of FP32 Cuda Cores per SM:* **32**
  - *Number of FP64 Cuda Cores per SM:* **16**
  - *Number of Tensor Core per SM:* **NA**
- *Number of TU:* **4**
- *Number of SFUs per SM:* **4**
- *Number of LD/ST per SM: **16***
- *Number of Warp Schedulers:* **2**
- *L1 Cache / Shared Memory:* **64KB**
- *Shared Memory:* **32KB of 32bits**
- *Registers:* **32KB of 32bits**

weakness resistance retreat cost

source : https://www.nvidia.com/content/PDF/fermi_white_papers/NVIDIA_Fermi_Compute_Architecture_Whitepaper.pdf

KEPLER

Year 2012

Micro-architecture. Chip Size : 28nm

Card specifications:

- *Clock frequency:* **1.1 GHz**
- *Peak Performance:* **3.1 TFlops**
- *Number of transistors:* **3.5 Billions**
- Or
  - *Total Number of FP32 Cuda Core:* **1536**
  - *Total Number of FP64 Cuda Core:* **X**
- *Global memory clock*: **6 GHz**
- *DRAM Bandwith :* **192 GB/s**
- *Max DRAM :* **4 GB**
- *DRAM Type:* **GDDR5**
- *L2 Unified Cache:* **768KB**
- *Number of SMs:* ***8 SMX***
- *Number of TPCs:* ***NA***

Streaming Multiprocessor (SM) specifications:

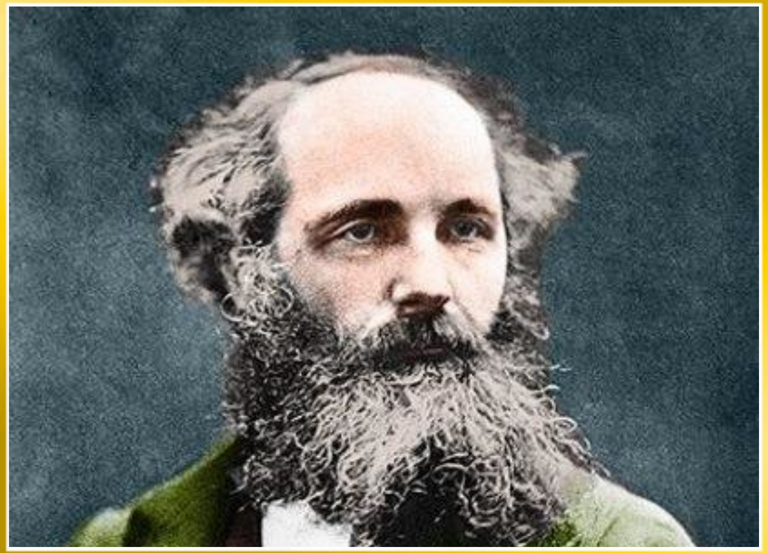
- *Number of CUDA Cores per SM*: **192**
- Or
  - *Number of FP32 Cuda Cores per SM:* **192**
  - *Number of FP64 Cuda Cores per SM:* **96**
- *Number of Tensor Core per SM:* **NA**
- *Number of TU:* **16**
- *Number of SFUs per SM:* **32**
- *Number of LD/ST per SM:* **32**
- *Number of Warp Schedulers:* **4**
- *L1 Cache / Shared Memory:* **up to 128KB**
- *Shared Memory:* **up to 128KB of 32bits**
- *Registers:* **up to 128KB of 32bits**

weakness resistance retreat cost

source : http://www.cse.msu.edu/~cse820/lectures/NvidiaGK110ArchNotes.pdf

MAXWELL

Year 2014



Micro-architecture. Chip Size : 28nm

Card specifications:

- *Clock frequency:* **1.1 GHz**
- *Peak Performance:* **4,6 TFlops**
- *Number of transistors:* **8.1 Billions**
- Or
  - *Total Number of FP32 Cuda Core:* **3072**
  - *Total Number of FP64 Cuda Core:* **96**
- *Global memory clock*: **1.7 GHz**
- *DRAM Bandwith :* **336 GB/s**
- *Max DRAM :* **12 GB**
- *DRAM Type:* **GDDR5**
- *L2 Unified Cache:* **2MB**
- *Number of SMs:* **24 SMM**
- *Number of TPCs:* ***NA***

Streaming Multiprocessor (SM) specifications:

- *Number of CUDA Cores per SM*: **128**
- Or
  - *Number of FP32 Cuda Cores per SM:* **128**
  - *Number of FP64 Cuda Cores per SM:* **4**
- *Number of Tensor Core per SM:* **NA**
- *Number of TU:* **8**
- *Number of SFUs per SM:* **32**
- *Number of LD/ST per SM:* **32**
- *Number of Warp Schedulers:* **4**
- *L1 Cache / Shared Memory:* **up to 128KB**
- *Shared Memory:* **up to 128KB of 32bits**
- *Registers:* **up to 256KB of 64bits**

weakness resistance retreat cost

source : https://international.download.nvidia.com/geforce-com/international/pdfs/GeForce-GTX-750-Ti-Whitepaper.pdf

PASCAL

Year 2016

Micro-architecture. Chip Size : 16nm

Card specifications:

- *Clock frequency:* **1.4 GHz**
- *Peak Performance:* **12 TFlops**
- *Number of transistors:* **15.6 Billions**
- *Total Number of FP32 Cuda Core:* **3840**
- *Total Number of FP64 Cuda Core:* **1920**
- *Global memory clock*: **1.4 GHz**
- *DRAM Bandwith :* **750 GB/s**
- *Max DRAM :* **16 GB**
- *DRAM Type:* **GDDR5X**
- *L2 Unified Cache:* **4MB**
- *Number of SMs:* **60**
- *Number of TPCs:* **30**

Streaming Multiprocessor (SM) specifications:

- *Number of CUDA Cores per SM*: **64**
- *Number of FP32 Cuda Cores per SM:* **64**
- *Number of FP64 Cuda Cores per SM:* **32**
- *Number of Tensor Core per SM:* **NA**
- *Number of TU:* **4**
- *Number of SFUs per SM:* **16**
- *Number of LD/ST per SM:* **16**
- *Number of Warp Schedulers:* **2**
- *L1 Cache / Shared Memory:* **up to 64KB of 32bits**
- Or
- *Shared Memory:* **up to 64KB of 32bits**
- *Registers: 64***KB of 32bits**

weakness resistance retreat cost

source : https://images.nvidia.com/content/pdf/tesla/whitepaper/pascal-architecture-whitepaper.pdf

VOLTA

Year 2017

Micro-architecture. Chip Size : 12nm

Card specifications:

- *Clock frequency:* **1.5 GHz**
- *Peak Performance:* **15.7 TFlops**
- *Number of transistors:* ***21.1*** **Billions**
- *Total Number of FP32 Cuda Core:* **5120**
- *Total Number of FP64 Cuda Core:* **2560**
- *Global memory clock*: **2 GHz**
- *DRAM Bandwith* : **900 GB/s**
- *Max DRAM :* **6GB**
- *DRAM Type:* **HBM2**
- *L2 Unified Cache:* ***6*****MB**
- *Number of SMs:* **84**
- *Number of TPCs:* ***42***

Streaming Multiprocessor (SM) specifications

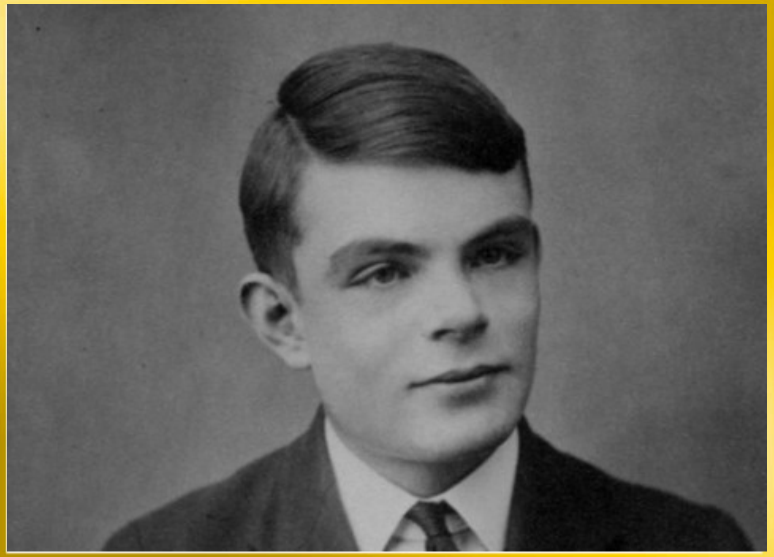
- *Number of CUDA Cores per SM*: **32**
- *Number of INT Cuda Cores per SM :* **64**
- *Number of FP32 Cuda Cores per SM:* **64**
- *Number of FP64 Cuda Cores per SM:* **32**
- *Number of Tensor Core per SM:* **8**
- *Number of TU:* ***4 (TEX)***
- *Number of SFUs per SM:* ***16***
- *Number of LD/ST per SM:* ***32***
- *Number of Warp Schedulers:* ***4***
- *L1 Cache / Shared Memory:* ***128*****KB**
- *Registers:* ***16*****K x 32 bits**

weakness resistance retreat cost

source : https://images.nvidia.com/content/volta-architecture/pdf/volta-architecture-whitepaper.pdf

TURING

Year 2018



Micro-architecture. Chip Size : 12nm

Card specifications:

- *Clock frequency:* **1.6 GHz**
- *Peak Performance:* **1.5 TFlops**
- *Number of transistors:* **18.6 Billions**
- *Total Number of FP32 Cuda Core:* **4608**
- *Total Number of Tensor Core:* **576**
- *Total Number of Ray Tracing Core:* **72**
- *Global memory clock*: **2 GHz**
- *DRAM Bandwith* : **672 GB/s**
- *Max DRAM* : **11GB**
- *DRAM Type:* **GDDR6**
- *L2 Unified Cache:* ***512*KB**
- *Number of SMs:* ***72***
- *Number of TPCs:* ***36***

Streaming Multiprocessor (SM) specifications

- *Number of CUDA Cores per SM*: **64**
- *Number of FP32 Cuda Cores per SM:* **64**
- *Number of FP64 Cuda Cores per SM:* **32**
- *Number of Tensor Core per SM:* **8**
- *Number of TU:* ***4***
- *Number of SFUs per SM:* **4**
- *Number of LD/ST per SM:* **16**
- *Number of Warp Schedulers:* **2**
- *L1 Cache / Shared Memory:* **64KB**
- *Shared Memory:* **16K or 48KB**
- *Registers:* ***4*16*K of 32 bits**

weakness **resistance** **retreat cost**

source : https://www.nvidia.com/content/dam/en-zz/Solutions/design-visualization/technologies/turing-architecture/NVIDIA-Turing-Architecture-Whitepaper.pdf

AMPERE

Year 2020

Micro-architecture. Chip Size : 7nm

Card specifications:

- *Clock frequency:* **1.6 GHz**
- *Peak Performance (FP32):* **19.5 TFlops**
- *Number of transistors:* **54 Billions**
- *Total Number of FP32 Cuda Core:* **6912**
- *Total Number of FP64 Cuda Core:* **3456**
- *Total Number of Tensor Core:* **432**
- *Global memory clock*: **2.4 GHz**
- *DRAM Bandwith* : **1.6TB/s**
- *Max DRAM* : **40GB**
- *DRAM Type:* **HBM2**
- *L2 Unified Cache:* ***40MB***
- *Number of SMs:* ***128***
- *Number of TPCs:* ***64***

Streaming Multiprocessor (SM) specifications

- *Number of CUDA Cores per SM*: **108**
- *Number of INT32 Cuda Cores per SM:* **64**
- *Number of FP32 Cuda Cores per SM:* **64**
- *Number of FP64 Cuda Cores per SM:* **32**
- *Number of Tensor Core per SM:* **4**
- *Number of TU:* ***4 (TEX)***
- *Number of SFUs per SM:* **4**
- *Number of LD/ST per SM:* ***32***
- *Number of Warp Schedulers:* ***4***
- *L1 Cache / Shared Memory:* ***192*KB**
- *Registers:* ***16K* x 32 bits**

weakness **resistance** **retreat cost**

source : https://devblogs.nvidia.com/nvidia-ampere-architecture-in-depth/

KERNEL

```
struct Vec3 { float x, y, z ; } ;

__global__ void my_kernel(  const Vec3* a,
                            Vec3 b,
                            float* c)
{
   int i = threadID;

   c[i] =  a[i].x * b.x
         + a[i].y * b.y
         + a[i].z * b.z;
}

dim3 DimGrid(100,50) ; // 100*50*1 = 5000 blocks
dim3 DimBlock(4,8,8) ; // 4*8*8 = 256 threads / blocks

my_kernel<<< DimGrid, DimBlock >>> (…) ;
```

Programming Model.

Definition:

A thread is a computation unit (function) that has a state and that can be paused and resumed that will be executed on the GPU or on the CPU.

You have 3 types of kernels:

- *__global__* : called by CPU but executed by GPU
- *__device__* : called and executed by GPU
- *__host__* : called and executed by CPU

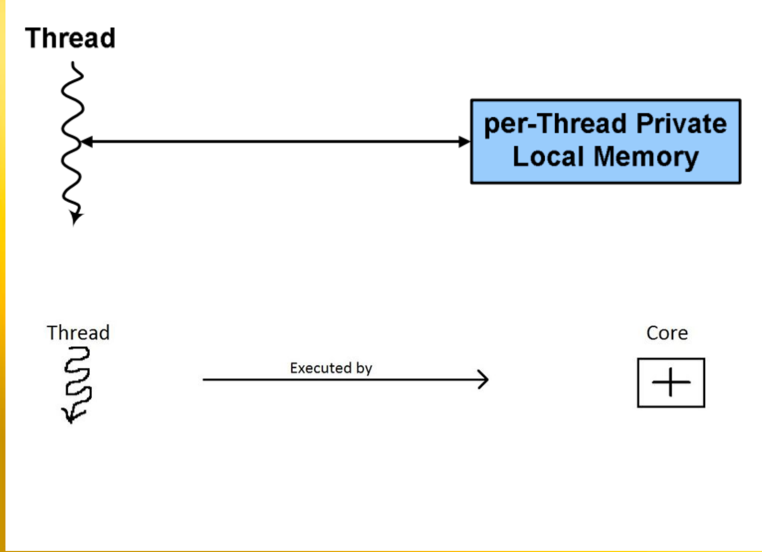
Calling kernel is made this way : *kernel <<< nBlocs, threadsPerBloc >>> (arguments);*

- *nBlocs* : size the thread grid to use
- *ThreadsPerBloc* : number of threads to execute simultaneously on each block

weakness **resistance** **retreat cost**

source : http://www.labri.fr/perso/guenneba/pghp_2015/Cours_PGHP_2015_02-IntroCUDA.pdf

THREAD



Programming Model.

Definition:

A GPU Thread is an instantiation of a function over a given data in a GPU Kernel (__*global*__ or __*device*__).
For parallel computing : 1 thread = 1 function application over 1 data.
Typically, each thread in a kernel will compute one element of an array. There is a common pattern to do this that most CUDA programs use are shown below.
Once a kernel is launched, it’s dimensions can’t change

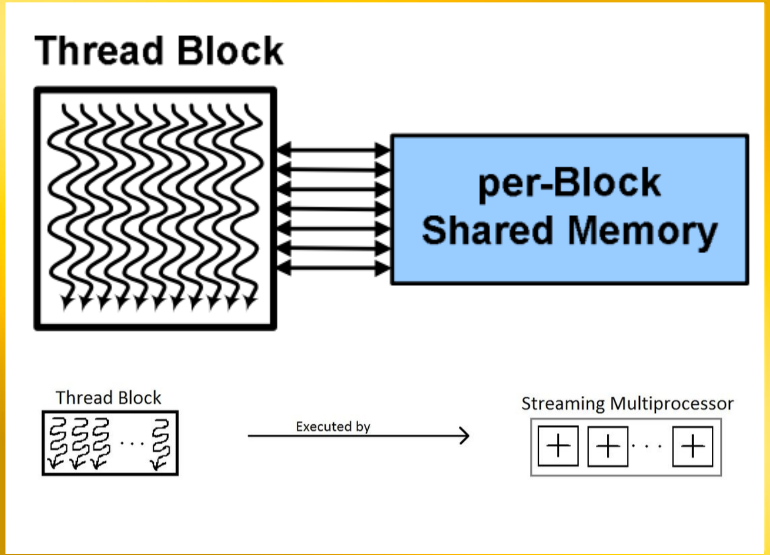
Memory: Local Memory

- Each *thread* has its own private local memory
- Only exists for the lifetime of the thread
- Generally handled automatically by the compiler

weakness resistance retreat cost

source : https://cs.nyu.edu/courses/fall15/CSCI-GA.3033-004/cuda-main.pdf

THREAD BLOCK



Programming Model.

Definition:

A **thread block** is a programming abstraction representing a group of threads that can be executed serially or in parallel. For better process and data mapping, threads are grouped into thread blocks. The number of threads in block varies with available shared memory. The threads in the same thread block run on the same stream processor. Threads in the same block can communicate with each other via shared memory, barrier synchronization or other synchronization primitives such as atomic operations.
Thread ID is unique within a block, Each block can execute in any order relative to other blocks.

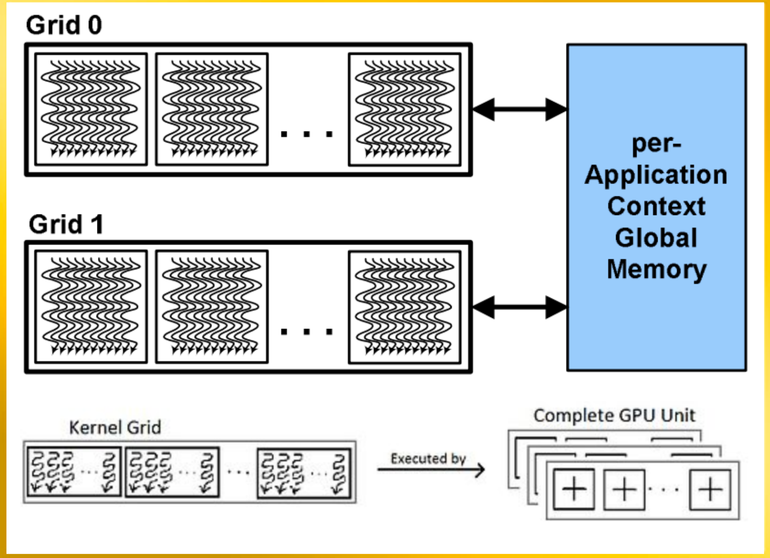
Memory: Shared Memory

- Each thread block has its own shared memory accessible only by threads within the block
- Much faster than local or global memory
- Requires special handling to get maximum performance
- Only exists for the lifetime of the block

weakness | resistance | retreat cost

source : https://cs.nyu.edu/courses/fall15/CSCI-GA.3033-004/cuda-main.pdf
https://en.wikipedia.org/wiki/Thread_block

GRID



Programming Model.

Definition:

Multiple thread blocks are combined to form a grid. All the blocks in the same grid contain the same number of threads. Grids can be used for computations that require a large number of thread blocks to operate in parallel.
The number of thread blocks in a grid is usually dictated by the size of the data being processed or the number of processors in the system, which it can greatly exceed.
All threads in a grid execute the same kernel function.
All blocks in a grid have the same dimensions.

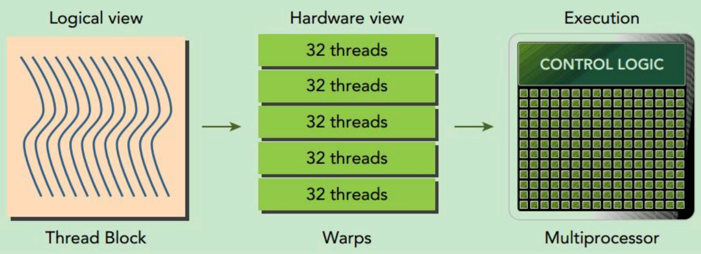
Memory: Global Memory

- This memory is accessible to all threads as well as the host (CPU).
- Global memory is allocated and deallocated by the host
- Used to initialize the data that the GPU will work on

weakness **resistance** **retreat cost**

source : https://cs.nyu.edu/courses/fall15/CSCI-GA.3033-004/cuda-main.pdf
https://cs.nyu.edu/courses/spring12/CSCI-GA.3033-012/lecture5.pdf https://en.wikipedia.org/wiki/Thread_block

WARP



Programming Model - Multi-core Units

Definition:

Warp are giving GPU the ability to execute the same application code on hardware with different number of execution resources is called transparent scalability. Warp are like the Software to Hardware translator.

A hardware design can exploit the commonality of the threads belonging to a warp by combining their memory accesses and assuming that it is fine to pause and resume all the threads at the same time, rather than deciding on a per-thread basis.

The warp size is the number of threads running concurrently on an *Multi-Processor*.

Warps are managed by warp scheduler that will orchestrate the execution of the Thread blocks on the physical architecture meaning *Multi-core Units aka CUDA/RT/Tensor Cores.*

weakness **resistance** **retreat cost**

source : https://images.nvidia.com/content/volta-architecture/pdf/volta-architecture-whitepaper.pdf

Streaming Multiprocessor (SM)

Architecture.

Definition:

Streaming Multiprocessor (SM) is the part where the magic happens. This designed was first introduced in 2010 with Fermi and was derived with SMX with Kepler (2012) and SMM with Maxwell (2014) but was reintroduced since 2016 with Pascal and Volta (2017).

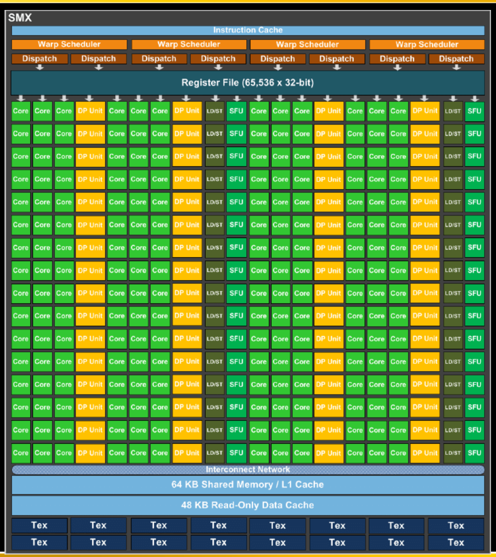
It's composed of :

- Scheduling tools (Dispatch Units, Warp Schedulers)
- Memory (L0, L1 Cache)
- Register File : that will link main memory data and computation components residing in Multi-core units
- Multi-core units : that will perform the calculations but also components that will manage the memory flows between Memory units and computation cores

weakness **resistance** **retreat cost**

Source : https://research.nvidia.com/sites/default/files/pubs/2012-12_Unifying-Primary-Cache/Gebhart_MICRO_2012.pdf
http://www.irisa.fr/alf/downloads/collange/cours/gpuprog_ufmg/gpuprog_1.pdf

Streaming Multiprocessor X (SMX)



Architecture.

Definition:

Streaming Multiprocessor X (SMX) is a variation of SM.

The main difference with SM is that NVIDIA tried at one point to reduce the number of SM and to make bigger SM. Basically SMX are SM under steroid in terms of number of cores but might be less efficient if you consider that the shared resources/cores are reduced. However packing everything like this saves space and leave room for more transistors (therefore cores) on the same GPU surface … Still, the trade off is interesting and was introduced with Kepler Micro-architecture (2012).

Just as SM, SMX are composed of :

- Scheduling tools (Dispatch Units, Warp Schedulers)
- Memory (L0, L1 Cache)
- Register File : that will link main memory data and computation components residing in Multi-core units
- Multi-core units : that will perform the calculations but also components that will manage the memory flows between Memory units and computation cores

weakness **resistance** **retreat cost**

Streaming Multiprocessor M (SMM)

Architecture.

Definition:

Streaming Multiprocessor M (SMM) is a variation of SM and SMX used for Maxwell Micro-architecture (2014).

If SMX are SM under steroid. One would describe SMM as a well balanced body building diet along with a small dose of steroid. The number of cores for the SMM is still higher that usual SM however the drawback of the ultra compact SMX design due to not so good ratio of available shared resources per core is more balanced in SMM with 4 subsections having their own dedicated shared resources such as dispatch Unit , instruction buffer, and warp schedulers.

Just as SM and SMX, SMM are composed of :

- Scheduling tools (Dispatch Units, Warp Schedulers)
- Memory (L0, L1 Cache)
- Register File : that will link main memory data and computation components residing in Multi-core units
- Multi-core units : that will perform the calculations but also components that will manage the memory flows between Memory units and computation cores

weakness **resistance** **retreat cost**

Special Function Unit (SFU)

Multi-core Units aka CUDA Cores

Definition:

Execute transcendental instructions such **as sin, cosine, reciprocal, and square root**. Each SFU executes one instruction per thread, per clock; a warp executes over eight clocks. The SFU pipeline is decoupled from the dispatch unit, allowing the dispatch unit to issue to other execution units while the SFU is occupied.

weakness resistance retreat cost

source : https://en.wikipedia.org/wiki/Fermi_(microarchitecture)

Texture Unit (Text/TMU)

Multi-core Units aka CUDA Cores

Definition:

A TMU is able to rotate, resize, and distort a bitmap image (performing texture sampling), to be placed onto an arbitrary plane of a given 3D model as a texture. This process is called texture mapping.

In the past TMU were separated physically from the SM but the Fermi Micro-Architecture introduced it as a component in the SM making it part of the GPGPU strategy.

weakness resistance retreat cost

source : https://www.nvidia.com/object/Projective_Texture_Mapping.html - https://en.wikipedia.org/wiki/Texture_mapping_unit

LOAD/STORE UNIT

Multi-core Units aka CUDA Cores

Definition:

To feed the computation cores it's needed at one point to fetch data from the memory (L1 cache data) and push it to the cores. This is called load and store instructions and it's handled by the SM LD/ST units.

LD/ST units operate on the register which size vary from one micro-architecture to another. Memory accesses are managed at each clock operations covering X-bytes block splitted over X memory addresses

Reading the memory for all ALU assigned in blocks operations (thanks to a warp) can take multiple cycles depending on memory address, core and LD/ST width.

weakness resistance retreat cost

source : https://devtalk.nvidia.com/default/topic/1016724/coalesced-access-and-hardware-load-store-units/
https://research.nvidia.com/sites/default/files/pubs/2012-12_Unifying-Primary-Cache/Gebhart_MICRO_2012.pdf

INT ALU (Half precision)

Multi-core Units aka CUDA Cores

Definition:

Floating Point Unit provide the capability to GPU to perform Fused Multiple Add instructions (FMA or Fused Multiply Accumulate - FMAC) but also addition, multiplication or divisions. Special/Complex operations are handled by the SFU.

INT or HP (stands for Half Precision) ALU (Arithmetic Logical Unit) are performing FMA over 16 Bits elements

weakness **resistance** **retreat cost**

source : https://images.nvidia.com/content/volta-architecture/pdf/volta-architecture-whitepaper.pdf

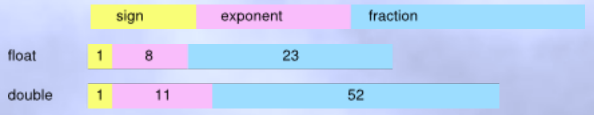
FP32 ALU (Single Precision)

Multi-core Units aka CUDA Cores

Definition:

Floating Point Unit provide the capability to GPU to perform Fused Multiple Add instructions (FMA or Fused Multiply Accumulate - FMAC) but also addition, multiplication or divisions. Special/Complex operations are handled by the SFU.

FP32 or SP (stands for Single Precision) ALU (Arithmetic Logical Unit) are performing FMA over 32 Bits elements
In the last GPU Generations starting from Pascal the FP32 units were also able to process Half Precision (HP) FP16.



weakness **resistance** **retreat cost**

source : https://images.nvidia.com/content/volta-architecture/pdf/volta-architecture-whitepaper.pdf
https://devblogs.nvidia.com/new-features-cuda-7-5/

FP64 ALU (Double Precision)

Multi-core Units aka CUDA Cores

Definition:

Floating Point Unit provide the capability to GPU to perform Fused Multiple Add instructions (FMA or Fused Multiply Accumulate - FMAC) but also addition, multiplication or divisions. Special/Complex operations are handled by the SFU.

FP64 or DP (stands for Double Precision) ALU (Arithmetic Logical Unit) are performing FMA over 64 Bits elements

	sign	exponent	fraction
float	1	8	23
double	1	11	52

weakness **resistance** **retreat cost**

source : https://docs.nvidia.com/cuda/floating-point/index.html

Tensor Core

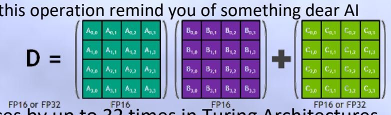
Multi-core Units aka CUDA Cores

Definition:

Tensor cores are pretty new to GPGPUs as it was introduced in 2017 with Volta Micro-architecture.

Tensor cores were introduced in 2017 with Volta Micro-architecture. As graphical rendering is all about 4x4 matrices as objects have x,y,z and rotation which makes object representation being referred as 4x4x4 matrices. To perform graphical rendering for an object you need to have the object in its referential, then move it to the real world referential and finally project it into the "camera" referential (clipping). Everything is just about Multiply and accumulate 4x4x4 matrices. This is also perfect for Deep Learning applications (https://www.ovh.com/fr/blog/deep-learning-explained-to-my-8-year-old-daughter/)

Easy enough the big thing with TensorCore is the smart way it was implemented as it's performing mixt precision calculations as presented below… does this operation remind you of something dear AI programmers #ConvolutionNeuralNetworks

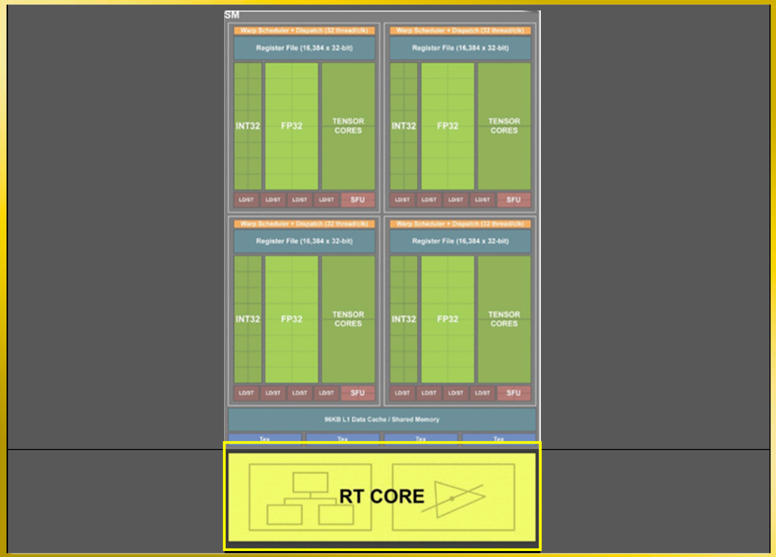


And here is how you improve performances by up to 32 times in Turing Architectures

weakness **resistance** **retreat cost**

source : https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/tesla-t4/Turing-Tensor-Core_30fps_FINAL_736x414.gif
https://devblogs.nvidia.com/programming-tensor-cores-cuda-9/

Ray Tracing Core (RT Core)



Multi-core Units aka CUDA Cores

Definition:

With Ray Tracing core the Unified shaders architecture is now being questioned. As explained in the "Fermi Microarchitecture card" GPU prior to GPGPU (started with Fermi) were designed with hardware specifications corresponding to image rendering pipeline.
By implementing RT Cores we are (partially) going back to the good old day of image rendering pipelines encoded into hardware where pixel and vertex shaders are separated.

Ray Tracing is a computing technic to emulate the light effects in image rendering. RT Core / RTX is a combination of Ray Tracing mathematical calculation combined with intuitive light effect prediction using Deep Learning Super Sampling (DLSS) executing on… Tensor cores. All of this should lead to augmented rasterization by using denoising and upsampling.

weakness **resistance** **retreat cost**

source : https://images.nvidia.com/content/volta-architecture/pdf/volta-architecture-whitepaper.pdf
https://en.wikipedia.org/wiki/Nvidia_RTX. https://www.youtube.com/watch?v=yKbmVJBqA_Y