Nvidia Ampere GA102

Deepti, Aditya, Jialin, Sowmya

Agenda

Graphics Processing Unit (GPU)

- GPU renders images, video and 2D or 3D animations for display. A GPU performs quick math calculations and frees up the CPU to do other things.
- Integrated GPUs are located on the CPU
- Discrete GPUs live on their own card

Machine Learning

3D Graphics Rendering

Video Editing

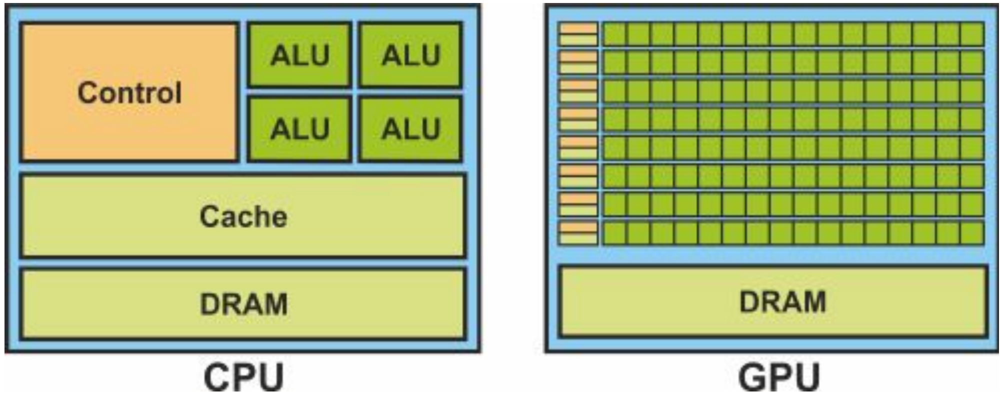
Cryptocurrency Mining

[1]

CPU vs GPU

- Low Compute Density
- Optimized for serial operations
- Shallow pipelines (<30 stages)
- Low latency tolerance
- Complex control logic

- High Compute Density
- Optimized for parallel operations
- Deep pipelines (100s)
- High latency tolerance
- High throughput

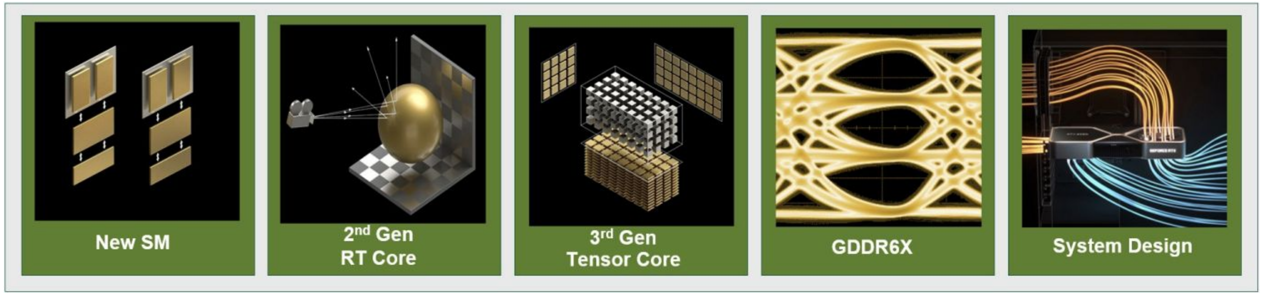


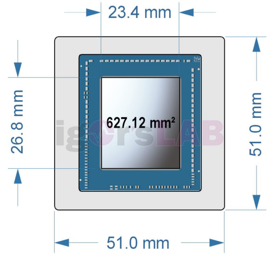
[1, 2]

Highlights

The GA102 GPU is NVIDIA's largest Ampere GPU for the gaming & consumer segment. It is used in the top of the line GeForce RTX 3090 and GeForce RTX 3080 graphics cards.

It was released on September 1, 2020.





Microarchitecture

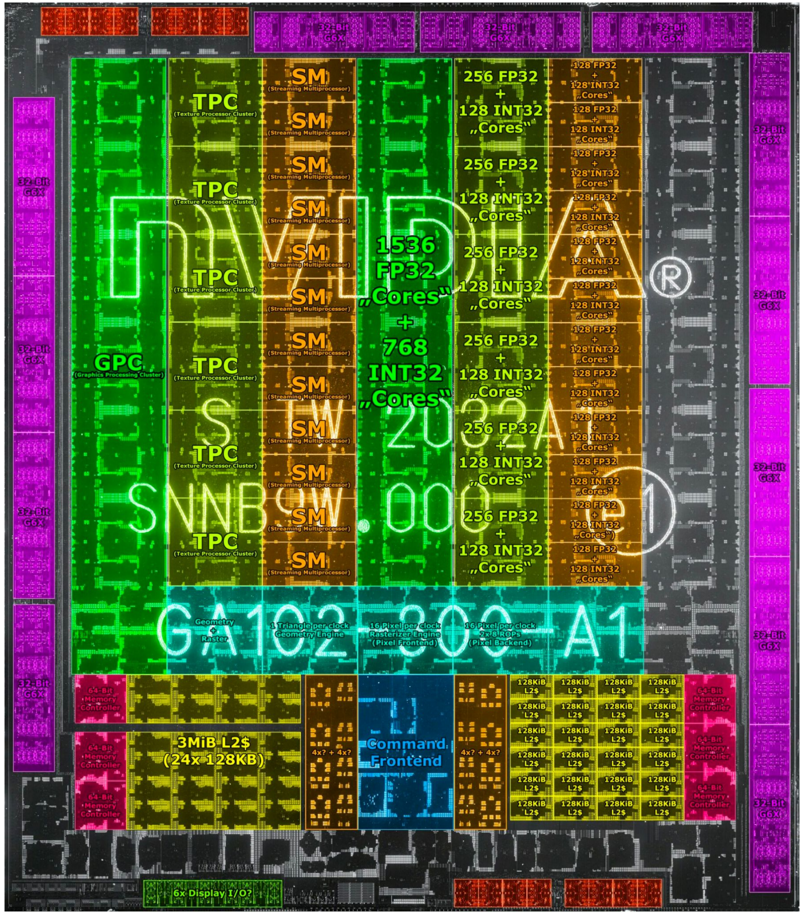
GigaThread Engine

Graphics Processing Clusters (GPCs)

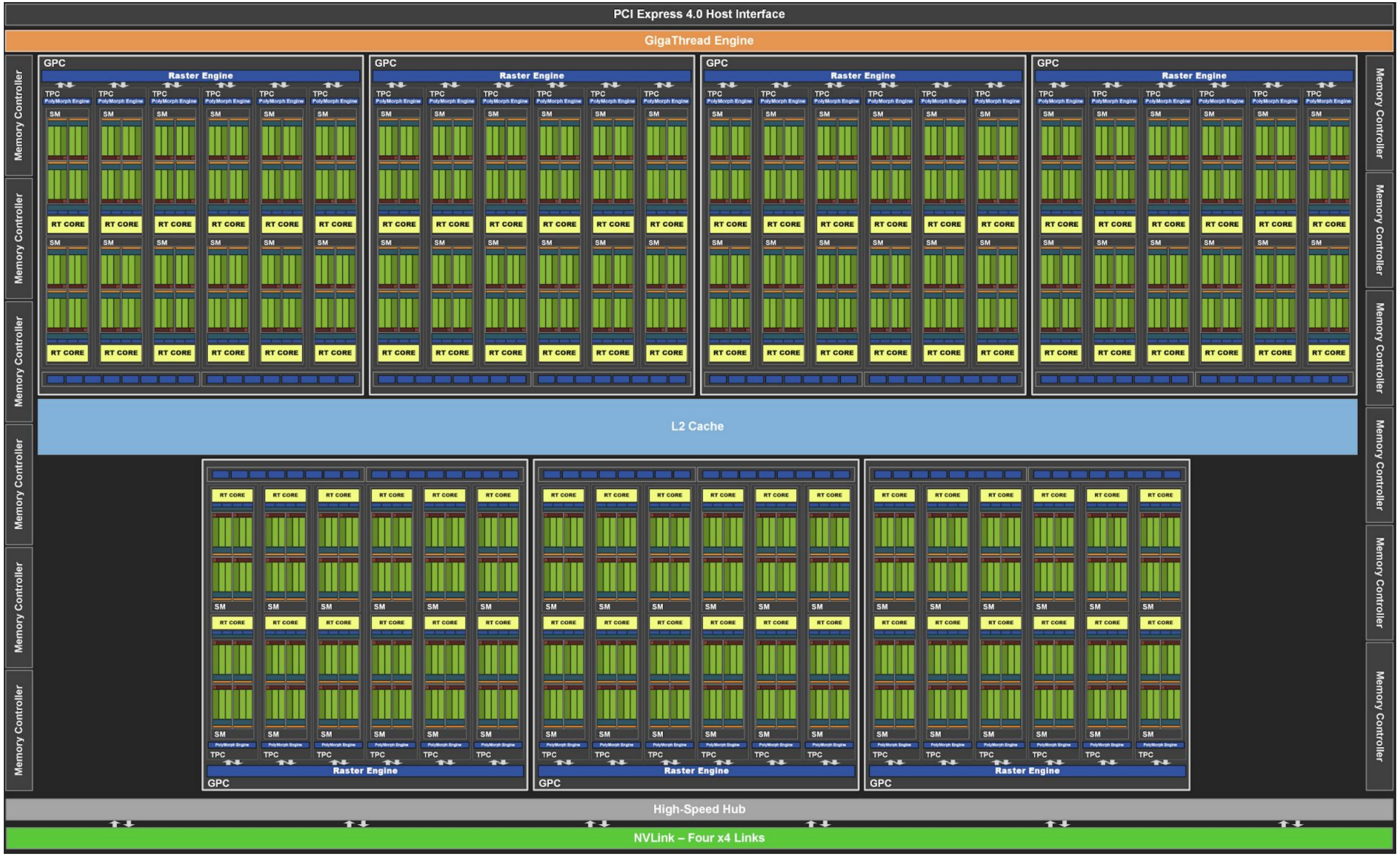
Texture Processing Clusters (TPCs)

Streaming Multiprocessors (SMs)

Memory controllers



32-Bit G6X
GPC
(Graphics Processing Cluster)
TPC
(Texture Processor Cluster)
SM
(Streaming Multiprocessor)
1536 FP32 „Cores" + 768 INT32 „Cores"
256 FP32 + 128 INT32 „Cores"
128 FP32 + 128 INT32 „Cores"
Geometry + Raster
1 Triangle per clock Geometry Engine
16 Pixel per clock Rasterizer Engine (Pixel Frontend)
16 Pixel per clock 2x8 ROPs (Pixel Backend)
64-Bit Memory Controller
3MiB L2$ (24x 128KB)
4x? + 4x?
Command Frontend
128KiB L2$
6x Display I/O?

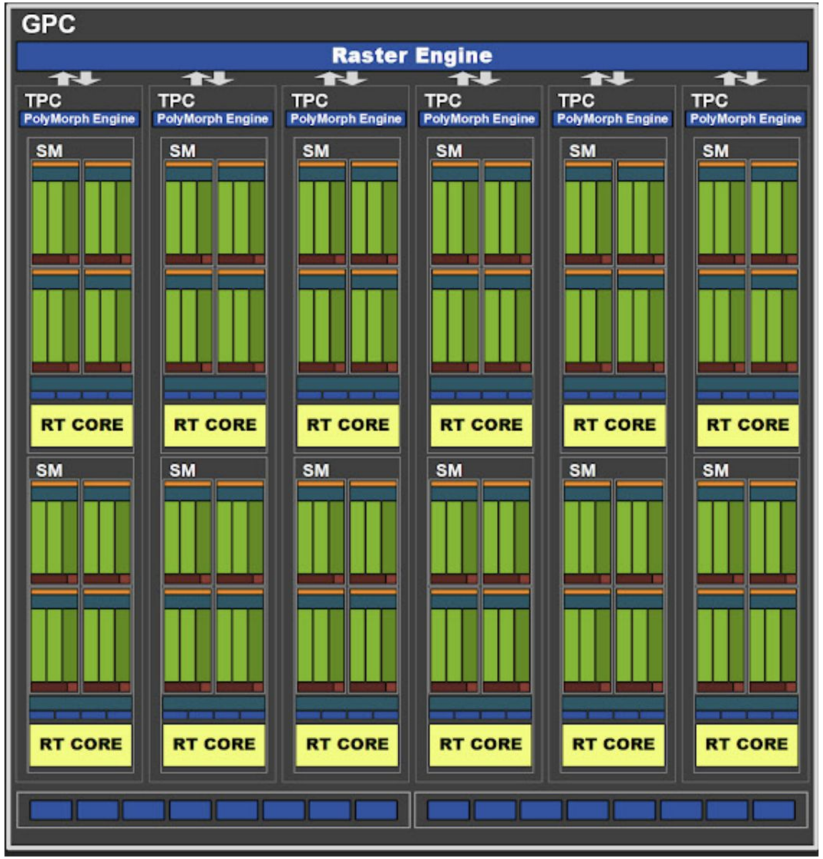

PCI Express 4.0 Host Interface
GigaThread Engine
GPC
Raster Engine
TPC
PolyMorph Engine
SM
RT CORE
Memory Controller
L2 Cache
High-Speed Hub
NVLink – Four x4 Links

Graphics Processing Clusters

The GPC is the dominant high-level hardware block with all of the key graphics processing units.

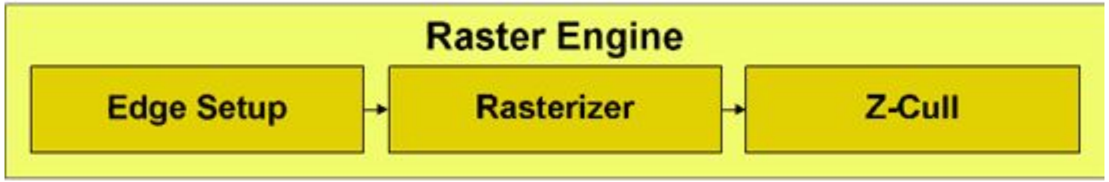
Each GPC includes,

1. a dedicated Raster Engine
2. two raster operator partitions (each partition containing eight ROP units)
3. six TPCs



Graphics Processing Clusters

1. Raster Engine



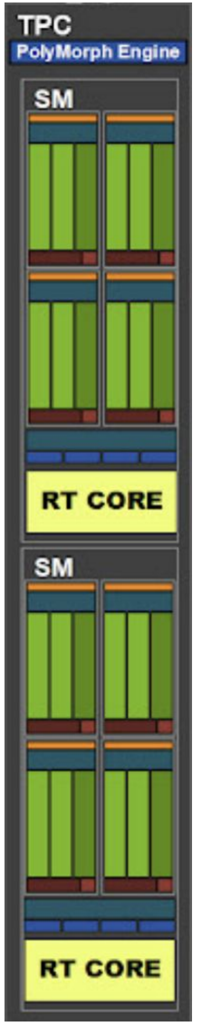
2. Raster Operators - GPU's output is assembled into a bitmapped image ready for display.
   - Previously ROPs were tied to the memory controller and L2 cache.
   - GA102 boosts performance of raster operations by increasing the total number of ROPs, and eliminating throughput mismatches between the scan conversion frontend and raster operations backend.

[1, 2]

Texture Processing Clusters (TPCs)

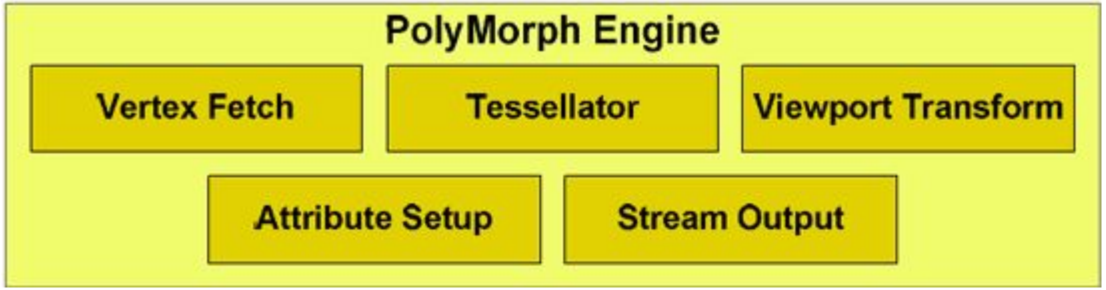
Each TPC includes,

1. two SMs
2. one PolyMorph Engine



Texture Processing Clusters (TPCs)

PolyMorph Engine - execution unit that handles geometry

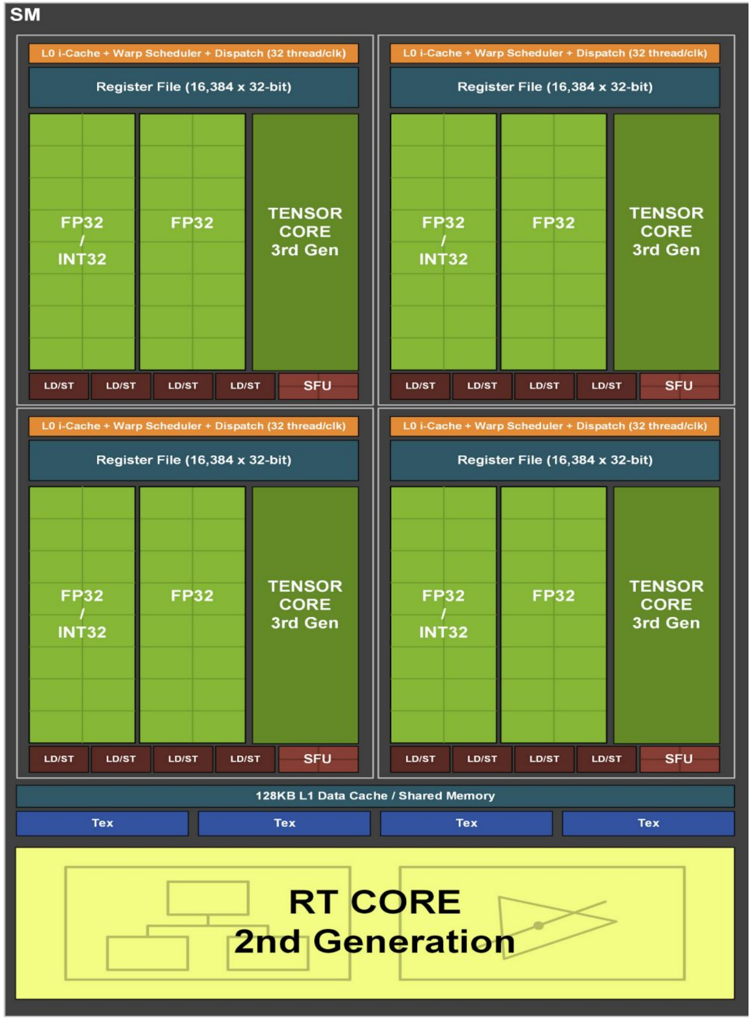


[1]

Streaming Multiprocessors (SMs)

Each SM contains,

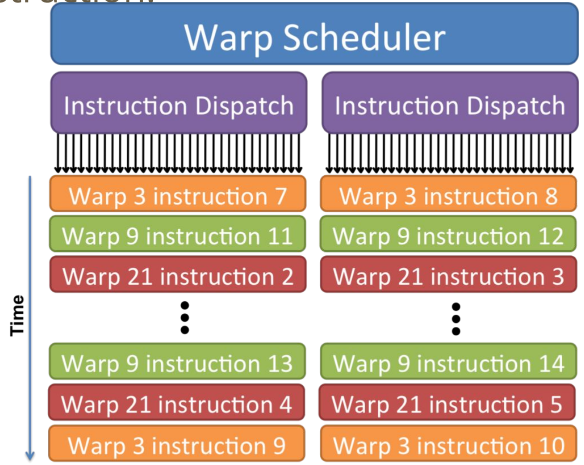
1. 128 CUDA Cores
2. 4 third-generation Tensor Cores
3. 4 Texture Units
4. 1 second-generation Ray Tracing Core
5. 128 KB of L1/Shared Memory, which can be configured for differing capacities
6. 256 KB Register File
7. L0 instruction cache, one warp scheduler, one dispatch unit

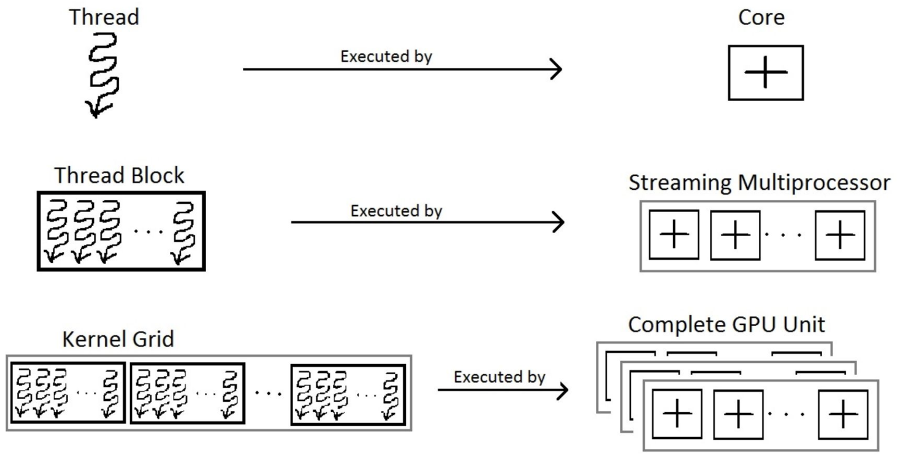

SM
L0 i-Cache + Warp Scheduler + Dispatch (32 thread/clk)
Register File (16,384 x 32-bit)
FP32 / INT32
FP32
TENSOR CORE 3rd Gen
LD/ST
SFU
128KB L1 Data Cache / Shared Memory
Tex
RT CORE 2nd Generation

Streaming Multiprocessors (SMs)

These are general purpose processors with a low clock rate target and a small cache. The primary task of an SM is that it must execute several thread blocks in parallel. They support instruction-level parallelism but not branch prediction.

A warp is a set of 32 threads within a thread block such that all the threads in a warp execute the same instruction.





Memory hierarchy background

- **Global Memory** - Default main memory of GPU
- **Local Memory** - Abstraction on global memory with thread-level scope
- **Shared Memory** - On chip memory accessible to active threads in an SM
- **Constant Memory** - Predefined read-only space within global memory

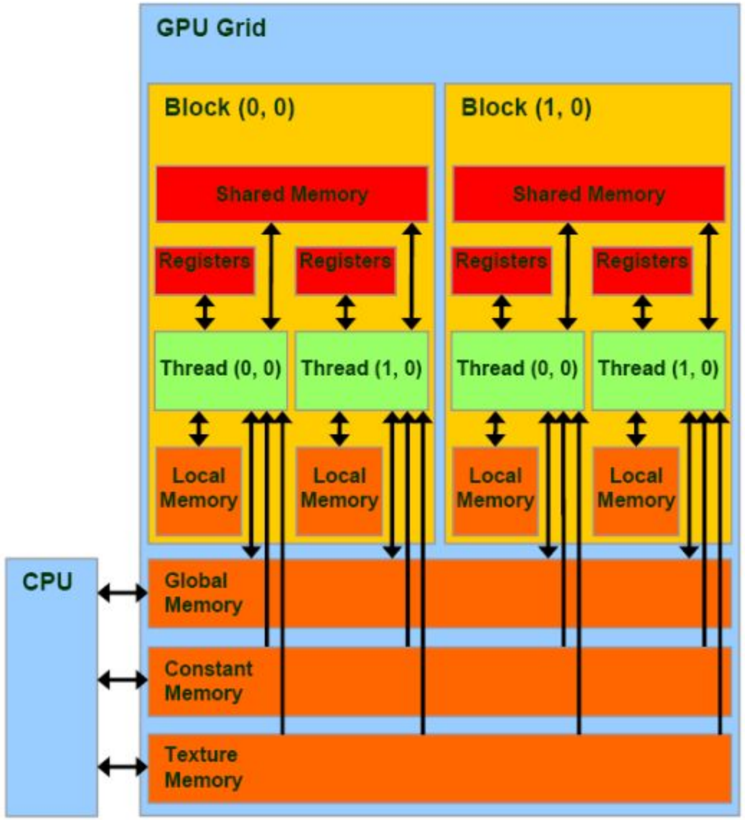
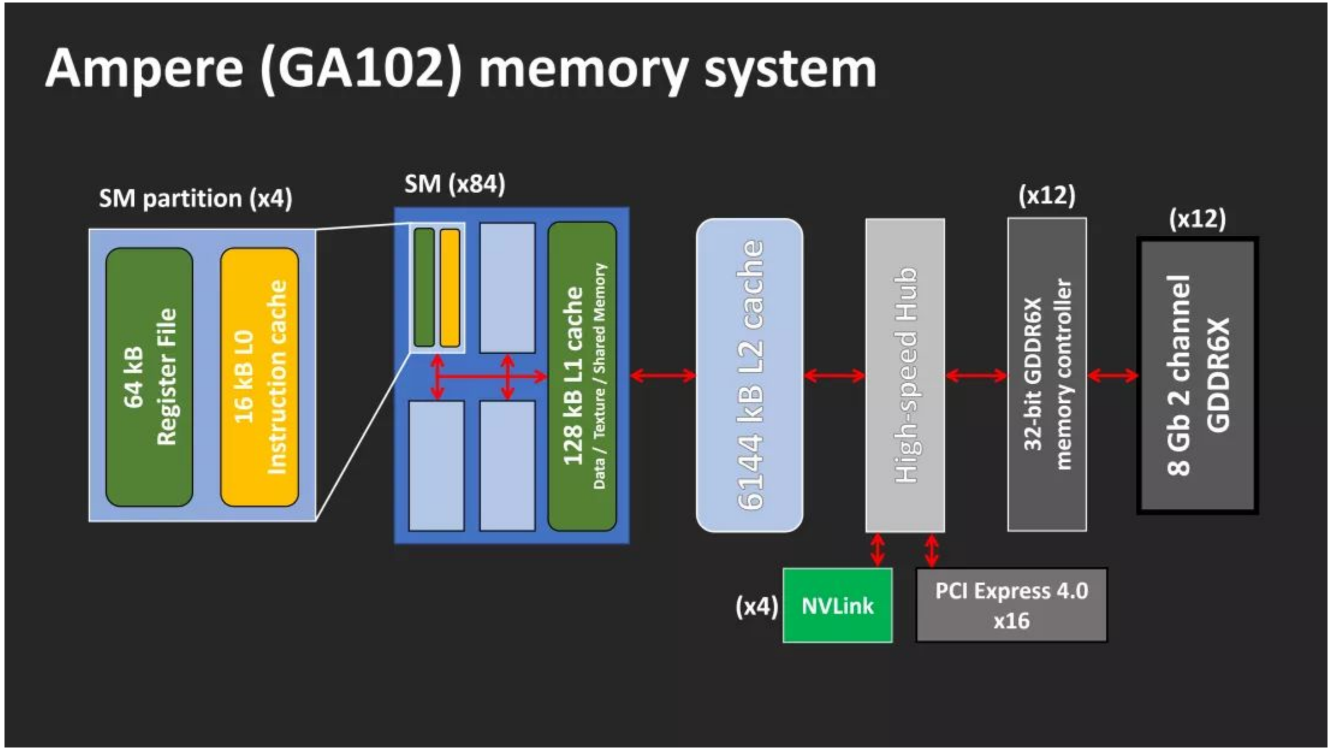


Figure 1: GPU memory hierarchy

[1]

Memory Hierarchy for Ampere



[1]

GDDR6x Memory

- GDDR6 transmitted 2 bits per clock cycles, 1 on the rising edge and 1 on the falling edge
- GDDR6x transmits 2 bits each clock edge, that is 4 bits per clock cycle
- Encoded using 4 different voltage levels sent on each clock edge
- GDDR6x can effectively double bandwidth as compared to GDDR6 at a given operating frequency

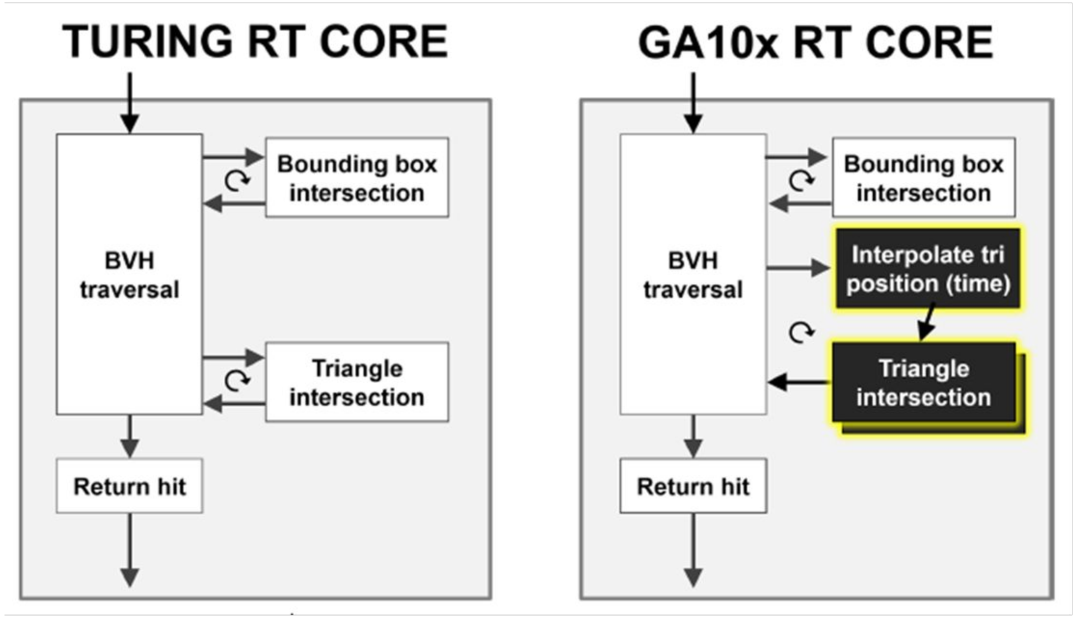
Unified shared memory and L1 data cache

- Unified architecture for shared memory, L1 data cache and texture cache, similar to Turing architecture
- Reconfigurable to allocate more memory to L1 cache or shared memory based on workload
- GA10x GPUs have double the L1 cache bandwidth as compared to Turing (128 bytes/clock per SM vs 64 bytes/clock)
- GA102 GPU has 10752 KB of L1 cache as compared to 6912 KB in TU102 GPU

Network

- Third Generation NVLink
  - Four x4 links with each link providing 14.0625 GB/s bandwidth between 2 GPUs
  - Four links provides 56.25 GB/s bandwidth in each direction, ie. 112.5 GB/s total bandwidth
- PCIe Gen 4
  - PCIe Gen 4 provides double the bandwidth as Gen 3 upto 16 GT/s
  - x16 PCIe 4.0 provides upto 64 GB/s peak bandwidth

Second-Generation Ray Tracing Engine in GA10x GPUs



Ampere Architecture Motion Blur Hardware Acceleration

$$\begin{tikzcd}\boxed{\text{SM}} \arrow[r, "\text{Cast}"] & \text{Ray}\end{tikzcd}$$

RT Core Perform all the calculations needed for BVH traversal and triangle intersection tests

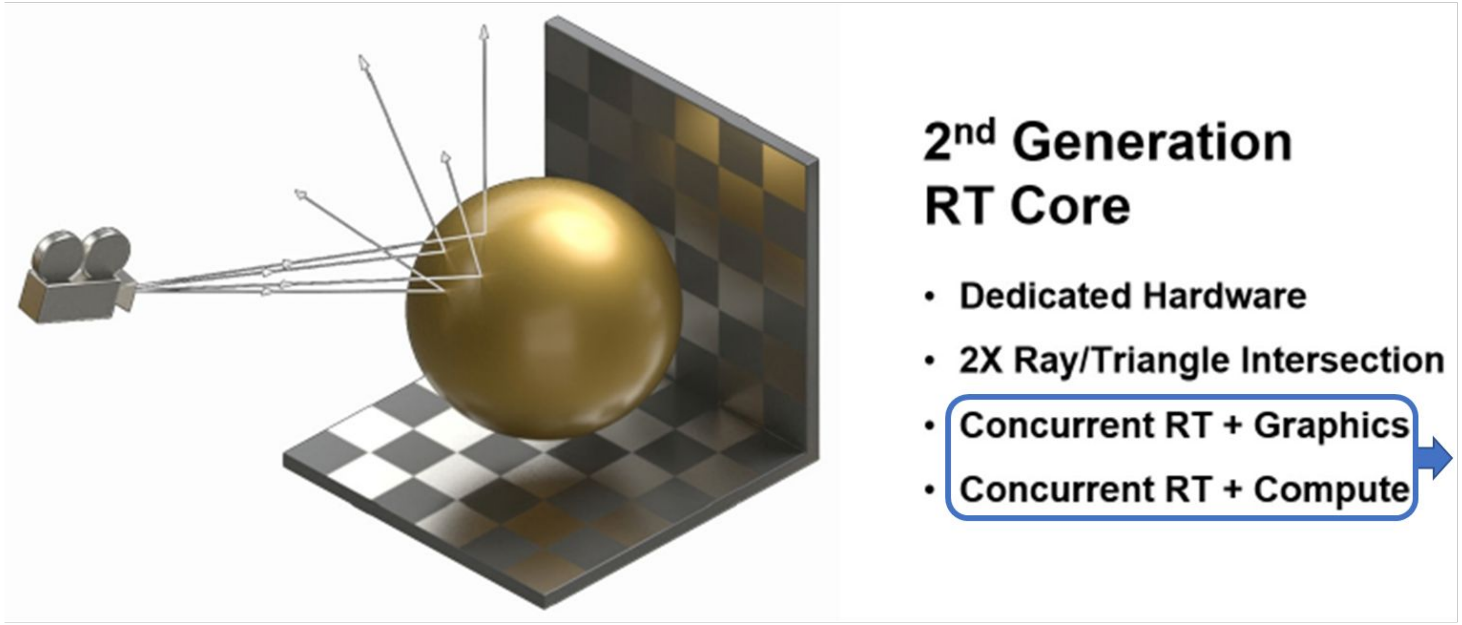
$$\begin{tikzcd}\boxed{\text{RT Core}} \arrow[r, "\text{Return}", "\text{Hit/No hit}"'] & \boxed{\text{SM}}\end{tikzcd}$$

MIMD Architecture

Table Ray Tracing Feature Comparison

	NVIDIA Turing Architecture (TU102 Full-Chip)	NVIDIA Ampere Architecture (GA102 Full-Chip)
Dedicated RT Cores	Yes (72 RT Cores)	Yes (84 RT Cores)
Ray / Bounding Box Acceleration	Yes	Yes
Ray / Triangle Acceleration	Yes	Yes
Tree Traversal Acceleration	Yes	Yes
Instance Transform Acceleration	Yes	Yes
Concurrent RT and Shading	No	Yes
Dedicated L1 Interface	Yes	Yes
Ray / Triangle Intersection Test Culling Rate Speedup	1.0x	2.0x
Overall GPU RT Speedup	1.0x	2.0x

Async Compute: a feature that allows the GPU to perform compute and graphics workloads simultaneously.

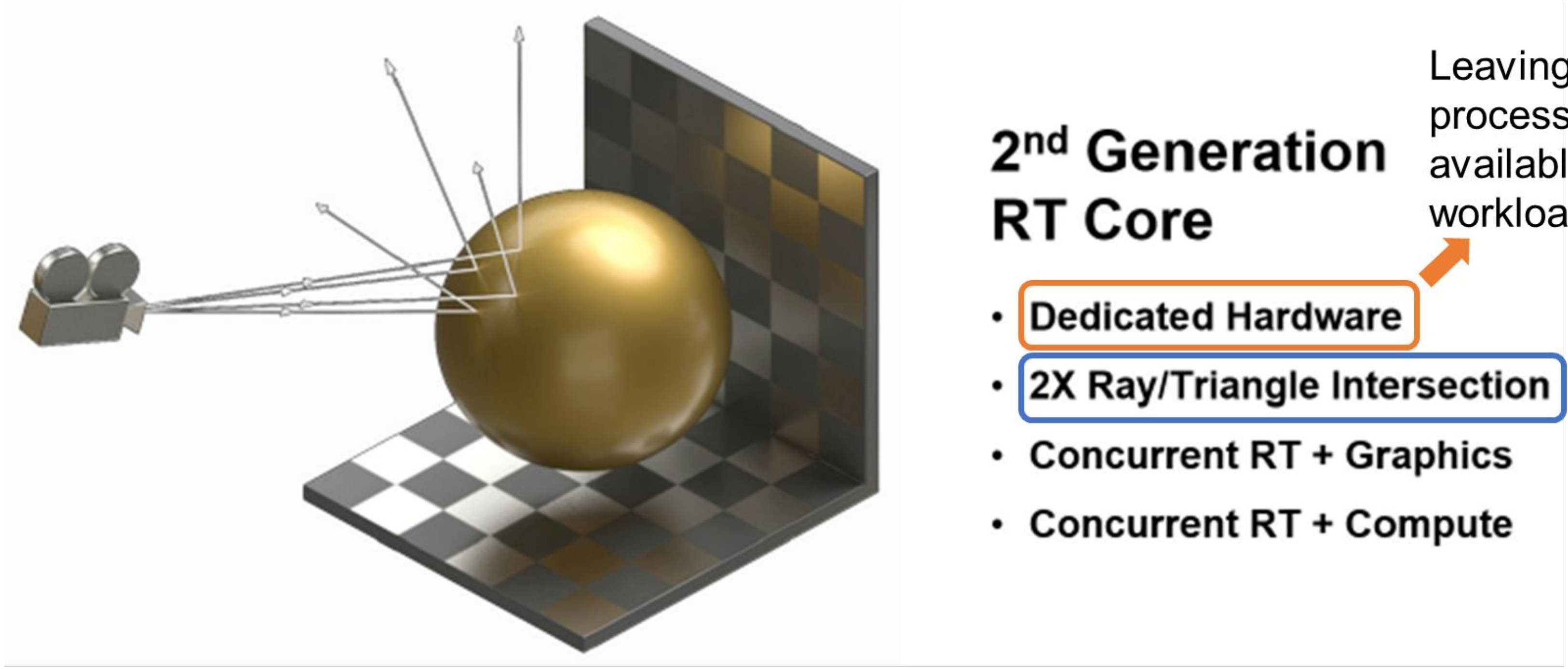


2nd Generation RT Core

- Dedicated Hardware
- 2X Ray/Triangle Intersection
- Concurrent RT + Graphics
- Concurrent RT + Compute

Also allow scenarios such as a compute-based denoising algorithm to run concurrently with RT Core-based ray tracing work

Second-Generation RT Core in GA10x GPUs

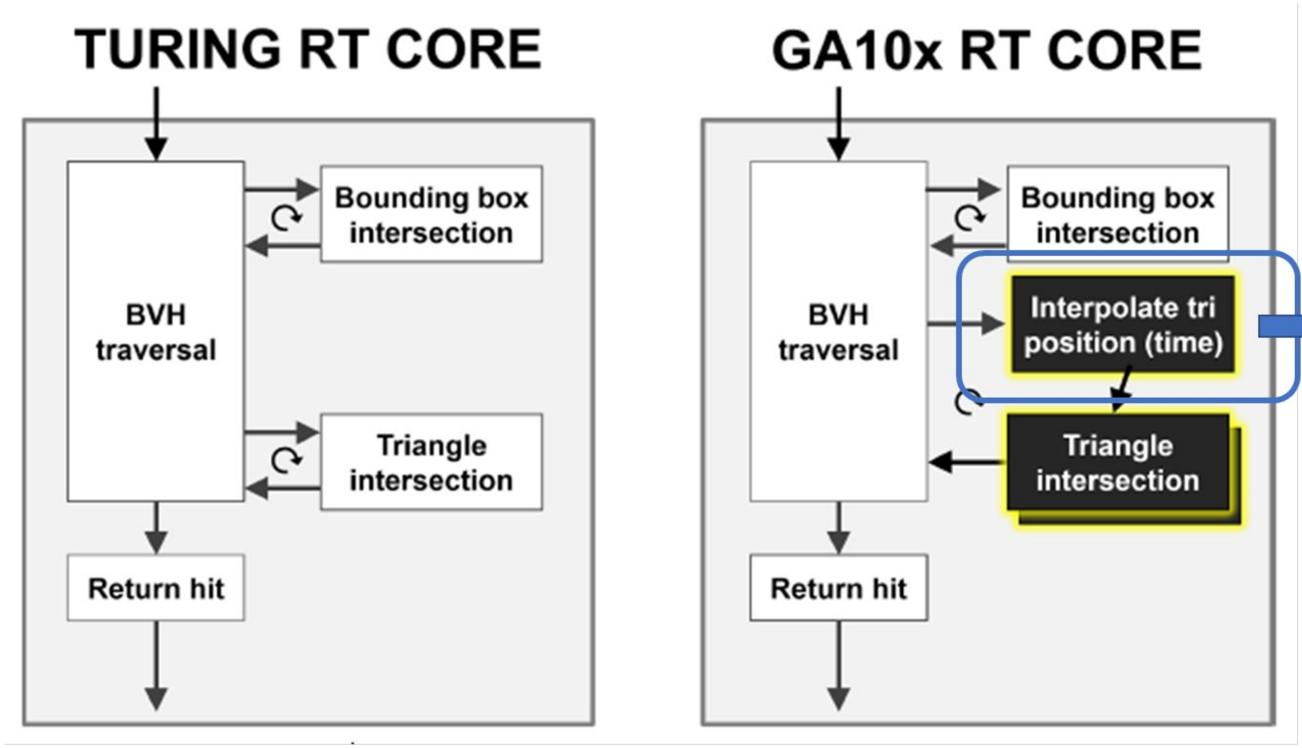


2nd Generation RT Core

- Dedicated Hardware
- 2X Ray/Triangle Intersection
- Concurrent RT + Graphics
- Concurrent RT + Compute

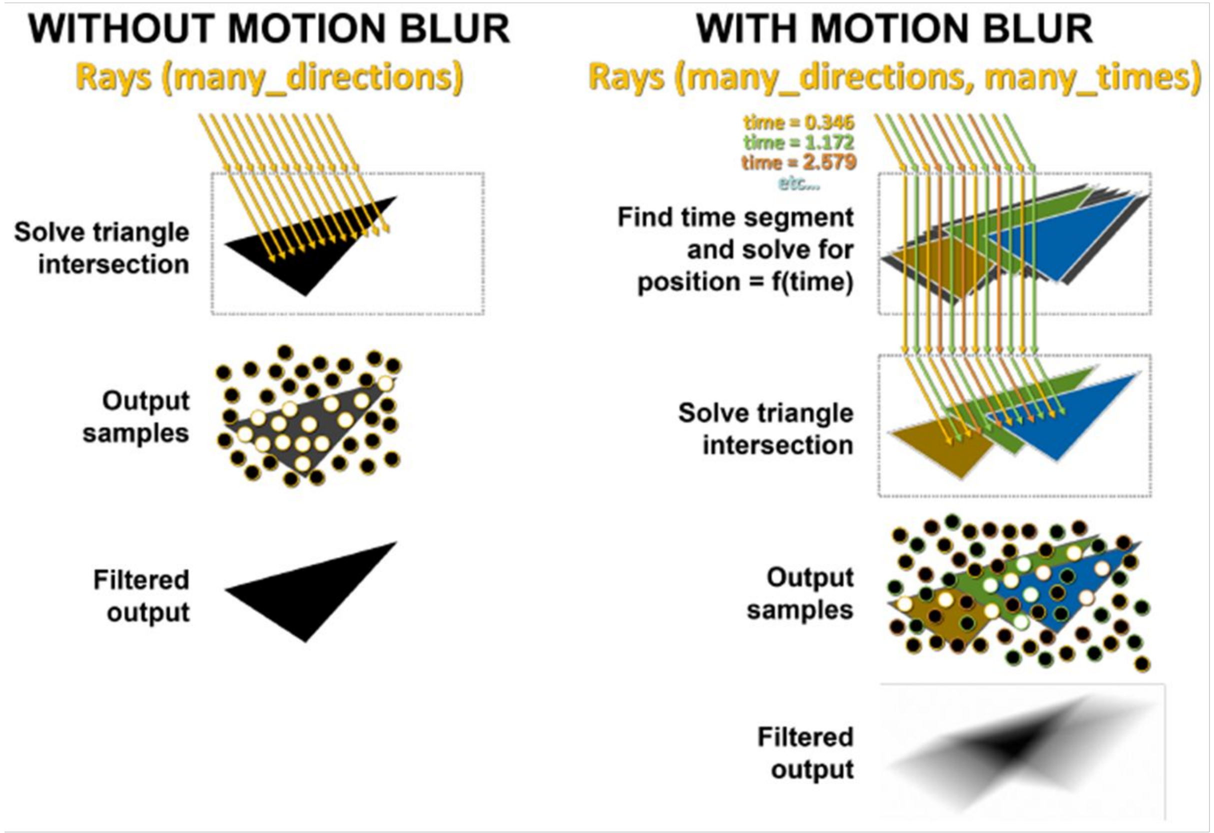
Leaving much SM processing power available for other workloads

Second-Generation RT Core in GA10x GPUs



Assist in ray-traced **motion blur** operations

Ampere Architecture Motion Blur Hardware Acceleration



Rendering Without vs With Motion Blur on GA10x

Interpolate Triangle Position unit

Generate triangles in the BVH in between existing triangle representations based on object motion

⬇

rays can intersect triangles at their expected positions in object space at the times specified by the ray timestamps

Allows accurate ray-traced motion blur rendering to occur up to **eight times** faster than the Turing GPU architecture.

Third-Generation Tensor Cores

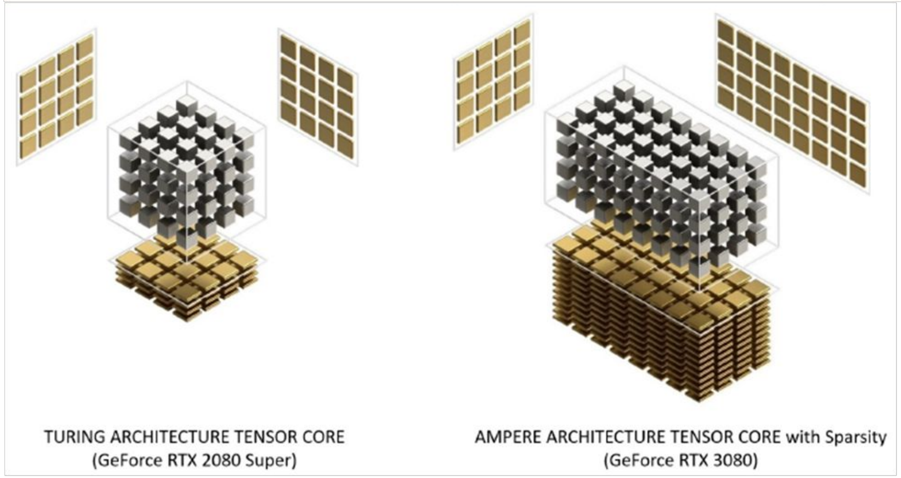
Specialized execution units designed specifically for performing the **tensor/matrix operations**

$\Big\downarrow$ Core Compute Function

Deep Learning

Table Comparison of NVIDIA Turing vs Ampere Architecture Tensor Core

	TU102 SM (RTX 2080 Super)	GA10x SM (RTX 3080)
GPU Architecture	NVIDIA Turing	NVIDIA Ampere
Tensor Cores per SM	8	4
FP16 FMA operations per Tensor Core	64	Dense: 128 Sparse: 256
Total FP16 FMA operations per SM	512	Dense: 512 Sparse: 1024


TURING ARCHITECTURE TENSOR CORE
(GeForce RTX 2080 Super)
AMPERE ARCHITECTURE TENSOR CORE with Sparsity
(GeForce RTX 3080)

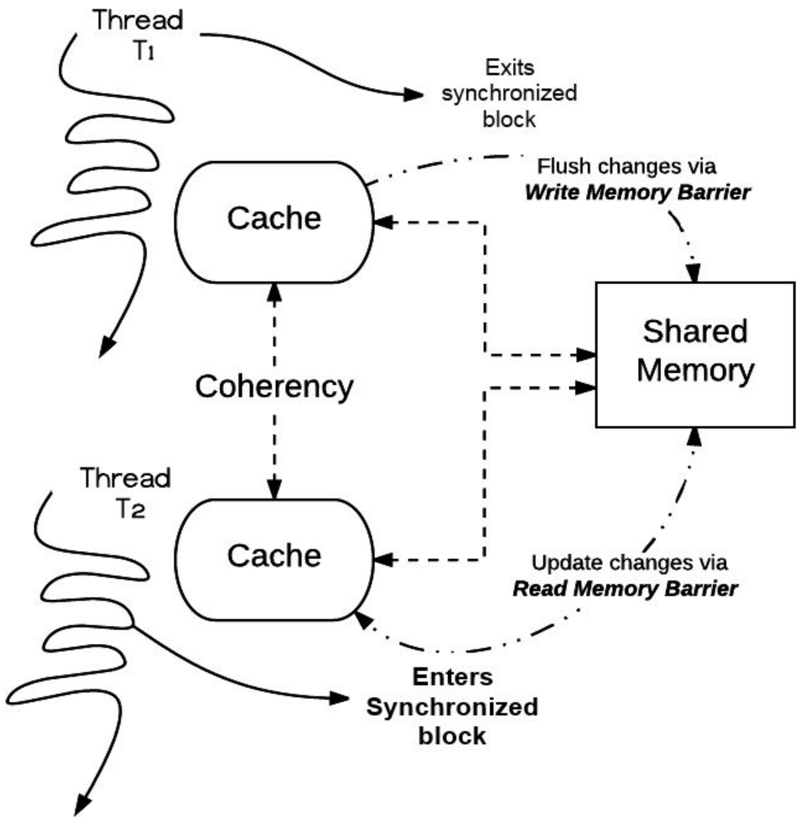
Ampere Architecture Tensor Core vs Turing Tensor Core

Each Ampere architecture SM:
Fewer Tensor Cores
More powerful Tensor Core

Third-generation Tensor Core

- Accelerates more data types
- Includes a new Sparsity feature
  - delivers up to a 2x speedup for matrix multiplication
- Increases raw performance and brings new precision modes such as TF32 and BFloat16

CUDA programming model


Thread T1
Exits synchronized block
Cache
Flush changes via *Write Memory Barrier*
Shared Memory
Coherency
Thread T2
Cache
Update changes via *Read Memory Barrier*
Enters Synchronized block

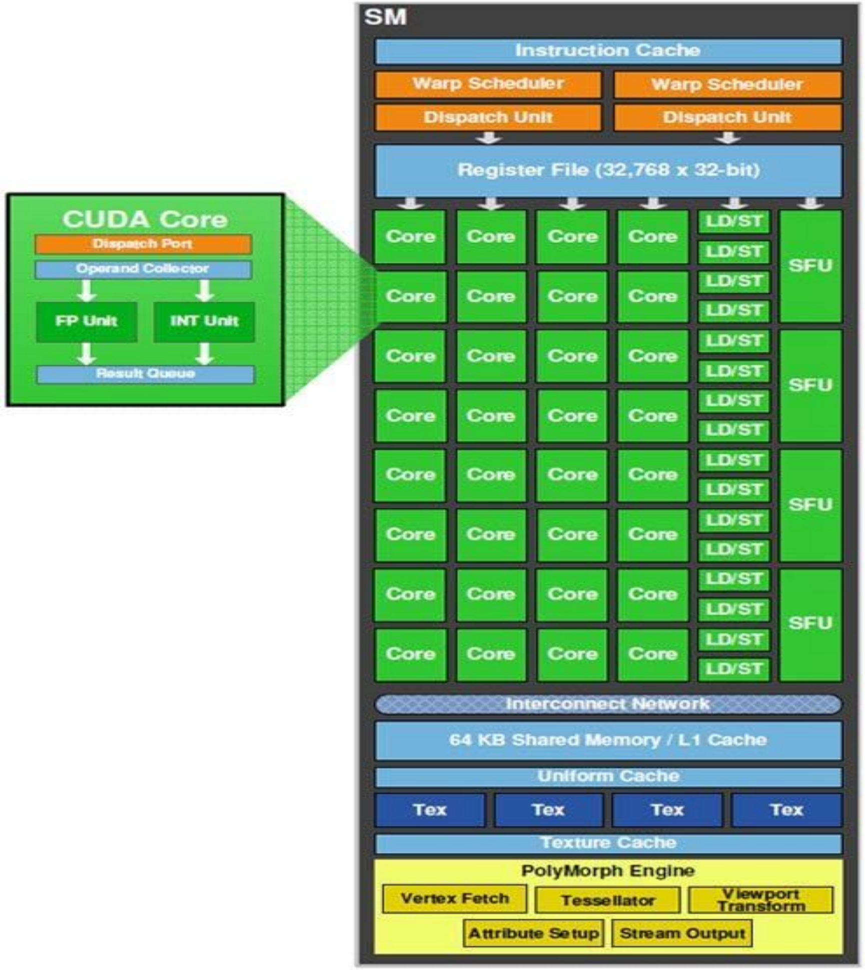
- **CUDA blocks**—A collection or group of threads.
- **Shared memory**—Memory shared within a block among all threads.
- **Synchronization barriers**—Enable multiple threads to wait until all threads have reached a particular point of execution before any thread continues.

* Easy to scale

[1]

CUDA Cores

- Nvidia's parallel processing platform: CUDA: Ampere GA102 has 10,752 CUDA cores
- CUDA Cores are the processing units inside a GPU→AMD's Stream Processors.
- Effect on the Performance?
  - "More number of CUDA cores → More is the processing speed"?
  - For example: Number of CUDA cores for:
    - GTX 980: 2816
    - GTX 1080: 2560

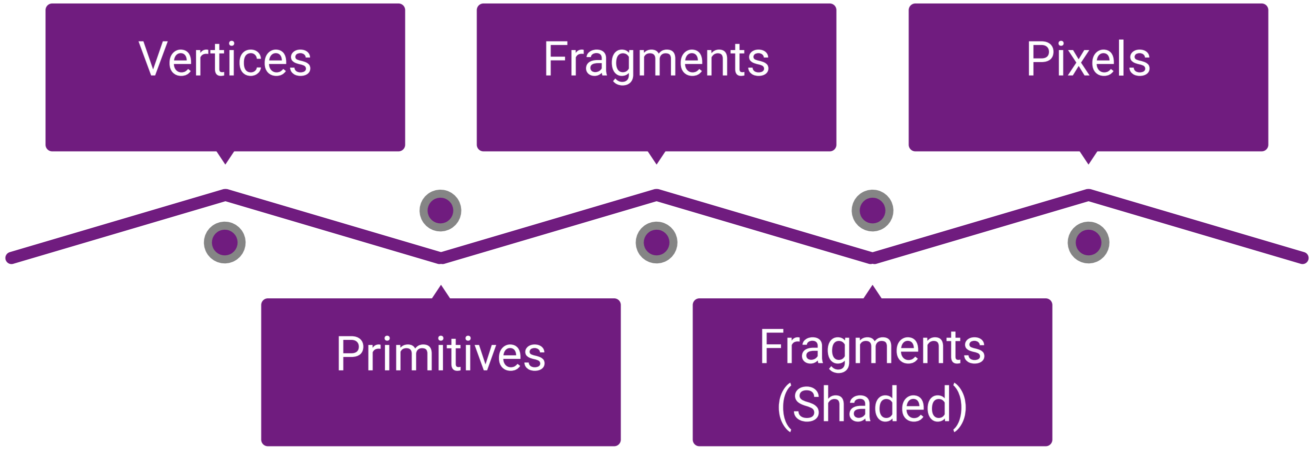


Streaming Multiprocessor (SM)

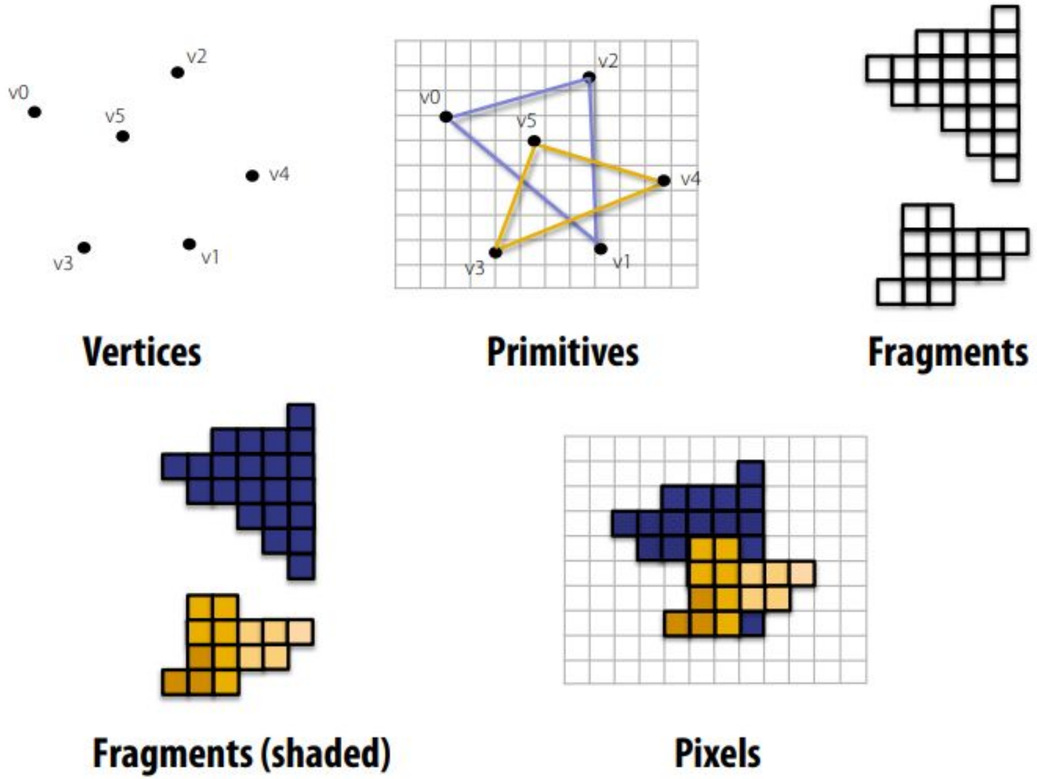
[2]

Programmable Shading Cores

- Consists of NVIDIA CUDA Cores
- Shading units are small processors within the graphics card - responsible for processing aspects of the image.
- Have dedicated units for different types of operations in the rendering pipeline
- GA102 GPU incorporates 10752 CUDA Cores, 84 second-generation RT Cores, and 336 third-generation Tensor Cores

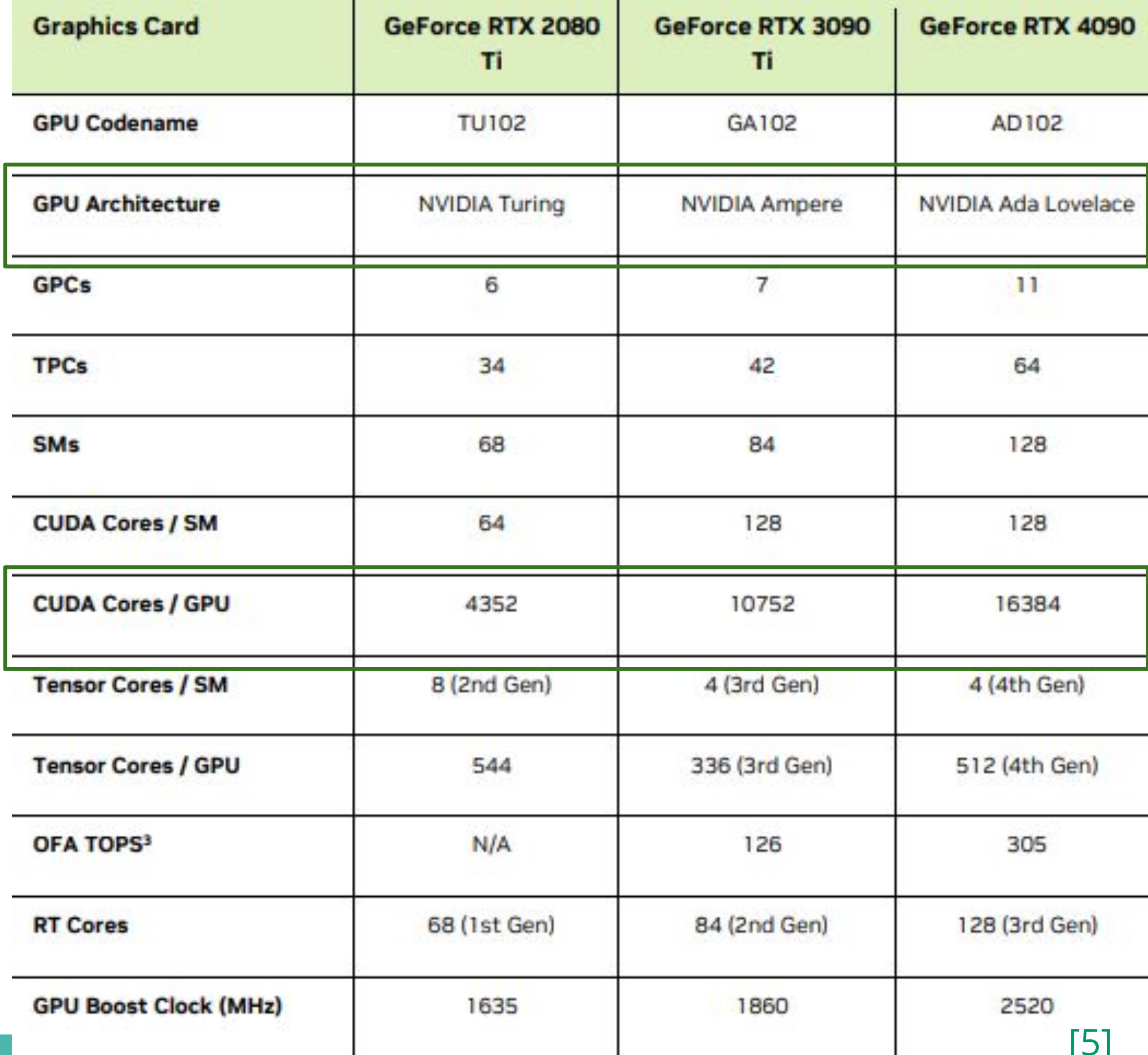


Pipeline Entities



More shading units a graphics card has → faster power allocation to process the workload

[4]



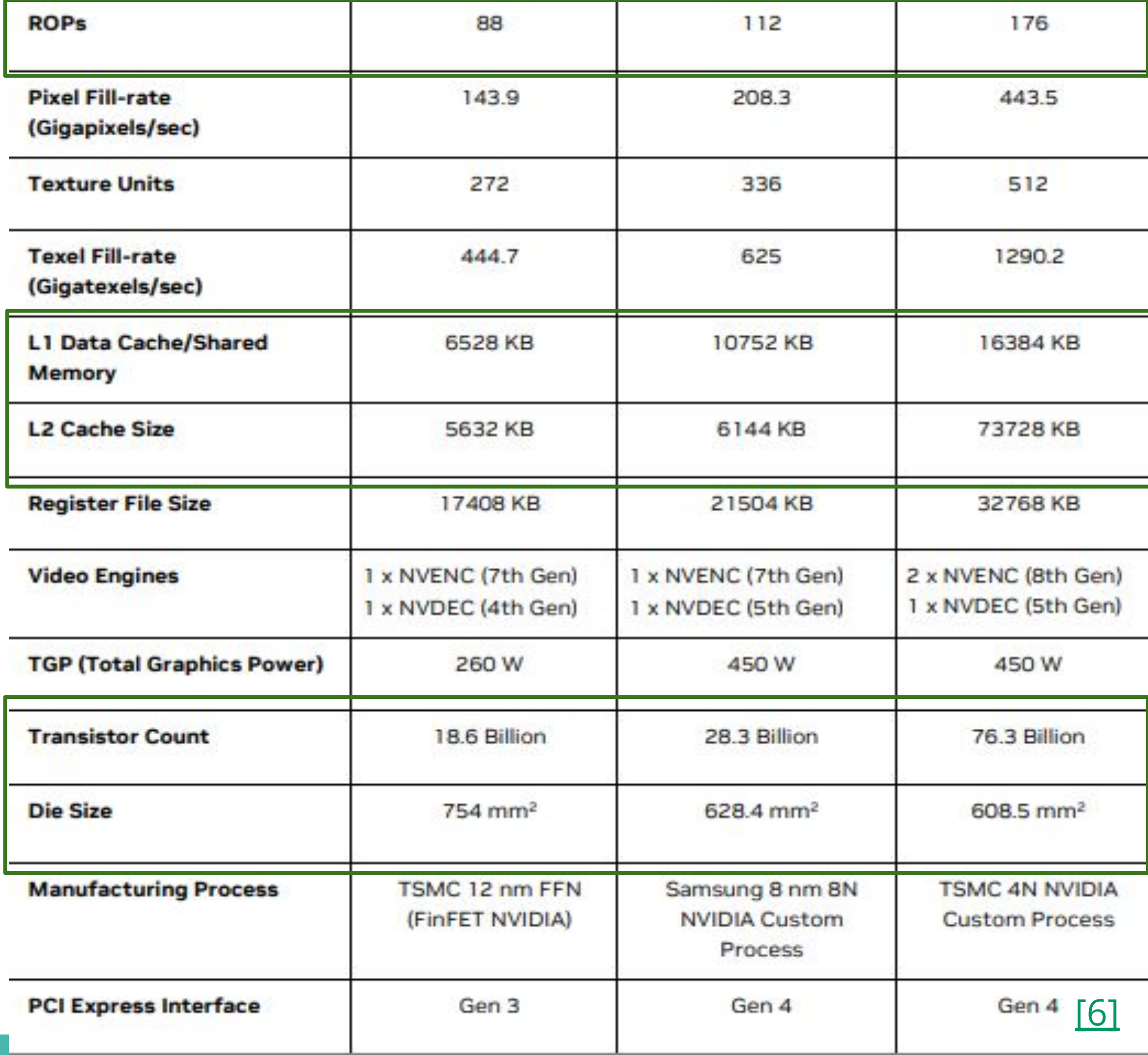
Graphics Card	GeForce RTX 2080 Ti	GeForce RTX 3090 Ti	GeForce RTX 4090
GPU Codename	TU102	GA102	AD102
GPU Architecture	NVIDIA Turing	NVIDIA Ampere	NVIDIA Ada Lovelace
GPCs	6	7	11
TPCs	34	42	64
SMs	68	84	128
CUDA Cores / SM	64	128	128
CUDA Cores / GPU	4352	10752	16384
Tensor Cores / SM	8 (2nd Gen)	4 (3rd Gen)	4 (4th Gen)
Tensor Cores / GPU	544	336 (3rd Gen)	512 (4th Gen)
OFA TOPS[3]	N/A	126	305
RT Cores	68 (1st Gen)	84 (2nd Gen)	128 (3rd Gen)
GPU Boost Clock (MHz)	1635	1860	2520

ADA vs Ampere vs Turing

[5]

ADA vs Ampere vs Turing

- 2x GPCs (Versus Ampere)
- 50% More Cores (Versus Ampere)
- 50% More L1 Cache (Versus Ampere)
- 16x More L2 Cache (Versus Ampere)
- Double The ROPs (Versus Ampere)
- 4th Gen Tensor & 3rd Gen RT Cores



ROPs	88	112	176
Pixel Fill-rate (Gigapixels/sec)	143.9	208.3	443.5
Texture Units	272	336	512
Texel Fill-rate (Gigatexels/sec)	444.7	625	1290.2
L1 Data Cache/Shared Memory	6528 KB	10752 KB	16384 KB
L2 Cache Size	5632 KB	6144 KB	73728 KB
Register File Size	17408 KB	21504 KB	32768 KB
Video Engines	1 x NVENC (7th Gen) 1 x NVDEC (4th Gen)	1 x NVENC (7th Gen) 1 x NVDEC (5th Gen)	2 x NVENC (8th Gen) 1 x NVDEC (5th Gen)
TGP (Total Graphics Power)	260 W	450 W	450 W
Transistor Count	18.6 Billion	28.3 Billion	76.3 Billion
Die Size	754 mm²	628.4 mm²	608.5 mm²
Manufacturing Process	TSMC 12 nm FFN (FinFET NVIDIA)	Samsung 8 nm 8N NVIDIA Custom Process	TSMC 4N NVIDIA Custom Process
PCI Express Interface	Gen 3	Gen 4	Gen 4

[6]

References

1. https://images.nvidia.com/aem-dam/en-zz/Solutions/geforce/ampere/pdf/NVIDIA-ampere-GA102-GPU-Architecture-Whitepaper-V1.pdf
2. https://images.nvidia.com/aem-dam/Solutions/geforce/ada/nvidia-ada-gpu-architecture.pdf
3. https://developer.nvidia.com/blog/nvidia-ampere-architecture-in-depth/
4. https://images.nvidia.com/aem-dam/Solutions/geforce/ada/nvidia-ada-gpu-architecture.pdf

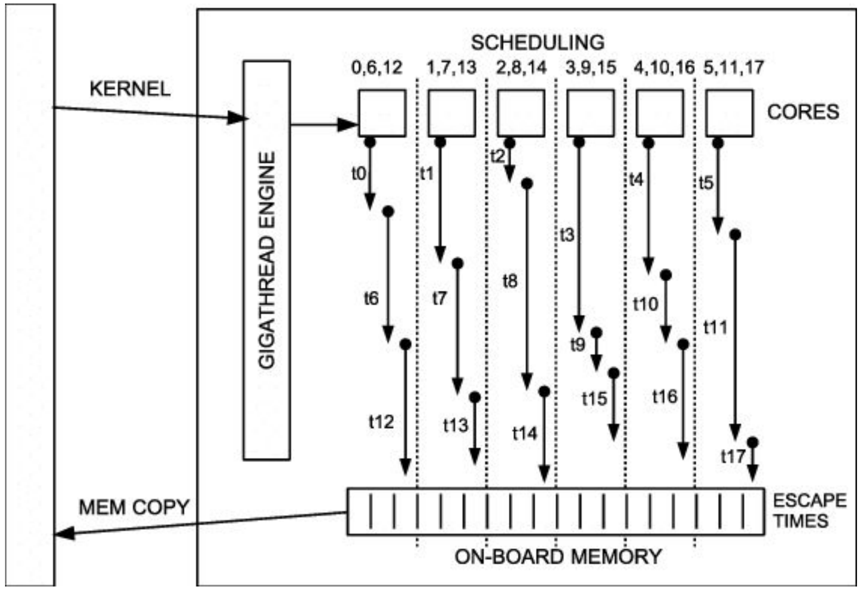
Backup Slides

GigaThread Engine

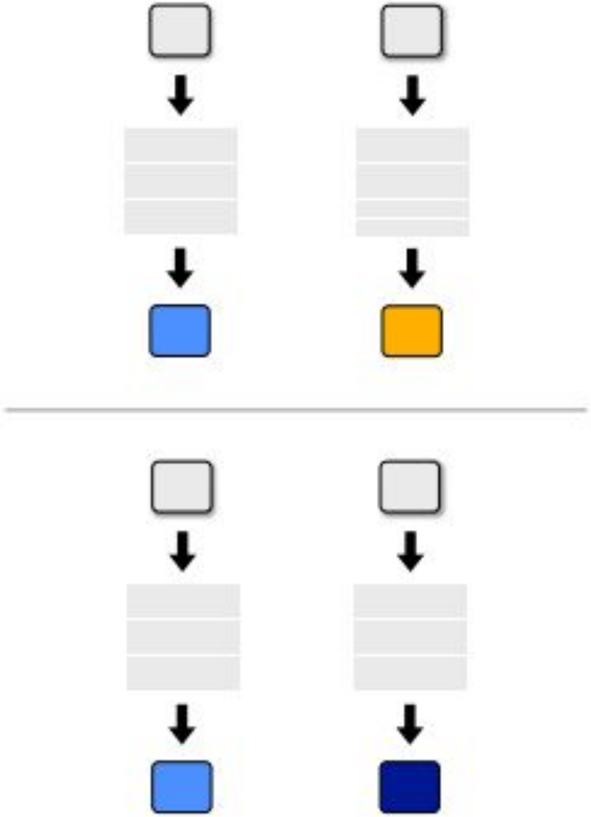
Scheduler and management cores that live inside of the GPU

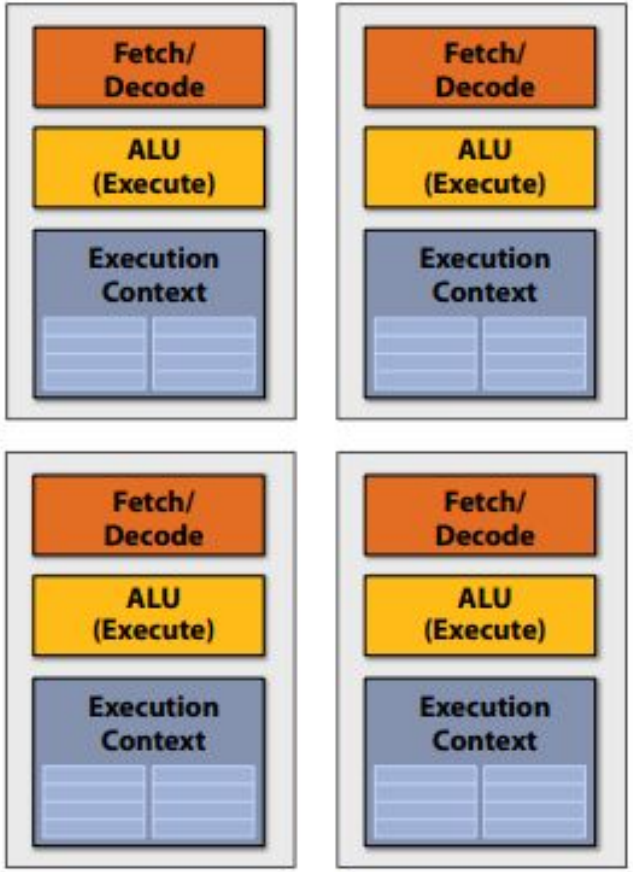
Incorporates central graphics work scheduling, handing the work over to other subsections of the GigaThread Engine and finally communicating with the compute cores themselves.

This main core essentially runs an extremely basic "operating system".

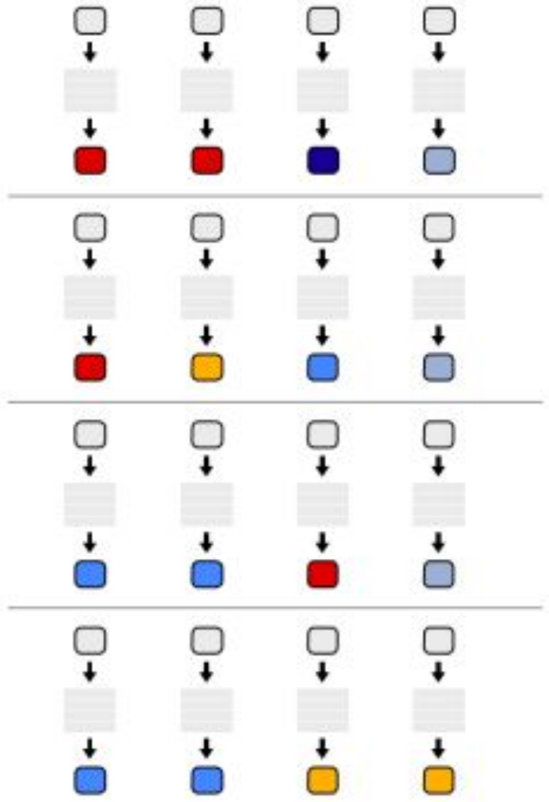


Four cores (four fragments in parallel)

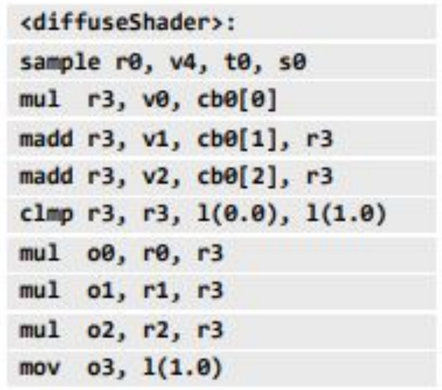




Instruction stream sharing

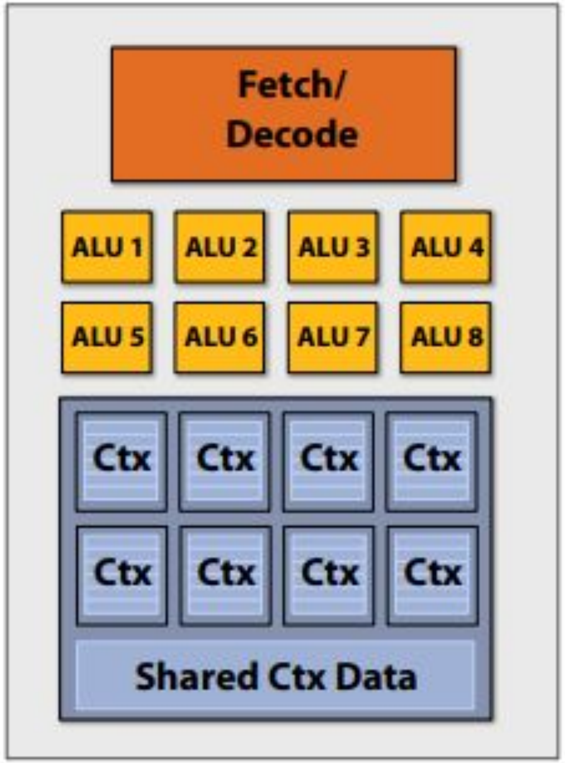


But ... many fragments should be able to share an instruction stream!



```
<diffuseShader>:
sample r0, v4, t0, s0
mul  r3, v0, cb0[0]
madd r3, v1, cb0[1], r3
madd r3, v2, cb0[2], r3
clmp r3, r3, l(0.0), l(1.0)
mul  o0, r0, r3
mul  o1, r1, r3
mul  o2, r2, r3
mov  o3, l(1.0)
```

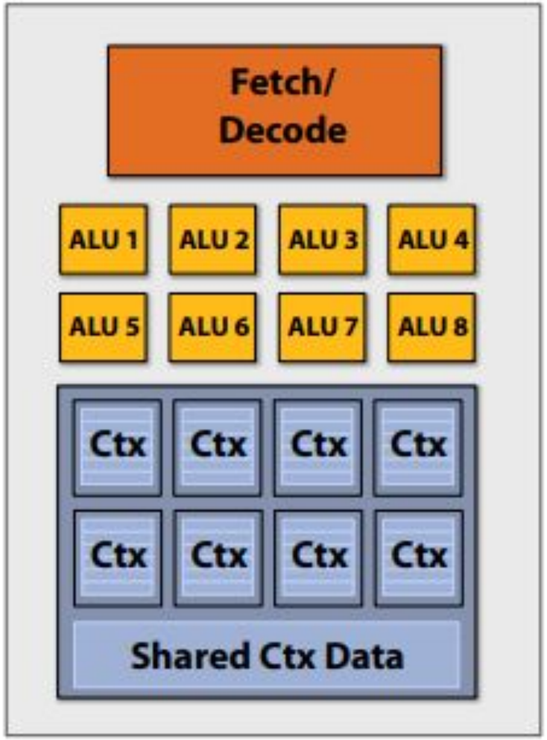
Add ALUs

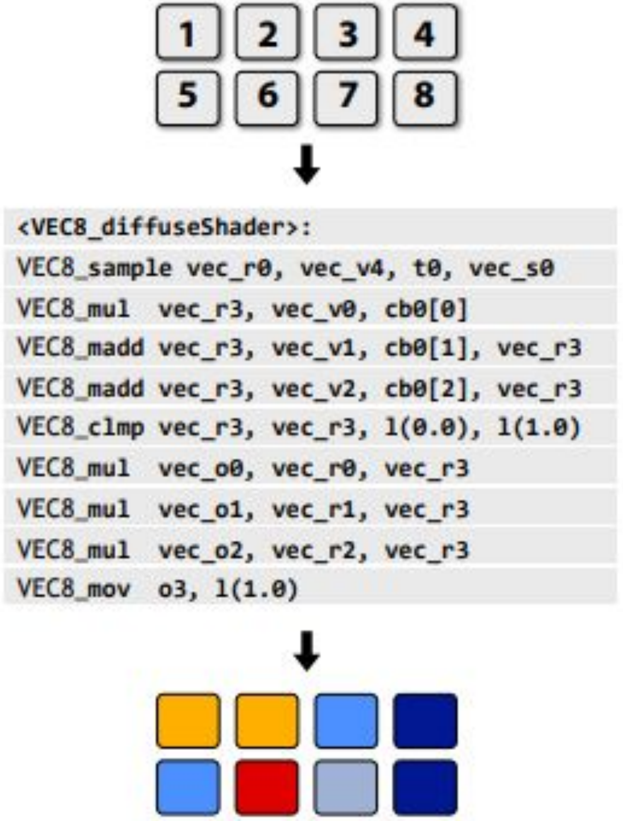


Idea #2:
Amortize cost/complexity of managing an instruction stream across many ALUs

SIMD processing

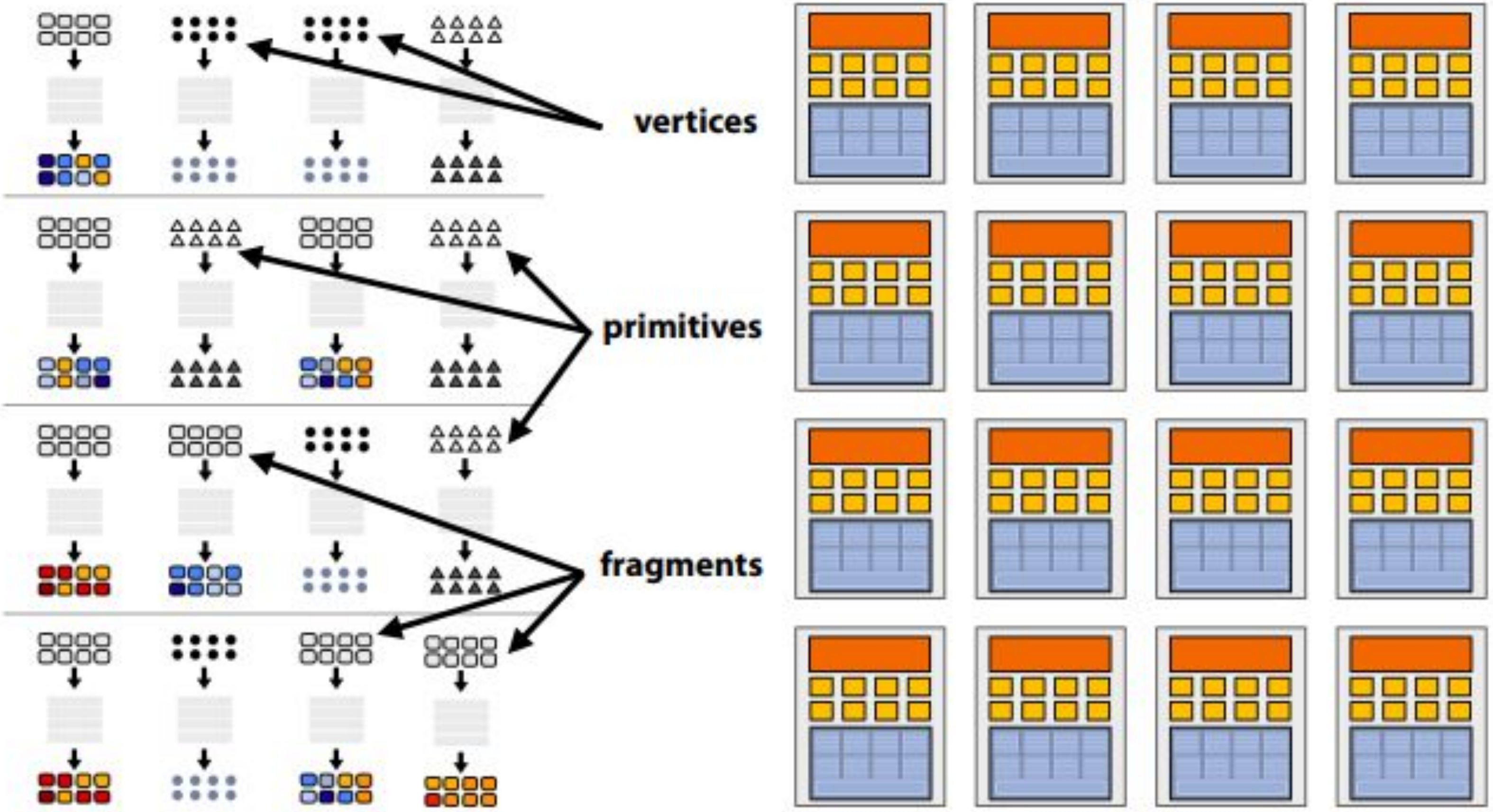
Modifying the shader





```
<VEC8_diffuseShader>:
VEC8_sample vec_r0, vec_v4, t0, vec_s0
VEC8_mul  vec_r3, vec_v0, cb0[0]
VEC8_madd vec_r3, vec_v1, cb0[1], vec_r3
VEC8_madd vec_r3, vec_v2, cb0[2], vec_r3
VEC8_clmp vec_r3, vec_r3, l(0.0), l(1.0)
VEC8_mul  vec_o0, vec_r0, vec_r3
VEC8_mul  vec_o1, vec_r1, vec_r3
VEC8_mul  vec_o2, vec_r2, vec_r3
VEC8_mov  o3, l(1.0)
```

128 [vertices/fragments primitives OpenCL work items CUDA threads] in parallel


vertices
primitives
fragments