

CS 380 - GPU and GPGPU Programming

Lecture 7: GPU Architecture, Pt. 4

Markus Hadwiger, KAUST

Reading Assignment #4 (until Sep 27)



Read (required):

- Read https://en.wikipedia.org/wiki/Instruction_pipelining
- Get an overview of NVIDIA Ampere (GA102) white paper:

<https://www.nvidia.com/content/PDF/nvidia-ampere-ga-102-gpu-architecture-whitepaper-v2.pdf>

- Get an overview of NVIDIA Ampere (A100) Tensor Core GPU white paper:

<https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/nvidia-ampere-architecture-whitepaper.pdf>

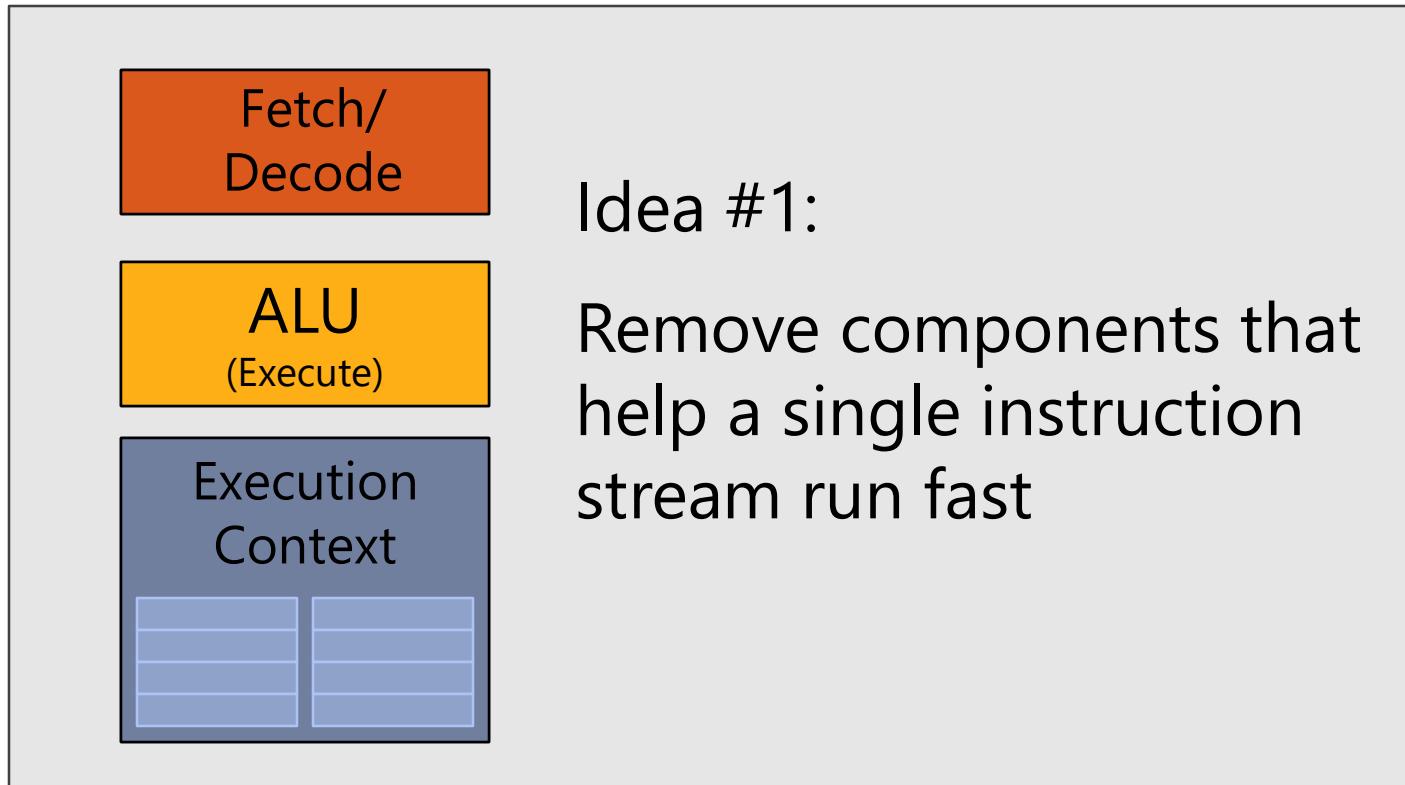
Where this is going...



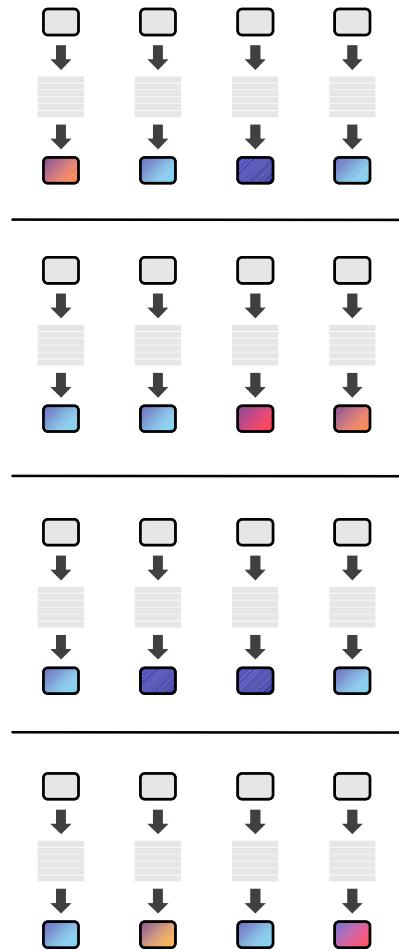
Summary: three key ideas for high-throughput execution

- 1. Use many “slimmed down cores,” run them in parallel**
- 2. Pack cores full of ALUs (by sharing instruction stream overhead across groups of fragments)**
 - Option 1: Explicit SIMD vector instructions**
 - Option 2: Implicit sharing managed by hardware**
- 3. Avoid latency stalls by interleaving execution of many groups of fragments**
 - When one group stalls, work on another group**

Idea #1: Slim down

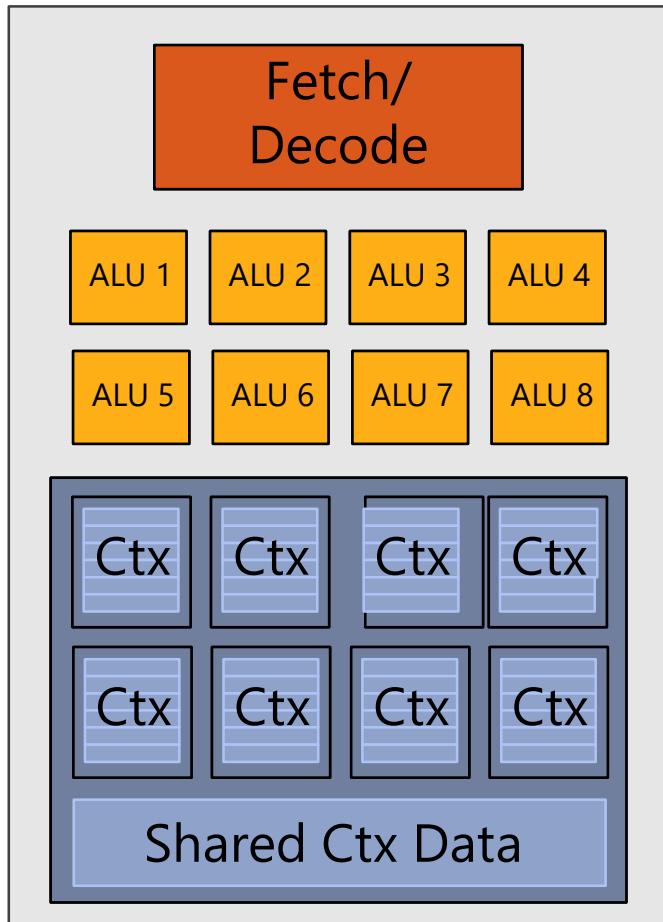


Sixteen cores (sixteen fragments in parallel)



16 cores = 16 simultaneous instruction streams

Idea #2: Add ALUs



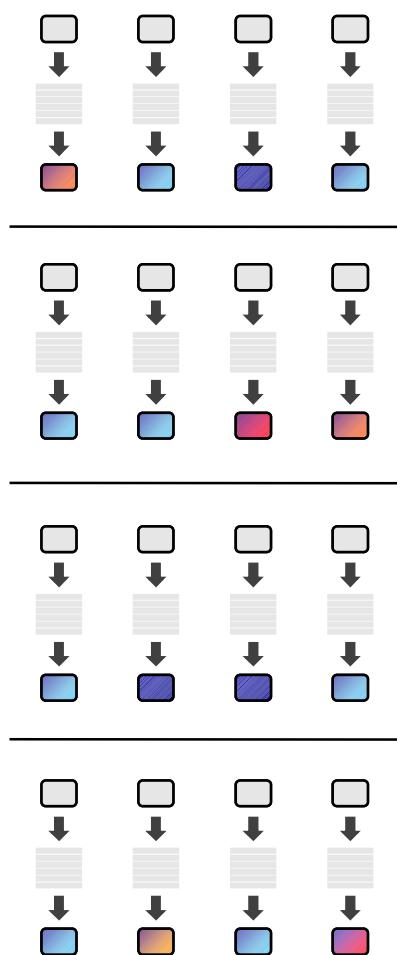
Idea #2:

Amortize cost/complexity of managing an instruction stream across many ALUs

SIMD processing

(or SIMT, SPMD)

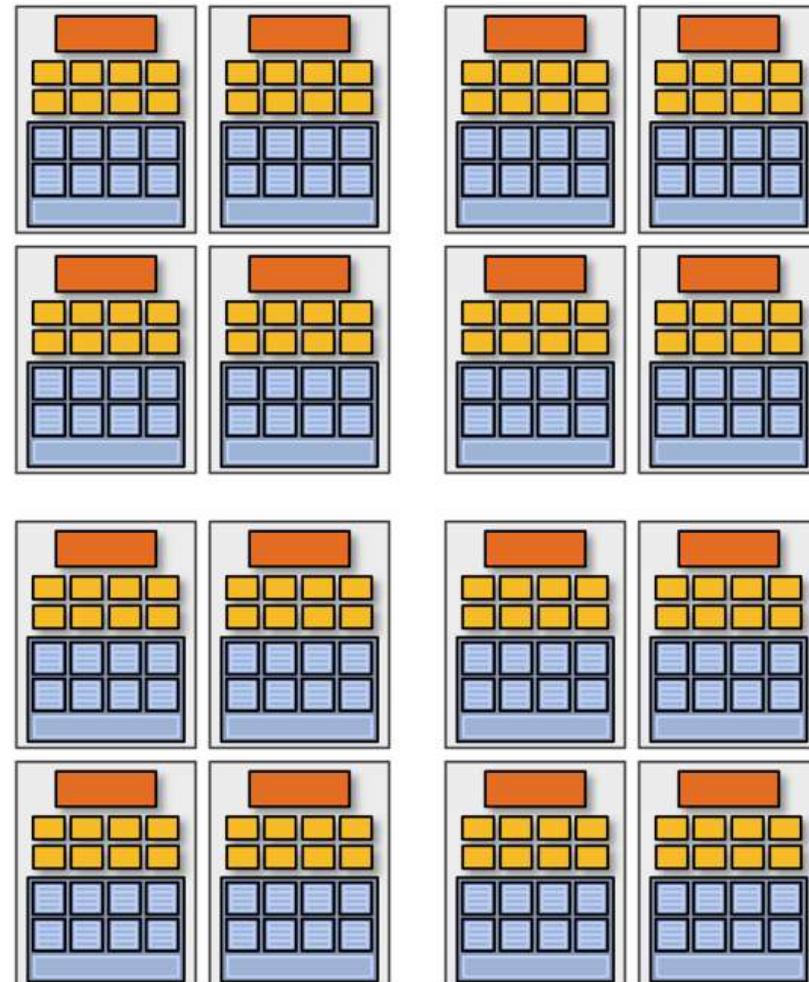
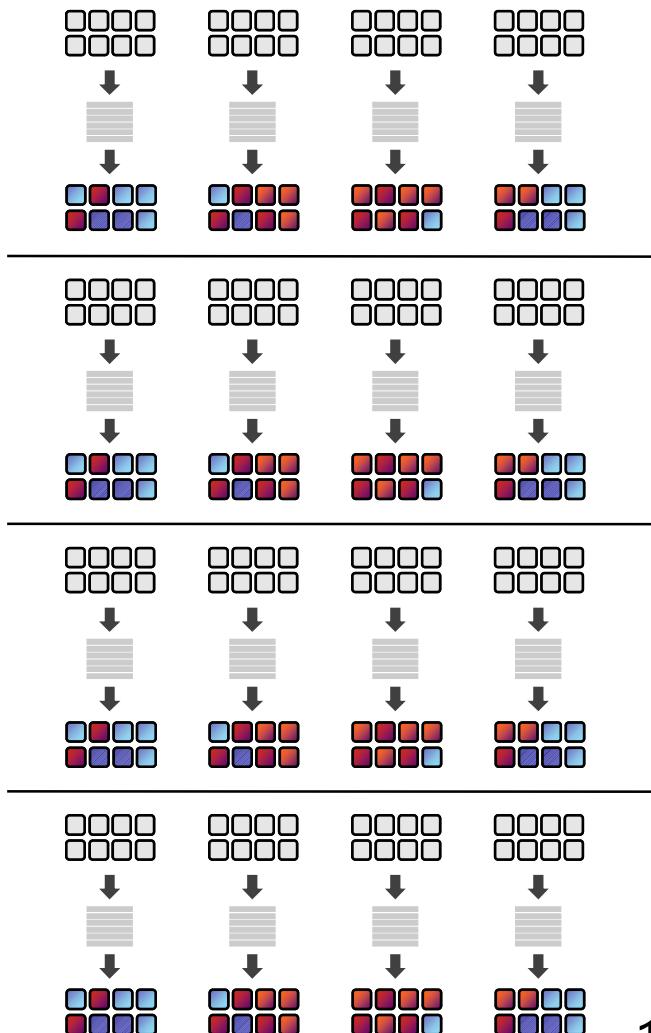
Instruction stream sharing



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, 1(0.0), 1(1.0)  
mul o0, r0, r3  
mul o1, r1, r3  
mul o2, r2, r3  
mov o3, 1(1.0)
```

128 fragments in parallel



16 cores = 128 ALUs
= 16 simultaneous instruction streams

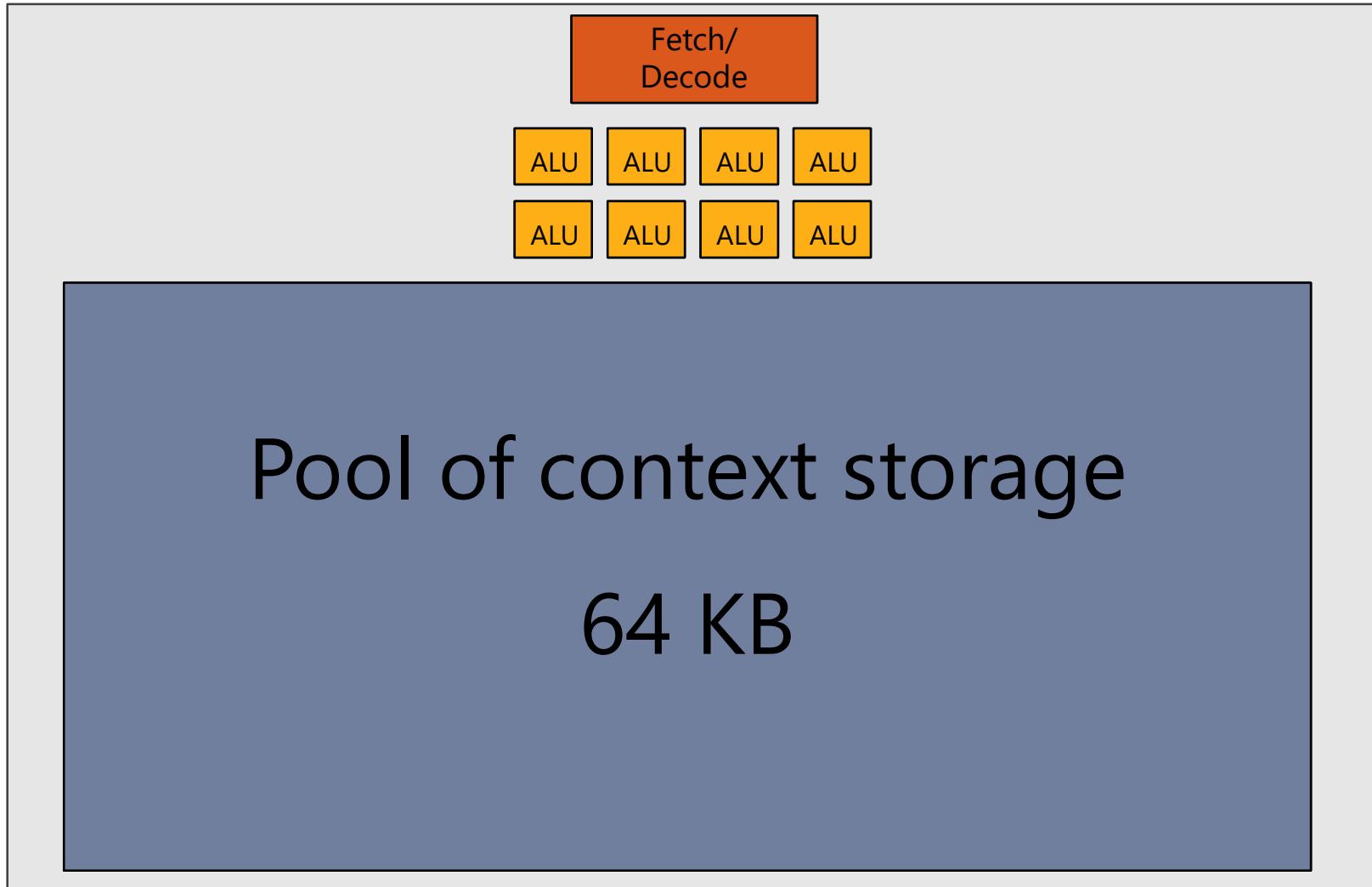
Idea #3: Interleave execution of groups

But we have **LOTS** of independent fragments.

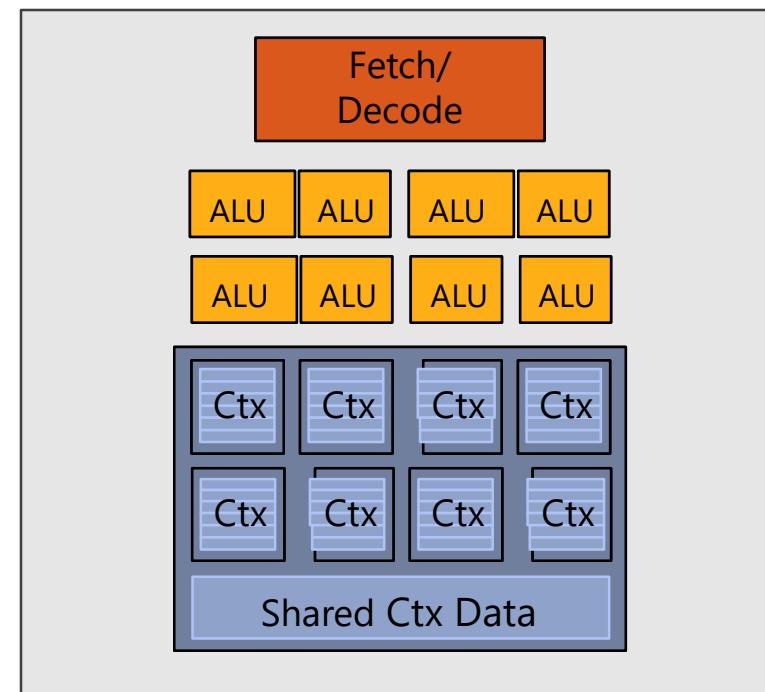
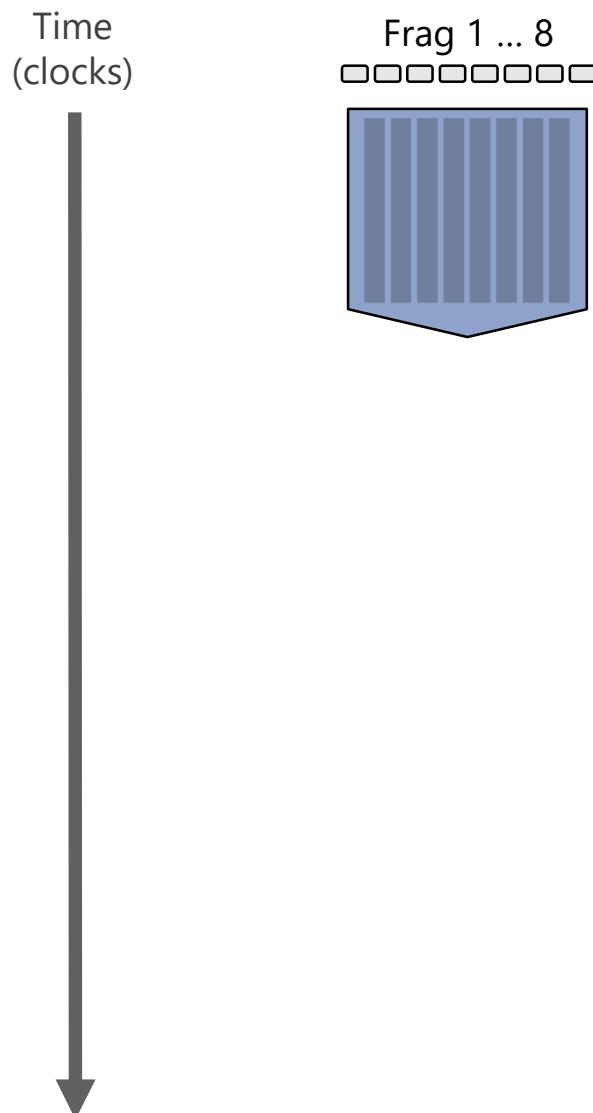
Idea #3:

Interleave processing of many fragments on a single core
to avoid stalls caused by high latency operations.

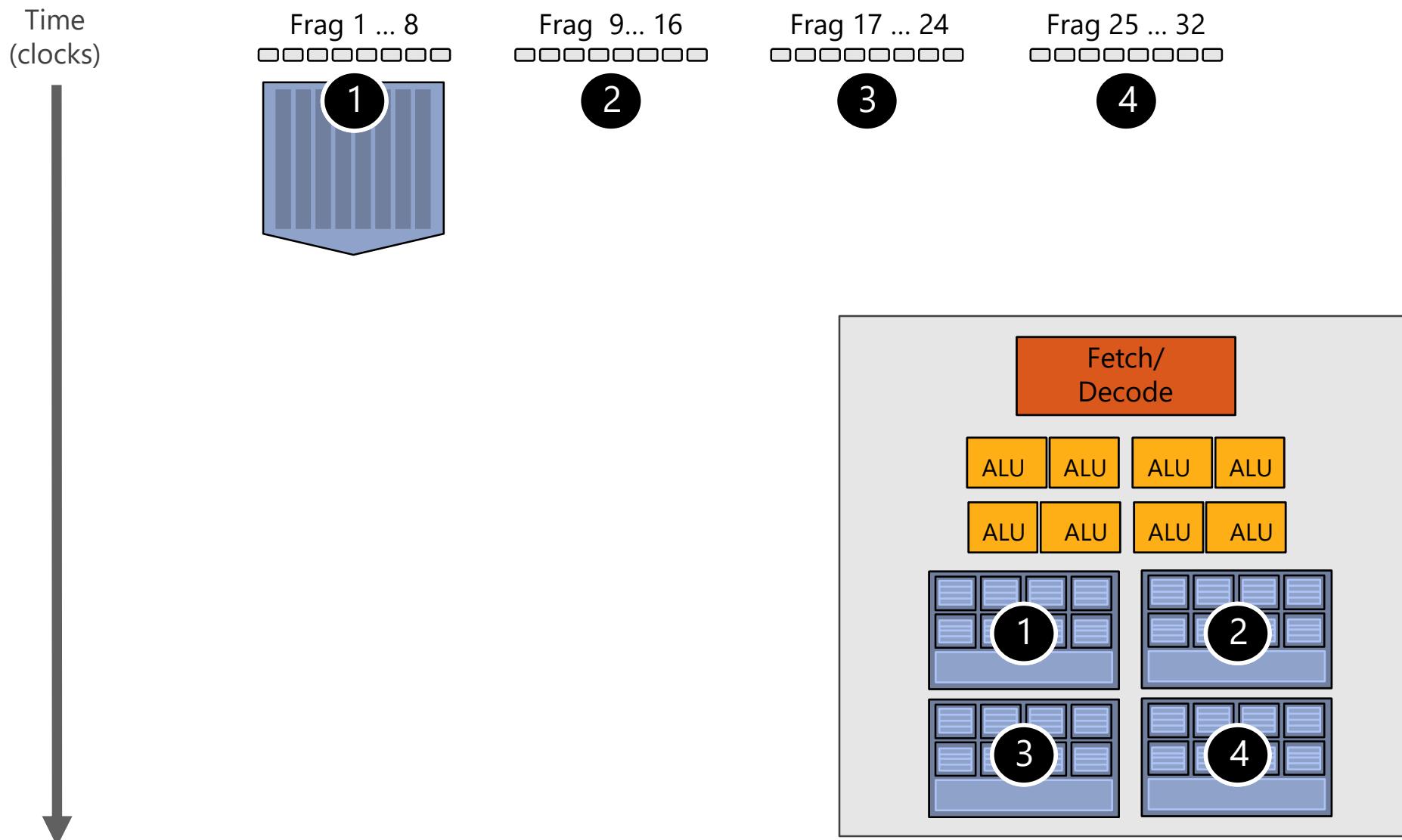
Idea #3: Store multiple group contexts



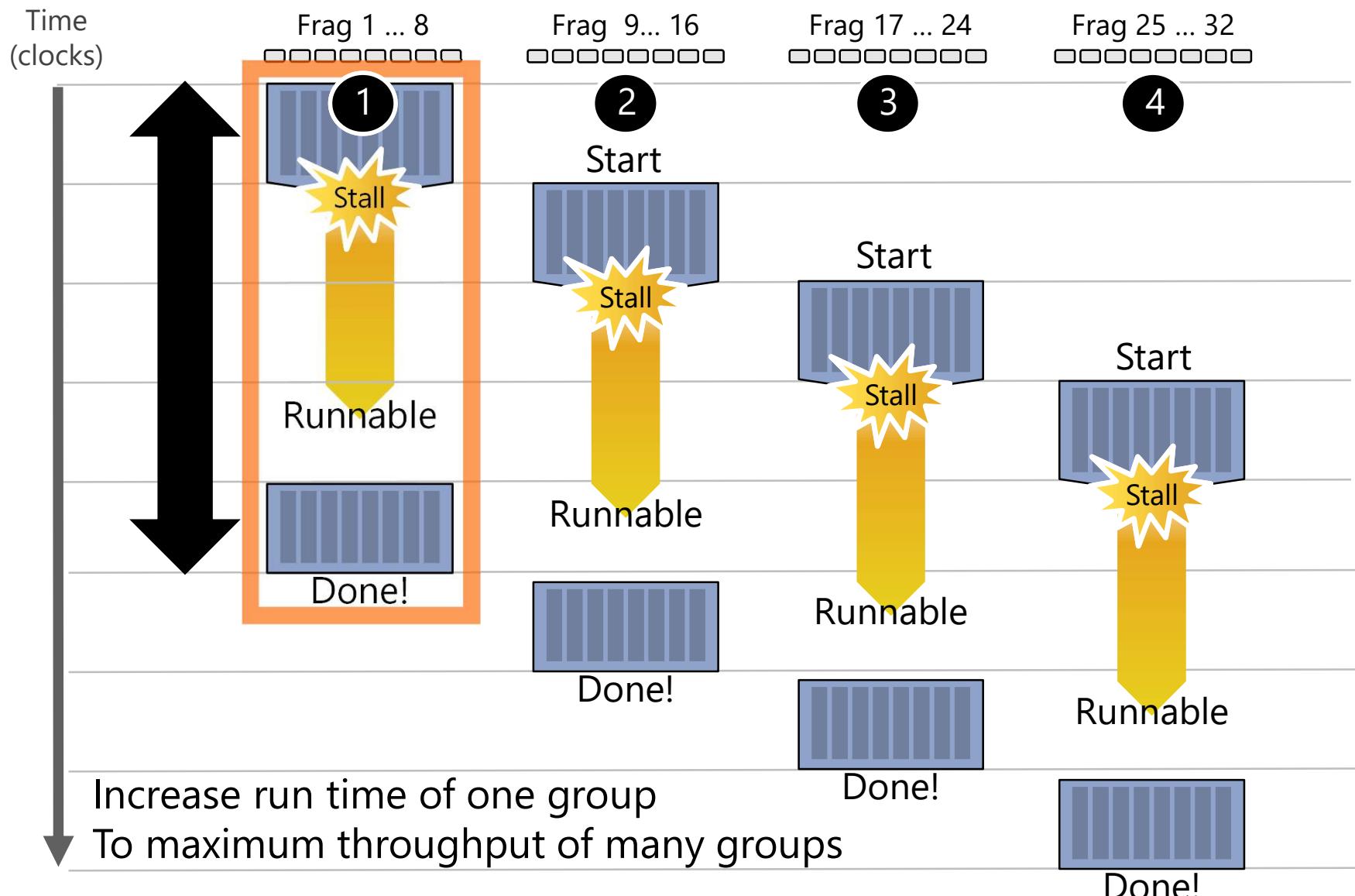
Hiding shader stalls



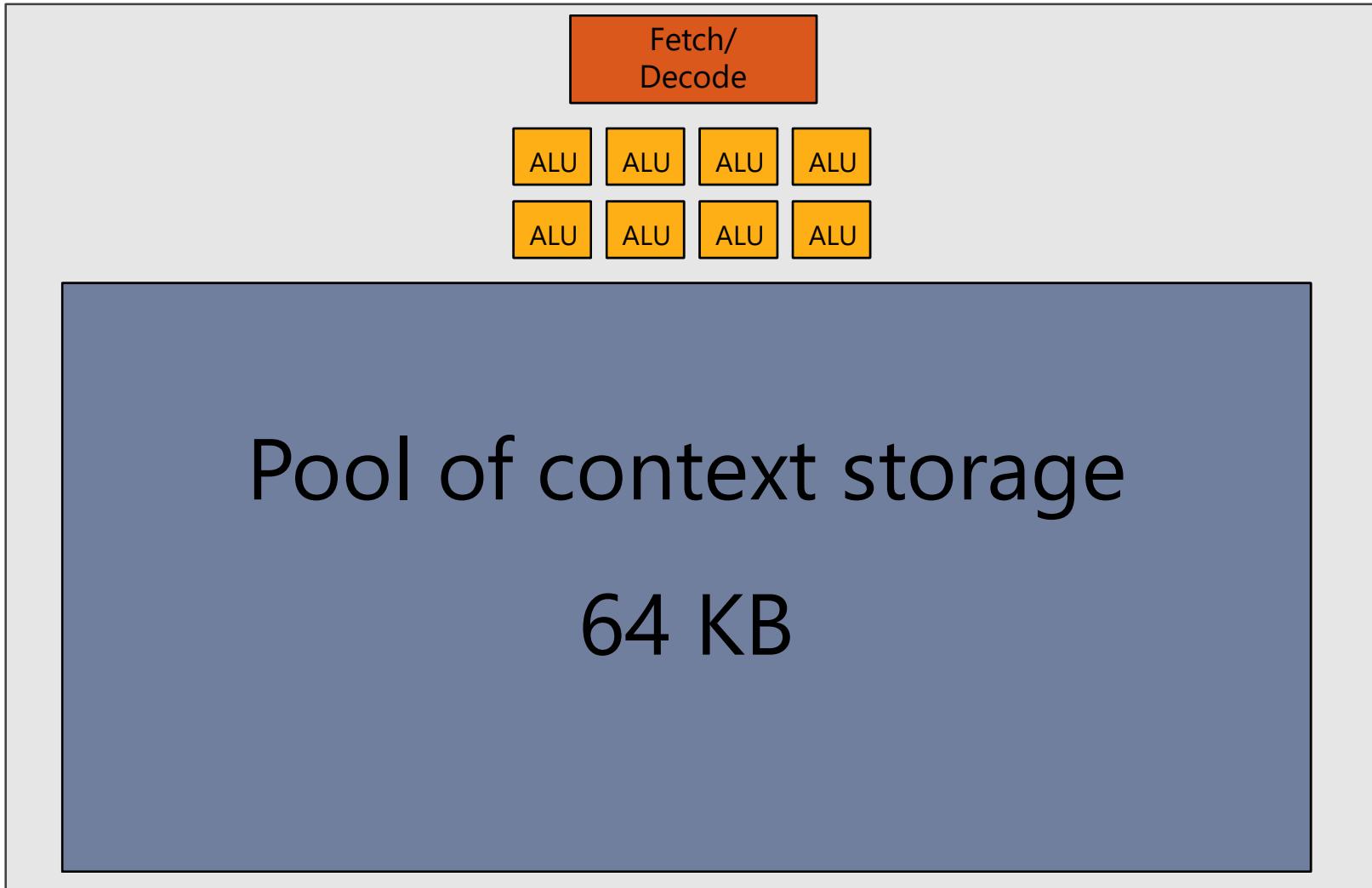
Hiding shader stalls



Throughput!

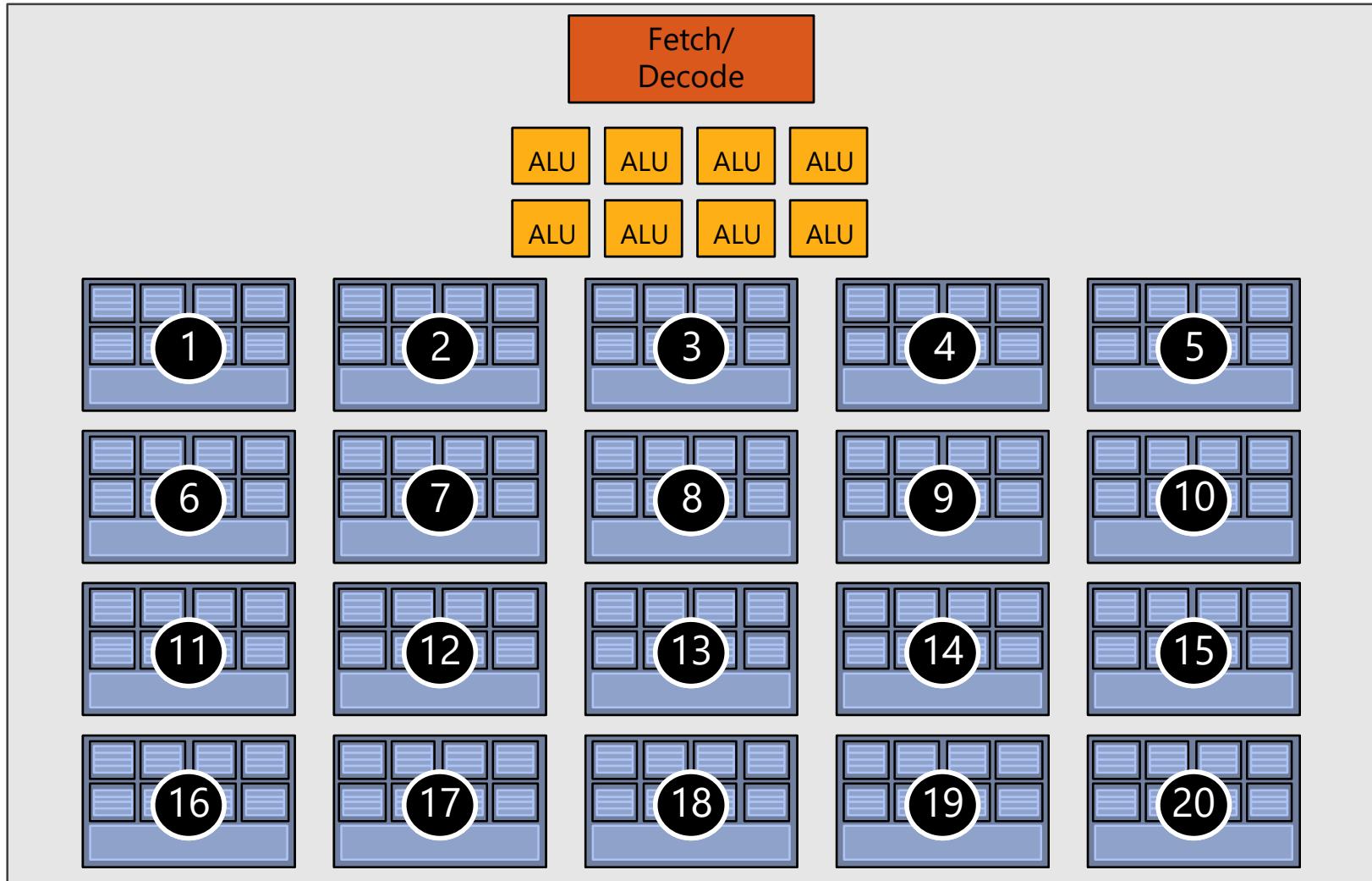


Storing contexts

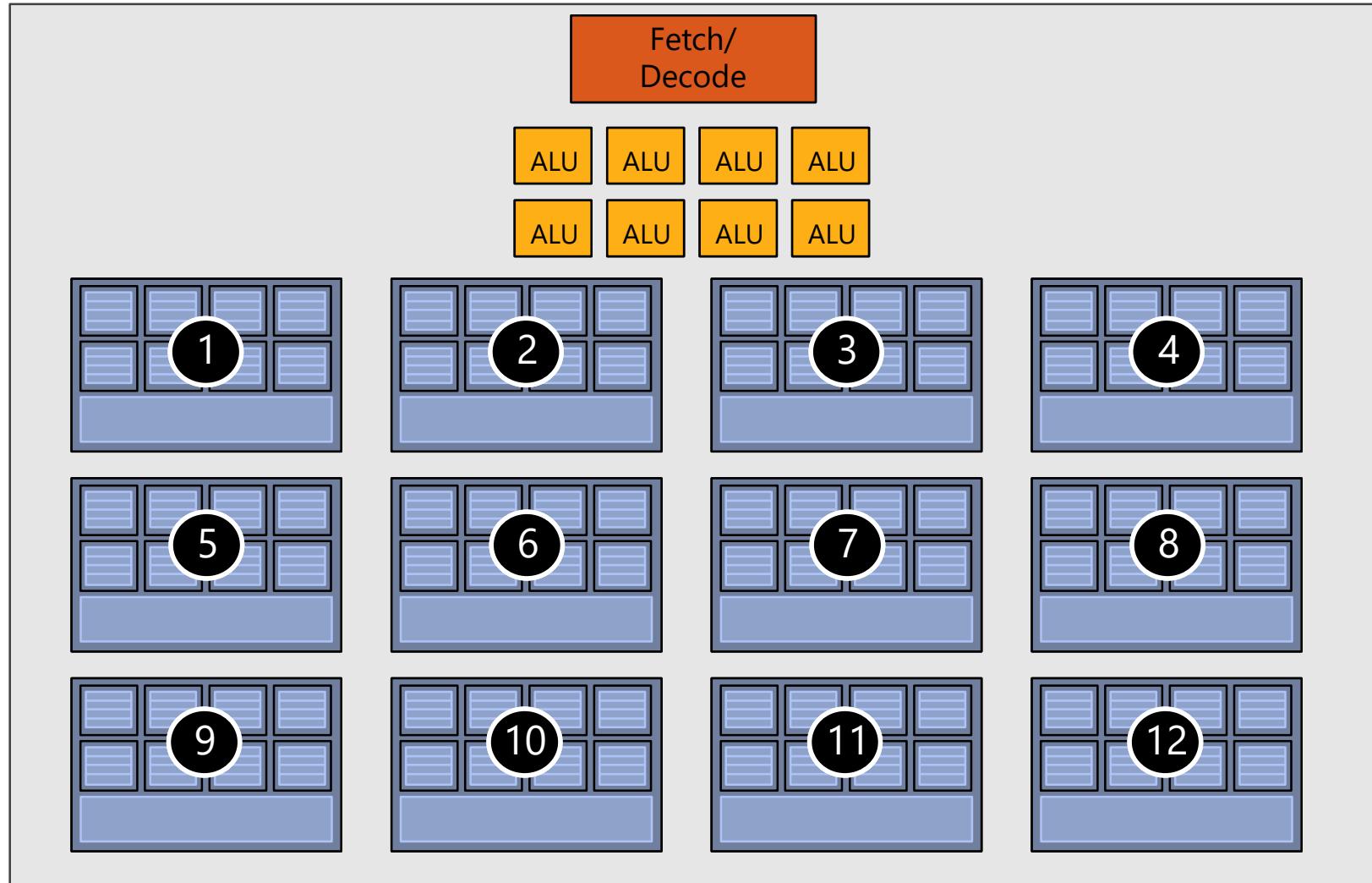


Twenty small contexts

(maximal latency hiding ability)

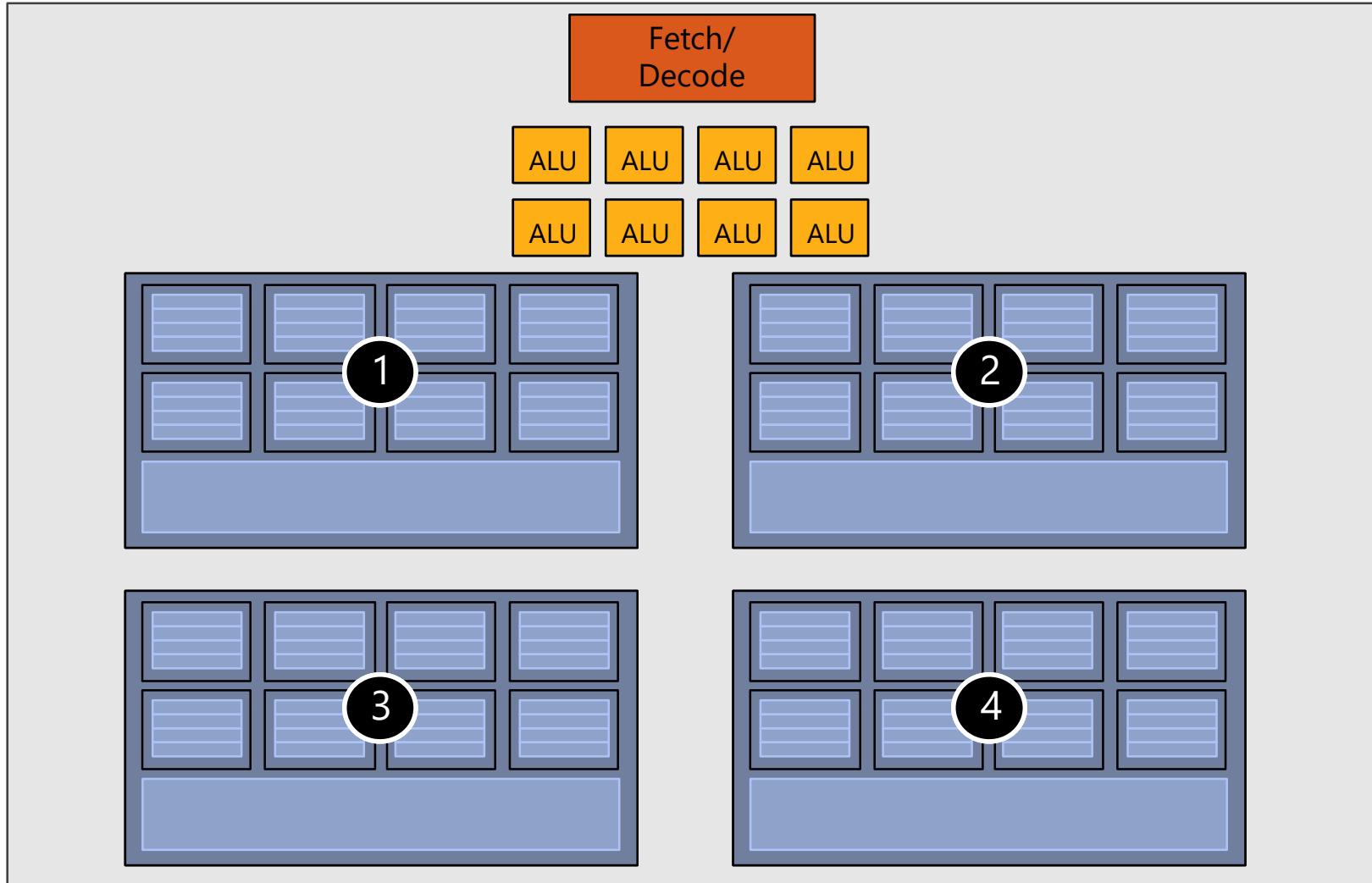


Twelve medium contexts



Four large contexts

(low latency hiding ability)



My chip!

16 cores

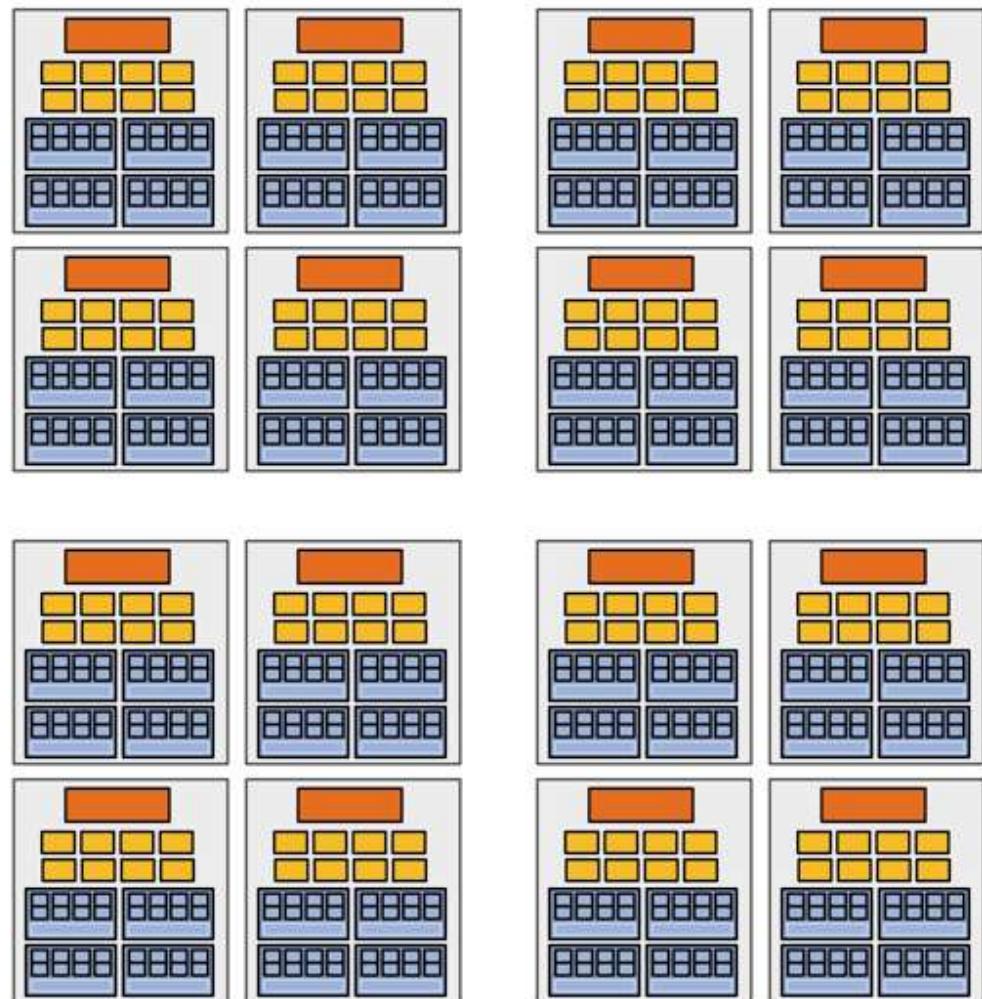
8 mul-add ALUs per core
(128 total)

16 simultaneous
instruction streams

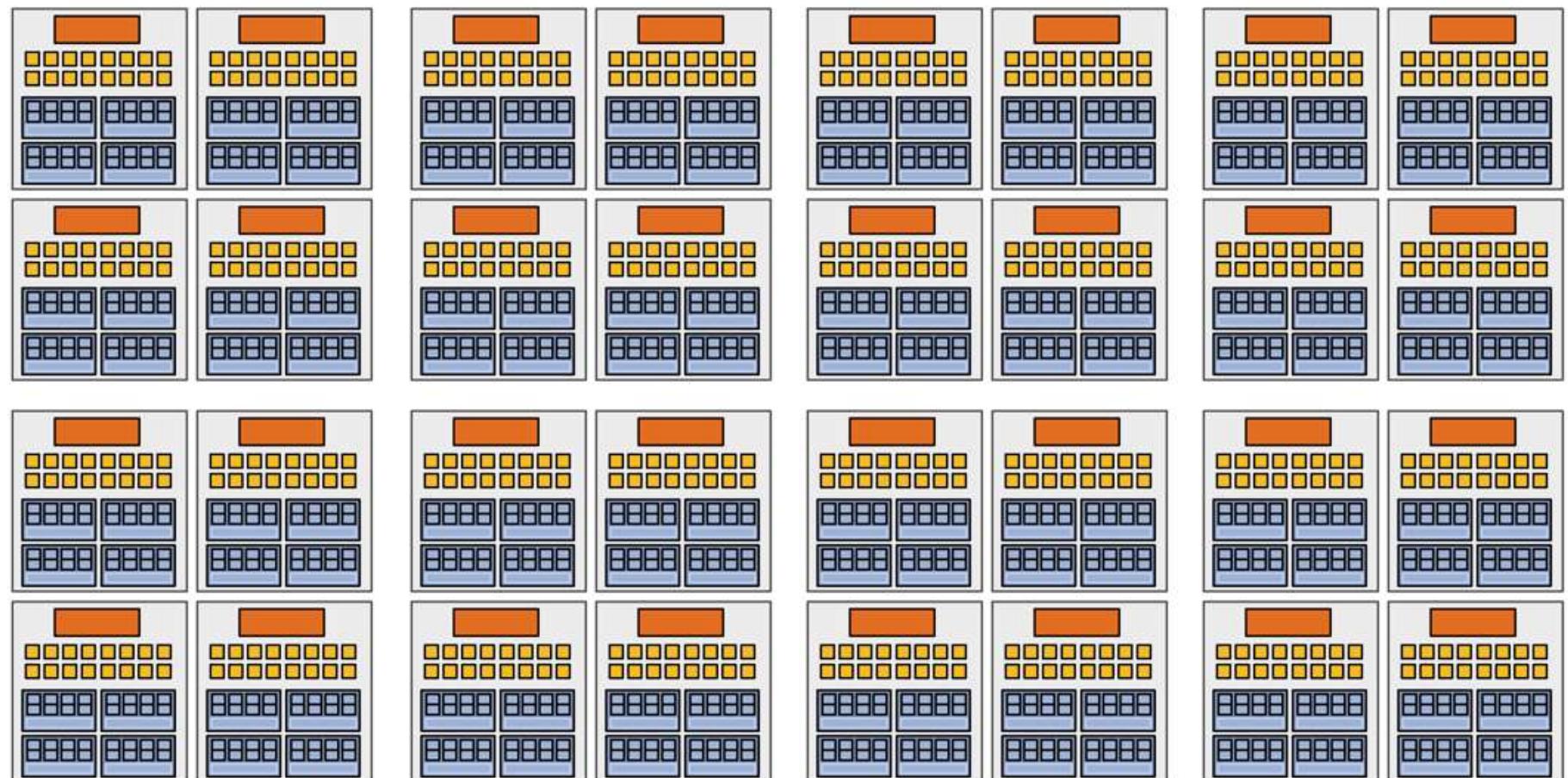
64 concurrent (but interleaved)
instruction streams

512 concurrent fragments

= 256 GFLOPs (@ 1GHz)



My “enthusiast” chip!



32 cores, 16 ALUs per core (512 total) = 1 TFLOP (@ 1 GHz)



Summary: three key ideas for high-throughput execution

- 1. Use many “slimmed down cores,” run them in parallel**
- 2. Pack cores full of ALUs (by sharing instruction stream overhead across groups of fragments)**
 - Option 1: Explicit SIMD vector instructions**
 - Option 2: Implicit sharing managed by hardware**
- 3. Avoid latency stalls by interleaving execution of many groups of fragments**
 - When one group stalls, work on another group**

NVIDIA Architectures (since first CUDA GPU)



Tesla: 2007-2009

- G80, G9x: 2007 (Geforce 8800, ...)
GT200: 2008/2009 (GTX 280, ...)

Fermi: 2010

- GF100, ... (GTX 480, ...)
GF110, ... (GTX 580, ...)

Kepler: 2012

- GK104, ... (GTX 680, ...)
GK110, ... (GTX 780, GTX Titan, ...)

Maxwell: 2015

- GM107, ... (GTX 750Ti, ...)
GM204, ... (GTX 980, Titan X, ...)

Pascal: 2016

- GP100 (Tesla P100, ...)
- GP10x: x=2,4,6,7,8, ...
(GTX 1080, Titan X *Pascal*...)

Volta: 2017/2018

- GV100, ...
(Tesla V100, Titan V, ...)

Turing: 2018/2019

- TU102, TU104, TU106, TU116, ...
(Titan RTX, RTX 2070, 2080, 2080Ti, ...)

Ampere: 2020

- GA100, GA102, GA104, ...
(A100, RTX 3070, 3080, 3090, ...)

Hopper, Lovelace: ?



NVIDIA Tesla Architecture

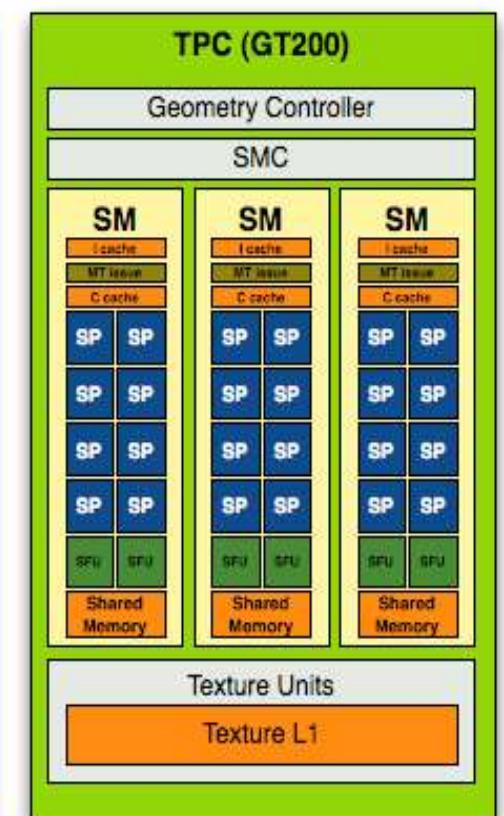
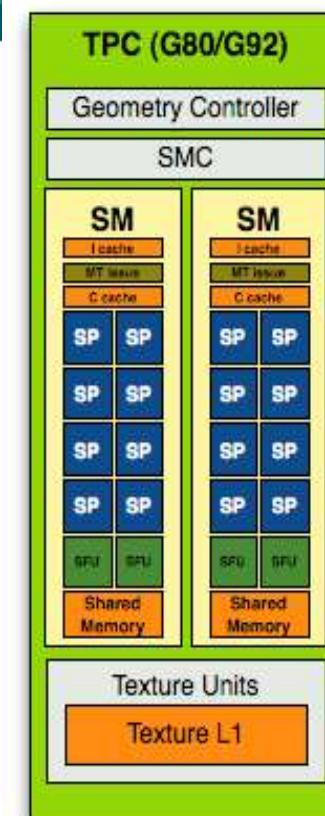
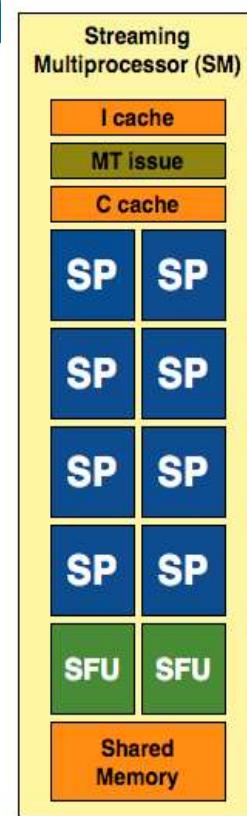
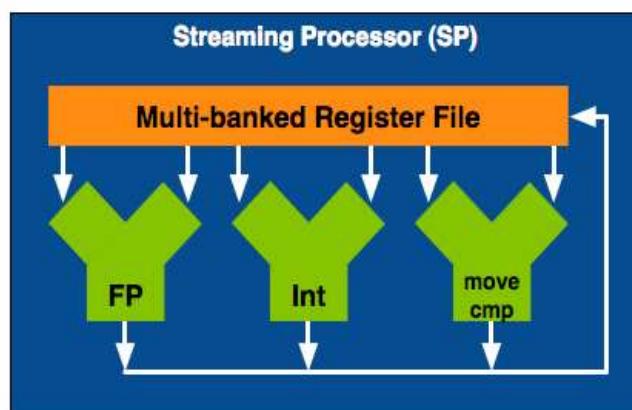
2007-2009

G80, G9x: 2007 (Geforce 8800, ...)

GT200: 2008/2009 (GTX 280, ...)

(this is not the Tesla product line!)

NVIDIA Tesla Architecture (not the Tesla product line!), G80: 2007, GT200: 2008/2009



G80: first CUDA GPU!

CC 1.x Multiprocessor

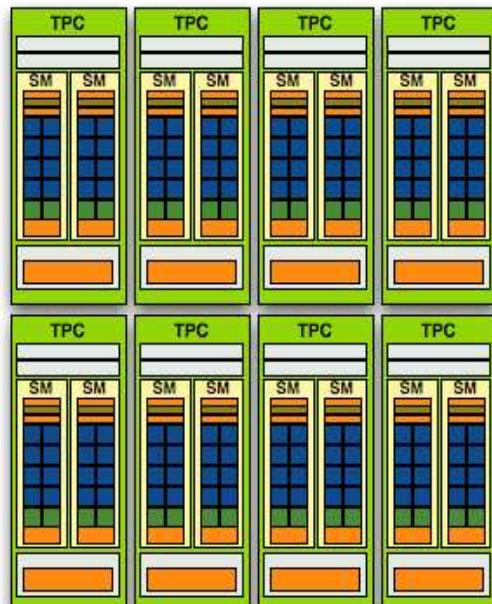
Courtesy AnandTech

- Streaming Processor (SP) [or: CUDA core; or: FP32 / FP64 / INT32 core, ...]
- Streaming Multiprocessor (SM)
- Texture/Processing Cluster (TPC)

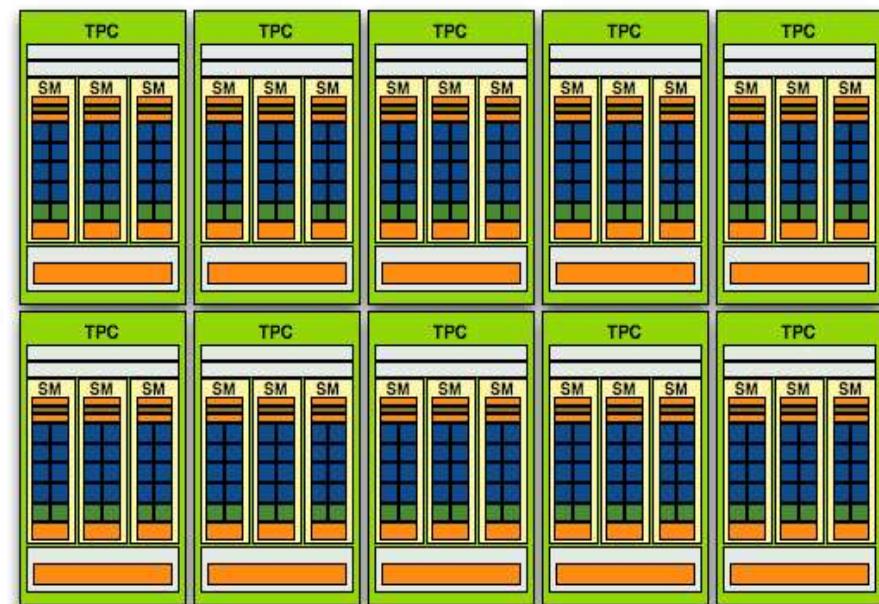
NVIDIA Tesla Architecture (not the Tesla product line!), G80: 2007, GT200: 2008/2009



- G80/G92: $8 \text{ TPCs} * (2 * 8 \text{ SPs}) = 128 \text{ SPs}$ [= CUDA cores]
- GT200: $10 \text{ TPCs} * (3 * 8 \text{ SPs}) = 240 \text{ SPs}$ [= CUDA cores]
- **Arithmetic intensity** has increased (num. of ALUs vs. texture units)



G80 / G92



GT200

Courtesy AnandTech



NVIDIA Fermi Architecture

2010

GF100, ... (GTX 480, ...)

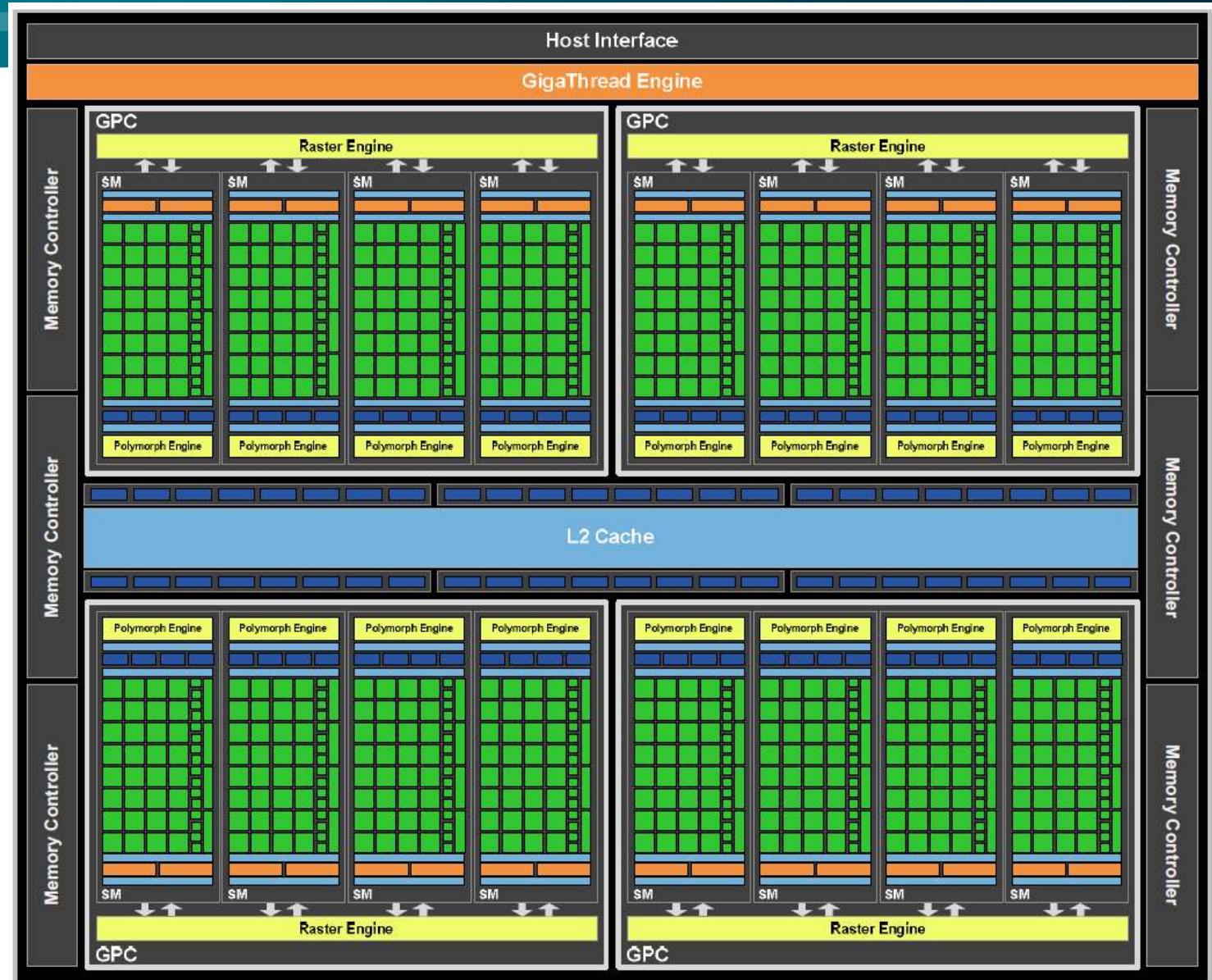
GF110, ... (GTX 580, ...)

NVIDIA Fermi Architecture (2010)



Full size

- 4 GPCs
- 4 SMs each
- 6 64-bit memory controllers (= 384 bit)

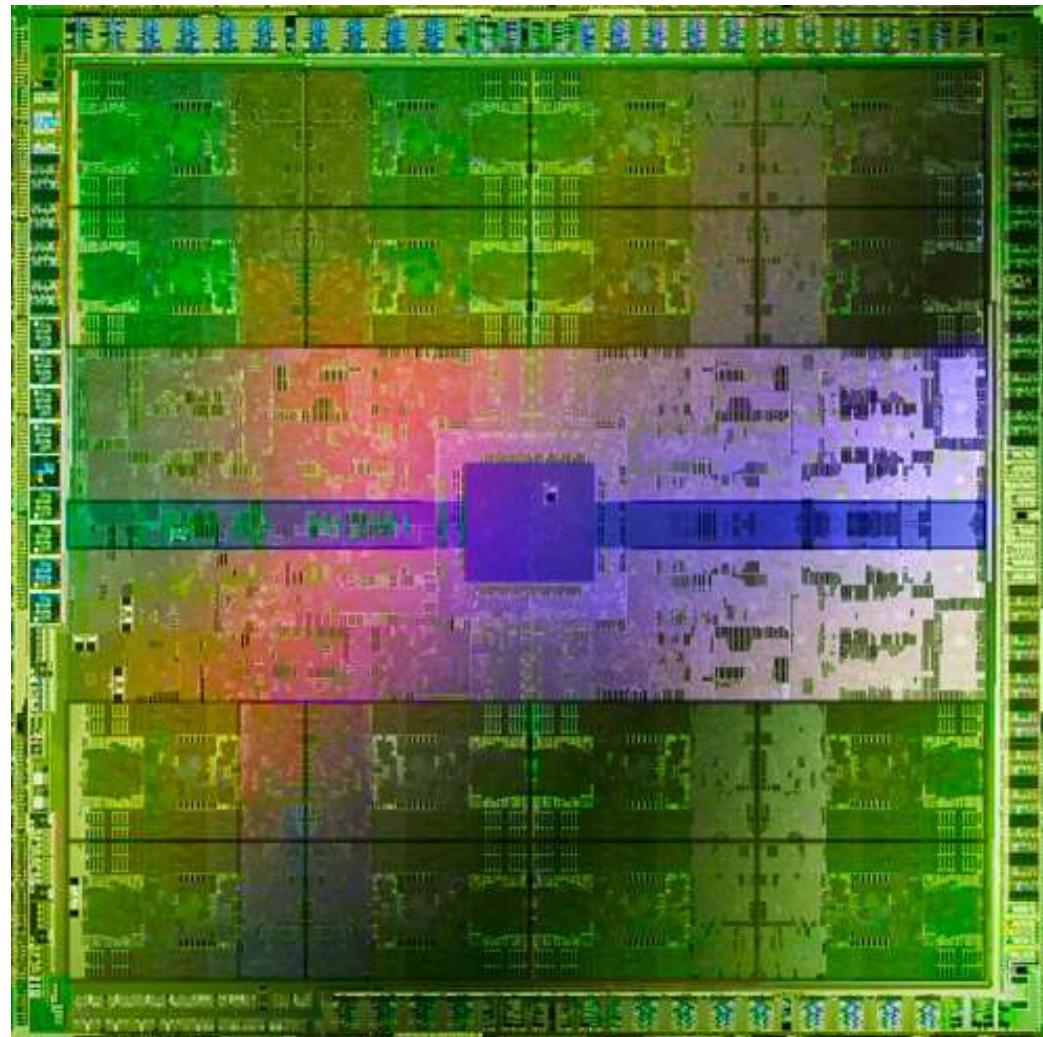


NVIDIA Fermi (GF100) Die Photo



Full size

- 4 GPCs
- 4 SMs each



NVIDIA Fermi SM (2010)

CC 2.x Multiprocessor

Streaming processors now called
CUDA cores

32 CUDA cores per Fermi
streaming multiprocessor (SM)

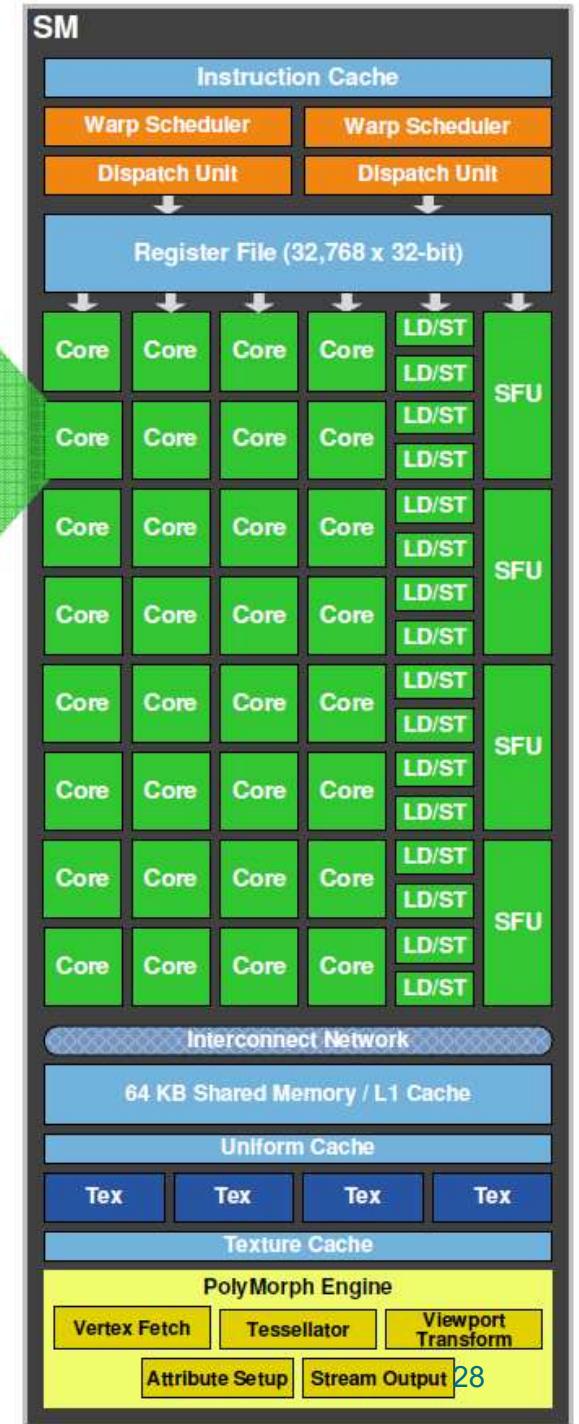
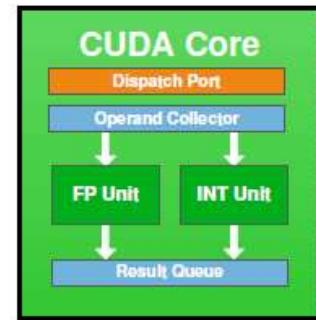
16 SMs = 512 CUDA cores

CPU-like cache hierarchy

- L1 cache / shared memory
- L2 cache

Texture units and caches now in SM

(instead of with TPC=multiple SMs in G80/GT200)





Graphics Processor Clusters (GPC)

(instead of TPC on GT200)

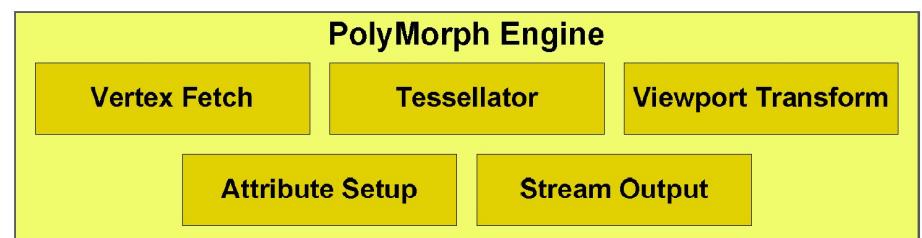
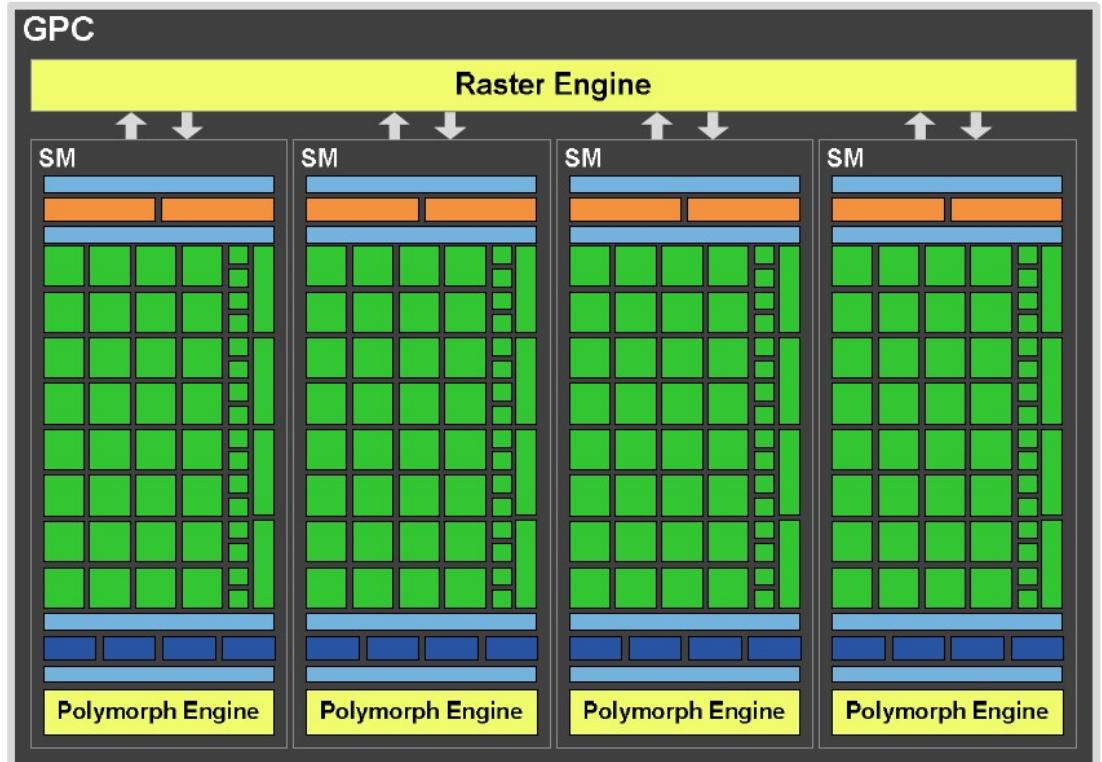
4 SMs

32 CUDA cores / SM

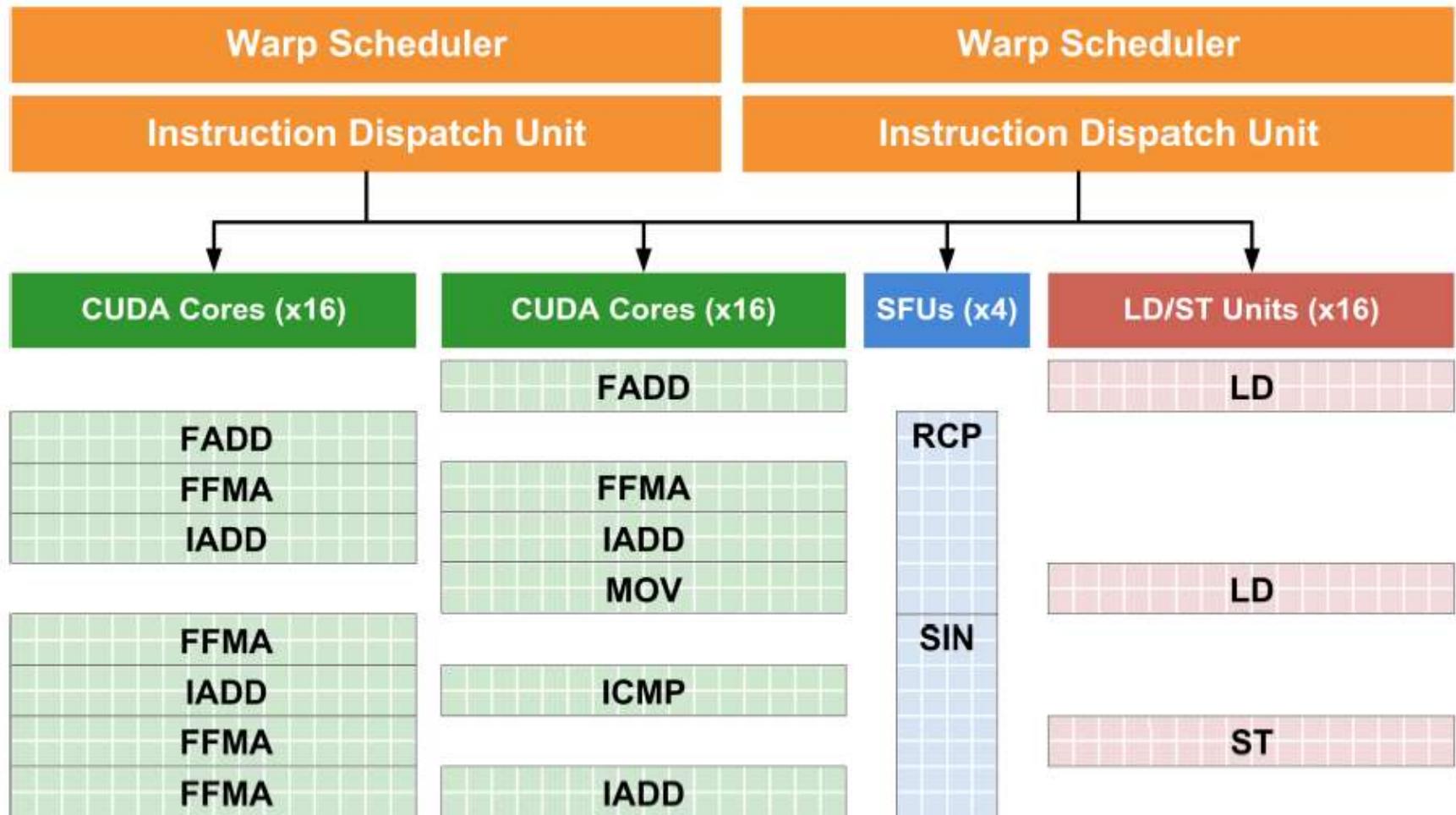
4 SMs / GPC =
128 cores / GPC

Decentralized rasterization
and geometry

- 4 raster engines
- 16 "PolyMorph" engines



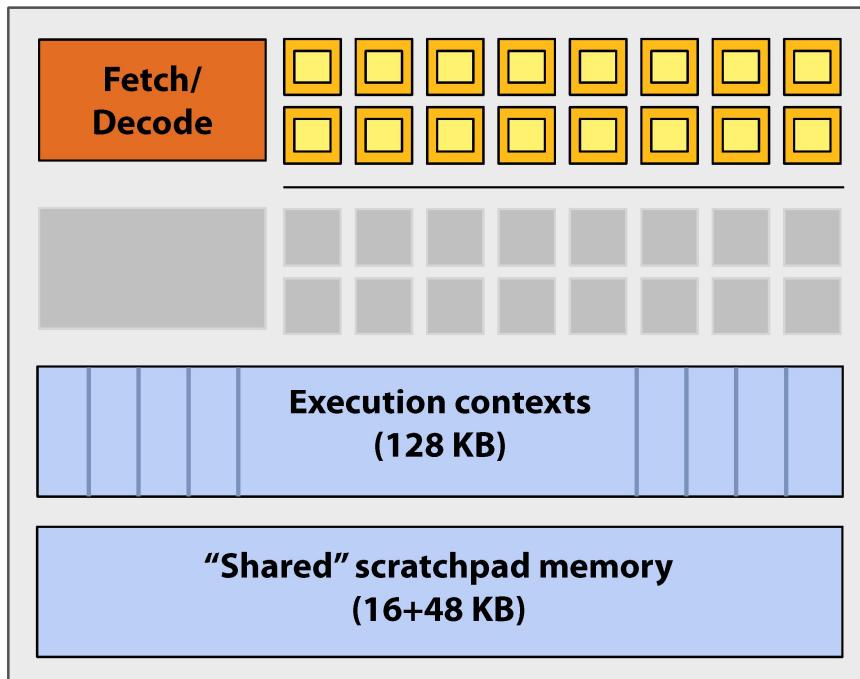
Dual Warp Schedulers



NVIDIA Fermi Architecture (2010)



NVIDIA GeForce GTX 480 “core”



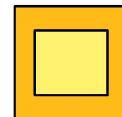
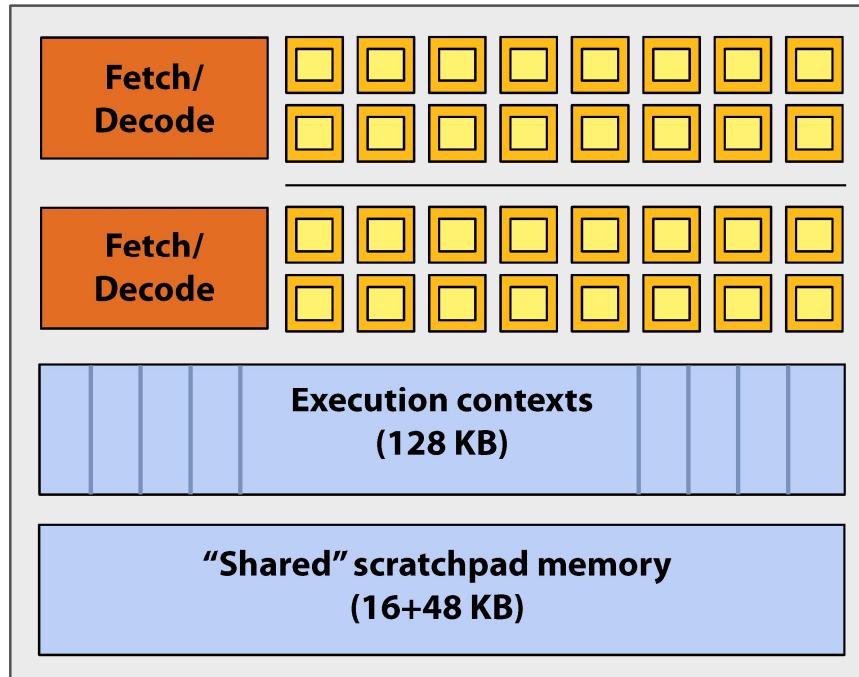
- **Groups of 32 fragments share an instruction stream**
- **Up to 48 groups are simultaneously interleaved**
- **Up to 1536 individual contexts can be stored**

Source: Fermi Compute Architecture Whitepaper
CUDA Programming Guide 3.1, Appendix G

NVIDIA Fermi Architecture (2010)



NVIDIA GeForce GTX 480 “core”



= SIMD function unit,
control shared across 16 units
(1 MUL-ADD per clock)

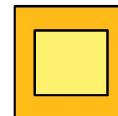
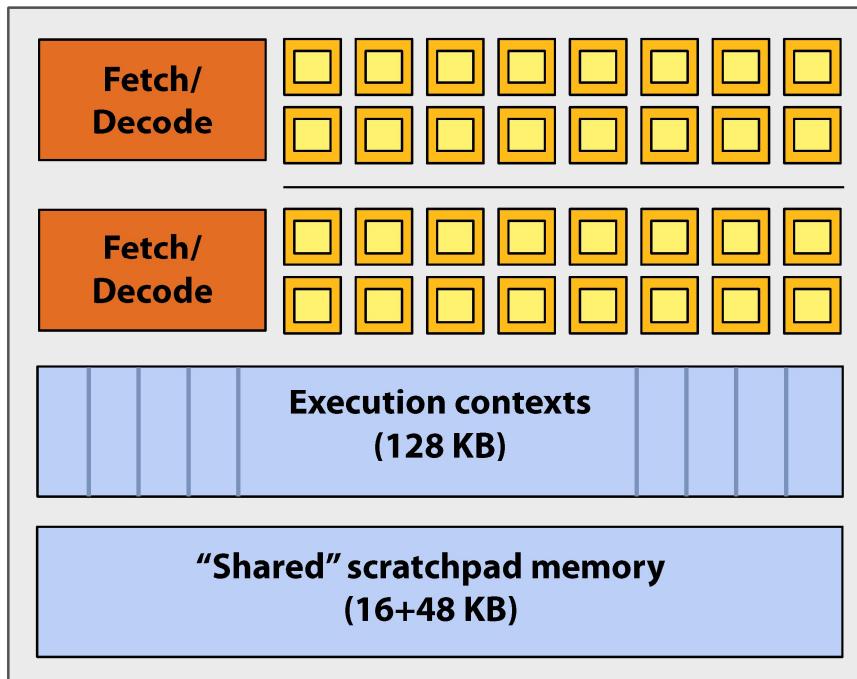
- The core contains 32 functional units
- Two groups are selected each clock
(decode, fetch, and execute two instruction streams in parallel)

Source: Fermi Compute Architecture Whitepaper
CUDA Programming Guide 3.1, Appendix G

NVIDIA Fermi Architecture (2010)



NVIDIA GeForce GTX 480 "SM"



= **CUDA core**
(1 MUL-ADD per clock)

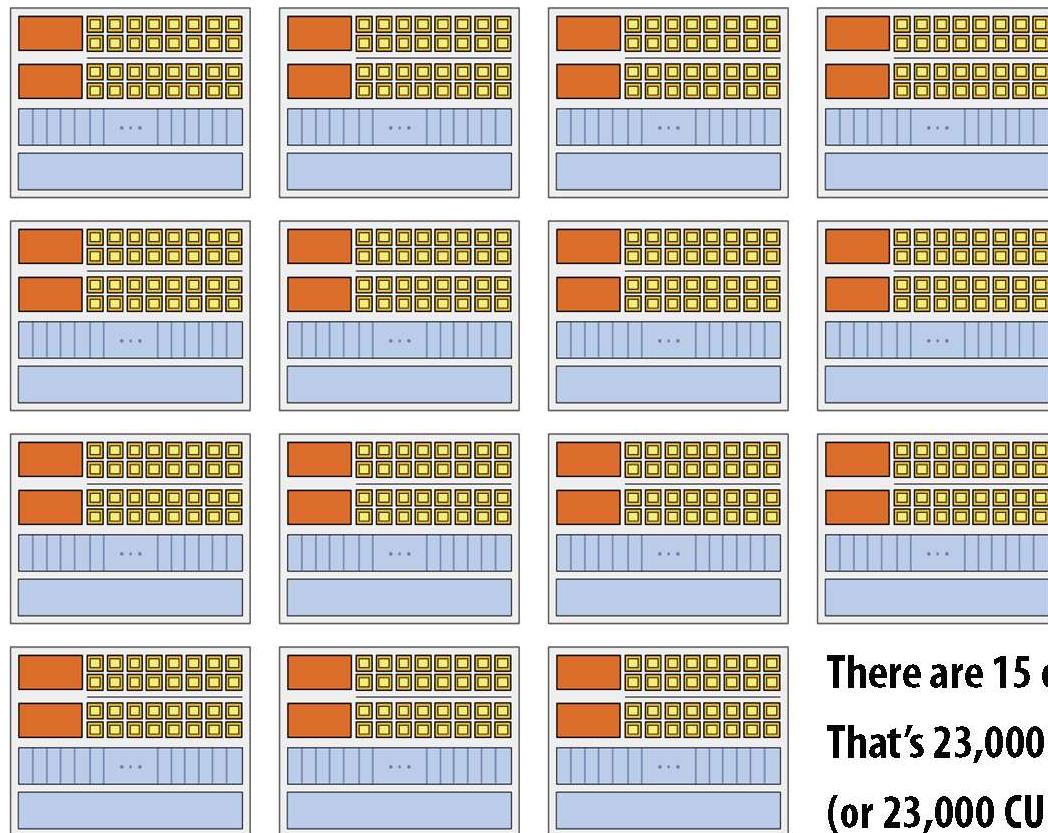
- The **SM** contains **32 CUDA cores**
- Two **warps** are selected each clock
(decode, fetch, and execute two **warps** in parallel)
- Up to **48 warps** are interleaved, totaling **1536 CUDA threads**

Source: Fermi Compute Architecture Whitepaper
CUDA Programming Guide 3.1, Appendix G

NVIDIA Fermi Architecture (2010)



NVIDIA GeForce GTX 480



**There are 15 of these things on the GTX 480:
That's 23,000 fragments!
(or 23,000 CUDA threads!)**



NVIDIA Kepler Architecture

2012

GK104, ... (GTX 680, ...)

GK110, ... (GTX 780, GTX Titan, ...)



NVIDIA Kepler Architecture (2012)

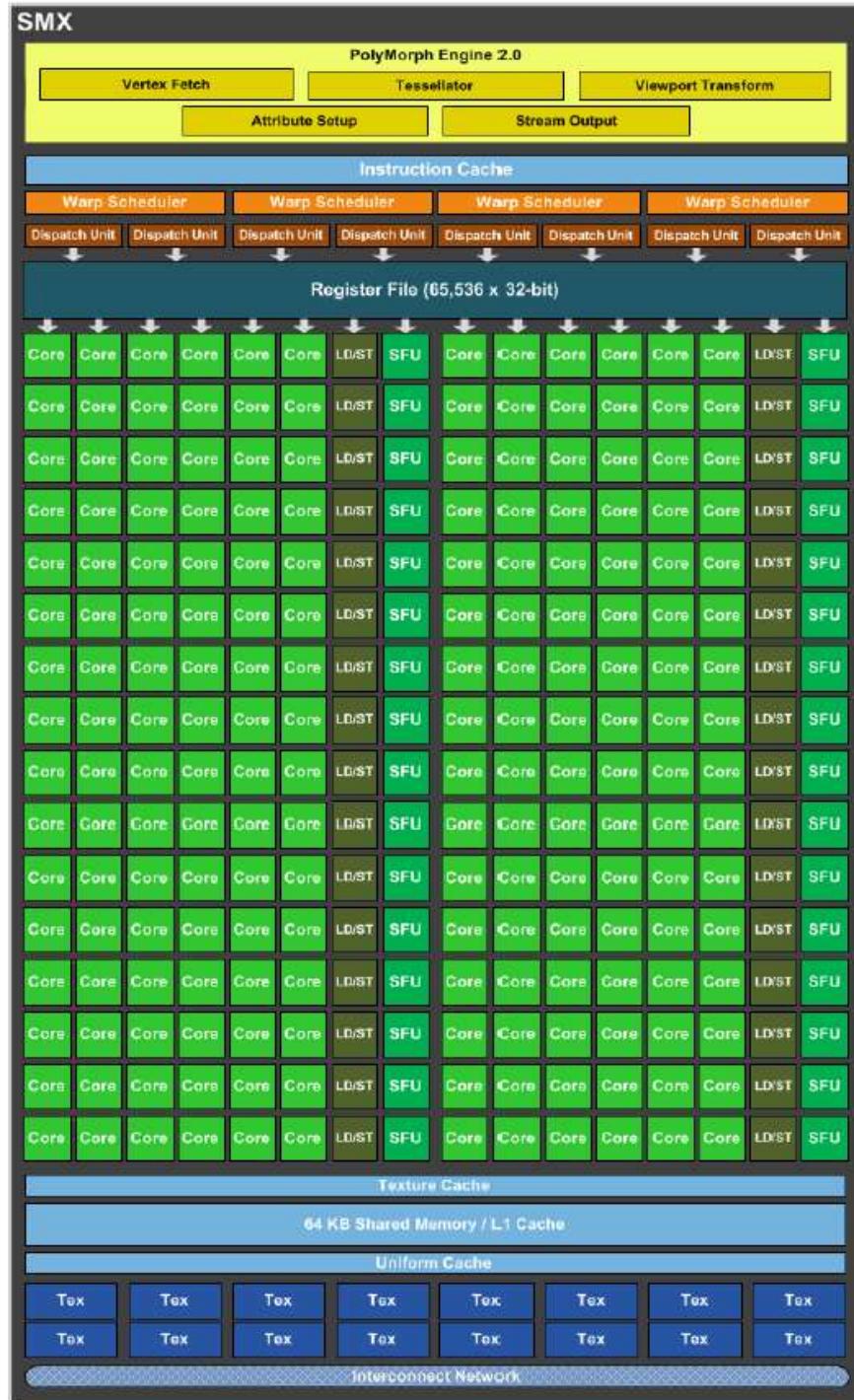


GK104 SMX

CC 3.0 Multiprocessor

- 192 CUDA cores
($192 = 6 * 32$)
- 32 LD/ST units
- 32 SFUs
- 16 texture units

Two dispatch units per warp scheduler
exploit ILP
(instruction-level parallelism)



GK110 SMX

CC 3.5 Multiprocessor

- 192 CUDA cores
($192 = 6 * 32$)
- 64 DP units
- 32 LD/ST units
- 32 SFUs
- 16 texture units

New read-only
data cache (48KB)



NVIDIA Kepler Architecture (2012)



Three different versions

- Compute capability 3.0 (GK104)
 - Geforce GTX 680, ...
 - Quadro K5000
 - Tesla K10
- Compute capability 3.5 (GK110)
 - Geforce GTX 780 / Titan / Titan Black
 - Quadro K6000
 - Tesla K20, Tesla K40
- Compute capability 3.7 (GK210)
 - Tesla K80
 - Came out much later (~end of 2014)



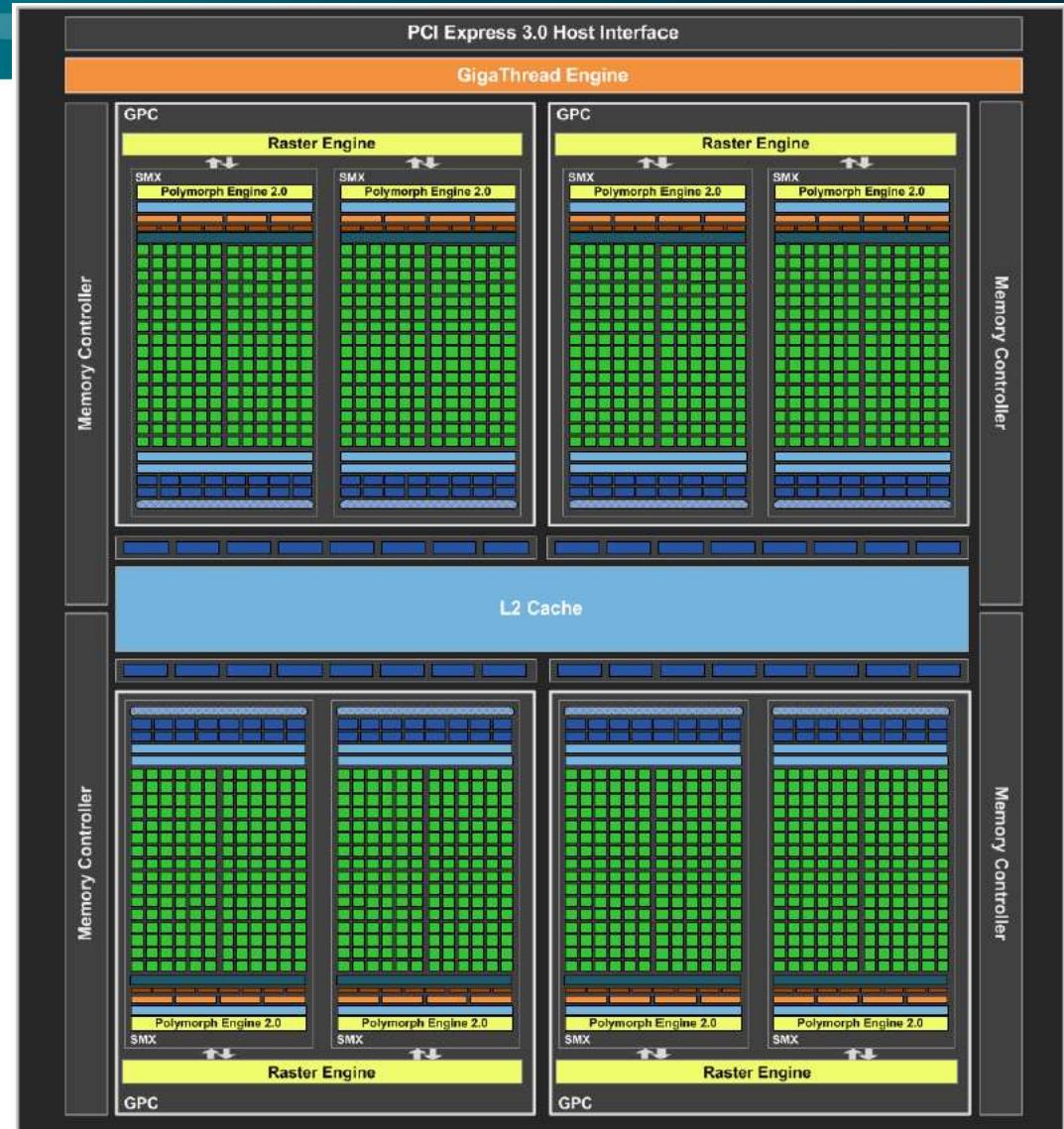
NVIDIA Kepler / GK104 Structure



Full size

- 4 GPCs
- 2 SMXs each

= 8 SMXs,
1536 CUDA cores





NVIDIA Kepler / GK110 Structure (1)

Full size

- 15 SMXs
(Titan Black;
Titan: 14)
- 2880 CUDA
cores
(Titan Black;
Titan: 2688)
- 5 GPCs of
3 SMXs each





NVIDIA Kepler / GK110 Structure (2)

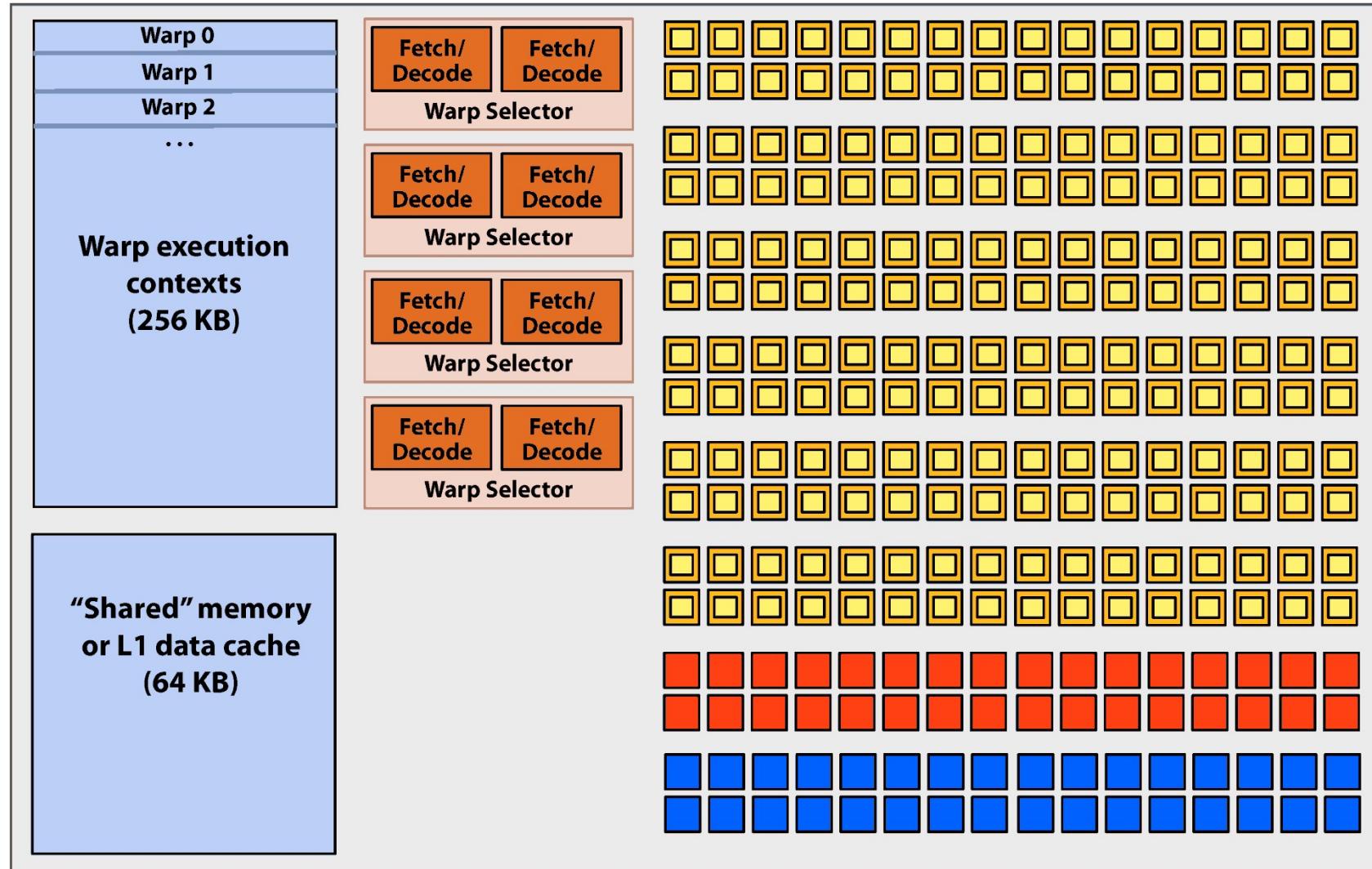
Titan (not Black)

- 14 SMXs
- 2688 CUDA cores
- 5 GPCs with 3 SMXs or 2 SMXs each



Bonus slides: NVIDIA GTX 680 (2012)

NVIDIA Kepler GK104 architecture SMX unit (one “core”)



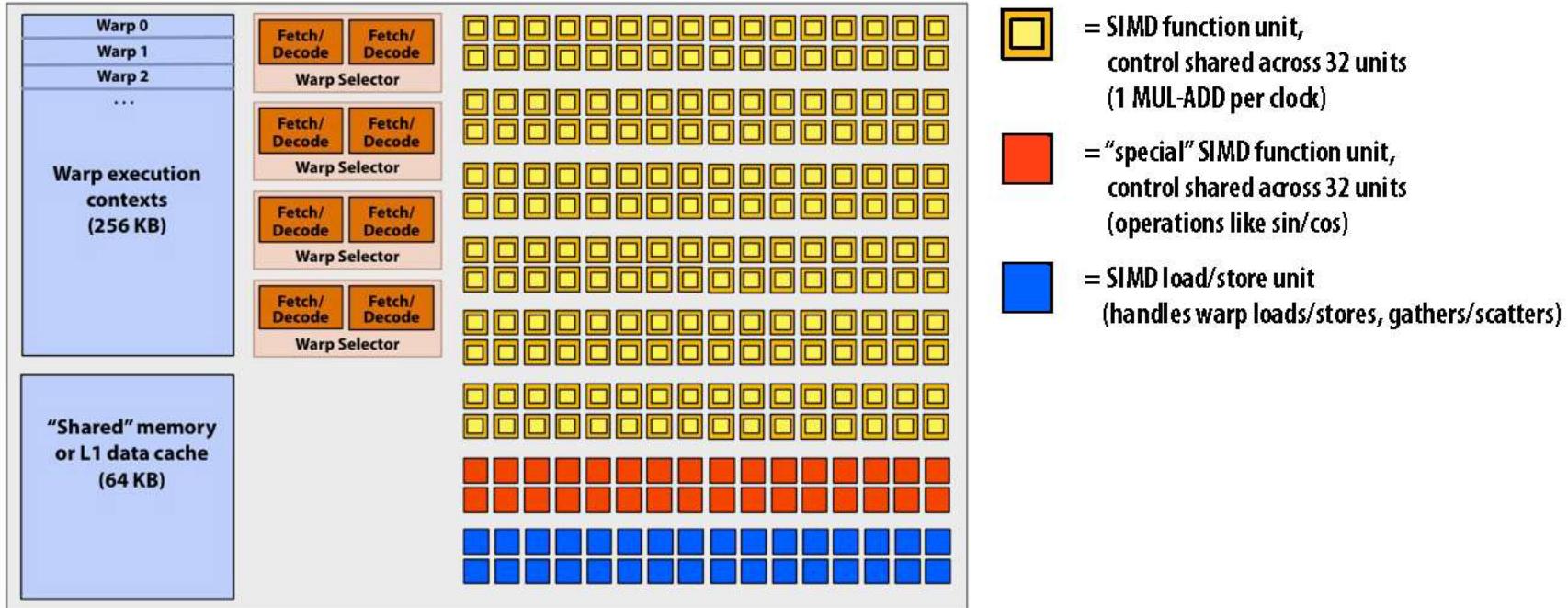
Yellow = SIMD function unit,
control shared across 32 units
(1 MUL-ADD per clock)

Red = “special” SIMD function unit,
control shared across 32 units
(operations like sin/cos)

Blue = SIMD load/store unit
(handles warp loads/stores, gathers/scatters)

Bonus slides: NVIDIA GTX 680 (2012)

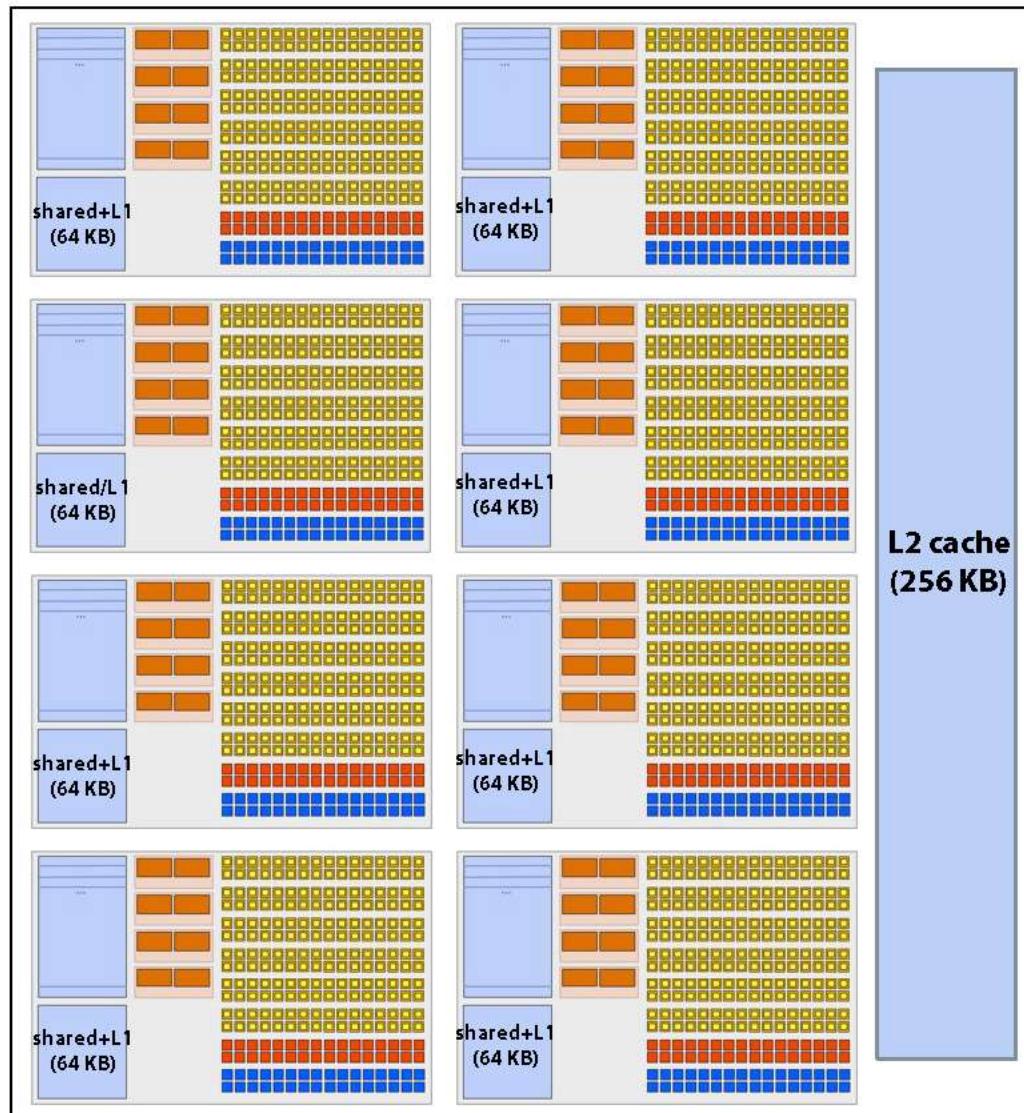
NVIDIA Kepler GK104 architecture SMX unit (one “core”)



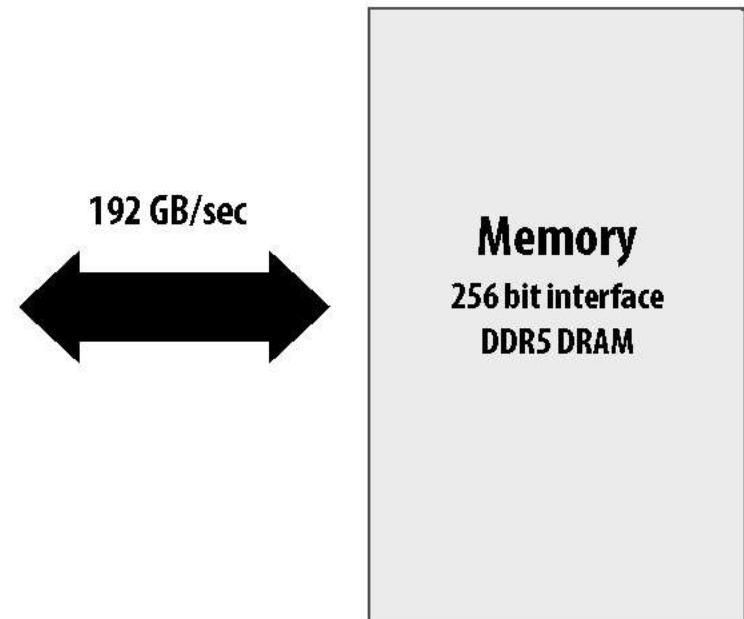
- **SMX core resource limits:**
 - Maximum warp execution contexts: 64 (2,048 total CUDA threads)
 - Maximum thread blocks: 16
- **SMX core operation each clock:**
 - Select up to four runnable warps from up to 64 resident on core (thread-level parallelism)
 - Select up to two runnable instructions per warp (instruction-level parallelism)
 - Execute instructions on available groups of SIMD ALUs, special-function ALUs, or LD/ST units

Bonus slides: NVIDIA GTX 680 (2012)

NVIDIA Kepler GK104 architecture



- 1 GHz clock
- Eight SMX cores per chip
- $8 \times 192 = 1,536$ SIMD mul-add ALUs
= 3 TFLOPs
- Up to 512 interleaved warps per chip
(16,384 CUDA threads/chip)
- TDP: 195 watts





NVIDIA Maxwell Architecture

2015

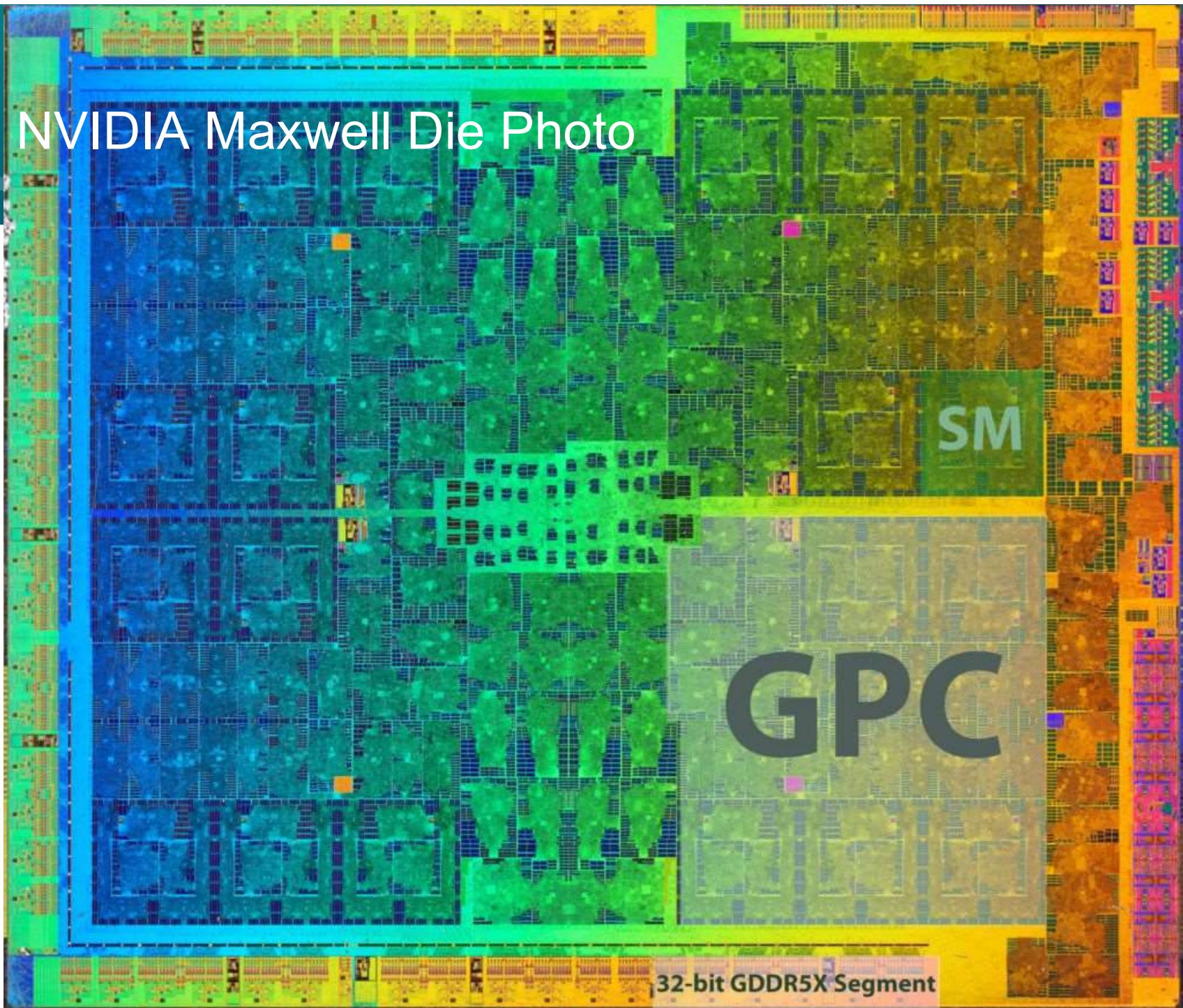
GM107, ... (GTX 750Ti, ...)
GM204, ... (GTX 980, Titan X, ...)



NVIDIA Maxwell Architecture (2015)



NVIDIA Maxwell Die Photo



Maxwell (GM) Architecture

CC 5.x Multiprocessor: SMM

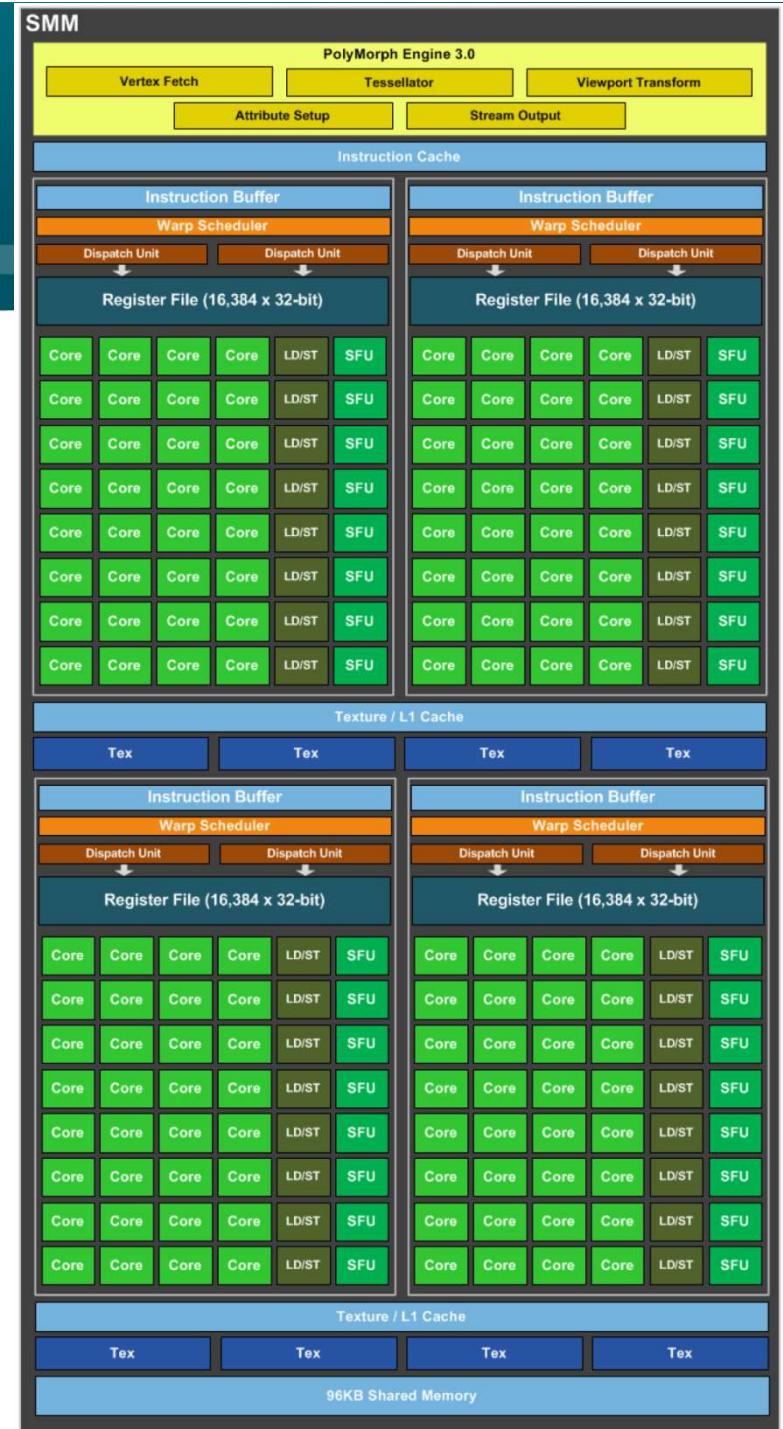
- 128 CUDA cores
- 4 DP units

4 partitions inside SMM

- 32 CUDA cores each
- 8 LD/ST units each
- Each has its own warp scheduler, two dispatch units, register file

Shared memory and L1 cache now separate!

- L1 cache shares with texture cache
- Shared memory is its own space



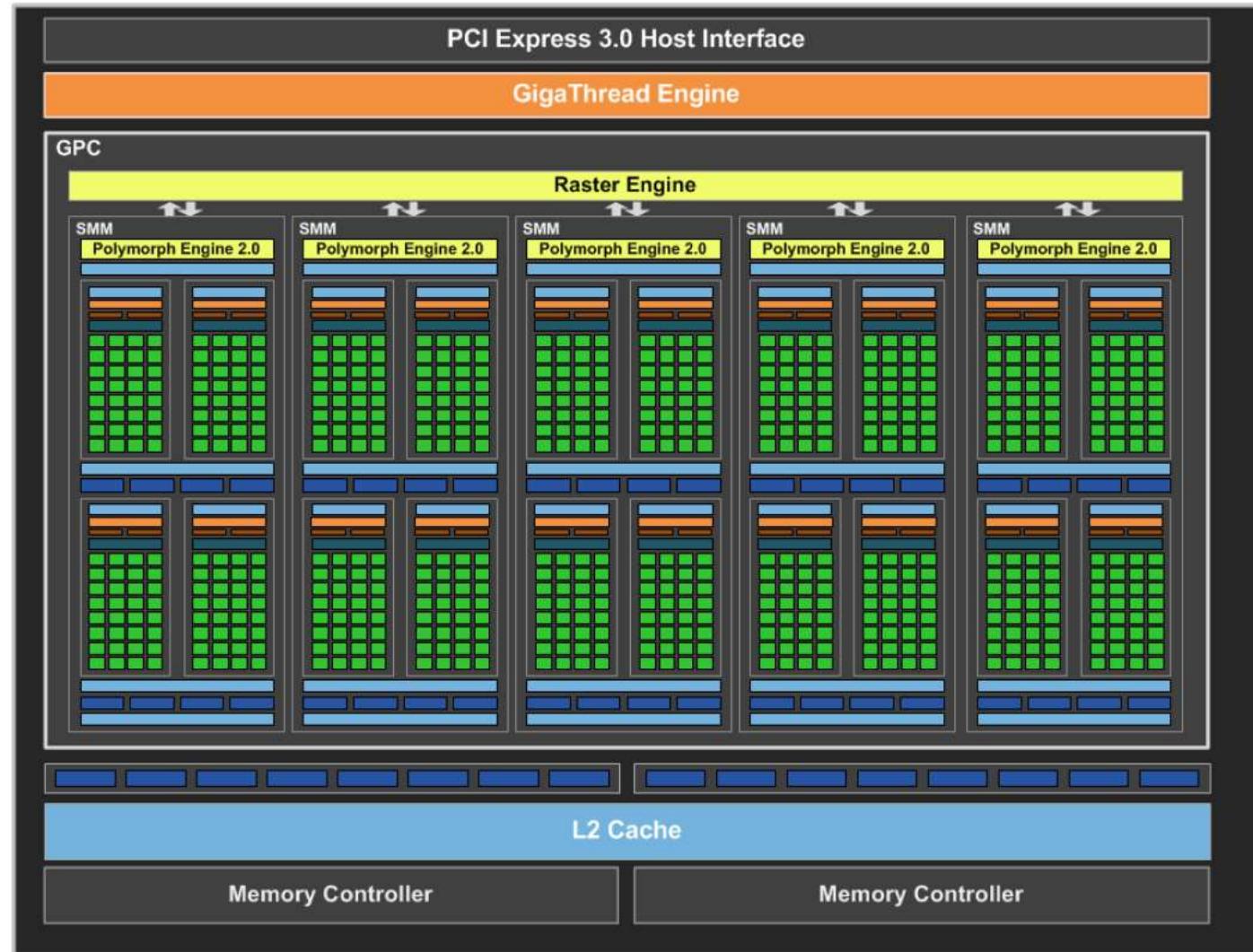
Maxwell (GM) Architecture



First gen.

GM107
(GTX 750Ti)

5 SMMs
(640 CUDA
cores in
total)



Maxwell (GM) Architecture



Second gen.

GM204
(GTX 980)

16 SMMs
(2048 CUDA
cores in
total)

4 GPCs of 4
SMMs





Maxwell (GM) vs. Kepler (GK) Architecture

GK107 vs. GM107

GPU	GK107 (Kepler)	GM107 (Maxwell)
CUDA Cores	384	640
Base Clock	1058 MHz	1020 MHz
GPU Boost Clock	N/A	1085 MHz
GFLOPs	812.5	1305.6
Texture Units	32	40
Texel fill-rate	33.9 Gigatexels/sec	40.8 Gigatexels/sec
Memory Clock	5000 MHz	5400 MHz
Memory Bandwidth	80 GB/sec	86.4 GB/sec
ROPs	16	16
L2 Cache Size	256KB	2048KB
TDP	64W	60W
Transistors	1.3 Billion	1.87 Billion
Die Size	118 mm ²	148 mm ²
Manufacturing Process	28-nm	28-nm



Maxwell (GM) vs. Kepler (GK) Architecture

GK107 vs. GM204

GPU	GeForce GTX 680 (Kepler)	GeForce GTX 980 (Maxwell)
SMs	8	16
CUDA Cores	1536	2048
Base Clock	1006 MHz	1126 MHz
GPU Boost Clock	1058 MHz	1216 MHz
GFLOPs	3090	4612 ¹
Texture Units	128	128
Texel fill-rate	128.8 Gigatexels/sec	144.1 Gigatexels/sec
Memory Clock	6000 MHz	7000 MHz
Memory Bandwidth	192 GB/sec	224 GB/sec
ROPs	32	64
L2 Cache Size	512KB	2048KB
TDP	195 Watts	165 Watts
Transistors	3.54 billion	5.2 billion
Die Size	294 mm ²	398 mm ²
Manufacturing Process	28-nm	28-nm

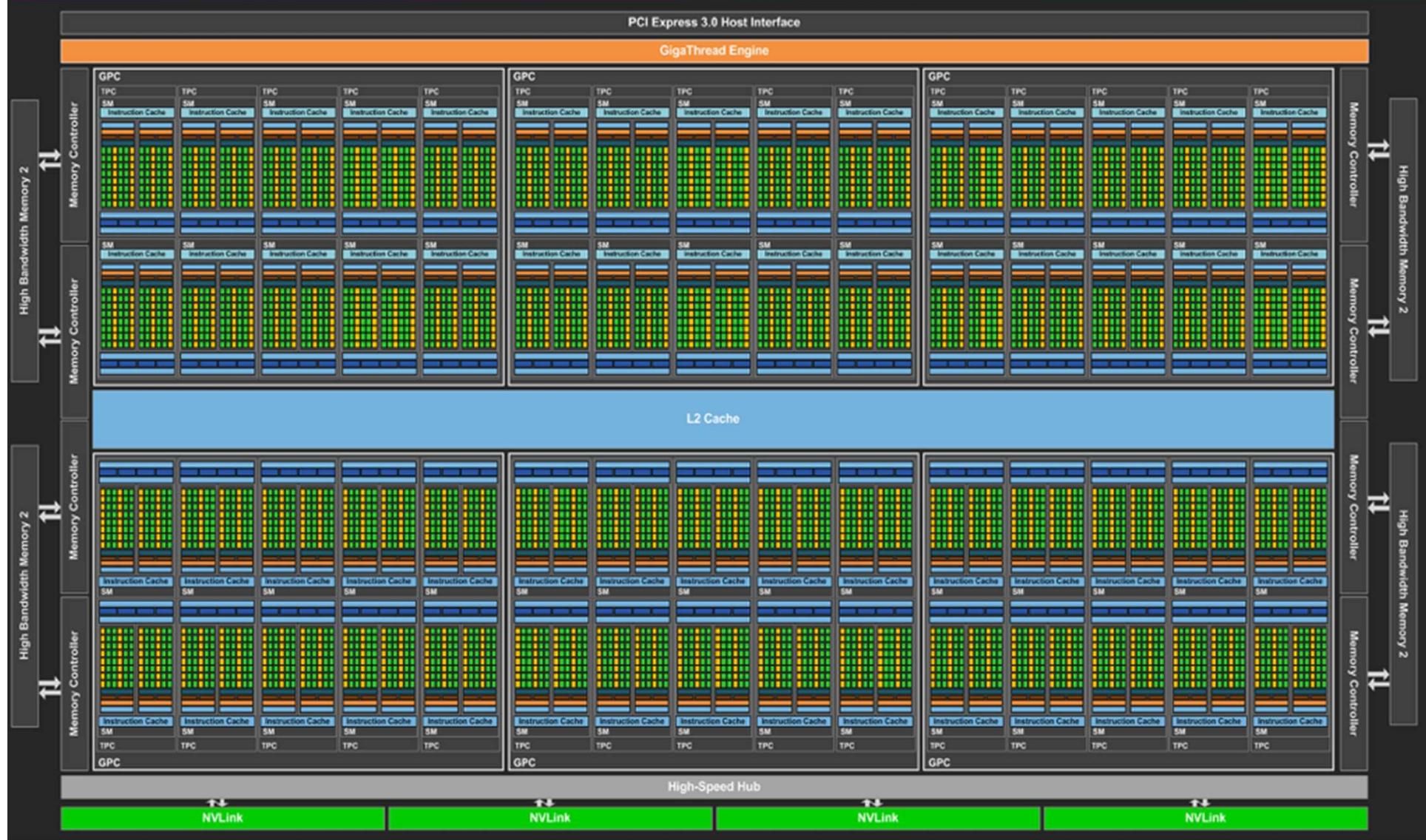


NVIDIA Pascal Architecture

2016

GP100, GP10x: x=2,4,6,7,8, ...
(GTX 1080, Titan X *Pascal*...)

NVIDIA Pascal Architecture (2016)



NVIDIA Pascal GP100 SM



CC 6.0 Multiprocessor

- 64 CUDA cores
- 32 DP units



2 partitions inside SM

- 32 CUDA cores each; 16 DP units each; 8 LD/ST units each
- Each has its own warp scheduler, two dispatch units, register file

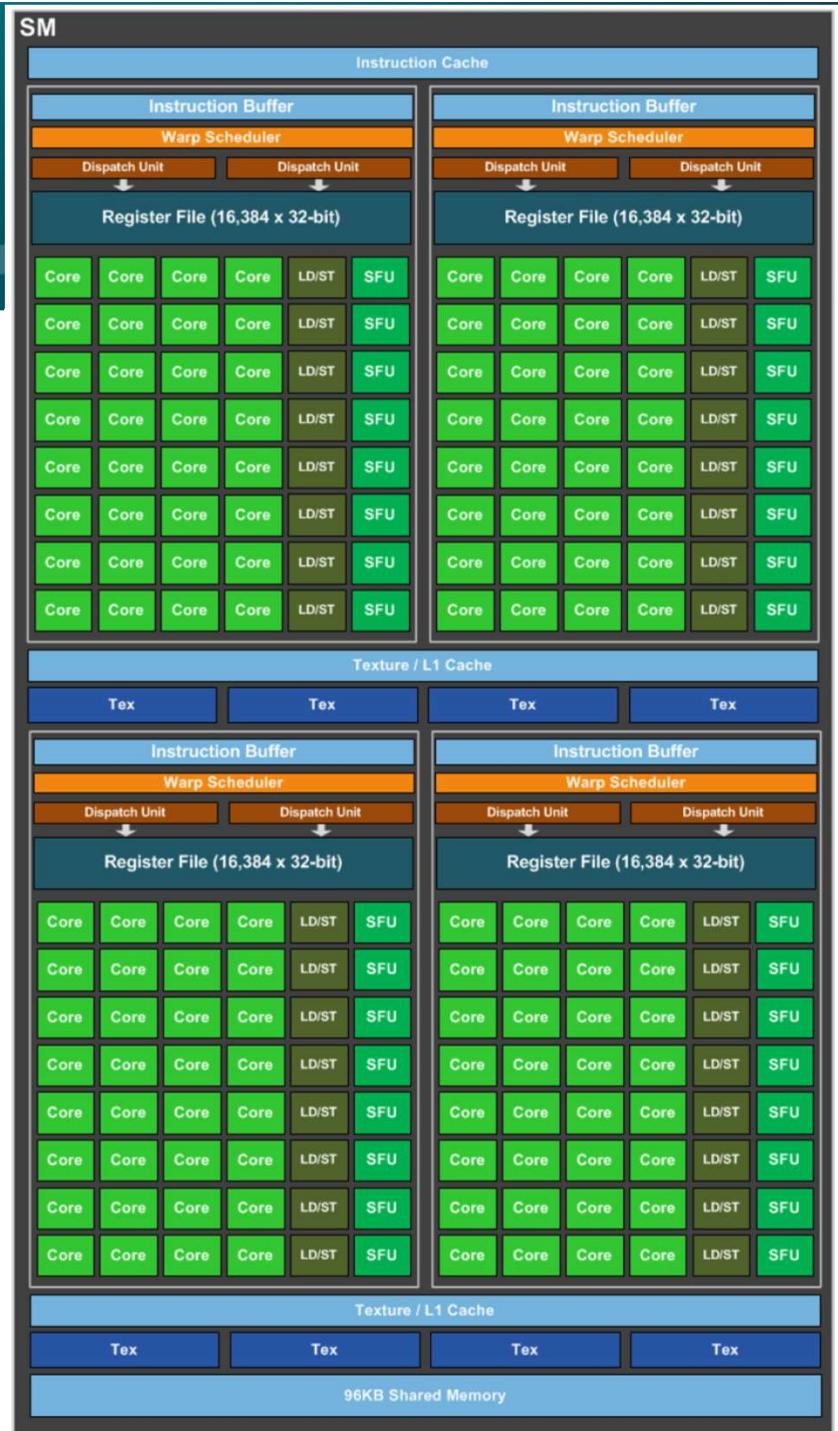
NVIDIA Pascal GP104 SM

CC 6.1/6.2 Multiprocessor

- 128 CUDA cores

4 partitions inside SM

- 32 CUDA cores; 8 LD/ST units
- Each has its own warp scheduler, two dispatch units, register file



NVIDIA Pascal Architecture (2016)



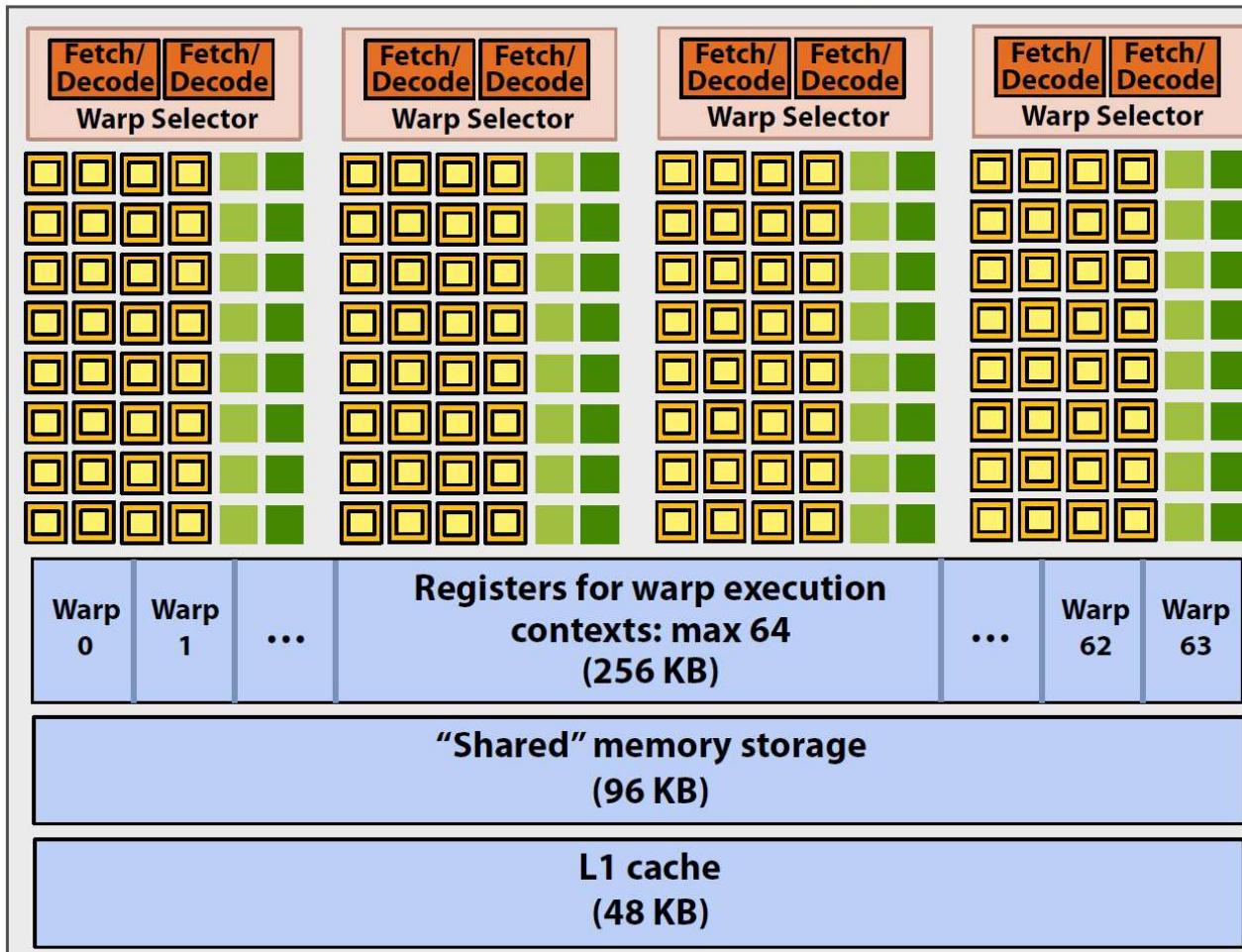
Total chip capacity on Tesla P100 (GP100)

- 56 SMs
 - 64 CUDA cores / SM = 3,584 CUDA cores in total
 - 32 DP units / SM = 1,792 DP units in total
- 28 TPCs (2 SMs per TPC)
- 6 GPCs

Maximum capacity would be 60 SMs and 30 TPCs

NVIDIA GTX 1080 (2016)

This is one NVIDIA Pascal GP104 streaming multi-processor (SM) unit



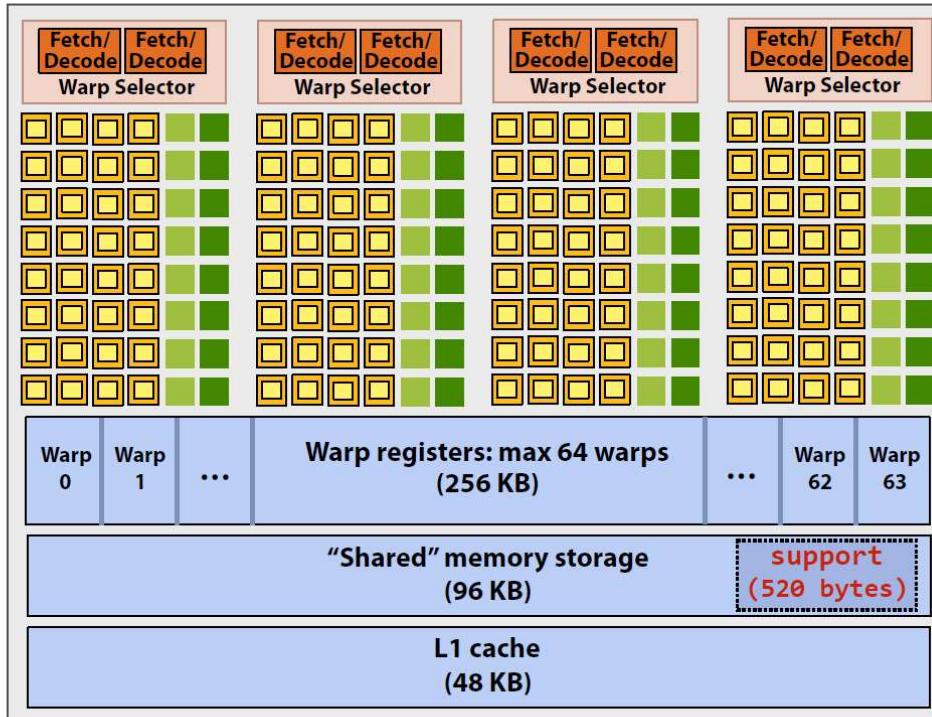
 = SIMD functional unit,
control shared across 32 units
(1 MUL-ADD per clock)

 = SIMD special function unit
(sin, cos, etc.)

SM resource limits:

- Max warp execution contexts: 64 (2,048 total CUDA threads)
- 96 KB of shared memory

Running a single thread block on a SM “core”



```
#define THREADS_PER_BLK 128

__global__ void convolve(int N, float* input,
                        float* output)
{
    __shared__ float support[THREADS_PER_BLK+2];
    int index = blockIdx.x * blockDim.x +
                threadIdx.x;

    support[threadIdx.x] = input[index];
    if (threadIdx.x < 2) {
        support[THREADS_PER_BLK+threadIdx.x]
            = input[index+THREADS_PER_BLK];
    }

    __syncthreads();

    float result = 0.0f; // thread-local
    for (int i=0; i<3; i++)
        result += support[threadIdx.x + i];

    output[index] = result;
}
```

Recall, CUDA kernels execute as SPMD programs

On NVIDIA GPUs groups of 32 CUDA threads share an instruction stream. These groups called “warps”.

A `convolve` thread block is executed by 4 warps (4 warps x 32 threads/warp = 128 CUDA threads per block)

(Warps are an important GPU implementation detail, but not a CUDA abstraction!)

SM core operation each clock:

- Select up to four runnable warps from 64 resident on SM core (thread-level parallelism)
- Select up to two runnable instructions per warp (instruction-level parallelism) *



NVIDIA Volta Architecture

2017/2018

NVIDIA Volta Architecture (2017/2018)



NVIDIA Volta SM

CC 7.0 Multiprocessor

- 64 FP32 + INT32 cores
- 32 FP64 cores
- 8 tensor cores
(FP16/FP32 mixed-precision)

4 partitions inside SM

- 16 FP32 + INT32 cores each
- 8 FP64 cores each
- 8 LD/ST units each
- 2 tensor cores each
- Each has: warp scheduler, dispatch unit, register file





Tensor Cores

Mixed-precision, fast matrix-matrix multiply and accumulate

$$\mathbf{D} = \left(\begin{array}{cccc} \mathbf{A}_{0,0} & \mathbf{A}_{0,1} & \mathbf{A}_{0,2} & \mathbf{A}_{0,3} \\ \mathbf{A}_{1,0} & \mathbf{A}_{1,1} & \mathbf{A}_{1,2} & \mathbf{A}_{1,3} \\ \mathbf{A}_{2,0} & \mathbf{A}_{2,1} & \mathbf{A}_{2,2} & \mathbf{A}_{2,3} \\ \mathbf{A}_{3,0} & \mathbf{A}_{3,1} & \mathbf{A}_{3,2} & \mathbf{A}_{3,3} \end{array} \right) \left(\begin{array}{cccc} \mathbf{B}_{0,0} & \mathbf{B}_{0,1} & \mathbf{B}_{0,2} & \mathbf{B}_{0,3} \\ \mathbf{B}_{1,0} & \mathbf{B}_{1,1} & \mathbf{B}_{1,2} & \mathbf{B}_{1,3} \\ \mathbf{B}_{2,0} & \mathbf{B}_{2,1} & \mathbf{B}_{2,2} & \mathbf{B}_{2,3} \\ \mathbf{B}_{3,0} & \mathbf{B}_{3,1} & \mathbf{B}_{3,2} & \mathbf{B}_{3,3} \end{array} \right) + \left(\begin{array}{cccc} \mathbf{C}_{0,0} & \mathbf{C}_{0,1} & \mathbf{C}_{0,2} & \mathbf{C}_{0,3} \\ \mathbf{C}_{1,0} & \mathbf{C}_{1,1} & \mathbf{C}_{1,2} & \mathbf{C}_{1,3} \\ \mathbf{C}_{2,0} & \mathbf{C}_{2,1} & \mathbf{C}_{2,2} & \mathbf{C}_{2,3} \\ \mathbf{C}_{3,0} & \mathbf{C}_{3,1} & \mathbf{C}_{3,2} & \mathbf{C}_{3,3} \end{array} \right)$$

FP16 or FP32 FP16 FP16 or FP32

From this, build larger sizes, higher dimensionalities, ...

[+Tensor cores on later architectures add more data types/precisions!]

NVIDIA Volta Architecture (2017/2018)



Total chip capacity on Tesla V100 (GV100 architecture)

- 80 SMs
 - 64 FP32 cores / SM = 5,120 FP32 cores in total
 - 64 INT32 cores / SM = 5,120 INT32 cores in total
 - 32 FP64 cores / SM = 2,560 FP64 cores in total
 - 4 FP16/FP32 mixed-prec. tensor cores = 650 tensor cores in total
- 40 TPCs (2 SMs per TPC)
- 6 GPCs

Maximum capacity would be 84 SMs and 42 TPCs

Kepler – Volta Specs

Tesla Product	Tesla K40	Tesla M40	Tesla P100	Tesla V100
GPU	GK180 (Kepler)	GM200 (Maxwell)	GP100 (Pascal)	GV100 (Volta)
SMs	15	24	56	80
TPCs	15	24	28	40
FP32 Cores / SM	192	128	64	64
FP32 Cores / GPU	2880	3072	3584	5120
FP64 Cores / SM	64	4	32	32
FP64 Cores / GPU	960	96	1792	2560
Tensor Cores / SM	NA	NA	NA	8
Tensor Cores / GPU	NA	NA	NA	640
GPU Boost Clock	810/875 MHz	1114 MHz	1480 MHz	1455 MHz
Peak FP32 TFLOP/s*	5.04	6.8	10.6	15
Peak FP64 TFLOP/s*	1.68	.21	5.3	7.5
Peak Tensor Core TFLOP/s*	NA	NA	NA	120
Texture Units	240	192	224	320
Memory Interface	384-bit GDDR5	384-bit GDDR5	4096-bit HBM2	4096-bit HBM2
Memory Size	Up to 12 GB	Up to 24 GB	16 GB	16 GB
L2 Cache Size	1536 KB	3072 KB	4096 KB	6144 KB
Shared Memory Size / SM	16 KB/32 KB/48 KB	96 KB	64 KB	Configurable up to 96 KB
Register File Size / SM	256 KB	256 KB	256 KB	256KB
Register File Size / GPU	3840 KB	6144 KB	14336 KB	20480 KB
TDP	235 Watts	250 Watts	300 Watts	300 Watts
Transistors	7.1 billion	8 billion	15.3 billion	21.1 billion
GPU Die Size	551 mm ²	601 mm ²	610 mm ²	815 mm ²
Manufacturing Process	28 nm	28 nm	16 nm FinFET+	12 nm FFN



Turing (vs. Pascal)

Apart from RT cores, Volta and Turing are very similar
(and both have compute capability 7.x: Volta: 7.0, Turing: 7.5)

GPU Features	GeForce GTX 1080	GeForce RTX 2080	Quadro P5000	Quadro RTX 5000
Architecture	Pascal	Turing	Pascal	Turing
GPCs	4	6	4	6
TPCs	20	23	20	24
SMs	20	46	20	48
CUDA Cores / SM	128	64	128	64
CUDA Cores / GPU	2560	2944	2560	3072
Tensor Cores / SM	NA	8	NA	8
Tensor Cores / GPU	NA	368	NA	384
RT Cores	NA	46	NA	48

TU104

TU104



NVIDIA Turing Architecture

2018/2019

TU102, TU104, TU106, TU116, ...
(RTX 2070, 2080, 2080Ti, Tesla T4, ...)

NVIDIA Turing Architecture (2018/2019)



TU 102

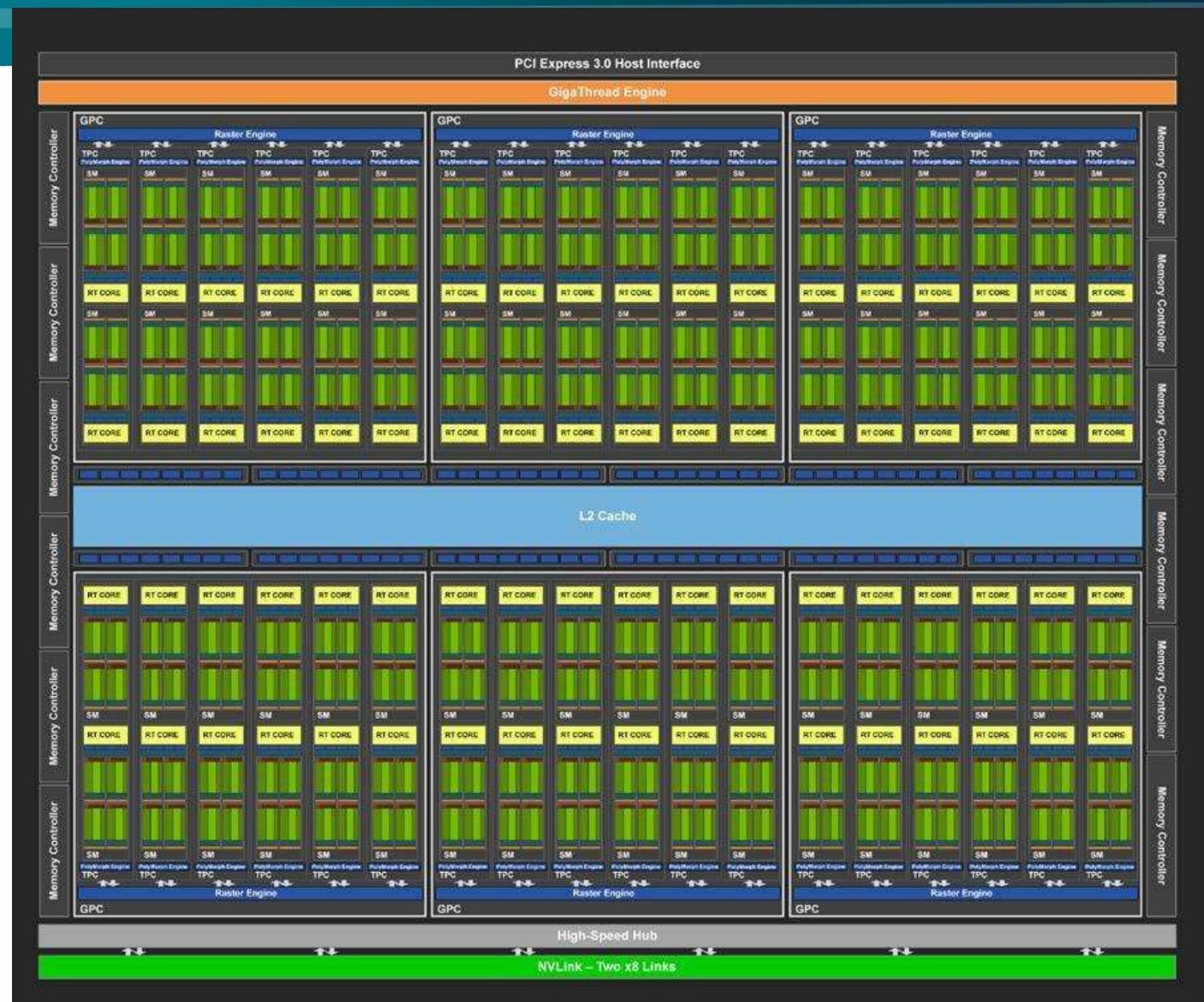
(Geforce:

RTX 2080 Ti,

Quadro:

RTX 6000,

RTX 8000, ...)





NVIDIA Turing Architecture (2018/2019)

TU 104

(Geforce:

RTX 2080,

Quadro:

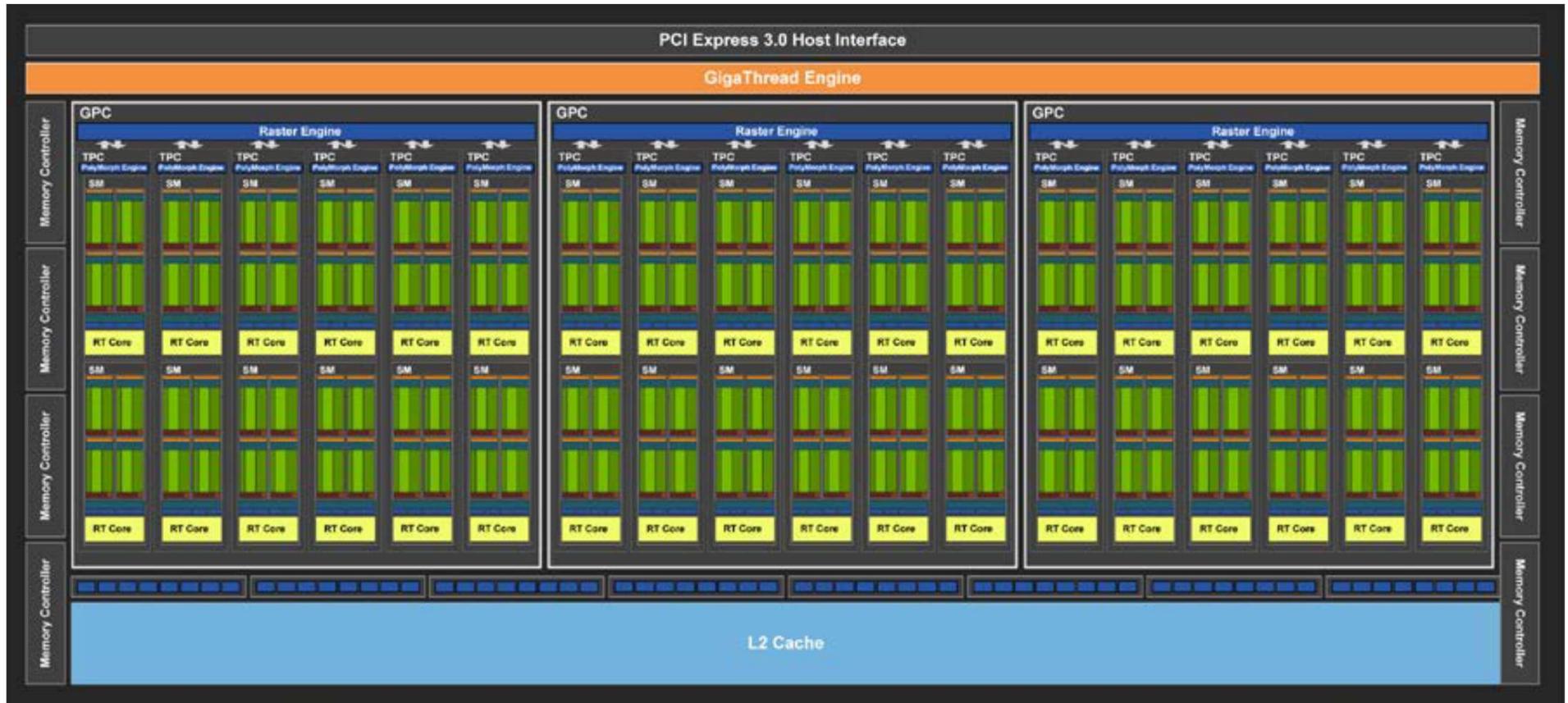
RTX 5000, ...)



NVIDIA Turing Architecture (2018/2019)



TU 106 (Geforce RTX 2070, ...)



NVIDIA Turing SM

CC 7.5 Multiprocessor

- 64 FP32 + INT32 cores
- 2 (!) FP64 cores
- 8 Turing tensor cores
(FP16/32, INT4/8 mixed-precision)
- 1 RT (ray tracing) core

4 partitions inside SM

- 16 FP32 + INT32 cores each
- 4 LD/ST units each
- 2 Turing tensor cores each
- Each has: warp scheduler,
dispatch unit, 16K register file





NVIDIA Ampere Architecture

2020

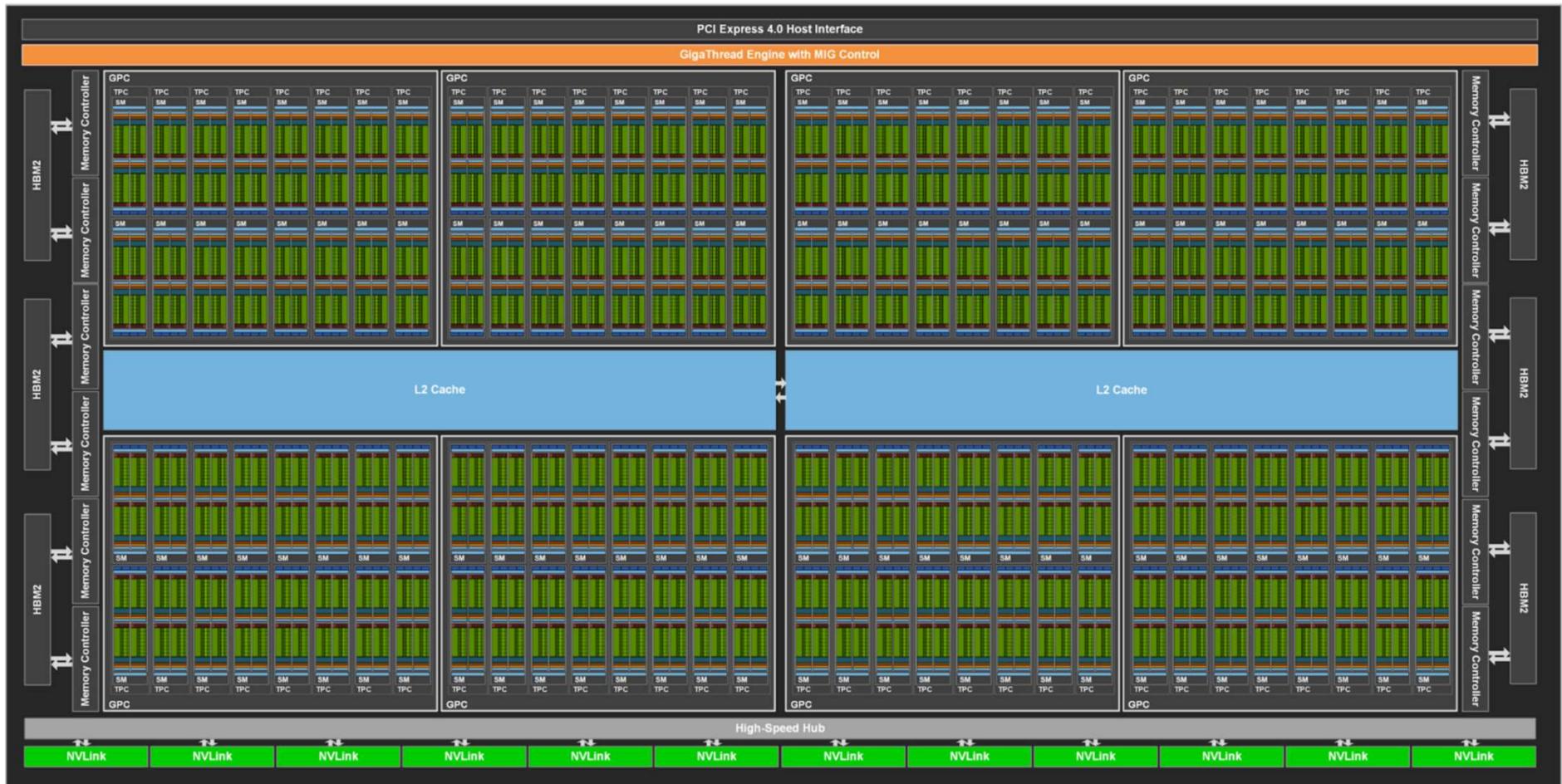
GA100, GA102, GA104, ...
(A100, RTX 3070, RTX 3080, RTX 3090, ...)

NVIDIA Ampere GA100 Architecture (2020)



GA 100 (A100 Tensor Core GPU)

Full GPU: 128 SMs (in 8 GPCs/64 TPCs)



NVIDIA GA100 SM

Multiprocessor: SM

- 64 FP32 + 64 INT32 cores
- 32 FP64 cores
- 4 3rd gen tensor cores
- 1 2nd gen RT (ray tracing) core

4 partitions inside SM

- 16 FP32 + 16 INT32 cores
- 8 FP64 cores
- 8 LD/ST units each
- 1 3rd gen tensor core each
- Each has: warp scheduler, dispatch unit, 16K register file



NVIDIA Ampere GA10x Architecture (2020)



GA 102 (RTX 3070, 3080, 3090)

Full GPU: 84 SMs (in 7 GPCs/42 TPCs)



NVIDIA GA10x SM

Multiprocessor: SM

- 128 (64+64) FP32 + 64 INT32 cores
- 2 (!) FP64 cores
- 4 3rd gen tensor cores
- 1 2nd gen RT (ray tracing) core

4 partitions inside SM

- 16+16 FP32 + 16 INT32 cores
- 4 LD/ST units each
- 1 3rd gen tensor core each
- Each has: warp scheduler, dispatch unit, 16K register file



Turing vs. Ampere GA102



Graphics Card	GeForce RTX 2080 Founders Edition	GeForce RTX 2080 Super Founders Edition	GeForce RTX 3080 10 GB Founders Edition
GPU Codename	TU104	TU104	GA102
GPU Architecture	NVIDIA Turing	NVIDIA Turing	NVIDIA Ampere
GPCs	6	6	6
TPCs	23	24	34
SMs	46	48	68
CUDA Cores / SM	64	64	128
CUDA Cores / GPU	2944	3072	8704
Tensor Cores / SM	8 (2nd Gen)	8 (2nd Gen)	4 (3rd Gen)
Tensor Cores / GPU	368	384 (2nd Gen)	272 (3rd Gen)
RT Cores	46 (1st Gen)	48 (1st Gen)	68 (2nd Gen)
GPU Boost Clock (MHz)	1800	1815	1710
Peak FP32 TFLOPS (non-Tensor) ¹	10.6	11.2	29.8
Peak FP16 TFLOPS (non-Tensor) ¹	21.2	22.3	29.8
Peak BF16 TFLOPS (non-Tensor) ¹	NA	NA	29.8
Peak INT32 TOPS (non-Tensor) ^{1,3}	10.6	11.2	14.9



Turing vs. Ampere GA102

Peak FP16 Tensor TFLOPS with FP16 Accumulate ¹	84.8	89.2	119/238 ²
Peak FP16 Tensor TFLOPS with FP32 Accumulate ¹	42.4	44.6	59.5/119 ²
Peak BF16 Tensor TFLOPS with FP32 Accumulate ¹	NA	NA	59.5/119 ²
Peak TF32 Tensor TFLOPS ¹	NA	NA	29.8/59.5 ²
Peak INT8 Tensor TOPS ¹	169.6	178.4	238/476 ²
Peak INT4 Tensor TOPS ¹	339.1	356.8	476/952 ²
Frame Buffer Memory Size and Type	8192 MB GDDR6	8192 MB GDDR6	10240 MB GDDR6X
Memory Interface	256-bit	256-bit	320-bit
Memory Clock (Data Rate)	14 Gbps	15.5 Gbps	19 Gbps
Memory Bandwidth	448 GB/sec	496 GB/sec	760 GB/sec
ROPs	64	64	96
Pixel Fill-rate (Gigapixels/sec)	115.2	116.2	164.2
Texture Units	184	192	272
Texel Fill-rate (Gigatexels/sec)	331.2	348.5	465
L1 Data Cache/Shared Memory	4416 KB	4608 KB	8704 KB



Turing vs. Ampere GA102

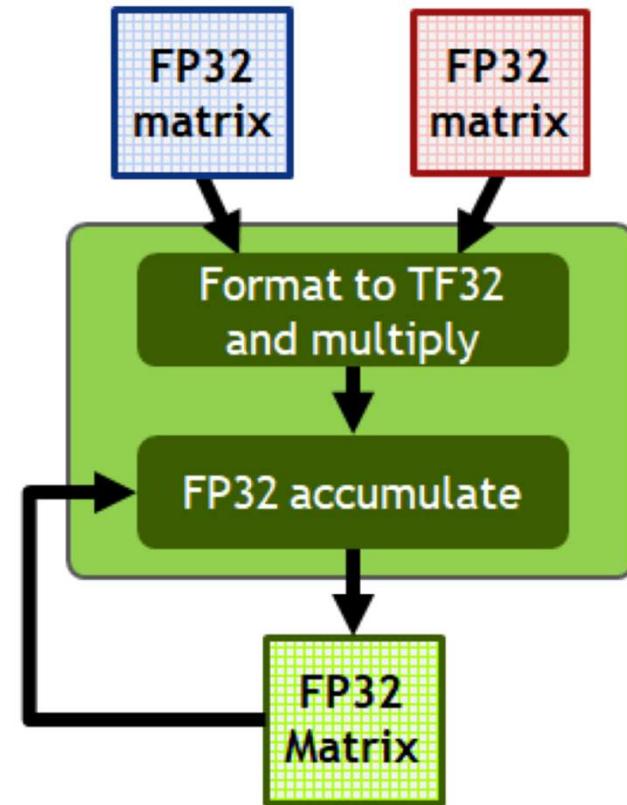
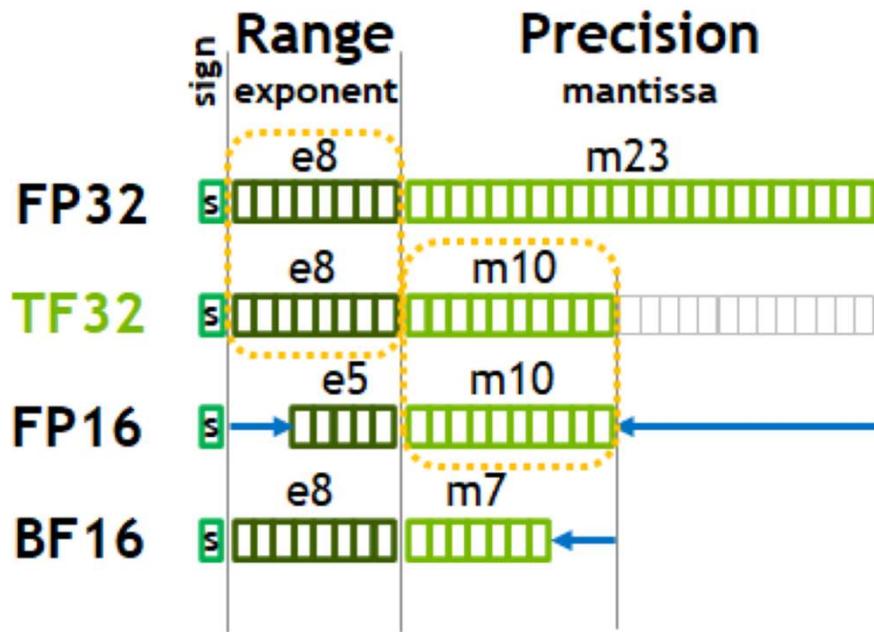
L2 Cache Size	4096 KB	4096 KB	5120 KB
Register File Size	11776 KB	12288 KB	17408 KB
TGP (Total Graphics Power)	225 W	250 W	320W
Transistor Count	13.6 Billion	13.6 Billion	28.3 Billion
Die Size	545 mm ²	545 mm ²	628.4 mm ²
Manufacturing Process	TSMC 12 nm FFN (FinFET NVIDIA)	TSMC 12 nm FFN (FinFET NVIDIA)	Samsung 8 nm 8N NVIDIA Custom Process

1. Peak rates are based on GPU Boost Clock.
2. Effective TOPS / TFLOPS using the new Sparsity Feature
3. TOPS = IMAD-based integer math

Tensor Cores: Many Mixed Precision Options



New in Ampere: TF32, BF16, FP64



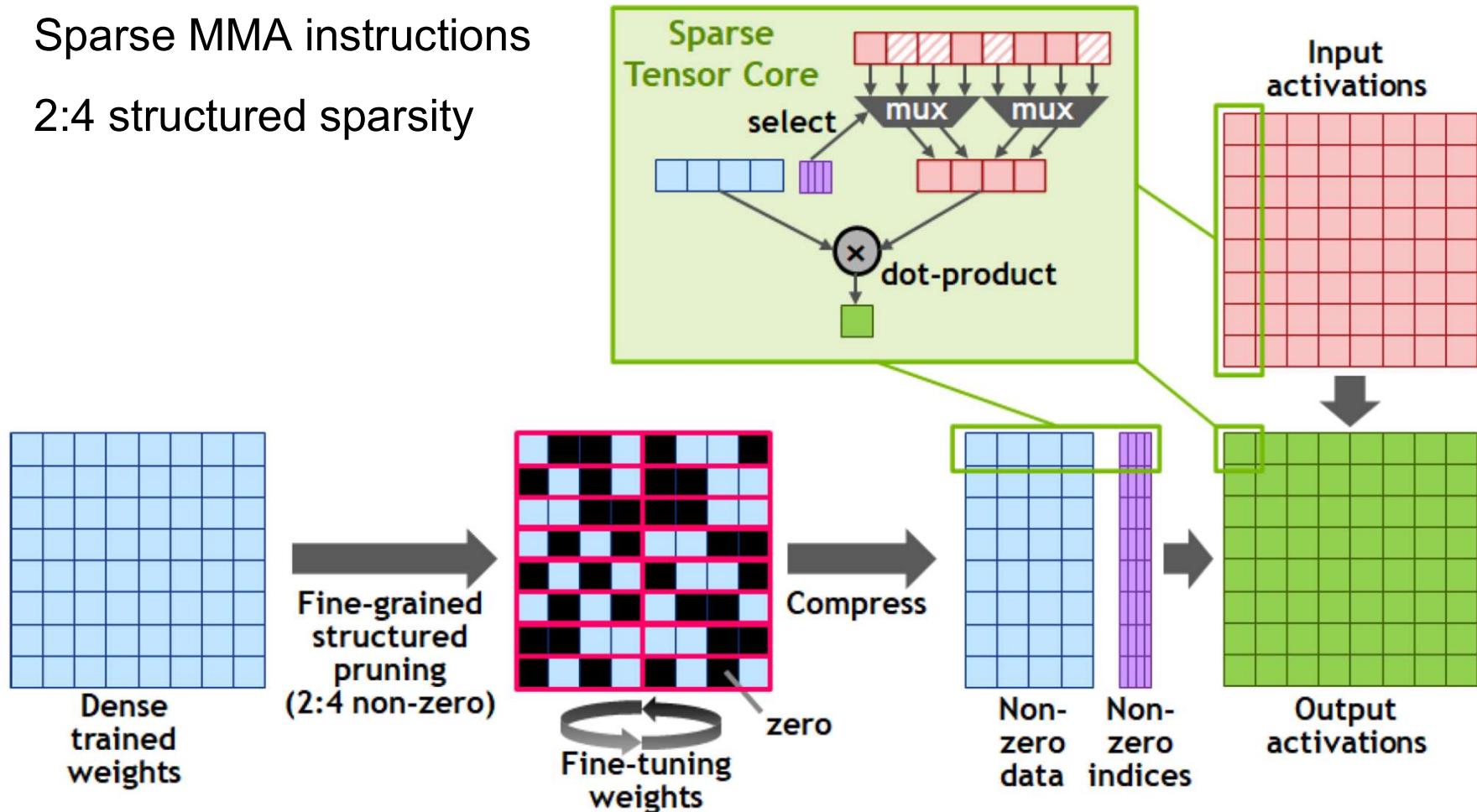
plus FP64 (new in Ampere; GA100 only)

plus INT4/INT8/binary data types (already introduced in Turing)



Tensor Cores: Sparsity Support

Sparse MMA instructions
2:4 structured sparsity



Thank you.