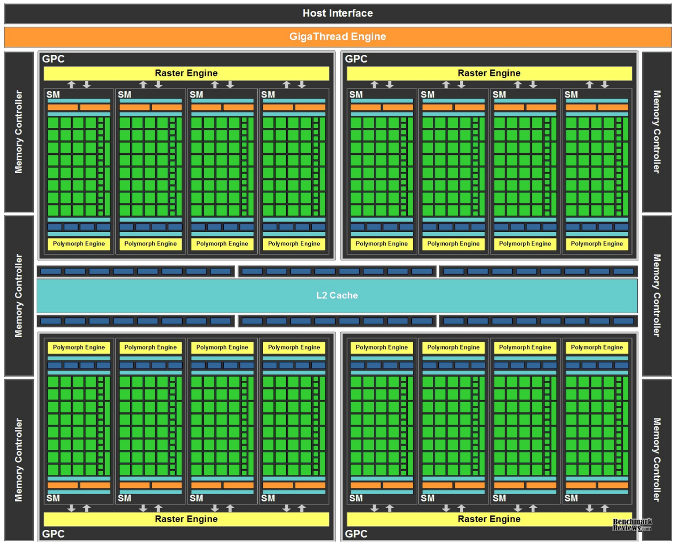
GPU 101

Oregon State University

Mike Bailey

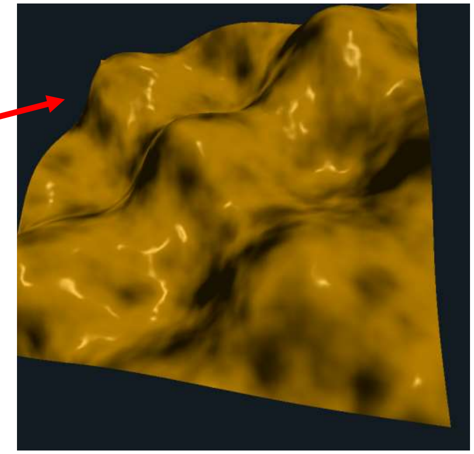
mjb@cs.oregonstate.edu

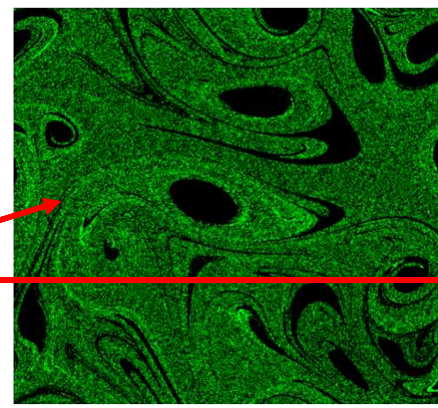


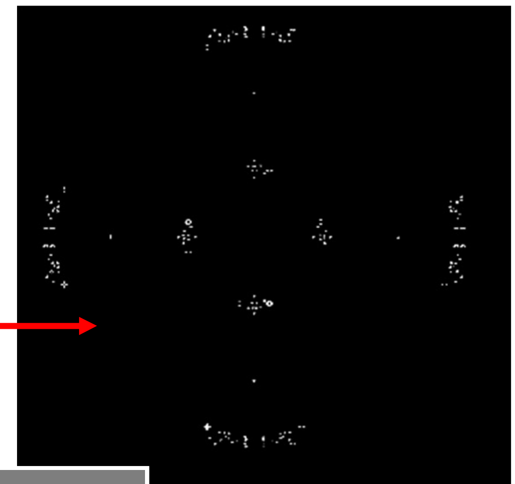
How Have You Been Able to Gain Access to GPU Power?

There have been three ways:

1. Write a graphics display program (≥ 1985)



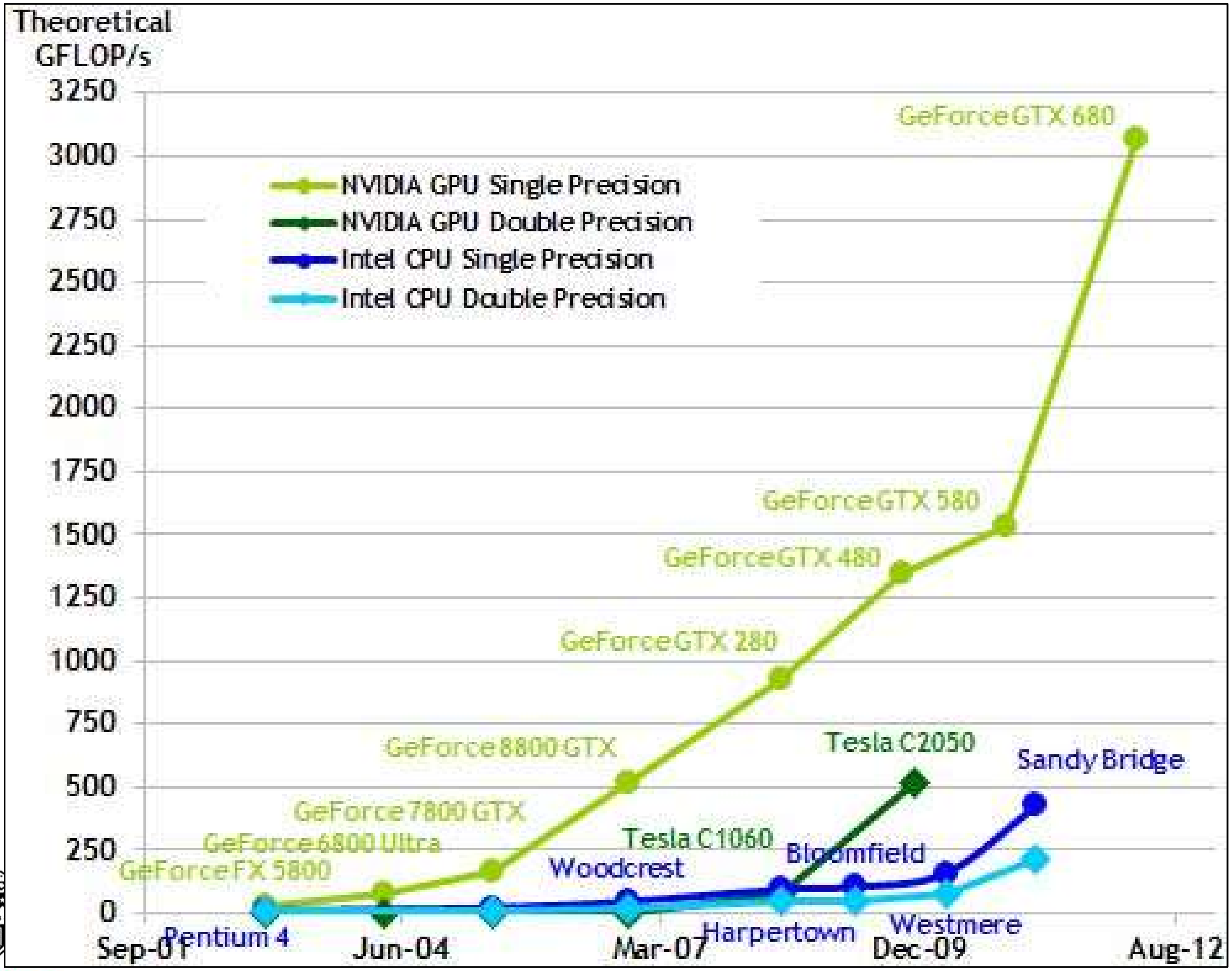




2. Write an application that looks like a graphics display program, but uses the fragment shader to do some per-node computation (≥ 2002)

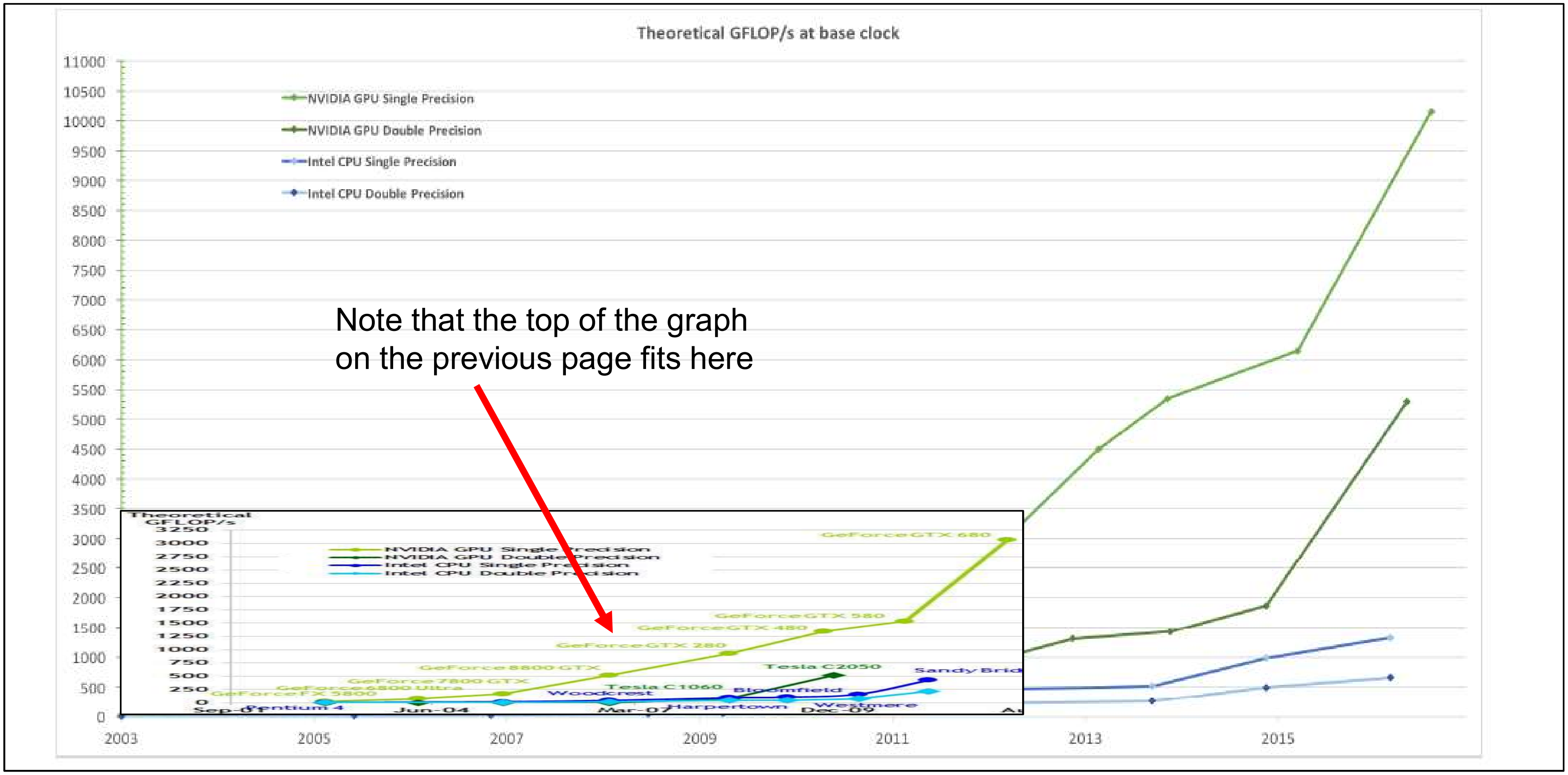
3. Write in OpenCL or CUDA, which looks like C++ (≥ 2006)

Why do we care about GPU Programming?
A History of GPU vs. CPU Performance



NVIDIA

Why do we care about GPU Programming? A History of GPU vs. CPU Performance



Oregon State University
Computer Graphics

NVIDIA

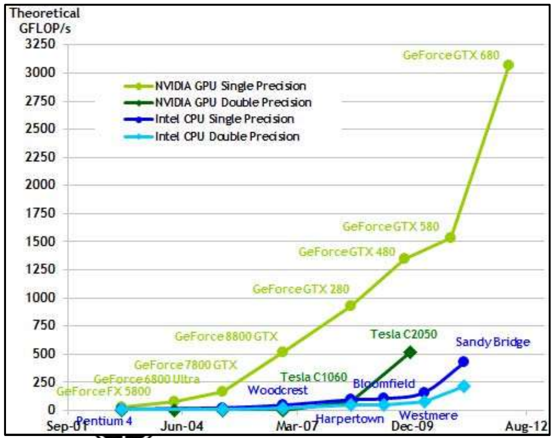
The “Core-Score”. How can this be?

Why have GPUs Been Outpacing CPUs in Performance?

Due to the nature of graphics computations, GPU chips are customized to stream **regular data**. General CPU chips must be able to handle **irregular data**.

Another reason is that GPU chips do not need the significant amount of **cache** space that occupies much of the real estate on general-purpose CPU chips. The GPU die real estate can then be re-targeted to hold more cores and thus to produce more processing power.

NVIDIA



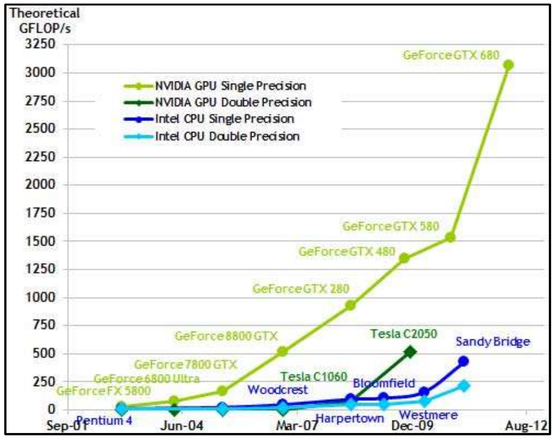
Why have GPUs Been Outpacing CPUs in Performance?

Another reason is that general CPU chips contain on-chip logic to do **branch prediction** and **out-of-order execution.** This, too, takes up chip die space.

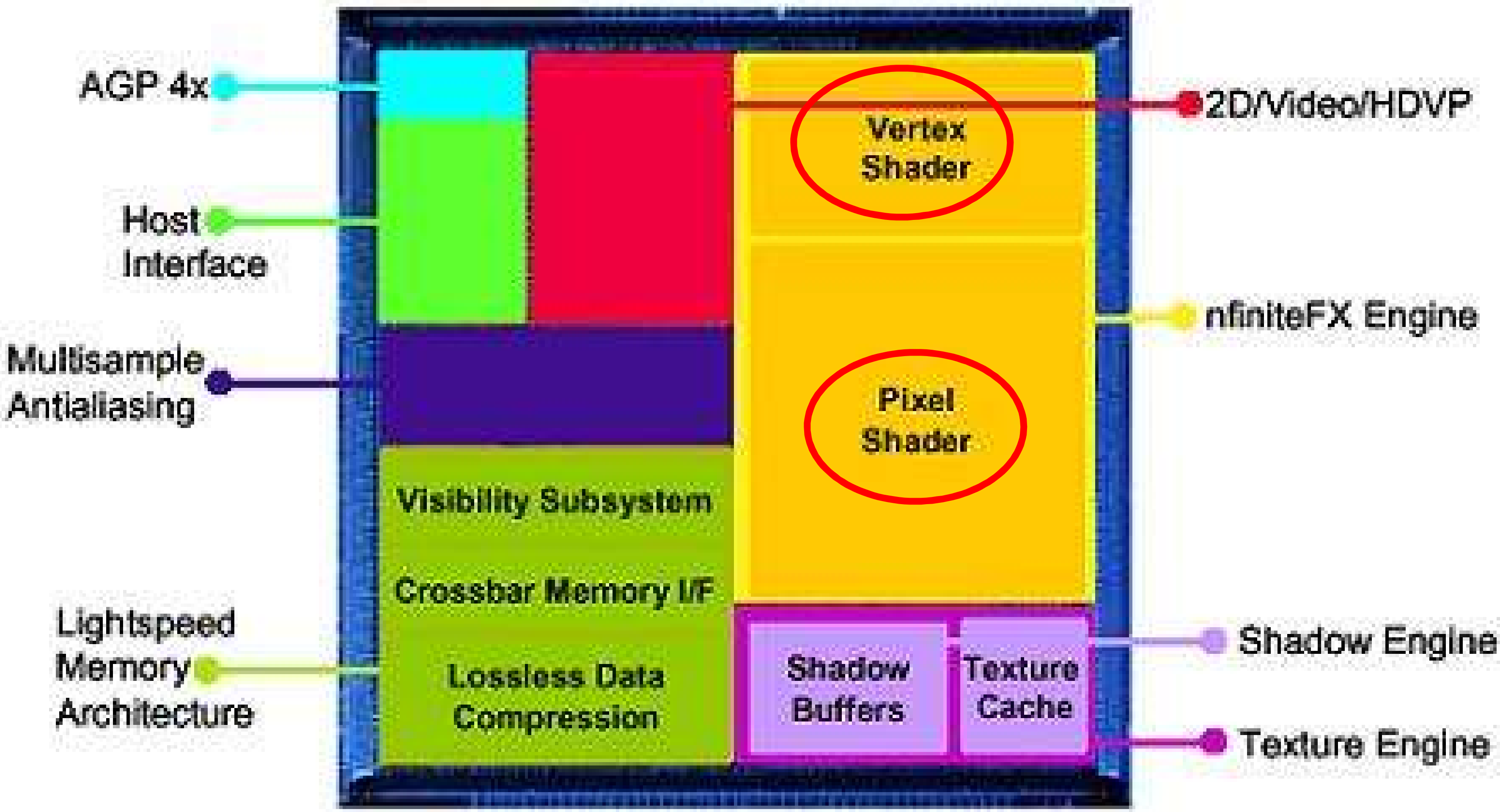
But, CPU chips can handle more general-purpose computing tasks.

So, which is better, a CPU or a GPU?

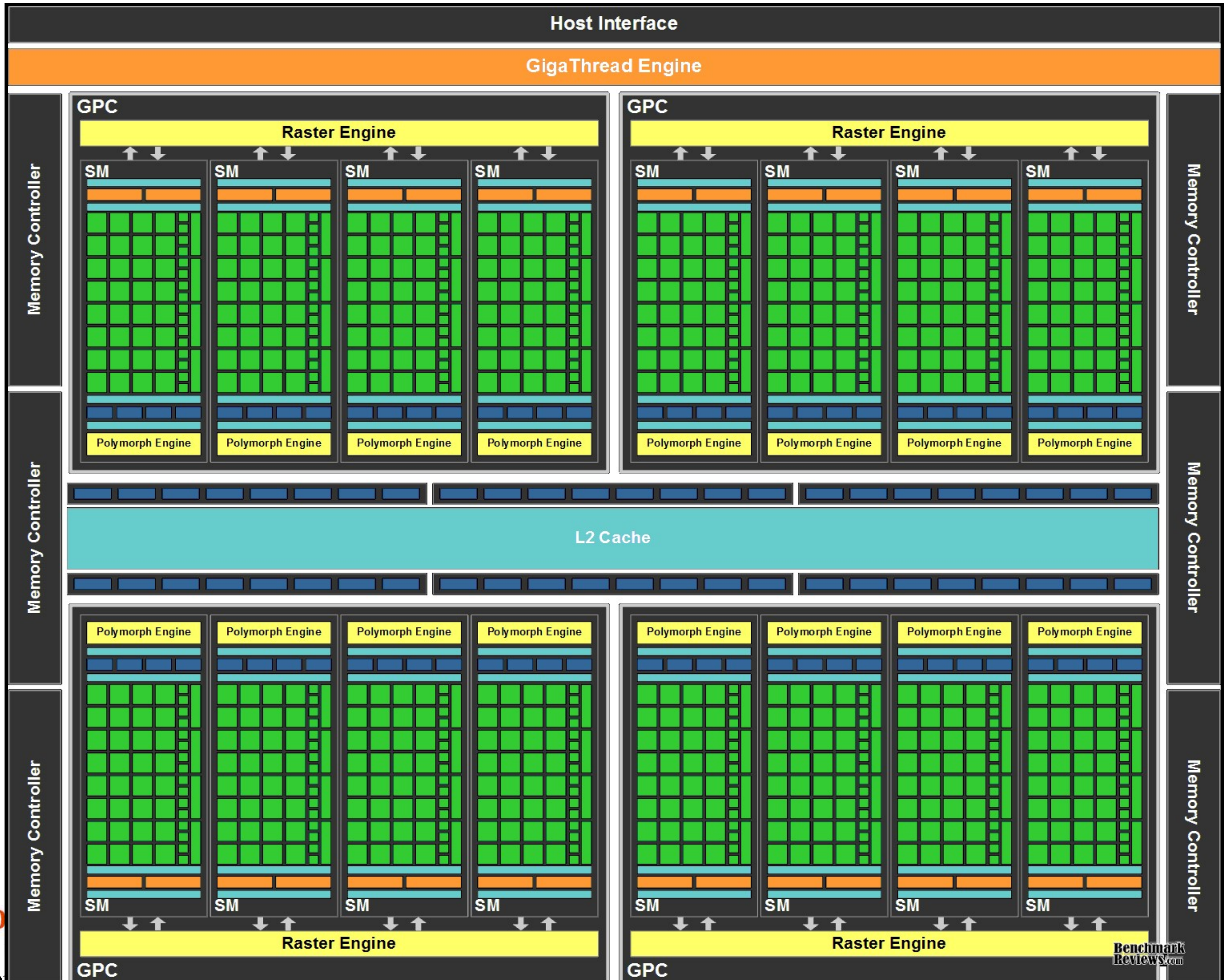
It depends on what you are trying to do!



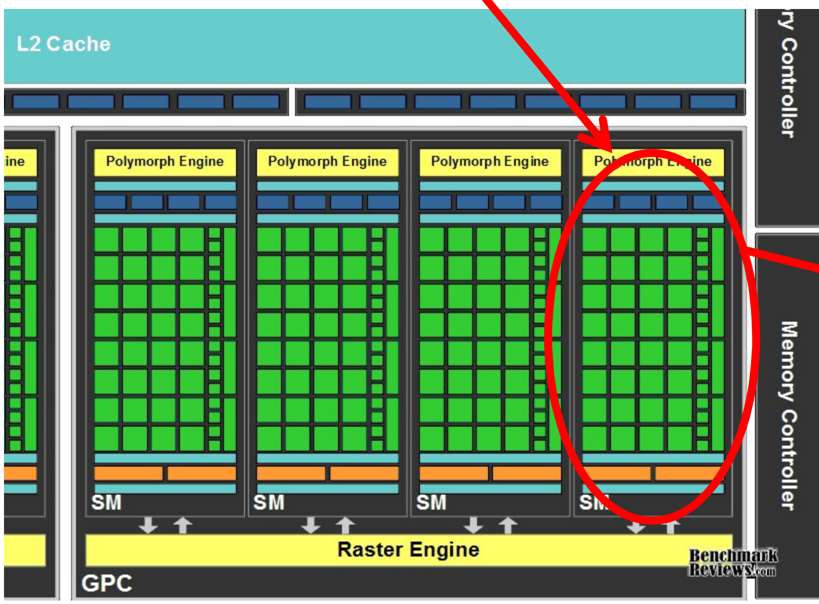
Originally, GPU Devices were very task-specific

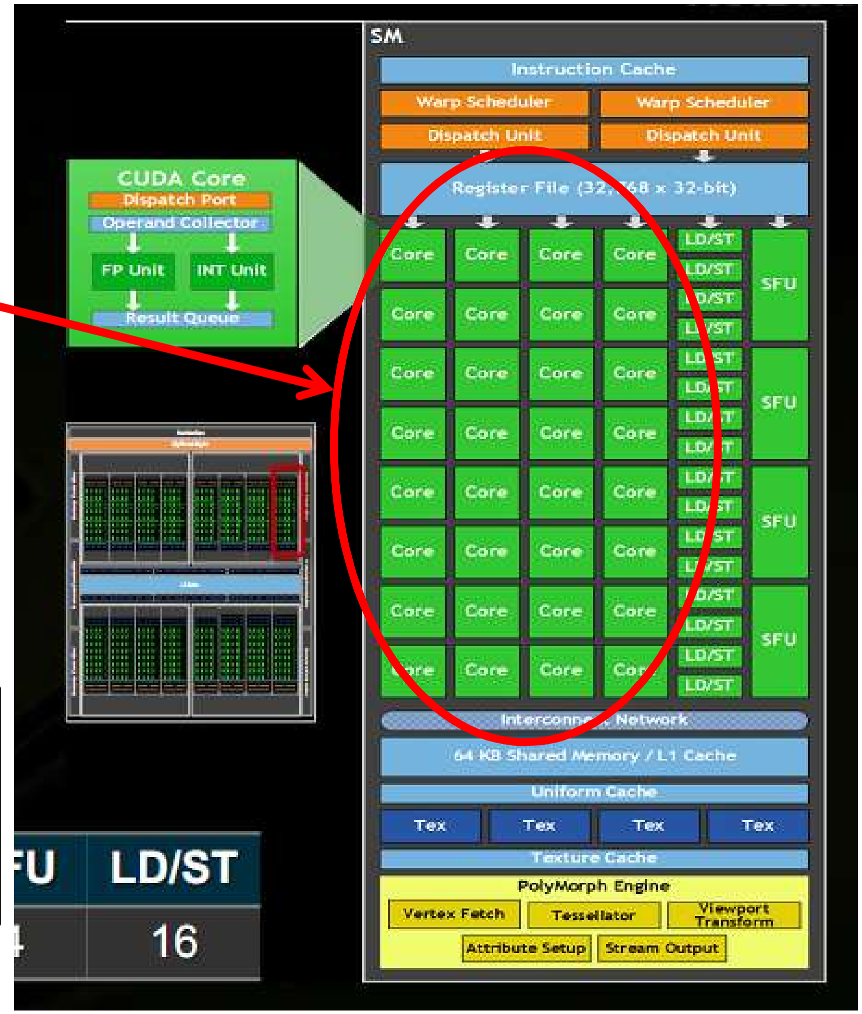


Today’s GPU Devices are not task-specific



Consider the architecture of the NVIDIA Tesla V100's that we have in our *DGX System*



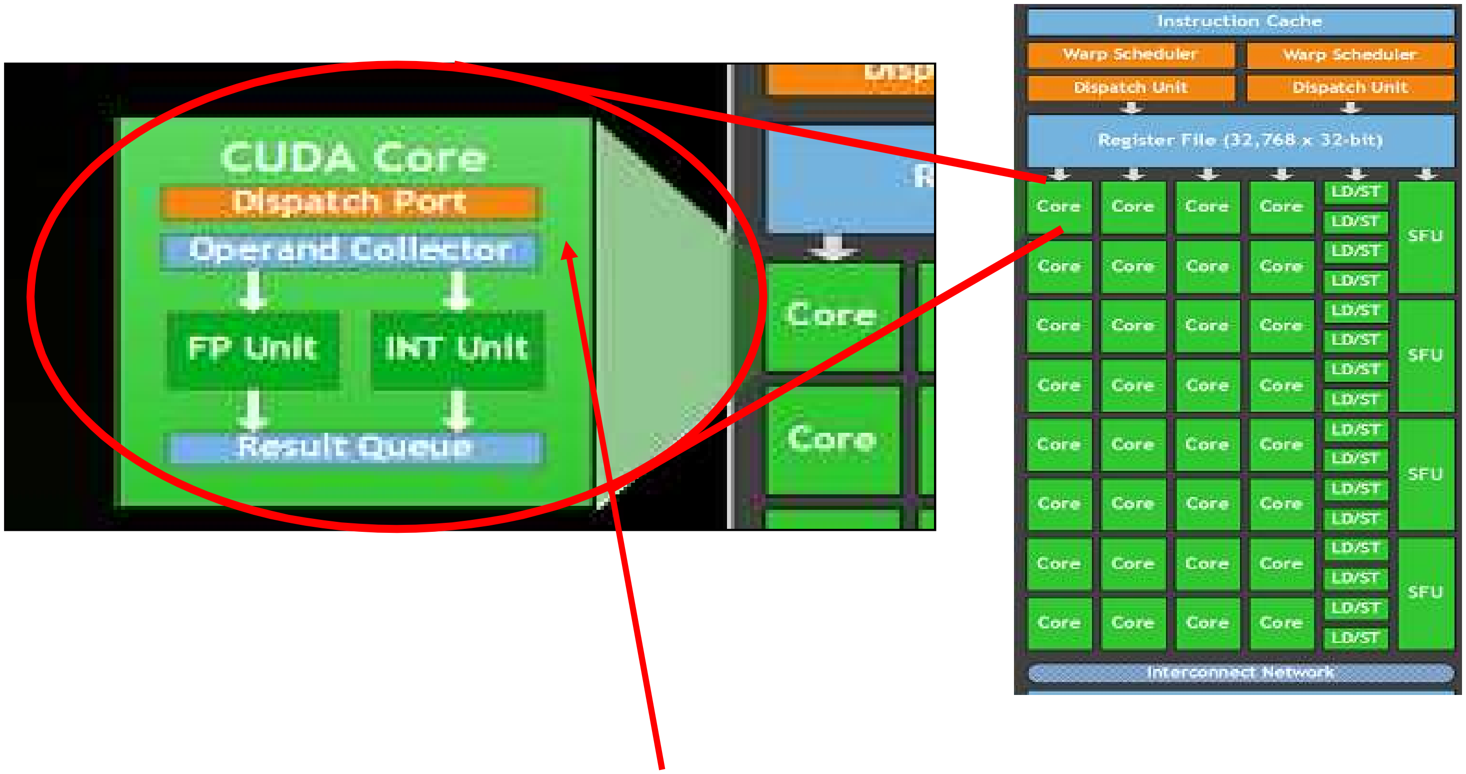


84 Streaming Multiprocessors (SMs) / chip

64 cores / SM

Wow! **5,396** cores / chip? Really?

What is a “Core” in the GPU Sense?

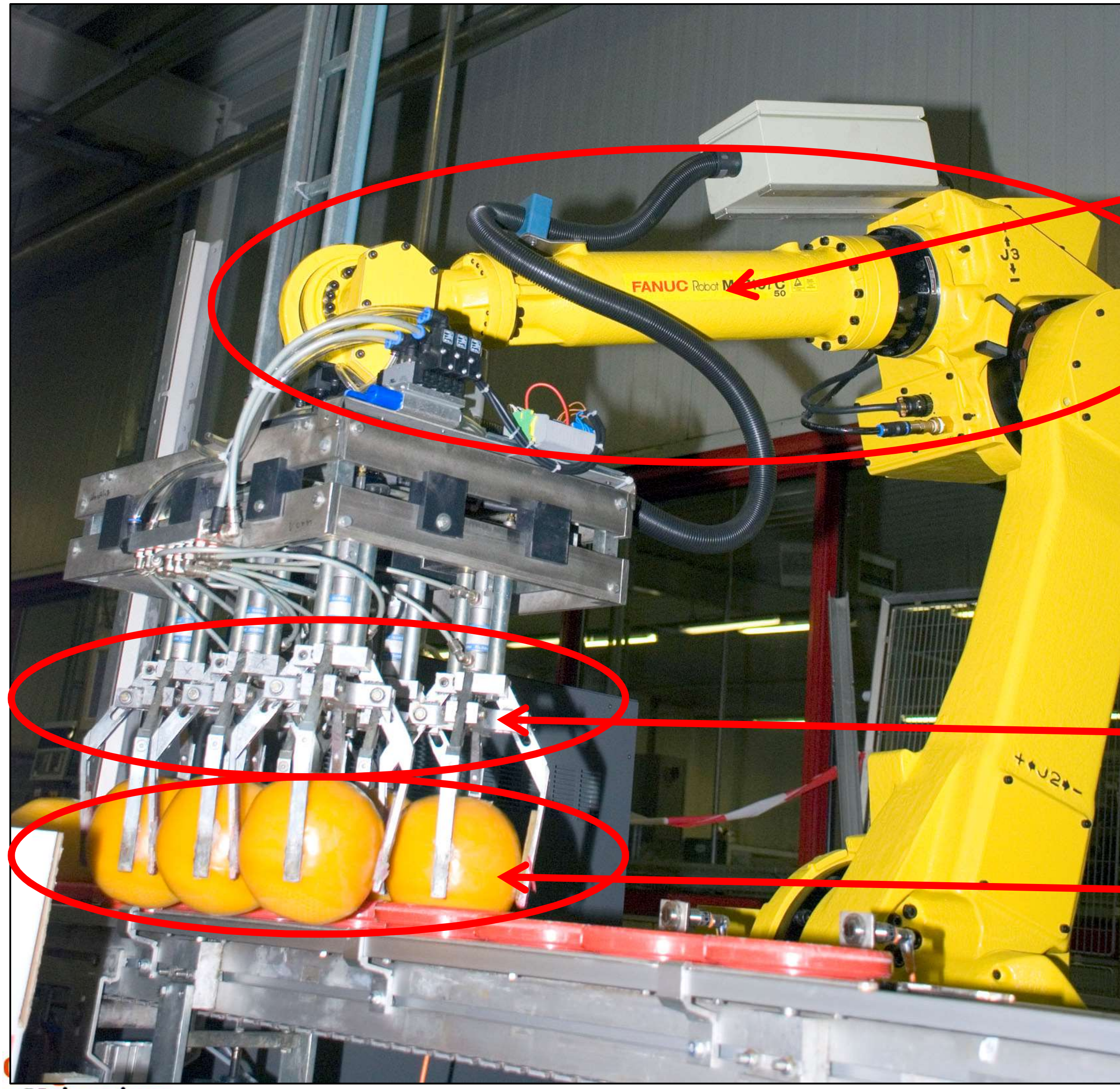


Look closely, and you’ll see that NVIDIA really calls these “CUDA Cores”

Look even more closely and you’ll see that these CUDA Cores have no control logic – they are **pure compute units**. (The surrounding SM has the control logic.)

Other vendors refer to these as “Lanes”. You might also think of them as 64-way SIMD.

A Mechanical Equivalent…



http://news.cision.com

How Many Robots Do You See Here?

12? 72? Depends what you count as a “robot”.

A Spec Sheet Example

NVIDIA Card 4000 Series	Number of CUDA Cores	Size of Power Supply **	Memory Type	Memory Interface Width	Memory Bandwidth GB/sec	Base Clock Speed	Boost Clock Speed	NOTES
RTX-4080	9728	750 watt	GDDR6X	256 bit	716.8 GB/s	2.21 GHz	2.51 GHz	16 GB of Memory
RTX-4090	16384	850 watt	GDDR6X	384 bit	1008 GB/s	2.23 GHz	2.52 GHz	24 GB of Memory

NVIDIA Card 3000 Series	Number of CUDA Cores	Size of Power Supply **	Memory Type	Memory Interface Width	Memory Bandwidth GB/sec	Base Clock Speed	Boost Clock Speed	NOTES
RTX-3050	2560	550 watt	GDDR6	128 bit	224 GB/s	1550 MHz	1780 MHz	Standard with 8 GB of Memory
RTX-3060	3584	550 watt	GDDR6	192 bit	384 GB/s	1320 MHz	1780 MHz	Standard with 12 GB of Memory
RTX-3060 Ti	4864	600 watt	GDDR6	256 bit	448 GB/s	1410 MHz	1670 MHz	Standard with 8 GB of Memory
RTX-3070	5888	650 watt	GDDR6	256 bit	448 GB/s	1580 MHz	1770 MHz	Standard with 8 GB of Memory
RTX-3070 Ti	6144	750 watt	GDDR6X	256 bit	608 GB/s	1500 MHz	1730 MHz	Standard with 8 GB of Memory
RTX-3080	8704	750 watt	GDDR6X	320 bit	760 GB/s	1440 MHz	1710 MHz	Standard with 10 GB of Memory
RTX-3080 Ti	10240	750 watt	GDDR6X	384 bit	912 GB/s	1370 MHz	1670 MHz	Standard with 12 GB of Memory
RTX-3090	10496	750 watt	GDDR6X	384 bit	936 GB/s	1400 MHz	1700 MHz	Standard with 24 GB of Memory
RTX-3090 Ti	10572	850 watt	GDDR6X	384 bit	936 GB/s	1670 MHz	1860 MHz	Standard with 24 GB of Memory

NVIDIA Card 2000 Series	Number of CUDA Cores	Size of Power Supply **	Memory Type	Memory Interface Width	Memory Bandwidth GB/sec	Base Clock Speed	Boost Clock Speed	NOTES
RTX-2060	1920	500 watt	GDDR6	192 bit	336 GB/s	1365 MHz	1680 MHz	Standard with 6 GB of Memory
RTX-2060 Super	2176	550 watt	GDDR6	256 bit	448 GB/s	1470 MHz	1650 MHz	Standard with 8 GB of Memory
RTX-2070	2304	550 watt	GDDR6	256 bit	448 GB/s	1410 MHz	1620 MHz	Standard with 8 GB of Memory
RTX-2070 Super	2560	650 watt	GDDR6	256 bit	448 GB/s	1605 MHz	1770 MHz	Standard with 8 GB of Memory
RTX-2080	2944	650 watt	GDDR6	256 bit	448 GB/s	1515 MHz	1710 MHz	Standard with 8 GB of Memory
RTX-2080 Super	3072	650 watt	GDDR6	256 bit	496 GB/s	1650 MHz	1815 MHz	Standard with 8 GB of Memory
RTX-2080 Ti	4352	650 watt	GDDR6	352 bit	616 GB/s	1350 MHz	1545 MHz	Standard with 11 GB of Memory
Titan RTX	4608	650 watt	GDDR6	384 bit	672 GB/s	1350 MHz	1770 MHz	Standard with 24 GB of Memory

NVIDIA

NVIDIA 4090 Spec Sheet

Graphics Processor

GPU Name:	AD102
GPU Variant:	AD102-300-A1
Architecture:	Ada Lovelace
Foundry:	TSMC
Process Size:	4 nm
Transistors:	76,300 million
Density:	125.5M / mm²
Die Size:	608 mm²

Clock Speeds

Base Clock:	2235 MHz
Boost Clock:	2520 MHz
Memory Clock:	1313 MHz 21 Gbps effective

Board Design

Slot Width:	Triple-slot
Length:	304 mm 12 inches
Width:	137 mm 5.4 inches
Height:	61 mm 2.4 inches
TDP:	450 W
Suggested PSU:	850 W
Outputs:	1x HDMI 2.1 3x DisplayPort 1.4a
Power Connectors:	1x 16-pin
Board Number:	PG139 SKU 330

Graphics Card

Release Date:	Sep 20th, 2022
Availability:	Oct 12th, 2022
Generation:	GeForce 40
Predecessor:	GeForce 30
Production:	Active
Launch Price:	1,599 USD
Current Price:	Amazon / Newegg
Bus Interface:	PCIe 4.0 x16
Reviews:	65 in our database

Theoretical Performance

Pixel Rate:	443.5 GPixel/s
Texture Rate:	1,290 GTexel/s
FP16 (half):	82.58 TFLOPS (1:1)
FP32 (float):	82.58 TFLOPS
FP64 (double):	1,290 GFLOPS (1:64)

Relative Performance

GeForce RTX 3080	54%
Radeon RX 6900 XT	56%
GeForce RTX 3080 Ti	60%
Radeon RX 6950 XT	60%
GeForce RTX 3090	61%
GeForce RTX 4070 Ti	63%
GeForce RTX 3090 Ti	69%
Radeon RX 7900 XT	69%
GeForce RTX 4080	80%
Radeon RX 7900 XTX	82%
GeForce RTX 4090	100%

Based on TPU review data: "Performance Summary" at 1920x1080, 4K for 2080 Ti and faster.

Memory

Memory Size:	24 GB
Memory Type:	GDDR6X
Memory Bus:	384 bit
Bandwidth:	1,008 GB/s

Graphics Features

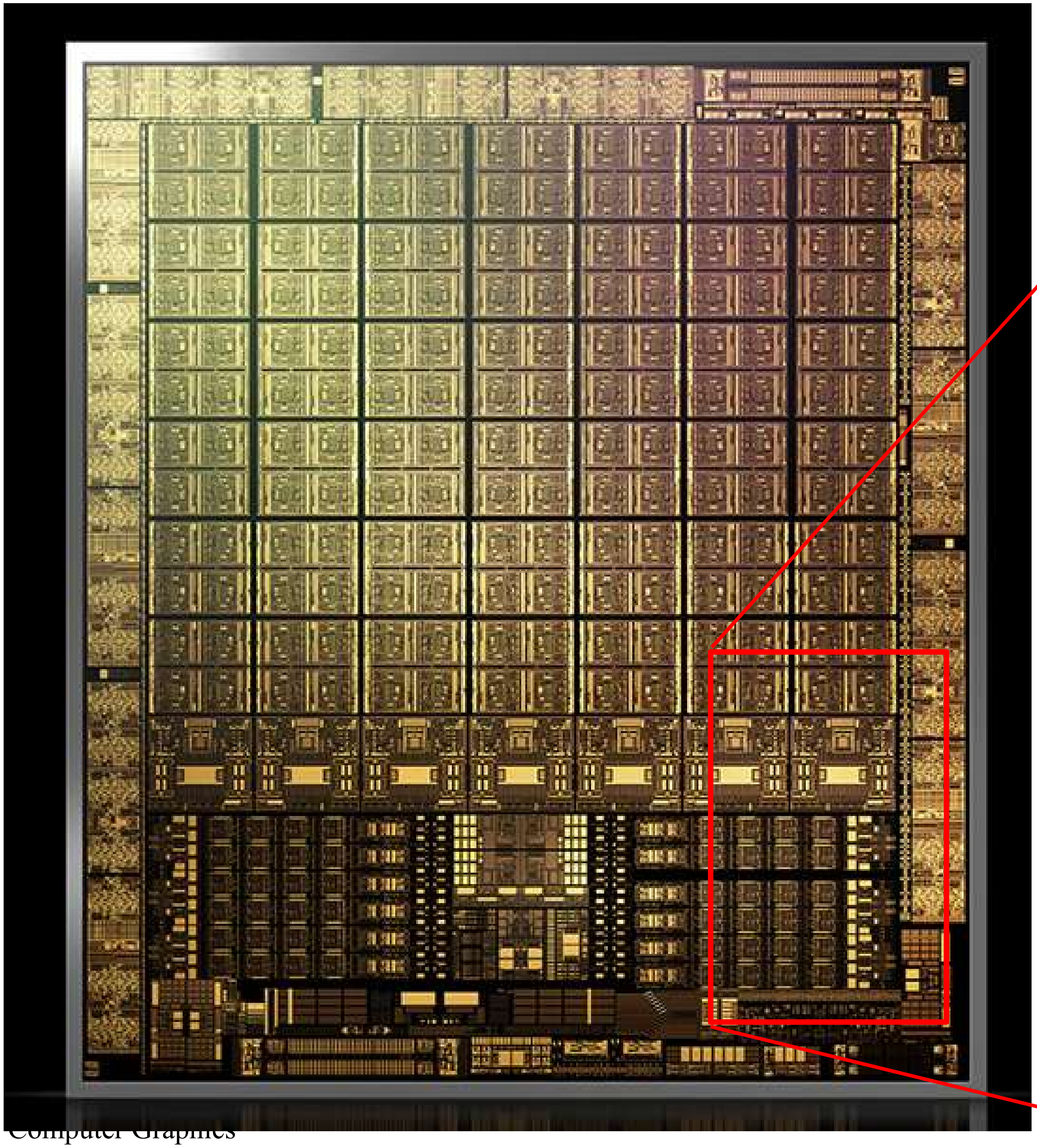
DirectX:	12 Ultimate (12_2)
OpenGL:	4.6
OpenCL:	3.0
Vulkan:	1.3
CUDA:	8.9
Shader Model:	6.7

Render Config

Shading Units:	16384
TMUs:	512
ROPs:	176
SM Count:	128
Tensor Cores:	512
RT Cores:	128
L1 Cache:	128 KB (per SM)
L2 Cache:	72 MB

NVIDIA

NVIDIA's Ampere Chip



NVIDIA

The Bottom Line is This

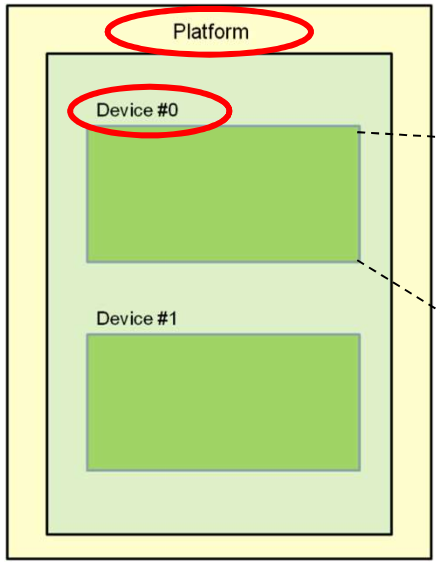
It is obvious that it is difficult to *directly* compare a CPU with a GPU. They are optimized to do different things.

So, let's use the information about the architecture as a way to consider what CPUs should be good at and what GPUs should be good at

CPU	**GPU**
General purpose programming	Data parallel programming
Multi-core under user control	Little user control
Irregular data structures	Regular data structures
Irregular flow control	Regular Flow Control

BTW,
The general term in the OpenCL world for an SM is a **Compute Unit**.
The general term in the OpenCL world for a CUDA Core is a **Processing Elemen**t.

Compute Units and Processing Elements are Arranged in Grids

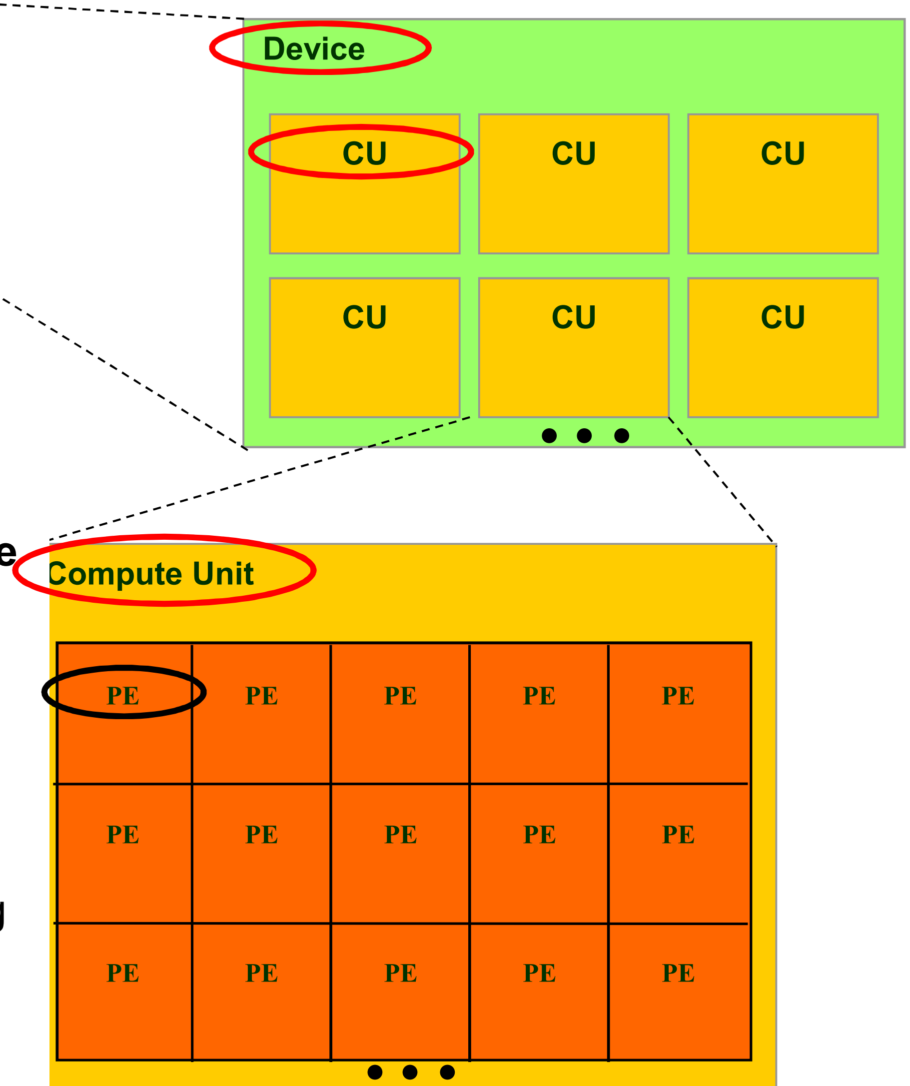


A GPU Platform can have one or more **Devices**.

A GPU **Device** is organized as a grid of **Compute Units.**

Each Compute Unit is organized as a grid of **Processing Elements**.

So in NVIDIA terms, their new V100 GPU has 84 Compute Units, each of which has 64 Processing Elements, for a grand total of 5,396 Processing Elements.



Thinking ahead to CUDA and OpenCL…

How can GPUs execute General C Code Efficiently?

• Ask them to do what they do best. Unless you have a very intense **Data Parallel** application, don't even think about using GPUs for computing.

• GPU programs expect you to not just have a few threads, but to have ***thousands*** of them!

• Each thread executes the same program (called the *kernel*), but operates on a different small piece of the overall data

• Thus, you have many, many threads, all waking up at about the same time, all executing the same kernel program, all hoping to work on a small piece of the overall problem.

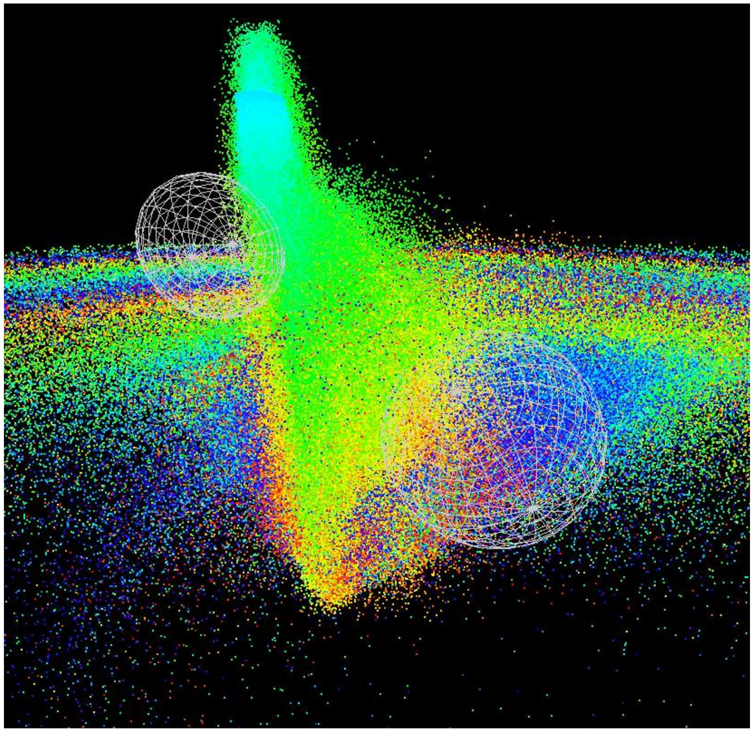
• CUDA and OpenCL have built-in functions so that each thread can figure out which thread number it is, and thus can figure out what part of the overall job it's supposed to do.

• When a thread gets blocked somehow (a memory access, waiting for information from another thread, etc.), the processor switches to executing another thread to work on.

So, the Trick is to Break your Problem into Many, Many Small Pieces

Particle Systems are a great example.

1. Have one thread per *each particle*.
2. Put all of the initial parameters into an array in GPU memory.
3. Tell each thread what the current **Time** is.
4. Each thread then computes its particle's position, color, etc. and writes it into arrays in GPU memory.
5. The CPU program then initiates OpenGL drawing of the information in those arrays.



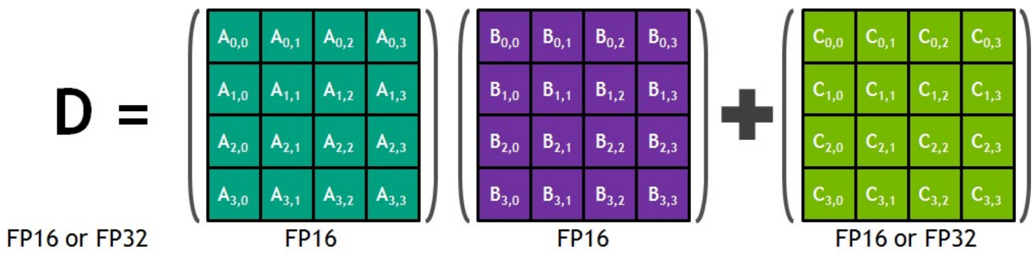
Ben Weiss

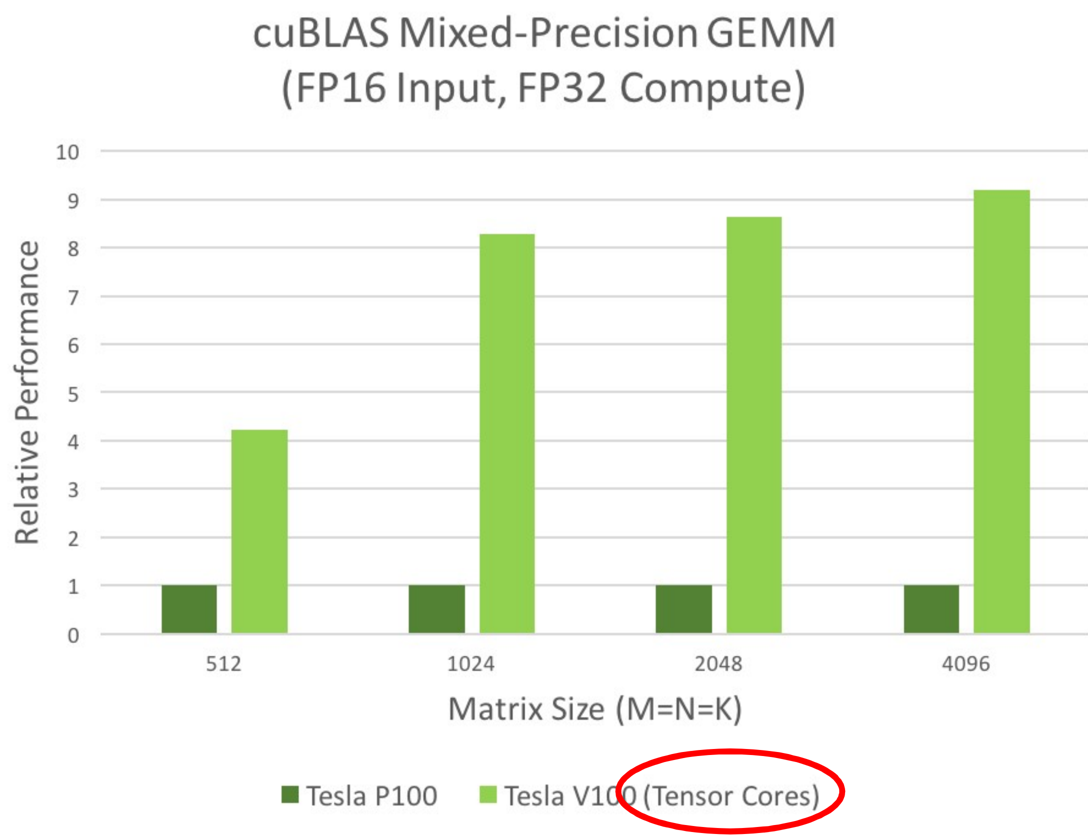
Note: once setup, the data never leaves GPU memory!

Something New – Tensor Cores

NVIDIA

Tensor Cores Accelerate Fused-Multiply-Add Arithmetic





Oregon State University
Computer Graphics

What is Fused Multiply-Add?

Many scientific and engineering computations take the form:
D = A + (B*C);

A "normal" multiply-add would likely handle this as:
tmp = B*C;
D = A + tmp;

A "fused" multiply-add does it all at once, that is, when the low-order bits of B*C are ready, they are immediately added into the low-order bits of A at the same time the higher-order bits of B*C are being multiplied.

Consider a Base 10 example: **789 + (123*456)**

```
   123
 x 456
 -----
   738
  615
 492
 -----
 + 789   Can start adding the 9 the moment the 8 is produced!
 -----
56,877
```

Note: "Normal" A+(B*C) ≠ "FMA" A+(B*C)

There are Two Approaches to Combining CPU and GPU Programs

1. Combine both the CPU and GPU code in the same file. The CPU compiler compiles its part of that file. The GPU compiler compiles just its part of that file.

2. Have two separate programs: a .cpp and a .somethingelse that get compiled separately.

Advantages of Each

1. The CPU and GPU sections of the code know about each others' intents. Also, they can share common structs, #define's, etc.

2. It's potentially cleaner to look at each section by itself. Also, the GPU code can be easily used in combination with other CPU programs.

Who are we Talking About Here?

1 = NVIDIA's CUDA

2 = Khronos's OpenCL

We will talk about each of these separately – stay tuned!

Looking ahead: If threads all execute the same program, what happens on flow divergence?

```
if( a > b )
        Do This;
else
        Do That;
```

1. The line "if(a > b)" creates a vector of Boolean values giving the results of the if-statement for each thread. This becomes a "mask".
2. Then, the GPU executes all parts of the divergence:
   Do This;
   Do That;
3. During that execution, anytime a value wants to be stored, the mask is consulted and the storage only happens if that thread's location in the mask is the right value.

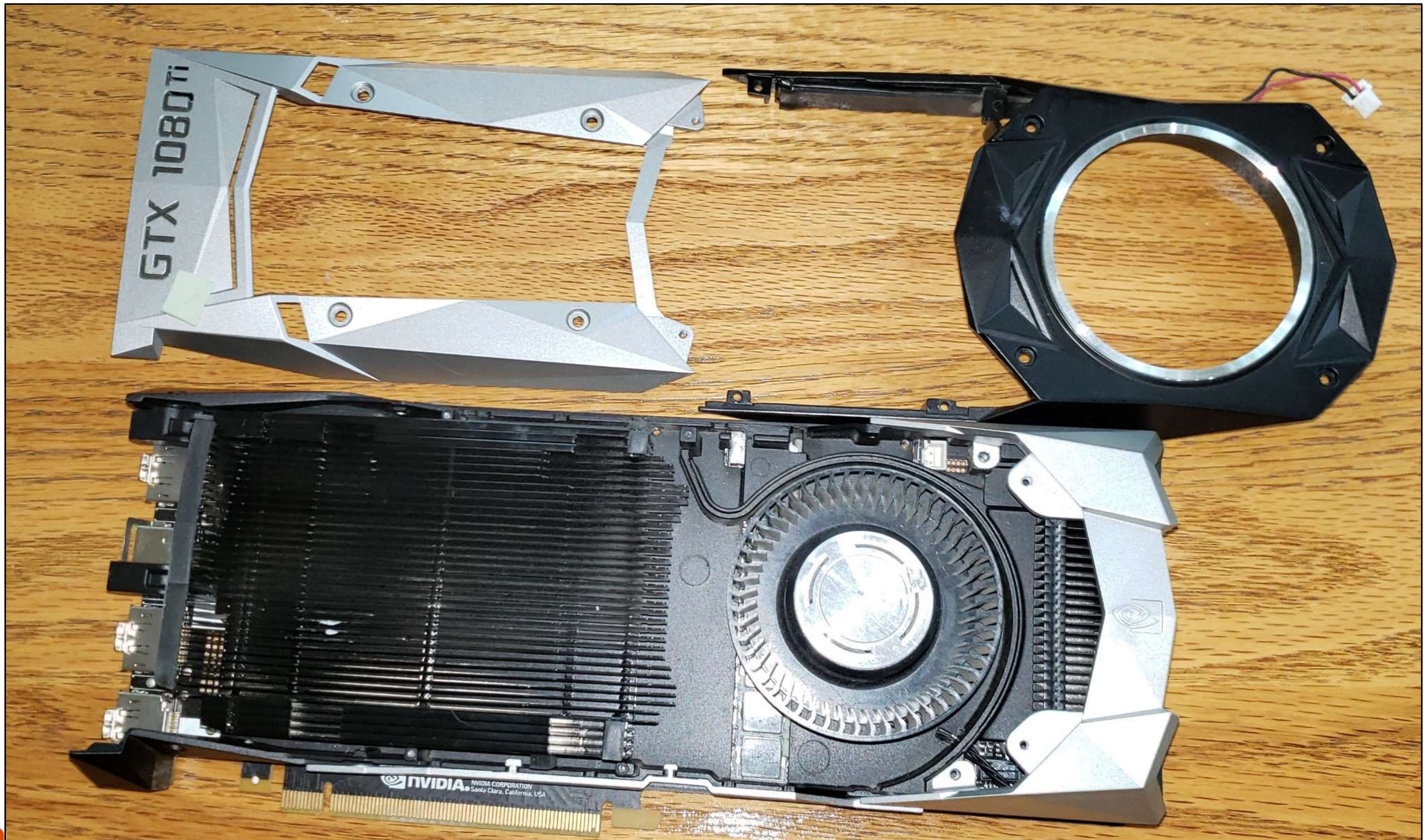
- GPUs were originally designed for the streaming-ness of computer graphics
- Now, GPUs are also used for the streaming-ness of data-parallel computing
- GPUs are better for some things. CPUs are better for others.

Dismantling a Graphics Card

This is an Nvidia 1080 ti card – one that died on us. It willed its body to education.

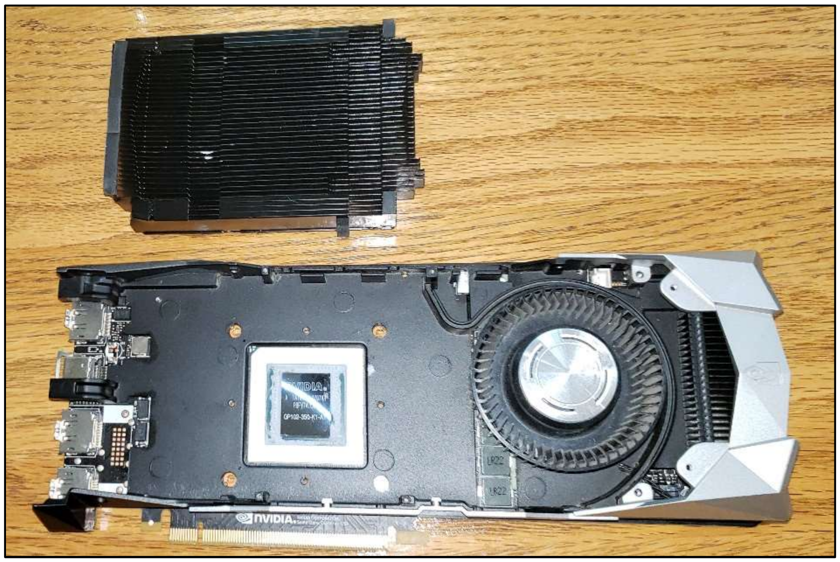
Dismantling a Graphics Card

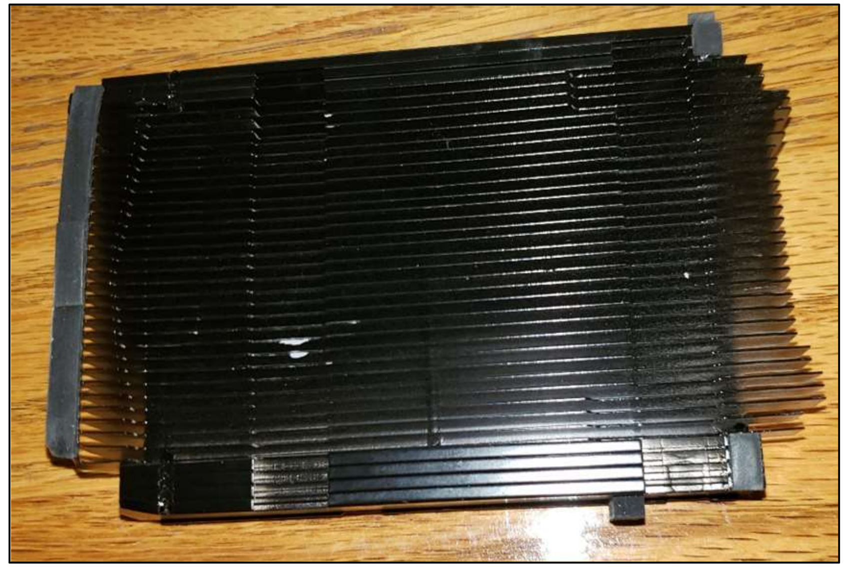
Removing the covers:



Dismantling a Graphics Card

Removing the heat sink:

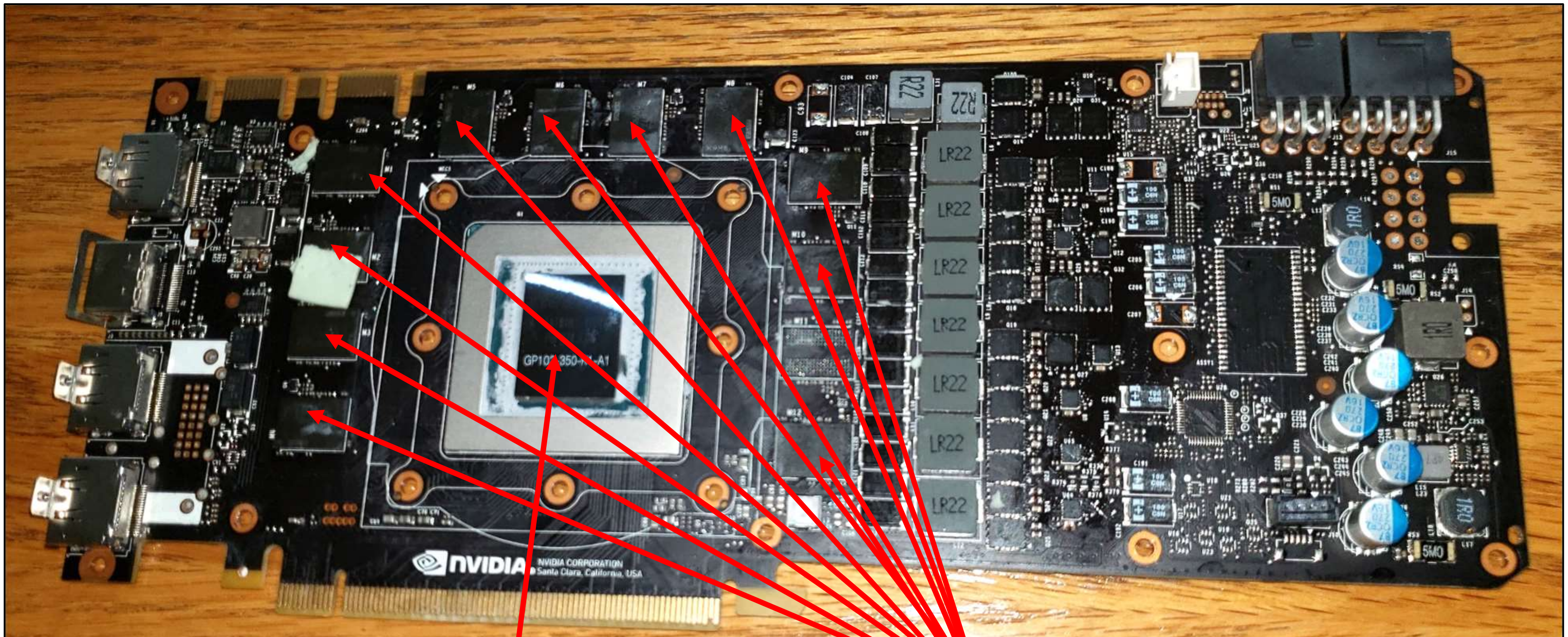




This transfers heat from the GPU Chip to the cooling fins

Dismantling a Graphics Card

Removing the fan assembly reveals the board:

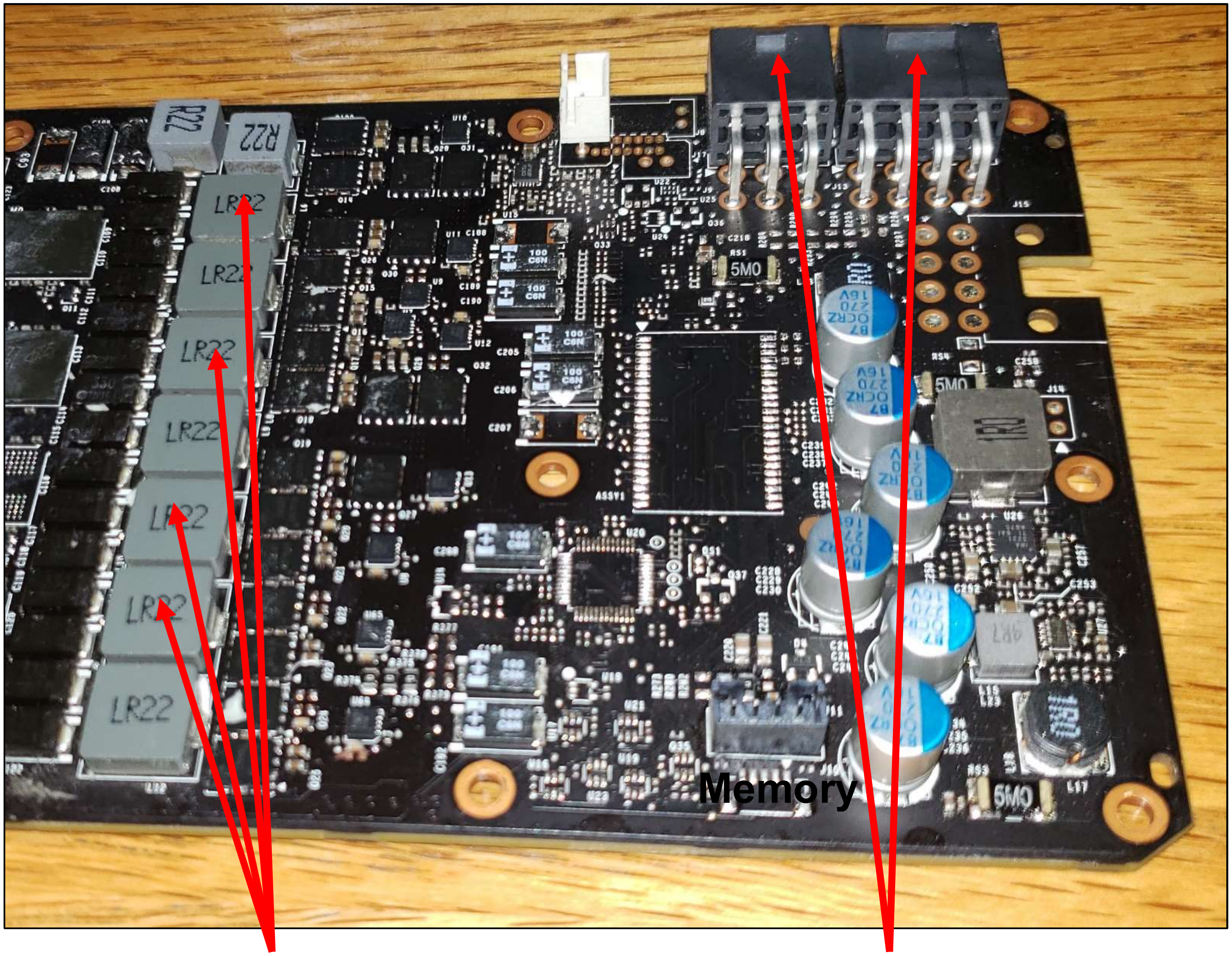


GPU Chip

Memory

Dismantling a Graphics Card

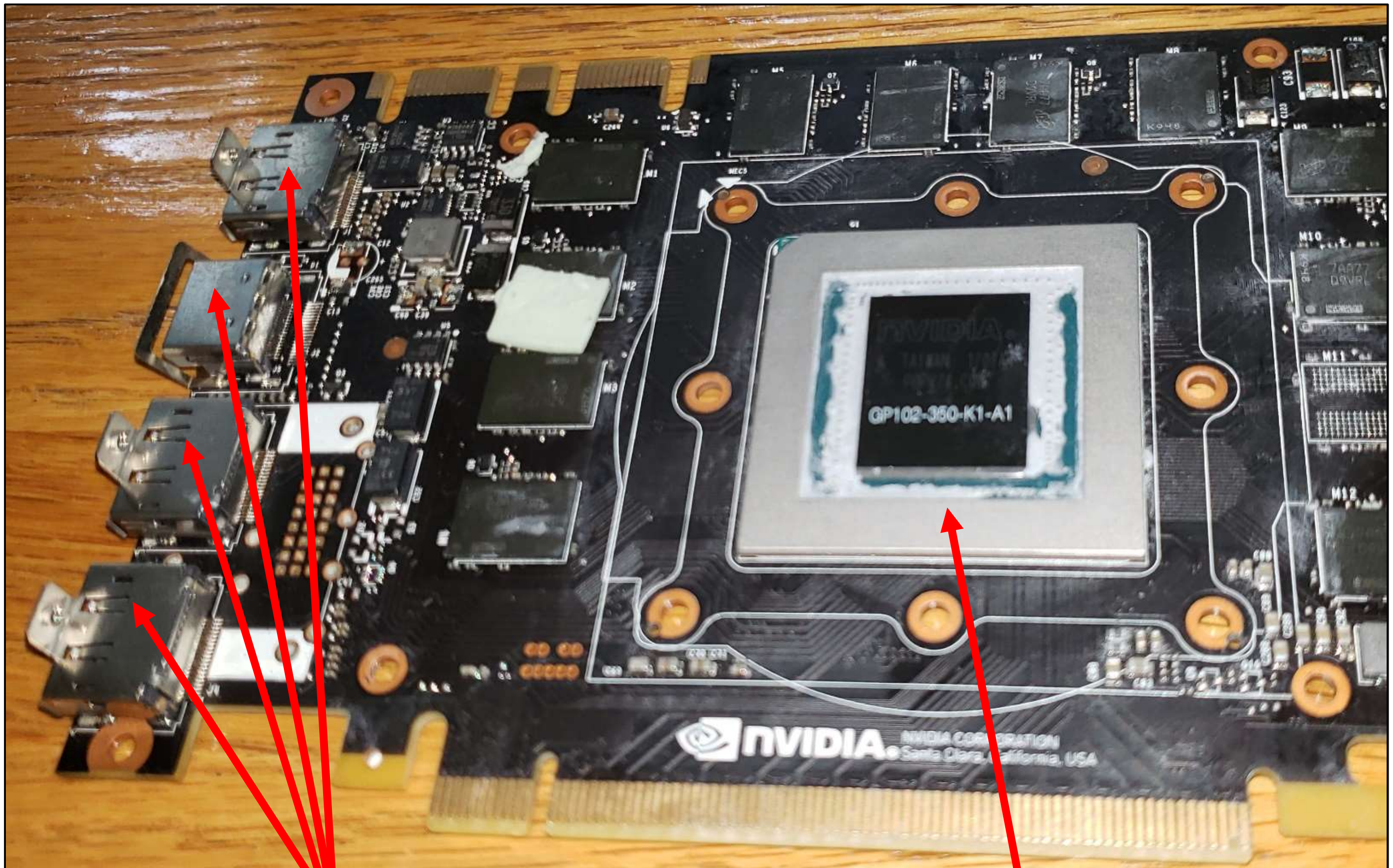
Power half of the board:



Oregon State University Computer Graphics

Dismantling a Graphics Card

Graphics half of the board:

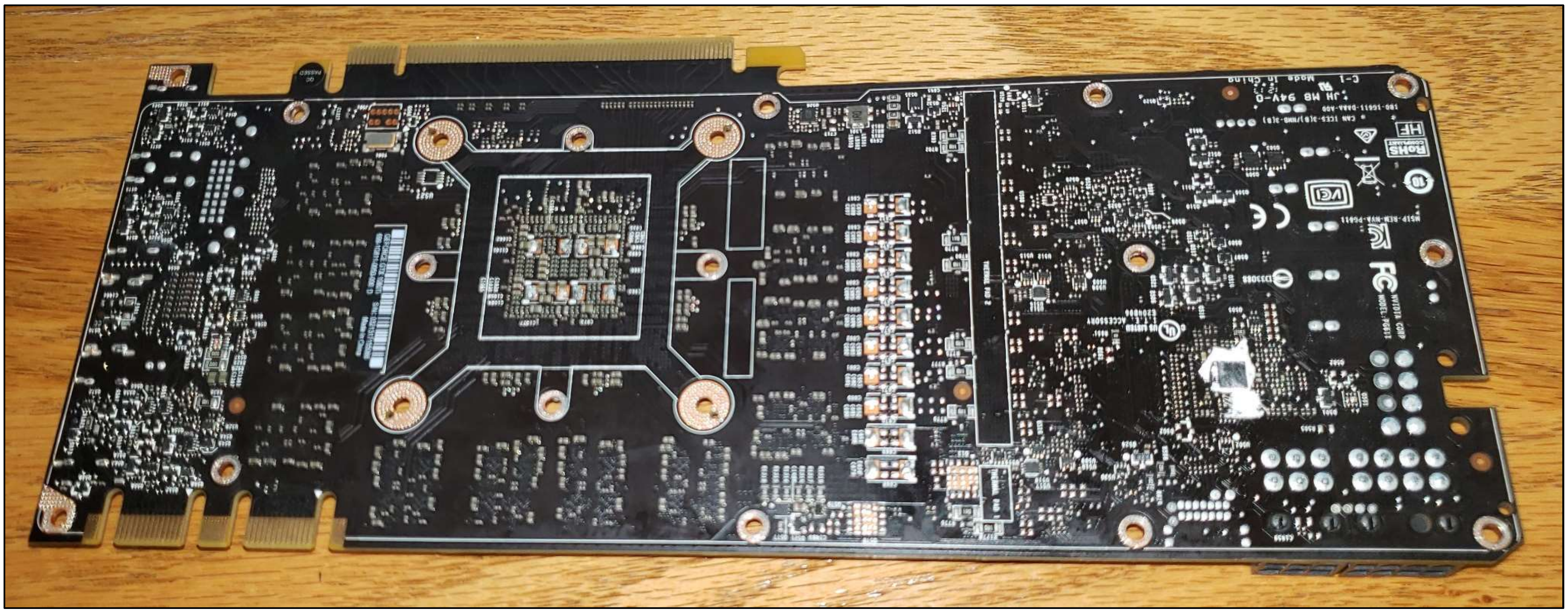


Video out

GPU Chip
This one contains 7.2 billion transistors!
The newer cards contain 70+ billion transistors.
(Thank you, Moore's Law)

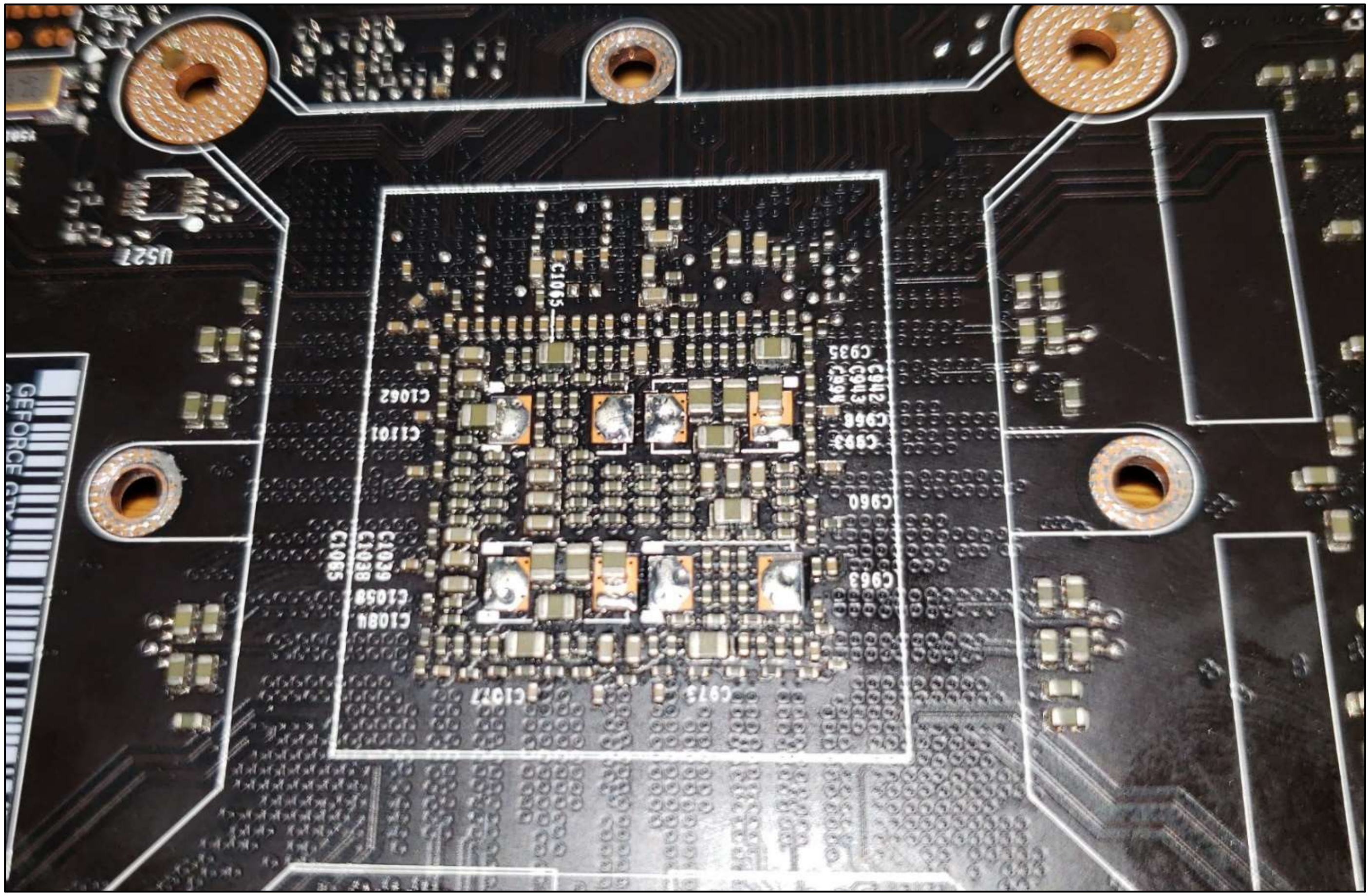
Dismantling a Graphics Card

Underside of the board:



Dismantling a Graphics Card

Underside of where the GPU chip attaches:



Here is a fun video of someone explaining the different parts of this same card:
https://www.youtube.com/watch?v=dSCNf9DIBGE