Main Parameters to evaluate the GPU performance

Hamdy Sultan

Main Parameters

This video is important for the whitepapers

Architecture name	Release year	Examples	Throughput (Tflops)	
Fermi	2010	GF100	1.5	Obsolete
Kepler	2012	k80	8.74	rare
Maxwell	2014	M40	7	Still in the market
Pascal	2016	P100	5.3 – 10.6 - 21.2	Still in the market
Volta	2017	V100	31.4 - 8.2 – 16.4	Widely used
Turing	2018			Widely used
Ampere	2020	A100 – RTX 30XX	78 - 19.5	Widely used
Hopper	2022	GH100 – H100	378	Just released

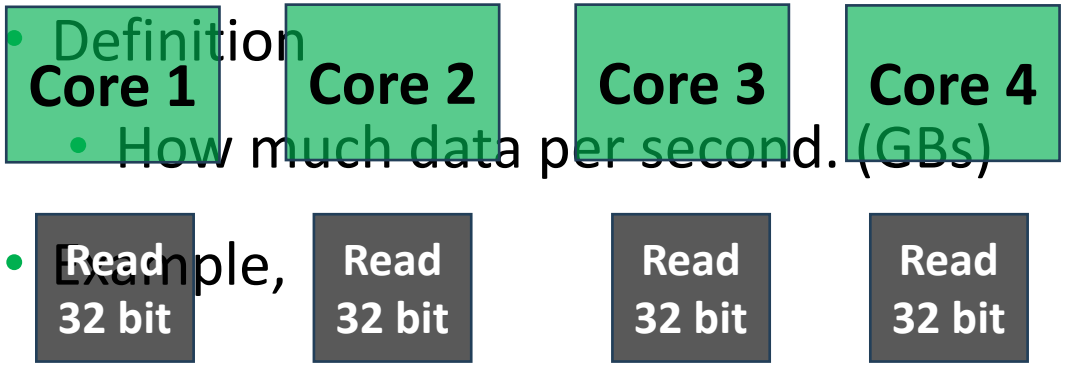
Main Parameters for performance

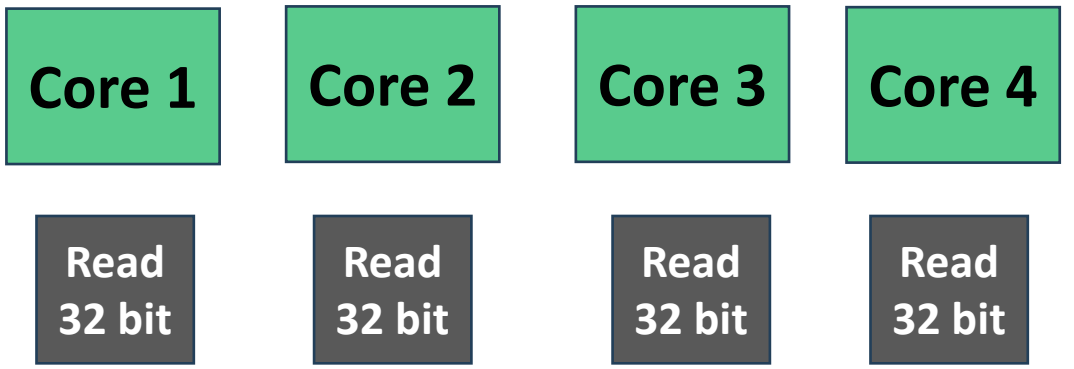
- What are the differences between the architectures ?

1. Memory bandwidth. (memory speed + bus width).
2. The throughput TFLOPS. (The core count + Speed)
3. New features. (supporting new data types – The tensor cores)

Memory Bandwidth effect

- Definition
  - How much data per second. (GBs)
- Example,





Memory 2: 128 bits/sec

Memory 1: 32 bits/sec

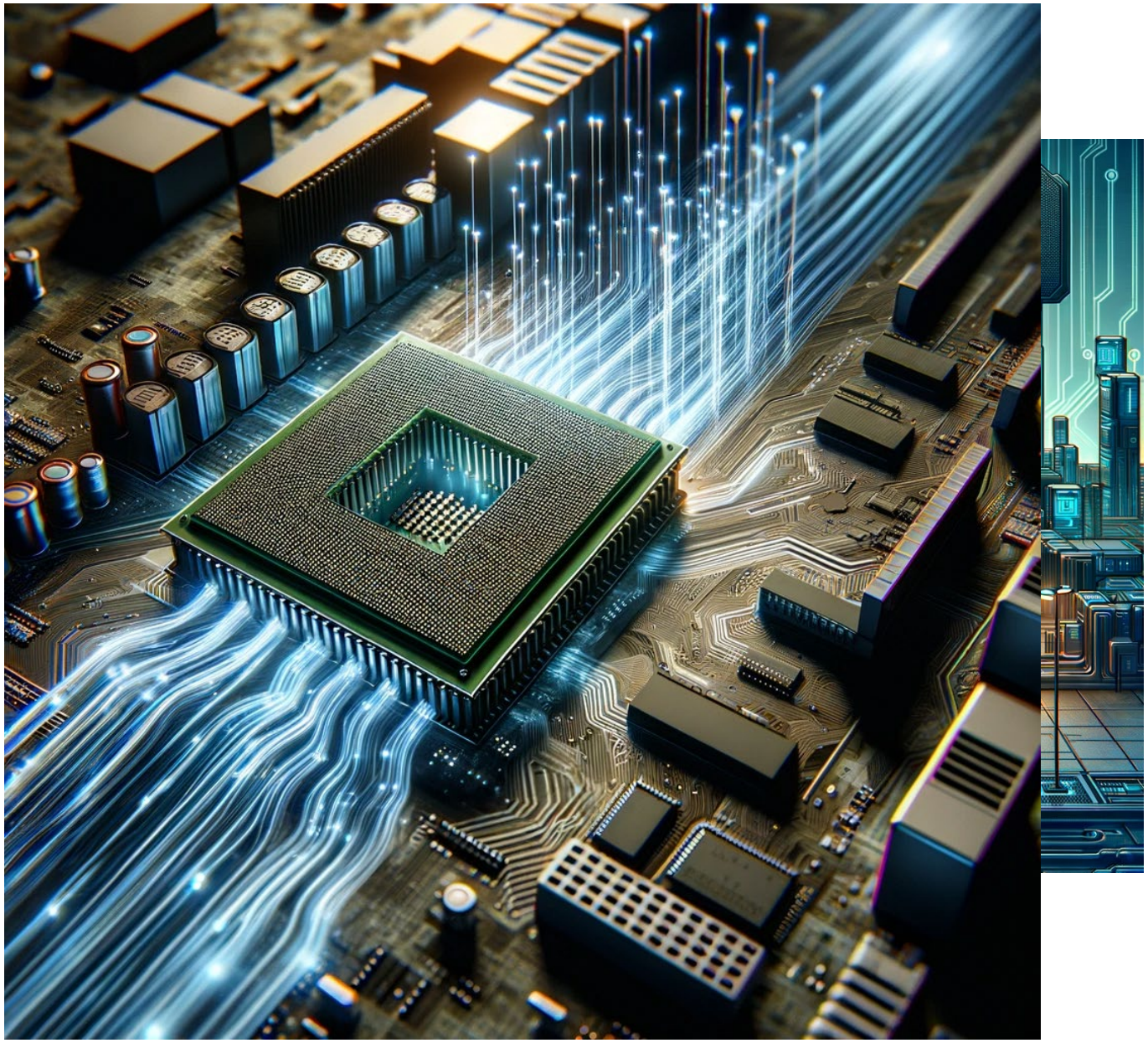
Main Parameters for performance

- What are the differences between the architectures ?

1. Memory bandwidth. (memory speed + bus width).
2. The throughput TFLOPS. (The core count + Speed)
3. New features. (supporting new data types – The tensor cores)

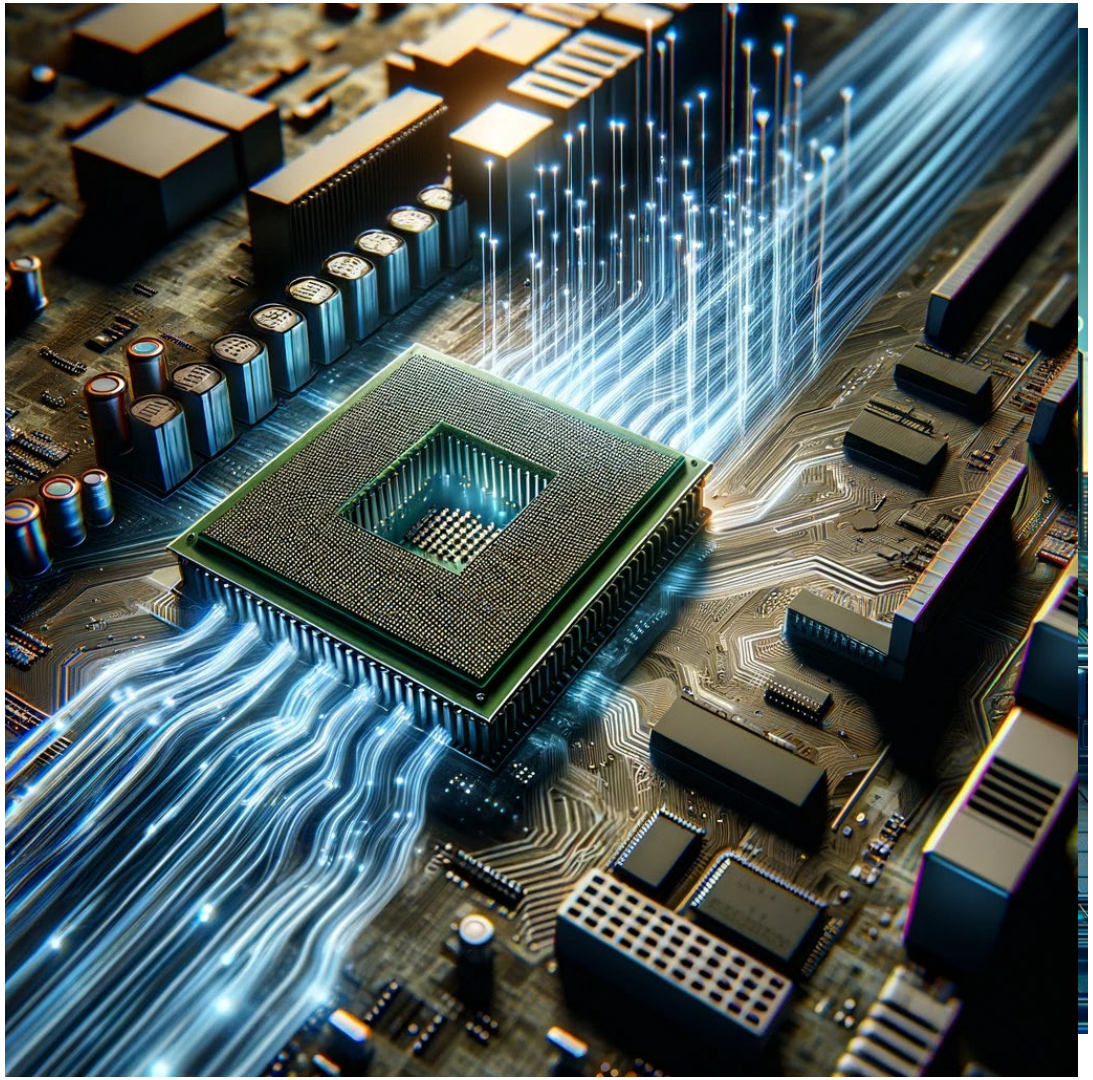
Bus width & bandwidth

- Bus width = road with many lanes.
  - bus width = The number lanes



Memory speed & bandwidth

- Memory speed is essential.
- How fast the memory can operate.
- Bandwidth increases if the memory speed increases.



Main Parameters for performance

- What are the differences between the architectures ?

1. Memory bandwidth. (memory speed + bus width).
2. The throughput TFLOPS. (The core count + Speed)
3. New features. (supporting new data types – The tensor cores)

The performance metric	Its features
Memory bandwidth	Bus width
	Memory speed (MHz)
	The memory technology

Core count and speed

GPU 2: 200 cores

GPU 1: 100 cores

Main Parameters for performance

- What are the differences between the architectures ?

1. Memory bandwidth. (memory speed + bus width).
2. The throughput TFLOPS. (The core count + Speed)
3. New features. (supporting new data types – The tensor cores)

The performance metric	Its features
Memory bandwidth	Bus width
	Memory speed (MHz)
	The memory technology
The throughput	The core count
	The core speed

Question ?!

GPU 2: 100 cores
100 MHZ

GPU 1: 200 cores
200 MHz

Main Parameters for performance

- What are the differences between the architectures ?

1. Memory bandwidth. (memory speed + bus width).
2. The throughput TFLOPS. (The core count + Speed)
3. New features. (supporting new data types – The tensor cores)

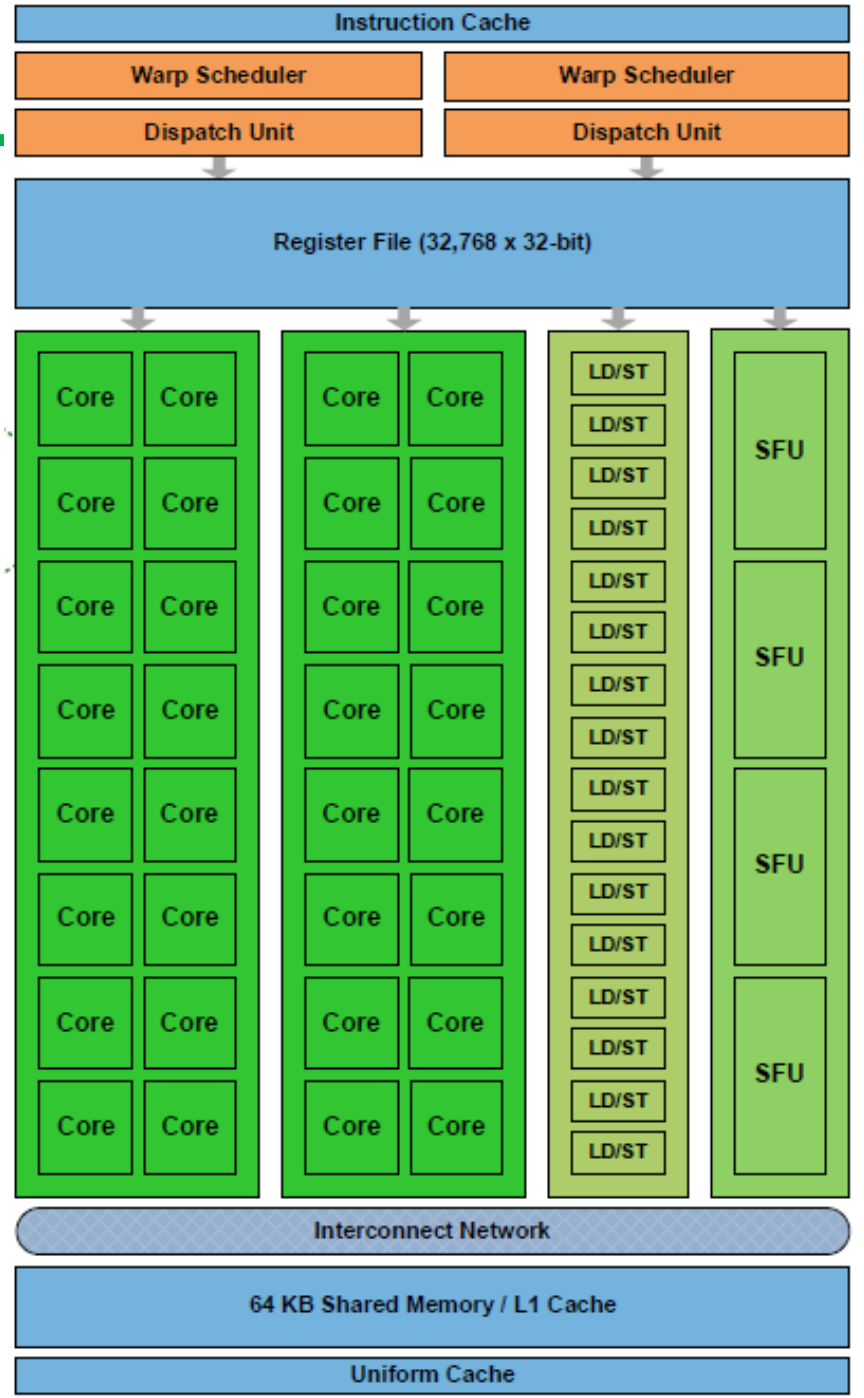

4 BIIT
BUS
ITE
CHECKPOINT


2 BUT W BIT
BIT CHECKPOIN

Bus width & bandwidth

- Bandwidth increases when the bus width increases.

- A picture of the Global memory and it's connection (bus):
- Example:
  - bus width = The number of Car lanes
  - One bit = one car .
  - One bit = one bit .
- Example 1: A 32-bit bus width

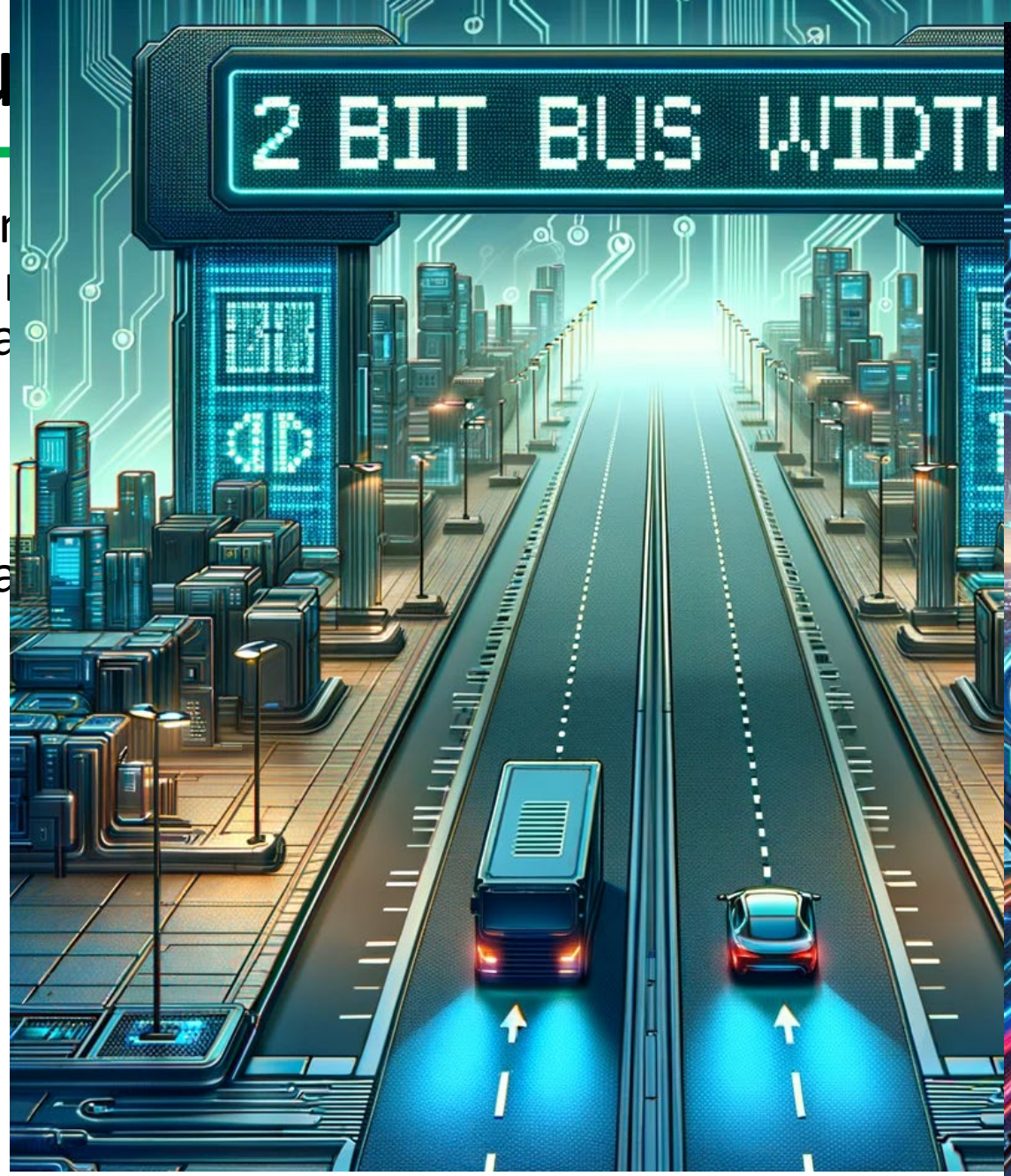


Bu

- Bar

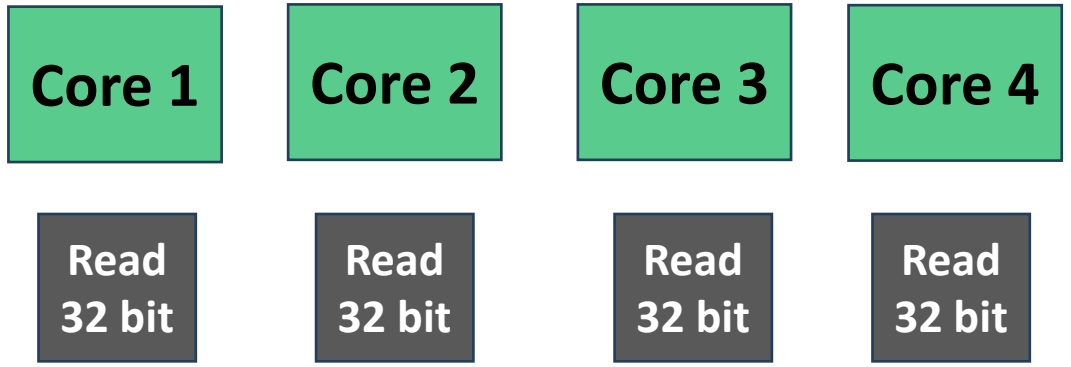
A pictu

- Exa
  -
  -
- Exa



Memory Bandwidth effect

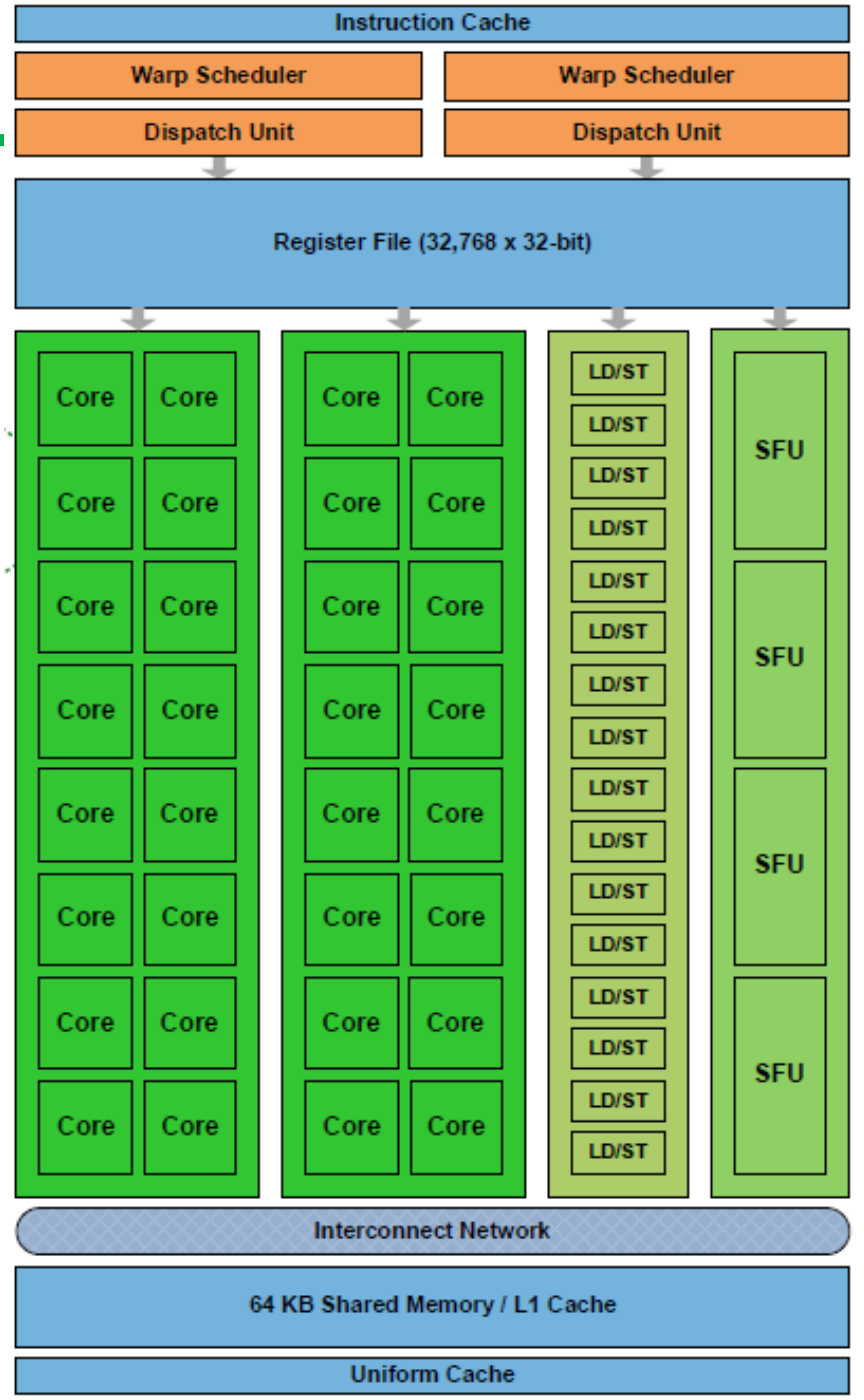
- Definition
  - How much data per second. (GBs)
- Example,



Memory 1: 128 bits/sec

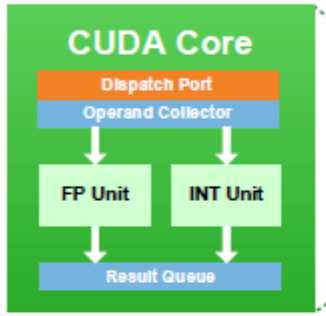
The core speed and count

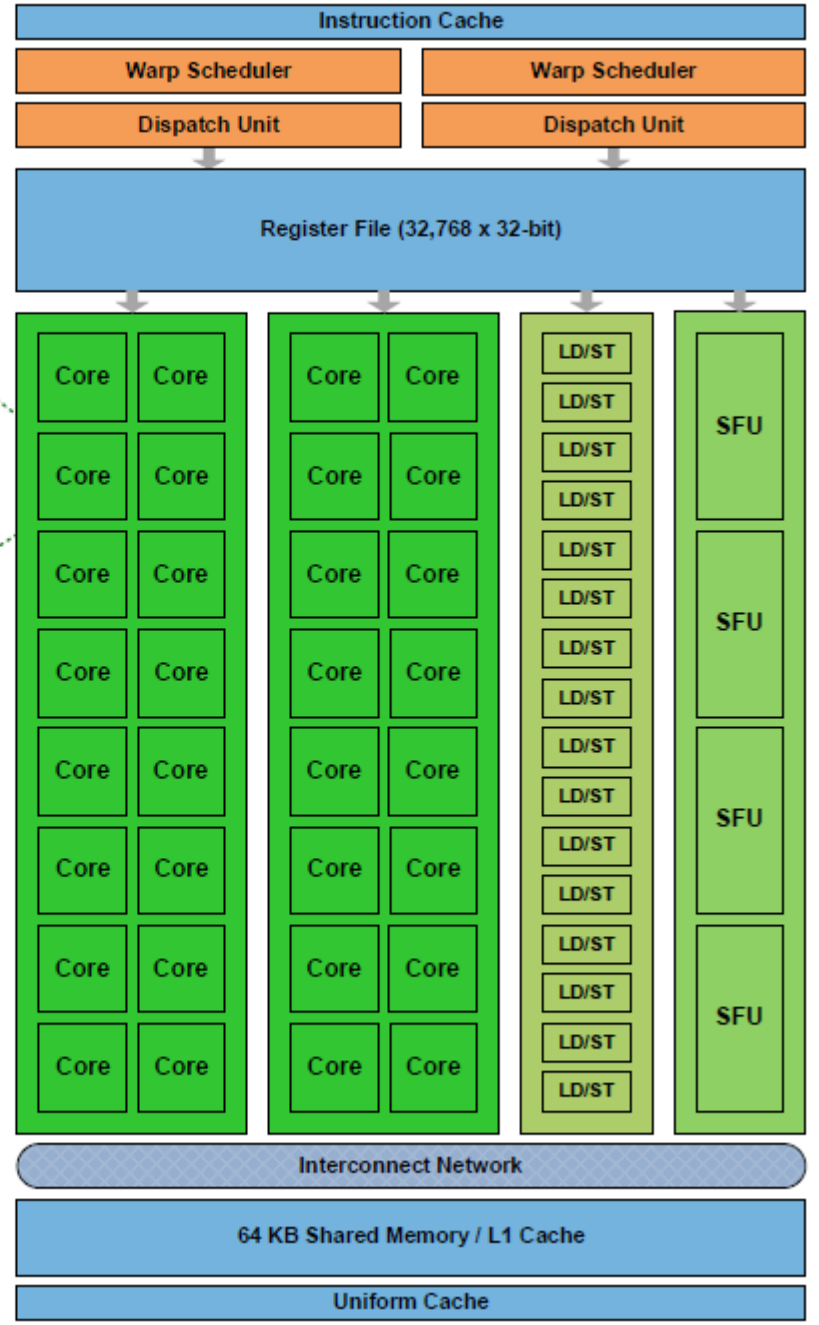
- Bandwidth increases when the bus width increases.
- Example:



SM – Fermi architecture - 2010

- Warp Scheduler
- Dispatch unit
- 16 Load store units (works in parallel)
- 32 Cores
  - Each has ALU and FPU.
  - Or double precision in 2 cycles
- 4 Special function unit
  - Used for performing special operation like sin,cos and exp
- Register file
- L1 cache
  - Accessed by all threads inside the block





DRAM
Host Interface
GigaThead
DRAM
L2 Cache
DRAM
DRAM
DRAM
DRAM

1-b - > z=1

1-b: second case -> z=1000

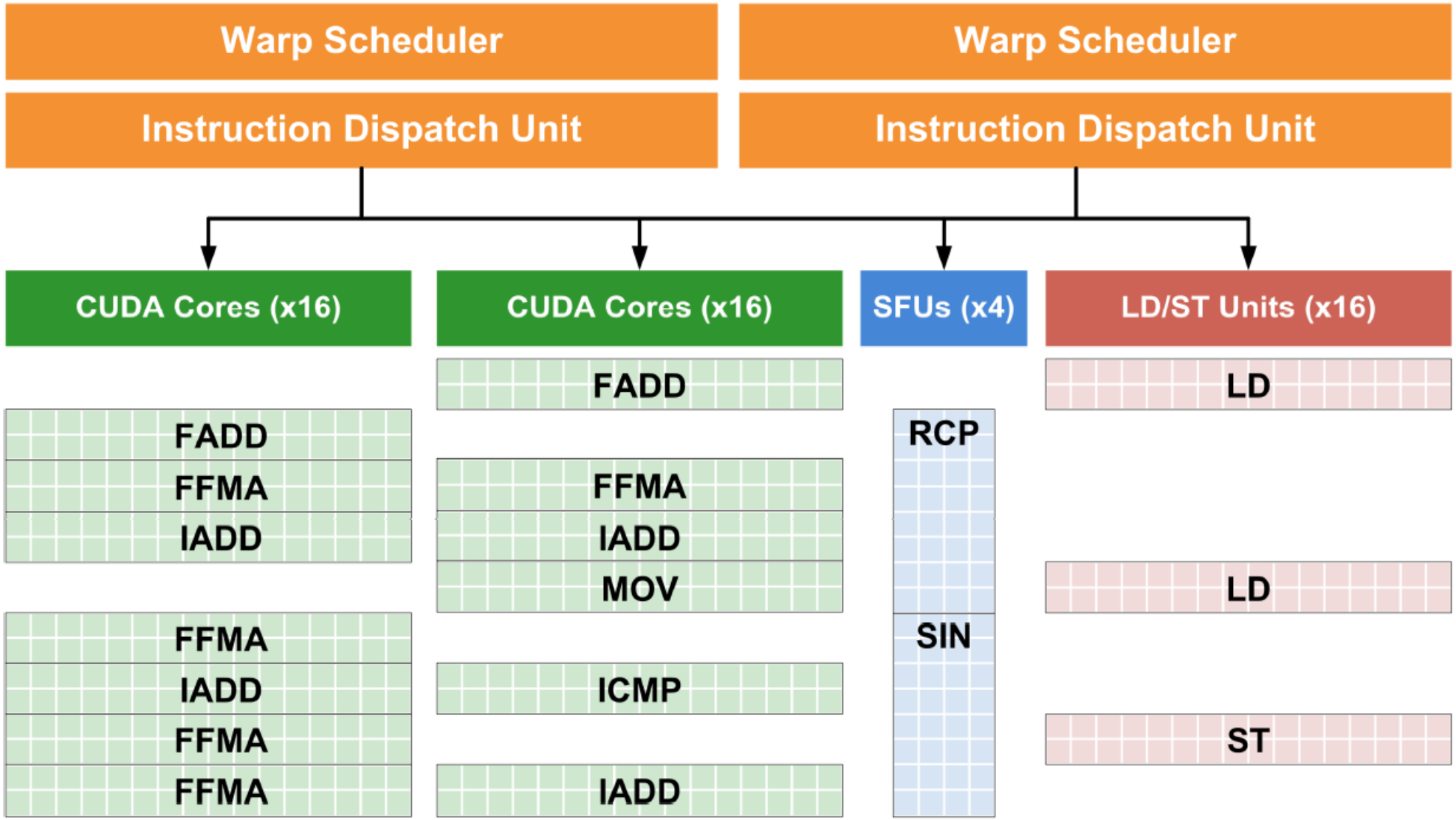
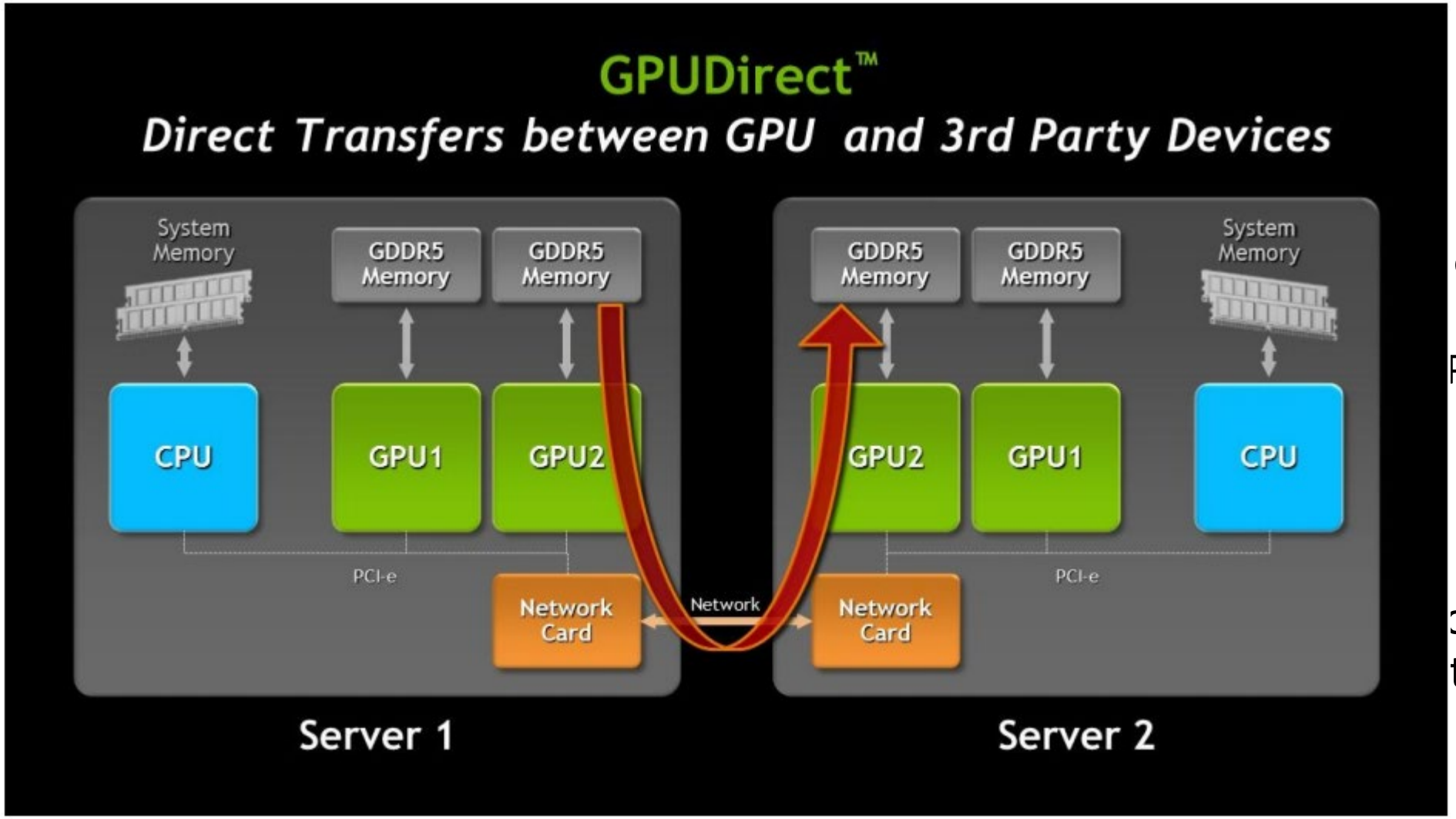


Figure 7. A total of 32 instructions from one or two warps can be dispatched in each cycle

Kepler – What's new ? -2012 Features

- **Designed for Tesla for** HPC compute workloads
- **Over 1 TI**
  - Greate
- **Hyper-Q -**
  - Hyper-C
    dramat
  - increas
    allowin
    connec
- **NVIDIA G**
  - NVIDIA
    GPUs i
    needin
  - Allows
    memory on multiple GPUs within the same system



Each stream receives its own work queue

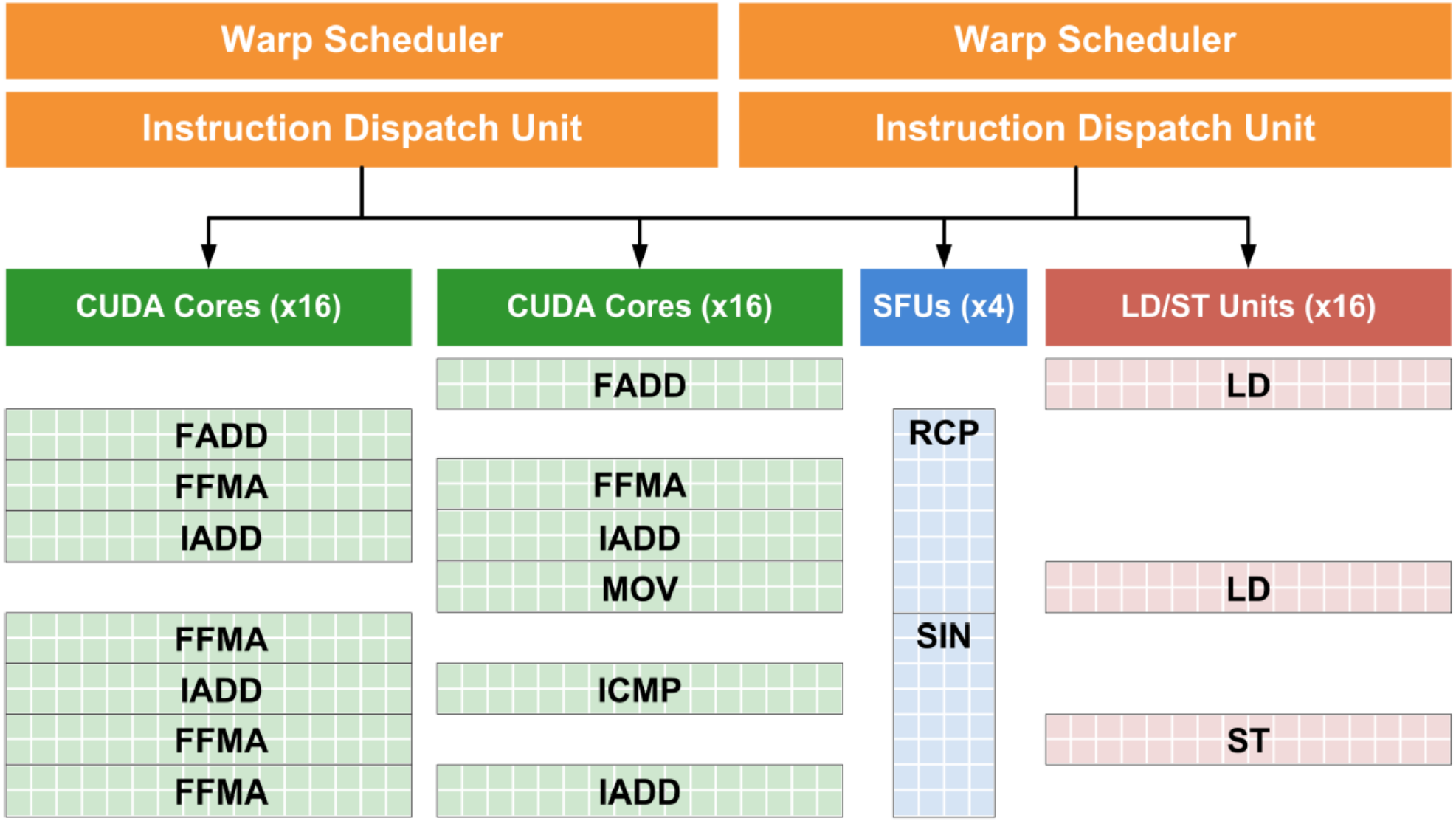
Kepler – What's new ? – Architecture change

- Increased # core, SFU, warp scheduler, dispatchers and the ld/st units in addition to enhancing them.
  - Ex: 192 single-precision CUDA cores. (32 in Fermi)

- Added DP units
  - For HPC applications

Maxwell– What's new ? - 2014

- Desi
- Enh
  - E
  - D
  - S



Pascal (GP100) – What's new ? - 2016

Key features of Tesla P100 include:

- **Extreme performance**
  Powering HPC, Deep Learning, and many more GPU Computing areas
- **NVLink™**
  NVIDIA's new high speed, high bandwidth interconnect for maximum application scalability
- **HBM2**
  Fast, high capacity, extremely efficient CoWoS (Chip-on-Wafer-on-Substrate) stacked memory architecture

  Using this memory technology, Pascal is able to deliver a whopping 720 GBps by using its ultra-wide 4096-bit memory bus. In comparison, K80 GM bandwidth is 240 GBps for each GPU.

- **Unified Memory, Compute Preemption, and New AI Algorithms**
  Significantly improved programming model and advanced AI software optimized for the Pascal architecture; enabling a single load/store instruction and pointer address to access any of the GPU memory spaces (global, local, or shared memory), rather than different instructions and pointers for each.

- 5.3 TFLOPS of double precision floating point (FP64) performance
- 10.6 TFLOPS of single precision (FP32) performance
- 21.2 TFLOPS of half-precision (FP16) performance Precision and throughput tradeoff

Volta

- New SM
  - Perfo
- Tensor c
  - Increa

cache read bandwidth of V100.

Ampere Architecture – What's new ? – 2020

Multi-Instance GPU (MIG)

The new Multi-Instance GPU (MIG) feature allows the A100 Tensor Core GPU to be securely partitioned into as many as seven separate GPU Instances for CUDA applications, providing multiple users with separate GPU resources to accelerate their applications and development projects.

With MIG, each instance's processors have separate and isolated paths through the entire memory system - the on-chip crossbar ports, L2 cache banks, memory controllers, and DRAM address busses are all assigned uniquely to an individual instance. This ensures that an individual user's workload can run with predictable throughput and latency, with the same L2 cache allocation and DRAM bandwidth, even if other tasks are thrashing their own caches or saturating their DRAM interfaces.

Third-Generation NVLink

The third-generation of NVIDIA's high-speed NVLink interconnect implemented in A100 GPUs and the new NVSwitch significantly enhances multi-GPU scalability, performance, and reliability. With more links per GPU and switch, the new NVLink provides much higher GPU-GPU communication bandwidth, and improved error-detection and recovery features.

Third-generation NVLink has a data rate of 50 Gbit/sec per signal pair, nearly doubling the 25.78 Gbits/sec rate in V100. A single A100 NVLink provides 25 GB/second bandwidth in each direction similar to V100, but using only half the number of signal pairs per link compared to

Ampere Architecture – What's new ? – 2020

Support for NVIDIA Magnum IO™ and Mellanox Interconnect Solutions

The NVIDIA A100 Tensor Core GPU is fully compatible with NVIDIA Magnum IO and Mellanox state-of-the-art InfiniBand and Ethernet interconnect solutions to accelerate multi-node connectivity. The NVIDIA Magnum IO APIs integrate computing, networking, file systems, and storage to maximize IO performance for multi-GPU, multi-node accelerated systems. It interfaces with CUDA-X™ libraries to accelerate IO across a broad range of workloads, from AI to data analytics to visualization.

The **full implementation** of the GA100 GPU includes the following units:

- 8 GPCs, 8 TPCs/GPC, 2 SMs/TPC, 16 SMs/GPC, 128 SMs per full GPU
- 64 FP32 CUDA Cores/SM, 8192 FP32 CUDA Cores per full GPU
- 4 Third-generation Tensor Cores/SM, 512 Third-generation Tensor Cores per full GPU
- 6 HBM2 stacks, 12 512-bit Memory Controllers

The **NVIDIA A100 Tensor Core GPU implementation** of the GA100 GPU includes the following units:

- 7 GPCs, 7 or 8 TPCs/GPC, 2 SMs/TPC, up to 16 SMs/GPC, 108 SMs
- 64 FP32 CUDA Cores/SM, 6912 FP32 CUDA Cores per GPU
- 4 Third-generation Tensor Cores/SM, 432 Third-generation Tensor Cores per GPU
- 5 HBM2 stacks, 10 512-bit Memory Controllers

Ampere Architecture – What's new ? – 2020

GPU Features	NVIDIA Tesla P100	NVIDIA Tesla V100	NVIDIA A100
GPU Codename	GP100	GV100	GA100
GPU Architecture	NVIDIA Pascal	NVIDIA Volta	NVIDIA Ampere
GPU Board Form Factor	SXM	SXM2	SXM4
SMs	56	80	108
TPCs	28	40	54
FP32 Cores / SM	64	64	64
FP32 Cores / GPU	3584	5120	6912
FP64 Cores / SM (excl. Tensor)	32	32	32
FP64 Cores / GPU (excl. Tensor)	1792	2560	3456
INT32 Cores / SM	NA	64	64
INT32 Cores / GPU	NA	5120	6912
Tensor Cores / SM	NA	8	4[2]
Tensor Cores / GPU	NA	640	432
GPU Boost Clock	1480 MHz	1530 MHz	1410 MHz
Peak FP16 Tensor TFLOPS with FP16 Accumulate[1]	NA	125	312/624[3]
Peak FP16 Tensor TFLOPS with FP32 Accumulate[1]	NA	125	312/624[3]
Peak BF16 Tensor TFLOPS with FP32 Accumulate[1]	NA	NA	312/624[3]

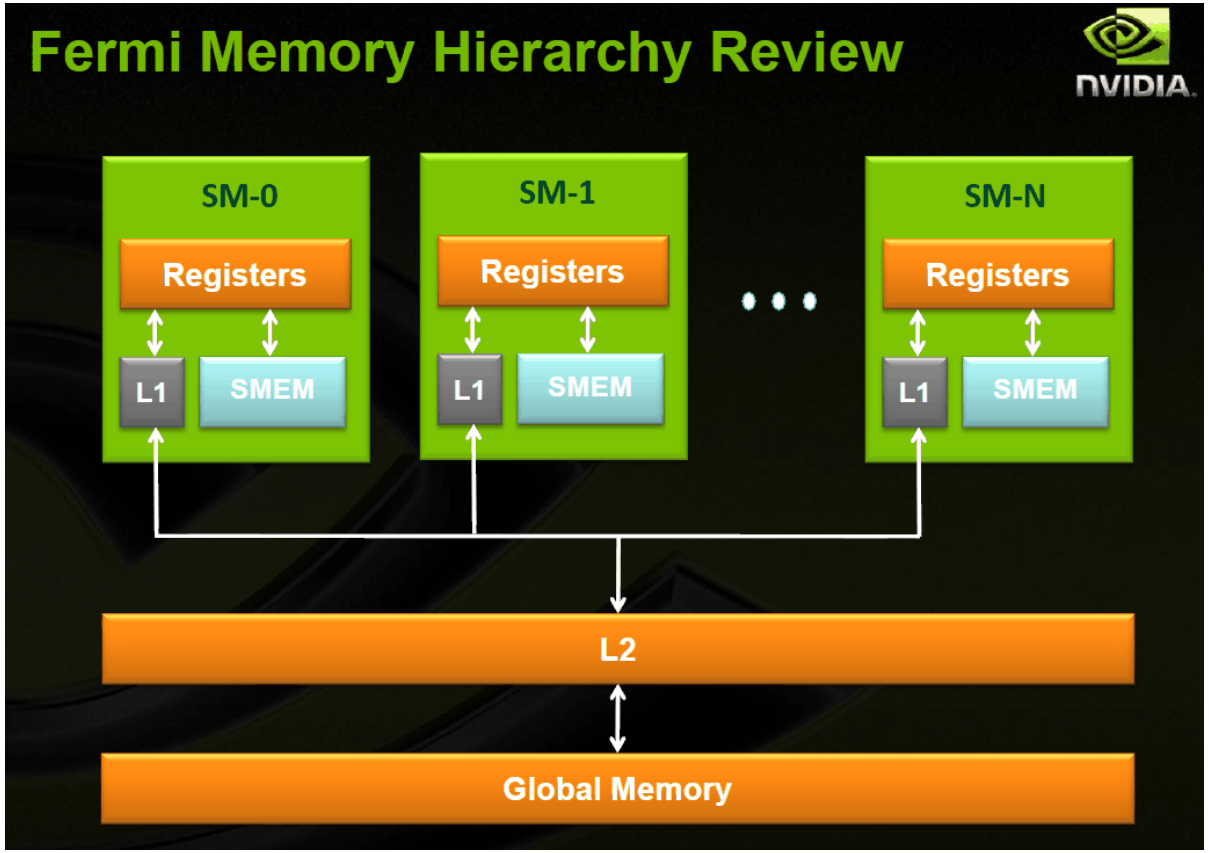
Ampere Architecture – What's new ? – 2020

Memory Interface	4096-bit HBM2	4096-bit HBM2	5120-bit HBM2
Memory Size	16 GB	32 GB / 16 GB	40 GB
Memory Data Rate	703 MHz DDR	877.5 MHz DDR	1215 MHz DDR

Memory Bandwidth	720 GB/sec	900 GB/sec	1555 GB/sec
L2 Cache Size	4096 KB	6144 KB	40960 KB
Shared Memory Size / SM	64 KB	Configurable up to 96 KB	Configurable up to 164 KB
Register File Size / SM	256 KB	256 KB	256 KB
Register File Size / GPU	14336 KB	20480 KB	27648 KB
TDP	300 Watts	300 Watts	400 Watts
Transistors	15.3 billion	21.1 billion	54.2 billion
GPU Die Size	610 mm^2	815 mm^2	826 mm2
TSMC Manufacturing Process	16 nm FinFET+	12 nm FFN	7 nm N7

GPU memory hierarchy

- Global memory
  - Accessible by all threads and the CPU –
- L2 cache
  - Accessible by all threads
- L1 cache
  - Accessible by all threads inside the block
  - (Hardware Cache)
- Shared memory
  - Accessible by all threads inside the block
  - Programmable (Software cache)
  - Allows for threads to communicate and share data between one another
- registers
  - Per thread (lifetime is the thread lifetime)



- **Src**:techpowerup.com

Heterogeneous Architecture

A typical heterogeneous compute node nowadays consists of two multicore CPU sockets and two or more many-core GPUs. A GPU is currently not a standalone platform but a co-processor to a CPU. Therefore, GPUs must operate in conjunction with a CPU-based host through a PCI-Express bus, as shown in Figure 1-9. That is why, in GPU computing terms, the CPU is called the *host* and the GPU is called the *device*.

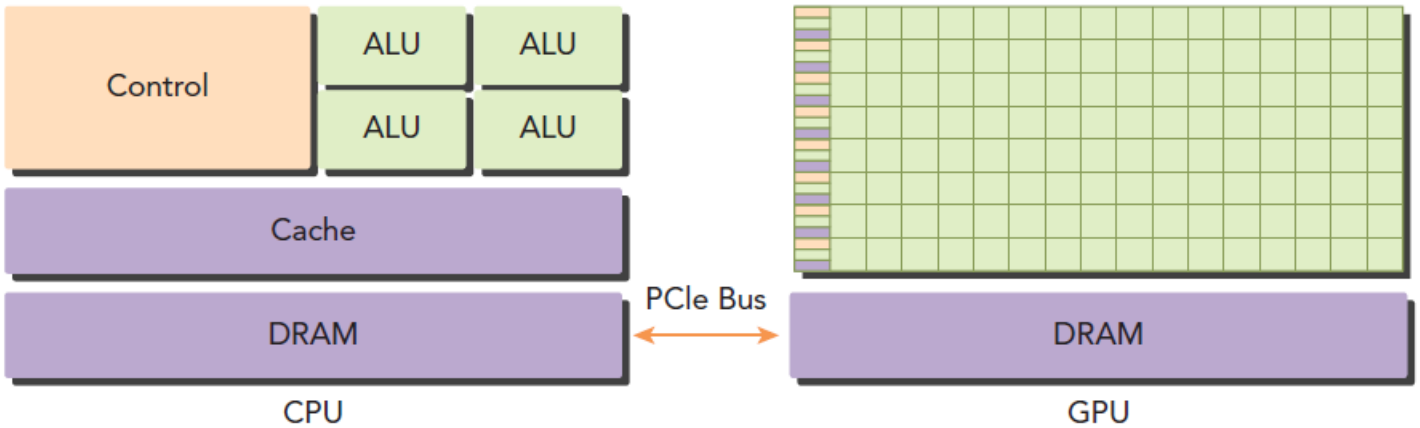
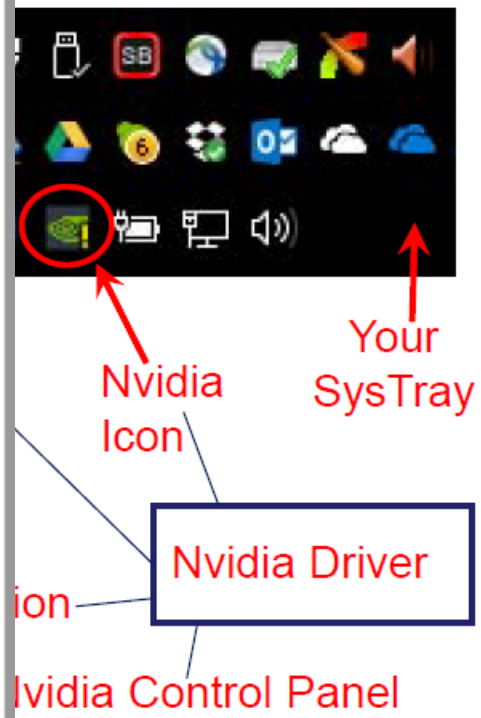


FIGURE 1-9

A heterogeneous application consists of two parts:

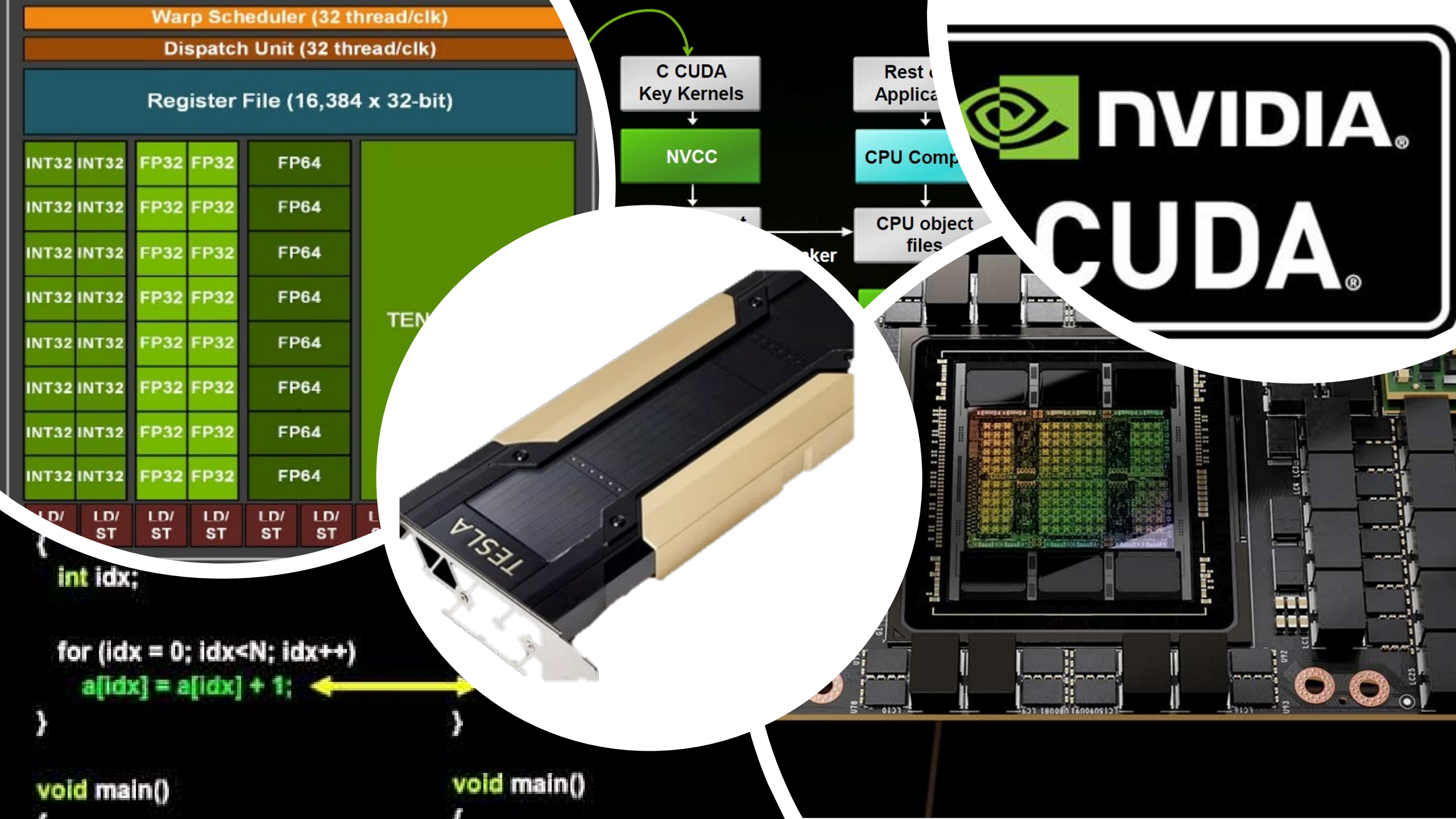
- Host code
- Device code

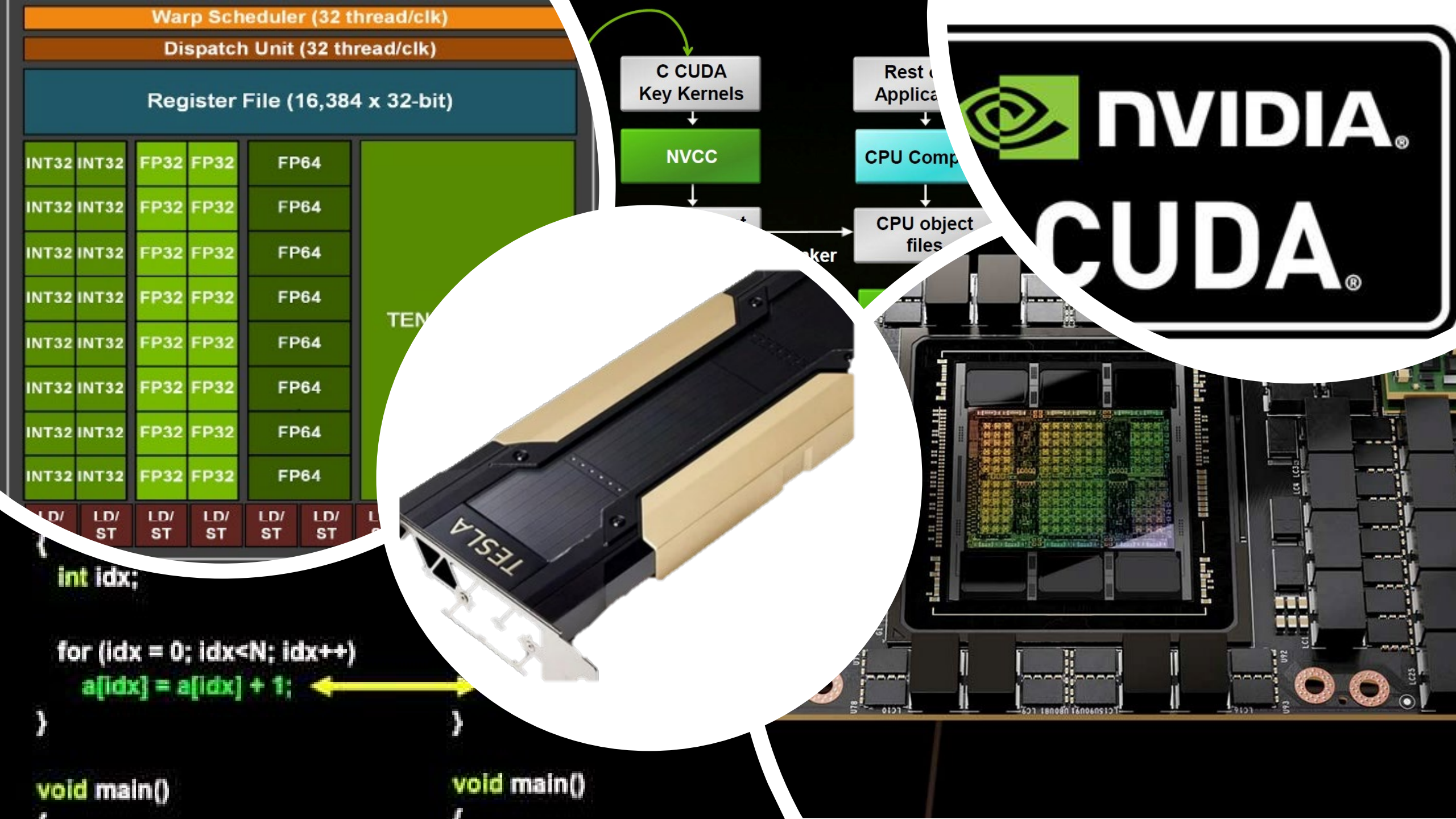


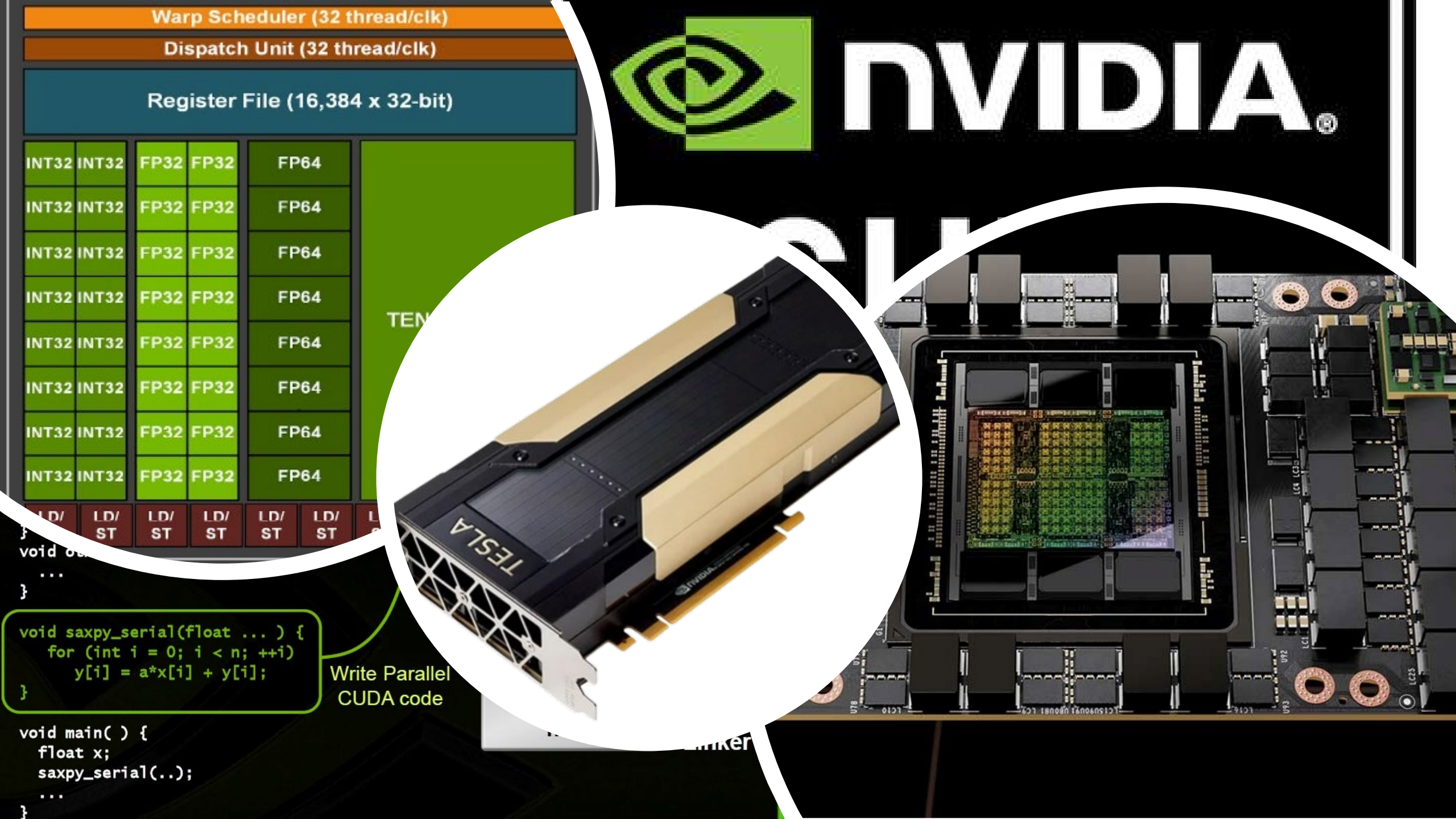
drivers, shown in your
ol, you can open the
Nvidia control panel to see the driver version as well as the parameters of your GPU(s).

Thinking in parallel and having a basic understanding of GPU architecture enables you to write parallel programs that scale to hundreds of cores as easily as you write a sequential program.

If you want to write effi cient code as a parallel programmer, you need a basic knowledge of CPU architectures. For example, *locality* is a very important concept in parallel programming. Locality refers to the reuse of data so as to reduce memory access latency. There are two basic types of reference locality. *Temporal locality* refers to the reuse of data and/or resources within relatively small time durations. *Spatial locality* refers to the use of data elements within relatively close storage locations. Modern CPU architectures use large caches to optimize for applications with good spatial and temporal locality. It is the programmer's responsibility to design their algorithm to efficiently use CPU cache. Programmers must handle low-level cache optimizations, but have no introspection into how threads are being scheduled on the underlying architecture because the CPU does not expose that information.


Warp Scheduler (32 thread/clk)
Dispatch Unit (32 thread/clk)
Register File (16,384 x 32-bit)
INT32
FP32
FP64
LD/ ST
C CUDA Key Kernels
NVCC
CPU Comp
CPU object files
NVIDIA
CUDA
TESLA
int idx;
for (idx = 0; idx<N; idx++)
a[idx] = a[idx] + 1;
void main()


Warp Scheduler (32 thread/clk)
Dispatch Unit (32 thread/clk)
Register File (16,384 x 32-bit)
INT32
FP32
FP64
LD/ ST
C CUDA Key Kernels
NVCC
CPU Comp
CPU object files
NVIDIA
CUDA
TESLA
int idx;
for (idx = 0; idx<N; idx++)
a[idx] = a[idx] + 1;
void main()


Warp Scheduler (32 thread/clk)
Dispatch Unit (32 thread/clk)
Register File (16,384 x 32-bit)
INT32 INT32 FP32 FP32 FP64
LD/ ST
NVIDIA
TESLA
void saxpy_serial(float ...) {
for (int i = 0; i < n; ++i)
y[i] = a*x[i] + y[i];
}
Write Parallel CUDA code
void main( ) {
float x;
saxpy_serial(..);
...
}


Warp Scheduler (32 thread/clk)
Dispatch Unit (32 thread/clk)
Register File (16,384 x 32-bit)
INT32 INT32 FP32 FP32 FP64
LD/ ST
NVIDIA
TESLA
void saxpy_serial(float ...) {
for (int i = 0; i < n; ++i)
y[i] = a*x[i] + y[i];
}
Write Parallel CUDA code
void main( ) {
float x;
saxpy_serial(..);
...
}

GPU Vs CPU

- I need to show them the physical core
- And the logical ones (threads).

- How many threads are available on the CPUs? performance scale with thread count?
  - (Each CPU can generally have two threads).
- Context switching:
  - The action of switching which thread is being proce
  - **High penalty** on the CPU (main computer)
  - Not a big issue on the GPU

Lessons from Graphics Pipeline

- **Throughput** is paramount
  - must paint every pixel within frame time
  - scalability
- Create, run, & retire **lots of threads** very rapidly
  - measured 14.8 Gthread/s on `increment()` kernel
- Use **multithreading** to hide latency
  - 1 stalled thread is OK if 100 are ready to run