

ECE408/CS483/CSE408 Fall 2021

Applied Parallel Programming

Lecture 21

GPU as part of the PC Architecture

Course Reminders


- MP 5.2
 - We are grading it now
- MP 6
 - due next week
- Project PM 1
 - Graded, but check your grade and email TA if not graded
- Project PM 2
 - due this Friday

11	Lecture 20	Nov. 2	Sparse Matrix II
	Lecture 21	Nov. 4	GPU as a part of the PC Architecture
	PM 2	Nov. 5	Baseline GPU Convolution Kernel
12	Lecture 22	Nov. 9	Task parallelism and asynchronous data transfer
	Lecture 23	Nov. 11	Introduction to OpenACC
	Lab 6	Nov. 12	Histogramming
13	Lecture 24	Nov. 16	Guest Lecture
	Lecture 25	Nov. 18	Guest Lecture
	Lab 7	Nov. 19	Sparse Matrix Multiply
	Fall break		
14	Lecture 26	Nov. 30	Guest Lecture
	Lecture 27	Dec. 2	Course Retrospective
	PM 3	Dec. 3	GPU Convolution Kernel Optimizations
15	Exam 2	Dec. 7	Midterm 2 Time TBD

Objectives

- To understand the impact of data transfers on performance when using a GPU as a co-processor
 - speeds and feeds of traditional CPU
 - speeds and feeds when employing a GPU
- To develop a knowledge base for performance tuning for modern GPU's

Review: Canonical CUDA Program Structure

- Global variables declaration
 - Kernel functions
 - `__global__ void kernelOne(...)`
 - Main () // host code
 - allocate memory space on the device – `cudaMalloc(&d_GlblVarPtr, bytes)`
 - transfer data from host to device – `cudaMemcpy(d_GlblVarPtr, h_Gl...)`
 - execution configuration setup
 - kernel call – `kernelOne<<<execution configuration>>>(args...);`
 - transfer results from device to host – `cudaMemcpy(h_GlblVarPtr,...)`
 - optional: compare against golden (host computed) solution
- 
- repeat
as needed

Bandwidth:

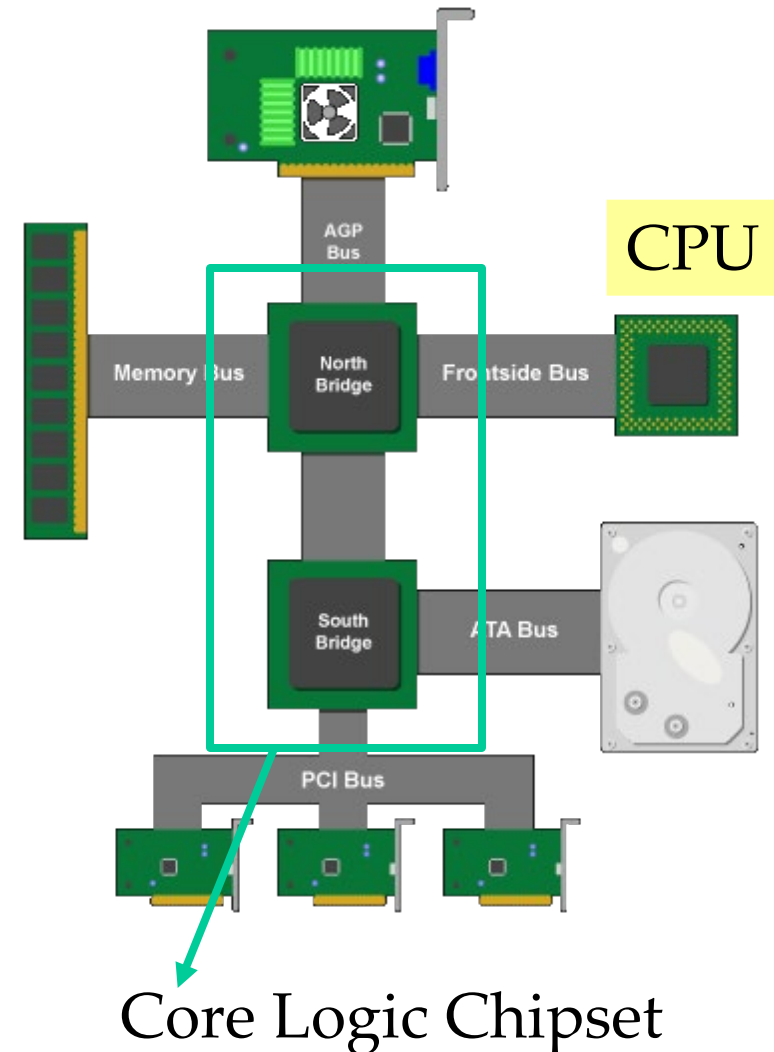
The Gravity of Modern Computer Systems

Bandwidth between key components ultimately **dictates system performance**

- **Especially for GPUs** processing large amounts of data.
- Tricks like buffering, reordering, caching can temporarily defy the rules in some cases.
- Ultimately, performance falls back to what the “speeds and feeds” dictate.

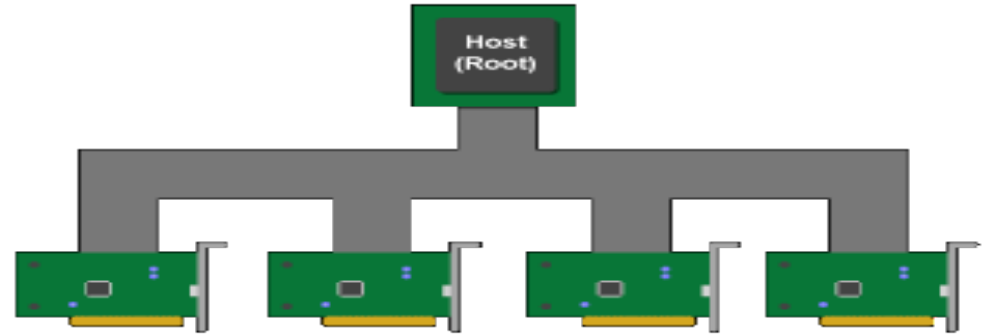
Classic (Historical) PC Architecture

- Northbridge connects 3 components that must communicate at high speed
 - CPU, DRAM, video
 - Video needs first-class access to DRAM
 - Previous NVIDIA cards are connected to AGP, up to 2 GB/s transfers
- Southbridge serves as a concentrator for slower I/O devices



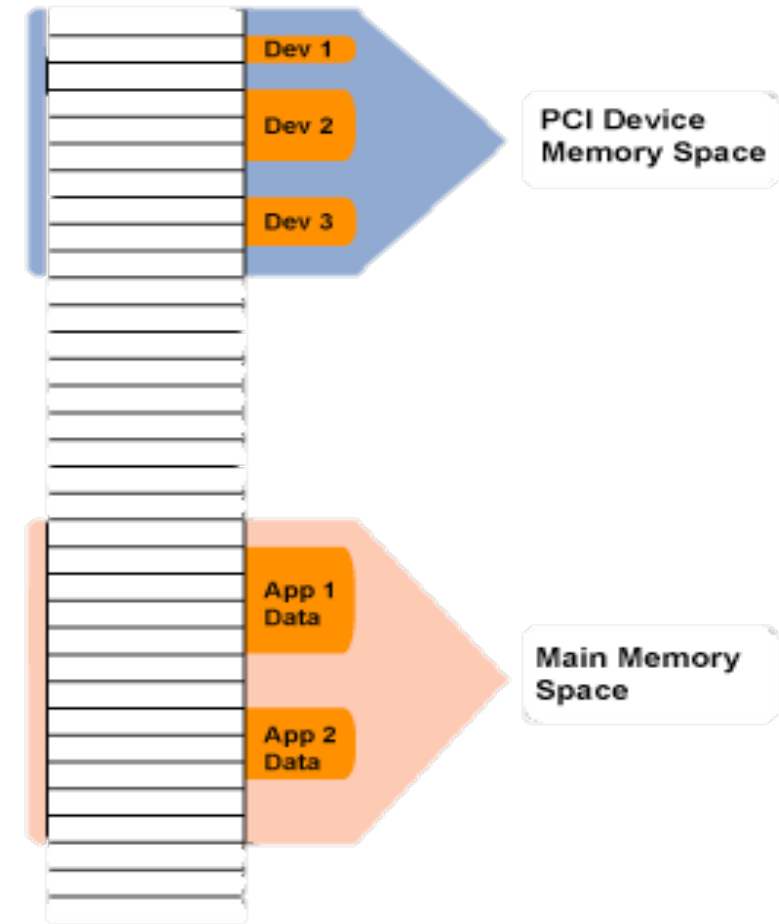
(Original) PCI Bus Specification

- Connected to the South Bridge
 - Originally 33 MHz, 32-bit wide, 132 MB/second peak transfer rate
 - Later, 66 MHz, 64-bit, 528 MB/second peak
 - Upstream bandwidth remain slow for device (~256MB/s peak)
 - **Shared bus with arbitration**
 - Winner of arbitration becomes bus master and can connect to CPU or DRAM through the southbridge and northbridge



PCI as Memory Mapped I/O

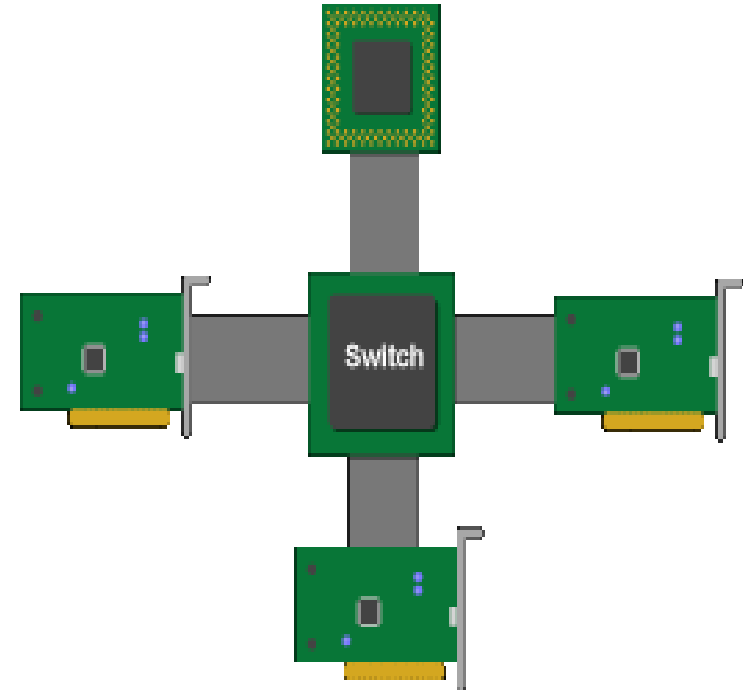
- PCI device registers are mapped into the CPU's physical address space
 - Accessed through loads/stores (kernel mode)
- Addresses are assigned to the PCI devices at boot time
 - All devices listen for their addresses



PCI Express (PCIe)

switched, point-to-point connection

- each card has dedicated “link” to the central switch, with no arbitration
- packet switches: messages form virtual channel
- prioritized packets for QoS (such as for real-time video streaming)

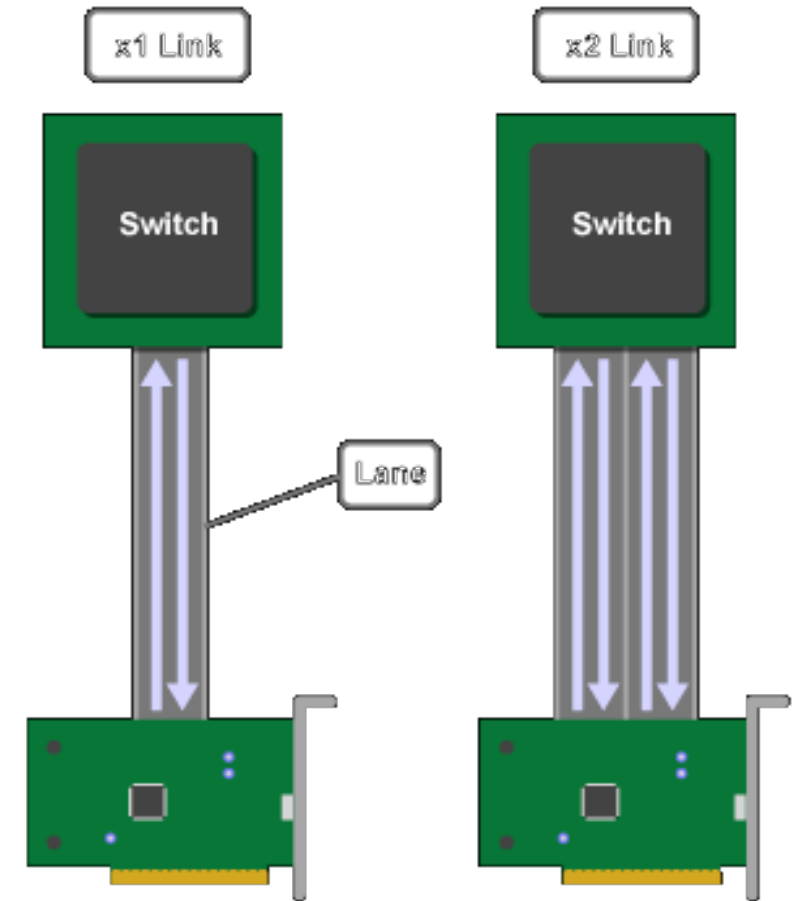


PCIe Generations

- Within a generation, number of lanes in a link can be scaled
 - using distinct physical channels (more bits / wider transfers)
 - $\times 1, \times 2, \times 4, \times 8, \times 16, \times 32, \dots$
- Each new generation aims to double the speed
 - Current generation is PCIe 5.0, however it is supported only on a very limited set of systems, e.g., IBM Power10
 - 32GT/s
 - PCIe 4.0 is supported on modern AMD, Intel, and IBM systems
 - However, PCIe Gen. 3 is still very widely used

PCIe Gen 3 Links and Lanes

- Each link consists of one or more lanes
 - Each lane is 1-bit wide (4 wires, each 2-wire pair can transmit 8Gb/s in one direction)
 - 2-wire pair is used for differential signaling
 - Upstream and downstream simultaneous and symmetric
 - Each Link can combine 1, 2, 4, 8, 12, 16 lanes- x1, x2, etc.
- Each byte data is **128b/130b** encoded into 130 bits with equal number of 1's and 0's; net data rate 7.8768 GB/s per lane each way.
 - Thus, the net data rates are 985 MB/s (x1) 1.97 GB/s (x2), 3.94 GB/s (x4), 7.9 GB/s (x8), 15.8 GB/s (x16), each way



Foundation: 8/10 bit encoding

- Goal is to maintain DC balance while have sufficient state transition for clock recovery
- The difference of 1s and 0s in a 20-bit stream should be ≤ 2
- There should be no more than 5 consecutive 1s or 0s in any stream
- 00000000, 00000111, 11000001 bad
- 01010101, 11001100 good
- Find 256 good patterns among 1024 total patterns of 10 bits to encode an 8-bit datum
- a 20% overhead

Current: 128/130 bit encoding

- Same goal: maintain DC balance while have sufficient state transition for clock recovery
- 1.5% overhead instead of 20%
- Scrambler function: long runs of 0s, 1s vanishingly small
- Instead of guaranteed run length of 8/10b
- At least one bit shift every 66 bits

Patterns Contain Many 0s and 1s

A question for fun:

- if we need **2^{128} code words**
- **chosen from** all 2^{130} **130-bit patterns**
- **how many 0s/1s** must we consider including?

Answer: 63-67 (of either type)

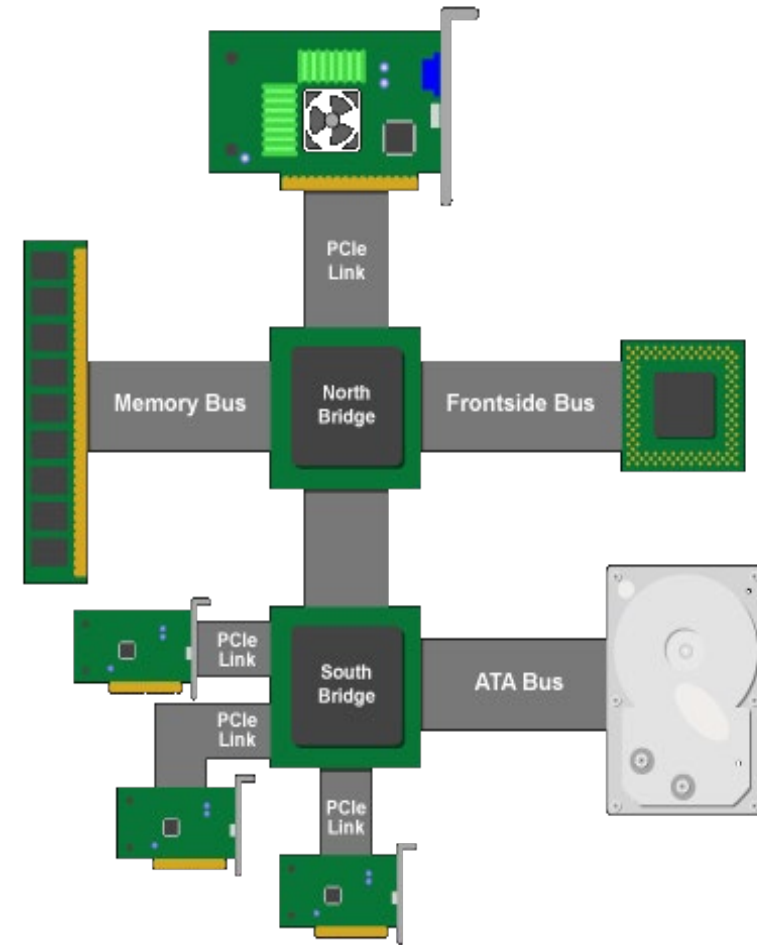
Thus 128b/130b code words are pretty well-balanced,
and have lots of 0-1 transitions (for clock recovery).

Recent PCIe PC Architecture

PCIe forms the interconnect backbone within PC.

Northbridge and Southbridge are PCIe switches.

Source: Jon Stokes, PCI Express: An Overview (<http://arstechnica.com/articles/paedia/hardware/pcie.ars>)



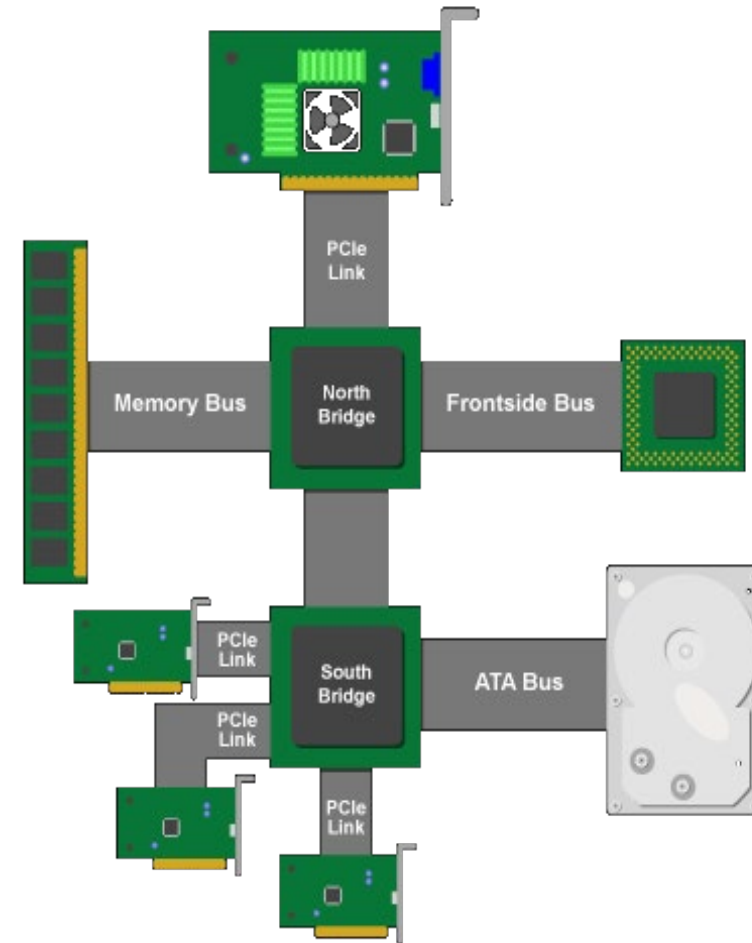
Recent PCIe PC Architecture

How is PCI supported?

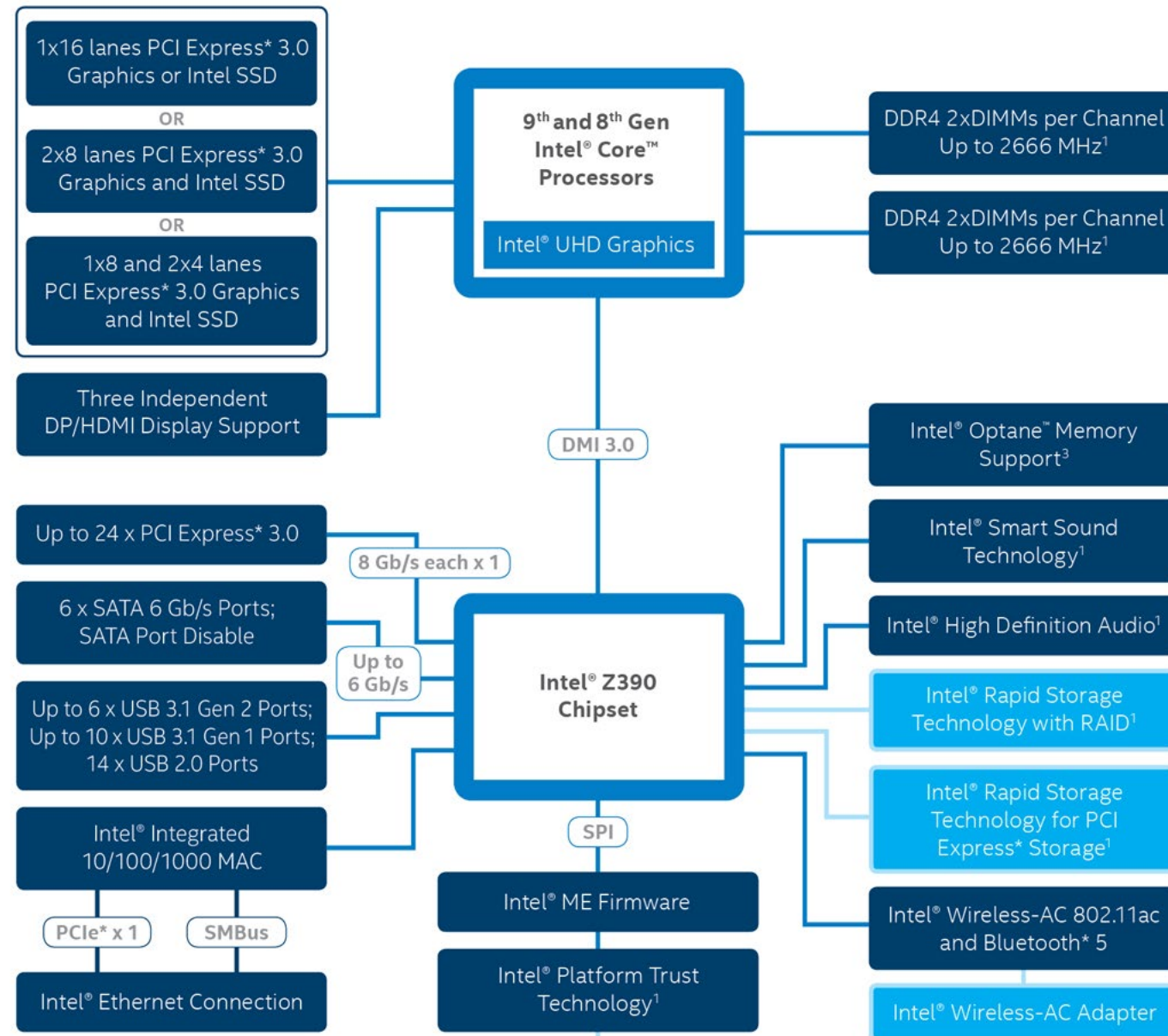
- Need a PCI-PCIe bridge, which is
- sometimes included as part of Southbridge, or
- can add as a separate PCIe I/O card.

Current systems integrate PCIe controllers directly on chip with CPU.

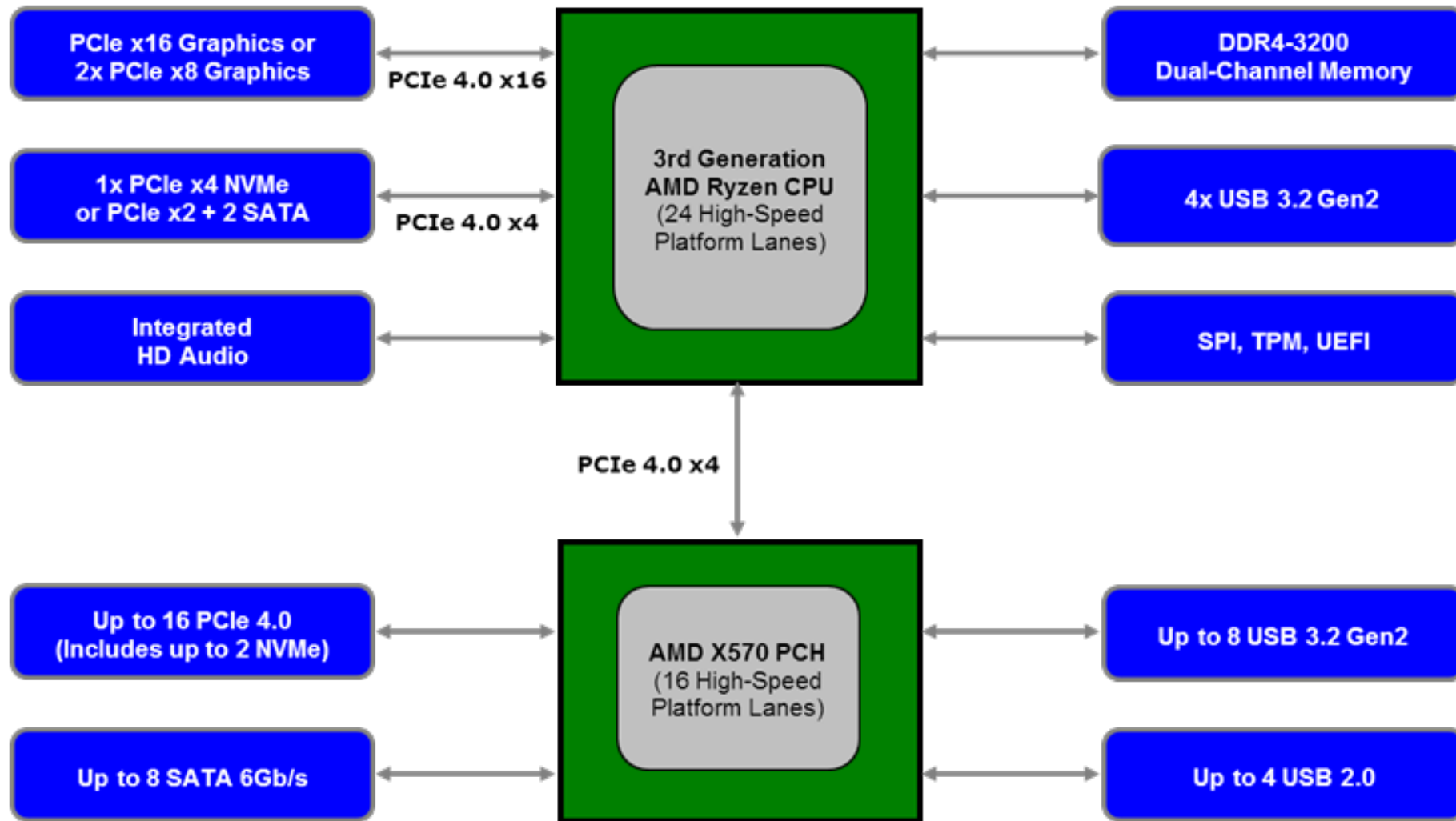
Source: Jon Stokes, PCI Express: An Overview (<http://arstechnica.com/articles/paedia/hardware/pcie.ars>)



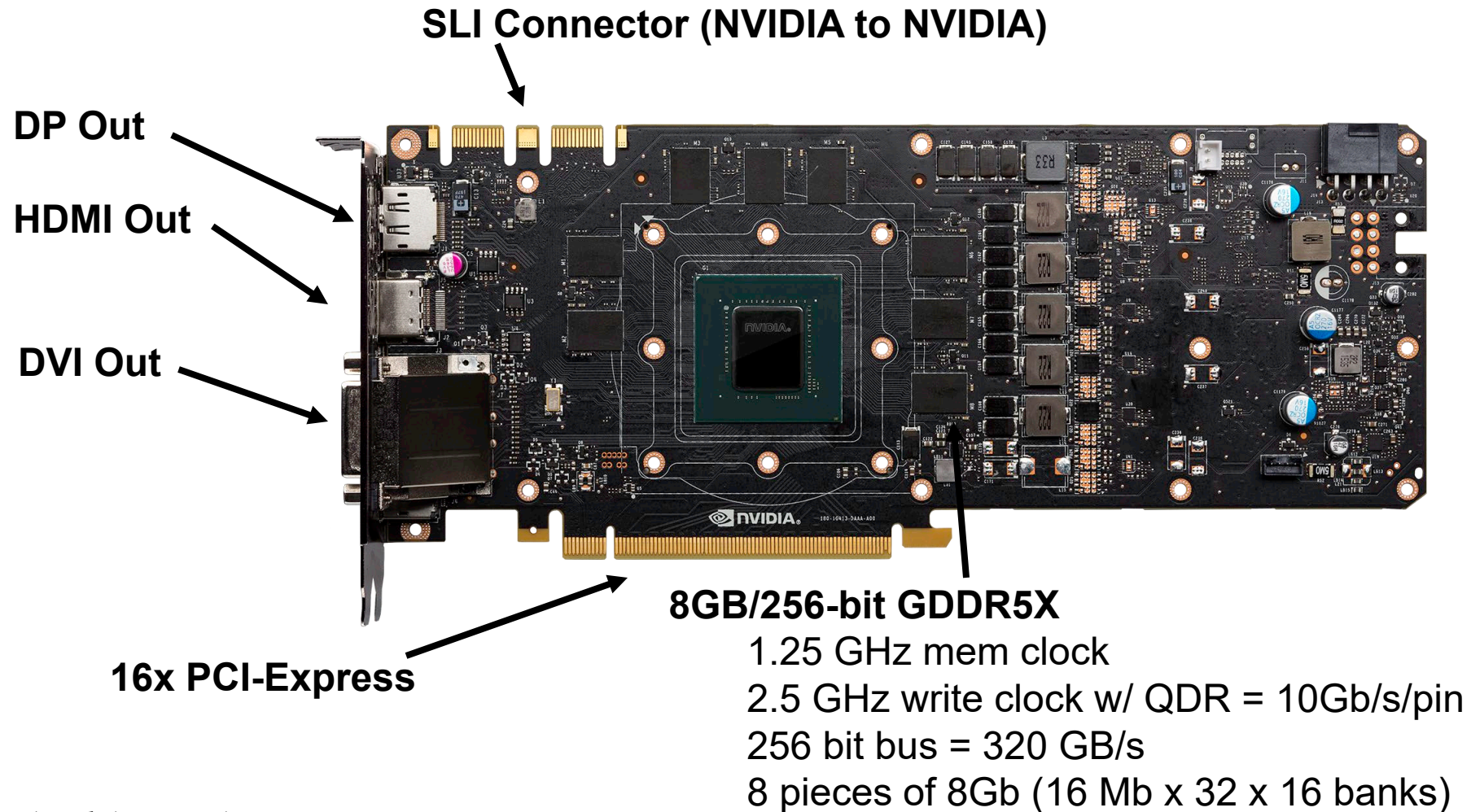
Modern Intel PCIe PC Architecture



Modern AMD PCIe PC Architecture



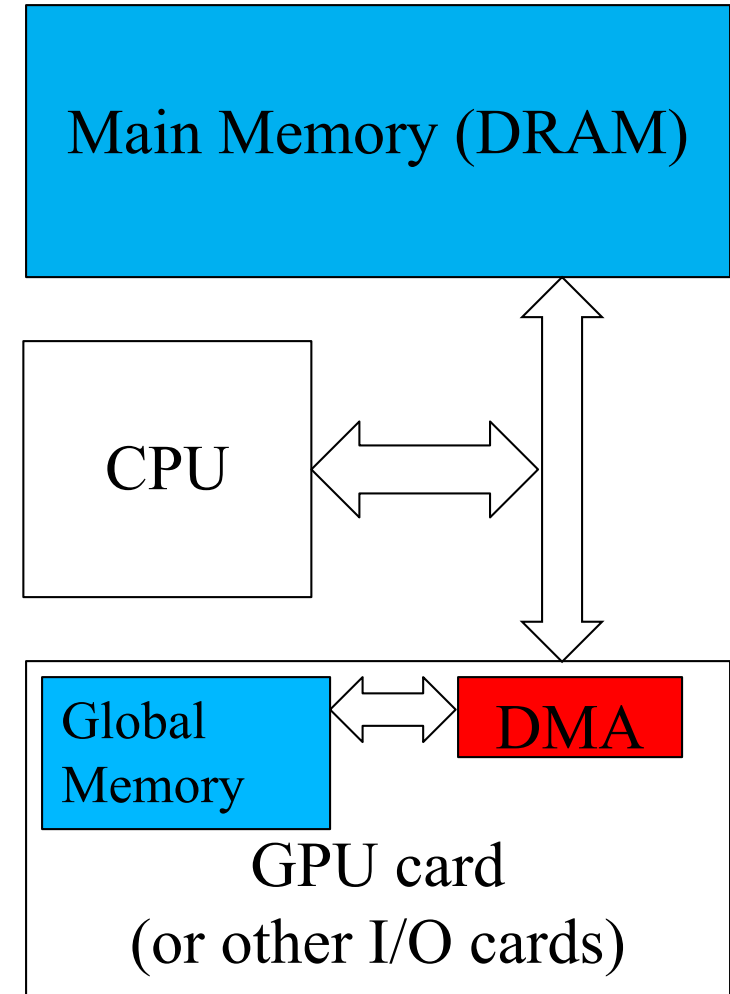
GeForce GTX 1080 (Pascal) GPU Consumer Card Details



PCIe Data Transfer using DMA

DMA (Direct Memory Access) is used to fully utilize the bandwidth of an I/O bus

- DMA uses physical address for source and destination
- Transfers a number of bytes requested by OS
- Needs pinned memory



Pinned Memory

- DMA uses physical addresses
- The OS could accidentally page out the data that is being read or written by a DMA and page in another virtual page into the same location
- Pinned memory cannot be paged out
- If a source or destination of a `cudaMemcpy()` in the host memory is not pinned, it needs to be first copied to a pinned memory – extra overhead
- `cudaMemcpy` is much faster with pinned host memory source or destination

Allocate/Free Pinned Memory (a.k.a. Page Locked Memory)

- `cudaHostAlloc()`
 - Three parameters
 - Address of pointer to the allocated memory
 - Size of the allocated memory in bytes
 - Option – use `cudaHostAllocDefault` for now
- `cudaFreeHost()`
 - One parameter
 - Pointer to the memory to be freed

Using Pinned Memory

- Use the allocated memory and its pointer the same way those returned by `malloc()`;
- The only difference is that the allocated memory cannot be paged by the OS
- The `cudaMemcpy` function should be about 2x faster with pinned memory
- Pinned memory is a limited resource whose over-subscription can have serious consequences

Important Trends

- Knowing yesterday, today, and tomorrow
 - The PC world is becoming flatter
 - CPU and GPU are being fused together
 - Outsourcing of computation is becoming easier...

Two vertical lines, one blue and one orange, are positioned on the left side of the slide.

ANY MORE QUESTIONS