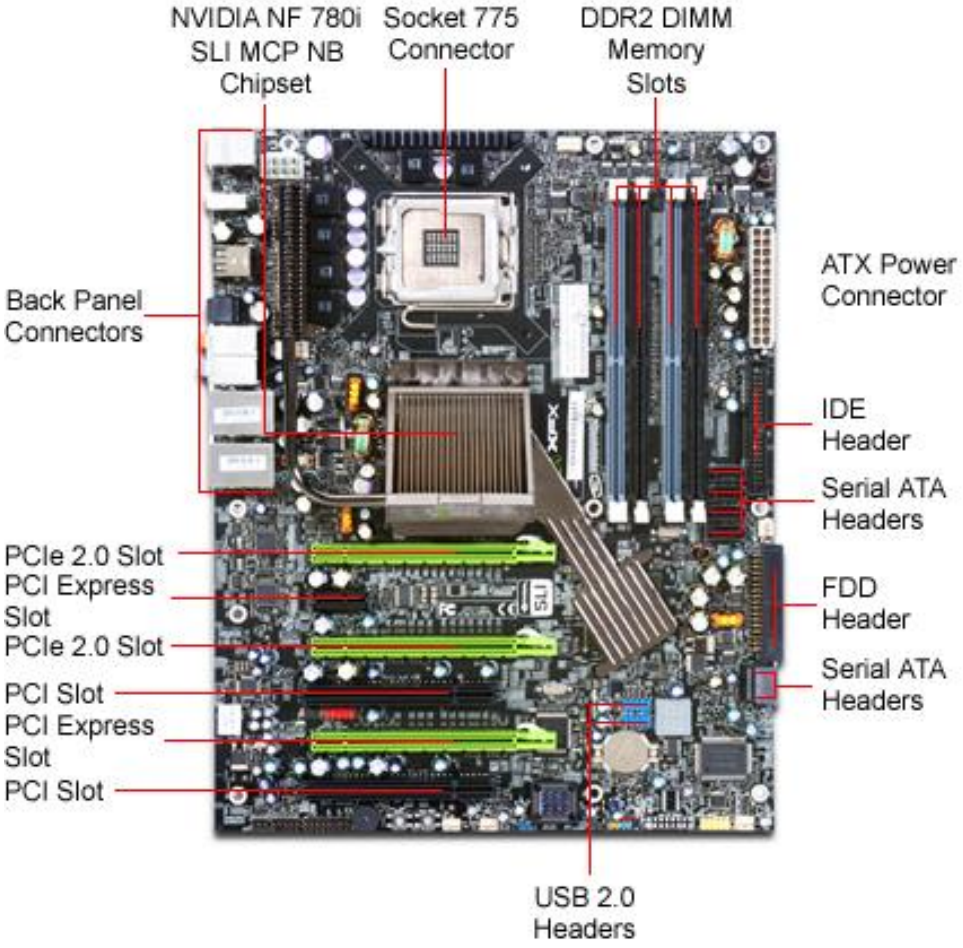
PC and GPU Architecture

Alptekin Temizel

atemizel@metu.edu.tr

A Modern PC Motherboard

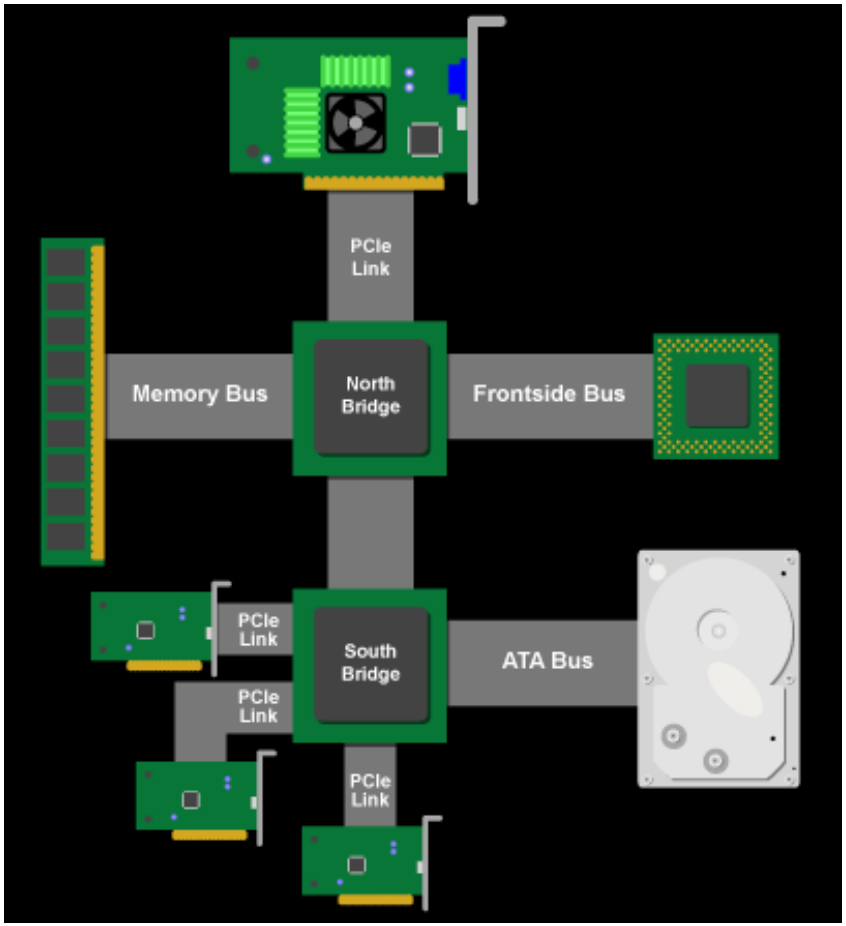


Bandwidth

- The Bandwidth between key components ultimately dictates system performance
  - Especially true for massively parallel systems processing massive amount of data
  - Tricks like buffering, reordering, caching can temporarily defy the rules in some cases

PCIe PC Architecture

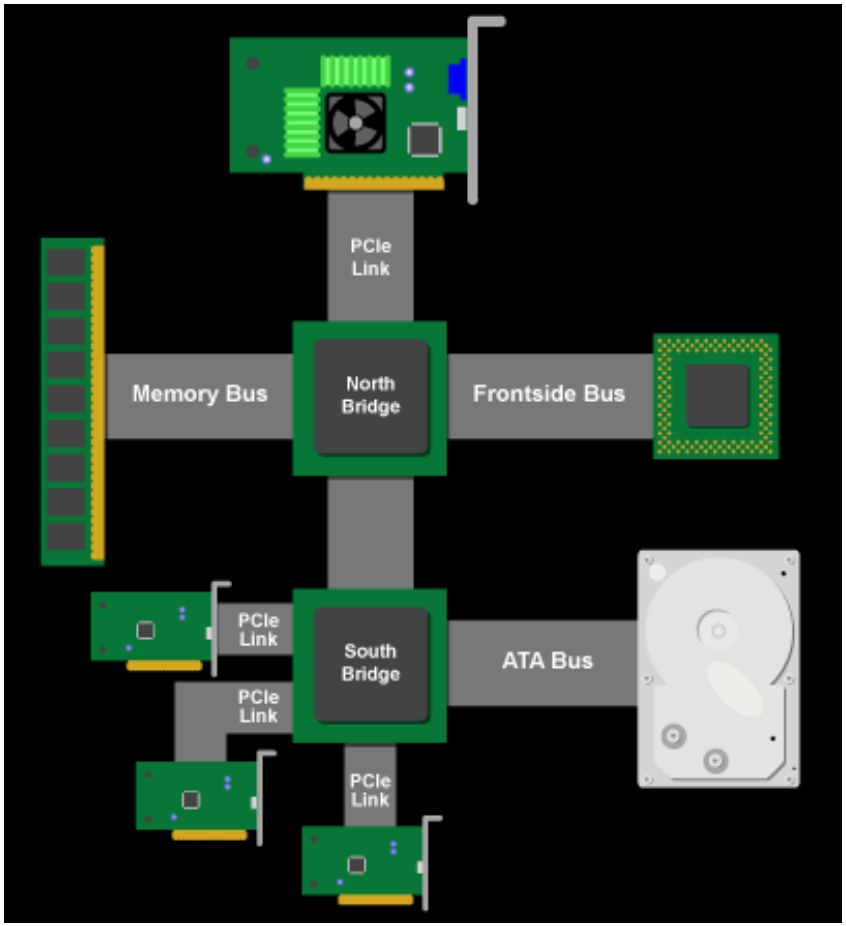
- Northbridge is connected directly to the CPU and thus responsible for tasks that require the highest performance
- Northbridge is usually paired with southbridge.
- These two chips manage communications between the CPU and other parts of the motherboard.



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

PCIe PC Architecture

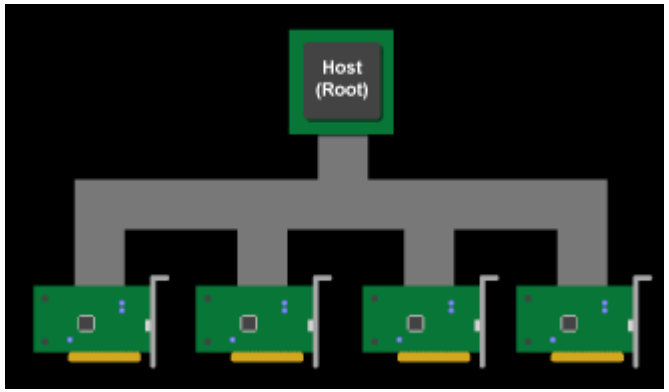
- PCI: Peripheral Component Interconnect
- PCI/PCIe forms the interconnect backbone
- Bridges connect the various busses together
  - Northbridge/Southbridge are both PCIe switches
  - Some Southbridge designs have built-in PCI-PCIe bridge to allow old PCI cards
  - Some PCIe cards are PCI cards with a PCI-PCIe bridge



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

(Original) PCI Bus Specification

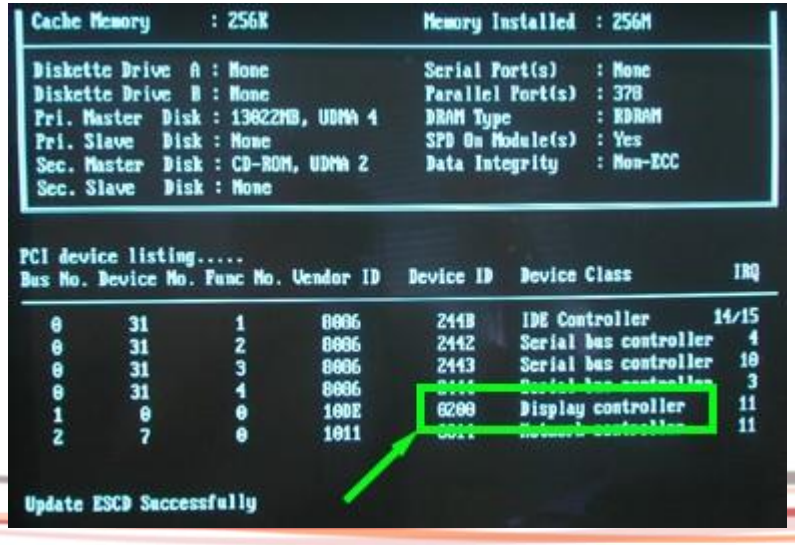
- Connected to the SouthBridge
  - Originally 33 MHz, 32-bit wide, 132 MB/second peak transfer rate
  - More recently 66 MHz, 64-bit-wide, 512 MB/second peak
  - Upstream bandwidth 256MB/s peak
  - **Shared bus** with arbitration
    - Winner of arbitration becomes bus master and can connect to CPU or DRAM through the southbridge and northbridge

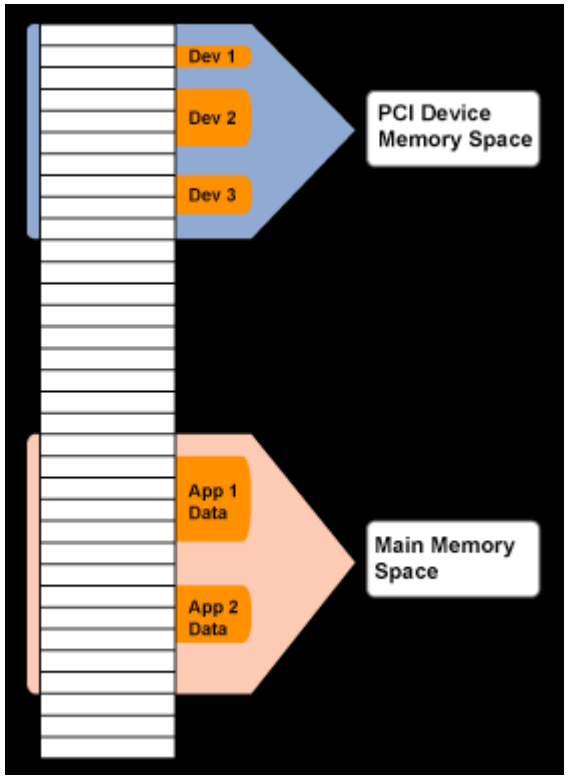


32 bits x $33x10^6$ clock pulses/second
= 4 bytes x $33x10^6$ clock pulses/second
= 132 MB/second

PCI as Memory Mapped I/O

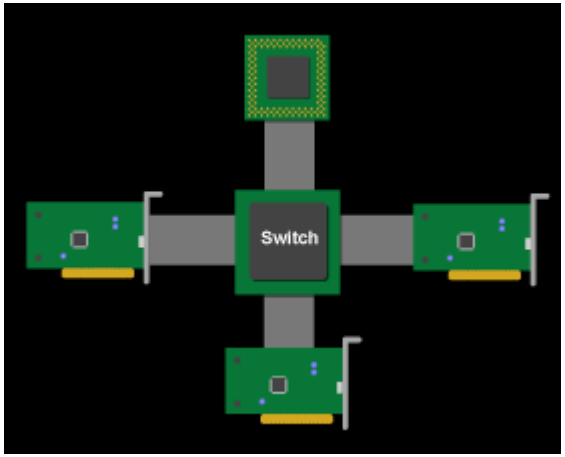
- PCI device registers are mapped into the CPU's physical address space
  - Accessed through loads/ stores (kernel mode)
- Addresses assigned to the PCI devices at boot time
  - All devices listen for their addresses





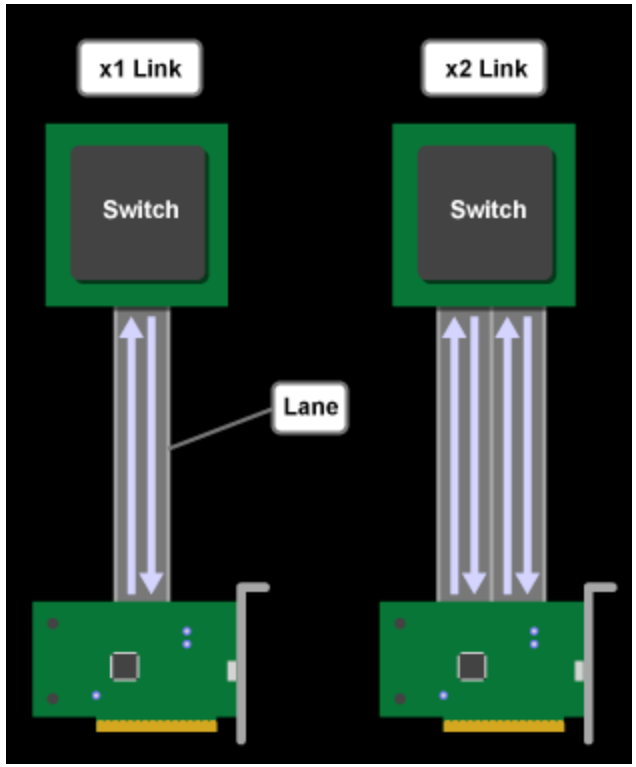
PCI Express (PCIe)

- Switched, point-to-point connection
  - Each card has a dedicated "link" to the central switch, no bus arbitration.
  - Prioritized packets for QoS
    - E.g., real-time video streaming



PCIe Links and Lanes

- Each link consists of one more lanes
  - 1, 2, 4, 8, 12, 16 lanes- x1, x2, etc.



- Thus, the net data rates are
  - 250 MB/s (x1)
  - 500 MB/s (x2)
  - 1GB/s (x4)
  - 2 GB/s (x8)
  - 4 GB/s (x16), each way
- PCI-e 2.0: doubles these speeds
  - 8 GB/s (x16), each way
- PCI-e 3.0: also doubles
  - 16 GB/s (x16), each way

PCIe Links and Lanes

PCIe Bandwidth & Frequency

Year	Bandwidth	Frequency/Speed
1992	133MB/s (32 bit simplex)	33 Mhz (PCI)
1993	533MB/s (64 bit simplex)	66 Mhz (PCI 2.0)
1999	1.06GB/s (64 bit simplex)	133 Mhz (PCI-X)
2002	2.13GB/s (64 bit simplex)	266 Mhz (PCI-X 2.0)
2002	8GB/s (x16 duplex)	2.5 GHz (PCIe 1.x)
2006	16GB/s (x16 duplex)	5.0 GHz (PCIe 2.x)
2010	32GB/s (x16 duplex)	8.0 GHz (PCIe 3.x)
2017	64GB/s (x16 duplex)	16.0 GHz (PCIe 4.0)
2019	128GB/s (x16 duplex)	32.0 GHz (PCIe 5.0)

PCIe Links and Lanes

Motherboard with Four Slots

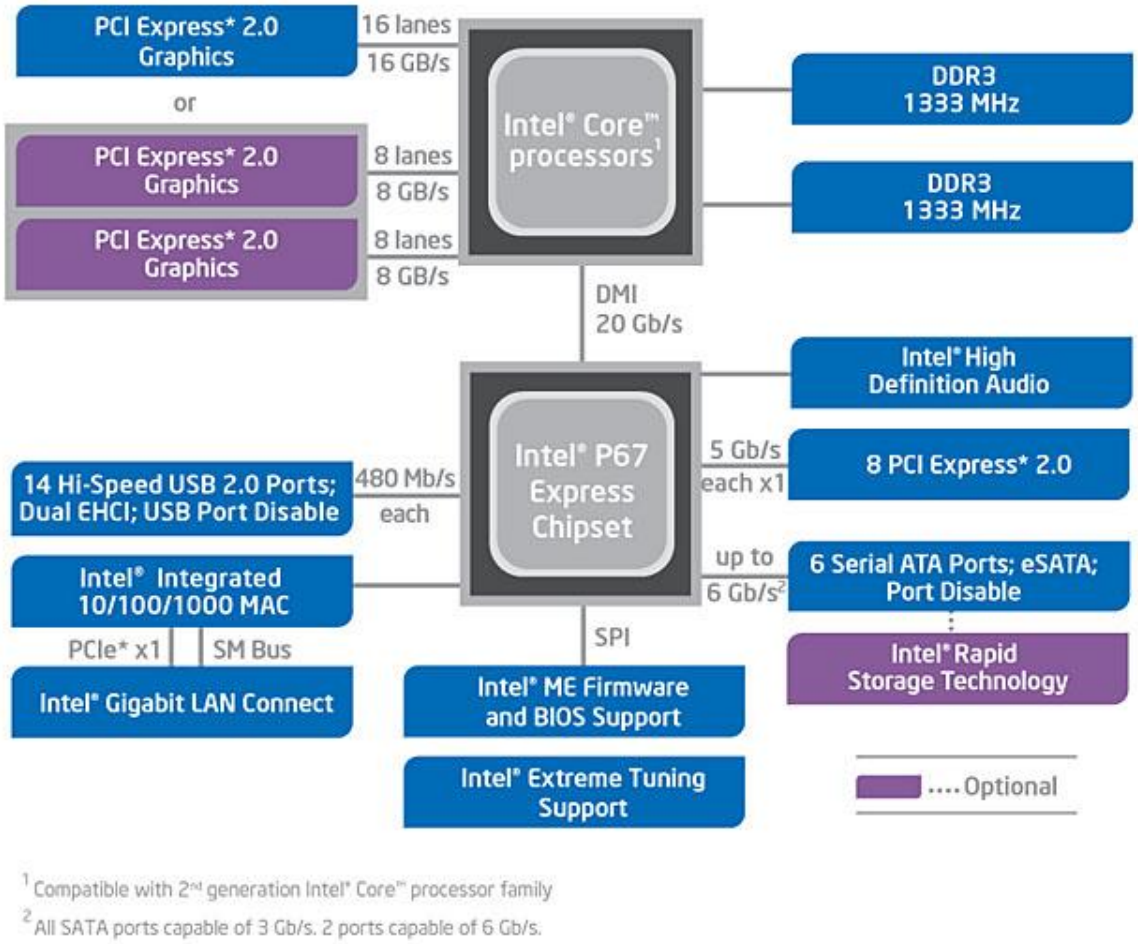
PCI Express x16

PCI

PCI Express x8

PCI-X

Current Intel Architecture

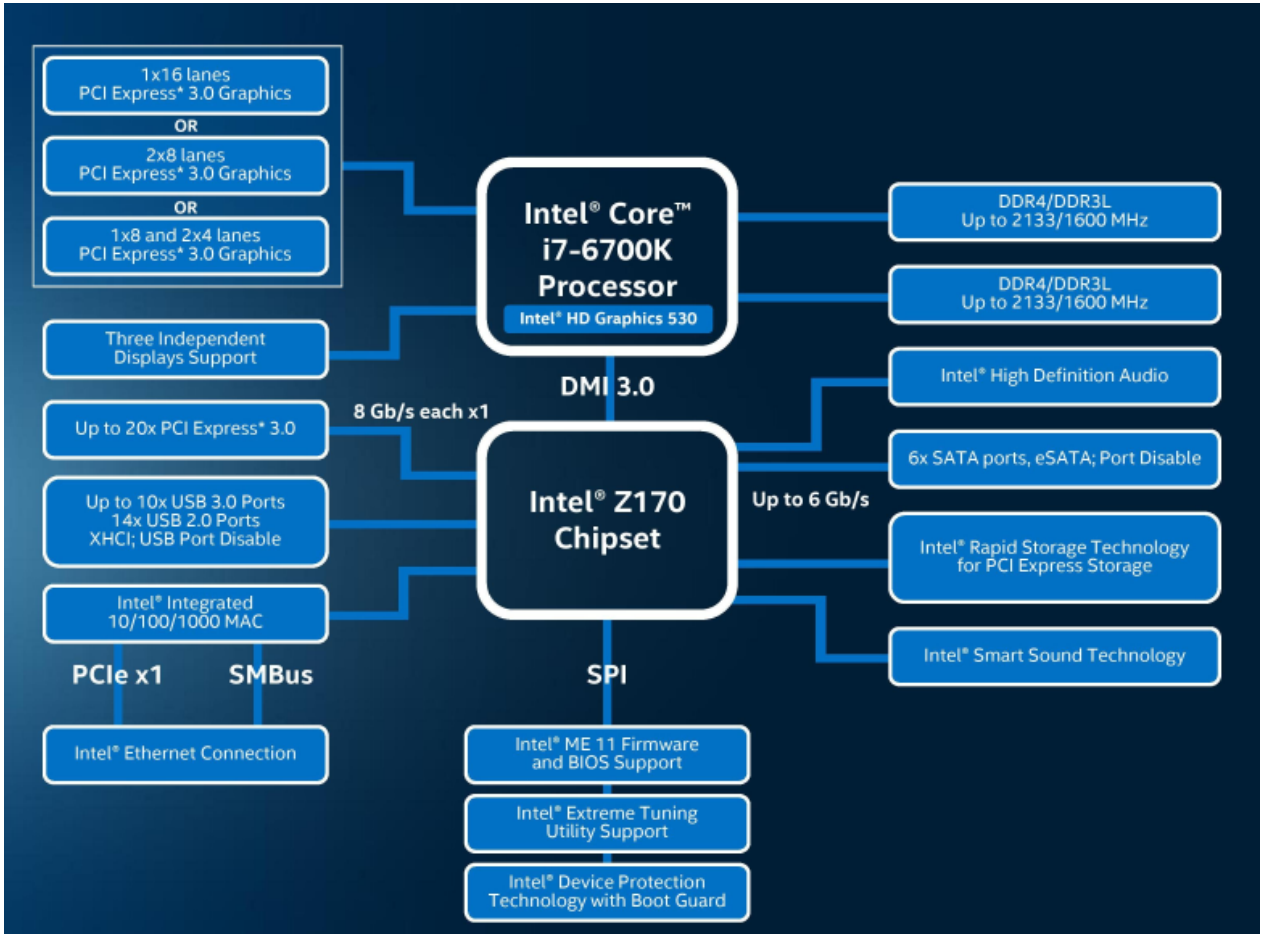


[1] Compatible with 2nd generation Intel® Core™ processor family

[2] All SATA ports capable of 3 Gb/s. 2 ports capable of 6 Gb/s.

Intel's "Sandy Bridge" processors feature full integration of northbridge functions onto the CPU chip, along with processor cores, memory controller and graphics processing unit (GPU).

6th Gen Intel Core Architecture



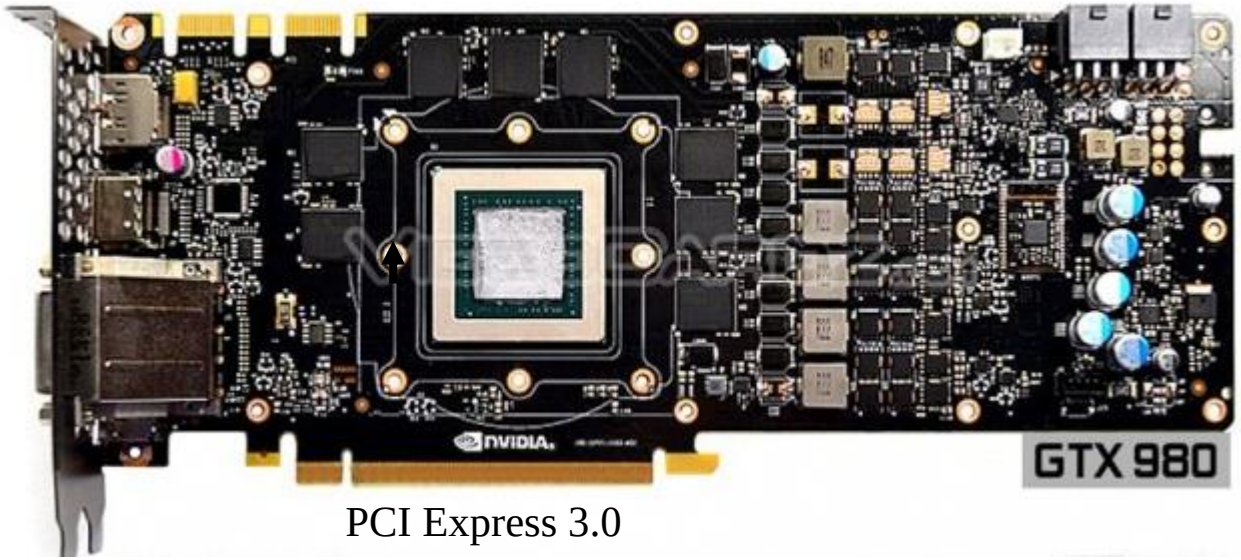
Additional PCI-E 3.0 lanes coming off the Southbridge to enable better implantation of PCI-E Express allowing users to use the next class of SSDs to their full potential without sacrificing GPU performance.

August 2015

Review- GPU Boards

SLI
Connector

DVI , HDMI
connectors



PCI Express 3.0

4 GB
256-bit DDR5
224 GB/sec

Review- GPU Architecture (GM204 – GTX980)

REMINDER: THREADS (1)

- Threads allow separation of loosely coupled parts within a program into **concurrently executing** tasks. Each of these tasks is called a thread.

- More formally, a thread is a sequential flow of control in a program.

- With multiple threads, a program may have multiple sequential flows of control which perform several tasks at the same time.

REMINDER: THREADS (2)

Single core , single CPU machines

• Switch among threads. In order to do this, the available processor time is divided among the threads that need it.

• Each thread is executed for some time and it is suspended when its allocated time slice elapses.

•After the currently executing thread is suspended, another thread waiting for CPU resumes running. By this way, an impression of concurrently executing tasks (or multiple CPUs) is given.

Computers with multiple CPUs or multiple cores:

• Can run threads concurrently in each core.

REMINDER: THREADS (3)

- In many cases, threads simplify program design due to its inherent support for **separation of concerns**.

- With threading, it is also possible to perform operations that take a large amount of time in the background while doing other tasks.

- For example, while doing some processor intensive task, the user interface may remain responsive. This is achieved by implementing the user interface and the task in separate threads.

- In addition, the tasks of varying priority may be distinguished by assigning **different priorities** to each task.

Thread vs. Process

- A *process* is an executing program.
  - Allocated memory by the operating system.
  - **Cannot** access other process' variables or other data structures directly.
    - Possible only via files, pipes or sockets…
- A *thread* is an execution or flow of control in the address space of a process; the program counter register points to the next instruction to be executed.
  - Can access other thread's variables directly
  - Share memory and other system's resources

Concurrency Problems

- *Race conditions:*
  - Threads can try to update the same data structure at the same time.
  - This garbles the data structure, typically causing the next thread that tries to use it to crash or give wrong results.
- Synchronization mechanisms:
  - Locks/Critical Sections
  - Mutexes
  - Queues

Multithreaded programming

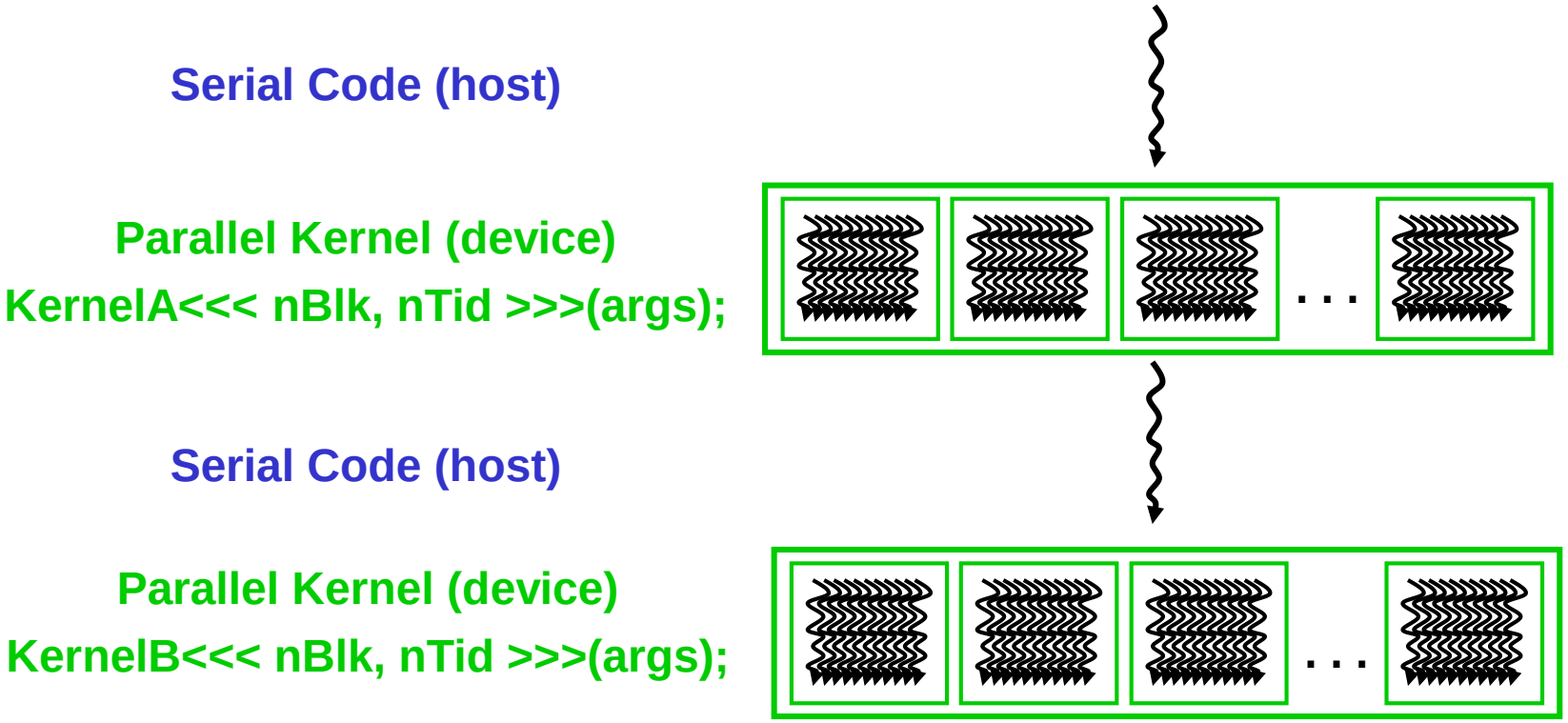
Theory

Practice

Images from 9gag.com

CUDA – C with no shader limitations

- Integrated host+device app C program
  - Serial or modestly parallel parts in **host** C code
  - Highly parallel parts in **device** SPMD kernel C code

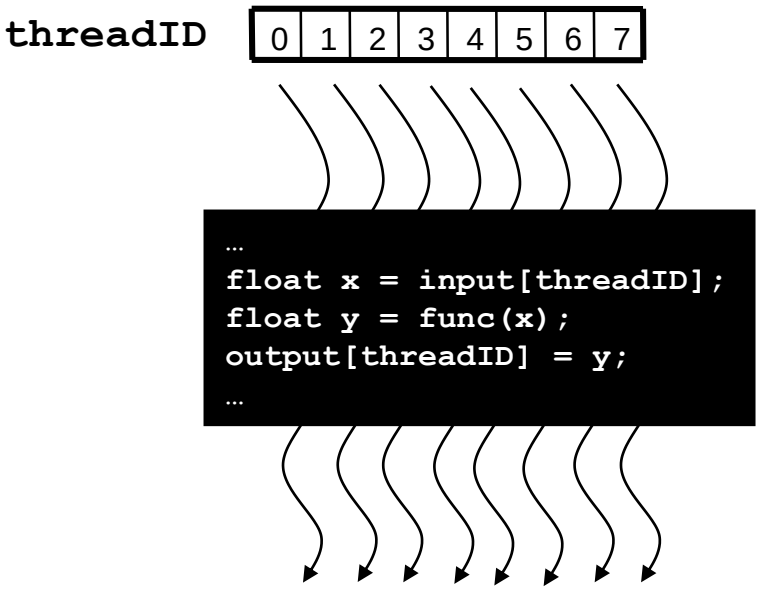


CUDA Devices and Threads

- A compute device
  - Is typically a GPU but can also be another type of parallel processing device
  - Is a coprocessor to the CPU (host)
  - Has its own DRAM (device memory)
  - Runs many threads in parallel
- Data-parallel portions of an application are expressed as device kernels which run on many threads
- Differences between GPU and CPU threads
  - **GPU threads are extremely lightweight**
    - Very little creation overhead
  - **GPU needs 1000s of threads for full efficiency**
    - Multi-core CPU needs only a few

Arrays of Parallel Threads

- A CUDA kernel is executed by an array of threads
  - **All threads run the same code** (SPMD)
  - Each thread has an ID that it uses to **compute memory addresses and make control decisions**



Thread Blocks: Scalable Cooperation

- Divide monolithic thread array into multiple blocks
  - Threads within a block cooperate via **shared memory, atomic operations** and **barrier synchronization**
  - Threads in different blocks cannot cooperate