8.2 GPU system architectures

(Original section[1])

In this section, we survey GPU system architectures in common use today. We discuss system configurations, GPU functions and services, standard programming interfaces, and a basic GPU internal architecture.

Heterogeneous CPU-GPU system architecture

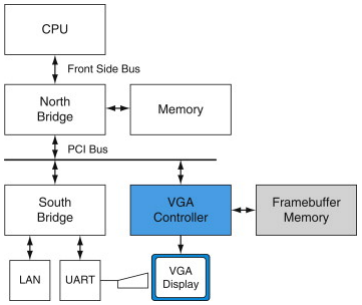
A heterogeneous computer system architecture using a GPU and a CPU can be described at a high level by two primary characteristics: first, how many functional subsystems and/or chips are used and what are their interconnection technologies and topology; and second, what memory subsystems are available to these functional subsystems. See COD Chapter 6 (Parallel Processor from Client to Cloud) for background on the PC I/O systems and chip sets.

The historical PC (circa 1990)

The figure below shows a high-level block diagram of a legacy PC, circa 1990. The north bridge (see COD Chapter 6 (Parallel Processor from Client to Cloud)) contains high-bandwidth interfaces, connecting the CPU, memory, and PCI bus. The south bridge contains legacy interfaces and devices: ISA bus (audio, LAN), interrupt controller; DMA controller; time/counter. In this system, the display was driven by a simple framebuffer subsystem known as a VGA (*video graphics array*) which was attached to the PCI bus. Graphics subsystems with built-in processing elements (GPUs) did not exist in the PC landscape of 1990.

Figure 8.2.1: Historical PC (COD Figure B.2.1).

VGA controller drives graphics display from framebuffer memory.

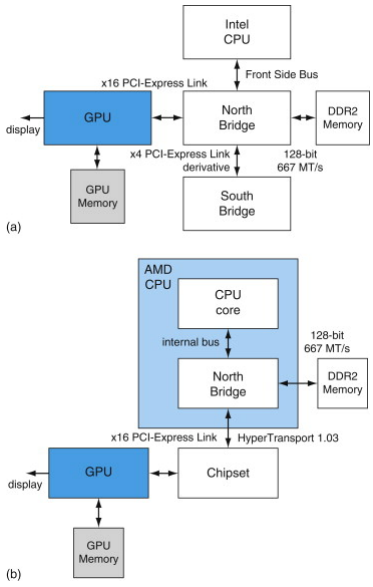


The figure below illustrates two configurations in common use today. These are characterized by a separate GPU (discrete GPU) and CPU with respective memory subsystems. In item a of the figure below, with an Intel CPU, we see the GPU attached via a 16-lane *PCI-Express* 2.0 link to provide a peak 16 GB/s transfer rate, (peak of 8 GB/s in each direction). Similarly, in item b of the figure below, with an AMD CPU, the GPU is attached to the chipset, also via PCI-Express with the same available bandwidth. In both cases, the GPUs and CPUs may access each other's memory, albeit with less available bandwidth than their access to the more directly attached memories. In the case of the AMD system, the north bridge or memory controller is integrated into the same die as the CPU.

PCI-Express (***PCIe***): A standard system I/O interconnect that uses point-to-point links. Links have a configurable number of lanes and bandwidth.

Figure 8.2.2: Contemporary PCs with Intel and AMD CPUs (COD Figure B.2.2).

See COD Chapter 6 (Parallel Processor from Client to Cloud) for an explanation of the components and interconnects in this figure.



A low-cost variation on these systems, a *unified memory architecture (UMA)* system, uses only CPU system memory, omitting GPU memory from the system. These systems have relatively low-performance GPUs, since their achieved performance is limited by the available system memory bandwidth and increased latency of memory access, whereas dedicated GPU memory provides high bandwidth and low latency.

Unified memory architecture (***UMA***): A system architecture in which the CPU and GPU share a common system memory.

A high-performance system variation uses multiple attached GPUs, typically two to four working in parallel, with their displays daisy-chained. An example is the NVIDIA SLI (scalable link interconnect) multi-GPU system, designed for high-performance gaming and workstations.

The next system category integrates the GPU with the north bridge (Intel) or chipset (AMD) with and without dedicated graphics memory.

COD Chapter 5 (Large and Fast: Exploiting Memory Hierarchy) explains how caches maintain coherence in a shared address space. With CPUs and GPUs, there are multiple address spaces. GPUs can access their own physical local memory and the CPU system's physical memory using virtual addresses that are translated by an MMU on the GPU. The operating system kernel manages the GPU's page tables. A system physical page can be accessed using either coherent or noncoherent PCI-Express transactions, determined by an attribute in the GPU's page table. The CPU can access GPU's local memory through an address range (also called aperture) in the PCI-Express address space.

Game consoles

Console systems such as the Sony PlayStation 3 and the Microsoft Xbox 360 resemble the PC system architectures previously described. Console systems are designed to be shipped with identical performance and functionality over a lifespan that can last five years or more. During this time, a system may be reimplemented many times to exploit more advanced silicon manufacturing processes and thereby to provide constant capability at ever lower costs. Console systems do not need to have their subsystems expanded and upgraded the way PC systems do, so the major internal system buses tend to be customized rather than standardized.

GPU interfaces and drivers

In a PC today, GPUs are attached to a CPU via PCI-Express. Earlier generations used *AGP*. Graphics applications call OpenGL [Segal and Akeley, 2006] or Direct3D [Microsoft DirectX Specification] API functions that use the GPU as a coprocessor. The APIs send commands, programs, and data to the GPU via a graphics device driver optimized for the particular GPU.

AGP: An extended version of the original PCI I/O bus, which provided up to eight times the bandwidth of the original PCI bus to a single card slot. Its primary purpose was to connect graphics subsystems into PC systems.

Graphics logical pipeline

The graphics logical pipeline is described in COD Section B.3 (Programming GPUs). The figure below illustrates the major processing stages, and highlights the important programmable stages (vertex, geometry, and pixel shader stages).

Figure 8.2.3: Graphics logical pipeline (COD Figure B.2.3).

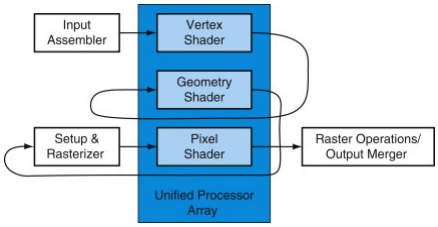
Programmable graphics shader stages are blue, and fixed-function blocks are white.

Mapping graphics pipeline to unified GPU processors

The figure below shows how the logical pipeline comprising separate independent programmable stages is mapped onto a physical distributed array of processors.

Figure 8.2.4: Logical pipeline mapped to physical processors (COD Figure B.2.4).

The programmable shader stages execute on the array of unified processors, and the logical graphics pipeline dataflow recirculates through the processors.



Basic unified GPU architecture

Unified GPU architectures are based on a parallel array of many programmable processors. They unify vertex, geometry, and pixel shader processing and parallel computing on the same processors, unlike earlier GPUs which had separate processors dedicated to each processing type. The programmable processor array is tightly integrated with fixed function processors for texture filtering, rasterization, raster operations, anti-aliasing, compression, decompression, display, video decoding, and high-definition video processing. Although the fixed-function processors significantly outperform more general programmable processors in terms of absolute performance constrained by an area, cost, or power budget, we will focus on the programmable processors here.

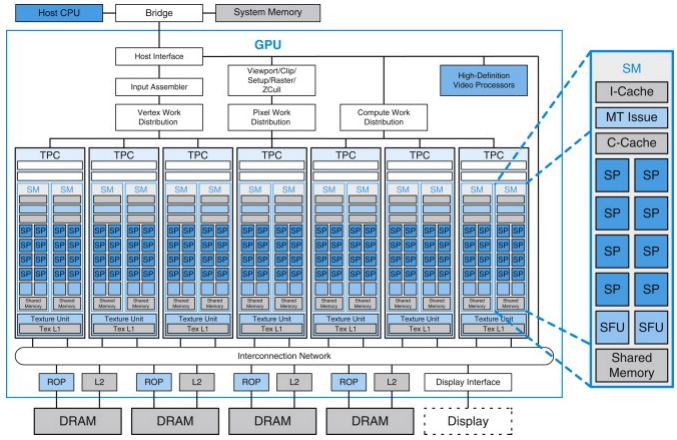
Compared with multicore CPUs, manycore GPUs have a different architectural design point, one focused on executing many parallel threads efficiently on many processor cores. By using many simpler cores and optimizing for data-parallel behavior among groups of threads, more of the per-chip transistor budget is devoted to computation, and less to on-chip caches and overhead.

Processor array

A unified GPU processor array contains many processor cores, typically organized into multithreaded multiprocessors. The figure below shows a GPU with an array of 112 *streaming processor* (SP) cores, organized as 14 multithreaded *streaming multiprocessors* (SMs). Each SP core is highly multithreaded, managing 96 concurrent threads and their state in hardware. The processors connect with four 64-bit-wide DRAM partitions via an interconnection network. Each SM has eight SP cores, two *special function units* (SFUs), instruction and constant caches, a multithreaded instruction unit, and a shared memory. This is the basic Tesla architecture implemented by the NVIDIA GeForce 8800. It has a unified architecture in which the traditional graphics programs for vertex, geometry, and pixel shading run on the unified SMs and their SP cores, and computing programs run on the same processors.

Figure 8.2.5: Basic unified GPU architecture (COD Figure B.2.5).

Example GPU with 112 *streaming processor* (SP) cores organized in 14 *streaming multiprocessors* (SMs); the cores are highly multithreaded. It has the basic Tesla architecture of an NVIDIA GeForce 8800. The processors connect with four 64-bit-wide DRAM partitions via an interconnection network. Each SM has eight SP cores, two *special function units* (SFUs), instruction and constant caches, a multithreaded instruction unit, and a shared memory.



The processor array architecture is scalable to smaller and larger GPU configurations by scaling the number of multiprocessors and the number of memory partitions. The figure above shows seven clusters of two SMs sharing a texture unit and a texture L1 cache. The texture unit delivers filtered results to the SM given a set of coordinates into a texture map. Because filter regions of support often overlap for successive texture requests, a small streaming L1 texture cache is effective to reduce the number of requests to the memory system. The processor array connects with *raster operation processors* (ROPs), L2 texture caches, external DRAM memories, and system memory via a GPU-wide interconnection network. The number of processors and number of memories can scale to design balanced GPU systems for different performance and market segments.

(*1) This section is in original form.

Provide feedback on this section