Register No

SRM Institute of Science and Technology

College of Engineering and Technology

School of Computing

SRM Nagar, Kattankulathur – 603203, Chengalpattu District, Tamilnadu

Academic Year: 2023-24 **SET-C-Answer Key**

Test: CLA-T1 **Date: 14.02.2024**
Course Code & Title: 18CSE419T-GPU Programming **Duration:** 1 Hour
Year & Sem: III Year / VI Sem **Max. Marks:** 25

Course Articulation Matrix:

S.No.	Course Outcome	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO10	PO11	PO12
1	CO1	3	2										

Part – A **(5 x 1 = 5 Marks)** **Instructions: Answer all Questions**						
Q. No	**Question**	**Marks**	**BL**	**CO**	**PO**	**PI Code**
1	The power dissipation of CPUs is proportional to a) Clock frequency b) Supply voltage c) Third power of clock frequency d) Third power of supply voltage ans: A	1	L1	1	1	1.3.1
2	NVIDIA GTX680 graphics processing unit (GPU) has a) 12,384 threads b) 16,384 threads, c) 4096 threads d) 32,384 threads Ans: B	1	L1	1	1	1.1.2
3	Which one is referred as a throughput oriented design? a) CPU b) GPU c) GPGPU d) CPU-GPU Ans: B	1	L2	1	1	1.1.2

4	Which one is mostly used in GPU? a) SISD b) SIMD c) MISD d) MIMD Ans:B	1	L1	1	1	1.3.1
5	How many double precision floating-point values could be stored on a 128-bit register? a) 1 b) 2 c) 3 d) 4 Ans:B	1	L2	1	2	2.1.1

Part – B(2 x 4 = 8 Marks)
Answer Any Two Questions

6. Compare and contrast CPU and GPU.

CPU	GPU
A smaller number of larger cores (up to 24)	A larger number (thousands) of smaller cores
Low latency	High throughput
Optimized for serial processing	Optimized for parallel processing
Designed for running complex programs	Designed for simple and repetitive calculations
Performs fewer instructions per clock	Performs more instructions per clock
Automatic cache management	Allows for manual memory management
Cost-efficient for smaller workloads	Cost-efficient for bigger workloads

7. Bring out the advantages and disadvantages of multi-thread processors.

Many-threads processors, especially the GPUs, have led the race of floating-point performance since 2003. As of 2012, the ratio of peak floating-point calculation throughput between many-thread GPUs and multicore CPUs is about 10. These are not necessarily application speeds, but are merely the raw speed that the execution resources can potentially support in these chips: 1.5 teraflops versus 150 gigaflops double precision in 2012.

•**Small cache memories** are provided to help control the bandwidth requirements of these applications so that multiple threads that access the same memory data do not need to all go to the DRAM.

•This design style is commonly referred to **as throughput-oriented design** since it strives to maximize the total execution throughput of a large number of threads while allowing individual threads to take a potentially much longer time to execute.

•Memory bandwidth is another important issue. The speed of many applications is limited by the rate at which data can be delivered from the memory system into the processors. Graphics chips have been operating at approximately six times the memory bandwidth of contemporaneously available CPU chips.

8. Draw the architecture of CUDA capable GPU and explain.

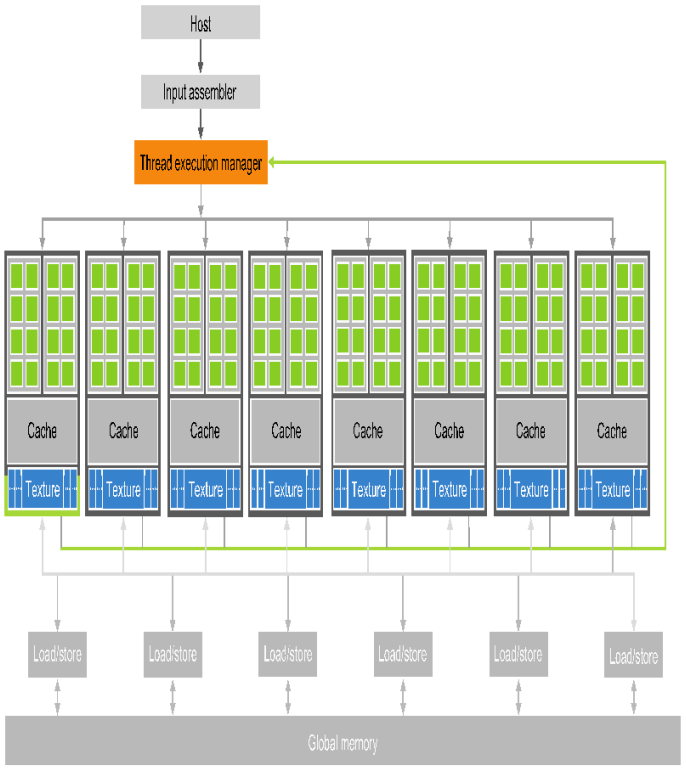


FIGURE 1.2

Architecture of a CUDA-capable GPU.

Part – C(1 x12 = 12 Marks)

Figure 1.2 shows the architecture of a typical CUDA-capable GPU. It is organized into an array of highly threaded streaming multiprocessors (SMs). In Figure 1.3, two SMs form a building block. However, the number of SMs in a building block can vary from one generation of CUDA GPUs to another generation. Also, in Figure 1.3, each SM has a number of streaming processors (SPs) that share control logic and an instruction cache. Each GPU currently comes with multiple gigabytes of Graphic Double Data Rate (GDDR) DRAM, referred to as global memory in Figure 1.3. These GDDR DRAMs differ from the system DRAMs on the CPU motherboard in that they are essentially the frame buffer memory that is used for graphics. For graphics applications, they hold video images and texture information for 3D rendering. But for computing, they function as very high bandwidth off-chip memory, though with somewhat longer latency than typical system memory. For massively parallel applications, the higher bandwidth makes up for the longer latency.
The G80 introduced the CUDA architecture and had 86.4 GB/s of memory bandwidth, plus a communication link to the CPU core logic over a PCI-Express Generation 2 (Gen2) interface. Over PCI-E Gen2, a CUDA application can transfer data from the system memory to the global memory at 4 GB/s, and at the same time upload data back to the system memory at 4 GB/s. Altogether, there is a combined total of 8 GB/s. More recent GPUs use PCI-E Gen3, which supports 8 GB/s in each direction. As the size of GPU memory grows, applications increasingly keep their data in the global memory and only occasionally use the PCI-E to communicate with the CPU system memory if there is need for using a library that is only available on the CPUs. The communication bandwidth is also expected to grow as the CPU bus bandwidth of the system memory grows in the future.
With 16,384 threads, the GTX680 exceeds 1.5 teraflops in double precision. A good application typically runs 5,000_12,000 threads simul- taneously on this chip. For those who are used to multithreading in CPUs, note that Intel CPUs support two or four threads, depending on the machine model, per core. CPUs, however, are increasingly used with SIMD (single instruction, multiple data) instructions for high numerical performance. The level of parallelism supported by both GPU hardware and CPU hardware is increasing quickly. It is therefore very important to strive for high levels of parallelism when developing computing applications.

9. With a suitable drawing illustrate the design and decision factors involved in choosing the CPU and GPU for heterogeneous parallel computing.

The design of a CPU is optimized for sequential code performance. It makes use of sophisticated control logic to allow instructions from a single thread to execute in parallel or even out of their sequential order while maintaining the appearance of sequential execution.

• More importantly, large cache memories are provided to reduce the instruction and data access latencies of large complex applications.

•Neither control logic nor cache memories contribute to the peak calculation speed. As of 2012, the high-end general-purpose multicore microprocessors typically have six to eight large processor cores and multiple megabytes of on-chip .

•Memory bandwidth is another important issue. The speed of many applications is limited by the rate at which data can be delivered from the memory system into the processors. Graphics chips have been operating at approximately six times the memory bandwidth of contemporaneously available CPU chips.

•The design philosophy of GPUs is shaped by the fast-growing video game industry that exerts tremendous economic pressure for the ability to perform a **massive number of floating-point calculations** per video frame in advanced games.

•This demand motivates GPU vendors to look for ways to maximize the chip area and power budget dedicated to floating-point calculations.

•The prevailing solution is to optimize for the execution throughput of massive numbers of threads. The design saves chip area and power by allowing pipelined memory channels and arithmetic operations to have long latency.

• The reduced area and power of the memory access hardware and arithmetic units allows the designers to have more of them on a chip and thus increase the total execution throughput.

•**Small cache memories** are provided to help control the bandwidth requirements of these applications so that multiple threads that access the same memory data do not need to all go to the DRAM.

•This design style is commonly referred to **as throughput-oriented design** since it strives to maximize the total execution throughput of a large number of threads while allowing individual threads to take a potentially much longer time to execute.

•The CPUs, on the other hand, are designed to minimize the execution latency of a single thread.

•Large last-level on-chip caches are designed to capture frequently accessed data and convert some of the long-latency memory accesses into short-latency cache accesses.

•The arithmetic units and operand data delivery logic are also designed to minimize the effective latency of operation at the cost of increased use of chip area and power.

•By reducing the latency of operations within the same thread, the CPU hardware reduces the execution latency of each individual thread. However, the large cache memory, low-latency arithmetic units, and sophisticated operand delivery logic consume chip area and power that could be otherwise used to provide more arithmetic execution units and memory access channels.
This design style is commonly referred to as **latency-oriented design**.

•GPUs are designed as parallel, throughput oriented computing engines and they will not perform well on some tasks on which CPUs are designed to perform well.

•For programs that have one or very few threads, CPUs with lower operation latencies can achieve much higher performance than GPUs.

•When a program has a large number of threads, GPUs with higher execution throughput can achieve much higher performance than CPUs.

•Therefore, one should expect that many applications use both CPUs and GPUs, executing the sequential parts on the CPU and numerically intensive parts on the GPUs.

•This is why the CUDA programming model, introduced by NVIDIA in 2007, is designed to support joint CPU-GPU execution of an application.

• The demand for supporting joint CPUGPU execution is further reflected in more recent programming models such as OpenCL , OpenACC , and C++ AMP.

-

Several other factors can be even more important.

- First and foremost, the processors of choice must have a very large presence in the marketplace, referred to as the installed base of the processor.
- The reason is very simple. The cost of software development is best justified by a very large customer population.
- Applications that run on a processor with a small market presence will not have a large customer base.
- This has been a major problem with traditional parallel computing systems that have negligible market presence compared to general-purpose microprocessors.

- Only a few elite applications funded by government and large corporations have been successfully developed on these traditional parallel computing systems.

- **Another important decision factor is practical form factors and easy accessibility.**
- Until 2006, parallel software applications usually ran on data center servers or departmental clusters. But such execution environments tend to limit the use of these applications. For example, in an application such as medical imaging, it is fine to publish a paper based on a 64-node cluster machine.
- But actual clinical applications on magnetic resonance imaging (MRI) machines have been based on some combination of a PC and special hardware accelerators.

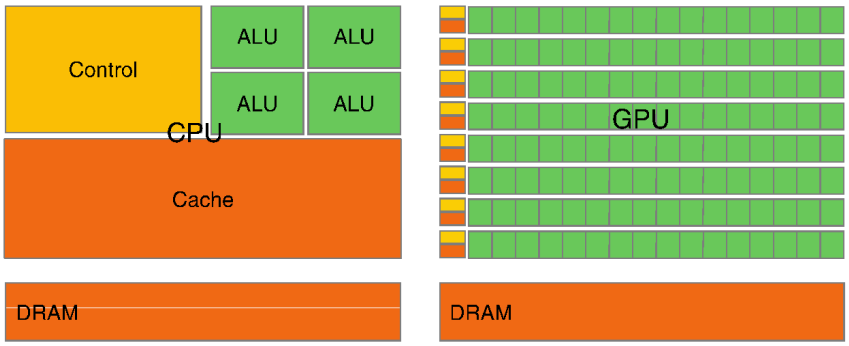


FIGURE 1.1

CPUs and GPUs have fundamentally different design philosophies.

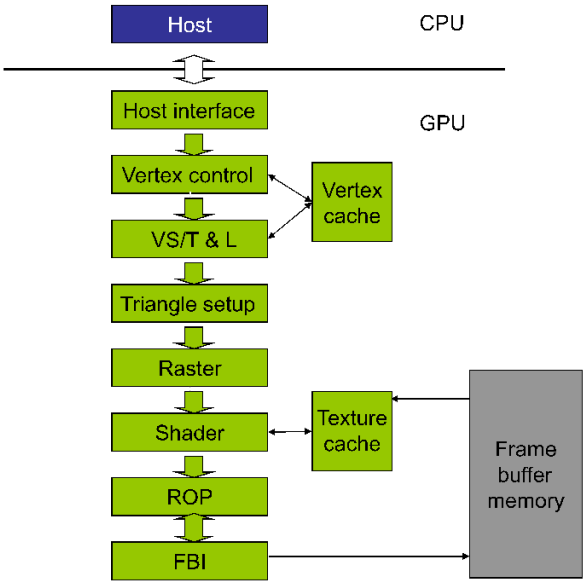
10. Draw the process of fixed function NVIDIA graphics pipeline operations in detail.

the leading performance graphics hardware was fixed-function pipelines that were configurable, but not pro- grammable. In that same era, major graphics Application Programming Interface (API) libraries became popular. An API is a standardized layer of software, that is, a collection of library functions that allows applica- tions (e.g., games) to use software or hardware services and functionality. For example, an API can allow a game to send commands to a graphics processing unit to draw objects on a display. One such API is DirectX, Microsoft's proprietary API for media functionality. The Direct3D compo- nent of DirectX provides interface functions to graphics processors. The other major API is OpenGL, an open-standard API supported by multiple vendors and popular in professional workstation applications. This era of fixed-function graphics pipeline roughly corresponds to the first seven generations of DirectX.

he host interface receives graphics commands and data from the CPU. The commands are typically given by application programs by calling an API function. The host interface typically contains a specialized DMA hardware to efficiently transfer bulk data to and from the host system memory to the graphics pipeline. The host interface also communicates back the status and result data of executing the commands.

Before we describe the other stages of the pipeline, we should clarify that the term vertex usually means the "corners" of a polygon. The

GeForce graphics pipeline is designed to render triangles, so vertex is typi- cally used to refer to the corners of a triangle. The surface of an object is drawn as a collection of triangles. The finer the sizes of the triangles are, the better the quality of the picture typically becomes. The vertex control stage in Figure 2.1 receives parameterized triangle data from the CPU. The vertex control stage converts the triangle data into a form that the hardware understands and places the prepared data into the vertex cache.
The vertex shading, transform, and lighting (VS/T&L) stage in Figure 2.1 transforms vertices and assigns per-vertex values (colors, nor- mals, texture coordinates, tangents, etc.). The shading is done by the pixel shader hardware. The vertex shader can assign a color to each vertex but it is not applied to triangle pixels until later. The triangle setup stage further creates edge equations that are used to interpolate colors and other per- vertex data (e.g., texture coordinates) across the pixels touched by the tri- angle. The raster stage determines which pixels are contained in each tri- angle. For each of these pixels, the raster stage interpolates per-vertex values necessary for shading the pixel, which includes color, position, and texture position that will be shaded (painted) on the pixel.
The shader stage in Figure 2.1 determines the final color of each pixel. This can be generated as a combined effect of many techniques: interpola- tion of vertex colors, texture mapping, per-pixel lighting mathematics, reflections, and more. Many effects that make the rendered images more realistic are incorporated in the shader stage



JRE 2.1

ked-function NVIDIA GeForce graphics pipeline.