**CUDA-Enabled GPU Architecture**

A **CUDA-enabled GPU** refers to a Graphics Processing Unit that supports **Compute Unified Device Architecture (CUDA)**, a parallel computing platform and API model developed by NVIDIA. CUDA allows developers to harness the GPU's massive parallel processing power for general-purpose computing (GPGPU), not just graphics rendering.

**1. High-Level Architecture of CUDA-Enabled GPUs**

A CUDA-enabled GPU is composed of several important architectural components:

* **Streaming Multiprocessors (SMs)**: The core execution units, containing multiple CUDA cores.
* **CUDA Cores**: Lightweight, simple ALUs (Arithmetic Logic Units) capable of handling thousands of threads simultaneously.
* **Warp Scheduler**: Handles execution of threads in groups of 32 (called warps).
* **Memory Units**:
  + **Registers** (per thread)
  + **Shared Memory** (per block/SM)
  + **Global Memory** (device-wide)
  + **L1 Cache** (per SM) and **L2 Cache** (shared across all SMs)
  + **Constant, Local, and Texture Memory**

**2. Thread Hierarchy in CUDA**

CUDA uses a hierarchical execution model:

* **Thread**: Basic unit of execution.
* **Block**: A group of threads that can share data through shared memory.
* **Grid**: A collection of thread blocks that execute a kernel function.

Example:

\_\_global\_\_ void add(int \*a, int \*b, int \*c) {

int index = threadIdx.x + blockIdx.x \* blockDim.x;

c[index] = a[index] + b[index];

}

Here, each thread performs one element addition.

**3. Execution Model: SIMT**

CUDA follows the **Single Instruction, Multiple Threads (SIMT)** model:

* Threads are grouped into **warps** (32 threads).
* All threads in a warp execute the same instruction, but on different data.
* Warp divergence (e.g., due to if-else branching) can slow performance.

**4. Memory Access in CUDA**

Efficient use of memory is critical for performance:

* **Registers**: Fastest, per-thread, but limited in number.
* **Shared Memory**: Fast and allows cooperation among threads in a block.
* **Global Memory**: Large, but slow—should be accessed in a coalesced manner.
* **Constant Memory**: Read-only and cached—efficient for uniform access.

**5. Key Features of CUDA Architecture**

* **Massive parallelism**: Thousands of threads executing concurrently.
* **Scalable model**: From small to very large workloads.
* **Hierarchical memory**: Enables flexible data sharing and access optimization.
* **Asynchronous execution**: Overlap computation and communication via streams.
* **Support for Unified Memory**: Allows shared memory space between CPU and GPU.

**6. Architectural Generations (Compute Capabilities)**

Each CUDA architecture generation has a *Compute Capability* (e.g., 3.0, 5.2, 7.5, 8.6):

| **Architecture Name** | **Compute Capability** | **Key Features** |
| --- | --- | --- |
| Kepler | 3.x | Energy-efficient, 192 cores/SM |
| Maxwell | 5.x | Improved efficiency and shared memory |
| Pascal | 6.x | Unified memory, larger shared memory |
| Volta | 7.0 | Tensor cores introduced |
| Turing | 7.5 | Ray tracing and AI support |
| Ampere | 8.0 / 8.6 | More tensor cores, performance boost |
| Hopper | 9.0 | For large-scale AI, advanced features |