Semiconductor Technology from A to Z

Digital technology

CPU vs. GPU

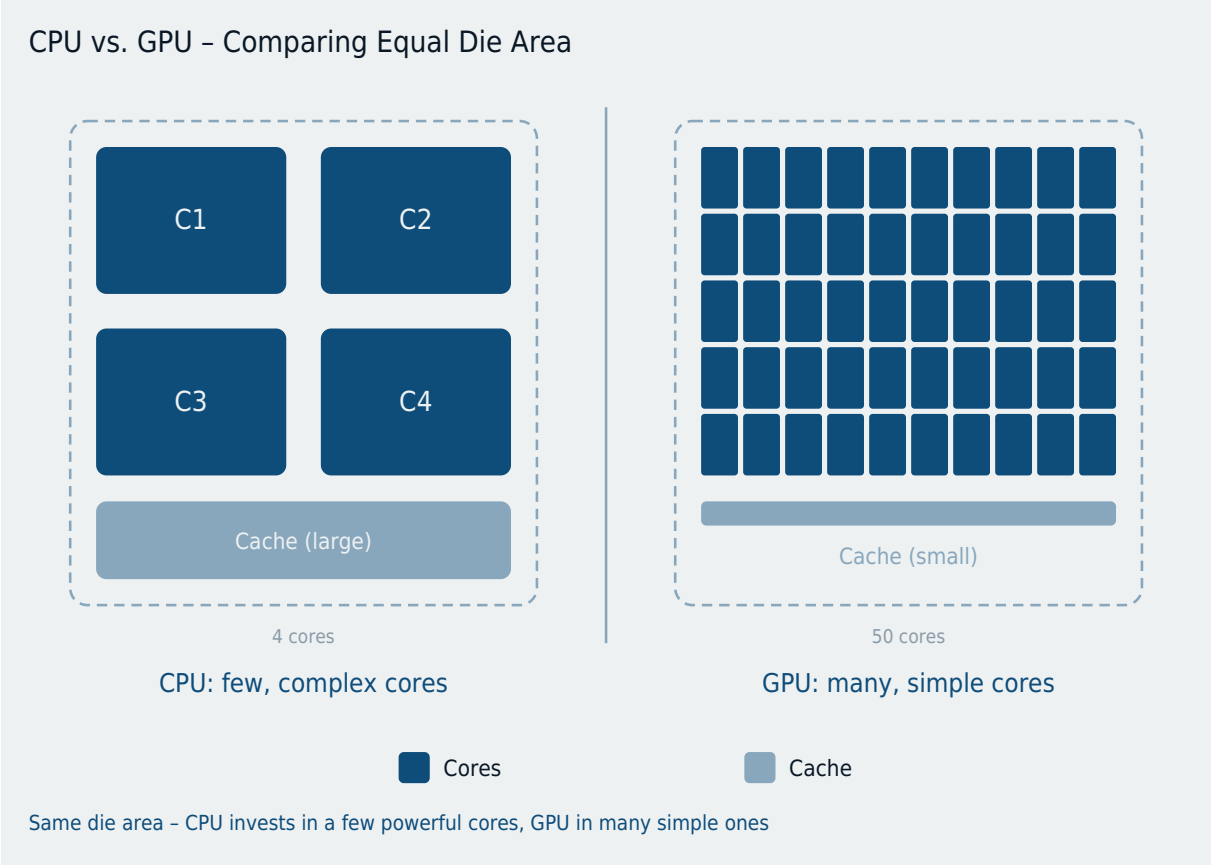
Overview

CPU and GPU are both processors in the sense of the previous chapter – both consist of a control unit, execution units, registers, and a cache hierarchy, connected via a bus. The decisive difference lies in their objective: the CPU (Central Processing Unit) is designed to work through a few, often interdependent tasks sequentially as fast as possible. The GPU (Graphics Processing Unit) is designed to process very many similar, independent tasks simultaneously. Both approaches make sense – just for different kinds of problems.

Architectural Difference

A CPU typically has only a few cores (currently around 4 to 16 in consumer hardware), but each individual core is highly complex: sophisticated control logic, deep pipelining, branch prediction, and out-of-order execution (instructions are not necessarily executed in program order, but as soon as their data is ready), to achieve high per-core speed even for complex, branch-heavy programs.

A GPU takes the opposite approach: instead of a few complex cores, it contains hundreds to thousands of simple, heavily stripped-down cores. These operate on the SIMT principle (Single Instruction, Multiple Threads) – the same instruction is applied simultaneously to many different data elements, for example to every pixel in an image region or every entry in a matrix. A single GPU core is considerably slower and less flexible than a CPU core – its strength lies in sheer numbers.



Memory Hierarchy and Bandwidth

Both architectures use the cache hierarchy described in the previous chapter, but with different goals. The CPU is optimised for low latency: each core has a comparatively large cache to answer individual memory accesses as quickly as possible – important when a sequential program flow is waiting on a result.

The GPU, by contrast, is optimised for high throughput: the cache per core is small, but main memory (usually GDDR6 or HBM rather than ordinary DDR DRAM) is connected with an extremely wide interface, delivering many times the bandwidth of a CPU's memory connection. Instead of avoiding wait times through a large cache, the GPU hides them: while one thread is waiting for data, the hardware switches without delay to another thread that is already ready – with thousands of threads active at once, a single wait barely registers.

Application Areas

The CPU is the right choice for sequential program logic with many branches: operating systems, application software, database queries – anywhere the next step depends on the result of the previous one and is hard to parallelise in advance.

The GPU was originally developed for the graphics pipeline, where the same calculation (lighting, texturing, transformation) is applied to millions of pixels or vertices simultaneously – a classic data-parallel problem. This exact property also makes GPUs attractive for other data-parallel tasks: matrix multiplications, scientific simulations, and above all training neural networks, which at its core consists of enormous numbers of similar, parallelisable matrix operations. This is the main reason GPUs are today's dominant hardware for AI training.

Comparison Table

Property	CPU	GPU
Number of cores	few (4–16)	many (hundreds to thousands)
Core type	complex, high clock speed	simple, lower clock speed
Execution model	out-of-order, speculative	SIMT (parallel, uniform)
Cache per core	large	small
Memory bandwidth	moderate	very high (GDDR6/HBM)
Optimisation goal	low latency	high throughput
Typical application	sequential programs, OS	graphics, AI training, simulation