FPGA vs CPU vs GPU vs Microcontroller

	CPU	FPGA	GPU	ASIC
Overview	Traditional sequential processor for general-purpose applications	Flexible collection of logic elements and IP blocks that can be configured and changed in the field	Originally designed for graphics; now used in a wide range of computationally intensive applications	Custom integrated circuit optimized for the end application
Processing	Single- and multi-core MCUs and MPUs, plus specialized blocks: FPU, etc.	Configured for application; SoCs include hard or soft IP cores (e.g., Arm)	Thousands of identical processor cores	Application-specific: may include third-party IP cores
Programming	OSes, APIs run huge range of high-level languages; assembly language	Traditionally HDL (Verilog, VHDL); newer systems include C/C++ via openCL & SDAccel	OpenCL & Nvidia's CUDA API allow general-purpose programming (e.g., C, C++, Python, Java, Fortran)	Application-specific: TensorFlow open-source framework for Google's TPU; CPU manufacturers (e.g., Intel) include tools with new ASIC releases
Peripherals	Wide choice of analog and digital peripherals in MCUs; MPUs include digital bus interfaces	SoCs include many transceiver blocks, configurable I/O banks	Very limited; e.g., only cache memory	Tailored to application: may include industry-standard functions (USB, Ethernet, etc.)
Strengths	Versatility, multitasking, ease of programming	Configurable for specific application; configuration can be changed after installation; high performance per watt; accommodates massively parallel operation; wide choice of features: DSPs, CPUs	Massive processing power for target applications—video processing, image analysis, signal processing	Custom-designed for application with optimum combination of performance and power consumption
Weaknesses	OS capability adds high overhead; optimized for sequential processing with limited parallelism	Relatively difficult to program; second-longest development time; poor performance for sequential operations; not good for floating-point operations	High power consumption, not suited to some algorithms; problems must be reformulated to take advantage of parallelism, but API frameworks provide abstraction	Longest development time; high cost; cannot be changed without redesigning the silicon

It's also worth considering how these choices stack up in some common applications. As shown in the table, designers can often use any or all of the options either alone or, more likely, in combination.

Applications	CPU	FPGA	GPU	ASIC	Comments
Vision & image processing		✓	✓	✓	FPGA may give way to ASIC in high-volume applications
AI training			✓		GPU parallelism well-suited for processing terabyte data sets in reasonable time
AI inference	✓	✓	✓	✓	Everyone wants in! FPGAs perhaps leading; high-end CPUs (e.g., Intel's Xeon) and GPUs (e.g., Nvidia's T4) address this market
High-speed Search	✓	✓	✓	✓	Microsoft's Bing uses FPGAs; Google uses TPU ASIC; CPU needed for coordination & control
Industrial motor control	(✓)	✓		✓	Many motor-control MCUs and ASICs available; FPGAs offer a quick-turn ASIC alternative
Supercomputer HPC	✓		✓		Majority of TOP500 supercomputers uses some combination of CPUs and GPUs
General-purpose computing	✓		(✓)		CPU most versatile, flexible option; GPUs beginning to perform some tasks
Embedded control	✓	✓		✓	CPUs (→ MCU) dominant in low-cost, space-constrained, low-power, mobile applications
Prototyping, low-volume		✓			FPGAs best choice for low-volume, high-end applications; also pre-silicon validation, post-silicon validation and firmware development