Heterogeneous Computing for Parallelism & Power Efficiency

Definitions, insights, and tools for balancing performance, power, and heat on today's embedded devices

Qualcomm
developer network

Embedded systems like in today's smartphones and IoT devices must handle a variety of workloads ranging from AI at the device edge to ultra-high resolution 3D graphics. In most cases, they must make efficient use of power-constrained sources like batteries while satisfying heat and thermal constraints.

The solution to these challenges lies in ***heterogeneous computing***. By incorporating a variety of processors and cores, devices with heterogeneous computing architectures allow developers to balance performance, power consumption, and thermal constraints across a variety of processing tasks.

In this eBook, we explore heterogeneous computing and give you the tools you need to build powerful yet energy-efficient solutions that meet thermal constraints.

Key Processors

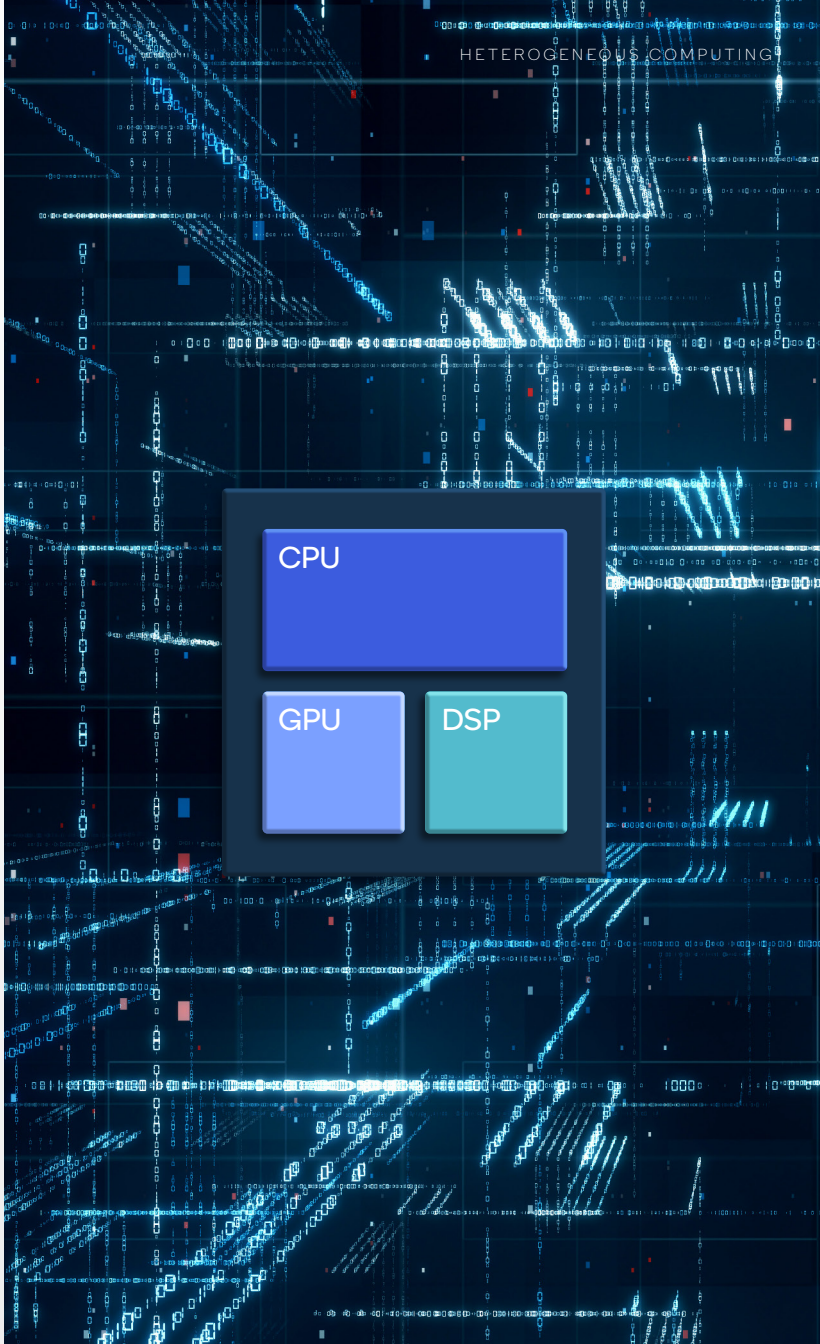
While conventional systems once employed power-hungry, general-purpose CPU cores to handle workloads, true heterogeneous computing architectures harness the power of specialized processors designed to maximize power and performance efficiencies. Three common compute resources found in these system-on-chips (SoCs) are:

CPU (Central Processing Unit): performs the bulk of the regular processing work, including the executional logic and general-purpose instructions found in all software applications. Today, **multi-core** processors allow multiple threads to be executed in parallel. Each **core** is essentially a separate processing unit within the processor, capable of reading and executing instructions. Efficient multi-threading and parallel processing is often accomplished by using small, energy-efficient cores for lighter tasks and big, powerful cores for larger tasks.

GPU (Graphics Processing Unit): is a specialized processor for rendering video and graphics. Modern GPUs excel at complex math computations like the vector processing used in 3D graphics. As a result, they're also widely used in machine learning (ML) which relies on similar operations.

DSP (Digital Signal Processor): is a specialized processor for high-performance compute ranging from processing digital signals collected from the outside world (e.g., via sensors) to running 3D vector operations. For example, the DSP can be used in cameras for image processing, while its vector support can further augment ML inference.

These processors are used to achieve **Parallelism**, where instructions are executed at the same time. This can be done both at the processor core *level* (i.e., running multiple threads across a CPU's cores) and by *offloading* tasks from the CPU to specialized processors, all of which run in concert asynchronously.



What's Driving Heterogeneous Computing?

The following are a few factors driving the need for heterogeneous computing:

New Device Types: an increasing number of device types beyond mobile phones now exist. These range from sophisticated new AR and VR headsets to a variety of new IoT devices, all of which demand high-performance compute.

Moore's Law is Ending: Moore's law, which famously stated that each year would see a doubling of transistors in an integrated circuit (IC), is slowly coming to an end.* Chip designers are reaching the limits of how many transistors can be crammed onto a single IC, leading designers to increase performance through other means like adding cores and specialized processors.

Growth of ML: ML relies heavily on vector math. Initially, this helped to drive the growth of GPUs, which in turn has led to the creation of specialized vector processing units like DSPs.

Growth of Gaming: the popularity of mobile gaming continues to grow at astounding rates, further driving the need for powerful yet efficient mobile GPUs.

New Types of Applications: CPUs offer large caches to minimize thread latency, while GPUs often have more cores running at lower frequencies, making them suitable for high throughput. The ability to harness both allows developers to support a growing variety of applications.

* https://www.brainspire.com/blog/end-of-moores-law-whats-next-for-the-future-of-computing

Heterogeneous Strategies & Considerations

Many heterogeneous systems include rich, flexible APIs that remove the complexity of writing multi-threaded, multi-processor applications. These APIs often provide high-level constructs which allow programmers to make efficient use of the on-board processors (e.g., to allocate tasks both at compile time and dynamically at runtime, to the most appropriate core or processor). Below are a few general strategies to consider:

- Identify which parts of your application require higher versus lower performance early on, and allocate them accordingly.
- You should also have an understanding of the types of operations performed best by each processor, and allocate work accordingly. Be prepared to embed your logic within the constructs provided by the API for a given processor (e.g., thread, processor image, RPC call, etc.).
- In general:
  - small cores can be used when an application has many small threads.
  - large cores are suitable for a few complicated threads (e.g., processing complicated control-flow graphs involving lots of pointers).

  Consequently, assigning complicated threads to simple cores can result in poor performance, while assigning simple threads to bigger, more powerful cores consumes more power than required. In general, developers strive to execute tasks on small cores as long as the results are produced within the required amount of time.
- Monitor your code and task system to avoid bottlenecking a single core with a thread while most other cores are idle or under-utilized.
- Excessive or unnecessary wakeups of idle cores, which can result in more power consumption. One solution is to pack tasks onto the same core while trying not to overload a core.

Example Use Cases for Allocating Computations

AI in General

Move workload from CPU to GPU, DSP, or some combo

Computational Photography

Move workload from CPU to GPU or DSP

Audio Processing

Move workload from the CPU to the DSP

Use Cases for Heterogeneous Computing

Heterogeneous computing architectures can benefit a number of segments:

3D Graphics, Mobile Gaming, Mixed Reality: 3D graphics-intensive apps must run on handheld and headsets devices while remaining cool enough for the user to hold/wear and fulfilling their demands for long battery life. While the CPU handles game logic, graphics processing tasks like 3D vector and matrix math are offloaded to the GPU. Similarly, the execution of shaders, which form a programmable graphics pipeline for pre- and post-processing effects, also run on the GPU.

Machine Learning (ML): like 3D graphics, today's advanced ML algorithms make heavy use of vector and matrix math. This is because many ML algorithms are based around neural networks where vectors represent collections of weights. These algorithms benefit by being run on GPUs and specialized DSPs, both capable of high-performance vector and matrix math, leaving the CPU to work on other tasks like business logic.

IoT: edge computing is pushing developers to build devices that can process data and even perform AI at the device edge, right where data is collected from sensors. This allows edge devices to make decisions and reduce the amount of data sent to the cloud. Heterogeneous platforms allow developers to optimize for power efficiency by allocating tasks to the most appropriate resources.

Wearables: smartwatches, ECG monitors, and other types of wearable sensors now provide users and medical practitioners with a wealth of information on vital signs. These devices often assume small form factors, and must remain both cool and power efficient (where small batteries are often the main power source). Here, specialized processors like DSPs can provide efficient processing for data like signals collected from onboard sensors.

Audio/Video: signal processing is a common task when dealing with audio and video data. This can range from compressing/decompressing (codecs) data to identifying patterns (e.g., to remove noise, sharpen images, etc.). Here, DSPs often take on this task, leaving the CPU free to run application logic.

Snapdragon® Mobile Platforms: Heterogeneous from the Core

Snapdragon mobile platforms are comprised of three main processors that make them inherently heterogeneous:

Qualcomm® Kryo™ CPU: an ARM-based CPU featuring multiple cores configured in a **big.LITTLE** architecture (i.e., consists of small and large cores). The Kryo CPU also supports multiple task scheduling methods, which programmers can control through APIs.

Qualcomm® Adreno™ GPU: primarily used for rendering, developers have the option to offload CPU operations (e.g., vector processing) to our Adreno GPU, which can be run in parallel with the CPU.

Qualcomm® Hexagon™ DSP: provides both CPU and DSP functionality to which developers can offload compute-intensive tasks. Our Hexagon DSP excels at handling vector data and provides hardware multithreading. It is optimized to work under reduced clock rates while accomplishing more work per clock cycle.

Hexagon has evolved from a DSP for image and audio processing, into a vector-processing powerhouse for neural networks, virtual reality, and other process-intensive applications. Developers effectively upload functions to our Hexagon processor, where it performs operations using its multiple hardware threads, very long instruction word (VLIW) instruction set, and real-time operating systems (for scheduling). Most notably, the Hexagon DSP features both a 64-bit wide scalar unit, and its two 128-bit wide hexagon vector extensions (HVX) for vector processing.

Tools and SDKs for Heterogeneous Computing

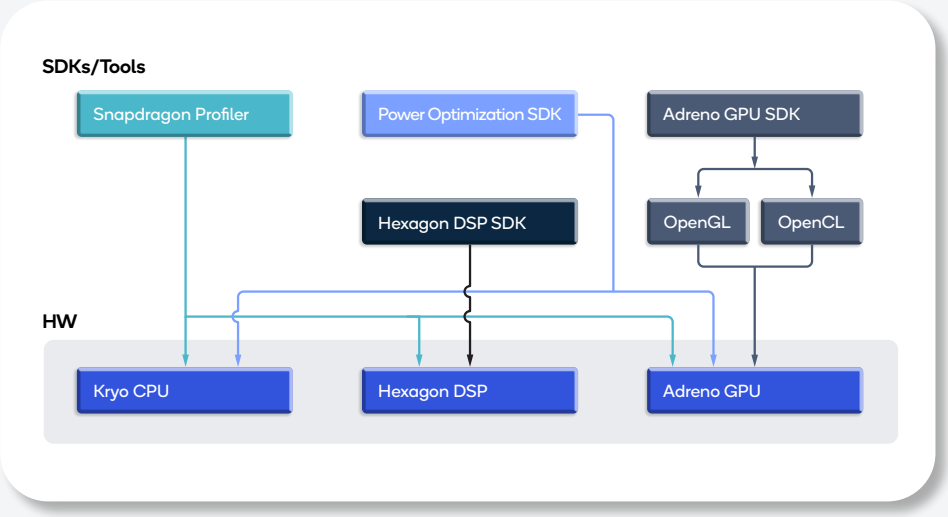
Developers can take advantage of a rich set of tools and SDKs for heterogeneous computing on Snapdragon:

Snapdragon Profiler: profiling software with real-time metrics of Snapdragon resources, including its CPU, GPU, and DSP.

Hexagon SDK: rich API for offloading tasks to the Hexagon DSP and coordinating access to resources. It exposes two general methods for accomplishing this: FastRPC for performing remote procedure calls and Dynamic Loading to load images onto the DSP at runtime.

Snapdragon Power Optimization SDK: API for controlling power consumption across the Kryo CPU and Adreno GPU. This includes static modes for balancing overall performance and power, and a dynamic mode for meeting power consumption targets at specifics points during application execution.

Adreno GPU SDK: includes tools, libraries, samples, documentation, and tutorials for graphics and games development on Adreno. This includes a Game Developer Guide, that provides guidance and best practice for heterogeneous computing across the Snapdragon's CPU, GPU, DSP.



OpenCL: open-source, cross-platform framework for executing code on heterogeneous platforms. Developers use this for running general compute tasks in parallel on the Adreno GPU. Qualcomm Technologies also provides additional OpenCL extensions for heterogeneous operations such as those related to ML.

OpenGL: an open-source, cross-platform API for rendering 2D and 3D graphics. Developers use this for hardware-accelerated rendering on Adreno. Qualcomm Technologies also provides additional OpenGL extensions for rendering operations, such as variable rate shading on Adreno.

Innovate together

Qualcomm Developer Network is a collection of software and hardware tools, inspiring our community of developers to push the boundaries of mobile. We're continuously creating some of the most innovative, powerful and disruptive technologies in the world, and Qualcomm Developer Network is the gateway through which you can discover the tools you need, whether you're building high-performance apps, smart Internet of Things (IoT) devices, immersive virtual reality experiences or for other emerging technologies.

developer.qualcomm.com

Qualcomm
developer network