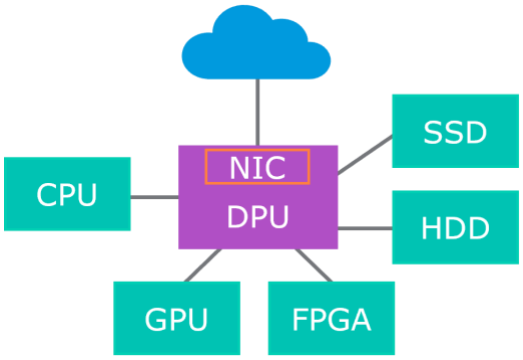
MIPS™
a GlobalFoundries company

The Benefits of Hardware Multi-Threading

Increase Silicon Efficiency for Higher Data Processing with the Same Area and Power

Introduction

The design of a processor is generally directed by its end application. There are multiple considerations for the processor requirements based around performance, cost, and power consumption that need to be taken into account when defining the processor architecture. For example, if you're designing something that's aimed at a wearable device, you have to minimize the cost (area), make it as low power as possible, and so on. If you're designing imaging equipment, you have other criteria that come to the forefront, like optimizing compute time, ensuring efficient memory access, and supporting multiple concurrent flows of data.



CPU = Central Processing Unit
GPU = Graphics Processing Unit
DPU = Data Processing Unit
NIC = Network Interface Controller
SSD = Solid State Drive
HDD = Hard Disk Drive

Figure 1: Datacenter Elements

Within the datacenter, the main functional elements include servers (CPU, GPU, and FPGA), storage (SSD and HDD), and networking (NIC, included in DPU). There are multiple processing devices within each of these, as well as the general infrastructure of power and cooling, with varied requirements for the capabilities of the embedded processor cores within them.

The Power of Multi-Threading for Hyperscale Data Centers

Within the server function, there are CPUs and GPUs for the main application processing, with high-performance and single- and multi-processor applications requirements. Storage devices include the hard drives and solid state drives needed to store information, with local processing to manage the host or network interface, local buffer management and DMA, and media control, security and encryption. Networking devices include the routers and switches enabling communication in between, with local processing to manage routing tables, firewall, encryption, quality-of-service, and more.

MIPS
a GlobalFoundries company
www.mips.com
2870 Zanker Rd #210 San Jose CA 95134

As the performance requirements for applications running in the datacenter continue to grow rapidly, and topologies expand to support disaggregated compute and storage resources available to many users, the bandwidth through the network and to/from memory and storage elements is often the bottleneck to application performance. To maximize throughput in these areas, the processing elements within networking and storage components utilize custom datapaths with specialized operations that keep data moving at the maximum permitted by the physical interfaces and storage medium, and couple this with embedded processors to handle the control and data movement. These embedded processors manage dataflow, with the amount of processing on the processor itself being relatively light, relative to each data access that it makes to and from local memory, which we refer to as data-centric processing.

Data-centric computing within the datacenter is naturally an area where the processing lends itself well to multi-threading hardware. There are multiple (many) concurrent run-to-completion tasks coordinating data passing through the subsystem, with relatively low compute cycles compared to the data accesses performed. Due to the nature of managing dataflow, the processor will often not be fully utilizing all its available compute resources, as its pipeline will stall as data is being accessed from memory. Additionally, as multiple concurrent data transfers flow through the network, there may be frequent handling of interrupts, which would also introduce pipeline stalls. These stalls are in addition to any natural pipeline stalls within the execute pipeline for data dependencies between operations.

For data-centric processing, hardware multi-threading allows for duplicating only a portion of the core needed to hold and manage the processor state for each independent thread while sharing the rest (e.g. the execution units, data access, interrupt logic, memory). This allows for the resources of the core, plus its local memory and cache subsystem, to be more efficiently used in such applications.

MIPS
a GlobalFoundries company
www.mips.com
2870 Zanker Rd #210 San Jose CA 95134

What is Simultaneous Multi-Threading

Let's first explain how the single-threaded architecture works. Very simply, a single-threaded CPU architecture executes one instruction-stream at a time, processing tasks sequentially, without parallel execution of multiple instruction streams or threads. In other words, a single execution flow is running through the processor core, and it can handle multiple streams of operation but only in a serial manner. Tasks are completed one at a time. An operating system can manage execution of multiple parallel software threads based on their respective priorities such that, over time, they run in parallel. However, at the cycle level, the CPU is handling them one at a time, and generally there is overhead to switch context from one software thread to another.

Figure 2: Multiple tasks, executing in series

Hardware multi-threading – as the name implies – provides the additional hardware mechanisms for multiple software threads to execute in parallel. There are different levels of hardware support possible for this but, fundamentally, a subset of CPU resources dealing with state are duplicated, along with thread identification of the executing operations, with operations from different threads in flight through the execution pipeline at the same time.

With more modern approaches to multi-threading, the hardware can alternate operations from different threads of execution on each processor cycle with no overhead, meaning every CPU cycle can have an operation executing.

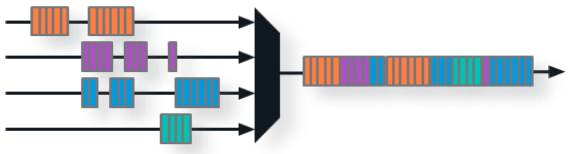


Figure 3: 4-way multi-thread CPU with single execute per cycle

MIPS
a GlobalFoundries company
www.mips.com
2870 Zanker Rd #210 San Jose CA 95134

Additionally, in a superscalar CPU architecture, where multiple concurrent operations are supported even for a single thread, the hardware can be executing instructions for two or more different threads even in the same cycle. This is referred to as simultaneous multi-threading (SMT).

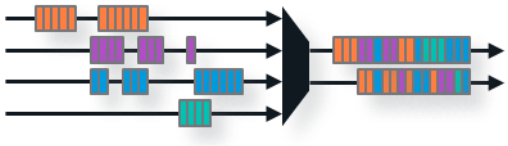


Figure 4: 4-way multi-thread CPU with 2 instructions executing simultaneously per cycle

With multi-threading supported directly in the processor hardware, each hardware thread is visible and looks like a complete core to the software operating system (OS). A symmetric multiprocessing (SMP) operating system can be run and natively leverage each of the hardware threads, or separate operating systems can run on each thread.

There Are Always Tradeoffs

Of course, there are always tradeoffs to be made. For example, two hardware threads running on a single multi-threaded core may not equal the performance of two single-threaded cores. But adding the support for an additional hardware thread (or more) to a single core can provide a significant performance boost with only a minor addition to power and area, especially in contrast to the cost of duplicating to add one (or more) complete single threaded core(s). So, when running multiple threads in parallel, the performance/Watt and performance/area efficiencies are much higher on a multi-threaded core than on multiple cores. In other words, multi-threading results in much higher compute density, as measured per unit area and per unit power. In an example case of a pointer-chasing application, where the processor is managing dataflow, the throughput performance of the pointer management for the same silicon area can be roughly doubled with multi-threading. Specific workload performance will vary depending on the processing required per data access as well as latencies to system

memory and companion hardware elements, which may increase or decrease the relative throughput.

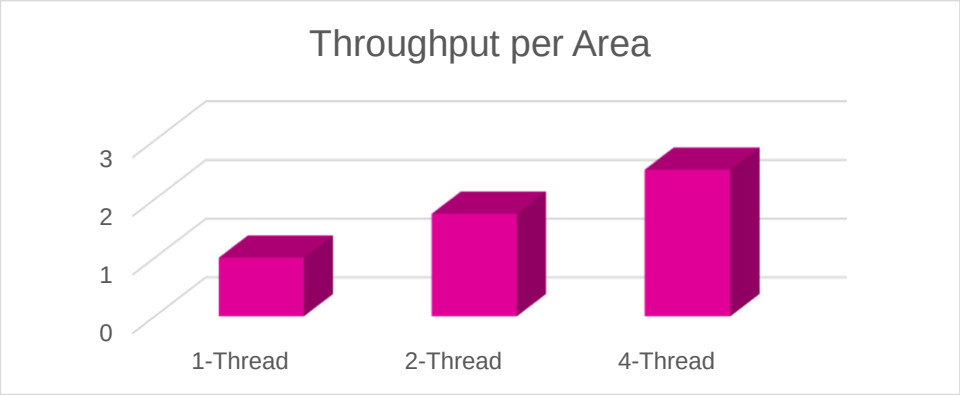


Figure 3: Relative performance for simultaneous multi-thread vs single thread

Of course, there is a point of diminishing returns where increasing the number of hardware threads supported on a single core does not translate into more performance or better efficiency. Hardware threads share physical resources within the core and, as such, performance will saturate in one core and, to further increase performance, another core needs to be added. Determining this design tradeoff point is very application specific, and also a function of how well the code is tuned to the CPU. For example, in applications that are more compute-intensive, such as image processing or handling neural nets directly in software, having more than one hardware thread may not result in increased performance.

Whereas if an application is data I/O intensive, with light processing per memory access, then multi-threading is more beneficial. Tasks may involve frequent I/O operations that require the core to wait for external resources to return data; as such, multi-threading keeps the CPU hardware resources busy working on other tasks. While one thread is waiting, another thread can continue execution, preventing idle time and improving overall throughput. For data-centric processing, multi-threading is much more efficient because the same resources can be utilized for parallel threads of execution by taking advantage of the stall/waiting periods within the processor, whether used to initiate additional memory and I/O accesses, performing processing on

incoming/outgoing data, or executing control tasks of managed resources in the system.

In the datacenter application spaces of storage and networking, where embedded processors are managing dataflow and control, much of the compute is handled in specialized hardware. In this use case, multi-threading can bring significant cost and power advantages in embedded cores, relative to burdening the specialized hardware with a need for instantiating additional cores. Overall throughput and compute requirements for the system can be handled with less overall processor resources.

There are other applications that map to the datacenter that would see comparable benefit. AI processing is a high-growth area that is putting significant strain on datacenter resources. Today, much of AI workloads are processed on CPU and GPU, though many companies are developing additional hardware to meet the demands of AI. Whether these will be included in a Data Processing Unit (DPU) or dedicated AI Processing unit remains to be seen, but the data sets for AI/ML processing are large and throughput requirements to "feed" the compute of these workloads tracks accordingly. As such, the notion of local processors managing dataflow through a specialized datapath applies for AI as well, and a multi-threaded architecture for a dataflow processor can provide better performance and overall compute density while providing programmability within specialized hardware for flexibility.

MIPS, RISC-V, and Multi-Threading

MIPS as a company has a rich history in microprocessor technology, including deployments into data center applications over several different generations. The company also has a long history with multi-threading and multi-core processor technology, having introduced the industry's first multi-threaded processor IP back in early 2006, and a coherent multi-core version utilizing those multi-threaded cores in 2008. The transition of the company's MIPS32/MIPS64 architecture to RISC-V is an evolutionary step. MIPS have recently introduced two new RISC-V cores, both of which support hardware multi-threading. The P8700 is an Out-of-Order core that offers 1 or 2 hardware threads. The I8500 is an In-Order core that supports configurations of 1, 2, and 4 hardware threads.

Both cores support coherent multi-core with up to 8 cores per cluster, which can be expanded to multi-cluster designs for higher core counts. The MIPS architecture has multiple options to closely integrate custom hardware logic along with the processor cores in a fully coherent cluster enabling deterministic, low-latency processing along with specialized processing elements. This means that all the cores, and accelerators, see the same data or have the same view of memory, so they can stay up to date with what the other cores may have in their local caches with limited software overhead. In addition, MIPS can support inter-thread communications across cores, allowing multi-threading efficiency in multi-core designs and not only within a single core.

Performance doesn't always scale linearly with the number of threads or the number of cores. As you dig deeper into CPU metrics, the use case, and associated software workloads, you'll see that there are other factors involved, including the kernel that's running on the CPU, and memory accesses that could invoke latencies and other system delays.

The operating system acts as the traffic cop in a multi-threaded system. The MIPS CPUs can run SMP Linux and other high-level operating systems (HLOS), as well as a number of real-time operating systems (RTOS), giving flexibility to the system architect to build up the software stack as needed for the application, and how to service events and respond in real time to manage the application flow. The RISC-V architecture allows for a broad ecosystem of software and tools that are directly supported on the MIPS CPUs.

Summary

With an understanding of multi-threading, you can see where to apply the technology and, in particular, why it is very beneficial in data-centric applications where the CPU may not be fully utilized each cycle for a single thread of execution. In such cases, hardware multi-threading can more than double the silicon efficiency of the processor solution, thereby doubling the processing (e.g. data throughput) for the same silicon area and power consumption. In addition to the data center networking and storage that we've already discussed, hardware multi-threading also finds a home in:

- AI and machine learning
- Multimedia and graphics processing
- Web and application servers
- Database systems
- Simulation and modeling
- Game and content development
- Batch processing and background tasks

More information is available on https://www.mips.com.