Optimizing Performance, Power, and Area in SoC Designs Using MIPS® Multi-threaded Processors

Hardware-based multi-threading technology has for some time been known in the industry as a feasible technique for improving system performance, but not too many people are aware of just how much traction the technology has gained since its early implementations in the 1960s. This article provides a brief history of hardware based multi-threading and some examples of its commercial adoption so far. It then gives an overview of the fundamental value of multi-threading in hardware, and describes MIPS Technologies' multi-threading architecture and product offerings. The article also provides several multi-threaded application examples—including those in the areas of driver assistance systems and home gateways—to demonstrate the broad applicability of multi-threading in real-world applications.

Document Number: MD00828
Revision 01.04
September 21, 2011

MIPS Technologies, Inc.
955 East Arques Avenue
Sunnyvale, CA 94085-4521

1 Hardware Multi-threading Background

Multi-threading is a hardware- or software-based processing technique which has the primary objective of exploiting the concurrency in a computational workload to increase performance. Multi-threading can also be used to isolate various tasks so that priority can be assigned to more time-sensitive traffic such as voice, video or critical data.

While software-based multi-threading techniques such as task switching and software-based thread scheduling are recognized to have been in existence for some time, less is known about the history of hardware based multi-threading. Hardware-based multi-threading techniques have in fact existed for quite some time, with implementations dating back to the 1960s with the CDC6600 [1]. In the CDC6600 computer, 10 threads in hardware were used to guarantee response time from the I/O processor to the approximately 16 peripherals. This example, where the processor ran much faster than the array of I/O devices, is a typical application which benefits greatly from multi-threading, as the idle processing time could be replaced with useful work in switching from thread to thread. In the 1970s, the Denelcor HEP machine [2] switched on real execution threads within the CPU rather than on I/O. Similar to the previous example, the net result was that instructions per cycle (IPC) were improved dramatically. Several other systems and academic studies were introduced over the next decade that further demonstrated the benefits of hardware multi-threading.

Today there is an array of multi-threaded processors in the market. Intel has brought a coarse grained version of the technology into the high-end computing application space with its Hyper-Threading technology. Furthermore, numerous other SoC manufacturers such as Broadcom, Lantiq, Mobileye, Netlogic Microsystems, PMC-Sierra, Ralink Technology and Sigma Designs have also shipped millions of products with multi-threaded CPUs. Many such SoCs are based on the multi-threaded MIPS32® 34K® core or the multi-threaded, multiprocessing MIPS32 1004K™ Coherent Processing System (CPS) based on the industry-standard MIPS® architecture. Today, hardware multi-threading has reached mainstream adoption and is increasingly recognized as an efficient method for extracting optimal performance in SoC designs.

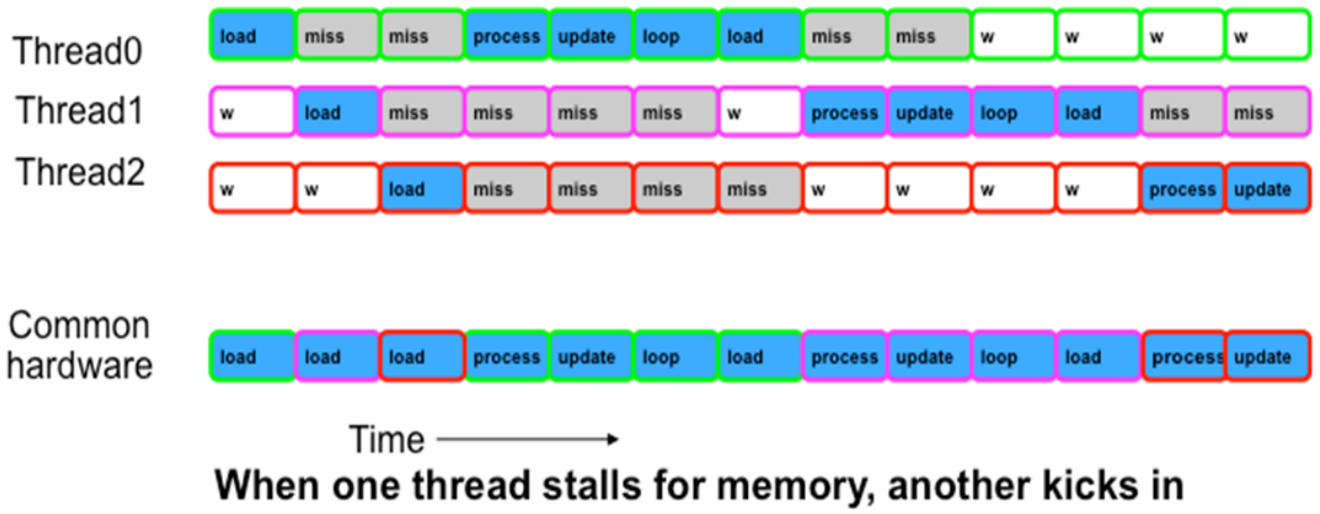
2 Increasing Pipeline Utilization with Multi-threading

As mentioned earlier, increasing the IPC number for a given single-threaded processor is a major objective. Often, even a very high performance processor spends much of its time idle, waiting for data to arrive. It is not uncommon for such advanced processors with a shared memory system to spend over 50% of their time waiting for data to return after a cache miss. This data retrieval wait time could last anywhere from tens of cycles to perhaps even hundreds of cycles in extreme cases. Whatever the number is, the processor effectively does no useful work during most of this time. Other instruction dependencies such as branch-mispredict, load-to-use, and others can also cause idle cycles. Multi-threaded processors are able to switch between multiple threads to make use of these idle cycles. Rather than letting unused cycles go to waste, cycles can now be filled with useful instructions from other threads. This leads to better pipeline utilization and increased system throughput.

One key aspect which ensures efficiency of thread switching has to do with the management of thread-related information, or contexts. Each task, when mapped to a thread, also has associated context information such as program counters and a subset of register information, which are loaded and updated in hardware. In a single-threaded processor, these contexts must be swapped in or out as the processor switches between threads. Additional processing associated with ‘save’ and ‘restore’ operations are involved when changing between threads. This tax can become very burdensome, especially with high context switching. With multi-threaded processors supporting full context storage for each hardware-supported thread, there is no need for ‘save’ and ‘restore’ operations. This mechanism supports zero-cycle overhead for switching threads or contexts.

Figure 1 shows the basic mechanism of how multi-threading improves pipeline utilization. In this case, there are three threads which experience cache misses at different points in time. During these stalls, other threads are doing useful work over the same pipeline, thereby increasing IPC.

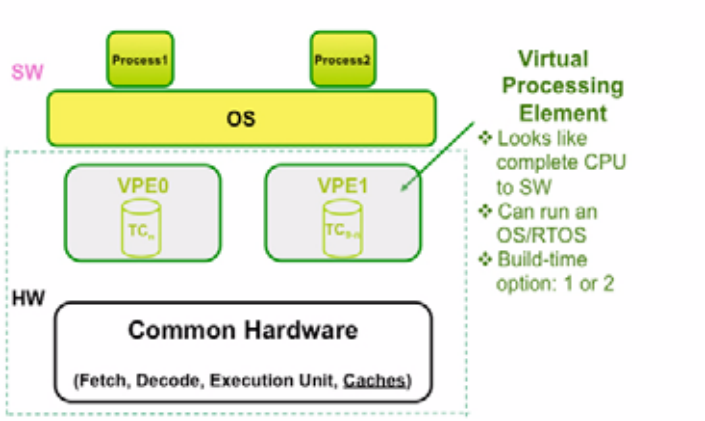
Figure 1 Multi-threading



3 Overview of MIPS Multi-threaded Technology

MIPS Technologies' multi-threaded technology is based on a two-layered framework involving Virtual Processing Elements (VPEs) and Thread Contexts (TCs), and it supports thread switching on every cycle. Each multi-threaded core can support up to two VPEs which share a single pipeline among other hardware resources. However, since each VPE includes a complete copy of the processor state as seen by the software system, each VPE appears as a complete standalone processor to an SMP Linux operating system. For more fine-grained thread processing applications, each VPE is capable of supporting multiple TCs. The TCs share a common execution unit but each has its own program counter and core register files so that each can handle a thread from the software. The 34K core can support up to nine TCs allocated across two VPEs, optimized and partitioned at run time. Figure 2 shows the relationship of the OS, VPEs, TCs, and the common hardware in the 34K core.

Figure 2 34K® Top-level Architecture

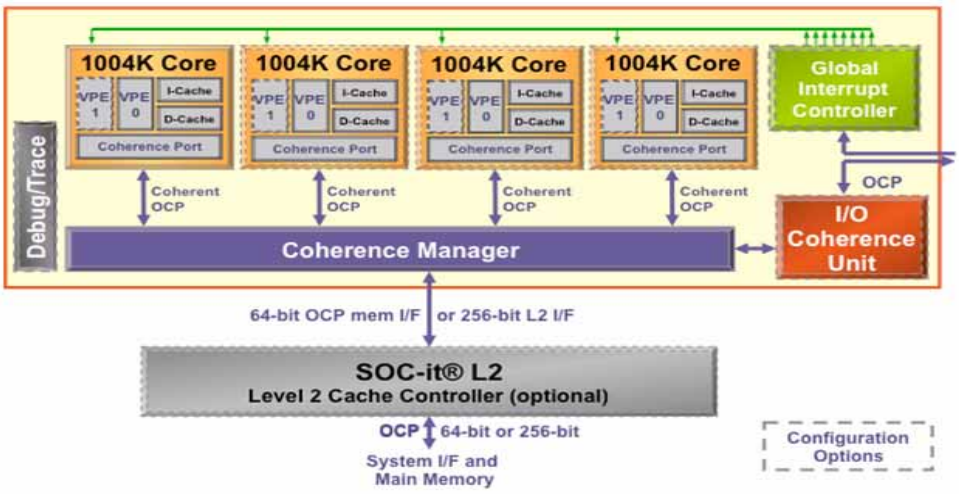


The 34K core also allows allocation of processor cycles to threads, and sets the relative thread priorities with an optional Quality of Service (QoS) manager block. This enables two prioritization mechanisms that determine the flow of information across the bus. The first mechanism allows the user to prioritize one thread over another. The second mechanism is used to allocate a specified ratio of the cycles to specific threads over time. The combined use of both mechanisms allows effective allocation of bandwidth to the set of threads, and better control of latencies. In real-time systems, system-level determinism is very critical, and the QoS block facilitates improvement of the predictability of a system. Hardware designers of advanced systems may replace the standard QoS block provided by MIPS Technologies with one that is specifically tuned for their application.

The 1004K CPS is the latest generation of multi-threaded processors from MIPS Technologies. The system supports up to four multi-threaded cores, each of which can be configured to support 2 VPEs like the 34K core. The multiple cores in the 1004K CPS are connected via a coherence management unit to maintain coherency between the L1 caches in each CPU. The system also includes an optional block to provide coherency on data transfers from I/O peripherals, enabling additional performance by off-loading I/O coherency schemes typically run in software as part of the operating system. The coherent processing system also includes a global interrupt controller that accepts up to 256 interrupts and distributes them down to the cores, or even hardware threads within each core. The whole system can be used with the MIPS L2 cache controller, which connects to the coherence management unit via an extended 256-bit wide interface for optimized throughput between the coherent system and the L2 cache. An EJTAG and a "coherence-aware" program/data trace block rounds out the system, providing synchronized visibility into each of the CPU cores and the coherency units in the system via development tools.

SMP Linux manages the interface to the VPEs, leaving the high level APIs untouched. This allows the leveraging of existing legacy core unchanged on the new core. In addition, some SMP versions of RTOS's such as ThreadX include support for fine-grained, TC-based multi-threading. It should be noted that the multithreaded and multicore hardware capabilities in the 1004K CPS leverage a common software programming model. That is, for example, someone developing a threaded application in SMP Linux can effectively run that on a 34K core or a 1004K CPS without modification, and the SMP kernel can make task affinity and load-balancing decisions on the workload across threads and cores.

Figure 3 1004K™ Top-level Architecture



4 Multi-threading Application Examples

There are several factors that a designer must consider when deciding whether a multi-threading system is appropriate for a specific application. First, a designer must consider the software and the desired feature set, including the different tasks, the ways these tasks might be partitioned, performance needs for each task, bus interaction for the different tasks, the special needs for tasks such as multimedia, and other considerations. Other key considerations include the level of coordination among the tasks, security, and of course power consumption and cost.

Needless to say, the analysis is often not straightforward, so a good understanding of priorities is critical. For example, if the goal is achieving the highest IPC in the lowest cost footprint, a designer may create the multi-threaded system to utilize the different threads in a single core and run at the highest frequency. In another design, power may be a key priority so a designer might choose to spread tasks over several threads over multiple cores, and scale down the frequency to reduce power consumption. The key to a MIPS-Based multi-threaded system is scalability, enabling designers to achieve the highest possible performance when needed, or to scale back when performance isn't the highest priority.

In another instance, a system may need to run 2 operating systems and meet very stringent QoS and costs constraints. The designer can run the software utilizing two VPEs, where one is running an RTOS and the other running bare iron while fully isolating QoS-sensitive applications such as voice or video. This can all be achieved without having to implement a second core. The following examples of use models that have been deployed in MIPS-Based multi-threaded SoCs for automotive and networking demonstrate the breadth of applications that can benefit from multi-threading.

4.1 Case Study – Driver Assistance Systems in Automotive

The first example of a real-world deployment of multi-threaded system design is from Mobileye, the leader in vision-based Driver Assistance Systems (DAS). Advancements in image and video analytics in terms of performance and cost have allowed these technologies to make their way into consumer markets. However, making continuous improvements on such technology is a very daunting task. Mobileye used the multi-threaded features of the MIPS32 34Kf processor to dramatically improve the performance and efficiency of its EyeQ2 vision system (based originally

on a competitor's single-threaded core) which takes data from a camera and looks for elements within the image to generate warnings for lane-departures, forward-collisions, vision/radar fusion, and pedestrian detection for example.

The EyeQ2 SoC system has an array of Vision Computer Engines (VCEs) which are connected as peripherals to the main CPU bus and provide and receive real-time data. In this system, large amounts of data and instructions are transferred from the CPU to the VCEs. This, along with the typical instruction cache misses, leads to a very challenging bottleneck which is not addressable by single-threaded CPUs. In fact, Mobileye's original EyeQ1 system had an IPC of only 0.3. Given the performance requirements, utilizing a single threaded architecture for EyeQ2 would have meant either slowing down the whole system or having to skip the processing of data, potentially causing errors. Increasing the CPU clock speed only exacerbates the problem, as this just increases the frequency of processor stalls. Furthermore, adding additional cores increases the number of bus contentions, thus impairing the real-time bandwidth.

Through simulation, Mobileye concluded that a multi-threaded system would provide a boost in performance by managing multiple operations across the entire system which included 8 VCEs. One 34Kf core with 4 threads was used to interact with the 8 VCEs. Furthermore, a QoS manager played an important role in tuning and prioritizing critical threads. In effect, this architecture helped in increasing the IPC by 3x from 0.3 to 0.9. Usage of the QoS manager was critical in taking the IPC from 0.6 to 0.9. Furthermore, with such a high IPC, increasing the CPU clock led to meaningful improvements. In fact, because of its use of multi-threaded and other architectural improvements, Mobileye was able to achieve its performance goals with just a modest increase in frequency from 110MHz to 166MHz while maintaining its power consumption at 3W, which was the same as the prior generation's power consumption. In addition, a second 34Kf core was instantiated to enable support for a user's proprietary algorithm. Communication between threads across the two processors was achieved by making modifications to an inter-thread communication block, which facilitated a more coherent system. For those interested in more details, a focused case study on the development of the EyeQ2 SoC is available in the *EE Times* [3]. For its future EyeQ3 system, even higher performance and lower cost will be required. Mobileye will be implementing a 4-CPU 1004Kf coherent processing system to ensure that the design meets these goals.

These features, along with standard codec, ADC, DAC, mixer/volume control and amplifier functions, form a family of completely "integrate-able" analog audio IP. Designers can select the features they need for each application, keeping the area needed to implement these features to a minimum.

4.2 Case Study – Broadband CPE / Residential Gateway

The residential gateway is another system that continues to experience increasing integration. Customer premises equipment for broadband have evolved to being more than just modem devices providing basic access to broadband services like DSL, Cable or PON. They have become integrated Residential Gateways or Integrated Access Devices which include features such as routing/switching, twisted-pair/co-axial/powerline/WiFi networking capabilities, security, USB and VoIP support, along with basic modem functionality. Some architectures also include integrated storage functionality in which the gateway acts as a media server. This integration has made gateway system design very challenging, and multi-threading can improve performance and/or improve the overall architecture for these devices.

One company that has successfully deployed a multi-threaded MIPS-Based SoC for gateways is Ralink Technology, a global technology leader in the wireless home networking and broadband access semiconductor markets. ADSL IAD is Ralink's first chip based on the 34K core. The chip is currently in mass production and has been deployed successfully in the network of a European telecommunications operator. Multi-threading has proven to be very appropriate for this triple-play system, as it helps to run multiple applications more efficiently. Furthermore, the isolation of time-sensitive traffic via the use of Virtual Processing Elements (VPEs) is key for ensuring deterministic response. The SoC implements a single 34K core and four threads over two VPEs. The first VPE, VPE0, implements three threads which support WLAN, Ethernet and USB processing, while the second VPE, VPE1, implements one thread

supporting ATM and VoIP. With this architecture, Ralink is able to efficiently provide deterministic VoIP response and support for multiple applications in a single device.

5 The Future of MIPS Multi-threading: Simultaneous Multi-threading

Today, MIPS' multi-threading is based around temporal—also known as interleaved—multi-threading technology, in which a maximum of one instruction can be issued into the execution pipeline in any given cycle. As we have discussed in this paper, temporal multi-threading offers tremendous benefits compared to single-threaded architectures. The next enhancement in multi-threading for MIPS processors will be Simultaneous Multi-threading (SMT), in which multiple instructions from different threads can be issued into the execution pipeline within the same clock cycle.

To better understand the advantage of SMT, let us first consider a non-threaded processor with a single execution pipeline. To try to increase performance in this case, a designer might start by simply increasing the number of execution pipelines to two, for example, in order to execute more instructions at the same time. However, due to the dependencies of the instructions and likely cache memory misses, the performance increase will be non linear. In other words, while there is an effective doubling of hardware execution resources and a desire for a similar increase in performance, the result is more typically on the order of only a 1.3X improvement.

Therefore, to augment performance, other techniques such as "out-of-order execution" or "speculative execution" (e.g. pre-fetching and branch prediction) can be employed in conjunction with the multiple execution pipelines to try to increase the IPC to be closer to 2X. Unfortunately, such techniques usually come with a sizable area/cost/power penalty. Hence these techniques are commonly reserved for designs where the highest performance is critical, and area/power are secondary.

In contrast, hardware multithreading provides a different means to maximize execution efficiency. It does so by having alternative workloads, rather than applying large amounts of logic and buffers to eke out IPC improvements via manipulating instruction ordering and predictive execution. SMT, because it is able to efficiently intersperse instructions from concurrent, independent threads, can better fill multiple execution unit pipelines. In this example with two pipelines, the net effect is that performance can get closer to the ideal 2X level in a more area- and power-efficient manner than alternate techniques.

SMT technology will be introduced in a future generation of MIPS Technologies' family of multi-threaded processors. Through SMT, multiple instructions from different threads can be executed concurrently over multiple execution pipelines. Depending on the nature of the application, significant further improvement can be seen in the IPC due to SMT.

SMT offers a significant performance advantage to Linux-based processors such as those used for applications processing in consumer electronics and mobile devices. Standard SMP Linux can map tasks into the virtual processors, thereby enabling the hardware to take advantage of the parallelism available in the multi-tasking operating system.

SMT offers particular benefits for applications including those in the Networking and Storage segments. These types of applications involve highly concurrent processes that could be handled very efficiently by SMT. With the addition of SMT, one can expect increased performance when doing common networking data plane processing functions like route lookups, NAT, and TCP reassembly. In Storage, many applications today use MIPS Technologies' multi-threading technology to increase efficiency of read/write operations, by up to 3X in some instances. This can be further increased with SMT.

In MIPS' processors with SMT, MIPS will also offer 64-bit processing, which will also be key as users now have a licensable core that allows them to take advantage of the increased virtual addressing (VA) and physical address (PA) space and more efficient data movement that comes with a 64-bit processor. Memory size requirements in networking applications continue to grow, and more and more SoC vendors are asking for a 64-bit core to integrate in next generation SoCs including those for networking. A 64-bit core gives users the ability to extend beyond the 4GB limits of 32-bit processors, to many terabytes and beyond. And storage applications, with their persistent movement of data in and out of memory, will benefit from the ability to move bigger chunks of data each clock cycle.

6 Summary

Hardware multi-threading is a technology that has reached the mainstream, and there exist today many SoC suppliers who provide MIPS-based multi-threaded SoCs for a variety of applications. We have focused specifically on applications in networking and automotive, but multi-threading can be used in any system which has many concurrent tasks, or in any system where QoS is critical. Other vendors have implemented MIPS-Based multi-threading in SoCs for set top boxes, high-end storage and networking infrastructure equipment. Some evaluation has also begun for the use of multi-threading in mobile applications such as smartphones and tablets. These devices continue to grow in complexity as integrated communications, productivity and multimedia entertainment devices. Multi-threading would be an ideal solution for efficiently processing multiple mobile applications, some of which are latency-sensitive.

Alternate "brute force" approaches of using multiple cores to process threads simultaneously can certainly be employed, but using multi-threading is by far a more elegant and lower-cost approach to achieving IPC and/or QoS performance requirements. By offering single-core and multi-core products which support hardware multi-threading today and with the next generation 64-bit processors with SMT support on the horizon, MIPS Technologies is uniquely positioned to enable designers to develop high-performance and low-power SoCs in a very cost efficient manner.

Delfin Rodillas is a Marketing Director for the Networks Markets at MIPS Technologies. He can be reached by email at delfin@mips.com.

7 References

[1] J.E. Thornton, "Parallel Operation in the Control Data 6600", Proceedings-Spring Joint Computer Conference, 1964

[2] B.J. Smith, "Architecture and applications of the HEP multiprocessor computer system", In SPIE, volume 298, pages 241-248, 1981

[3] R. Elchanan and P. Del Vecchio, "Multi-threaded design tackles SoC performance bottlenecks: Part 1," EE Times, http://www.eetimes.com/design/automotive-design/4011117/Multi-threaded-design-tackles-SoC-performance-bottlenecks-Part-1, 2006

Template: nW1.03, Built with tags: 2B