Multi-core Processors

1. Aniket V. Kulkarni, 2. Omkar S. Nathi
1,2. B.E E&TC, Pune University, Pune, India

***Abstract*— Our paper includes the overview of single core processor, multi core processors and alternate suggestions to increase the number of cores along with the efficiency of the device. The speed of device can be improved by increasing number of cores, code optimisation, power optimisation and bus optimisation. If the application actually needs the high speed processing then multiple cores of processors can be used.**

I. INTRODUCTION

A multi-core processor is a single computing component with two or more independent actual central processing units (called "cores"), which are the units that read and execute program instructions. The instructions are ordinary CPU instructions such as add, move data, and branch, but the multiple cores can run multiple instructions at the same time, increasing overall speed for programs amenable to parallel computing. Manufacturers typically integrate the cores onto a single integrated circuit die (known as a chip multiprocessor or CMP), or onto multiple dies in a single chip package.

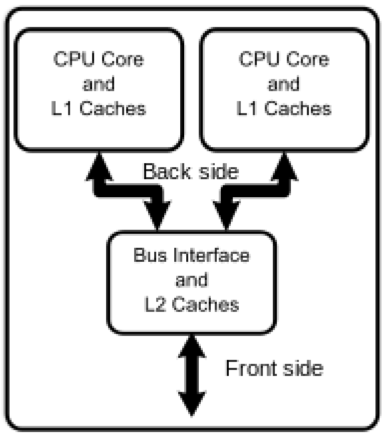

fig. 1 architecture of multi-core processor

Processors were originally developed with only one core. A dual-core processor has two cores (e.g. AMD Phenom II X2, Intel Core Duo), a quad-core processor contains four cores (e.g. AMD Phenom II X4, Intel's quad-core processors, see i5, and i7 at Intel Core), a 6-core processor contains six cores (e.g. AMD Phenom II X6, Intel Core i7 Extreme Edition 980X), an 8-core processor contains eight cores (e.g. Intel Xeon E7-2820, AMD FX-8350), a 10-core processor contains ten cores (e.g. Intel Xeon E7-2850), a 12-core processor contains twelve cores. A multi-core processor implements multiprocessing in a single physical package. Designers may couple cores in a multi-core device tightly or loosely. For example, cores may or may not share caches, and they may implement message passing or shared memory inter-core communication methods. Common network topologies to interconnect cores include bus, ring, two-dimensional mesh, and crossbar. Homogeneous multi-core systems include only identical core, heterogeneous multi-core systems have cores that are not identical. Just as with single-processor systems, cores in multi-core systems may implement architectures such as superscalar, VLIW, vector processing, SIMD, or multithreading.

The improvement in performance gained by the use of a multi-core processor depends very much on the software algorithms used and their implementation. In particular, possible gains are limited by the fraction of the software that can be run in parallel simultaneously on multiple cores; this effect is described by Amdahl's law. In the best case, so-called embarrassingly parallel problems may realize speedup factors near the number of cores, or even more if the problem is split up enough to fit within each core's cache(s), avoiding use of much slower main system memory. Most applications, however, are not accelerated so much unless programmers invest a prohibitive amount of effort in re-factoring the whole problem.

II. THE NEED FOR MULTIPROCESSING

Mobile devices perform a wide variety of tasks such as Web browsing, video playback, mobile gaming, SMS text messaging, and location-based services. Due to the growth in the availability of high speed mobile and Wi-Fi networks, mobile devices will also be used for various performance-intensive tasks that were previously handled by traditional PCs. The next generation of smart-phones (called “Super phones”) and tablets will be used for a wide variety of tasks such as playback of high definition 1080p videos, Adobe Flash-based online gaming, Flash-based streaming high definition videos, visually rich gaming, video editing, simultaneous HD video downloads, encode and uploads, and real-time HD video conferencing.

The current generation of mobile processors is not designed to deal with this tidal wave of high performance use cases. The quality of experience on devices based on single core CPUs rapidly degrades when users run several applications concurrently, or run performance intensive applications such as games, video conferencing, video editing, and more. In order to improve CPU performance, engineers employ several techniques, such as using faster and smaller semiconductor processes, increasing core operating frequency and voltage, using larger cores, and using larger on-die caches. Increasing the size of the CPU core or cache delivers performance increases only up to a certain level, beyond which thermal and heat dissipation issues make any further increase in core and cache size impractical.

From basic semiconductor physics we know that increasing operating frequency and voltage can exponentially increase power consumption of semiconductor devices. Even though engineers may be able to squeeze out higher performance by increasing frequency and voltage, the performance increase would drastically reduce battery life. In addition, processors that consume higher power would require larger cooling solutions resulting in an undesired expansion in device size. Therefore, increasing the operating frequency of the processor to meet the ever-increasing performance requirements of mobile applications is not a viable solution for the long run. To satisfy the rapidly growing demand for performance and form factor sleekness in mobile devices, the industry has begun to adopt newer technologies such as Symmetrical Multiprocessing and Heterogeneous Multi-core computing.

III. Symmetrical Multiprocessing (SMP)

Symmetrical Multiprocessing technology enables mobile processors to not only deliver higher performance, but also meet peak performance demands while staying within mobile power budgets. A multi-core architecture with SMP is defined by the following characteristics:

- Architecture consists of two or more identical CPU cores.
- All cores share a common system memory and are controlled by a single Operating system.
- Each CPU is capable of operating independently on different workloads and whenever possible, is also capable of sharing workloads with the other CPU.

Imagine a mobile phone that has a dual core CPU with SMP support-- if the phone’s navigation application is running concurrently with a streaming audio application, the OS can assign the navigation task to one CPU core and the streaming audio task to the second CPU.

Another example is a single multi-threaded application that can benefit from multiple CPUs. The OS can assign the threads to run on both CPUs concurrently and finish the task faster by sharing the workload across the two CPUs. Since the workload is split across the two cores, these cores can run at a reduced speed while achieving excellent performance and also conserving power (running at lower frequency lessens the voltage required, resulting in a reduction in power by the square of voltage decrease). It contains several ActiveX and JavaScript-based menu options, and includes two embedded Flash videos and Flash animation. A single core CPU would have to bear the entire load of processing the ActiveX/JavaScript content, Flash player processing, and the embedded video decode tasks. In addition, the processor will have to handle background tasks such as Twitter streams, voice applications, navigation, and others. Under these heavy multitasking conditions the processor is typically running at maximum frequency and voltage, therefore consuming large amounts of power.

IV. Processors in Mobiles

This year the first octa-core smart phone will hit the market, but as the number of cores inside mobiles and tablets grow, so does the toll on the battery. The problem with cramming more cores on to processors is that energy efficiency doesn't scale with the number of cores stacked onto chips. As more cores are added, power consumption grows faster than performance.

Tackling the rapacious appetite for power of many-core processors is relevant to more than just letting computers do more on the move. As an increasing number of cloud services such as Gmail,Salesforce.com and Spotify are accessed over the internet the need to keep down the energy demands of data centres full of densely packed server clusters is also becoming pressing.

If unaddressed, the rising power consumption of many-core processors may limit future increases in computing performance, with predictions that within three processor generations CPUs will need to be designed to use as little of 50 percent of their circuitry at one time, to limit energy draw and prevent waste heat from destroying the chip.

Worrying about how to limit the draw of processors with hundreds of cores might sound academic, something that won't be an issue for mainstream processors for more than a decade. But many-core processors don't seem quite so distant when you consider there are already octa-core processors in desktops and servers – as well as specialist many-core processors such as the Xeon Phi Co-processor— and Moore's Law predicts there will be processors with more than 16 times the transistors of today's chips by 2020.

The International Technology Roadmap for Semiconductors 2011, a roadmap forecasting semiconductor development drawn up by experts from semiconductor companies worldwide, forecasts that by 2015 there will be an electronics product with nearly 450 processing cores, rising to nearly1,500 cores by 2020.

V. New approach for multi-core processors

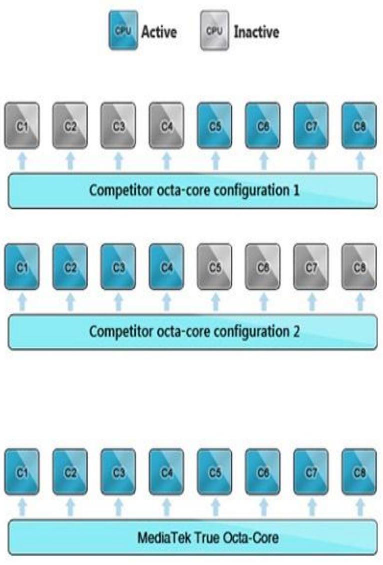
Active
Inactive
Competitor octa-core configuration 1
Competitor octa-core configuration 2
MediaTek True Octa-Core
C1
C2
C3
C4
C5
C6
C7
C8

fig. 2 Comparison between true octa core and partial octa core processor

The dynamic power management would require current computer hardware, operating systems and applications to be enhanced. Additional circuitry, such as performance and energy counters, would need to be added to processors to capture more data on how much work a CPU was doing and how workloads were distributed across cores. This data could include the level of current being consumed and the operating frequency of each core. This data would then be interrogated by the operating system to capture a snapshot of the load on the processor and how the CPU was handling it. Interrogating this data in detail would require changes to power management routines within the kernels of operating systems.

Lastly, each application would likely need to come with a profile that described its power and performance needs to the power management system in the OS kernel. All these changes would allow an OS to constantly monitor the performance and power usage of a CPU, scaling its power usage to precisely match the needs of the app outlined in its profile. This adjustment would take place via methods such as reducing the clock speed and voltage flowing to the CPU and shutting down parts of cores. More sophisticated manipulation of processors — for example, switching between heterogeneous cores, a la Arm's big .LITTLE, and homogeneous cores — would require further modification of processor hardware.

Bus Utilization: Do not interleave variable reads and writes. Group them and separate them with some computational instructions

Cache: Define or align variables on 64 and 128 byte boundaries. Use local variables. When defining variables, define them grouped as they are used in the code, group variable into arrays. When using large variables define them in 4K byte pages.

Branch Prediction: Separate Branch Instructions. All that is needed is 3 μop instructions between branches. When Branch Instructions are contiguous, branch prediction is defeated.

Instruction Execution Core: The newer Intel execution engines are capable of executing six simultaneous instructions. By using bitwise operators the compiler will use less complex single μop instructions. The programmer can assist the compiler by using functions that give the compiler a clue as how to use the optimum instructions.

x = x/2 the compiler, if multi-pass, may use a shift instruction
x >> 2 leaves no doubt
Loop Optimization: More than 64 loops. No more than 4 taken branches. It's not the number of branches, but the number of branches that actually occur.

VI. Conclusions

I think if we are going to keep adding more cores, there may actually be a push to make CPUs simpler in order to fit more of them on a die. GPUs have already demonstrated that a simple enough execution units can be replicated hundreds, even thousands, of times on a single die. But I suspect CPUs never made the jump because they're more complex.

We're also approaching the point where scaling up doesn't make much sense from a consumer standpoint - very, very little consumer software can take full advantage of multiple cores. Your average PC user probably has 2 or 4 cores. Cell phones are up to four cores as well, but whether it really helps a mobile device is questionable, and it does drain more battery.

The question may be - what's the consumer app that makes people want to have more than 2 to 4 cores? Other than games, there's not much out there that consumers want that would really push multi-core to its limits. Now for the real twist in the story: APUs. Your average GPU has to deal with the PCIE bus, but the upcoming APUs have no such limitation, and I'm sure that before long we'll be seeing the GPU side of APUs have 1000+ cores.

But many-core processors don't seem quite so distant when you consider there are already octa-core processors in desktops and servers – as well as specialist many-core processors such as the Xeon Phi Co-processor — and Moore's Law predicts there will be processors with more than 16 times the transistors of today's chips by2020."

Moore's "law" has pretty much ended from a "we're gonna scale up GHz and/or # of cores" standpoint. Maybe in servers octa-core is common, but it's difficult to find a consumer with more than two cores. You still need to find an enthusiast or gamer to find quad or octa-core, and honestly I don't wonder if we shouldn't have 16 or 32 cores by now for Moore's law to hold true

REFERENCES

[1] www.zdnet.com
[2] www.qualcomm.com
[3] www.gsmarena.com
[4] www.mediatek.com
[5] Multicore Processors and systems by Springers
[6] Tanenbaum

Appendix

number of cores	Common names
1	single-core
2	dual-core
3	tri-core triple-core
4	quad-core
5	penta-core
6	hexa-core
7	hepta-core
8	octa-core, octo-core
9	nona-core
10	deca-core
11	hendeca-core
12	dodeca-core
13	trideca-core
14	tetradeca-core
15	pentadeca-core
16	hexadeca-core
17	heptadeca-core
18	octadeca-core
19	enneadeca-core
20	icosa-core