Multiprocessing in Mobile Platforms: the Marketing and the Reality

Marco Cornero, Andreas Anyuru
ST-Ericsson

Introduction

During the last few years the mobile platforms industry has aggressively introduced multiprocessing in response to the ever increasing performance demanded by the market.
Given the highly competitive nature of the mobile industry, multiprocessing has been adopted as a marketing tool. This highly visible, differentiating feature is used to pass the oversimplified message that more processors mean better platforms and performance. The reality is much more complex, because several interdependent factors are at play in determining the efficiency of multiprocessing platforms (such as the effective impact on software performance, chip frequency, area and power consumption) with the number of processors impacting only partially the overall balance, and with contrasting effects linked to the underlying silicon technologies.

In this article we illustrate and compare the main technological options available in multiprocessing for mobile platforms, highlighting the synergies between multiprocessing and the disruptive FD-SOI silicon technology used in the upcoming ST-Ericsson products. We will show that compared to personal computers (PCs), the performance of single-processors in mobile platforms is still growing and how, from a software performance perspective, it is more profitable today to focus on faster dual-core rather than slower quad-core processors.

We will also demonstrate how FD-SOI benefits dual-core processors even more by providing higher frequency for the same power consumption, together with a wider range of energy efficient operating modes. All of this results in a simpler, cheaper and more effective solution than competing homogeneous and heterogeneous quad-processors.

Multiprocessing is a necessity, not a choice!

It is important to recall that multiprocessors are a necessity, not a choice. Historically they have been introduced in mainstream products such as PCs, once frequency scaling reached the chips' heat dissipation limits - not before. Indeed since the beginning of silicon integration up until 2003-2005, the scaling of frequency and of the number of transistors have been used in mainstream computing mostly for single-processor evolutions (illustrated in Figure 1). During this period, applications performance simply scaled at the same rate as the hardware evolution, without the need to make any change to the growing software legacy, producing the phenomenal growth in computing that we have enjoyed in the past.

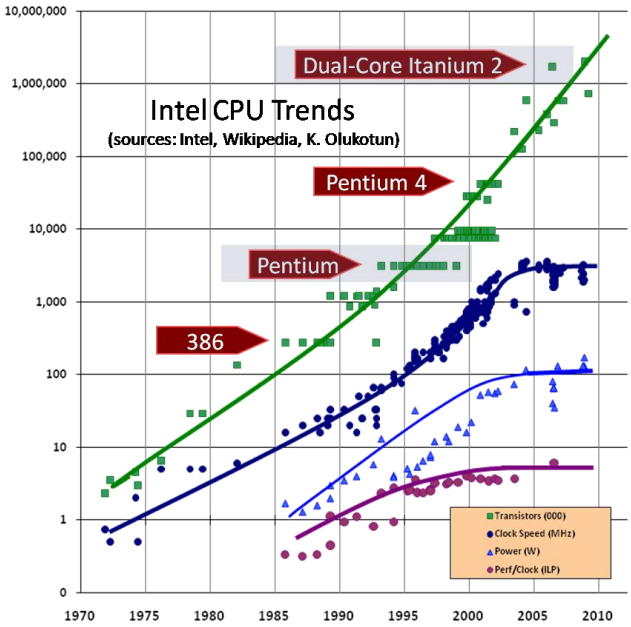

Figure 1: Intel CPU Trends

This trend continued for as long as it was possible – until reaching the heat dissipation limits of the chip, due in large part to the growing frequencies, not to the number of transistors. At this point multiprocessing was adopted as the only way to keep exploiting the still growing number of transistors, while maintaining power consumption within manageable limits. Indeed, although counterintuitive, one microprocessor at a given frequency consumes more power than two cores at half the frequency. This is due to the fact that a higher voltage is needed for sustaining higher frequencies, combined with the fact that dynamic power consumption grows quadratically with respect to voltage (see Figure 2 for a simplified mathematical explanation).

$$DynPower = C \cdot V^2 \cdot f$$

$$P_{single} = C \cdot {V_1}^2 \cdot f$$

$$P_{dual} = 2 \cdot \left(C \cdot {V_2}^2 \cdot \frac{f}{2} \right) = C \cdot {V_2}^2 \cdot f$$

$$\frac{P_{dual}}{P_{single}} = \left(\frac{V_2}{V_1} \right)^2 < 1 \; since \; V_2 < V_1, hence \; P_{dual} < P_{single}$$

Figure 2: Dual-processor versus single-processor dynamic power consumption

So multiprocessing has indeed enabled hardware to keep scaling with Moore's law (number of transistors doubling every 18 months), but at the cost of a fundamental disruption in software performance scaling. For the software, "*the free lunch is over*", as well expressed by Herb Sutter's famous 2005 paper [1]: from this point on, in order to keep exploiting the hardware evolution, software had to be written in a concurrent and well balanced fashion in order to map efficiently on the multiple processors, or otherwise stated software needs to be "parallelized." Too bad that software parallelization is still one of the toughest challenges in computing in general, for which no satisfying general-purpose solution in terms of productivity has been found yet, despite the initial enthusiasm [2], and for many intrinsically serial applications there are no solutions at all.

The problem is so difficult that very little progress has been made in the last decade in PC applications, as well illustrated by the 2010 paper [3] of G. Blake et al. showing two fundamental points:

- initially dual-processors had the immediate benefit of improving the user perceived responsiveness, but
- after a decade most software, including demanding games, office applications, multimedia playback and Web browsing, still shows a good use of two processors only, with very few applications (like video authoring) being able to use more.

The fact is that, except for very few applications, PC software in large hasn't been parallelized at all since the introduction of multiprocessing a decade ago. Some concurrency is indeed naturally available, thanks to multi-tasking operating systems and to software engineering practices within applications like for event-driven code used to handle asynchronous events, such as user interface interactions. However these natural concurrent threads are typically very unbalanced - most often they are dependent on each other, and only a few of them make significant parallel use of the available processors, ending up in benefitting from no more than two processors.

The reason for which software developers don't parallelize their code is that for most of the PC applications it is simply not necessary or not economical to do so, with the exception of some niche markets for which performance is a compelling differentiating feature, such as for some multimedia applications, CAD and some specific professional areas. Video gaming is an area where good use of multiprocessing is expected, but as shown in the paper above it is actually still largely not the case. One reason for this is that Graphics Processing Units (GPUs) have evolved even faster than CPU multiprocessing in recent years, so it has been way more productive to invest in exploiting GPUs rather than on the tougher and less rewarding effort of parallelizing complex game engines for multiprocessing CPUs.

The situation is completely different in other areas, such as in Web data centers, where multiprocessing scales well with the nature of the parallel workloads, or in scientific computing where the effort of software parallelization is justified.

Is mobile like PC?

The smartphones performance evolution is fundamentally an accelerated and shifted-in-time version of what happened for desktops. The Apple iPhone evolution is a good example to use because it is easy to find consistent data over time, and it is also representative of most other mobile platforms as well.

In Figure 3 the CPU performance is illustrated by taking Dhrystone MIPS (DMIPS) as a measure of the CPU architectural efficiency, using data reported by ARM to characterize their processors. It should be interpreted similarly to the Instruction Level Parallelism (ILP) reported for Intel CPU's in Figure 1, i.e. as a measure of architecture efficiency, independent from the processor frequency. DMIPS-single represents the relative performance evolution of a single-processor, computed by multiplying the DMIPS/MHz by the frequency, while DMIPS-dual is simply DMIPS-single multiplied by two, indicating the overall peak performance of the dual-processors introduced by Apple starting with the iPhone

4S. For software performance we have simply taken the benchmarks scores reported by the recent iPhone 5 review from Anandtech [4]: Sunspider and Browsermark are Web browser (Javascript) benchmarks that being single-threaded cannot take advantage of multi-processing, while Geekbench is a multi-threaded benchmark and so it should also show the benefits of multiprocessing.
From Figure 3 we can notice very clearly two fundamental facts:

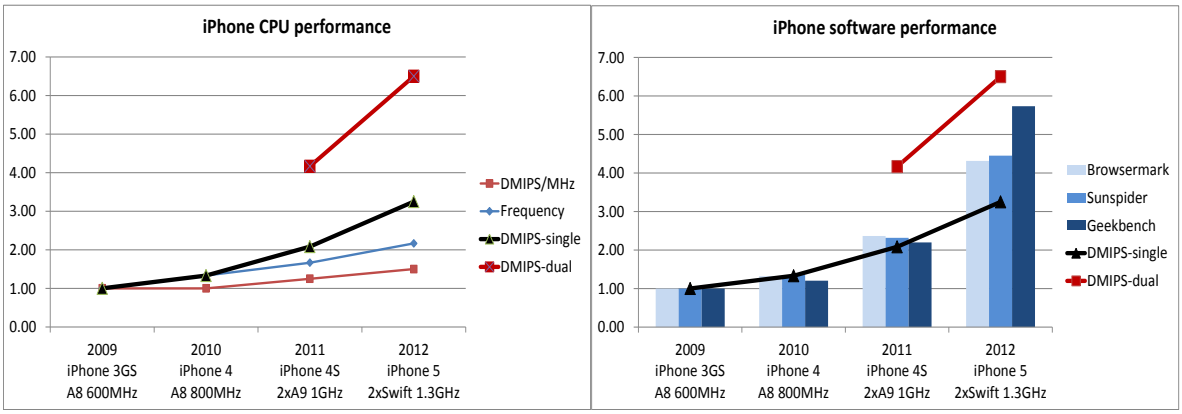

Figure 3: iPhone CPU and software relative performance evolution (sources: Wikipedia, Anandtech)

1. CPU architectural efficiency (DMIPS/MHz), frequency, and consequently single-core performance (DMIPS-single) are not saturated at all, on the contrary there is still a very strong acceleration. This trend is completely different from the evolution of PC single-processors that instead clearly saturated starting from 2003. A first preliminary conclusion is therefore that ***differently from PC processors, single-processor performance in mobile is not saturated at all yet***.
2. Software performance scales proportionally with single-processor performance as expected. Actually Sunspider and Browsermark grow more than proportionally when moving from iPhone 4S to iPhone 5. However this has nothing to do with multiprocessing since both phones are dual-processors, and the benchmarks are single-threaded. The reason is the significant improvements on Web browser software, Javascript optimizations in particular, and most likely other hardware optimizations as well, such as an improved memory subsystem. Geekbench instead is multi-threaded and less sensitive to system software evolution, so the significant growth measured for iPhone 5 seems indeed at least partially due to multiprocessing exploitation – although strangely the same scaling is not visible when moving from the single-processor iPhone 4 to the dual-processor 4S . In any case all benchmarks, Geekbench included, remain well below the theoretical peak performance of the dual-processor. We will provide more software scaling data later, but for what these few benchmarks can indicate we can at least preliminarily conclude that ***like for PC, software scales proportionally with single-processor performance, while depending on the application, it scales less than proportionally or not at all for multi-processors***.

The reason for which mobile platforms don't show yet the same saturation pattern observed for desktop CPUs is that the smartphone market exploded only very recently, giving strong motivation to platform vendors to optimize processor architectures and silicon technologies to a much greater extent compared to the less competitive market of classical embedded systems. It is clear that there was a large gap between the more mature and super-optimized architecture and silicon technologies

applied to PC processors, and the starting point of smartphone platforms: for PC processors there were no more margins once the heat limits had been reached in 2003, while in mobile platforms, when the smartphone era began around 2007, there were still huge single-processor performance growth margins. Today there are still no signs of saturation on mobile platforms – on the contrary actually.

The interesting question is: why have mobile platforms jumped into multiprocessing well before reaching the single-core performance saturation, differently from what happened for PCs? We can think of two main reasons: mobile computing leverages decades of previous knowledge. In particular it was already well known that dual-processors can be exploited effectively by modern operating systems, and the fundamental technologies required, such as cache coherency and multiprocessing capable operating systems, were already in place. There was no need to wait before making good use of dual-processors, and indeed it happened very quickly. The second reason is aggressive marketing.

What is less clear is the motivation to move right after to quad-processors, as happened for most of the commercial platforms, since the PC experience tells us that even after a decade of multi-processors in the market, using more than two processors is useless for the large majority of the software. The only technical explanation would be if mobile software were fundamentally more multiprocessing-friendly than on PCs, but as we will see shortly this is not yet the case.

The fact is that there are no strong technical reasons. The motivation appears to be all in the marketing which is very aggressive in smartphones given the highly competitive context. Today, the number of processors is currently used for differentiation - even for the end-user. Ironically nothing new is being invented here, not even in marketing, since the same message was used at the beginning of PC multi-processors. Like they did with PCs, people will soon realize that the number of cores in a mobile device does not correspond to true value added for the customers.

Web browsing is among the most important mobile applications that benefit most from higher processing capabilities. It is a good example of a demanding PC-class application whose performance is very important because it impacts directly user-visible reactivity. With ever increasing network bandwidth, processing speed is today in the critical path, with HTML5 rich APIs and growing programmatic (javascript) content making it even more demanding in the future. Web browsing is also a good example of efficient use of today's dual-processors thanks to the several concurrent activities induced for operating system support, for software engineering reasons, for interactivity (avoidance of UI freeze) and for security and robustness (multiple tabs in separate processes). However, very little or no advantage at all is measured when moving from dual to quad-processors, for lack of sufficient and well-balanced software parallelism needed for exploiting more than two processors.

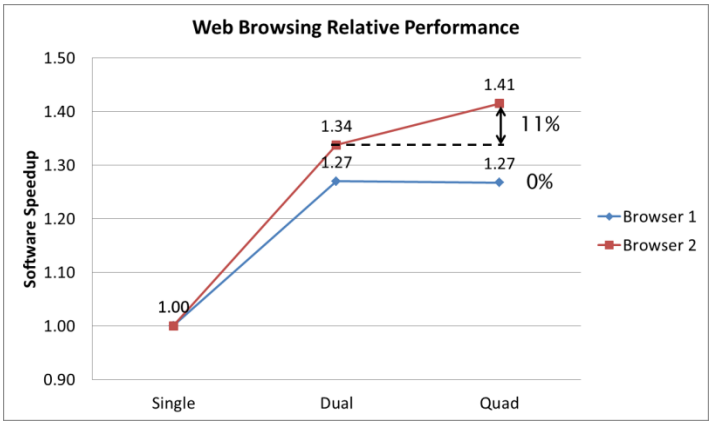

Figure 4: Web browsing & multiprocessing

In Figure 4 we show the relative measurements of two popular Android browsers on the same quad-processor hardware/software platform, enabling a different number of processors via software settings (hotplugging), so that measurements are relative to exactly the same hardware/software environment. The frequency is the same in all the configurations. Relative scores refer to page load times on a number of popular Web sites averaged over multiple repeated measures – so the use case represents real Web browsing, not an artificial benchmark. As already anticipated a relatively good speedup in the range of 30% is registered when moving from single to dual-processor, while only a modest 0-11% is obtained from the dual to the quad-processor configuration. Similar measures on our dual-processor platforms have shown better single to dual-processor scaling of up to 50%.

On the other hand, frequency improvements always pay off for software performance, whatever the number of processors, as already mentioned for the PC growth analysis. This fact is important when comparing multiprocessing solutions because increasing the number of cores has a negative effect on the frequency, due to conflicts on shared resources such as interconnect and memory, and because of the limited scaling of the cache coherency circuitry. So to obtain an overall gain, the lower frequency needs to be more than compensated by the software multiprocessing scaling. In a previous analysis of this tradeoff [5] we have shown that for example for a ~27% frequency penalty of a quad-processor versus a dual-processor, software needs to expose a parallel portion of at least 70% in order for the quad-processor to win, which is a huge proportion that very few applications can reach - certainly not through natural concurrency, as specific dedicated parallelization work needs to be done to obtain such high levels of parallelism.

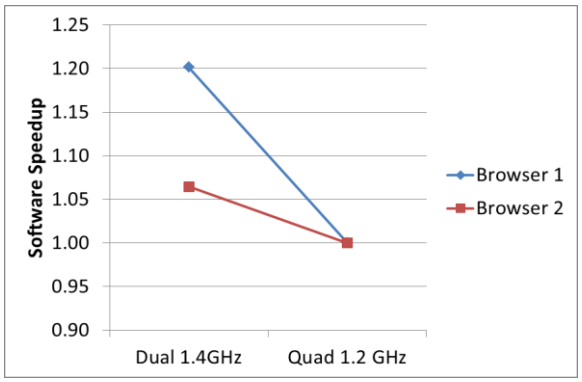

Figure 5: Faster dual-processor versus slower quad-processor

In particular for the Web browsing use case, given the already mentioned small level of effective software parallelism, a faster dual-processor easily outperforms a slower quad-processor. This is illustrated in Figure 5 where we have artificially lowered the frequency of the quad-processor configuration to 1.2GHz, to compare it against the dual-processor mode at 1.4GHz, to show that the faster dual-processor always wins, even for a modest frequency difference of less than 20%.

We have performed similar experiments on other important applications, like mobile video games, application start-up time, and even multimedia, obtaining similar results over and over again: marginal or no benefits at all when moving from dual to quad-processors at the same frequency, and 15-20% faster dual always beating slower quad-processors.

On the positive side, the availability of quad-processors in smartphones should stimulate software developers to use them more effectively than is currently the case, hopefully more successfully than on PCs in the last decade. There is indeed more sensitivity on available resources on smartphones than on PCs, and even if processors in mobile are getting closer to the ones in PCs, there is still a considerable gap, while on the software side there is a very strong pressure to run PC-class applications on mobile platforms. This, combined with the much lower power constraints and the stronger competition, should give mobile software developers enough motivations to invest more in better exploitation of multi-processors.

An area of particular interest is multimedia, which is often demanding and at the same time it is naturally well suited for parallelization. New exciting areas such as augmented reality, computational photography, gesture recognition etc. are good candidates for parallel processing, as they are not yet stable enough for hardware acceleration. There is a competing technology for these kinds of applications though, General Purpose GPU (GPGPU), also providing a programmable and multiprocessing solution mapped on GPUs rather than on CPUs. GPGPU is currently more difficult to program compared to multiprocessing CPUs, but GPU hardware is evolving very fast and progress is made in programming models and tools too, so it is not clear yet how things will evolve.

Heterogeneous multiprocessing

Heterogeneous multiprocessing has been introduced in mobile to cover demanding requirements with high-performance but power-hungry processors, in combination with slower but energy-efficient processors for less demanding loads. Heterogeneous quad-processor platforms have already been introduced, like NVIDIA's Variable SMP [6] technology in the Tegra 3, and others using the ARM big.LITTLE [7] solution are expected soon. The idea is certainly interesting and it will probably be successful over time; but, like for dual versus quad-core processors, simpler solutions are always preferable as long as they are feasible, especially if complex software changes are involved. We will see later how the FD-SOI silicon technology allows us to obtain similar benefits but in a much simpler and more effective way.

Heterogeneous multiprocessing brings complexity both in hardware and software. As an example the ARM big.LITTLE hardware solution is illustrated in Figure 6.

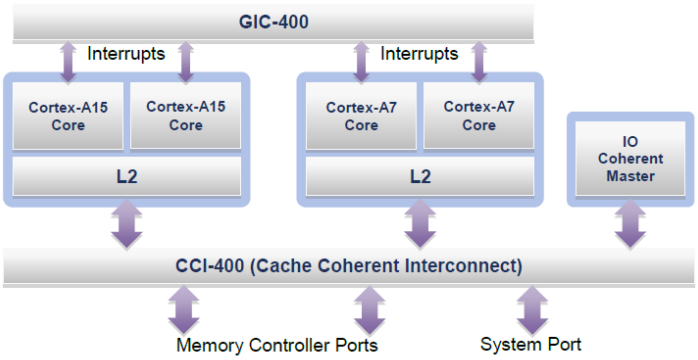

Figure 6: ARM big.LITTLE (source: [7])

The hardware complexity comes mainly from the fact that the processor caches must be kept coherent in order to be used in a shared memory system, as assumed by all smartphone operating systems, for which the ARM ACE interface has been introduced in the interconnect, at the expense of additional hardware complexity and coherency traffic.

On the software side, the management of the system heterogeneity by the operating system is potentially a very complex task, if the full potential of the architecture is to be exploited. Ideally the operating system should be sufficiently intelligent to distinguish demanding applications to execute on the big processor from less demanding ones to be executed in the little processors, and in order to limit expensive inter-cluster software migrations, such decisions should be relatively stable. At the same time applications often change behavior suddenly, so the system must also be able to react quickly to changes. This is mixed with dynamic voltage and frequency scaling, which is done independently on the two clusters, multiplying the number of power states to consider, etc. Now not only the kind of sophistication required for an optimal exploitation is far beyond from the capabilities of today's operating systems, but wrong decisions can actually be very counterproductive and user-visible in the form of system glitches caused by excessive software migrations, or poor performance and power waste derived from wrong decisions. Years of research and fine tuning will be required to optimally support these heterogeneous configurations.

In the meantime intermediate solutions are proposed to limit the possibility for the system to take wrong decisions, at the cost of reduced exploitation of the potential benefits. For example, one solution is to use the big and LITTLE clusters in an exclusive fashion, i.e. not concurrently, switching from one cluster to the other depending on the overall system load, avoiding the complexities of mapping intelligently each separate software thread. The downside of this approach is that clusters are never used concurrently, so the full potential performance is never reached.

Another more recent approach is to group big and little processors in fixed couples. Each big.LITTLE couple is then managed by the operating system as if it was a single abstract processor with extended voltage/frequency operating points. One limitation of this approach is the risk of frequent software migrations across clusters, in addition to the unnatural coupling of operating modes of the abstract processors due to the fact that in hardware the operating points are implemented by cluster, not by big.LITTLE processor couples.

In conclusion, heterogeneous multiprocessing is certainly a promising technology, especially for mobile; however, because of its complexity - especially in software - it will take years of research and fine tuning before expressing its full potential.

Further progress in silicon process technology: FD-SOI

There has been a lot of research recently in new technologies to guarantee further silicon process scaling, beyond the limits that the traditional techniques were about to reach, resulting in very successful breakthroughs, namely FinFET and FD-SOI. ST-Ericsson has already adopted STMicroelectronics' FD-SOI (Fully Depleted Silicon On Insulator) technology for its next generation 28nm platforms with extremely promising results. Here we briefly summarize the main advantages of FD-SOI from the perspective of computing only, showing how it enables further single-processor scaling, enabling us to keep focusing on faster dual-cores for the benefit of higher software performance.

Thanks to the insertion of the ultra-thin buried oxide substrate illustrated in Figure 7, FD-SOI brings a number of fundamental improvements in the transistor electrical characteristics, while it keeps using the very mature planar technology for chip fabrication.

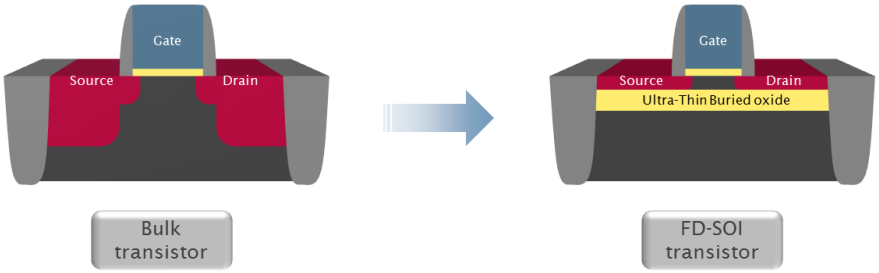

Figure 7: Bulk versus FD-SOI transistor

Among the most notable improvements we recall:

- Faster: in the same technology node, the FD-SOI transistor channel is actually shorter than for the bulk transistor. In addition, in contrast to the bulk transistor, the FD-SOI channel is fully depleted and free of dopants. These two factors together result in significantly faster switching at the same voltage, resulting in up to 35% faster operating frequency for the same power consumption at high voltage, and up to 100% faster at low voltage.
- Cooler: several factors contribute to lower power consumption: the fully depleted channel removing drain-induced parasitic effects and lower power operation, better confinement of carriers from source to drain, thicker gate dielectric reducing gate leakage current, and better control of body biasing techniques (voltage applied to transistor body to better control speed and power consumption). The result is very significant power reductions of 35% at high performance and up to 50% at low operating points.
- Simpler: process steps are 90% the same as for bulk 28nm technology, with even a reduction of 15% of the total number of steps, resulting in shorter cycle time. In addition FD-SOI doesn't require stressors or similarly complex techniques used in other processes. The

resulting process is less complex than bulk already, and far less complex than FinFET technology.

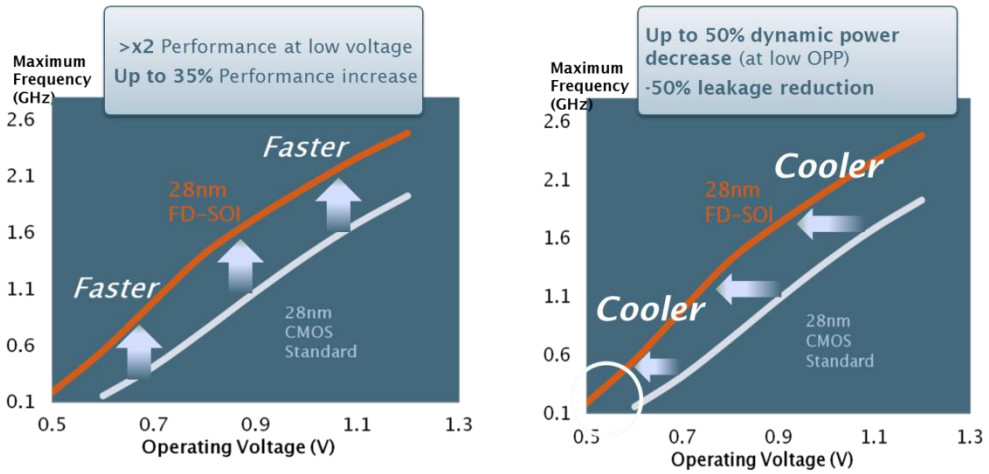

Figure 8: FD-SOI: faster and cooler

From a microprocessor design perspective the advantages of FD-SOI versus bulk are quite obvious (see Figure 8):

- Higher frequencies are achievable for the same voltage/power, or equivalently lower power is consumed at the same frequency;
- The maximum achievable frequency is higher;
- The processor can operate at lower voltages with still very respectable frequencies (1GHz at 0.65V). It is indeed in low power modes that the FD-SOI relative advantages become even more spectacular, as indicated in the low-voltage regions in Figure 8, reaching up to 100% higher frequency versus equivalent bulk low power modes.

The ~35% increased efficiency at high frequencies is already more than enough for a FD-SOI dual-processors to outperform slower bulk quad-processors in the vast majority of the use cases, due to the currently limited software scalability as we have previously discussed.

On the low-power side, the impact of FD-SOI is even more fundamental, as it reduces or eliminates the need to adopt the way more complex and still immature heterogeneous multiprocessing solutions.

The extended operating modes described above are achieved through a combination of the FD-SOI advantages described earlier, with body biasing playing a major role. Body biasing is the application of specific voltages to the transistor body, in order to optimally adjust its characteristics to each specific operating mode. Compared to bulk, FD-SOI enables the application of a wider range of biasing thanks to the elimination of the parasitic diodes between the body and the source and drain present in bulk technology.

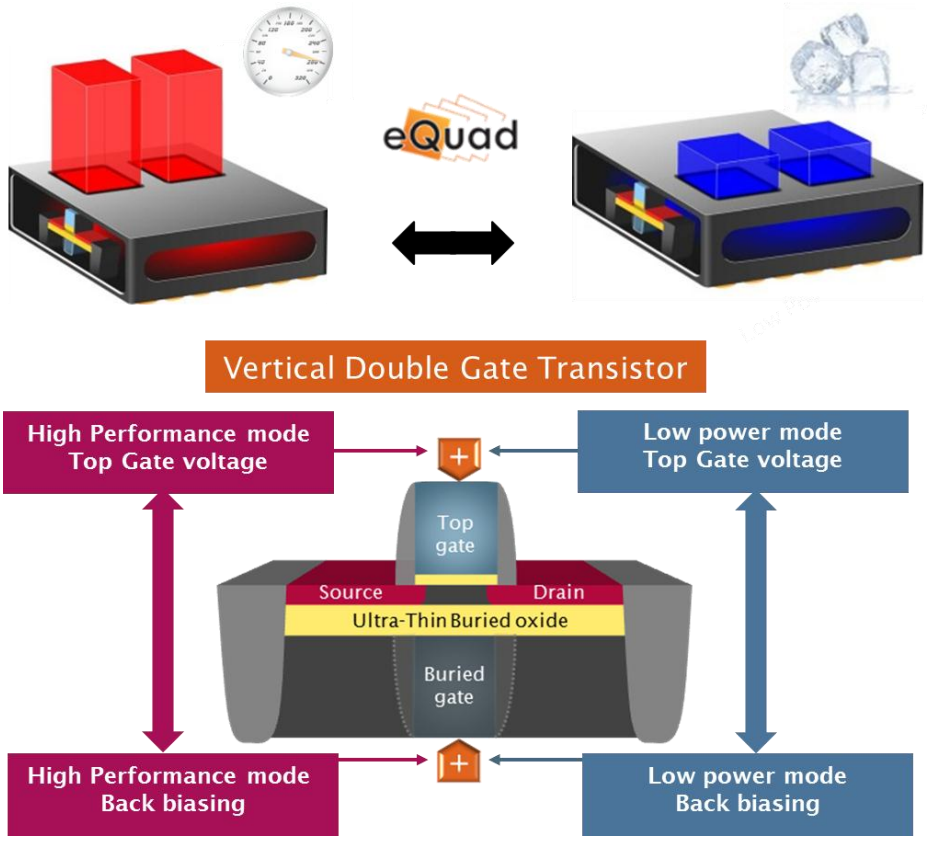

Figure 9: FD-SOI improves effectiveness of body biasing

In practice we obtain the same effect as if we had two distinct processor designs (illustrated in Figure 9). One optimized for high performance and the other one for low power, except that we do so using a single circuit and by switching electrically between high-performance and low-power modes, by changing the body bias voltage. ST-Ericsson calls this concept “eQuad” (eQuad: e = electrical), and the resulting behavior is equivalent to or better than the heterogeneous quad-processors that we illustrated earlier.

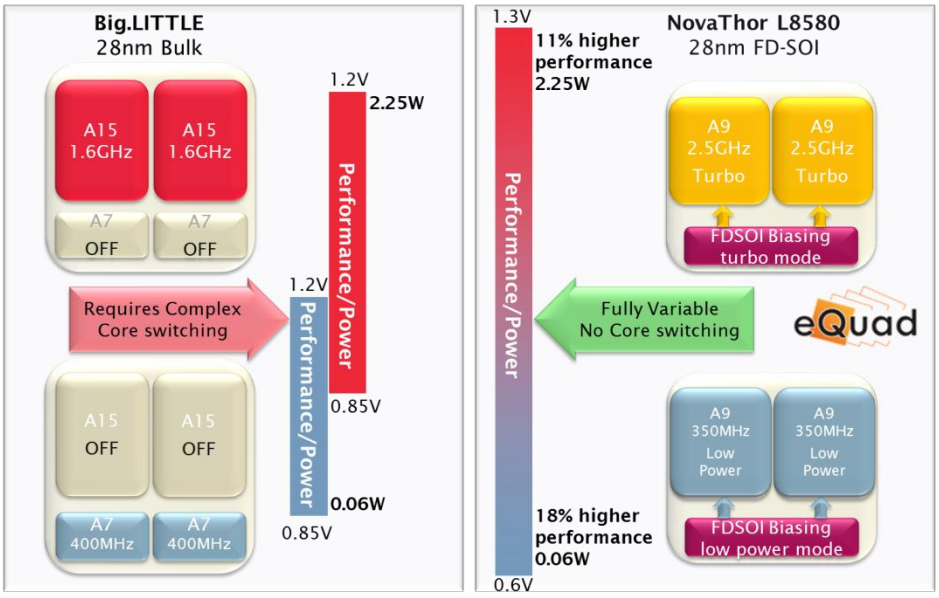

Figure 10: eQuad versus big.LITTLE

Figure 10 compares our first FD-SOI product, the NovaThor™ L8580 with an eQuad processor based on two ARM Cortex-A9 processors, versus a quad-processor in the big.LITTLE configuration running at frequencies that are representative of some competing products that have been recently announced. Thanks to the extended voltage/frequency range, the NovaThor™ L8580 delivers better performance for the same power consumption both at high-frequency and low-power operating points. And this is achieved with the Cortex-A9 processors which is the previous ARM architecture compared to the Cortex A15/A7. Beyond the better performance and power consumption, the other fundamental advantage of the NovaThor™L8580 is that it uses a traditional simple dual-processor configuration managed with mature software techniques, differently from the big.LITTLE product that instead requires the complex hardware/software management previously explained for heterogeneous architectures.

ST-Ericsson computing roadmap

The ST-Ericsson computing strategy and roadmap (Figure 11) reflect the observations illustrated so far: maximum exploitation of the still growing performance offered by the evolutions of silicon technologies, in a form that is best exploitable by the current generation of mobile software, which simply translates into faster dual-processor cores.

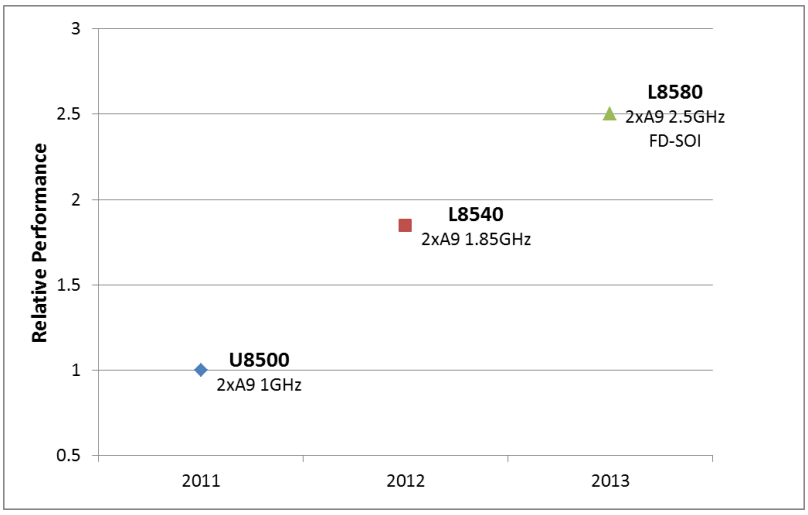

Figure 11: Simplified ST-Ericsson Computing Roadmap

It is worth noting that designing higher frequency dual-processors, while still respecting the mobile power consumption budget, requires more efforts than duplicating less aggressive designs resulting in lower frequency quad-processors. The reason for which we've been insisting in this direction is indeed for the software, which benefits directly from higher frequencies without the need to make any change, versus the huge efforts required to exploit more than two processors.

Looking ahead we are investigating various options, including moving to quad-processors and beyond as well. Sooner or later frequency scaling will be limited by power consumption, as it happened in PCs. For the next generation of products we can still benefit from the breakthrough in silicon technology provided by FD-SOI, but further ahead frequencies start getting close to those used in PCs, so inevitably we will also reach saturation for which moving to quad-processors will be

the only solution to keep scaling. Hopefully by that time software will also have evolved, so that more aggressive multiprocessors will start to be better utilized.

Conclusions

In this article we have highlighted a fundamental difference in the evolution of computing between PCs and smartphones: multiprocessors in PCs have been introduced as a necessity, back in 2005, once the heat dissipation limits caused the frequency increase to saturate, while in smartphones they have been adopted even before starting to observe any saturation in the frequency increase.

The reason for the early rush to dual-processors in mobile is that, thanks to PCs, the technology was already mature, so it could be readily implemented and exploited. Dual-processors are effective because of the natural yet limited software concurrency existing in operating systems and applications. The further move to quad-processors instead is mainly motivated by the aggressive marketing in the smartphone business. Technically speaking there are no convincing motivations, since the software is simply not ready to exploit effectively more than two processors. Indeed for a decade in PCs, and more recently in smartphones, most developers have not been sufficiently motivated to invest in the expensive and difficult task of software parallelization required to scale beyond dual-processors, resulting in poor exploitation of the current quad-processors.

Heterogeneous multiprocessing has also been recently introduced in order to extend the high-performance and low-power operating points of mobile platforms, through the use of different kinds of processors in the same system. This technology is certainly promising for smartphones, but it requires more complex hardware and very sophisticated control mechanisms in the operating systems. Preliminary solutions are available, but with limited support. A lot of further research and fine tuning is required for the full exploitation of this technology.

ST-Ericsson has been among the early adopters of dual-processors [8] [9], for their proven benefits. We have however resisted moving to quad-processors, giving instead preference to increase the operating frequencies of our dual-processors, since this is the most effective way to provide readily exploitable performance improvements to today's software.

In addition, the recent introduction of the FD-SOI silicon technology has allowed us to further extend the performance of our dual-processors beyond the reach of competing bulk silicon-based platforms. FD-SOI not only provides a very significant frequency increase at the same power consumption, but it also extends even more significantly the low-power operating modes, removing at least for now the necessity to adopt the more complex and less efficient heterogeneous multiprocessing configurations. This is achieved thanks to the several FD-SOI advantages, including the possibility to apply a wider range of body biasing for dynamically adapting the transistor characteristics from low power to high performance. ST-Ericsson calls "eQuad" the application of this extended set of operating modes, since our dual-processors becomes equivalent to heterogeneous quad-processors by electrically (eQuad) biasing the transistors to be optimized for low-power or high-performance.

Operating frequencies will eventually saturate, also with FD-SOI, forcing us to move to quad-processors and beyond as well. By that time we will still leverage FD-SOI for better performance and lower power, and software will have hopefully evolved for better multiprocessors exploitation.

References

[1] Herb Sutter, “The Free Lunch Is Over: A Fundamental Turn Toward Concurrency in Software”, Dr. Dobb's Journal, 30(3), March 2005.

[2] Krste Asanovic, Ras Bodik, Bryan Christopher Catanzaro, Joseph James Gebis, Parry Husbands, Kurt Keutzer, David A. Patterson, William Lester Plishker, John Shalf, Samuel Webb Williams and Katherine A. Yelick, “The Landscape of Parallel Computing Research: A View from Berkeley”, EECS Department, University of California, Berkeley, Technical Report No. UCB/EECS-2006-183, December 18, 2006.

[3] Geoffrey Blake, Ronald G. Dreslinski, Trevor Mudge (University of Michigan), Krisztián Flautner (ARM), “Evolution of Thread-Level Parallelism in Desktop Applications”, ISCA’10, June 19-23, 2010, Saint-Malo, France.

[4] Anandtech The iPhone 5 Review - Six Generations of iPhones: Performance Compared (http://www.anandtech.com/show/6330/the-iphone-5-review/9)

[5] Marco Cornero, “Quad cores in mobile platforms: is the time right?”, EETimes Europe October 2011 (also available here: http://www.power-eetimes.com/en/quad-cores-in-mobile-platforms-is-the-time-right.html?cmp_id=71&news_id=222903513&vID=35)

[6] “Variable SMP (4 -PLUS PLUS -1™) – A MultiA Multi -Core CPU Architecture for Low Power and High Performance”, white paper, NVIDIA (http://www.nvidia.com/object/white-papers.html)

[7] Peter Greenhalgh, “Big.LITTLE Processing with ARM Cortex™-A15 & Cortex-A7, Improving Energy Efficiency in High-Performance Mobile Platforms”, white paper, ARM September 2011 (http://www.arm.com/products/processors/technologies/biglittleprocessing.php)

[8] Press release: “ST-ERICSSON COLLABORATES WITH ARM TO DEMONSTRATE THE WORLD'S FIRST SYMMETRIC MULTI PROCESSING MOBILE PLATFORM TECHNOLOGY ON SYMBIAN OS”, February 16, 2009 (http://www.stericsson.com/press_releases/SMP_ARM_Symbian.jsp)

[9] Marco Cornero, “Enabling Symmetric Multi-processing for Symbian Mobile Devices”, Symbian Exchange and Exposition 2009 conference, London, 27-28 October 2009 (http://www.scribd.com/doc/22063340/Device-Creation-Marco-Cornero-ST-Ericsson)