DARK SILICON OVERVIEW, ANALYSIS AND FUTURE WORK

Advanced VLSI
Dr. Ram Sridhar
Spring 2015

Jordan Radice
jordanra@buffalo.edu

Contents

Introduction

For most of the history of the transistor, we were able to scale down the minimum gate lengths of the transistor while increasing the frequency all while maintaining a fairly constant power output. However, this trend seemed to come to a stop around 2005 as with each successive process generation, the percentage of a chip that can switch at the full potential frequency gain has dropped exponentially due to power constraints [1].

This is because we have departed from the traditional Dennardian scaling, which used to work in conjunction to Moore's law. In Moore's law, we would scale down the transistor lengths by half about every generation (~1.5 – 2.0 years) which would allow us to put twice as many transistors on the same chip in that time frame. Dennard's law used to work with Moore's law in that as we shrink the transistors down in size, we would proportionally achieve lower power consumption per transistor, maintaining the overall power density. However, due to the physical characteristics of scaling, factors like increased leakage current started to interrupt this which has led designers to come up with novel techniques to continue to make use of the increased number of transistors in a device.

This has eventually led to the term "dark silicon" since not all of the transistors are on at the same time or are not operating to their fullest frequency in order to maintain power requirements. In the following sections I will explain how we have gotten to this point and will detail many of the methods being employed today to mitigate the effects of dark silicon to make the most of the real estate.

Background

In both Dennardian and Post-Dennardian (Leakage Limited Scaling), the transistor count will scale by $\varkappa^2$, and transistor switching frequency will scale by $\varkappa$ [1]. Thus our net increase in compute performance from scaling is $\varkappa^3$ since we have $\varkappa^2$ as many transistors running simultaneous computations operating more $\varkappa$ as quick. The total power consumption can be seen from Equation 1 where as we scaled up the frequency by $\varkappa$ and increased the transistor count by $\varkappa^2$, the net power increase would increase by $\varkappa^3$. However, in order to maintain a constant power output, these gains must be offset by a corresponding reduction in transistor switching energy. In both cases, scaling reduces transistor capacitance (C_{ox}) by $\varkappa$, improving energy efficiency by $\varkappa$ as shown again in Equation 1. In Dennardian Scaling, we are able to scale the threshold voltage and thus the operating voltage, which gives us another $\varkappa^2$ improvement in energy efficiency (Equation 1). Below in Table 1, the Constant Field Scaling rules are shown in greater detail.

$$P = P_{dynamic} = (1\text{-}a)\times N\times C_{ox}\times f\times V_{dd}^2$$

Equation 1: Dennardian era power dissipation in a CMOS circuit where power (P) is related to supply voltage V_{dd}, transistor count (N), average fraction of transistors in off state (not switching) (a), I_{leak} is the leakage current, C is the transistor capacitance, and f is the operation frequency

Constant Field Scaling Rules (Pre-Dennardian Scaling Breakdown)		
Scaling Assumptions	MOSFET Device and Circuit Parameters	Multiplicative Factor ($\varkappa > 1$)
	Device dimensions (t_{ox}, L, W, x_j)	$1/\varkappa$
	Doping Concentration (N_a, N_d)	$\varkappa$
	Voltage (V_{dd}, V_t)	$1/\varkappa$
Derived scaling behavior of device parameters	Electric Field **E**	1
	Carrier velocity *v*	1
	Depletion-layer width (W_d)	$1/\varkappa$
	Capacitance ($C = \varepsilon \times L \times W/t_{ox}$)	$1/\varkappa$
	Inversion-layer charge density (Q_i)	1
	Current, Drift (I)	$1/\varkappa$
	Channel resistance (R_{ch})	1
Derived scaling behavior of circuit parameters	Circuit delay time ($\tau \sim CV/I$)	$1/\varkappa$
	Power dissipation per circuit ($P \sim VI$)	$1/\varkappa^2$
	Power-delay product per circuit ($P\,\tau$)	$1/\varkappa^3$
	Circuit density ($\propto 1/A$)	$\varkappa^2$
	Power Density (P/A)	1

Table 1: Scaling MOSFET Device and Circuit Parameters [2]

Scaling Breakdown

While there are a lot of other factors that come into consideration when scaling is done, such as gate-oxide leakage current, junction leakage and short channel effects, however, the main mechanism for preventing current densities from decreasing in future technologies is mainly the subthreshold conduction. In today's Post-Dennardian era, leakage-limited regime, we cannot scale threshold voltage without exponentially increasing leakage between the source and drain junctions of a MOSFET as shown in Equation 2. This is when there is a subthreshold leakage current while the gate-to-source voltage, V_{gs} is zero and while the drain-to-source voltage, V_{ds} is V_{dd}. So as a result of this exponential dependence, we must hold the operating voltage roughly constant. The new power equation is shown in Equation 3 which now includes the new idle (leakage) power dissipation. In Equation 3, the notable new variables are "N," which is the total transistor count, "a" which is the average fraction of the transistors in the off state and "K" which is a circuit design constant. The end result is that today, we have a shortfall of $\varkappa^2$, or $2^{\times}$ per process generation [1]. This is an exponentially worsening problem that accumulates with each process generation which will give way to an exponentially increasing amount of dark silicon in a chip.

$$I_{leak} = \mu_{eff} C_{ox} \frac{W}{L} (m-1) \left(\frac{kT}{q}\right)^2 e^{-qVt/mkT}$$

Equation 2: Leakage current (Off-current) of a MOSFET ($V_{ds} = V_{dd} >> kT/q$) [2].

$$P = P_{static} + P_{dynamic} = a \times N \times I_{leak} \times V_{dd} \times K + (1-a) \times N \times C_{ox} \times f \times V_{dd}^2$$

Equation 3: Post-Dennardian era Power dissipation in a CMOS circuit [3].

In order to maintain power density, sacrifices in other areas had to be made. This term "dark silicon" is namely transistors that aren't operating to their fullest capacity at all times. Some of these sacrifices included the scaling back of the frequency and depending on parallelization as well as increased usage in specialized hardware that is used during specific operations. Since Moore's law has continued to take place undisturbed by the constant field scaling breakdown, this dark silicon has continued to rise in the percentage of the chip and it is projected to increase upwards exponentially every year. What this means is that architecture engineers now have realized that

in order to increase the performance, it must be done in cleverer ways as scaling alone won't guarantee unprecedented performance as seen in the past.

Intel Pentium 4: Peak Performance with Constant Scaling

It was in 2004 when Intel released their 90 nm single core Pentium 4 Prescott processor [4]. Theoretically, from a 90 nm 3.8 GHz single core processor being scaled down to a 22 nm processor, we should have each of the 16 cores ($\varkappa^2 = (\frac{\text{Lmin}}{\text{Lmin}_*})^2 = (\frac{90\text{ nm}}{22\text{ nm}})^2$) operating at 15.2 GHz ($\varkappa = (\frac{\text{Lmin}}{\text{Lmin}_*}) = (\frac{90\text{ nm}}{22\text{ nm}})$) processor without any change in the power consumption with the corresponding C_{ox} and V_{th} being scaled back appropriately. Obviously, this never happened due to rapidly increasing power densities and overall power consumption. What did happen is we scaled back the number of processors we could have theoretically made on the same surface area to 6 and we scaled back the frequency gain to 3.6 GHz (Ex. Intel Core i7-4960X Ivy Bridge-E 6-Core 3.6GHz (Turbo 4GHz) 130W). This would theoretically yield a 5.7x peak serial instruction throughput improvement ($\varkappa = (\frac{3.6\text{ GHz}}{3.8\text{ GHz}})*(\frac{6\text{ Core}}{1\text{ Core}})$) [4] which is far less than the expected 64x improvement.

Post-Dennardian Scaling Breakdown

In order to avoid further increasing power densities, computer architecture engineers have taken consideration of many different alternatives and combinations thereof to make use of the additional transistors we now have. This in turn has led an exponential increasing percentage of the chip remaining dark. Assuming no sudden changes in fabrication techniques in the near future, we could be seeing percentage well over 90% dark in the coming years [4]. So because of this, architecture designers have instead focused on other techniques to mitigate the problem. These general strategies can be summarized in the following: The Shrinking Die, Bigger Caches, Dynamic Voltage and Frequency Scaling (DVFS), Near Threshold Voltage Processors (NTV), Parallelism, Hardware Specialization, and Deus Ex Machina.

The Shrinking Die

In the realm of dark silicon, one option is to simply decrease the size of the area of the chip accordingly as opposed to implementing dark silicon in the design. This can in fact improve the overall power consumption and speed of the same chip, while taking up less area, however, it will as well increase the power density of the chip. Initially that may not be an issue, but with an exponential dependence, this could be treacherous in the future to try and cool effectively with stringent TDP requirements [4].

This also poses the problem of profitability since companies would be selling the same chip that offers slight improvements in both power consumption and speed, but creating exponentially smaller chips doesn't yield exponentially smaller costs [4]. As a result, it not only makes sense from a profitability standpoint, but also a technological standpoint to better utilize this dark silicon as opposed to ignoring it – After all, you can always turn transistors off if you have them.

While the following example doesn't demonstrate the shrinking die to its fullest extent, it is worth pointing out that some die shrinks may be needed to accommodate space in mobile electronics as well as making the most of the chip you have – If there is no use for all of the dark silicon generated from a process shrink then why keep it?

Apple's A8 vs. A7 processor

Apple's latest two processors for their iPhone 6 and iPhone 5S demonstrate this notion of shrinking the die areas. In 2013, Apple released their iPhone 5S which used Apple's A7 processor which has over a billion transistors using 28 nm technology, operating at 1.3 GHz, on a 102 mm^2 die [5]. Just a year later, the iPhone 6 was released with the A8 processor which has over 2 billion transistors making use of 20 nm technology operating at 1.4 GHz, all while on a die with an area of 89 mm^2 [6]. This was all done while maintaining or improving the phone's overall battery life on a single charge when using the phone's

different functions [7]. While the iPhone 6 has a battery that is 16% larger ($\frac{1810\ \text{mAh} - 1560\ \text{mAh}}{1560\ \text{mAh}}$) than the iPhone 5S, it is worth noting that the iPhone 6 also larger screen maintain its pixel density resulting in a resolution that is 38% larger ($\frac{1\ \text{MP} - 0.727\ \text{MP}}{0.727\ \text{MP}}$) [7-9]. So with that, this is because between the A8 and the A7, there is a 50% improvement in efficiency from the processor despite the fact there are more transistors running at a slightly higher clock all in a die that is slightly smaller [6].

Dim Silicon

Dim silicon refers to the techniques that put large amounts of otherwise dark silicon area to productive use by heavily underclocking or infrequently using certain areas of the chip to meet the power budget [4]. Within this domain, therein lies several techniques such as increasing the cache size since it is often infrequently used but can improve the performance during cache calls, or varying the frequency of the transistors, operating at near threshold or taking advantage of parallelism.

Bigger Caches

In the past when area was expensive, computer architecture designers had to make the most of the area so having large amounts of SRAM was not always an option. But nowadays, area is cheap, power is expensive so increasing the cache size is probably the simplest way to improve performance while making more use of the chip since cache is inherently dim as it is only periodically accessed. This is because only a subset of cache transistors (such as a wordline) is accessed each cycle resulting in cache memories having a low duty cycle. Compared to general-purpose logic, a level-1 (L1) cache clocked at its maximum frequency can be about 10x darker per square millimeter, and larger caches can be even darker [4]. Thus, adding cache is one way to simultaneously increase performance and lower power density per square millimeter. Suppose that expanding per-core cache at a rate that soaks up the remaining dark silicon area, this would provide 1.4 to 2x more cache per core per generation. However, many applications do not benefit much from additional cache, and upcoming TSV-integrated DRAM will reduce the cache benefit for those applications that do [4].

Optimizing Cache Sizes

To demonstrate the optimum cache size in a processor, we can simulate a technology node, 20 nm high performance double-gate FinFET on a core design (Conventional general-purpose cores modeled after Sun UltraSPARC), with DRAM memory off chip all within a die area of 310 mm^2 while using application characteristics with cache misses modeled after TPC-H on DB2 with 99% parallel code [3]. The plot in Figure 1 shows the collective chip performance as a function of L2 cache size on the chip. For each point in the figure, a fraction of the die area is dedicated to an L2 cache size as indicated on the x-axis, 25% of the die area is used for the supporting structures and the remaining area can be populated with cores [3].

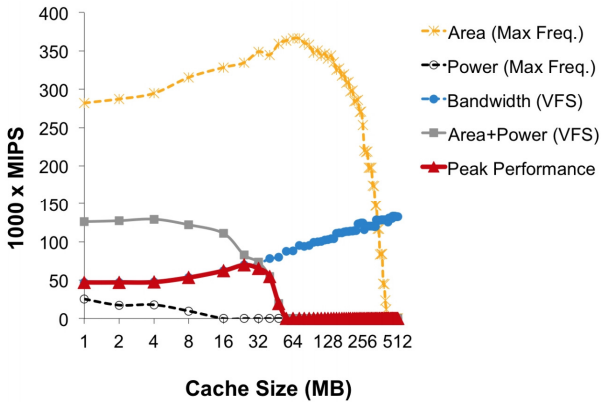

Figure 1: Performance of a multicore with general-purpose cores running TPC-H queries against a DB2 database at 20 nm [3]

The Peak Performance curve shows that only feasible multicore designs that conform to all physical constants (Area, Power, Bandwidth). At small cache sizes, the off-chip bandwidth is the performance limiting factor whereas as the cache size is increased, we notice that the performance increases, despite the fact that there are fewer cores operating in parallel [3]. However, it is at 24 MB that we find the maximum point on the Peak Performance plot, which is the optimum point for our cache size when trying to justify how large to make it at the expense of parallelism due to the reduction in the number of cores.

Dynamic Voltage and Frequency Scaling (DVFS)

Dynamic voltage and frequency scaling is done by adjusting the supply voltage V_{dd} and therefore the frequency to get a dramatic performance boost (Refer again to Equation 1). In the power equation, the voltage is taken to the second power while frequency is multiplied to it as well. This is done when the computational load is instantly increased in which a sudden burst of computational power is needed. In DVFS, the frequency of the core can be increased as the supply voltage is increased. When the frequency is ramped up, we incur a cubic power increase, but a cubic power decrease is not observed. This comes from the fact that during the voltage (and frequency) ramp up, we are increasing the temperature rapidly as well, whereas during the ramp down, the device isn't cooled nearly as quickly as it was heated due to the thermal dissipation properties. The speed can remain high until either the task is completed or until the thermal limit is reached in which the processor is then throttled back to a more temperature appropriate speed to prevent thermal runaway [4]. This technique is heavily being utilized in the mobile industry the most since in this industry, there is a desire to have a thin design that has few to no fans in it all while preserving the battery life.

Apple's 2015 MacBook

A perfect example of this is Apple's new fanless MacBook, which has a base 1.3 GHz dual-core Intel Core M Processor which utilizes Intel's Turbo boost which speeds up the processor up to 2.9 GHz based on the load [10]. The laptop features thin dimensions of 28.05 cm by 19.65 cm by 0.35 to 1.31 cm (Depending on the side) all while weighing 0.92 kg [10]. The laptop also has a 12" screen with a 2304 x 1440 resolution while using a 39.7-watt-hour battery that provides for 9 hours of wireless web usage. While there has been a lot of other improvements in processor technology, it is Intel's Turbo Boost technology must be implemented in order to provide smooth performance under varying loads.

Parallelism

Parallelism is the ability to take a task and allow it to be worked on simultaneously amongst several cores. This didn't start to become a popular way to do computing until after Intel's Pentium 4, mentioned earlier, when the performance literally peaked in a single core device. It was for this very reason that the successor, the Pentium D Smithfield was introduced the following year in 2005 [11]. While parallelism has its benefits when utilized properly, the problem is that there aren't a lot of user applications that benefit from parallelism and the ones that due rarely benefit from more than two cores. Within multicore systems, there are several architectures one can use.

Symmetric Multicore: It is a symmetric or homogenous multicore topology consisting of multiple copies of the same core operating at the same voltage and frequency [12]. In a symmetric multicore system, the resources including the power and area budget are shared equally amongst all cores [12]. While there is the immediate benefit of parallelism, this design does not have the ability to throttle back individual cores reduce power consumption, which results in degrading the efficiency of the system.

Asymmetric Multicore: In the asymmetric multicore topology, there is one large monolithic core and many identical small cores [12]. This design utilizes the high performing core for the serial portion of the code and then leverages the smaller cores as well as the large core to compute the parallel portion of the code [12]. This architecture is better suited for programs in which most of the heavy workload can only be utilized by a single core while the parallel workload is light, thus benefiting from the smaller, less powerful cores.

Dynamic Multicore: The dynamic multicore topology is a variation of the asymmetric multicore topology in which during the parallel portions of the code, the large core is shut down and during the serial portion, the small cores are turned off and the code only runs on the larger one [12]. This is likely done since the smaller cores would share the same design, thus making it easier for parallelism to take place.

Composed Multicore: The composed multicore topology consists of a collection of the small cores that can logically be combined together to compose a high performance larger core for the execution of the serial portion of the code [12]. Each of these methods have their tradeoffs, which means that their implementations largely depend on the environment that it would be implemented in.

ARM's big.LITTLE

ARM's big.LITTLE has three main implementations, depending on the scheduler implemented in the kernel [13, 14]. This is where there are four powerful cores and 4 low power cores, each for their own tasks. When in use on a mobile device, the powerful cores are used for intensive tasks, like opening a web browser or an app, whereas lower power tasks (Or idle states) are used power efficiency. The different implementations can be shown below in Figure 2,Figure 3, Figure 4.

Clustered switching arranges the processor into identically sized clusters of "big" or "LITTLE" cores (Figure 2) [15]. The operating system scheduler only sees one cluster at a time. When the load on the system changes from low to high, the system would then change from the low cluster to the higher cluster. All of the relevant information is passed through the common L2 cache [15]. The lower core cluster is then turned off and the higher core cluster is then turned on. This method is being used on the Samsung Exynos 5 Octa (5410) [15].

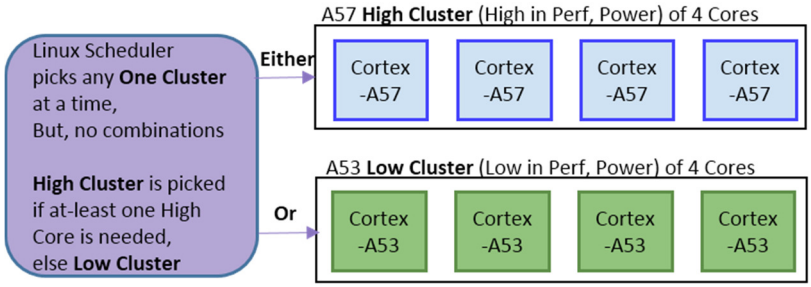

Figure 2: Clustered switching [15]

In-Kernal switching involves having virtual cores where the powerful core and the power efficient core are paired together (Figure 3) [14]. Only one real core is actually powered up at a given time (Either the high or the low in a given virtual core) [14]. Without question, the high performing core is turned on when the load is high (power efficient core is turned off), and the vice-versa for the idle or lower demand situations. This one power core to one power efficient core is the simplest example – one could pair up multiple power efficient cores with a power core for lower power multitasking requirements.

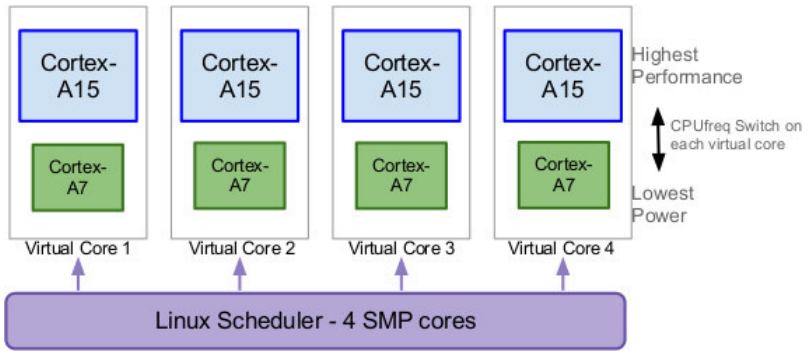

Figure 3: In-Kernal Switcher [14]

The most powerful usage of this combination between high performing cores and power efficient cores lies within heterogeneous multi-processing [14]. This is where we enable the use of all of the physical cores at the same time [14]. Threads with higher priority or computational demand can be allocated to the big cores while the lower priority or less intensive tasks can be sent to the LITTLE cores.

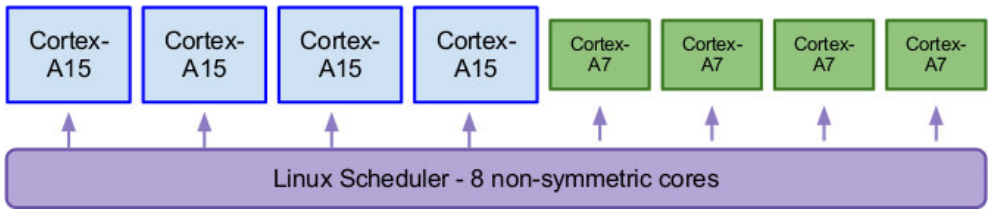

Figure 4: Heterogeneous multi-processing [14]

Near Threshold Voltage Processors (NTV)

A relatively new emerging approach is to operate the on state of the transistor near the threshold voltage. In order to utilize this effectively, it will have to make use of the previously described parallelism technique. This is because in NTV, the processors suffer a major performance hit for a significant (but not greater) energy savings. Recently, researchers have looked at wide-SIMD implementations of NTV processors which seek to exploit data parallelism, the most efficient form of parallelism and also NTV many-core implementations and an NTV x86 (IA32) implementation [1].

Although per-processor performance of NTV drops faster than the corresponding savings in energy-per-instruction (Say a 5x energy improvement for an 8x performance cost), the performance loss could be offset by using 8x more processors in parallel if the workload and the architecture allow it [1]. So assuming perfect parallelization, NTV could offer the throughput improvements while absorbing 40x the area – about eleven generations of dark silicon [1]. If 40x more free parallelism exists in the workload relative to the parallelism “consumed” by an equivalent energy-limited—super-threshold many core processor, then there is a net benefit to employ NTV in deep-dark silicon limited technology [1].

These descriptions as stated above can be found below in Figure 5 which shows both the frequency and power output vs. the supply voltage as well as the energy efficiency and active leakage power vs supply voltage. Figure 5b clearly indicates that that for 65 nm CMOS technology, the near most optimized point to operate at for NTV is at 0.32 V. It is also worth noting that on Figure 5a, you can see that both x-axis are in log scale and from this, one can find the magnitude of the frequency slope is slightly larger than the power slope, proving the notions stated above.

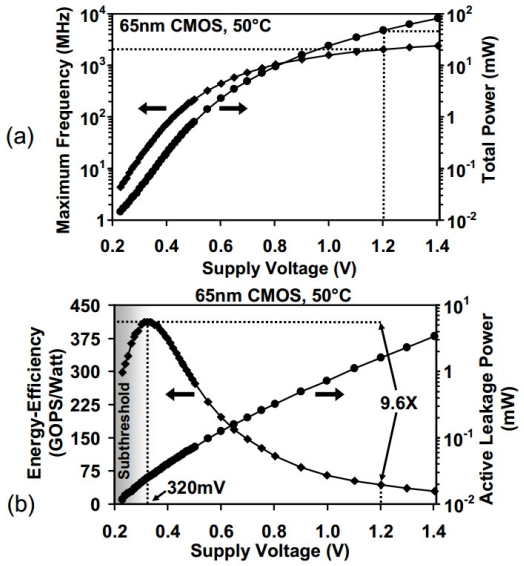

Figure 5: Energy Efficiency of NTV Operation [16]

NTV presents some circuit related challenge since it is very susceptible to process variation given its operation at near the threshold. As the operating voltage is decreased, the variation in the transistor threshold due to random dopant fluctuation (RDF) is proportionally higher, and the variation in operating frequency can vary greatly [1]. Since NTV designs expand the area consumption designs greatly (In this case by 8x or more), process variation issues are exacerbated, especially in SIMD machines which typically have tightly synchronized lanes [1]. Recent efforts have looked at making SIMD designs more robust to these variations [1]. Other challenges include penalties involved in designing SRAMs that can operate at lower voltages and the increased energy consumption due to longer interconnect caused by the spreading of computation across a large number of slower processors [1].

Hardware Specialization

Another way to improve the usage of silicon is through special-purpose cores (As opposed to only general-purpose processors). This can be done since we are at a time where power is now more expensive than area [4]. During the younger days of silicon, area was the most important parameter when designing a chip, but due to the scaling and the mobile usage of electronics, the engineering priorities have changed. These specialized cores can be used to fill in the dark silicon (Silicon that previously was unused due to power requirements). They can be of importance since they are specialized for certain tasks, which can then save energy and time as compared to a general-purpose processor. They can be upwards of 10-1000x more energy efficient than a general-purpose processor [4]. This would ultimately lead to an increase in the usage of the silicon as well as a lower overall power budget. These coprocessors are called “Coprocessor Dominated Architectures” or CoDAs.

These specialized cores can be designed specifically based on trending usages. For instance, if most of the power consumption is from web usage, then we can make cores that are better suited for those apps [4]. Of course the drawback is that the hardware itself is likely to age more quickly since software and computer usage varies a lot year to year. This is the inherent tradeoff from hardware specialization – difficult to change with new software demands on the same processor. This would mean that there would be more stress on hardware designers to come up with newer specialized cores more quickly in the volatile software environment. Of course, in a company where the software and hardware is made together, this can be done in a much more harmonious way. In the end, it is about making better utilization of the area on the chip since area has gotten to be much cheaper over the years. These techniques are already being utilized in graphics processors and have their own specialized languages such as CUDA, which are not mutually intelligible between similar architectures like AMD and Nvidia [1].

UCSD's GreenDroid

The GreenDroid architecture is an example of a Coprocessor-Dominated Architecture (CoDA) that seeks to address both complexity issues and Amdahl limits [1, 4]. The GreenDroid mobile applications processor (Figure 6a) is made up of 16 non-identical tiles. Each tile (Figure 6b) holds components common to every tile – the CPU, on-chip network (OCN), and shared L1 data cache and provides space for multiple c-cores of various sizes [1]. Figure 6c shows connections among these components and c-cores.

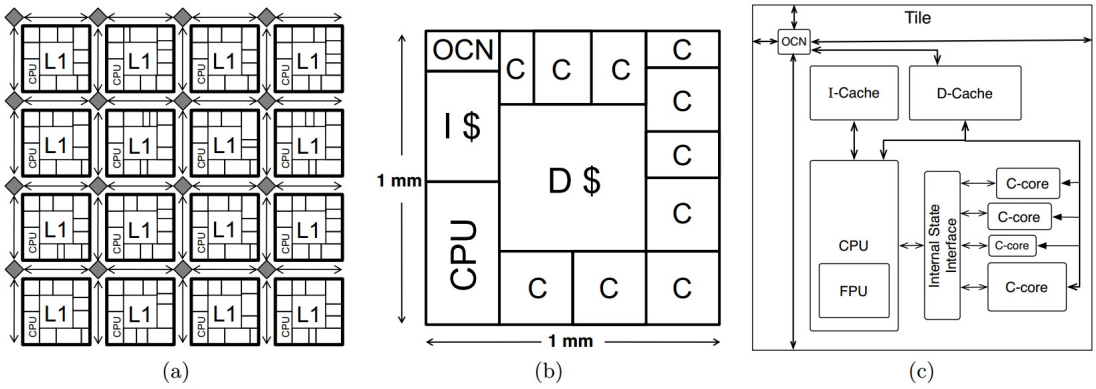

Figure 6: GreenDroid Architecture [1]

GreenDroid is a mobile application processor that implements Android mobile environment hotspots using hundreds of specialized cores called conservation cores (c-cores) [4]. C-cores, which target both irregular and regular code are automatically generated from C or C source code, and support patching mechanisms that lets them track software changes [4]. They attain an estimated ~8 to 10x energy efficiency improvement at no loss in serial performance, even on nonparallel code and without end user or programmer intervention [4]. Unlike NTV processors, c-cores don't need to find additional parallelism in the workload to cover serial performance loss. Thus, c-cores are likely to work across a wider range of workloads.

Elastic Fidelity

Although traditional designs correct all errors and provide accurate information, not all computations and data in a workload need to maintain 100% accuracy. Rather, different portions of the execution and data exhibit different sensitivity to errors and perfect computation is not always required to output acceptable results. For example, computations that involve human perception provide a lot of leeway for occasional errors since visual and auditory after-effects compensate for them [3]. Applications like networking already have built in error correction techniques as they assume unreliable components [3]. Computations performed on noisy data such as ADCs are usually equipped with techniques to correct inaccuracies within reason. When these requirements are less stringent, therein lies the ability to operate components at a significantly lower voltage, quadratically reducing the dynamic energy consumption.

Elastic fidelity capitalizes on these power gains by laxing the reliability guarantees of the hardware based on the software requirements, and carefully letting errors manifest in the error-resilient portion of an application's data set [3]. Programming language subtyping designates error-tolerant sections of the data set, which are communicated through the compiler to the hardware subsystem [3]. The hardware then operates components (functional units, cache banks etc..) occasionally at low voltage to reduce energy consumption. Portions of the application that are error-sensitive execute at full reliability, while the error-tolerant computations run on low-voltage units to produce an acceptable result.

Similarly, error-tolerant sections of the data are stored in low-power storage elements (low-voltage cache banks, low-refresh-rate DRAM banks etc..) which allow for occasional errors. By not treating all code and all data the same with respect to reliability, elastic fidelity exploits sections of the computation that are error-tolerant to lower power and energy consumption, without negatively impacting executions that require full reliability. Overall, 13–50% energy savings can be obtained without noticeably degrading the output, while reaching the reliability targets required by each computational segment [3].

Deus Ex Machina

This problem could very well be solved in the future with some sort of new and unexpected breakthrough in technology when it seemingly feels like there are no other options left. In the past this has been represented in new technologies such as FinFets, Trigate and High-K dielectrics which allowed transistors to scale further without severe leakage issues [4]. These advancements are however improved as one-time enhancements and are still limited to the 60 mV/decade subthreshold slope. This could be something like Tunnel Field Effect Transistors (TFETs) which have better subthreshold slopes of around 40 mV/decade at lower voltages [17]. It could possibility lend itself to NEMS, which have essentially a near-zero subthreshold slope but slow switching times [4].In the future however, this isn't necessarily limited to scaling techniques in the typical a FET device, but it could be the possibility of the invention of a whole new way of computing, like quantum computing, which would bring unprecedented gains, unseen in anything before and would essentially nullify any of our current work due to quantum computing's insane potential [18].

Conclusion

Barring any major advancements in the technology itself, it'll be up to the computer architecture engineers to find better ways to utilize the chip to its fullest potential. While there are a lot of different areas to improve the efficiency of our computers, at its heart, all this we are doing is making the most of what we have. In the past, we have been very fortunate to simply scale most of the design parameters accordingly without any serious adverse effects. But due to the focus in today's society on mobile processors, and the fact that we have a separation from Moore's Law and Dennard's law, we can no longer simply scale things down and leave everything running at full speed. It is through the combination of these different techniques that we will be able to push devices even further, bringing about new capabilities in the mobile field.

Future Work

With everything outlined above, if time permitting, I would like to investigate further into ways in which mobile class processors can be simulated to try and find a sweet spot in core clusters in the parallelism architectures and dynamic voltage and frequency scaling to optimize performance and power consumption. I'm not sure how this could be done, but I believe that if we can find the right balance between large cores and smaller ones to work together, efficiently, it could lead to longer lasting batteries on smartphones which can be significant since most smartphones nowadays don't last much longer than a day.

References

1. Taylor, M.B. *Is dark silicon useful? Harnessing the four horsemen of the coming dark silicon apocalypse*. in *Design Automation Conference (DAC), 2012 49th ACM/EDAC/IEEE*. 2012.
2. Taur, Y. and T.H. Ning, *Fundamentals of Modern VLSI Devices*. 2009: Cambridge University Press. 680.
3. Hardavellas, N., *The Rise and Fall of Dark Silicon.* USENIX 37, 2, 2012.
4. Taylor, M.B. *A landscape of the new dark silicon design regime*. in *Energy Efficient Electronic Systems (E3S), 2013 Third Berkeley Symposium on*. 2013.
5. Shimpi, A.L. *The iPhone 5s Review*. AnandTech 2013; Available from: http://anandtech.com/show/7335/the-iphone-5s-review/2.
6. Anthony, S. *Apple's A8 SoC analyzed: The iPhone 6 chip is a 2-billion-transistor 20nm monster*. ExtremeTech 2014 2015-04-25]; Available from: http://www.extremetech.com/computing/189787-apples-a8-soc-analyzed-the-iphone-6-chip-is-a-2-billion-transistor-20nm-monster.
7. *Compare*. 2014; Available from: https://www.apple.com/iphone/compare/.
8. Klug, B. *Apple Increases iPhone 5C and 5S Battery Sizes relative to iPhone 5*. 2013; Available from: http://www.anandtech.com/show/7324/apple-increases-iphone-5c-and-5s-battery-sizes-relative-to-5.
9. Lee, P. *iPhone 6 Teardown*. 2014 2015-04-25]; Available from: https://www.ifixit.com/Teardown/iPhone+6+Teardown/29224.
10. *MacBook - Tech Specs*. 2015 2015-04-25]; Available from: http://www.apple.com/macbook/specs/.
11. *Intel® Pentium® D Processor 800Δ Sequence and Intel® Pentium® Processor Extreme Edition 840 Δ*. 2006; Available from: http://download.intel.com/support/processors/pentiumd/sb/306832.pdf.
12. Esmaeilzadeh, H., et al., *Power Limitations and Dark Silicon Challenge the Future of Multicore.* ACM Trans. Comput. Syst., 2012. **30**(3): p. 1-27.
13. Jeff, B. *Ten Things to Know About big.LITTLE*. 2013 2015-04-28]; Available from: http://community.arm.com/groups/processors/blog/2013/06/18/ten-things-to-know-about-biglittle.
14. Grey, G. *big.LITTLE Software Update*. 2013; Available from: https://www.linaro.org/blog/hardware-update/big-little-software-update/.
15. Clarke, P. *Benchmarking ARM's big-little architecture*. 2013; Available from: http://www.embedded.com/electronics-news/4419448/Benchmarking-ARM-s-big-little-architecture.
16. Kaul, H., et al. *Near-threshold voltage (NTV) design — Opportunities and challenges*. in *Design Automation Conference (DAC), 2012 49th ACM/EDAC/IEEE*. 2012.
17. Appenzeller, J., et al., *Band-to-Band Tunneling in Carbon Nanotube Field-Effect Transistors.* Physical Review Letters, 2004. **93**(19): p. 196805.
18. Lenstra, A.K., *Integer Factoring.* Designs, Codes and Cryptography, 2000. **19**: p. 101-128.