Energy Aware Software: Issues, Approaches and Challenges

Sébastien Lafond*, Simon Holmbacka †, Johan Lilius‡
Faculty of Science and Engineering, Åbo Akademi University
Turku, Finland
Email: *sebastien.lafond@abo.fi, †simon.holmbacka@abo.fi, ‡johan.lilius@abo.fi

***Abstract*—Reducing the power dissipation and energy consumption of computing systems is an important challenge for system designers. From hand-held devices, drawing energy from batteries, to large scale datacenters, having as energy consumption several MWh, the development of energy reduction techniques is a key enabler to get a continuous performance increase of computing systems. The development of energy reduction techniques requires an overall view and deep understanding of the underlying software, middleware and hardware. At the same time, computing system designers have always a set of system requirements to fulfill. Manifestly, this always leads to trade-offs between energy consumption and other system requirements.**

This paper provides insights on the current issues, approaches and challenges computing system designers are facing when developing energy aware platforms. The main focus is put on two important aspects of system design: a) how to energy efficiently map a software application on hardware components and b) how to energy efficiently manage system behavior changes at runtime.

I. Introduction

To provide a continuous performance increase of computing systems, the clock frequency wall was avoid in the beginning of the 2000s by the introduction of multi-core based systems [1]. This led to a new challenge in mid-2000s: the CPU power wall [1], [2]. Indeed, due to physical limitations of semiconductor materials, the power dissipation of a fixed chip area is limited. With the evolution of semiconductor device fabrication, from $10\mu m$ manufacturing process in the 70s until the commercialization of $14nm$ based technology CPU in 2015, more transistors are laid down into smaller silicon area introducing an increase of the power density [3]. This eventually introduced drack silicon, when all the components of a chip can not be operated at the same time due to its power dissipation and generated heat. The effects of the power wall was until now limited by the design of more energy efficient transistors with for example lower voltage levels. This had the advantage to increase the transistor efficiency in proportion to the increase of transistor density, a phenomenon called the Dennard scaling [4]. However the current increase of the transistor efficiency is not anymore proportional to the increase of transistor density [2], and the semiconductor device fabrication is currently in a post-Dennard scaling area, where a substantial increase of the performance of computing systems can not be obtained before solving the current power dissipation and related energy consumption issues.

In 2007 alone the energy consumed by data-centers in western Europe was already 56TWh. In 2010 the energy consumption of data center was estimated to represent between 1.1 to 1.5 % of the worldwide electrical energy production [5]. In 2012 data centers had an electrical consumption of 270 TWh, representing an compound annual growth rate of 4.4% since 2007 [6]. A recent study [7] predicts the energy consumption of data centers in US will reach 73 TWh in 2020, representing more than 5 times the production capacity of one new EPR nuclear reactor.

Thefore energy efficiency is, for already several years and for almost any type of computing systems, a key issue. Increasing the energy efficiency is one of the main objectives when trying to lower the energy bill of large data centers or aiming at increasing the lifetime of battery operating devices. Being able to decrease the energy needed to perform a required number of operations necessitates an overall view and deep understanding of the underlying software, middleware and hardware. Different approaches can be used on these components to increase the energy efficiency of the overall systems. However, as energy is never the unique requirement for a computing system, increasing the energy efficiency always leads to a trade-off between energy consumption and other systems requirements. Generally, system designers end-up with the challenging problem of solving the trade-off between energy consumption and performance. The non-linearity between energy consumption and performance was first highlighted as a major problem in 2007 by Luiz André Barroso and Urs Hölzle [8]. Achieving energy proportional computing is an important goal, especially for computing systems having variable loads and operating at different utilization rates [9].

In a CPU, the non-linearity between energy consumption and performance originates from the level of static power dissipation, the thermal effects and the exponential relationship between the supply voltage and power dissipation. For long time the power dissipated by a processing element was mainly due to the switching activities of the load capacitance in the circuit gates. However, due to technology scaling the static power dissipation is exponentially increasing and started to dominate the overall power dissipation in microprocessors in the 2000s decade [10] [11]. At the same time the temperature of a microprocessor directly influences the static power dissipation as an increase of temperature triggers an increase of leakage currents [12]. The exponential relationship between the supply voltage and power dissipation is widely exploited by run-time systems adapting the voltage levels according to current resource requirements. However, dues to propagation delays, lowering the supply voltage implies a reduction of the clock frequency, which might translate into lower performance.

II. Power and Energy in Processing Elements

The power dissipation in a CPU originates from two distinct sources: the dynamic power Pd and the static power Ps . The static power dissipation comes from leaking currents in transistors, mainly the subthreshold [10] and the tunneling currents [11]. The magnitude of the static power dissipation depends on the manufacturing technology of the semiconductor, its operating temperature and the source voltage range. Beside the static power dissipation, the dynamic power dissipation comes from the switching activities of the logical gates when executing machine instructions. The dynamic power is due to the charge and discharge of the load capacitance Cl, which is roughly proportional to the chip area. The dynamic power dissipation can be expressed by Equation 1:

$$Pd = \alpha \cdot Cl \cdot f \cdot V^2 \qquad (1)$$

where f is the clock frequency, V the supply voltage and α the activity factor (i.e. the fraction of the circuit that is switching). As the dynamic power is quadratically dependent on the supply voltage, any technique lowering the supply voltage will have a substantial impact on the level of dynamic power dissipation. However, lowering the supply voltage implies a increase of the propagation delays between the gates, which require to also decrease the clock frequency. Although lowering the clock frequency has a positive impact on the dynamic power dissipation, it will scale down the performance of the CPU as the machine instruction execution time will be increased. Figure 1, presented in 2003 in [10], shows the evolution of the static and dynamic over the past years. Up to early 2000s, the static power dissipation could be ignored as the dynamic power dissipation was orders of magnitude larger. Although high-k dielectrics were introduced in 2007 with the $45nm$ technologies, it didn't have the radical effect as shown on Figure 1. The 2015 technology roadmap for semiconductor predicts that in the futur the static power dissipation will be particularly difficult to control while scaling up the performance. One of the proposed approaches to tackle the static power dissipation problem is the use of multiple types of transistors on the chip, some providing high performance with high leakage, some providing low performance but with low leakage currents.

A. Thermal influence

The static power dissipation of a chip is directly influenced by its temperature as an increase of temperature will trigger an increase of the leakage currents in transistors. Moreover, this might create a positive feedback loop as by increasing the leaking currents, the temperature will further increases, which in turn will increase the leakage current. The subthreshold

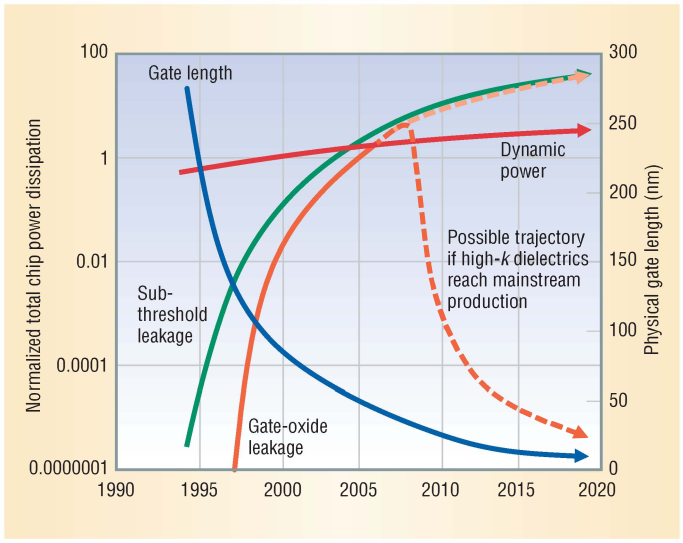


Fig. 1: Static and dynamic power dissipation and the trends [10]

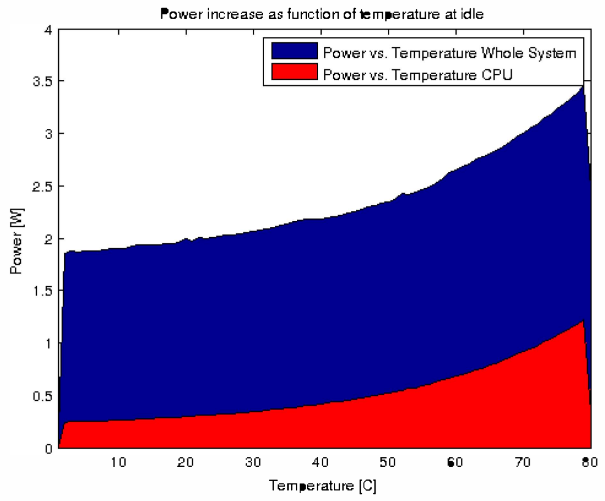


Fig. 2: Influence of temperature on the power dissipation for an ARM processor [12]

leakage current is exponentialy dependent of the thermal voltage V_θ [10], which is in turn proportional to $\frac{kT}{q}$ where k is Boltzmann's constant, q is the electron charge and T the temperature.

The effect of temperature on the leakage currents and the static power dissipation was demonstrated in [12], [13]. Figure 2 shows the power dissipation for an idling ARM CPU, a Quad-Core Exynos 4 implementation of the ARM Cortex-A9 architecture, at fixed clock frequency as a function of ambient temperature. The static power dissipation on the CPU and board levels increases exponentially with the temperature until 80 degrees Celsius. At this temperature, the CPU is automatically switched off as a safety functionality to avoid any physical damage of the semiconductors.

B. Energy consumption

The energy consumed by a processor is defined as the dissipated power integrated over a defined time window as in Equation 2:

$$E = \int_{t_1}^{t_2} P(t) \cdot dt \qquad (2)$$

Therefore, reducing the energy consumption of a computing system is a two variable optimization problem.

This results in a trade-off between performance, i.e number of executed operations over a period of time, and the corresponding average power dissipation over this period.

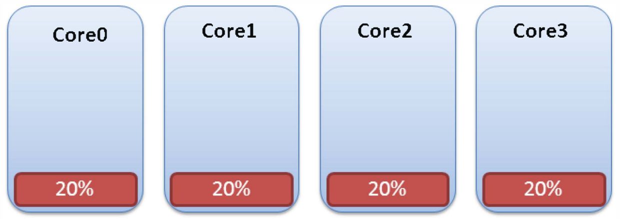
III. Mapping Challenges

The mapping problem is essentially an assignment and scheduling problem. It consists of assigning tasks (software components) on resources and decide on the execution order of the tasks. It exists at different level of resource abstractions, from hardware processing elements [14]–[17] to virtualized distributed computing infrastructures [18]–[21]. In all cases, the mapping is about optimizing conflicting metrics such as performance, cost, energy efficiency, power dissipation, flexibility, memory footprint, etc.

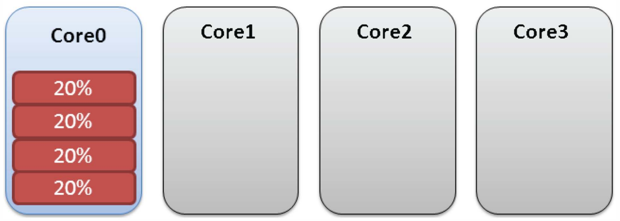
As a consequence of the frequency wall, effort was put in the 2000s on application parallelization to take advantage of the introduction of multicore based processors [22]–[26]. Later from the mid-2000s the power wall and post-Dennard scaling area brought more heterogeneity in hardware processing elements and computing platform. One prominent example is the introduction of the BIG.little architecture by ARM for mobile-class processor. Such architecture offers multiple types of CPU, some providing high performance with high power dissipation, some providing low performance with low power dissipation [27], [28]. Changes in system architecture also occurred in data- center servers with the introduction of a mix of server-, mobile-and graphics-class processors [29]–[31]. The introduction of heterogeneity obviously increased the possibility to get better fits when mapping tasks into more diverse resources [32]. However, it also increases the mapping complexity as the relative characteristics of different type of processing elements need to be taken into account. As example, on a height cores symmetric multiprocessor (SMP) with ten different possible frequency levels, we can define 80 (8×10) different configurations in terms of number of cores and frequency levels. In comparison, a BIG.little processing has four high performance cores and four energy efficient cores. As each type of core has its own clock frequency domain, we can define 1680 $(4 \times 10 + 4 \times 10 + 4 \times 10 \times 4 \times 10)$ possible configurations in terms of number and type of cores and frequency levels.

A. Task Mapping and Workload Consolidation

When mapping a set of task on SMP, the straightforward allocation strategy (also the one used in most operating systems) is to fairly allocate all tasks on all available processing elements. This strategy enables a reduction of the dynamic power dissipation, has it allows a minimal clock frequency since the workload is processed in parallel. However it maximizes the static power dissipation as all processing elements are utilized. The complete opposite strategy would be to map as many tasks on as few cores as possible while not overloading them. With this approach unutilized cores can be shut down to reduce the static power dissipation. Figure 3 illustrates these two opposite strategies when mapping four tasks on a quad-core processor.



(a) Fair allocation of tasks

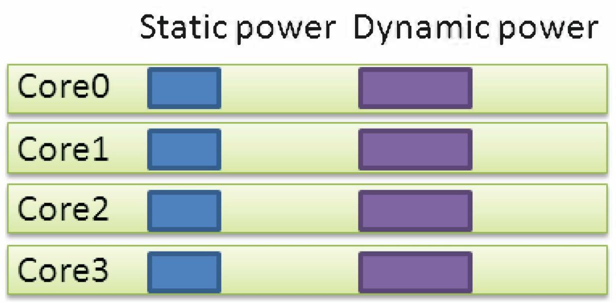


(b) Consolidation of tasks

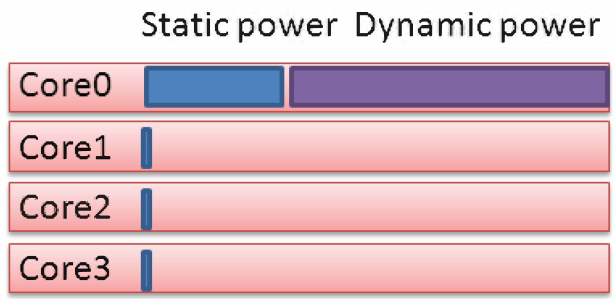
Fig. 3: Opposite task mapping strategies [12]

Figure 4 illustrates the relative differences in terms of static and dynamic power dissipation for the fair allocation (Figure 4a) and the consolidation (Figure 4b) strategies. As for the fair allocation strategy, every core dissipates a small amount of dynamic power since the clock can be set on a low level. At the same time, the cores also dissipate static power since all cores must be enabled to process the workload. The same set of tasks using the consolidation strategy is shown in Figure 4b. Since all four tasks are mapped on only one core, the core must quadruple its frequency to provide the same performance as the fair allocation strategy (Figure 4a). As the frequency increases, the dynamic power increases due to the frequency factor f and the voltage factor V^2 as was shown in Equation 1. Furthermore, when frequency and supply voltage increase the thermal dissipation also increases to form a thermal hotspot. This increases the overall static power dissipation. While the consolidation policy results in high power dissipation for the busy core, the idle cores can be shut off.

The evaluation of the different strategies for a large range of system load levels was performed in [12] on a Odroid-X board equipped with a Quad-Core Exynos processor. This evaluation took into account the thermal influence on the static power dissipation by running experiments at different ambient temperature conditions. Figure 5 presents the efficiency, as the number of operations per second per watt, of the fair allocation and consolidation strategies for different load levels in three different ambient temperatures. From the Figure we can notice that at low load levels more work can be

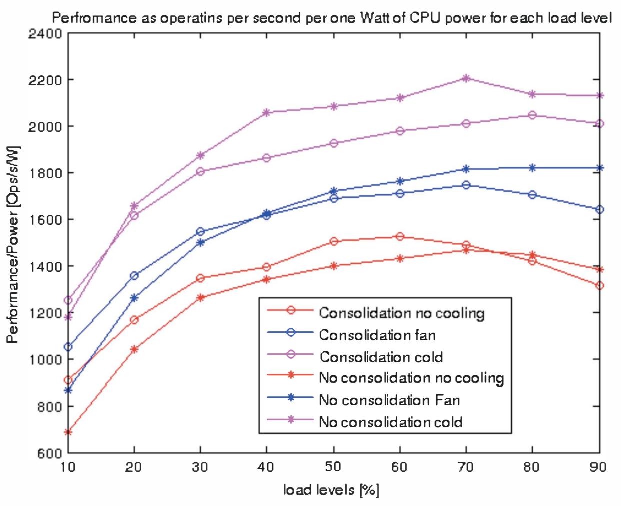


(a) Power dissipation for the fair allocation strategy



(b) Power dissipation for the consolidation strategy

Fig. 4: Power dissipation for opposite mapping strategies [12]



under

Fig. 5: Power dissipation as a function of the system load for different mapping strategies and ambient temperatures

accomplished for the same power budget by consolidating the workload onto a minimal number of cores. When using the consolidation strategy with low load levels, the reduction in static power dissipation is larger than the increase in dynamic power over the remaining cores, resulting in power savings. Furthermore, since the static power consumption is higher at higher temperatures the resulting gain is more apparent with higher ambient temperature. Moreover, it can be observed that when increasing the load level, the efficiency of the consolidation strategy is degrading in comparison with the fair allocation strategy. This results in crossover points between the efficiency of the two approaches. These crossover points depict the point at which the reduced static power and increased dynamic power compensate each other. Even though the gains of using load consolidation strategy on the chip level are considerably less than expected when compared to similar techniques used on the server level [33], [34], the results indicate that consolidation on the chip level can, only in some cases, prove to be a valid approach to improve energy efficiency.

IV. Runtime Systems Challenge

Because software behaviors are typically dynamic at runtime, keeping a required level of performance and efficiency necessitates runtime solutions to dynamically manage resource allocations. Ideally resource allocation should strictly follow what applications requires and should avoid any over- and under-allocation situations.

The most widely used actuator to alter at runtime the resource allocation, and therefore alter the efficiency and performance of applications, is the Dynamic voltage scaling (DVFS) mechanism available on most hardware platforms. Using metrics collected at runtime a manager can adjust the required clock frequency, and as a consequence the supply voltage (see section II), according to application needs. However, most of runtime managers controlling CPU resources, like for example in Linux distributions, use only workload as the metric for resource allocation [35].

Workload is defined in most operating systems, as in Linux, as the ratio between CPU execution and CPU idle states for a given window of time. However it is a arguable metric for controlling CPU resources when energy efficient need to be taken into account. Indeed, the concept of workload does not integrate the notion of application performance. Therefore with such monitored metric, when applying high workload on a CPU, the runtime system will increase the clock frequency as long as the perceived workload remains high. Since applications will execute as long as work is available for execution, the workload will remain high for the entirety application execution. This leads to Race-to-Idle conditions [36], in which the CPU is executing the work as fast as possible before reaching the idling state. The popularity of this execution model relates to simple programming; the programmer specifies only the program functionality, and the operating system scales the clock frequency indirectly according to perceived workload.

When this approach is applied on heterogeneous processor, this leads to unnecessary execution on the large, computationally powerful, cores. Power optimization of DVFS in multi-core systems has been extensively studied in the past [37]–[39]. However, a critical difference between traditional multi-cores and heterogeneous multi-cores, like the ARM big.LITTLE, is the significant energy reduction potential of executing tasks on power efficient cores.

A. *Execution models*

Clock frequency in Linux based systems is driven by a kernel module called frequency governor. The frequency governor is monitoring the workload of the system and adjusts the clock frequency accordingly. A number of different governors can be used on a system, but in most Linux distributions the default governor is the Ondemand [40]. The Ondemand governor monitors and up-threshold value after which the workload is considered too important. As the threshold value is reached, the governor switches the clock frequency automatically to the highest value as illustrated in Figure 6. After the maximum value is reached, the governor decreases the clock frequency step-wise in order to find the most suitable frequency.

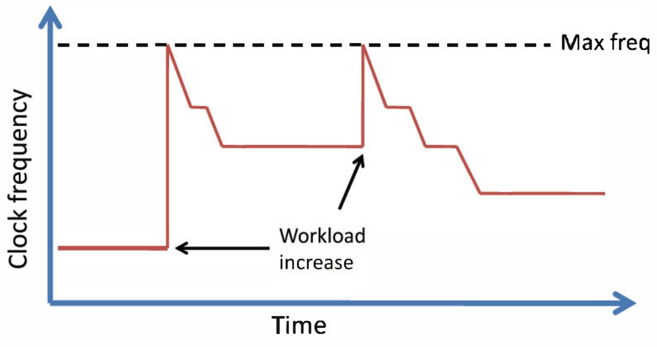


Fig. 6: Behavior of the default ondemand governor

This strategy was designed to rapidly respond to changes in workload without performance penalty, and potentially still save energy by allowing a step-wise scaling down of the frequency and voltage. However, this strategy a) forces the CPU to always execute some part of the workload on the maximum clock frequency and b) for Race-to-Idle conditions, most of the workload will execute on the maximum (or a high) frequency since the workload will remain high as long as jobs are available for execution.

However, on heteregenous multi-cores the notion of workload is not a sufficient metric when energy efficiency is also required. To create a runtime managment system controlled by software requirements, we proposed a framework [41], [42] to inject application specific performance directly into a new type of runtime power manager (further explained in [43]). The power manager monitors the performance of the applications to determine the magnitude of the CPU resource allocation. The proposed power manager supports an execution strategy called QoS-Aware. The strategy is illustrated in Figure 7 (B), in which the execution time can be stretched out. By executing only at the required clock frequency, the power efficient cores on an heterogeneous platform can be utilized as long as the provided performance is sufficient.

B. *Power model*

The QoS power manager uses a power model to determine the increase in power when altering the clock frequency. The

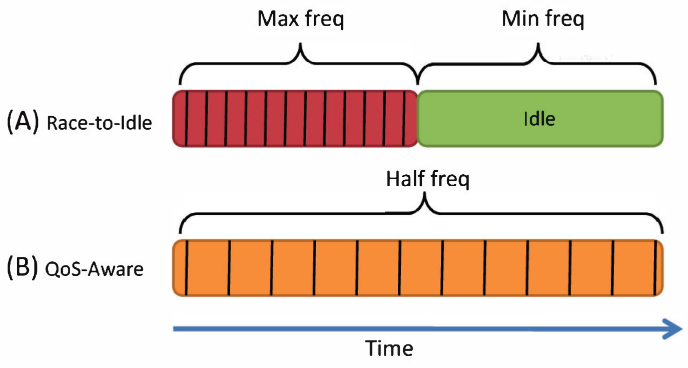


Fig. 7: Illustration of (A) Race-to-Idle strategy and (B) QoS-Aware strategy

performance values provided by the executed applications are compared against a power model in order for the runtime manager to deduct the power output caused by the CPU resource allocation.

As an application demands more resources, the aim of the runtime monitor is to chose a frequency which results in minimum power increase and sufficient performance increase. For experiments, a model was constructed based on a big.LITTLE processor. Such model is constructed by mathematical expressions including architecture based parameters in order for the the runtime manager to account for the core type currently in use. Since a big.LITTLE system with cluster switching was used [44], only one type of core can be active at one time.

Two separate power models for each core type were created based on the stress measurements [43]. Figure 8 (1) shows the LITTLE measurements from 250 MHz to 600 MHz and (2) the big measurements from 800 MHz to 1800 MHz. Because the aim is to keep the system executing on the efficient core as much as possible, the model for the LITTLE cores was overlapped with the lowest frequency of big cores in the LITTLE measurements. This generates a steep cost increase when transitioning from the LITTLE to the big model. Similarly, the highest clock frequency setting (600 MHz) of the LITTLE cores was included in the big-core measurement profile (seen in Figure 8 (2)), which drives the runtime manager to step down to this configuration if the provided performance is sufficient.

These real-world measurements were then transformed [43] into two mathematical functions (as seen on Figure 8 (3)) using plane fitting methods [45]. With traditional non-linear optimization methods [46], we can then at runtime minimize the cost (power in our case) by selecting the optimal clock frequency, number and type of cores for a given application based on performance requirements.

C. *Evaluation*

To evaluate the proposed runtime power aware manager two applications were executed on an Exynos 5410 processor, containing four cortex A15 and four cortex A7 cores. A video decoder application, decoding a 720p video stream, and a

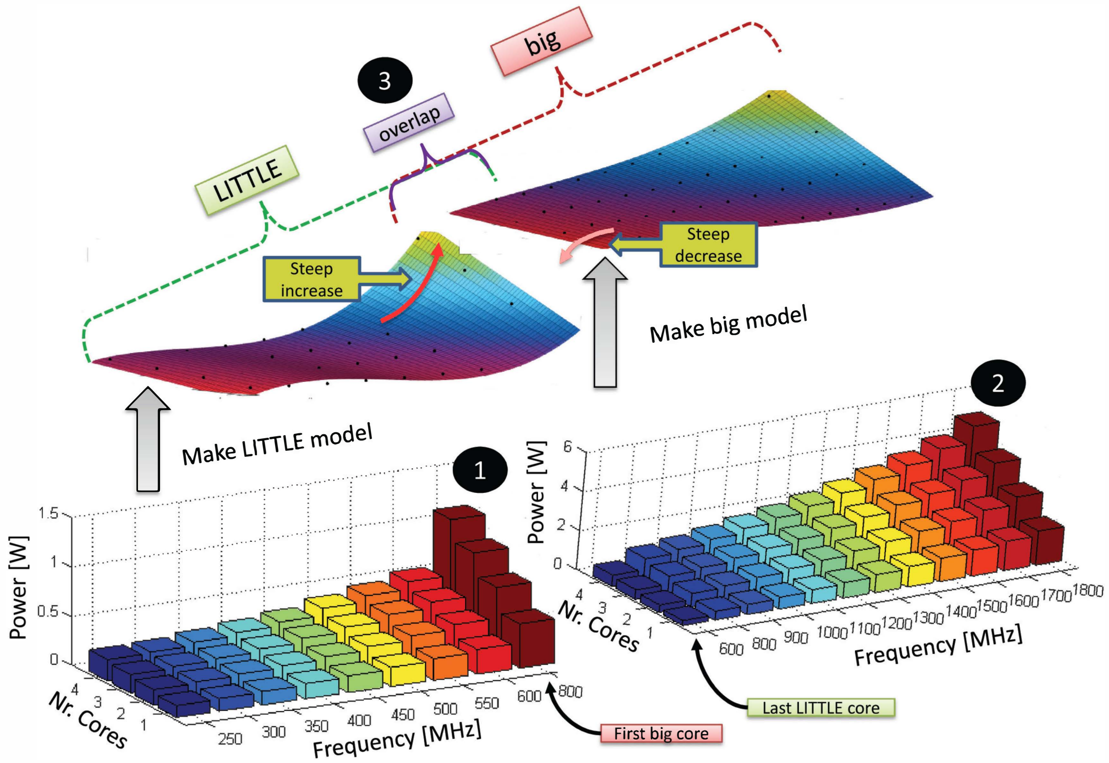


Fig. 8: big.LITTLE power model. Separate reference measurements on the LITTLE and the big cores are used to generate a mathematical model which overlaps in the [600 800] MHz range.

face detection algorithm, detecting faces on the decoded video frames, were used. The targeted QoS for the applications was 30 frames per second for the video decoder, and 10 frame scans per second for the face detection algorithm. Also the face detection algorithm was set on a higher execution priority than the video decoder.

Figure 9 presents the power dissipated by the processor when both applications are executed with (Figure 9 B) and without (Figure 9 A) the runtime manager. When using the default ondemand governer (Figure 9 A), the race-to-idle conditions result in the selection of relatively high clock frequency levels, which in turns results in executing the applications on the big cores. On the other hand, the runtime manager (Figure 9 B) is able to balance the use of both cores by constantly allocating the required resources based on the current application performances and resulting power dissipation.

Table I provides the achieved energy consumption and QoS levels for both approaches. Compared to the default ondemand approach, the runtime power aware manager was able to save about 40% of energy at the cost of a 1% QoS degradation for the video decoder, and 6% QoS degradation for the face detection algorithm.

	Energy	QoS Decoder	QoS Face
Ondemand	334,3	100 (1 drop)	92 (52 late)
Runtime manager	201,5	99 (97 drop)	86 (108 late)

TABLE I: Energy (in Joule) and QoS (in %)

V. Conclusion

Reducing the energy consumption of computing system while maintaining a required performance is a considerable challenge for system designers. Following the post-Dennard scaling area and the emergence of dark silicon, heterogeneous processors appeared on the market as a potential solution against the power wall. While heterogeneous processors dramatically increase the mapping complexity, they allow a more fine grained fit of software components on resources.

We showed why workload alone is not a sufficient metric when trying to run-time manage in an energy efficient ways heterogeneous platforms. When integrating application performance metrics into the run-time control system, along with a power model of the different computing elements, more energy efficient resource allocation decisions can be taken. The proposed analytic approach can easily handle two types of cores and 19 frequency levels. However, if the number of core types found on processor significantly increase in the future, new approach will be required as the size of the resulting possible configuration states will explode.

References

[1] A. Danowitz, K. Kelley, J. Mao, J. P. Stevenson, and M. Horowitz, "Cpu db: Recording microprocessor history," *Queue*, vol. 10, no. 4, pp. 10:10–10:27, Apr. 2012. [Online]. Available: http://doi.acm.org/10.1145/2181796.2181798

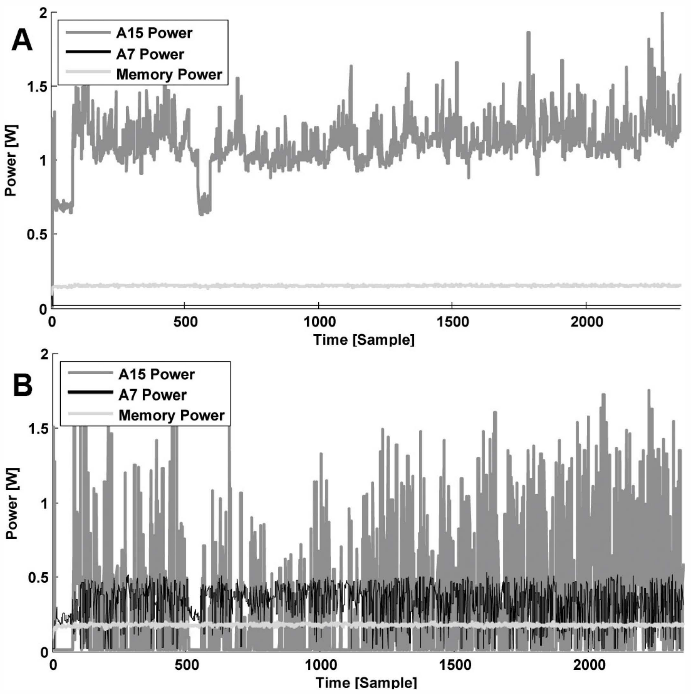


Fig. 9: Power dissipation when execution two application with and without the runtime manager

[2] L. Wang and K. Skadron, "Implications of the power wall: Dim cores and reconfigurable logic," *IEEE Micro*, vol. 33, no. 5, pp. 40–48, 2013.

[3] M. D. Powell and T. N. Vijaykumar, "Resource area dilation to reduce power density in throughput servers," in *Low Power Electronics and Design (ISLPED), 2007 ACM/IEEE International Symposium on*, Aug 2007, pp. 268–273.

[4] R. H. Dennard, F. H. Gaensslen, V. L. Rideout, E. Bassous, and A. R. LeBlanc, "Design of ion-implanted mosfet's with very small physical dimensions," *IEEE Journal of Solid-State Circuits*, vol. 9, no. 5, pp. 256–268, Oct 1974.

[5] P. M. Corcoran and A. Andrae, "Emerging trends in electricity consumption for consumer ict," National University of Ireland, Tech. Rep., 2013.

[6] W. V. Heddeghem, S. Lambert, B. Lannoo, D. Colle, M. Pickavet, and P. Demeester, "Trends in worldwide {ICT} electricity consumption from 2007 to 2012," *Computer Communications*, vol. 50, pp. 64 – 76, 2014, green Networking. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0140366414000619

[7] A. Shehabi, S. J. Smith, D. A. Sartor, R. E. Brown, M. Herrlin, J. G. Koomey, E. R. Masanet, N. Horner, I. L. Azevedo, and W. Lintner, "United states data center energy usage report," Ernest Orlando Lawrence Berkeley National Laboratory, Tech. Rep., 06/2016 2016.

[8] L. A. Barroso and U. Holzle, "The case for energy-proportional computing," *Computer*, vol. 40, no. 12, pp. 33–37, 2007.

[9] L. A. Barroso and U. Hölzle, *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines*, 2009.

[10] N. S. Kim, T. Austin, D. Baauw, T. Mudge, K. Flautner, J. S. Hu, M. J. Irwin, M. Kandemir, and V. Narayanan, "Leakage current: Moore's law meets static power," *Computer*, vol. 36, no. 12, pp. 68–75, Dec 2003.

[11] H. Singh, K. Agarwal, D. Sylvester, and K. J. Nowka, "Enhanced leakage reduction techniques using intermediate strength power gating," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 15, no. 11, pp. 1215–1224, Nov 2007.

[12] F. Hällis, S. Holmbacka, W. Lund, R. Slotte, S. Lafond, and J. Lilius, "Thermal influence on the energy efficiency of workload consolidation in many-core architectures," in *Digital Communications - Green ICT (TIWDC), 2013 24th Tyrrhenian International Workshop on*, Sept 2013, pp. 1–6.

[13] F. J. Mesa-Martinez, E. K. Ardestani, and J. Renau, "Characterizing processor thermal behavior," *SIGPLAN Not.*, vol. 45, no. 3, pp. 193–204, Mar. 2010. [Online]. Available: http://doi.acm.org/10.1145/1735971.1736043

[14] Y.-K. Kwok and I. Ahmad, "Static scheduling algorithms for allocating directed task graphs to multiprocessors," *ACM Comput. Surv.*, vol. 31, no. 4, pp. 406–471, Dec. 1999. [Online]. Available: http://doi.acm.org/10.1145/344588.344618

[15] T. Kempf, M. Doerper, R. Leupers, G. Ascheid, H. Meyr, T. Kogel, and B. Vanthournout, "A modular simulation framework for spatial and temporal task mapping onto multi-processor soc platforms," in *Proceedings of the Conference on Design, Automation and Test in Europe - Volume 2*, ser. DATE '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 876–881. [Online]. Available: http://dx.doi.org/10.1109/DATE.2005.21

[16] D. Grewe and M. F. P. O'Boyle, *A Static Task Partitioning Approach for Heterogeneous Systems Using OpenCL.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 286–305. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-19861-8_16

[17] M. Mandelli, L. Ost, E. Carara, G. Guindani, T. Gouvea, G. Medeiros, and F. G. Moraes, "Energy-aware dynamic task mapping for noc-based mpsocs," in *2011 IEEE International Symposium of Circuits and Systems (ISCAS)*, May 2011, pp. 1676–1679.

[18] F. Teng, "Ressource allocation and schelduling models for cloud computing," Theses, Ecole Centrale Paris, Oct. 2011. [Online]. Available: https://tel.archives-ouvertes.fr/tel-00659303

[19] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," *Future Generation Computer Systems*, vol. 28, no. 5, pp. 755 – 768, 2012, special Section: Energy efficiency in large-scale distributed systems. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X11000689

[20] P. T. Endo, A. V. de Almeida Palhares, N. N. Pereira, G. E. Goncalves, D. Sadok, J. Kelner, B. Melander, and J. E. Mangs, "Resource allocation for distributed cloud: concepts and research challenges," *IEEE Network*, vol. 25, no. 4, pp. 42–46, July 2011.

[21] G. Lee, B.-G. Chun, and H. Katz, "Heterogeneity-aware resource allocation and scheduling in the cloud," in *Proceedings of the 3rd USENIX Conference on Hot Topics in Cloud Computing*, ser. HotCloud'11. Berkeley, CA, USA: USENIX Association, 2011, pp. 4–4. [Online]. Available: http://dl.acm.org/citation.cfm?id=2170444.2170448

[22] J.-Y. Mignolet, R. Baert, T. Ashby, P. Avasare, H.-O. Jang, and J. C. Son, "Mpa: Parallelizing an application onto a multicore platform made easy," *IEEE Micro*, vol. 29, no. 3, pp. 31 – 39, 2009.

[23] U. Bondhugula, M. Baskaran, A. Hartono, S. Krishnamoorthy, J. Ramanujam, A. Rountev, and P. Sadayappan, "Towards effective automatic parallelization for multicore systems," in *Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on*, April 2008, pp. 1–5.

[24] H. Bae, D. Mustafa, J.-W. Lee, Aurangzeb, H. Lin, C. Dave, R. Eigenmann, and S. P. Midkiff, "The cetus source-to-source compiler infrastructure: Overview and evaluation," *International Journal of Parallel Programming*, vol. 41, no. 6, pp. 753–767, 2013. [Online]. Available: http://dx.doi.org/10.1007/s10766-012-0211-z

[25] K. Nishihara, A. Hatabu, and T. Moriyoshi, "Parallelization of h.264 video decoder for embedded multicore processor," in *2008 IEEE International Conference on Multimedia and Expo*, June 2008, pp. 329–332.

[26] V. Pankratius, C. Schaefer, A. Jannesari, and W. F. Tichy, "Software engineering for multicore systems: An experience report," in *Proceedings of the 1st International Workshop on Multicore Software Engineering*, ser. IWMSE '08. New York, NY, USA: ACM, 2008, pp. 53–60. [Online]. Available: http://doi.acm.org/10.1145/1370082.1370096

[27] P. Greenhalgh, "Big. little processing with arm cortex-a15 & cortex-a7," *ARM White paper*, pp. 1–8, 2011.

[28] B. Jeff, "Big.LITTLE system architecture from ARM: saving power through heterogeneous multiprocessing and task context migration," in *DAC*. ACM, 2012, pp. 1143–1146.

[29] M. Guevara, B. Lubin, and B. C. Lee, "Navigating heterogeneous processors with market mechanisms," in *High Performance Computer Architecture (HPCA2013), 2013 IEEE 19th International Symposium on*, Feb 2013, pp. 95–106.

[30] J. Burge, P. Ranganathan, and J. L. Wiener, "Cost-aware scheduling for

heterogeneous enterprise machines (cash'em)," in *2007 IEEE International Conference on Cluster Computing*, Sept 2007, pp. 481–487.

[31] S. P. Crago and J. P. Walters, "Heterogeneous cloud computing: The way forward," *Computer*, vol. 48, no. 1, pp. 59–61, Jan 2015.

[32] R. Kumar, D. M. Tullsen, N. P. Jouppi, and P. Ranganathan, "Heterogeneous chip multiprocessors," *Computer*, vol. 38, no. 11, pp. 32–38, Nov. 2005. [Online]. Available: http://dx.doi.org/10.1109/MC.2005.379

[33] N. Tolia, Z. Wang, M. Marwah, C. Bash, P. Ranganathan, and X. Zhu, "Delivering energy proportionality with non energy-proportional systems: Optimizing the ensemble," in *Proceedings of the 2008 Conference on Power Aware Computing and Systems*, ser. HotPower'08. Berkeley, CA, USA: USENIX Association, 2008, pp. 2–2. [Online]. Available: http://dl.acm.org/citation.cfm?id=1855610.1855612

[34] C. Mastroianni, M. Meo, and G. Papuzzo, "Multi-resource workload consolidation in cloud data centers," in *Utility and Cloud Computing (UCC), 2013 IEEE/ACM 6th International Conference on*, Dec 2013, pp. 297–298.

[35] D. Brodowski, "Cpu frequency and voltage scaling code cpu frequency and voltage scaling code," https://www.kernel.org/ https://www.kernel.org/doc/Documentation/cpu-freq/governors.txt, 2013.

[36] B. Rountree, D. K. Lownenthal, B. R. de Supinski, M. Schulz, V. W. Freeh, and T. Bletsch, "Adagio: Making dvs practical for complex hpc applications," in *Proceedings of the 23rd International Conference on Supercomputing*, ser. ICS '09. New York, NY, USA: ACM, 2009, pp. 460–469. [Online]. Available: http://doi.acm.org/10.1145/1542275.1542340

[37] M. A. Haque, H. Aydin, and D. Zhu, "Energy-aware task replication to manage reliability for periodic real-time applications on multicore platforms," in *Green Computing Conference (IGCC), 2013 International*, June 2013, pp. 1–11.

[38] T. Rauber and G. Rünger, "Energy-aware execution of fork-join-based task parallelism," in *2012 IEEE 20th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, Aug 2012, pp. 231–240.

[39] I. Takouna, W. Dawoud, and C. Meinel, "Accurate mutlicore processor power models for power-aware resource management," in *Dependable, Autonomic and Secure Computing (DASC), 2011 IEEE Ninth International Conference on*, Dec 2011, pp. 419–426.

[40] V. Pallipadi and A. Starikovskiy, "The ondemand governor, past, present and future," in *Proceedings of the Linux Symposium*, 2016.

[41] S. Holmbacka, D. Ågren, S. Lafond, and J. Lilius, "Qos manager for energy efficient many-core operating systems," in *2013 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, Feb 2013, pp. 318–322.

[42] S. Holmbacka, S. Lafond, and J. Lilius, "Performance monitor based power management for big.little platforms," in *Workshop on Energy Efficiency with Heterogeneous Computing*, D. Nikolopoulos and J.-L. Nunez-Yanez, Eds. HiPEAC, 2015, pp. 1 – 6.

[43] S. Holmbacka, E. Nogues, M. Pelcat, S. Lafond, and J. Lilius, "Energy efficiency and performance management of parallel dataflow applications," in *Design and Architectures for Signal and Image Processing (DASIP), 2014 Conference on*, Oct 2014, pp. 1–8.

[44] S. C. Mathieu Poirier, "Heterogeneous multi-processing solution of exynos 5 octa with arm big.little technology," 2013.

[45] C. T. Kelley, *Iterative methods for optimization*, ser. Frontiers in applied mathematics. Philadelphia: SIAM, 1999. [Online]. Available: http://opac.inria.fr/record=b1096699

[46] P. E. Gill, W. Murray, and M. A. Saunders, "Snopt: An sqp algorithm for large-scale constrained optimization," *SIAM J. on Optimization*, vol. 12, no. 4, pp. 979–1006, Apr. 2002. [Online]. Available: http://dx.doi.org/10.1137/S1052623499350013