Thermal Influence on the Energy Efficiency of Workload Consolidation in Many-Core Architectures

Fredric Hällis, Simon Holmbacka, Wictor Lund, Robert Slotte, Sébastien Lafond, Johan Lilius
Department of Information Technologies, Åbo Akademi University
Joukahaisenkatu 3-5 FIN-20520 Turku
Email: firstname.lastname@abo.fi

***Abstract*—Webserver farms and datacenters currently use workload consolidation to match the dynamic workload with the available resources since switching off unused machines has been shown to save energy. The workload is placed on the active servers until the servers are saturated. The idea of workload consolidation can be brought also to chip level by the OS scheduler to pack as much workload to as few cores as possible in a many-core system. In this case all idle cores in the system are placed in a sleep state, and are woken up on-demand. Due to the relationship between static power dissipation and temperature, this paper investigates the thermal influence on the energy efficiency of chip level workload consolidation and its potential impact on the scheduling decisions. This work lay down the foundation for the development of a model for energy efficient OS scheduling for many-core processors taking into account external factors such as ambient and core level temperatures.**

I. Introduction

Energy efficiency is becoming a key issue in all types of computing systems, from hand-held devices to large scale distributed systems. The energy efficiency and proportionality characteristics can be studied on different levels, from the level of enterprise server farms and datacenters to the level of cores in a many-core processor. On the level of datacenters, a mismatch between the energy-efficiency region [1] and the typical processed workload is usually observed. This leads to a non-proportional relationship between the produced work and the corresponding energy consumption i.e. all energy is not used for useful work. Several approaches based on load consolidation [2], [3] were proposed to solve this issue and to accomplish better energy efficiency.

On the level of cores in a many-core processor, the current Advanced Configuration and Power Interface (ACPI) standard defines processor power states, called C-states or sleep states, and performance states called P-states. Using only P-states by exploiting dynamic voltage and frequency scaling (DVFS) mechanisms does not fully solve the power proportionality problem as an idling core still dissipates a non-negligible static power. The overall static power of a chip can be reduced by removing cores from the set of active cores and by using their C-states. Therefore, mapping portions of the workload to appropriate processing elements at any time and by using the processor and performance states influences the power dissipation of the processor. This mapping decision is usually done within the OS scheduler which can make scheduling and load balancing decisions leading to different operating states. As example, fairly distributing all tasks on all cores will disable the possibility to exploit any processor power states but will allow the use of performance states on all cores. On the other hand, consolidating all tasks on as few cores as possible will drastically limit the use of performance states on the remaining active core, but will take full advantage of the power states on the unused cores.

However a side effect of consolidating the workload to few cores is an increase of the temperature of the active cores, while the idle cores, on the other hand, remain cool. In an extreme case, because a hot core dissipates more static leakage power [4], a positive feedback effect of thermal runaway might lead to a continuously increase of temperature and static power dissipation over time. Moreover this factor tend to become more significant as the manufacturing technology decreases [4]. Although chip aging, calculation errors and thermal breakdown are also affected by the temperature, the scope of this paper is only focused on the relation to power dissipation. In this paper we will investigate the issue of energy efficient workload mapping in many-core systems and workload mapping guidelines will be given by investigating and considering:

- The spatial location of workload
- The ambient temperature conditions
- The processor temperature

All measurements and implementations have been obtained from a benchmark running on real hardware and using Linux 3.6.11 as the underlying software platform.

With the given insight into efficient workload mapping, this paper demonstrates that under certain conditions an OS scheduler should not only consider the inherited characteristics of the measured workload, but also account for external factors such as ambient and core level temperatures.

II. Related Work

Several strategies based on load consolidation to improve the system energy efficiency have been proposed in the literature. For example load consolidation is a well studied approach to reduce the energy consumption on the level of cellular access networks [5] and datacenters [2]. Improvements in energy efficiency based on the consolidation of virtual machines on a reduced number of physical servers, as shown by [6], has even resulted in commercially available implementations [7].

Also, on the level of many-core processors techniques to improve the energy efficiency have been proposed. The optimization problem of achieving a minimum energy consumption with the use of load consolidation and performance states on a many-core processor is mathematically formulated in [8]. Previous research has demonstrated the power saving potential of load consolidation in many-core systems [9]–[11]. For example, the behavior of the currently implemented power-awareness feature in the Linux scheduler is discussed in [11]. Without taking into account the thermal state of the cores, this paper demonstrates that the effectiveness of the power saving functionality in the Linux scheduler, described in more details in [12], on a many-core processor is workload dependent. In particular the current Linux scheduler is unable to consolidate load consisting of short running jobs and high rate of process creation and deletion. This paper indicates that the consolidation of tasks to keep as many cores as possible in long idle state is needed in order to reach the most optimal processor power state.

Indeed, idle states can be disturbed by needless interrupts, even when implementing an idle friendly interrupt scheme, such as the dynamic ticks in Linux [11]. To solve this issue, a form of interrupt migration framework to remove all needless interrupts from idling cores is needed [11]. Such a mechanism can be found as part of OS:es, such as the Linux hotplug functionality, which does not only remove interrupts but, depending on the underlying architecture, can shut down an entire core and remove it from the reach of the system scheduler. Migrating interrupts also cost time and it is important that the magnitude of this cost justifies the use of the mechanism [11]. However, since this functionality is capable of removing all activity on a core and placing it in a deep sleep state, it is argued that it can be used as a form of load consolidating power saving measure. By turning off cores at times of low load the remaining load would be consolidated over the remaining active cores.

It is often assumed that consolidating tasks onto fewer cores will result in a trade off between power and performance. However, load consolidation applied in conjunction with DVFS can, in some circumstances, also increase performance [10], [11]. Nevertheless, placing the CPU in a higher performance state, as a result of the load consolidation increasing the load over the active cores, the dynamic power of these cores as well as their heat production will increase, in turn increasing their static power consumption [13]. Since load consolidation can, depending on the load situation, a) either increase or degrade the performance, b) directly affect the dynamic power and c) indirectly, through thermal fluctuations, affect the static power, a prediction of its potential power saving is challenging. The situation is further complicated by the ambient temperature, since it also directly affects the system's leakage power.

Research has shown that load consolidation is in some cases a viable power saving technique on many-core platforms [14]. However, due to different variables such as load perception, interrupt handling, core and ambient temperature, DVFS behavior, and their impact on the overall energy consumption and performance, a best practice regarding load consolidation is yet to be found. This paper discusses the load consolidation challenges on many-core processor in order to discern scenarios in which load consolidation approaches are beneficial.

III. Power Dissipation and Energy Consumption in Microprocessors

A. Power breakdown

The total power dissipated by a processing element origins from two distinct sources: a) the dynamic power dissipation P_d due to the switching activities and b) the static power dissipation P_s mainly due to leakage currents. The dynamic power is dissipated when the load capacitance of the circuit gates is charged and discharged. Such activities occur when the CPU functional units are active. Because the dynamic power is proportional to the square of the supply voltage $P_d \sim Vdd^2$, much effort was put into the design of integrated circuits being able to operate at a low supply voltage. However decreasing the supply voltage of integrated circuits increases propagation delays which force the clock frequency down accordingly. Therefore by dynamically adjusting the clock frequency along the supply voltage when using performance states maximizes the power savings. The dynamic power is given in Eq. 1.

$$P_d = C \cdot f \cdot Vdd^2 \quad (1)$$

Where C is the circuit capacitance, f is the clock frequency and Vdd is the core voltage.

The static power is dissipated due to leakage current through transistors. Moreover, when lowering the supply voltage of integrated circuits, the threshold leakage current increases which also increases the dissipated static power [15], [16]. In addition to this, scaling down the technology process of integrated circuits increases the gate tunneling current which also leads to an increased static power [15]. Until recently, the power dissipated by a processing element was mainly consisting of the switching activities i.e. $P_d \gg P_s$ [17]. However due to technology scaling, the static power dissipation is exponentially increasing and starts to dominate the overall power consumption in microprocessors [4], [15], which leads to increased research efforts in minimizing static power e.g. with the use of sleep states.

B. Thermal influence

The temperature of a microprocessor directly influences the static power dissipation of the chip since the leakage current increases with increased temperature. The rate at which the static power is increased depends on the architecture and manufacturing techniques, and in this paper we mainly focus on mobile many-core processors. In order to determine the temperature-to-power ratio, we let a quad-core processor (ARM based Exynos 4) idle with no workload in different ambient temperatures.

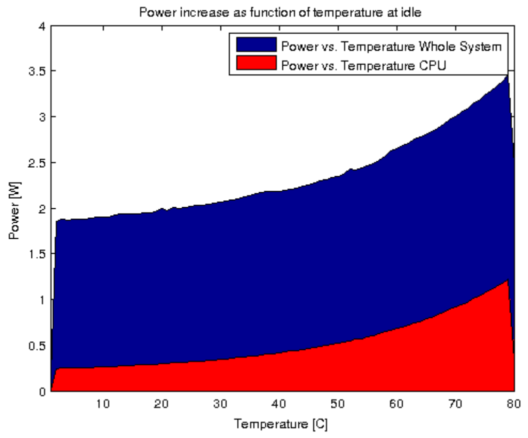

Fig. 1. Static power dissipation as function of ambient temperature for idling chip

Figure 1 shows the increase in static power as a function of the temperature for both board and CPU level measurements. At the left hand side of the curve, the chip was put in a freezer and its internal temperature was measured to be $1\,°C$, and afterwards it was placed in room temperature and heated up to $80\,°C$ with an external heat source. As seen from the figure, the power dissipation of the chip increases more than twofold depending on the ambient temperature conditions. The sudden drop in chip power at the $80\,°C$ point is due to the chip's frequency throttling mechanism, which is automatically activated at this point in order to prevent overheating leading to physical and functional damage.

C. Energy consumption

The amount of energy consumed by a processor is the product of the processors power P_{tot} and the time t as shown in Eq.2

$$E = P_{tot} \cdot t \quad (2)$$

where P_{tot} is the sum of dynamic and static power $P_{tot} = P_d + P_s$. The linear combination of power and time results in a two-variable optimization problem for minimizing the energy consumption. A strategy called race-to-idle [14] primarily used in handheld devices aims to execute work as fast as possible in order to minimize the execution time and save energy. On the other hand, decreasing the clock frequency and supply voltage at the cost of longer execution time might reduce the total energy if the processing elements are not overheated due to hot ambient temperature.

We will therefore investigate the energy consumption of different workload placement policies with respect to both power and execution time.

IV. Workload Mapping Policies and Energy Consumption

The main goal of this paper is to investigate how different workload placement policies affect the dissipated power and execution time and how it impacts the energy consumption. Practically the workload on an OS is defined in a certain work quanta called *process* or *task*. A task executing on a core may *utilize* or *load* a certain percentage of the core's capacity. The methods

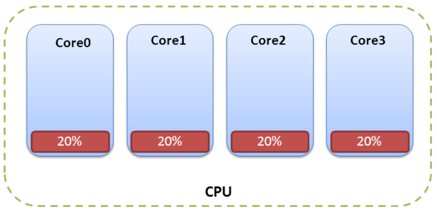

Fig. 2. Workload placement for the evenly balanced policy

for calculating load varies but are usually defined as the ratio between core execution and core idling over a certain time window.

Figure 2 shows the traditional method of task mapping in Linux [18]. Four theoretical CPU-bound tasks, each imposing 20% load, spread out evenly over a quad core CPU to create a balanced schedule.

The complete opposite mapping policy would be to pack as much work onto as few cores as possible.

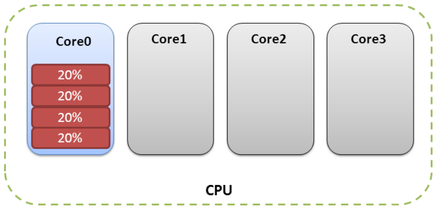

Fig. 3. Workload placement for the packing policy

Figure 3 illustrates a scenario in which four tasks have been mapped on only one core and the remaining cores are turned off. In this case the system must insert the notion of *overloading* a core because as soon as the loaded core is overloaded, a new core must be woken up to offload the overloaded core. In a non-ideal (and more realistic) case the workload is not ideally divisible over the complete platform.

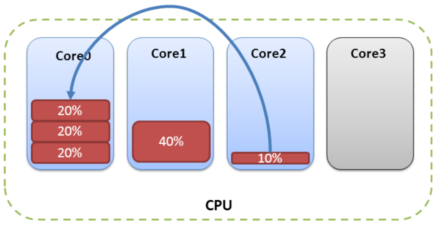

Fig. 4. Workload migration in a non-ideal workload case

Figure 4 shows how the core with the least workload offloads a task to the most loaded (but not overloaded) core, in this case Core 0. Similarly, if a core becomes overloaded, the core offloads a sufficiently large portion of the workload to the most loaded but not overloaded core.

The key issue for these different mapping policies is the power breakdown and thermal gradient of the chip. Consider a case with four tasks, each utilizing a core running at 400 MHz to 100%, and a chip implementing P- and C-states. Figure 5 illustrates the relative differences in terms of static and dynamic power dissipation for the balanced policy (leftmost part) and the packing policy (rightmost part). As for the balanced strategy, every core (in this case a quad-core) dissipates a small amount of dynamic power since the clock is only running at 400 MHz and still the core is not overloaded. 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 packing policy is shown in Figure 5, (rightmost part). Since all four tasks

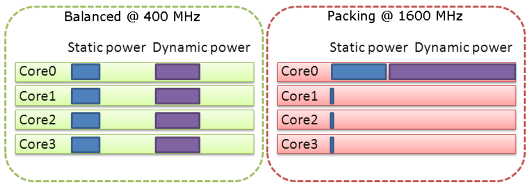

Fig. 5. Power distribution of two workload placement policies

in this case are mapped on only one core, the core must quadruple its frequency in order to not become overloaded. As the frequency increases, the dynamic power increases due to the frequency factor f and the voltage factor Vdd^2 as was shown in Eq. 1. Furthermore, when frequency and supply voltage increase the thermal dissipation also increases to form a thermal hotspot. This increases the static power dissipation because of increased leakage currents in the semiconductors.

While the packing policy results in high power dissipation for the busy core, the idle cores can be shut off and their total power dissipation is in best case zero. The following sections present a set of benchmarks to determine the most energy efficient workload mapping policy with different workload scenarios in different ambient temperatures.

V. Experiment Setup

A. Hardware platform

The hardware platform used for the experiments was an Odroid-X board equipped with a Quad-Core Exynos 4 implementation of the ARM Cortex-A9 architecture. The CPU had a maximum clock frequency of 1.6 GHz and 1 GB of DRAM. The board has 15 P-states corresponding to 15 different clock frequencies and voltage settings. The highest P-state corresponds to a frequency of 200 MHz and each P-state step changes the frequency by 100 MHz. We ran each experiment in three ambient temperature conditions: 1) hot temperature (the board was using only a passive heatsink), 2) normal temperature (an external fan was used), 3) cold temperature (the board was put in a freezer at -20°C). The power was measured with a probe attached to the current feed pins on the ARM cores and the temperature was measured by reading internal registers.

B. Software platform

Linux 3.6.11 was chosen as the underlying software platform because of the possibilities to alter the task mapping with simple scheduling tweaks and already implemented CPU hotplugging capabilities. All comparisons were run with two policies a) spread out the workload as evenly as possible, and b) packing the workload to as few cores as possible without overloading them. The used platform did have capabilities for DVFS tweaking and CPU hotplugging. The used DVFS governor was the default Linux OnDemand, which sets the clock frequency to the appropriate level depending on the measured workload by utilizing the P-states. All OnDemand parameters were constant in all sets of experiments.

C. Spurg Bench

Spurg-Bench [19] is used to generate controlled load levels on many-core processors. This benchmark is designed to test a system with different types of load levels. It consists of a load generator and a runner script able to generate one or more single-threaded load operation schedulable to any core in the system. Between each portion of operations, the benchmark idles for a set amount of time to create the desired workload percentage. This means that the test is able to generate a specific amount of operations to calculate by utilizing the CPU to a certain level.

The chosen operation we have used stresses the CPU and the CPU's floating point unit with floating-point multiplications. The C code for the operation is shown in Listing 1.

```
int operation(){
  int i; double a = 2.0;
  for (i = 0; i < 1000; i++)
    a *= 2.0;
  return 0;
}
```

Listing 1. Example of a operation using the processors floating-point units.

D. Results

Spurg-Bench was initially set to execute 100000 operations for a certain set of load level setpoints: [10, 20, 30, 40, 50, 60, 70, 80, 90]. All the tests were run in the three different ambient temperature conditions with both the consolidation policy and the balanced policy.

Figure 6 presents the performance, as the number of operations per second, per watt of CPU power, during each of the different load level set points for both scheduling policies in three different ambient temperatures. The figure shows that during lower load levels more work can be accomplished, for the same power budget of one watt, by consolidating the workload.

In situations when the load is low, consolidating does not increase dynamic power nor the temperature, and in extension the static power of the remaining cores considerably. This leads to a situation where the reduction in static power is larger than the increase in total power over the remaining cores, resulting in power savings. Furthermore, since the static power consumption is higher at higher temperatures the effect is more discernible in the high temperature case.

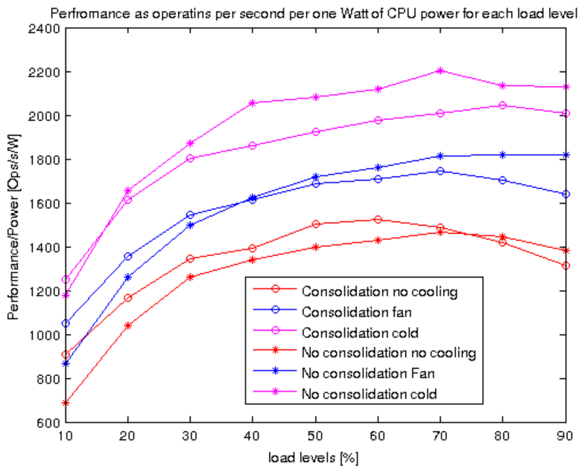

Fig. 6. Power of CPU as a function of operations per second for different ambient temperatures

From the figure it can be seen that as the load increases the performance per watt, for the consolidation policy, degrades in relation to the non consolidation policy. This results in crossover points between the performance per power of the two different policies. These crossover points depict the point at which the trade off between power and performance, when utilizing consolidation, is no longer energy efficient. This behaviour is due to the inherent trade-off between power and performance brought upon by consolidation. Even though the average power drawn by consolidation is lower, at higher loads it takes longer to complete the tasks resulting in decreased performance per watt. The decrease in performance is also partly due to the consolidation policy itself not being as fast at rearranging tasks as the default scheduling policy.

From Figure 6 we notice that the crossover point for the studied CPU with no cooling fan is around load level 70% at 1440 operations per second per watt, and at load level 40% with 1600 operations per seconds per watt if the CPU is cooled down by a fan. In case the CPU is in a freezing environment the crossover point already happens around load level 20% at 1500 operations per second per watt. The different placement of the crossover points are due the static power consumption per core being higher at higher temperatures, which effects the power saving effect of consolidation.

Even though the gains of using load consolidation on the chip level where considerably less than expected, when compared to similar techniques used on the server level [6] [7], the results indicate that consolidation on the chip level can, only in some cases, prove to be a valid measure to improve energy efficiency. Although we expect comparable behaviour on similar types of hardware, we intend to extend our analyses to other architectures as future work. The evaluation of architectures having hardware controlled P- and C- states might produce different results.

VI. Conclusion

This paper has investigated the performance and energy efficiency of a fairly distributed scheduling policy compared to a workload consolidation policy in different ambient temperature conditions. The aim of this work was to determine if under different temperature conditions the static power saved by shutting off idle cores weighs up against the increased dynamic power obtained when increasing the clock frequency and consolidating the workload on the remaining active cores.

The results show that as the workload is fairly distributed over the whole chip, the power and thermal dissipation of the chip remains rather proportional and predictable. However, this is not always the most energy efficient way of scheduling tasks in a many-core system; for low workloads consolidation provides a more energy efficient scheduling policy as the unused cores are completely shut down. Due to the power dissipation of the CPU increasing exponentially as a function of the temperature, the effects of consolidation on energy efficiency is more prominent during higher temperatures. On the other hand, the improved energy efficiency of consolidation degrades as the load increases. Reaching a point where it is more energy efficient to utilize all cores in the system at a slightly higher power dissipation.

We have found that it is difficult to apply a general scheduling policy suitable for any environment – as ambient and chip temperature conditions change, the energy efficiency of scheduling policies varies. Consequently, scheduling decisions should not only be based on internal workload measurement, but should also integrate external conditions such as ambient temperature. On the studied platform, the energy efficiency can be improved by adding a temperature sensitive consolidation policy to a modular scheduler, such as the Linux scheduler. The scheduler can use consolidation during periods of low load where the switch-over point between policies will be dependent on the chip temperature. This would enable energy efficient load balancing mechanisms under variable temperature conditions.

References

[1] L. A. Barroso and U. Hölzle, "The case for energy-proportional computing," *Computer*, vol. 40, pp. 33–37, December 2007. [Online]. Available: http://portal.acm.org/citation.cfm?id=1339817.1339894

[2] C. Mastroianni, M. Meo, and G. Papuzzo, "Analysis of a self-organizing algorithm for energy saving in data centers," in *The Ninth Workshop on High-Performance, Power-Aware Computing*, 2013.

[3] S. Srikantaiah, A. Kansal, and F. Zhao, "Energy aware consolidation for cloud computing," in *Proceedings of the 2008 conference on Power aware computing and systems*, ser. HotPower'08. Berkeley, CA, USA: USENIX Association, 2008, pp. 10–10. [Online]. Available: http://dl.acm.org/citation.cfm?id=1855610.1855620

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

[5] M. Marsan, L. Chiaraviglio, D. Ciullo, and M. Meo, "Optimal energy savings in cellular access networks," in *Communications Workshops, 2009. ICC Workshops 2009. IEEE International Conference on*, 2009, pp. 1–5.

[6] 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 *HotPower '08: Workshop on Power Aware Computing and Systems*. ACM, Dec. 2008.

[7] [Online]. Available: http://www.eco4cloud.com/

[8] M. Ghasemazar, E. Pakbaznia, and M. Pedram, "Minimizing energy consumption of a chip multiprocessor through simultaneous core consolidation and dvfs," in *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*, 2010, pp. 49–52.

[9] J. Hopper, "Reduce Linux power consumption, Part 3: Tuning results," 2009, accessed: September 10, 2012. [Online]. Available: http://www.ibm.com/developerworks/linux/library/l-cpufreq-3/index.html

[10] V. W. Freeh, T. Bletsch, and F. Rawson, "Scaling and packing on a chip multiprocessor," in *Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International*, 2007, pp. 1–8.

[11] V. Srinivasan, G. R. Shenoy, S. Vaddagiri, D. Sarma, and V. Pallipadi, "Energy-Aware Task and Interrupt Management in Linux," vol. 2, Aug. 2008.

[12] V. Pallipadi, "cpuidle - Do nothing, efficiently..." [Online]. Available: http://ols.108.redhat.com/2007/Reprints/pallipadi-Reprint.pdf

[13] V. Jimenez, R. Gioiosa, E. Kursun, F. Cazorla, C.-Y. Cher, A. Buyuktosunoglu, P. Bose, and M. Valero, "Trends and techniques for energy efficient architectures," in *VLSI System on Chip Conference (VLSI-SoC), 2010 18th IEEE/IFIP*, 2010, pp. 276–279.

[14] M. J. Johnson and K. A. Hawick, "Optimising energy management of mobile computing devices,," in *Proc. Int. Conf. on Computer Design (CDES12)*. Las Vegas, USA: WorldComp, 16-19 July 2012, pp. 1–7.

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

[16] S. Borkar, "Design challenges of technology scaling," *Micro, IEEE*, vol. 19, no. 4, pp. 23 –29, jul-aug 1999.

[17] A. Chandrakasan, S. Sheng, and R. Brodersen, "Low-power cmos digital design," *Solid-State Circuits, IEEE Journal of*, vol. 27, no. 4, pp. 473 –484, apr 1992.

[18] M. T. Jones, "Inside the linux scheduler," Jun 2006. [Online]. Available: http://www.ibm.com/developerworks/linux/library/l-scheduler/

[19] W. Lund. Spurg-bench. https://github.com/ESLab/spurg-bench. Åbo Akademi University.