University of Zurich[UZH]

Department of Informatics

Faculty of Economics, Business Administration and Information Technologies

Energy Efficient Programming

An overview of problems, solutions and methodologies

Bachelor Thesis

by

Fethullah Goekkus

Konya, Turkey

07-722-838

University of Zurich

Informatics and Sustainability Research

Prof. Dr. Lorenz Hilty

Supervisor: Dr. Wolfgang Lohmann

Zurich, 01.12.2013

Abstract

There has been a lot of research activity in the field of energy efficiency of ICT systems, which mainly focus on the hardware side, where the software aspects remained relatively unexplored. However software can influence the energy efficiency of hardware significantly, since all hardware is controlled by software. This bachelor thesis provides an overview of the most important problems and proposed solutions regarding the energy efficiency of software and to elucidate the software methodologies and designs that can be used to reduce energy demand of ICT systems, as well as describe diverse tools that assist in development of energy-efficient software.

Zusammenfassung

Softwarentwickler haben grossen Einfluss darauf, wie viel Energie die vorgesehene Hardware zur Ausführung eines Programms benötigt. Im Bereich der Energieeffizienz von IKT-Systemen sind viele Forschungen zu finden, die sich auf Hardwareaspekte konzentrieren. Dagegen blieben die Softwareaspekte dieses Gebiets, trotz grossem Verbesserungspotential, relativ unerforscht. Diese Bachelorarbeit erarbeitet einen ersten Überblick im Sinne einer Landkarte von Problemen und Lösungen in diesem Bereich. Ausserdem werden diverse Methoden für energieeffizientes Programmieren und verschiedene Tools, die Softwareengineers bei der Entwicklung unterstützen, vorgestellt. Im Anhang befindet sich noch eine kommentierte Literaturliste.

Table of Contents

Table of Figures

Table of Tables

1 Introduction

1.1 Context

The energy efficiency in computing, expressed in computations per energy input, has doubled every 1.57 years between 1947 and 2009 [1]. On the other hand the computing performance per personal computer doubled every 1.5 years between 1975 and 2009, so that these early improvements have since then been cancelled out. Additionally the number of installed computers doubled on average every three years between 1980 and 2008. Based on these facts we can observe a rapid increase in the energy consumption of "*Information and Communication Technology*" (ICT) [2].

Soaring energy consumption of ICT has many environmental, technical and economic implications. Increasing energy consumption of today's ICT solutions significantly contributes to "Green House Gas Emissions" (GHGe) leading to accelerate "Global Warming" (GW) [3]. Increased energy usage of ICT generates more heat, which can lead to malfunction of hardware [4]. In order to prevent the heat from affecting the user or the system's electronics, systems require increasingly complex thermal packaging and heat-extraction solutions, adding more costs. According to Capra et al. the current annual power and cooling costs of servers represent almost 60 percent of the servers' initial acquisition cost [5].

There has been a lot of research activity, mainly focusing on the hardware side of ICT systems, whereas the software aspects remained relatively unexplored [6]. However software can influence the energy usage of hardware significantly, since the behavior of hardware is controlled by software. Therefore software perspective of energy efficient computing is an open research field with a lot of potential for improvements [7], [8], [9],

[10].

1.2 Goals and Scope

The purpose of this study is to provide an overview of the most important problems and proposed solutions regarding the energy efficiency of software and to elucidate the software methodologies and designs that can be used to reduce energy demand of ICT systems, as well as describe diverse tools that assist in development of energy-efficient software.

1.3 Structure

After the introduction, the foundations such as metrics and methods are explained in section 2. Section 3 overviews common problems regarding energy efficiency of software and also suggests solutions in order to avoid or minimize these problems. The section 4 introduces some useful tools and technologies that assist in developing energy efficient software. Section 5 concludes the study and gives an outlook for future work. In the section 6 the references are listed and finally section 7 provides an appendix, including an annotated bibliography.

2 Foundations

In this section we will introduce foundations. After we discuss the economic and environmental aspects, we will present metrics and methods to measure energy efficiency of software.

2.1 Cost

According to Winter and Jelschen [11] ICT systems were responsible for %2 of the global energy consumption in 2007, equivalent to the annual production of eight nuclear plants [12]. Especially bigger systems such as supercomputers or data centers consume

a significant amount of energy. According to the IT analysis firm IDC the world wide spending on power management for enterprises was around 40 billion USD in 2009 [13]. Murugesan [14] states that nearly 30 percent of a data center's operating expenses are the energy costs. There are also indirect implications of inefficient software arising from energy wasting. Increased power consumption leads to increased complexity in the design of power supplies to be able to supply sufficient power entailing additional costs. Bigger systems that need a large amount of energy creates a great amount of heat as well, leading to increased cooling costs in order to prevent the heat from affecting ICT systems' behavior [14]. The environmental issues arising from energy wasting cause additional costs as well. According to the Stern Review, written by former World Bank Chief Economist Lord Stern, without action the overall costs and risks of climate change could be the equivalent to losing at least 5 percent of the global gross domestic product (GDP) each year [15].

Despite the fact that a lot of effort has been put in developing more efficient ICT systems, many researches have shown that there is still room for improvements to prevent energy wasting [7], [8], [9], [10], [16], [17]. A significant amount of cost saving can be achieved by enhancing energy efficiency.

2.2 Environmental Aspects

The term "Green ICT" is a widely used term to express environmental-friendly information and communication technologies (ICT). Green ICT is defined as the study and practice of environmentally sustainable IT including designing, manufacturing, using, and disposing of computers, servers, and associated subsystems efficiently and effectively with minimal or no impact on the environment [14]. “Green ICT” emphasizes the necessity for reducing the environmental impacts of ICT systems by reducing their energy usage and their green house gas emissions [18]. In this section we will present the environmental aspects of energy efficiency.

2.2.1 Greenhouse Gas (GHG) Emissions

Energy consumption is closely related to carbon emissions [19]. According to the Kyoto protocol there are 6 major Greenhouse Gasses (GHGs): carbon dioxide (CO_2), methane (CH_4), nitrous oxide (N_2O), sulfur hexafluoride (SF_6), HFCs, and PFCs. When discussing carbon emissions we should consider all of the anthropogenic (GHG) emissions [19]. Figure 1 illustrates Global annual CO_2 and GHG emissions, historic and projected business as usual assuming no significant efforts to reduce emissions [3].

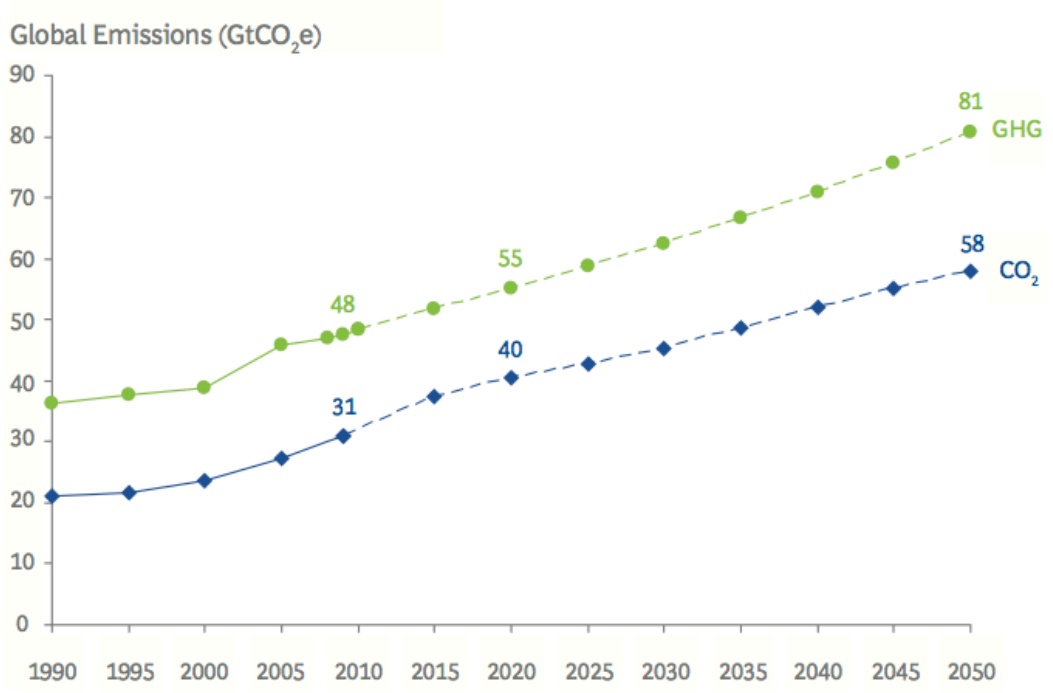

Figure 1: Global Emissions ($GtCO_2e$) [3]

2.2.2 Global Warming

Murugesan [14] states that each PC in use generates about a ton of carbon dioxide every year. The release of carbon dioxide and other GHGs into the atmosphere contributes to global warming. GHGs create a warming affect by absorbing radiation reflected from the Earth that would otherwise escape back into space. Each of major the

GHGs has a different global warming potential (GWP). Table 1 shows the GWPs of major GHGs [19].

Table 1: Global warming potentials of GHGs [19]

GHG	GWP (CO_2e)
CO_2	1
CH_4	25
N_2O	298
SF_6	22800
HFCs	124 - 14800
PFCs	7390 - 12200

These GWPs are measured relative to the GWP of carbon dioxide in CO_2e (CO_2 equivalent). For example, for sulfur hexafluoride, which has a GWP of 22800, an emission with 1 ppmv (parts per million by volume) of methane is equivalent to an emission with 22800 ppmv of carbon dioxide [19].

Global warming has many impacts on the environment. We will briefly outline a few of the negative implications of global warming. Further details can be found in the SMARTer 2020 report of GeSI in 2012 [3].

Cutoff of Thermohaline Circulation (THC)

THC is defined as large-scale ocean circulation that is driven by global density gradients

created by surface heat and freshwater fluxes. Potential changes to the thermohaline circulation are may have significant impact on the Earth's radiation budget. These changes also play an important role in determining the concentration of carbon dioxide in the atmosphere. Warming increases freshwater inflow in the North Atlantic therefore weakening THC and further increasing warming in the North Atlantic Ocean region [3].

Amazon Rainforest Dieback

Global warming disrupts this recycling, causing rainforest dieback, which then leads to further dieback [3].

Permafrost Melting

Atmospheric warming causes permafrost to melt, releasing large amounts of methane and further increasing warming [3].

Disintegration of West Antarctic Ice Sheet

Melting causes ground-line to retreat, potentially leading to ocean water undercutting the ice sheet and more rapid disintegration [3].

Disruption of Indian Monsoon

"*Warming disrupts complex self-sustained moisture circulation of the Indian monsoon*" [3].

Melting of Greenland Ice sheet

Warming leads to melt water residues left on the ice sheet surface, which increases surface temperature and leads to further melting [3].

2.3 Measuring Energy Efficiency

Above all we need metrics and methods to be able to understand, measure, and

compare energy efficiency. Since no standard metrics or methods for determining energy efficiency of software are existent, it is necessary to define appropriate metrics and methods. Defining "metrics" and "methods" requires an understanding of these terms. Hence it is important to investigate these terms further in order to understand them and to be able to distinguish them from each other. In the next two sections the terms "metric" and "method" will be studied in detail.

2.3.1 Metrics

The Oxford dictionary defines metric as: "a system or standard of measurement" [20]. Metric defines a unit and provides a basis to compare two measurements.

In general, efficiency can be defined as:

Efficiency = Useful Work Done / Used Effort

In the context of energy efficiency we can adapt this formula as follows:

Energy Efficiency = Useful Work Done / Used Energy

Typically "useful work done" and "used energy" are observed over a limited time interval as Hilty and Coroama [21] state: "Usually both the output and the energy consumption are related to a period of time, which obviously leads to the elimination of time and yields the dimension "services per energy" such as kByte/Ws for an internet service" [21].

One of the reasons that makes measuring energy efficiency of software complicated is that there are no standard metrics or methods, since the "useful work done" part varies dramatically depending on the software. For example it can be "to sort 100000 double's in an array" or it can also be "to send 100 MB to a host in a network" etc. The unit of efficiency would be "sorted numbers / Ws" or "sent data / Ws" accordingly. Obviously there is an almost unlimited number of possibilities, which makes it difficult or even impossible to set a standard, which fits all kinds of software. This lack of standards

necessitates defining appropriate methods and metrics depending on the particular software or software component. Johann et al. [20] state this the following way: "*Metrics for energy efficient software rely on its useful work done. Since, modern software consists of manifold modules that all have a special purpose, there can be more than just one metric. The software parts can be measured individually or combined. For a proper comparison of software the measured modules should be as similar as possible*" [20]. For example: it would be appropriate to define a "useful work done" for a web browser and compare different web browsers with this metric. Figure 1 illustrates an example of such a comparison.

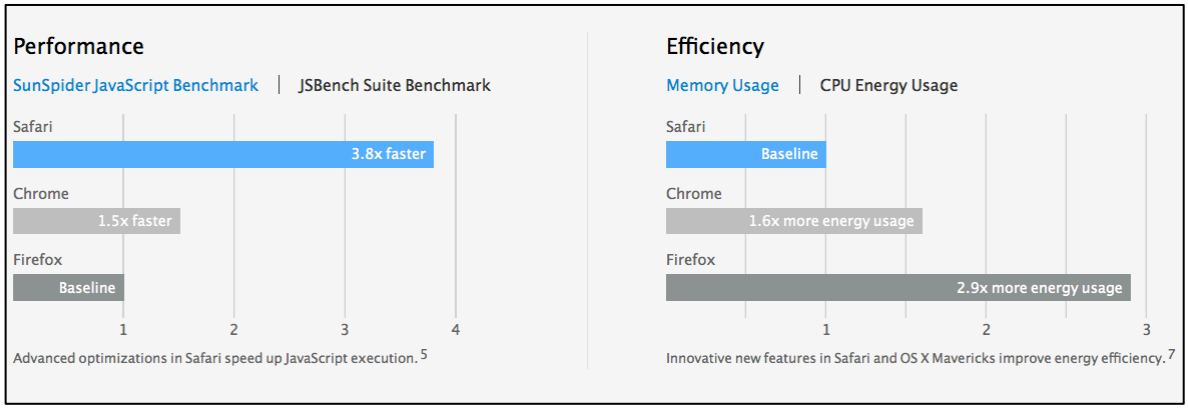

Figure 2: Comparison of web browsers

The following metrics could be useful regarding energy efficiency: [23]

Aspect	**Metric**
Energy Efficiency	Energy / Unit of Work
CPU-Intensity	CPU Cycle Count
Memory Usage	Memory Consumption
Peripheral Intensity	Peripheral Usage Time
Idleness	Idle Time

The Energy-Delay Product

EDP is the product of power consumption (averaged over a switching event) times the input–output delay, or length of the switching event. It has the dimension of energy, and measures the energy consumed per switching event [24]. The advantage of EDP is that it balances energy and performance. Shore [10] described EDP as : "*Although it has neither standard units nor methodology, it combines energy consumption with a measure of performance. Increasing energy use or decreasing performance will increase EDP, so what we seek is the lowest acceptable value of EDP – in other words, the lowest energy use consistent with carrying out the required tasks within the time allowed*."

2.3.2 Methods

According to the Oxford Dictionary [25], a method is "a particular procedure for accomplishing or approaching something, especially a systematic or established one". [25] It is a body of techniques which describe how the measurements are done.

Johann et al. [20] introduced a systematic classification of methods for measuring energy efficiency of software. They state that the three commonly known methods are: "Benchmarking", "Individual measurement", and "Source Code Instrumentation". In the next section these methods will be investigated in more detail.

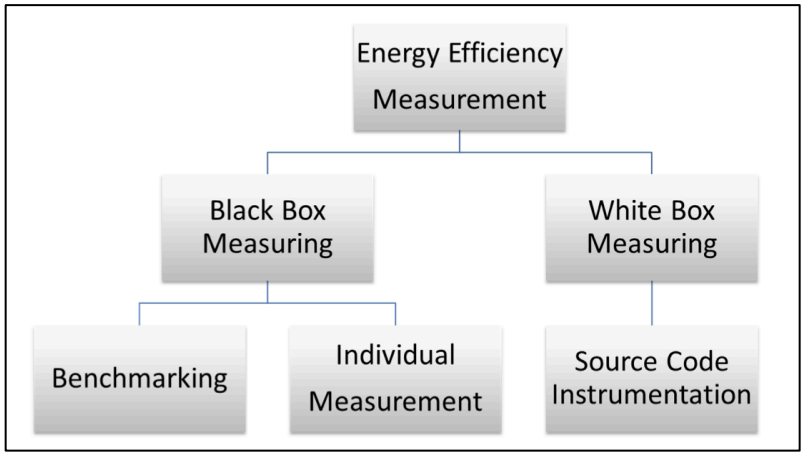

Figure 3: Methods for measuring energy efficiency [20]

2.3.2.1 Black Box Measuring

Black-box measuring treats the software as a "black box", examining efficiency of software as a whole without any knowledge of the efficiency of individual components of the software. Even though it can detect inefficient software by comparing it to others, black box methods cannot identify the actual source of inefficiency. This is why black box methods are not as suitable as the white box methods for detecting inefficiencies and improving them. According to Johann et al. [20] there are two common black box methods: "Benchmarking" and "Individual Measurement".

Benchmarking

According to Johann et al. [20] "Benchmarking methods are able to measure a system as a black box and can generate a statement on how the entire system (software and hardware) performs on the whole. When it comes to measuring a given software, one cannot apply these benchmarks because each of them is customized for one specific group of tasks (database benchmarks, graphic benchmarks, etc.)." Kern et al. [23] list some common Benchmarks: "Common benchmarks for energy efficiency are EnergyBench according micro controller, SpecPower according server hardware or TPC-Energy according databases. Overall, all of the benchmarks produce a standardized workload on a given system and measure the energy consumption simultaneously. Though, just in case of TPC-Energy, a specific metric regarding the energy efficiency of software exists"[23].

Individual Measurement

Another approach is to measure the energy consumption by means of specified use cases and compare different types or configurations of software. In this case, an appropriate use case scenario for the type of software is required. This scenario is applied to the different systems and the energy consumption is measured simultaneously. Based on the results, software systems can be compared regarding their energy efficiency for every use case. This method is a black box measurement, since the

software system is measured and compared as a whole. Certainly, it does not denote an insight into the software itself to point out the reason of the energy consumption or the specific part of the software that is responsible for the results. [22] For example "The approach of Dick et al. [26] defines individual scenarios for a specific group of software (e.g. Browsers) and then measures concrete occurrences with the same scenario, which can give a better statement about the software's energy consumption" [20]. There are many other studies which apply this method such as [23], [27], [28], [29] and more.

2.3.2.2 White Box Measuring

In contrast to the black box method, the white box method takes a look at the code by the way of code instrumentation. This way internal data can be acquired and taken as indicators for creating metrics. According to Johann et al. [20] "A white box method is better suited to find resource intensive parts of programs and to improve them" [20]. White box methods show the energy efficiency of individual components, which allows identifying actual problem zones or in other words the "bottle necks". Therefore white box methods enable efficiency improvements.

Source Code Instrumentation

In the context of computer programming, instrumentation refers to an ability to monitor or measure the level of a product's performance, to diagnose errors, and to write trace information. Programmers implement instrumentation in the form of code instructions that monitor specific components in a system (for example, instructions may output logging information to appear on screen). When an application contains an instrumentation code, it can be managed using a management tool. Instrumentation is necessary to review the performance of the application.

There are very few experiments where this method is applied, Johann et al. [20] introduce two of them. One of them is a "Multi User Web Application", which will be discussed in detail in section 6.2 in the appendix.

3 Energy Efficient Programming Methodologies and Common Problems

To understand the software energy efficiency, methodologies for two kinds of software will be analyzed: application software and system software. For the system software the focus will be particularly on the operating systems. Afterwards problems and solutions for these two types of software will be discussed and some common problems and general methodologies will be introduced.

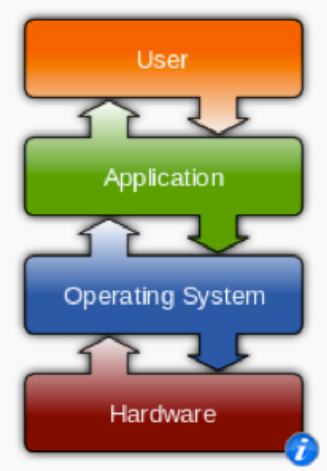

Figure 4: Computer system hierarchy [30]

There are many studies that test a particular method using a specific tool in order to achieve energy efficiency [31], [32], [33], [34], [35], [36]. However, the aim of this study is to provide an overview rather than analyzing a specific solution. Therefore different approaches and concepts for energy efficient programming were analyzed and categorized into the following categories: [8], [19], [26]:

- Application software efficiency
- Operating system optimizations
- Common problems and solutions

3.1 Application Software Efficiency

3.1.1 Computational Efficiency

Computational efficiency can be defined as getting the work done as quickly as possible. Energy efficiency aims at minimizing the energy used to complete a task. The task does not necessarily have to be completed in a shorter time for energy efficiency, however, completing a task faster allows the computer to return to a low-power-state so that more energy can be saved. There are many different approaches to achieve computational efficiency such as [8], [10], [9]:

- Using efficient algorithms and data structures
- Multi Threading
- Efficient use of loops
- Vectorizing and instruction sets
- Efficient use of programming language
- Energy efficient libraries and drivers

Algorithms and Data Structures

Problem: Insensitive choice of algorithms and data structures lead to significant energy wasting.

Algorithms and data structures are a long-standing area of research in computer science. The careful and targeted selection of algorithms and data structures can make a massive difference in the software performance. Energy efficient programming requires using high performance algorithms that complete tasks faster, allowing the processor to idle [8].

Central processor unit (CPU) is a dominant source of power consumption, therefore the optimization of the CPU can lead to significant energy efficiency improvements [7], [9]. Software engineers should investigate efficient solutions for a particular problem in order to exploit this energy saving potential. Following examples show the great impact of algorithm efficiency on the energy usage of software.

Johann et al. illustrate the impact of algorithm efficiency on the energy consumption of software with an experiment. In this experiment, 200 000 double values were sorted first using "*bubble sort*", which runs in $O(n^2)$ time and then the same sorting is done using "*heap sort*", which runs in O(n.log (n)) time. Bubble sort consumed 10800 Joules to complete the task, whereas heap sort used only 7325 Joules [20]. As this simple experiment shows, choosing efficient algorithms can have a massive impact on the energy consumption of ICT systems.

Another example illustrating the importance of algorithm efficiency is presented by Noureddine et al. [28]. In this work, the authors tested the impact of algorithms on energy efficiency with a CPU intensive algorithm to solve the Towers of Hanoi puzzle. This puzzle consists of three rods, and a number of disks of different sizes, which can slide onto any rod. At the beginning all of the disks are in a regular stack in ascending order of size on one rod, being the smallest one on the top. The goal is to move this conical stack to another rod following the 3 rules:

1. Only one disk can be moved at a time
2. Only the disks on top can be moved
3. A disk cannot be placed on top of a smaller disk.

The authors solved this problem first using a recursive algorithm and then they repeated the same procedure with an iterative algorithm. The recursive algorithm implemented in C++ used in average 322.23 joules while the iterative version consumed 1656.26 joules. The iterative solution has consumed approximately 5.14 times more energy than the recursive version. On the other hand, heavily recursive algorithms can be inefficient, as they often add overhead by using or exercising more stack than non-recursive algorithms [8].

In summary, choosing the best algorithm and data structure is a complex decision that depends on many factors. The effort that has been put into choosing the most efficient algorithms can be compensated over time.

Efficient use of loops

Problem: Careless programming of loops and overuse of spinning and polling loops lead to inefficiency

Designing the loops by following some simple rules can lead to important energy savings. For instance, using unsigned integer counters or zero as a termination condition for count down results in faster loops which use fewer registers [10]. Energy efficiency can be improved more drastically by loop unrolling. Loop unrolling is defined as combining the instructions that are called in multiple iterations of the loop into a single iteration. Loop unrolling reduces the comparison and testing overhead associated with the loops [8].

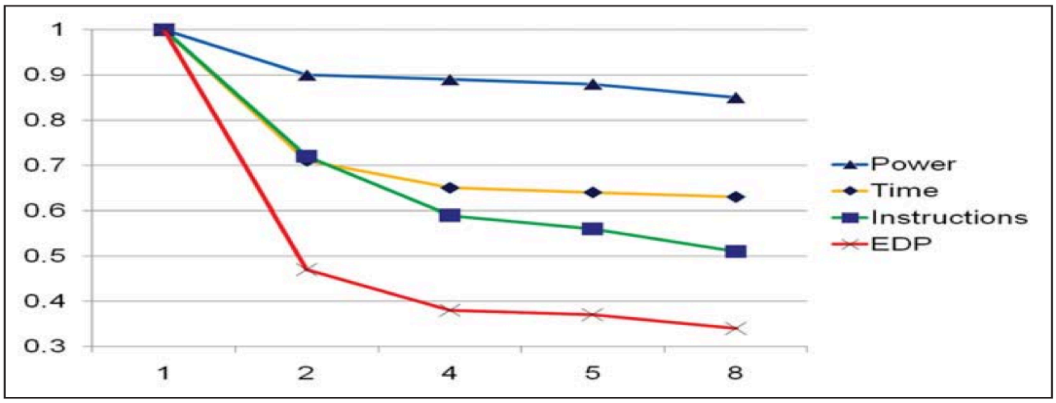

Figure 5: The effect of unrolling factor on the power, execution time and instruction count[10]

Figure 5 shows the effect of the unrolling factor on the execution time, used power, and instruction count. If the unrolling factor increases, power consumption, execution time, and the number of instructions decrease.

Unrolling loops reduces the branch predictor accuracy, since there are fewer branches on which the predictor can train its behavior. However, it also reduces the disruption frequency of the continuous stream of sequential fetches so that, as a combined effect, energy consumption per instruction decreases [10]. More register usage and expanded code size are the possible side effects of loop unrolling [8].

Spinning and polling loops are other potential sources of inefficiency. Spinning loop is a technique in which a process or a thread repeatedly checks to see if a condition is true, such as whether keyboard input or a lock is available [37]. When the thread is in the spinning loop, it remains active without performing a useful task. Using spinning loops can be efficient if threads are locked for short periods of time. This way the overhead from operating system process re-scheduling or context switching can be avoided [7]. Alternatively, blocking a thread is another technique that handles the locks in an event-driven fashion. In this case, a thread changes its state to sleep until some event occurs. There is a tradeoff between the energy saving gained by staying in a lower power state longer and overheads arising from re-scheduling or frequent state transitions. Blocking a thread can be efficient if threads are locked for longer periods of time but this approach can also lead to inefficiency if threads change their states too frequently. Further information can be found in [31], where the both approaches are compared in details.

Multi Threading

Problem: Single threaded applications are inefficient and waste energy

A thread of execution is the smallest sequence of programmed instructions that can be managed independently by an operating system scheduler. The process scheduling can be defined as handling the removal of the running process from the CPU and the selection of another process on the basis of a particular strategy [38].

The implementation of threads and processes depends on the operating system, but usually a thread is contained inside a process. While different processes do not share resources such as memory, multiple threads can exist within the same process and share these resources [39]. Multiprogramming operating systems allow more than one process to be loaded into the executable memory at a time and loaded processes share the CPU using time multiplexing [38].

Multithreading can be implemented by time-division multiplexing on a single processor (see Figure 6). In this case the processor switches between different threads. The

context switching usually occurs frequently enough so that the user perceives the threads or tasks as running at the same time. Many modern ICT systems have multiprocessors or processors with multiple cores. These multiprocessor or multi-core systems can execute threads concurrently, with every processor or core executing a separate thread simultaneously.

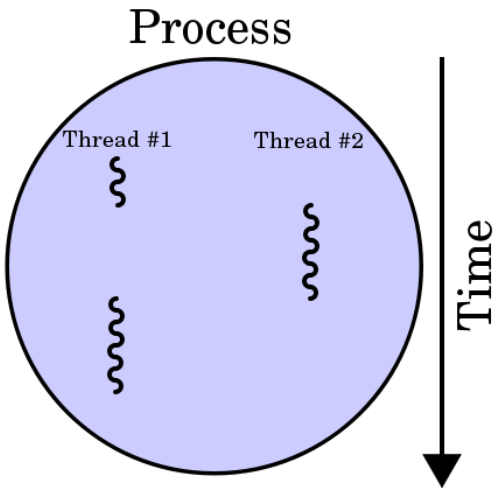

Figure 6: Multithreading with two threads of execution on a single processor [39]

Many modern operating systems allow programmers to manipulate threads via the system call interface. They support both time-sliced and multiprocessor threading with a process scheduler [39]. Using multiple threads and multiple cores delivers better performance than using a single thread [7], [8], [9], [40]. Figure 7 shows the average energy consumption when the same benchmark is executed having different number of threads [9]. It can be seen that a 8-threaded run was completed approximately 4 times faster than the single threaded run and used about 25% less energy [9].

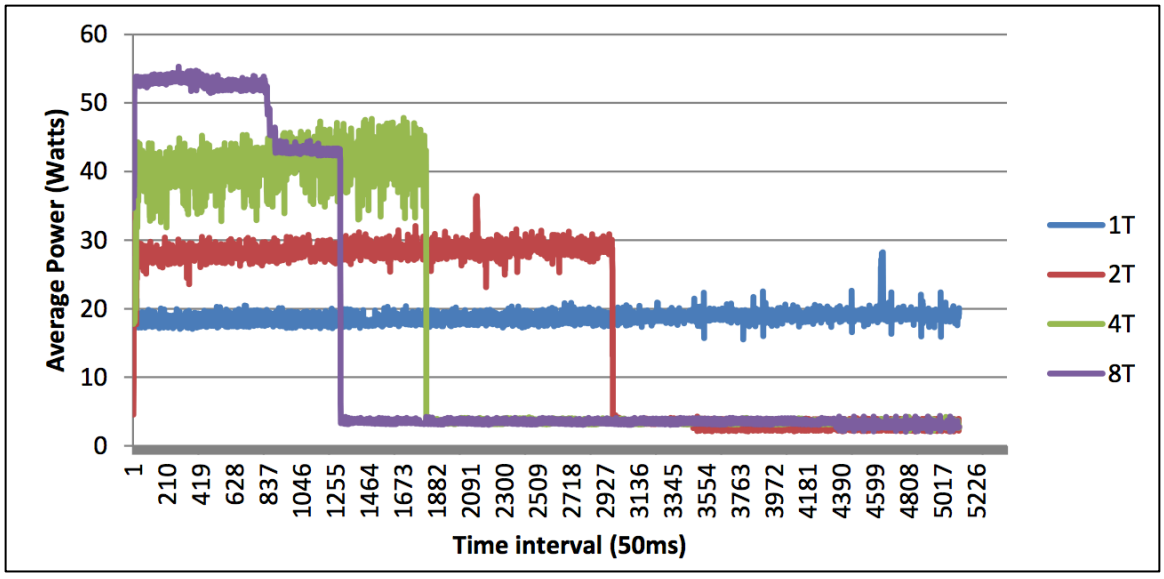

Figure 7: The impact of multithreading on the average energy consumption [9]

Vectorization and Instruction Sets

Problem: Use of Scalar C-code rather than Vectorizing lead to inefficient software that waste energy.

Vectorizing the code instead of scalar C-code using instruction set extensions such as SIMD can help improving performance and energy efficiency. SIMD is the abbreviation for "single instruction, multiple data", and can be described as performing the same action on two or more pieces of data with a single instruction, in order to exploit data level parallelism [9], [41], [42]. There are many useful tools such as "SSE Instructions" and "Intel Advanced Vector Instructions" (Intel AVX) which assist in vectorizing the existing software. More details about these tools can be found in the section 4.

Programming Language

Problem: Choosing an inefficient programming language can lead to significant energy waste.

Programming languages have a great impact on the energy consumption of software, since the energy consumed by the same algorithm varies dramatically depending on the programming language of implementation. Figure 8 shows the energy consumption of the recursive implementation of Tower of Hanoi program in different languages (using a

base 10 logarithmic scale) [28]. According to this experiment the most energy efficient programming language for this program was C++ with O3 optimization consuming only 53.20 joules while the most energy consuming language Perl has spent 25,516 joules. If we compare the both programming languages we can assess that Perl has spent approximately 479.6 times more energy than C++ with O3. These numbers are not representative to make a statement about the energy efficiency of these programming languages in general, since this experiment does not cover all possible CPU utilization levels or all possible workload combinations. However, we can observe that in this particular situation the right choice of the programming language can change the needed amount of energy to complete a task in the most extreme case up to 479 times [28]. Thus, choosing the most efficient programming language is crucial for software energy efficiency.

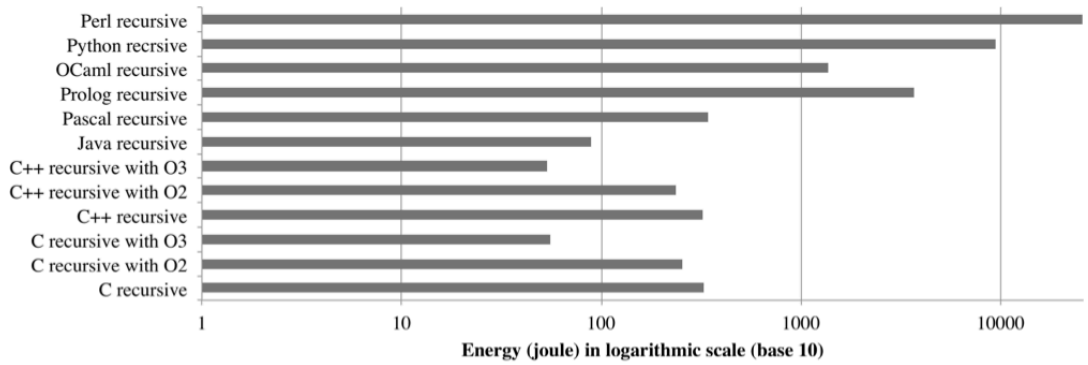

Figure 8: Energy consumption of different programming languages [28]

Energy Efficient Libraries and Drivers

Problem: Not exploiting the well-proven energy efficient solutions can lead to inefficient software.

Libraries contain optimized implementations of common algorithms. Utilizing libraries and drivers, which are optimized for energy consumption, can improve the energy efficiency of an application [9].

3.1.2 Data Efficiency

Fetching data from cache or memories costs both energy and time. Data efficient software minimizes the energy consumption because of data movement through the memory hierarchy and improves the performance of software as well. The two main goals of data efficient programming are moving data over shorter distances and accomplishing tasks with fewest memory accesses [10], [34], [4].

Minimizing Data Movement:

Problem: Unnecessary data movement leads to energy wasting.

The less energy is consumed for a memory access, the closer data is stored to the processing entity. Thus, a data efficient memory hierarchy should store data as close as possible to the processing entity, since the energy consumption of different levels of memory hierarchy varies dramatically. For instance, with the same energy to access external RAM once, the computer can execute 7 instructions or access cache 40 times or access Tightly Coupled Memory (TCM) around 170 times [10], [34].

One approach used to reduce data movement is to buffer and batch data requests into one operation. Buffering the data transferred between memory and typical storage devices such as hard disks and optical disks avoids frequent reads and writes which lets the device to idle [4], [8].

Minimizing Memory Accesses:

Problem: Unnecessary memory access leads to energy wasting.

Algorithms that minimize memory access lead to more efficient software. There are many studies about reducing memory access such as [43]. In this study, Li and Absar propose a method to reduce memory access related power consumption by reducing the number of data transfers between processor and memory, or between a higher (closer to processor) level of memory and a memory at a lower level using source program transformation. They confirm the contribution of their method by illustrating experimental results on a number of benchmarks. Efficient use of memory hierarchy by

moving the most used data to higher levels and having greater caches reduces memory access. On the other hand, bigger cache size causes more energy consumption. Because of the inverse proportional relationship between power and performance, we need to find the optimum. As introduced in section 2.2.1, the optimum has to be identified. Figure 9 illustrates the impact of cache size on the power consumption and EDP [44].

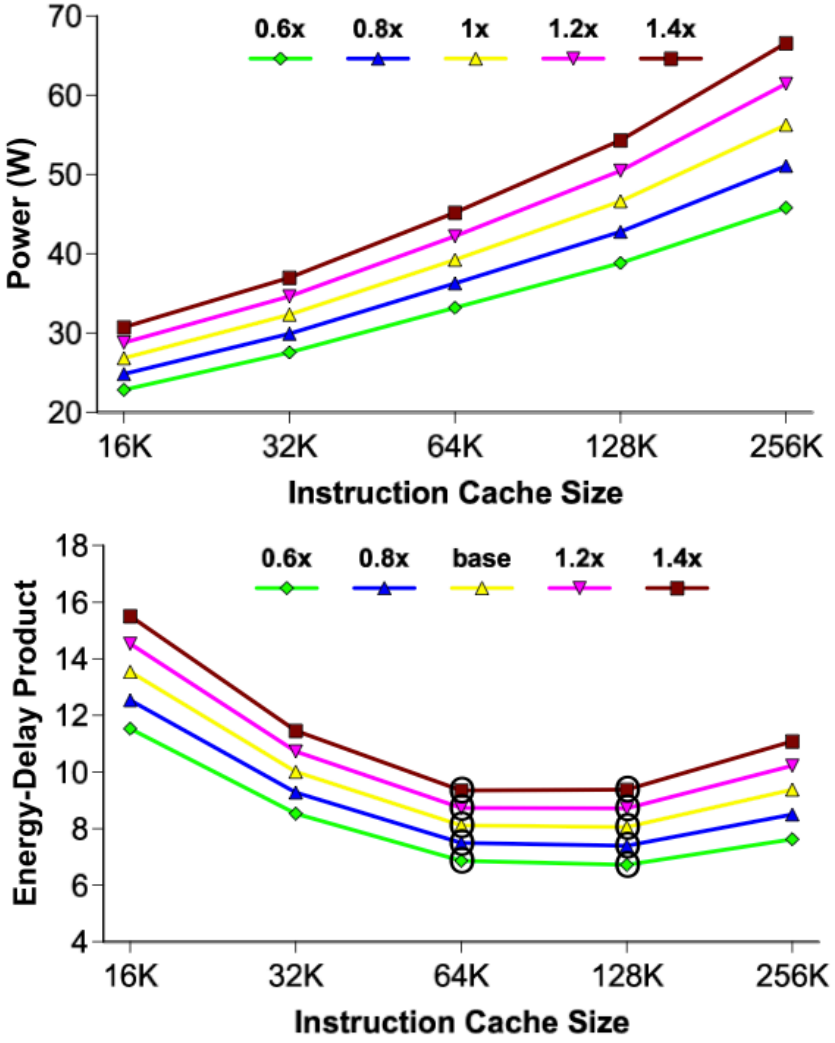

Figure 9: The impact of cache size on the power consumption and EDP [44]

Brooks tested the impact of cache size on power consumption and EDP (Energy-Delay Product) using the two simulators Wattch and PowerTimer [44]. Figure 9 shows that

that the power consumption of the system increases continuously, as the EDP first drops and then increases. In this case, 128k cache size results in the lowest EDP so that it is the most energy efficient variation.

Minimizing energy-delay product has also its disadvantages. EDP provides a balance between the performance and the energy, however it is impossible to guarantee the maximum system performance, and it is also impossible to obtain the maximum energy savings when performance is of no concern [45].

3.2 Operating Systems

Operating systems need to be able to measure or estimate dynamic power consumption, forecast a task's workload and apply some power saving mechanisms in order to reduce the energy usage and increase energy efficiency [46]. "An operating system (OS) is a collection of software that manages computer hardware resources and provides common services for computer programs. For hardware functions such as input and output and memory allocation, the operating system acts as an intermediary between application software and the computer hardware, although the application code is usually executed directly by the hardware and will frequently make a system call to an OS function or be interrupted by it" [47].

Modern operating systems such as Windows 8 have power management policies, which allow configuring the Power Management settings on the managed devices by creating power schemes such as "*plugged in*" or "*battery*" [48]. The main purpose of a power management policy is to decide on the order and timing of component state transitions, based on system history, workload, and performance constraints [49]. In this section, power saving mechanisms and power policies will be discussed.

3.2.1 Power Saving Mechanisms

Problem 1: The existing operating systems can still not exploit the power saving opportunities completely despite many efforts has been put in to operating system power optimizations.

Operating systems should be designed considering the energy consumption and existing operating systems should be revised to improve energy efficiency. There are many power saving mechanisms, which operating systems may apply. These techniques are mostly based on predictions of dynamic power consumption and workload of ICT systems and require compatible hardware, which allow measuring current energy usage and workload. Thus both the accuracy of the estimations and the compatibility of hardware play an important role in achieving good results [50].

The most widely employed power saving techniques regarding operating systems can be listed as [40], [9], [35], [46], [17], [51]:

- Dynamic Voltage and Frequency Scaling (DVFS)
- Advanced Configuration and Power Interface (ACPI)
- Operating System Timer Resolution

Dynamic Voltage and Frequency Scaling

Many modern microprocessors can adjust their clock speed at runtime such as the Intel SpeedStep and the AMD PowerNow [35]. Dynamic frequency scaling (also known as CPU throttling) is a technique in computer architecture whereby the frequency of a microprocessor is adjusted dynamically as the processor burden changes leading to lower power consumption and reducing the heat produced by CPU [52]. Every computer has an internal clock that coordinates all its components and controls the frequency at which instructions are executed. The CPU needs a fixed number of clock ticks (or clock cycles) to execute each instruction. Higher clock rates allow the CPU to execute more instructions in a given time [53]. The clock speed of a CPU can be defined as the

frequency at which processor executes instructions and is measured in megahertz (MHz) or gigahertz (GHz).

Dynamic frequency scaling is often combined with dynamic voltage scaling. Since the energy consumption is proportional to the square of voltage, reducing the voltage when full performance is not needed can lead to significant energy savings [46].

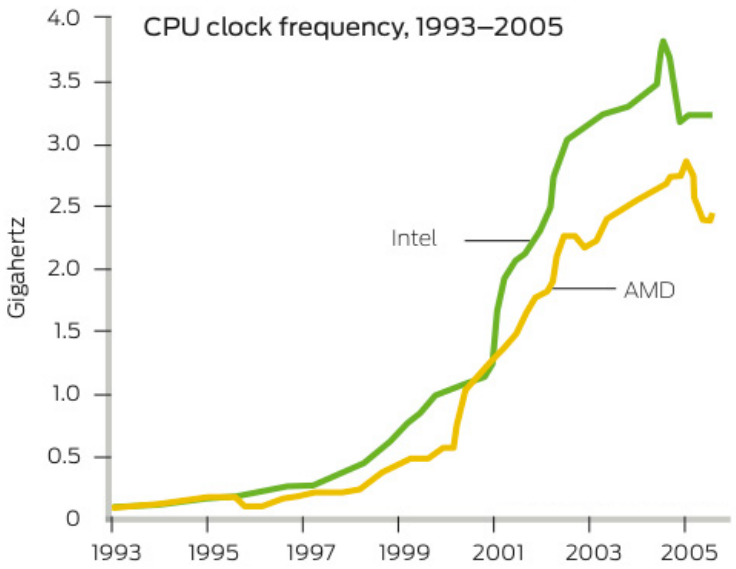

Figure 10: Development of CPU clock frequency between 1993-2005 [54]

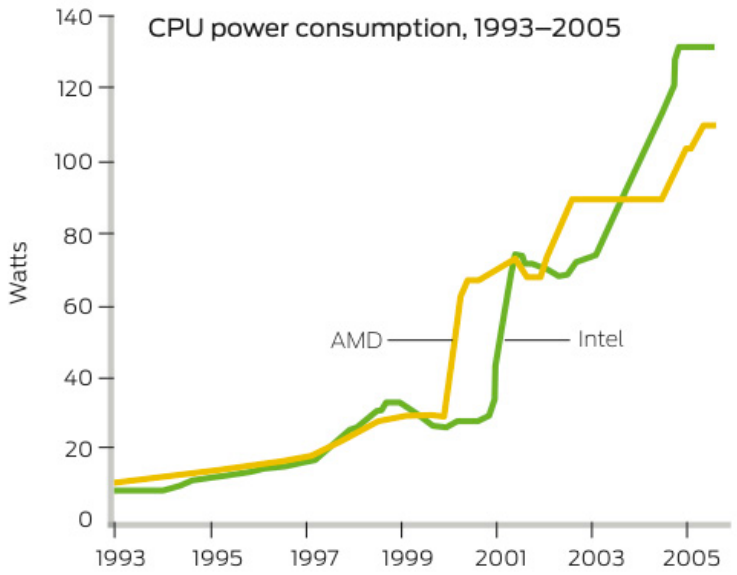

Figure 11: The development of CPU power consumption between 1993-2005 [54]

Figure 10 shows that the CPU clock frequencies of Intel and AMD processors have been increasing constantly between 1993 and 2005. Figure 11 shows that in the same time the power consumption of processors have been increasing continuously as well. When comparing both graphs, a strict parallelism between power consumption and clock frequency of CPUs can be observed. It can be concluded that the frequency of CPU has a great impact on the energy usage; lowering CPU clock frequency when high performance is not needed, can therefore lead to significant energy savings.

Energy efficiency can be achieved only then, when all of the components of the ICT system work together in harmony. For example the capability of dynamic frequency and voltage adaption of a CPU will not improve energy efficiency, if the operating system using the CPU cannot adjust the frequency and voltage at runtime. Thus it is crucial for operating systems to be able to exploit the potential of hardware in order to improve energy efficiency.

On the other hand one should keep in mind that reducing frequency and voltage will also decrease performance and thus not necessarily improve energy efficiency. Higher frequency and voltage provides better performance and can be more effective although

higher clock speed leads to higher energy consumption. Completing a task faster allows the CPU to idle which in turn could lead to greater energy saving. As introduced in section 4.1.2, the EDP balances energy saving and performance and delivers the most efficient trade-off between these two. Thus the goal of a programmer should be to minimize EDP rather than reducing energy usage in exchange for performance loss [55].

Advanced Configuration and Power Interface (ACPI)

ACPI (Advanced Configuration and Power Interface) is a platform-independent power management specification developed by Hewlett-Packard, Intel, Microsoft, Phoenix, and Toshiba in 1996. The purpose of ACPI is to consolidate, check and improve upon existing power and configuration standards for hardware devices. ACPI removes device management responsibilities from legacy firmware interfaces and enables the operating system to control the amount of power given to each device attached to the computer. ACPI allows the operating system to turn off peripheral devices, such as monitors or hard drives after set periods of inactivity [56], [57]. Microsoft's Windows 98 was the first operating-system with full support for ACPI, with Windows 2000, Windows XP, Windows Vista, Windows 7, eComStation, FreeBSD, NetBSD, OpenBSD, HP-UX, OpenVMS, Linux, and PC versions of SunOS all having support for ACPI [56].

ACPI defines four global states (G0-G3), four device dependent device states (D0-D3) and four processor states (C0-C3 or more) and a implementation dependent number of performance states (allowing a maximum of 16 per device), which affect the component’s operational performance while running (P0-P15 for each device). The global states are at the whole system level and have seven sub-states, which are called as sleep states (S0-S6). Some manufacturers may define additional processor states such as Intel's Haswell platform that has eleven states and distinguishes between core states and package states. As the number of a state grows, more precautions are taken to save energy. For example in the S1-state all the processor caches are flushed, and the CPU(s) stop(s) executing instructions. Neither CPU nor RAM are powered off. Devices may be powered off, if they do not indicate that they must remain on. In the S2-state the CPU is

powered off additionally and dirty cache is flushed to RAM [17], [56], [58]. A brief summary of the states can be found in the appendix and all details regarding these states can be found in the ACPI specification [57].

Operating System Timer Resolution

Operating systems use "timer tick" to track the time of day and thread quantum times. The operating system timer resolution determines the frequency of the timer ticks. For example Windows has a standard timer resolution of 15.6 ms, which means that every 15.6 ms the operating system receives a clock interrupt from the system timer hardware. After receiving each interrupt, the timer tick count is updated.

It is crucial for an operating system to optimize its timer resolution. Both too high and too low system timer resolutions can lead to inefficiency. As introduced in section 4.2.1, CPU clock rates have grown constantly and are still increasing. If the operating systems do not adapt their clock rates accordingly, the operating system scheduler can no longer identify processes that do not use too much CPU, because they all use less than a single tick. On the other hand, having operating system timer resolution less than 10 ms can lead to energy-wasting as well. Modern processors and chipsets, particularly in mobile platforms, use the idle time between system timer intervals to decrease system power usage. Many processor and chipset components are switched into low-power idle states between timer intervals. But when the system timer interval is less than 10 ms, changing to a low-power state can be inefficient. Figure 12 shows that the average platform energy consumption constantly grows as the operating system timer resolution increases. Higher timer resolution results in less frequent clock interruptions.

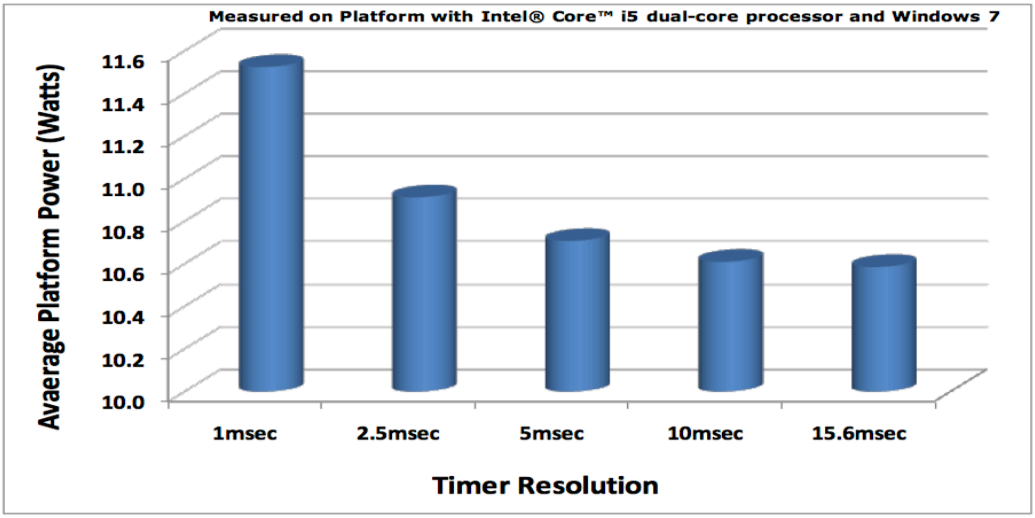

Figure 12: The impact of timer resolution on average platform power [9]

Figure 12 shows that the average platform energy consumption constantly grows as the operating system timer resolution increases. Higher timer resolution results less frequent clock interruptions.

Problem 2: The success of the power saving methodologies depends on the accuracy of estimations and the compatibility of hardware.

Researchers should avoid unrealistic assumptions and perform necessary tests and analysis to confirm these assumptions in order to avoid this problem. To leverage advanced energy efficient computer hardware, the operating environment must be capable of using hardware functionality, and it must deliver advanced performance and power management capabilities for the user and administrator.

Allowing User to Save Energy

Problem: Not allowing user involvement leads to energy wasting.

Operating systems are the intermediate systems between users and the applications. Even if the applications were very energy efficient, there would still be room for improvement by allowing user to contribute to save energy. Energy wasting could be

reduced, if user gets involved consciously trying to save energy. For example the user can easily turn off to computer if it is not needed any more but for the computer it takes certain time and energy to take the same decision based on a long period of inactivity. Thus operating systems should allow user involvement to improve energy efficiency.

3.2.2 Power Policies

In this section we will investigate the power policies in details. After we suggest solutions to the following questions, we will discuss the classification of power policies briefly.

Problem 1: There are a lot of different power policies all claiming to be the best power policy, however "Best Policy" depends on many factors and changes dynamically, as the environment changes. Thus choosing a power policy is a complex decision, which requires great attention.

Problem 2: For general-purpose machines, many different combinations of software applications may be executed at any given time and it is unlikely that experiments will be performed to cover all possible workload mixes [59].

A policy is an algorithm that decides the power states to be used and chooses when to change a component's power states [59]. It has been the main focus of OS-directed power management research to find better policies in recent years. Many power management policies have been proposed, which cover each level of the computing hierarchy[60]. However operating system is the only entity with a global view of resource usage and has detailed information about tasks and their requirements. Therefore power management is one of the main tasks of an operating system [61], [62].

Choosing the best policy is not trivial, unless the policy leads to less energy use, less power and improved performance. In reality reducing energy consumption is often

needs to be balanced with performance based on varying workload and other constraints and there are hundreds of policies to achieve the most efficient trade-off between power and performance. Each of these policies has its advantages and disadvantages. In addition, the policies are evaluated using a single hardware component in most studies. However, it is not clear whether it is appropriate to apply the same policy to a similar hardware component. For example, hard drives and CD-ROM drives are both block devices, which have different workload behaviors. Even if the workload is same, different policies increase energy savings for different devices. For general-purpose machines, many different combinations of software applications can be executed at the same time and it is impossible for an experiment to test all possible combinations of workload [59].

We can conclude that there is no global best policy for all computers. Choosing a policy a priori can be efficient for embedded systems where the workload is relatively homogenous. However, in personal computers, the usage patterns may change dramatically since the user can execute different programs. In this scenario, a group of policies can be eligible at runtime, and one policy can be selected based on the changing request patterns. An example of automatic selection of power policies at run time can be found in [63].

Classification of Power Policies

Power policies can be categorized into 3 major categories: timeout-based, predictive, and stochastic. In this section each of these categories will be explained briefly [59], [64].

Timeout Policies

The timeout policy is the one most widely used in many applications because of its simplicity. Timeout policy allows the user to set the timeout value statically or dynamically. Static timeout policies use a fixed timeout value over time. Many adaptive timeout policies have been proposed to lower wasted idle time by changing the timeout

threshold depending on the history of previous idle periods. [65], [66], [67] Static timeout policies can lead to inefficiencies because they waste power while waiting for the timeout to expire [49], [59], [64].

Predictive Policies

Predictive policies do not shut down the system for a fixed timeout, instead they predict the length of idle periods and shut down the system only when the idle period is long enough to amortize the cost (in latency and power) of shutting down and reactivating the system later [64].

One shortcoming of predictive policies is that they cannot accurately balance performance losses caused by transition delays between states of operation and power savings. Another problem with predictive policies is that they cannot be used in general system models where multiple incoming requests can be queued while waiting for service [49], [64].

Stochastic Policies

Stochastic policies use the Markov model and compute a transition probability matrix for devices. Based on this probability matrix the states can be changed even after a long burst of accesses or before further idleness has occurred. Stochastic policies provide a flexible way to control the trade-off between energy use and performance penalty depending on the optimization constraints.

A shortcoming of the stochastic policies is that they assume the Markov model to be stationary and known in advance. However in reality, the system can experience also non-stationary workloads and in such a case stochastic policies may make wrong decisions for changing the state, which can lead to significant energy wasting [64].

3.2.3 Context Awareness

Problem: Lack of context awareness leads to inefficiency.

Context awareness was first introduced by Schilit in 1994 [68]. Context awareness can be defined as the monitoring of environmental changes and adapting the operations based on these changes. Contextual conditions such as the state of various energy consuming devices (i.e. WiFi and Bluetooth) have great impact on energy consumption of ICT systems [40]. A well-managed context can lead to significant energy savings. For instance some applications may write cached data to flash when the battery is getting critically low or when sensors detect that a notebook PC is falling, the hard drive heads can be parked to prevent a head crash [40], [69].

From the perspective of energy efficiency, context aware computing is the ability to sense the environment in which ICT systems operate and react to these changes in order to reduce energy consumption and improve energy efficiency [9]. In some cases energy consumption can be reduced without improving the energy efficiency. Although the energy efficiency of the ICT system remains the same, such precautions can still be useful as they deliver more positive user experiences.

Improving Energy Efficiency Through Context Awareness

Battery lifetime has become an important issue, especially for rapidly evolving mobile devices with high-performance processors. These devices interact with their environment very often using various sensors. These interactions deliver useful information about the context such as: lighting, noise level, network connectivity, communication costs, communication bandwidth etc. Significant energy savings can be achieved using this context information [8], [9], [68].

As a representative example, scaling the screen brightness automatically based on the lighting of the environment can improve both user experience and energy efficiency. Energy efficiency is defined as "useful work done per energy unit." Useful work for a screen can be defined as showing the content to the user providing good user experience. The definition of "good user experience" clearly varies depending on the user preferences. But for all variations, the required brightness level to maintain clear

sight reduces, as the environment gets darker. This way the screen brightness can be reduced without affecting the user experience and can therefore lead to energy efficiency improvement by preventing energy wasting of too high brightness.

Being aware of the status of peripherals is a further significant aspect of context-aware computing. Application software and operating systems should be able to detect inactivity of the peripherals and turn them off to avoid energy wasting. For instance, if an application exclusively uses Bluetooth on the system, the Bluetooth device can be disabled temporarily in the absence of activity [8].

Improving User Experience through Context Awareness

Most of the context aware computing techniques with regard to energy usage aim at reducing energy consumption. These methods do not necessarily improve energy efficiency but they improve user experience. For example, reducing energy consumption of an application in penalty of performance loss can improve user experience by extending battery lifetime, however, energy efficiency can remain the same [8], [9], [40].

Operating systems and the application software should be aware of the power events such as changes in power source (power adapter vs battery) and adapt their behavior based on these events. For example, if the battery is running low, the operating system should change its state from high performance to power saver. In addition, application software should minimize their power consumption by dismissing the non-critical tasks. Although these adjustments do not necessarily improve energy efficiency, they lead to significantly better user experience.

3.3 General Problems and Solution Proposals

In sections 3.1 and 3.2 specific problems and methodologies to achieve energy efficient software were discussed. In this section general problems, which are not specific for a particular methodology or field, will be discussed.

3.3.1 Lack of Software Engineering Practices for Energy Efficiency

Many researches such as [6], [70], [71] have shown that energy efficiency or other sustainability aspects are not fully supported as a relevant and first-class concern by the traditional software engineering models, despite increased efforts having been put into improving the sustainability of ICT systems [6], [70], [71], [72]. Limiting the necessary precautions for energy efficiency of a software to its programming phase would be a too narrow perspective [2]. The whole life cycle of software should be optimized in order to achieve energy efficiency. In order to do so, energy efficiency should be integrated as a non-functional requirement into the software engineering process models, and the improvements and optimizations should cover the whole life cycle of the software and not just the development phase. There are some studies that suggest life cycle based software engineering models to achieve energy efficiency and sustainability in general. One of the most cited models is the GREENSOFT Model. After defining some basic terms, this model will be outlined briefly.

Green Software and Green Software Engineering

In the sustainable informatics literature the terms "Green Software" and "Green Software Engineering" occur frequently. Dick et al. [73] define green software as follows: "Green and Sustainable Software is software, whose direct and indirect negative impacts on economy, society, human beings, and environment that result from development, deployment, and usage of the software are minimal and/or which has a positive effect on sustainable development" [73]. Green Software Engineering can be defined as the art of defining and developing software products in a way, so that the negative and positive impacts on sustainable development that result and/or are expected to result from the software product over its whole life cycle are continuously assessed, documented, and used for a further optimization of the software product [74].

Energy Efficiency as non-functional Requirement

Non-functional requirements describe the non-behavioral aspects of a system, capturing the properties and constraints under which a system must operate where functional requirements define the intended services, tasks, or functions the system is required to perform [75], [76], [77]. Some examples of non-functional requirements are reliability, ease of use, flexibility, and safety. Based on this definition it can be concluded that energy efficiency is a non-functional requirement, because a lack of efficiency can lead to serious issues such as battery life time problems for mobile devices, or too much heating and too high energy consumption for servers. In order to avoid these issues, the energy efficiency should be accepted as a must-have requirement. On the other hand energy efficiency is not the main purpose or main functionality of software; it is rather a constraint that should be kept while achieving the main purpose like calling someone with a smart phone. As a conclusion it can be stated that energy efficiency is a non-functional requirement.

The GREENSOFT Model

The GREENSOFT Model is a conceptual reference model for green and sustainable software that includes a product life cycle model for software products, sustainability metrics and criteria for software, software engineering extensions for sustainably sound software design and development, as well as appropriate guidance. The GREENSOFT Model aims to support software developers, administrators, and software users in creating, maintaining, and using software in a more sustainable way. The GREENSOFT Model integrates the whole aspects of sustainable development into the software development process rather than improving only the energy efficiency [6], [23]. Figure 13 provides an overview of the GREENSOFT Model.

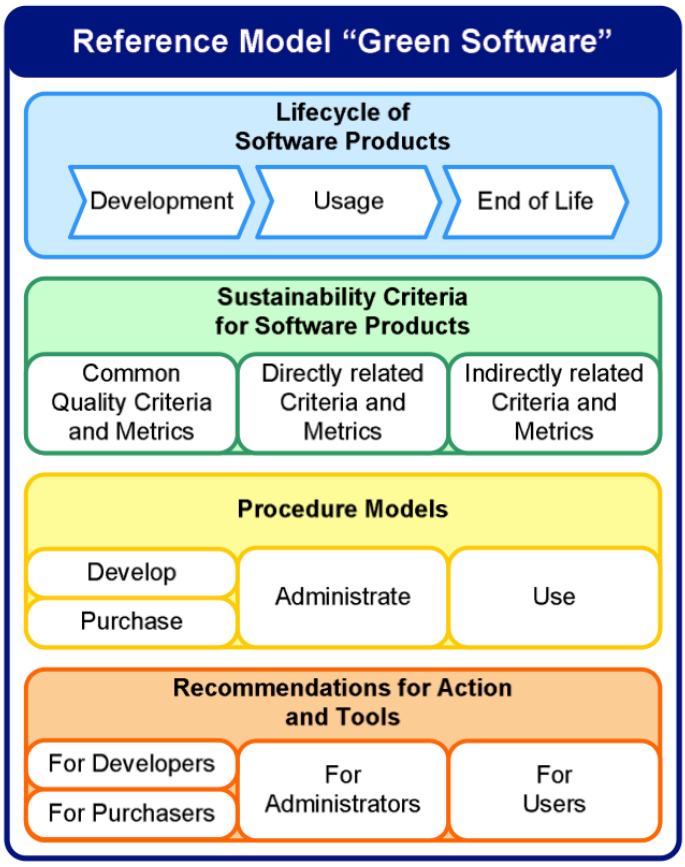

Figure 13: The GREENSOFT Model [6]

As figure 11 illustrates, the GREENSOFT Model has 4 main components: Life cycle of software products, sustainability criteria for software products, procedure models, and recommendations for action and tools.

In contrast to traditional life cycles of software the GREENSOFT Model comprises a "Life Cycle Thinking" (LCT), aiming to assess the ecological, social, human, and economic compatibility of a product during its whole life cycle beginning with the early stages of product development and ending with the product's disposal and recycling [6]. The results of these assessments can be used for product optimizations [78]. The "Sustainable Criteria and Metrics" component of the GREENSOFT Model provides quality models and standardized metrics. Software can be revised using these metrics and quality aspects to improve energy efficiency and sustainability in general [6].

To achieve energy efficient and sustainable software, the whole life cycle of the engineered software product has to be taken into account as well as the circumstances under which it will be produced. The model component "Procedure Models" includes procedure models, based on the different usage types: Developers, purchasers, administrators, and users. Focusing on green and sustainable software engineering, the proposed models can be implemented to support the optimization of the different processes. The models can be adapted to different contexts, since they are general models [6].

The last component of the model contains Recommendations and Tools, which are made available to the different stakeholders. These tools and recommendations such as checklists, guidelines, best practice examples, software tools, as well as other tools (like paper-based data collection sheets) support stakeholders with different professional skill levels in applying green or sustainable techniques in general, when developing, purchasing, administrating, or using software products [23]. Software developers, professional and private users, software acquirers, and administrators are some examples of possible stakeholders [6], [23].

Although there are few researches suggesting green software engineering models, no widely used models that became industry standards exist. As a next step the existing methods should be evaluated by real life projects, and industry and educational institutions should encourage research in the field of “green software engineering”.

3.3.2 The Importance of Software Energy Efficiency is not Perceived Well

Problem: Despite the fact that software can influence the energy consumption of ICT systems dramatically, the potential of software in energy efficiency is still not perceived well.

In the field of energy efficiency of ICT systems, it is tempting for many researchers to focus on hardware, since software does not use energy directly [7]. However all hardware are driven by software and software has great impact on the energy

consumption of ICT systems as well. Despite there is great potential for improvements, the software aspects of energy efficiency of ICT systems remained relatively unexplored compared to hardware [8], [7], [16], [9], [40].

There are responsibilities for educational institutions, research labs and industry in order to solve this problem. Further research in the area of software energy efficiency should be encouraged and energy awareness of all stakeholders of ICT systems such as software developers, professional or amateur users, system administrators, system architects etc. should be improved.

3.4 Summary of the Problems and Solutions

Following the mentioned problems and solutions are summarized in a table.

Table 2: Summary of the problems and solutions: Application software efficiency

Application Software Efficiency	Computational Efficiency	**1. Problem**: Insensitive choice of algorithms and data structures lead to significant energy wasting. **Solution**: C*areful and targeted selection of algorithms and data structures.*
		2. Problem: Careless programming of loops and overuse of spinning and polling loops lead to inefficiency. **Solution**: *Designing the loops by following the rules provided in section 3.1.1* .
		3. Problem: Single threaded applications are inefficient and waste energy. **Solution:** *Multithreading*
		4. Problem: Use of Scalar C-code rather than Vectorizing lead to inefficient software that wastes energy. **Solution:** *Vectorization and instruction sets*
		5. Problem: Choosing an inefficient programming language can lead to huge energy wasting. **Solution: C***areful and targeted selection of programming language.*
		6. Problem: Not exploiting the well-proven energy efficient solutions can lead to inefficient software. **Solution:** *Using energy efficient libraries and drivers.*
	Data Efficiency	**7. Problem:** Unnecessary data movement leads to energy wasting. **Solution:** *Minimizing data movement*
		8. Problem: Unnecessary memory access leads to energy wasting. **Solution:** *Minimizing memory access*

Table 3: Summary of the problems and solutions: Operating system optimizations

Operating System Optimizations	Power Saving Mechanisms	**1. Problem:** The existing operating systems can still not exploit the power saving opportunities completely despite many efforts having been put in operating system power optimizations. **Solution:** *Operating systems should be designed considering the energy consumption and existing operating systems should be revised to improve energy efficiency.*
		2. Problem: The success of the power saving methodologies depends on the accuracy of estimations and the compatibility of hardware. **Solution:** *Researchers should avoid unrealistic assumptions and test the applied methodology and results meticulously.*
		3. Problem: Not allowing user involvement leads to energy wasting. **Solution:** *Involve user in saving energy.*
	Power Policies	**4. Problem:** There are a lot of different power policies all claiming to be the best, however, "Best Policy" depends on many factors and changes dynamically as the environment changes. Thus choosing a power policy is a complex decision, requiring great attention. **Solution:** *Researchers should avoid unrealistic assumptions and test the applied methodology and results meticulously, and power policy should be adapted dynamically.*
		5. Problem: For general-purpose machines, many different combinations of software applications may be executed at any given time and it is unlikely that experiments will be performed to cover all possible workload mixes. [54] **Solution:** *Researchers should avoid unrealistic assumptions and test the applied methodology and results meticulously, and power policy should be adapted dynamically.*
	Context Awareness	**6. Problem:** Lack of context awareness leads to inefficiency. **Solution:** *Context awareness should be improved.*

Table 4: Summary of the problems and solutions: General problems

General Problems	Lack of software engineering practices	**1. Problem:** Traditional software engineering models do not support energy efficiency as a relevant concern. **Solution:** *The whole life cycle of software should be optimized and energy efficiency be integrated as a non- functional requirement into the software engineering process models.*
	Importance of software energy efficiency	**2. Problem:** Despite the fact that software can influence the energy consumption of ICT systems dramatically, the importance of software aspects of energy efficiency is still not perceived well. **Solution:** *Encouraging further research and better education of all stakeholders of an ICT system.*

4 Tools and Technologies

In this section various tools for energy efficient programming will be presented.

4.1 Tools for Data Efficiency

PowerEscape Insight:

PowerEscape Insight aims to reduce power consumption by improving the "data efficiency" of any given C-language code as it is being written [79].

PowerEscape Analyzer:

PowerEscape Analyzer works with any ANSI C-compliant software module compiled using GCC (Gnu C Compiler) 2.95 or higher and produces detailed reports on the memory sub-system's memory access and energy consumption, pinpointing the source code that causes the most costly data transfers. The tool also records the source-code location at which memory use peaks, along with data structures in memory at that time. The Analyzer + Cache tool works with the Analyzer and provides simulation of L1, L2, and L3 caches in a variety of configurations [79].

PowerEscape Architect:

The tool works for power optimization strategies for both hardware and software engineers. It reveals the ideal memory architecture to the system architect while exposing power bottlenecks in embedded code to the software developer [79].

VTune Analyzer:

VTune Analyzer is a programming language and Performance Analyzer tool with a graphical user interface from Intel, which facilitates the application performance tuning without recompilation. The VTune Analyzer supports all compilers that follow industry standards including Microsoft and Intel compilers for C, C++ and Fortran. The VTune Analyzer also supports the most commonly used managed runtime environments such as Microsoft, NET, JAVA, C# and Visual Basic. The VTune Analyzer offers an extensive set of tuning events for all the latest Intel processors. The Intel compilers and the VTune Performance Analyzer can be used together for software optimization [80].

Intel Performance Tuning Utility:

The Intel Performance Tuning Utility (Intel PTU) is a cross-platform performance analysis tool set. Alongside with such traditional features as identifying the hottest modules and functions of the application, tracking call sequences, and identifying performance-critical source code, Intel PTU has new, more powerful capabilities of data collection, analysis, and visualization. For experienced users, Intel PTU offers the processor hardware event counters for in-depth analysis of the memory system performance, architectural tuning, and others. It associates performance issues with the source code. If symbol sources for an analyzed application are not available, Intel PTU represents data with basic block granularity and provides a graph of the function execution flow (control flow graph) to navigate the disassembly. The Intel Performance Tuning Utility is available for both Windows and Linux operating systems [81].

4.2 Tools for Context Awareness

The Microsoft System Event Notification Service (SENS)

The Microsoft System Event Notification Service (SENS) can help alleviate mobile application issues. The SENS API, distributed with the Intel Mobile Platform SDK, provides a simple function call for checking if the network connection is alive and another one pinging a specified address for the user. In addition to these simple functions, the user can also register with the service to receive events when a connection is made or lost, to ping a destination, or even as an alternative method to detect when the system changes power states (battery on, AC on, or battery low).

Intel Laptop Gaming TDK

The Intel Laptop Gaming TDK enables game developers and game engine developers to design and develop laptop-aware games. The Intel Laptop Gaming TDK does this by providing functionalities that dynamically detect changes in system-state information such as power, network and processor. Using this information, game developers can alter game behavior to maximize game play on laptops. Game developers can also use the Intel Laptop Gaming TDK to receive notifications of platform events to achieve maximum game play. In this context, “maximum game play” means utilizing mobility features in a game in a way that reduces power consumption, which in turn allows for maximum battery life while playing the game. The Intel Laptop Gaming TDK also allows game developers to monitor system status, such as remaining battery life, allowing them to alert the game player to the situation [82].

Intel Web APIs

"The Intel Web APIs allow developers to retrieve information about the platform's configuration (e.g., display, storage, and processor) and the platform’s context (e.g., bandwidth, connectivity, power, and location) using JavaScript and HTML [8], [83]."

Koala

Koala is a platform which uses a pre-characterized model at run-time to predict the performance and energy consumption of a piece of software. An arbitrary policy can then be applied in order to dynamically trade performance and energy consumption [50].

4.3 Tools for Measuring Energy Efficiency

Intel Power Checker

Intel Power Checker provides developers a way to evaluate the idle power efficiency of their applications on mobile platforms with Intel Core processor or Intel Atom technology running the Microsoft Windows operating system. Any compiled language application, especially those designed to run on technology based on Intel products and Java Framework applications, can be analyzed by Intel Power Checker. The checker can be used with or without a supported external power meter [84].

Perfmon

The equivalent of System Monitor on Windows 2000, Windows XP, and Windows 7 is called Performance Monitor. Performance Monitor can display information as a graph, a bar chart, or numeric values and can update information using a range of time intervals. The categories of information that you can monitor depend on which networking services are installed on your system, but they always include File System, Kernel, and Memory Manager. Other possible categories include Microsoft Network Client, Microsoft Network Server, and protocol categories [85].

PwrTest/Windows Driver Kit

The power management test tool (PwrTest) enables developers, testers, and system integrators to exercise and record power management information from the system. PwrTest can be use to automate sleep and resume transitions and record processor

power management and battery information from the system over a period of time.

PwrTest (PwrTest.exe) features robust logging and a command-line interface. PwrTest is capable of logging information to both a Microsoft Windows Test Technologies (WTL) and XML file format [86].

Windows Event Viewer/Event Log

Event Viewer is a component of Microsoft's Windows NT line of operating systems and lets administrators and users view the event logs on a local or remote machine. In Windows Vista, Microsoft overhauled the event system.

Event logs are special files that record significant events on a computer, such as when a user logs on to the computer, or when a program encounters an error. Whenever these types of events occur, Windows records the event in an event log that can be read by using Event Viewer. Advanced users might find the details in event logs helpful when troubleshooting problems with Windows and other programs [87].

Windows ETW

The core architecture of Windows ETW has been illustrated in Figure 1. As shown, there are four main types of components in ETW: event providers, controllers, consumers, and event trace sessions. Buffering and logging take place in event tracing sessions, which accept events and create a trace file. There are a number of logging modes available for ETW sessions. For instance, a session can be configured to deliver events directly to consumer applications or to overwrite old events in a file by wrapping around when a certain size is reached. A separate writer thread created for each session flushes them to a file or to real-time consumer applications. To enable high-performance, per-processor buffers are used to eliminate the need for a lock in the logging path [88].

PowerInformer

PowerInformer is a tool developed to provide basic power relevant statistics to a developer who can use these statistics to optimize his application in a way that it

matches battery life constraints (e.g. the game must run for at least 90 minutes on a notebook with a 56 Whr battery) while also meeting performance requirements [89].

PowerTOP

PowerTOP is a Linux tool for diagnosing issues with power consumption and power management. In addition to being a diagnostic tool, PowerTOP also has an interactive mode where you can experiment with various power management settings for cases where the Linux distribution has not enabled those settings.

PowerTOP reports which components in the system are most likely to blame for a higher-than-needed power consumption, ranging from software applications to active components in the system. Detailed screens are available for CPU C and P states, device activity, and software activity [90].

Battery Life Toolkit

Battery Life Tool Kit is a set of scripts and programs to monitor and log power consumption of Linux laptops/notebooks under different workloads.

Battery life measurements require repeatable workloads. While BAPCoR MobileMarkR 2005 is widely used in the industry, it runs only on Windows XP. Intel's Open Source Technology Center has developed a battery life measurement tool-kit to enable reliable battery life measurements on Linux R without special lab equipment [91].

4.4 Tools for Operating System Optimizations

The Homogeneous Architecture for Power Policy Integration (HAPPI)

HAPPI is a software framework that simplifies the implementation, configuration, and automatic selection of power policies. The best power policy depends on a device's power parameters and workload. HAPPI simplifies this configuration process by automatically selecting the appropriate policy for each device, achieving energy

consumption within 4 percent of the best individual policy without knowing the workloads in the beginning [59].

PowerCfg

Using the energy option, powercfg can be used to determine whether an application has increased the platform timer resolution. The PowerCfg utility is run when the application is running and the resulting energy report can be examined to see if the application changed the platform timer resolution. Powercfg will also show the entire call stack for the request. The report lists all of the instances of increased platform timer resolution and indicates whether the process hosting the application increased the timer resolution [9].

Wattch

Wattch is a framework for analyzing and optimizing microprocessor power allocation at the architecture-level. Wattch has the benefit of low-level validation against industry circuits, while opening up power modeling to researchers at abstraction levels above circuits and schematics [92].

PowerAPI

POWERAPI estimates the energy consumption of running processes, in real-time, based on raw information collected from hardware devices (e.g., CPU, network card) through the operating system. [28] Unlike current state-of-the-art technology, PowerAPI does not require any external device to measure energy consumption. This is a purely software approach where the estimation is based on energy analytical models that characterize the consumption of various hardware components (e.g. CPU, memory, disk). PowerAPI is based on a highly modular architecture where each module represents a measurement unit for a specific hardware component [93], [12].

4.5 Instruction Set Extensions

SSE Instructions

SSE is a SIMD extension to the Intel Pentium III and AMD AthlonXP microprocessors. SSE adds a separate register space to the microprocessor. Because of this, SSE can only be used on operating systems supporting it. All versions of Windows since Windows 98 support SSE, as do Linux kernels since 2.2.SSE adds 8 new 128-bit registers, divided into 4 32-bit (single precision) floating point values. These registers are called XMM0-XMM7. An additional control register, MXCSR, is also available to control and check the status of SSE instructions. SSE provides access to 70 new instructions that operate on these 128-bit registers, MMX registers, and sometimes even regular 32-bit registers [94].

Intel Advanced Vector Instructions

Intel® Architecture Instruction Set Extensions Programming Reference includes the definition of Intel® Advanced Vector Extensions 512 (Intel® AVX-512) instructions. These instructions represent a significant leap to 512-bit SIMD support. Programs can pack eight double precision or sixteen single precision floating-point numbers, or eight 64-bit integers, or sixteen 32-bit integers within the 512-bit vectors. This enables processing of twice the number of data elements that AVX/AVX2 can process with a single instruction and four times that of SSE.

Intel AVX-512 instructions are important because they offer higher performance for the most demanding computational tasks. Intel AVX-512 instructions offer the highest degree of compiler support by including an unprecedented level of richness in the design of the instructions. Intel AVX-512 features include 32 vector registers each 512 bits wide, eight dedicated mask registers, 512-bit operations on packed floating point data or packed integer data, embedded rounding controls (override global settings), embedded broadcast, embedded floating-point fault suppression, embedded memory fault suppression, new operations, additional gather/scatter support, high speed math instructions, compact representation of large displacement value, and the ability to

have optional capabilities beyond the foundational capabilities. It is interesting to note that the 32 ZMM registers represent 2K of register space [95].

5 Conclusion and Outlook

We have analyzed several software approaches for energy efficiency and provided an overview of problems and solutions as well as methodologies. Many researches have shown that software can influence the overall energy consumption of ICT systems dramatically. Even very simple precautions can result in great achievements. For example targeted choice of the programming language can reduce the energy consumption up to nearly 430 times, as we presented in section 3.1.1. This great potential for energy savings could not be exploited yet, since the software aspects of energy efficiency remained as a relatively unexplored research field.

In the future we need to understand the role of software in energy efficiency better. Energy efficiency should be integrated in software engineering models as a non-functional requirement and we need to optimize the whole life cycle of software and not just the development phase.

We believe that in the near term, a number of simple methodologies will be used to reduce the most obvious energy waste associated with the highest-power components, such as CPUs. In the future we need to achieve more comprehensive energy optimizations. Further research should be encouraged to explore more general and comprehensive methodologies. Software may not be able to solve all problems with regard to energy wasting however it could lead to significant improvements.

6 References

[1] J. G. Koomey, S. Berard, M. Sanchez, and H. Wong, “Implications of historical trends in the electrical efficiency of computing,” *Ann. Hist. Comput. IEEE*, vol. 33, no. 3, pp. 46–54, 2011.

[2] L. M. Hilty, W. Lohmann, and E. M. Huang, “Sustainability and ICT–An overview of the field,” *Not. Polit.*, vol. 27, no. 104, pp. 13–28, 2011.

[3] GeSI, “SMARTer 2020,” 2012.

[4] P. Ranganathan, “Recipe for efficiency: principles of power-aware computing,” *Commun ACM*, vol. 53, no. 4, pp. 60–67, Apr. 2010.

[5] E. Capra, C. Francalanci, and S. A. Slaughter, “Measuring application software energy efficiency,” *IT Prof.*, vol. 14, no. 2, pp. 54–61, 2012.

[6] S. Naumann, M. Dick, E. Kern, and T. Johann, “The greensoft model: A reference model for green and sustainable software and its engineering,” *Sustain. Comput. Inform. Syst.*, vol. 1, no. 4, pp. 294–304, 2011.

[7] C. Siebra, P. Costa, R. Miranda, F. Q. B. Silva, and A. Santos, “The software perspective for energy-efficient mobile applications development,” in *Proceedings of the 10th International Conference on Advances in Mobile Computing & Multimedia*, New York, NY, USA, 2012, pp. 143–150.

[8] P. Larsson, “Energy-efficient software guidelines,” *Intel Softw. Solut. Group Tech Rep*, 2011.

[9] B. Steigerwald and A. Agrawal, “Developing green software,” *Intel White Pap.*, 2011.

[10] C. Shore, “Developing Power-Efficient Software Systems on ARM Platforms,” *Technol. -Depth*, pp. 48–53, 2009.

[11] A. Winter and J. Jelschen, “Energy-Efficient Applications,” 2012.

[12] “PowerAPI: A Software Library to Monitor the Energy Consumed at the Process-Level.” [Online]. Available: http://ercim-news.ercim.eu/en92/special/powerapi-a-

software-library-to-monitor-the-energy-consumed-at-the-process-level. [Accessed: 29-Nov-2013].

[13] "IDC Home: The premier global market intelligence firm." [Online]. Available: http://www.idc.com/. [Accessed: 01-Dec-2013].

[14] S. Murugesan, "Harnessing green IT: Principles and practices," *IT Prof.*, vol. 10, no. 1, pp. 24–33, 2008.

[15] N. Stern, "Key elements of a global deal on climate change," 2008.

[16] "Winter and Jelschen - 2012 - Energy-Efficient Applications.pdf." .

[17] D. J. Brown and C. Reams, "Toward energy-efficient computing," *Commun ACM*, vol. 53, no. 3, pp. 50–58, Mar. 2010.

[18] "Software Engineering Aspects of Green Computing." [Online]. Available: http://www.green-se.net/segc/2014/. [Accessed: 01-Jul-2013].

[19] W. Vereecken, W. Van Heddeghem, D. Colle, M. Pickavet, and P. Demeester, "Overall ICT footprint and green communication technologies," in *Communications, Control and Signal Processing (ISCCSP), 2010 4th International Symposium on*, 2010, pp. 1–6.

[20] T. Johann, M. Dick, S. Naumann, and E. Kern, "How to measure energy-efficiency of software: Metrics and measurement results," in *Green and Sustainable Software (GREENS), 2012 First International Workshop on*, 2012, pp. 51–54.

[21] L. M. Hilty and V. C. Coroama, *The Role of ICT in Energy Consumption and Energy Efficiency*. 2009.

[22] "Apple - OS X Mavericks - Do even more with new apps and features." [Online]. Available: http://www.apple.com/osx/preview/. [Accessed: 09-Oct-2013].

[23] E. Kern, M. Dick, S. Naumann, A. Guldner, and T. Johann, "Green Software and Green Software Engineering–Definitions, Measurements, and Quality Aspects," *Inf. Commun. Technol.*, p. 87, 2013.

[24] "Power–delay product - Wikipedia, the free encyclopedia." [Online]. Available: http://en.wikipedia.org/wiki/Power%E2%80%93delay_product. [Accessed: 01-Dec-2013].

[25] “method: definition of method in Oxford dictionary (British & World English).” [Online]. Available: http://oxforddictionaries.com/definition/english/method?q=method. [Accessed: 08-Oct-2013].

[26] M. Dick, J. Drangmeister, E. Kern, and S. Naumann, “Green Software Engineering with Agile Methods,” 2013.

[27] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani, “Energy consumption in mobile phones: a measurement study and implications for network applications,” in *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, 2009, pp. 280–293.

[28] A. Noureddine, A. Bourdon, R. Rouvoy, and L. Seinturier, “A preliminary study of the impact of software engineering on GreenIT,” in *Green and Sustainable Software (GREENS), 2012 First International Workshop on*, 2012, pp. 21–27.

[29] E.-Y. Chung, “Software approaches for energy efficient system design,” stanFord university, 2002.

[30] “Operating system - Wikipedia, the free encyclopedia.” [Online]. Available: http://en.wikipedia.org/wiki/Operating_system. [Accessed: 19-Nov-2013].

[31] Y. D. Liu, “Energy-efficient synchronization through program patterns,” in *Green and Sustainable Software (GREENS), 2012 First International Workshop on*, 2012, pp. 35–40.

[32] K. Grosskop and J. Visser, “Identification of Application-level Energy Optimizations,” *Inf. Commun. Technol.*, p. 101, 2013.

[33] A. Sampson, W. Dietl, E. Fortuna, D. Gnanapragasam, L. Ceze, and D. Grossman, “EnerJ: approximate data types for safe and general low-power computation,” *SIGPLAN Not*, vol. 46, no. 6, pp. 164–174, Jun. 2011.

[34] G. Arnout, “Data-Efficient Software and Memory Architectures are Essential for Higher Performance and Lower Power,” *Inf. Q.*, vol. 4, no. 3, 2005.

[35] S. Albers, “Energy-efficient algorithms,” *Commun ACM*, vol. 53, no. 5, pp. 86–96, May 2010.

[36] L. A. Barroso and U. Holzle, “The Case for Energy-Proportional Computing,” *Computer*, vol. 40, no. 12, pp. 33–37, Dec. 2007.

[37] “Busy waiting - Wikipedia, the free encyclopedia.” [Online]. Available: http://en.wikipedia.org/wiki/Busy_waiting. [Accessed: 30-Nov-2013].

[38] “Operating System Process Scheduling.” [Online]. Available: http://www.tutorialspoint.com/operating_system/os_process_scheduling.htm. [Accessed: 30-Nov-2013].

[39] “Thread (computing) - Wikipedia, the free encyclopedia.” [Online]. Available: http://en.wikipedia.org/wiki/Multithreading_(software)#Multithreading. [Accessed: 30-Nov-2013].

[40] B. Steigerwald, R. Chabukswar, K. Krishnan, and J. D. Vega, *Creating energy efficient software*. Technical report, Intel, 2007.

[41] “SIMD - Wikipedia, the free encyclopedia.” [Online]. Available: http://en.wikipedia.org/wiki/SIMD. [Accessed: 04-Nov-2013].

[42] “What is SIMD? - A Word Definition From the Webopedia Computer Dictionary.” [Online]. Available: http://www.webopedia.com/TERM/S/SIMD.html. [Accessed: 04-Nov-2013].

[43] S. Li and M. J. Absar, “Minimizing Memory Access By Improving Register Usage Through High-level Transformations,” 1997.

[44] D. Brooks, P. Bose, and M. Martonosi, “Power-performance simulation: design and validation strategies,” *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 31, no. 4, pp. 13–18, 2004.

[45] D. Snowdon, “Operating System Directed Power Management,” The University of New South Wales, 2010.

[46] C. Lang, “Components for Energy-Efficient Operating Systems,” 2013.

[47] W. Stallings, *Operating Systems: Internals and Design Principles*, 7th ed. Upper Saddle River, NJ, USA: Prentice Hall Press, 2012.

[48] “Novell Doc: ZENworks 11 Configuration Policies Reference - Power Management Policy.” [Online]. Available:

http://www.novell.com/documentation/zenworks11/zen11_cm_policies/data/br5knmk.html. [Accessed: 20-Nov-2013].

[49] L. Benini, A. Bogliolo, G. A. Paleologo, and G. De Micheli, "Policy optimization for dynamic power management," *Comput.-Aided Des. Integr. Circuits Syst. IEEE Trans. On*, vol. 18, no. 6, pp. 813–833, 1999.

[50] D. C. Snowdon, E. Le Sueur, S. M. Petters, and G. Heiser, "Koala: A platform for OS-level power management," in *Proceedings of the 4th ACM European conference on Computer systems*, 2009, pp. 289–302.

[51] A. Govindasamy and S. Joseph K, "Optimization of Operating Systems towards Green Computing," *Int. J. Comb. Optim. Probl. Inform.*, vol. 2, no. 3, pp. 39–51, 2011.

[52] "Dynamic frequency scaling - Wikipedia, the free encyclopedia." [Online]. Available: http://en.wikipedia.org/wiki/Dynamic_frequency_scaling. [Accessed: 23-Nov-2013].

[53] "What is clock speed? - A Word Definition From the Webopedia Computer Dictionary." [Online]. Available: http://www.webopedia.com/TERM/C/clock_speed.html. [Accessed: 23-Nov-2013].

[54] "Why CPU Frequency Stalled - IEEE Spectrum." [Online]. Available: http://spectrum.ieee.org/computing/hardware/why-cpu-frequency-stalled. [Accessed: 24-Nov-2013].

[55] E. Le Sueur and G. Heiser, "Dynamic voltage and frequency scaling: The laws of diminishing returns," in *Proceedings of the 2010 international conference on Power aware computing and systems*, 2010, pp. 1–8.

[56] "Advanced Configuration and Power Interface - Wikipedia, the free encyclopedia." [Online]. Available: http://en.wikipedia.org/wiki/Advanced_Configuration_and_Power_Interface. [Accessed: 21-Nov-2013].

[57] "ACPI - Advanced Configuration and Power Interface." [Online]. Available: http://www.acpi.info/. [Accessed: 21-Nov-2013].

[58] “AnandTech | Intel’s Haswell Architecture Analyzed: Building a New PC and a New Intel.” [Online]. Available: http://www.anandtech.com/show/6355/intels-haswell-architecture/3. [Accessed: 24-Nov-2013].

[59] N. (Eddie) Pettis and Y.-H. Lu, “A Homogeneous Architecture for Power Policy Integration in Operating Systems,” *IEEE Trans. Comput.*, vol. 58, no. 7, pp. 945–955, Jul. 2009.

[60] L. Benini and G. de Micheli, “System-level power optimization: techniques and tools,” *ACM Trans. Des. Autom. Electron. Syst. TODAES*, vol. 5, no. 2, pp. 115–192, 2000.

[61] D. Grunwald, C. B. Morrey III, P. Levis, M. Neufeld, and K. I. Farkas, “Policies for dynamic clock scheduling,” in *Proceedings of the 4th conference on Symposium on Operating System Design & Implementation-Volume 4*, 2000, pp. 6–6.

[62] Y.-H. Lu, L. Benini, and G. De Micheli, “Operating-system directed power reduction,” in *Proceedings of the 2000 international symposium on Low power electronics and design*, 2000, pp. 37–42.

[63] N. Pettis, J. Ridenour, and Y.-H. Lu, “Automatic run-time selection of power policies for operating systems,” in *Design, Automation and Test in Europe, 2006. DATE’06. Proceedings*, 2006, vol. 1, pp. 1–6.

[64] E.-Y. Chung, L. Benini, A. Bogliolo, Y.-H. Lu, and G. De Micheli, “Dynamic power management for nonstationary service requests,” *Comput. IEEE Trans. On*, vol. 51, no. 11, pp. 1345–1361, 2002.

[65] R. Golding, P. Bosch, and J. Wilkes, “Idleness is not sloth,” in *USENIX Winter*, 1995, pp. 201–212.

[66] D. P. Helmbold, D. D. Long, T. L. Sconyers, and B. Sherrod, “Adaptive disk spin-down for mobile computers,” *Mob. Netw. Appl.*, vol. 5, no. 4, pp. 285–297, 2000.

[67] Y.-H. Lu and G. De Micheli, “Adaptive hard disk power management on personal computers,” in *VLSI, 1999. Proceedings. Ninth Great Lakes Symposium on*, 1999, pp. 50–53.

[68] B. Schilit, N. Adams, and R. Want, “Context-aware computing applications,” in *Mobile Computing Systems and Applications, 1994. WMCSA 1994. First Workshop on*, 1994, pp. 85–90.

[69] A. K. Dey, “Providing architectural support for building context-aware applications,” Georgia Institute of Technology, 2000.

[70] S. S. Mahmoud and I. Ahmad, “A Green Model for Sustainable Software Engineering,” 2013.

[71] B. Penzenstadler, “Towards a Definition of Sustainability in and for Software Engineering,” in *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, 2013, pp. 1183–1185.

[72] N. Amsel, Z. Ibrahim, A. Malik, and B. Tomlinson, “Toward sustainable software engineering: NIER track,” presented at the Software Engineering (ICSE), 2011 33rd International Conference on, 2011, pp. 976–979.

[73] M. Dick, S. Naumann, and N. Kuhn, “A Model and Selected Instances of Green and Sustainable Software,” in *What Kind of Information Society? Governance, Virtuality, Surveillance, Sustainability, Resilience*, vol. 328, J. Berleur, M. Hercheui, and L. Hilty, Eds. Springer Berlin Heidelberg, 2010, pp. 248–259.

[74] M. Dick and S. Naumann, *Integration of environmental information in Europe proceedings of the 24th International Conference on Informatics for Environmental Protection Cologne/Bonn, Germany*. Aachen: Shaker, 2010.

[75] A. I. Anton, “Goal Identification and Refinement in the Specification of Software-based Information Systems,” Georgia Institute of Technology, Atlanta, GA, USA, 1997.

[76] M. Glinz, “On Non-Functional Requirements,” 2007, pp. 21–26.

[77] R. Malan, D. Bredemeyer, and B. Consulting, “Functional requirements and use cases,” *Functreq Pdf 39k June*, 1999.

[78] F. Rubik, U. Tischner, E. Schmincke, and M. Prosler, *How to Do Ecodesign?: A Guide for Environmentally and Economically Sound Design*. 2000.

[79] “PowerEscape Launches Second Generation Toolset Enabling New Power Optimization Strategies | Business Wire.” [Online]. Available: http://www.businesswire.com/news/home/20041025005755/en/PowerEscape-Launches-Generation-Toolset-Enabling-Power-Optimization. [Accessed: 18-Nov-2013].

[80] “VTune Analyzer.” [Online]. Available: http://it.toolbox.com/wiki/index.php/VTune_Analyzer. [Accessed: 18-Nov-2013].

[81] “Intel® Performance Tuning Utility 4.0 Update 5 | Intel® Developer Zone.” [Online]. Available: http://software.intel.com/en-us/articles/intel-performance-tuning-utility. [Accessed: 18-Nov-2013].

[82] “Using Intel® Laptop Gaming TDK in Game Application | Intel® Developer Zone.” [Online]. Available: http://software.intel.com/en-us/articles/using-intel-laptop-gaming-tdk-in-game-application. [Accessed: 18-Nov-2013].

[83] “Intel® Web APIs.” [Online]. Available: http://software.intel.com/sites/whatif/webapis/. [Accessed: 18-Nov-2013].

[84] “Using Intel® Power Checker to measure the energy performance of a compute-intensive application | Intel® Developer Zone.” [Online]. Available: http://software.intel.com/en-us/articles/using-intel-power-checker-to-measure-the-energy-performance-of-a-compute-intensive. [Accessed: 18-Nov-2013].

[85] “System Monitor - Wikipedia, the free encyclopedia.” [Online]. Available: http://en.wikipedia.org/wiki/System_Monitor. [Accessed: 18-Nov-2013].

[86] “PwrTest (Windows Drivers).” [Online]. Available: http://msdn.microsoft.com/en-us/library/windows/hardware/ff550682%28v=vs.85%29.aspx. [Accessed: 18-Nov-2013].

[87] “Open Event Viewer.” [Online]. Available: http://windows.microsoft.com/en-us/windows7/open-event-viewer. [Accessed: 18-Nov-2013].

[88] “Event Tracing: Improve Debugging And Performance Tuning With ETW.” [Online]. Available: http://msdn.microsoft.com/en-us/magazine/cc163437.aspx#S1. [Accessed: 18-Nov-2013].

[89] “Intel® PowerInformer | Intel® Developer Zone.” [Online]. Available: http://software.intel.com/en-us/articles/intel-powerinformer. [Accessed: 18-Nov-2013].

[90] “PowerTOP v2.0 Release | PowerTOP.” [Online]. Available: https://01.org/powertop/blogs/ceferron/2012/powertop-v2.0-release. [Accessed: 18-Nov-2013].

[91] L. Brown, K. A. Karasyov, V. Lebedev, A. Starikovskiy, and R. Stanley, “Linux Laptop Battery Life,” in *Proc. of the Linux Symposium*, 2006, pp. 127–146.

[92] D. Brooks, V. Tiwari, and M. Martonosi, “Wattch: A Framework for Architectural-level Power Analysis and Optimizations,” in *Proceedings of the 27th Annual International Symposium on Computer Architecture*, New York, NY, USA, 2000, pp. 83–94.

[93] “PowerAPI by abourdon.” [Online]. Available: http://abourdon.github.io/powerapi-akka/. [Accessed: 29-Nov-2013].

[94] “SSE Instruction Set.” [Online]. Available: http://softpixel.com/~cwright/programming/simd/sse.php. [Accessed: 18-Nov-2013].

[95] “AVX-512 instructions | Intel® Developer Zone.” [Online]. Available: http://software.intel.com/en-us/blogs/2013/avx-512-instructions. [Accessed: 18-Nov-2013].

[96] R. N. Mayo and P. Ranganathan, “Energy consumption in mobile devices: why future systems need requirements–aware energy scale-down,” in *Power-Aware Computer Systems*, Springer, 2005, pp. 26–40.

[97] T. Johann, M. Dick, E. Kern, and S. Naumann, “Sustainable development, sustainable software, and sustainable software engineering: An integrated

approach," in *Humanities, Science & Engineering Research (SHUSER), 2011 International Symposium on*, 2011, pp. 34–39.

[98] A. Noureddine, R. Rouvoy, and L. Seinturier, "A review of middleware approaches for energy management in distributed environments," *Softw. Pract. Exp.*, 2012.

[99] L. M. Hilty and W. Lohmann, "The Five Most Neglected Issues in Green IT," *CEPIS UPGRADE*, vol. 12, no. 4, pp. 12–15, 2011.

[100] L. M. Hilty and W. Lohmann, "An Annotated Bibliography of Conceptual Frameworks in ICT for Sustainability," 2013, pp. 288–300.

[101] S. Ruth, "Green it more than a three percent solution?," *Internet Comput. IEEE*, vol. 13, no. 4, pp. 74–78, 2009.

[102] J. W. Smith and I. Sommerville, "Green Cloud: A literature review of Energy-Aware Computing and Cloud Computing," 2010.

[103] C.-T. D. Lo and K. Qian, "Green Computing Methodology for Next Generation Computing Scientists," 2010, pp. 250–251.

[104] C. S. Ellis, "The case for higher-level power management," in *Hot Topics in Operating Systems, 1999. Proceedings of the Seventh Workshop on*, 1999, pp. 162–167.

[105] E. Saxe, "Power-Efficient Software," *Queue*, vol. 8, no. 1, pp. 10:10–10:17, Jan. 2010.

[106] M. Dick, S. Schade, and P. Smits, *Innovations in sharing environmental observations and informations: proceedings of the 25th EnviroInfo International Conference, October 5-7, 2011, Ispra, Italy*. Aachen: Shaker Verlag, 2011.

[107] "c10.png (PNG Image, 2488 × 1366 pixels) - Scaled (51%)." [Online]. Available: http://images.anandtech.com/reviews/cpu/intel/Haswell/ULT/c10.png. [Accessed: 25-Nov-2013].

7 Appendix

7.1 Annotated Bibliography

1. **Naumann, Dick et al. -2011- The GREENSOFT model: A reference model for green and sustainable software and its engineering [6]**

   Naumann and Dick propose the GREENSOFT reference model, which has its focus on the life cycle of software products aiming to improve it with regard to sustainability. The model has four parts: “Life Cycle of Software Products”, “Sustainability Criteria and Metrics”, “Procedure Models”, and “Recommendations and Tools” (Naumann, Dick et al., 2011, 296). The first two parts explicitly borrow from Life Cycle Analysis, treating software like a material product that goes through a development (or production) phase, a use, and an end-of-life phase. The last two parts of the GREENSOFT model aim at improving the processes in each phase of the life-cycle in order to meet the sustainability criteria, which are intended to cover all types of ICT impacts (from first to third order).

2. **Larsson -2011- Energy-efficient software guidelines [8]**

   Larsson presents guidelines for optimizing software for energy efficiency. These guidelines are operating system and architecture agnostic and are categorized into 4 main aspects. "Computational efficiency", "Maximize Idle", "Data efficiency", and "Context/Power aware behavior". After explaining each of these aspects, some tools are presented, which can help analyze an applications' energy efficiency.

3. **Steigerwald et al. -2007- Creating energy efficient software [40]**

   Similarly to Larsson's approach, the 3 main categories in this paper are introduced in order to improve the energy efficiency of software as well. The categories are "(1) Computational efficiency-methods to reduce energy costs by improving application performance, (2) Data efficiency-methods reduce energy costs by minimizing data movement and using the memory hierarchy effectively, and (3) Context awareness-enabling applications to make intelligent decisions at runtime based on the current state of the platform". For each of these titles, some methodologies and designs are

suggested. In addition, the section Operating Systems (OS) presents how to take advantage of the resources offered by the OS to save energy. In the last section some tools and technologies are provided which support creating energy efficient applications.

4. **Chung -2002- Software approaches for energy efficient system design [29]**

   In his PhD thesis Chung introduces 2 adaptive Dynamic Power Management (DPM) techniques, the sliding window technique and the adaptive learning tree technique, and shows the effectiveness of these by applying them to hard disc drives. Afterwards he compares these 2 DPM to other DPM policies. At the end Chung describes a "low energy software optimization framework". This thesis was published in 2002, so that some information might be outdated.

5. **Siebra et al. -2012- The software perspective for energy-efficient mobile applications development [7]**

   This paper discusses several software methods, which could be explored to develop energy efficient programs. The authors claim that significant energy savings are possible if energy efficiency of software is considered during the development phase. After the introduction, the problems of a low-level hardware perspective for energy efficient ICT Systems are illustrated in section 2. In section 3 various articles/papers about software aspects of green IT are presented and possible research directions suggested. The authors suggest, based on this literature, a general approach of 5 steps to develop green software. (1) Define the operations intended to be investigated, (2) measure the energy that such operations use, (3) relate these operations to high level development artifacts (detect which part of software uses how much energy), (4) define refactoring methods to change these artifacts, considering the energy-saving issue, (5) and finally test/validate the gains. In section 4 an experiment is presented using this 5-step methodology. Section 5 presents the adaptations carried out in the previous evaluation environment. Section 6 compares the methods used in this experiment with other strategies for measurements. Section 7 concludes this article.

6. **Winter and Jelschen -2012- Energy-Efficient Applications [11]**

   This research agenda discusses software reengineering tools and techniques, like static and dynamic program analysis, and systematic code transformations like refactoring, which can be used to obtain more energy efficient applications. "To investigate this topic a research seminar was held in winter semester 2011 / 2012. From this seminar three student papers emerged as well as an additional pro-seminar work. The first one is ”Specialized Processors for Energy- Efficient Use of Applications” by Marion Gottschalk, which focuses on recent processor development for increasing the energy-efficiency of applications. The second one is ”Energy Aware Computing” by Cosmin Pitu, which gives an overview about development in the field of energy aware computing. The last regular paper is ”Towards an Energy Abstraction Layer” by Mirco Josefiok, which aims at proposing an abstract energy measurement platform for mobile computing devices. The pro-seminar work by Michel Falk focuses on evaluating options for energy efficient development on Android devices." [11]

7. **Mayo and Ranganathan -2005- Energy consumption in mobile devices: why future systems need requirements-aware energy scale-down [96]**

   This paper focuses on energy scale-down from different perspectives. Firstly, the concept of energy scale-down elaborates with the comparison of special-task devices and general-task devices. Secondly display, wireless, and processor components are investigated in the concept of energy scale-down optimization. In the first part the authors conclude that special-task devices reduce energy consumption and are a good example for developing multi-purpose devices. In the second part general ideas are given for the three purposes which need to be integrated with the previous studies voltage and frequency scaling: architectural gating, selective memory usage, and disk spin-down.

8. **Ranganathan -2010- Recipe for efficiency principles of power-aware computing [4]**

   In this article Ranganathan suggests general methods (mostly regarding software engineering aspects) to create power-aware software.

9. **Johann et al. -2012- How to measure energy-efficiency of software metrics and measurement results [20]**

   This work presents a generic metric to measure software and a white-box method to apply it in a software engineering process. This method allows programmers to measure energy consumption of specific parts of software so that programmers are able to identify a resource-intensive code, which is an advantage compared to black box methods.

10. **Johann et al. -2011- Sustainable development, sustainable software, and sustainable software engineering: An integrated approach [97]**

    The authors make a proposal for a life cycle model which assists in developing green and sustainable software products. First, the life cycle of a software product is analyzed, and an approach on how to evolve the standard model for life cycle assessment to a model that follows sustainable principles is outlined. Then some recommendations are made for developers, administrators, and users, considering the whole life cycle of software. At the end two tools are suggested, which help to make software more sustainable.

11. **Dick et al. -2013- Green Software Engineering with Agile Methods [26]**

    This paper combines the two concepts, which are getting more and more popular: agile development and sustainable development. Dick at al. "integrates sustainable issues to the software development process by presenting a generic process model as well as integrating the Green and Sustainable Software Engineering enhancements into Scrum." Then two supporting methods are presented: "One integrates energy measurements into Continuous Integration where the other addresses the calculation of Carbon Footprints." In the "Related Work" section, there are good references to other studies in this research area.

12. **Noureddine et al. -2012- A preliminary study of the impact of software engineering on GreenIT [28]**

    Noureddine et al. (2012) define Green IT from a software perspective as a “discipline concerned with the optimization of software solutions with regards to their energy consumption” (Noureddine et al., 2012, 21). Their focus lies on the environmental

impacts caused by software, mainly CO_2 emissions related to power consumption; the approach is thus restricted to first-order effects. Conceptually the approach includes energy models showing the energy use caused by software in hardware resources (in particular processors, working memory, and hard disks), power monitoring at runtime, and the use of “power-aware information to adapt applications at runtime based on energy concerns”. [28]

13. **Noureddine et al. -2012- A review of middleware approaches for energy management in distributed environments [98]**

    Noureddine et al. define a comparative taxonomy for middleware approaches targeted at managing energy. In this study a number of existing approaches are compared and two major research fields, where contributions are lacking for energy management middleware, are identified: autonomic approaches, and generic approaches for energy management middleware platforms. "In Section 2, we review middleware approaches for energy management in distributed environments, proposing a detailed overview of the energy management issues in each approach. Section 3 overviews a number of approaches where a middleware layer is used for energy management of applications and devices in an intelligent environment. We compare the reviewed middleware approaches on the basis of an energy taxonomy introduced in Section 4. Finally, we conclude in Section 5." [98]

14. **Hilty and Lohmann -2011- The Five Most Neglected Issues in Green IT [99]**

    Hilty and Lohmann (2011) state that many studies have been published focusing on the energy consumption of ICT or the role of ICT as an enabler of energy efficiency. But this article argues that this approach is too narrow and a deeper understanding of the multifaceted relationships between ICT, society, and nature is needed. In this study the necessity of a broader perspective is emphasized and the five most challenging issues of Green IT from this perspective are identified.

15. **Hilty and Lohmann -2013- An Annotated Bibliography of Conceptual Frameworks [100]**

    "This bibliography covers articles published in journals, conference proceedings or as book chapters that reflect on the role of Information and Communication Technology

(ICT) in society's challenge of developing more sustainable patterns of production and consumption. The bibliography is focused on contributions presenting conceptual frameworks intended to structure this interdisciplinary field of research. Some sources not explicitly presenting a conceptual framework were included for their contribution to structuring the research field." [100]

16. **Hilty and Coroama -2009- The Role of ICT in Energy Consumption and Energy Efficiency [21]**

    This study analyzes the energy consumption of ICT and ICT's potential to induce energy efficiency and compares them. "The study looks both at today's situation as well as future opportunities and risks. The study discusses the following research questions:

    a.) Estimates of the current energy consumption of ICT,

    b.) Prospective future developments in this energy consumption, and

    c.) Future energy efficiency potentials induced by ICT in various economic sectors."

    The methodology relies on a literature review and expert interviews.

17. **Hilty et al. -2011- Sustainability and ICT – An overview of the field [2]**

    "This article gives an overview of existing approaches to using Information and Communication Technology (ICT) in the service of sustainability: Environmental Informatics, Green ICT, and Sustainable Human-Computer Interaction (HCI). This consideration leads to the conclusion that a combination of efficiency and sufficiency strategies is the most effective way to stimulate innovations which will unleash ICT's potential to support sustainability." [2]

18. **Ruth -2009- Green IT – more than a three percent solution [101]**

    This short article presents different aspects of the Term "Green IT" in a superficial manner under the following titles and delivers useful numbers:

    a. Data centers and servers

    b. PC's monitors and workstations

    c. Software

    d. Telecommuting

    e. Metrics

This article contains some helpful information, showing that Green IT is getting more and more important every day and "greening the ICT" is becoming a global tendency. It is also possible to find some useful statistics (i.e. examples of energy savings by data centers), which illustrate how important energy savings can be. In addition some "further readings" are suggested.

19. **Grosskop and Visser -2013- Identification of Application-level Energy Optimizations [32]**

    This article focuses on application level optimizations and argues that optimizations on this level have the best chance of being effective. The importance of visibility of applications' energy efficiency is emphasized and a method for identifying energy-efficiency optimizations in software applications is introduced. "In this article we present the Green Software Scan, an approach to effectively model software application energy consumption and to arrive at recommendations for optimization." [32] This method was tested on two industrial cases where such Green Software Scans were applied to an e-government and a mobile banking application. The result was that "significant energy savings can be realized with targeted modifications in software code, architecture, or configuration, that development of energy-efficient applications requires attention to detail throughout the development process, and that a close collaboration between operations and development is one of the main success factors for energy-efficient applications." [32]

20. **Brown and Reams -2010- Toward energy-efficient computing [17]**

    "This article suggests an overall approach to energy efficiency in computing systems. It proposes the implementation of energy-optimization mechanisms within systems software, equipped with a power model for the system’s hardware and informed by applications that suggest resource-provisioning adjustments so that they can achieve their required throughput levels and/or completion deadlines." [17]

21. **Sampson et al. -2011- EnerJ approximate data types for safe and general low-power computation [33]**

    "Recent research has begun to explore energy-accuracy trade-offs in general-purpose

programs." [33] Based on the observation that systems spend a significant amount of energy guaranteeing correctness, this study proposes that a system can save energy by exposing faults to the application, because some parts of applications can also be approximated, without causing any serious errors. "While approximate computation can save a significant amount of energy, distinguishing between the critical and non-critical portions of a program is difficult. In this study the EnerJ type system is presented. [...]Our type system provides a general way of using approximation: we can use approximate storage by mapping data to cheaper memory, cache, and registers; we can use approximate operations by generating code with cheaper, approximate instructions; and we can use method overloading and class parameterization to enable algorithmic approximation."

22. **Arnout -2005- Data-Efficient Software and Memory Architectures are Essential for Higher Performance and Lower Power [34]**

    Arnout presents tools to analyze the effects of C code running against a model of the target memory system. He defines general characteristics of data efficient software and compares data efficiency and computational efficiency. Then he introduces a methodology with a concrete example for improving data efficiency of systems. "Algorithm developers, system architects and embedded software engineers can each use their own expertise to improve data efficiency and make cost/power/performance trade-offs without changing their familiar design practices. The individual contributions of software developers, system architects and device optimization experts all lead to systems that are more data efficient, perform faster, consume less power, run cooler and are more cost effective."

23. **Smith and Sommerville -2010- Green Cloud – A literature review of Energy-Aware Computing and Cloud Computing [102]**

    "Smith and Sommerville have undertaken a systematic literature review of recent work in the areas of Cloud Computing and Energy Aware Computing in an effort to understand how Large Scale Complex IT systems consume electricity and what steps may be taken to improve their efficiency." [102]

24. **Albers -2010- Energy-efficient algorithms [35]**

    This article was summarized well in its introduction part:

    "This article focuses on the system and device level: How can we minimize energy consumption in a single computational device? We first study power-down mechanisms that conserve energy by transitioning a device into low-power standby or sleep modes. Then we address dynamic speed scaling in variable-speed processors. This relatively new technique saves energy by utilizing the full speed/frequency spectrum of a processor and applying low speeds whenever possible. Finally, we consider some optimization problems in wireless networks from an energy savings perspective." [35]

25. **Amsel et al. -2011- Toward sustainable software engineering: NIER track [72]**

    The aim of this study is to "raise awareness that similar software systems can have quite different levels of energy consumption, and therefore different environmental footprints, and that these and other environmental impacts are an important part of the software engineering field. [...] First, we investigated the extent to which users thought about the environmental impact of their software usage. Second, we created a tool called GreenTracker, which measures the energy consumption of software in order to raise awareness about the environmental impact of software usage. Finally, we explored the indirect environmental effects of software in order to understand how software affects sustainability beyond its own power consumption." [72]

26. **Shore -2009- Developing Power-Efficient Software Systems on ARM Platforms [10]**

    This study mainly focuses on the cost of badly placed memory accesses and also presents other aspects of energy efficiency of software systems such as computational efficiency, data efficiency, algorithm efficiency, efficient use of CPU, and impacts of operating systems. For each of these issues a general overview is provided.

27. **Mahmoud and Ahmad -2013- A Green Model for Sustainable Software Engineering [70]**

    Mahmoud and Ahmad focus on the software engineering process and introduce a software model to develop green software. They further present a brief literature overview in the field of green IT in the Introduction part. "We developed a green

software model with two levels. The first level proposes a new green software engineering process we designed based upon the development processes of sequential, iterative, and agile methods. We further also identify how each software engineering stage can be environmentally sustainable through green processes and/or green guidelines thus ending up with a green software product, and finally we include the metrics we consider relevant to measure the greenness of each software stage. The second level is composed of approaches taken by software itself to contribute to green computing. We categorize these different approaches and concepts into five main categories. Finally, we relate both levels to each other to indicate how software tools built to promote green computing and software concepts can be used and referred to in the stages of a software engineering process to help output a green and sustainable software product and have a green development process." [70]

28. **Lo and Qian -2010- Green Computing Methodology for Next Generation Computing Scientists [103]**

    Lo and Qian emphasize the importance of greening ICT Systems and show their huge energy saving potential exemplarily. They also emphasize that greening IT is not trivial and illustrate its complexity with 2 sample applications. The paper shows the lack of existing "green software engineering practices" and elucidates the necessity of those rather than suggesting any concrete solutions.

29. **Penzenstadler -2013- Towards a Definition of Sustainability in and for Software Engineering [71]**

    This paper provides an overview of the different aspects of software engineering regarding sustainability."This paper presents a definition of the aspects of sustainability in and for software engineering. Thereby, for software engineering is how to make software engineering itself more sustainable and in software engineering is how we improve the sustainability of the systems we develop."[71]

30. **Murugesan -2008- Harnessing green IT Principles and practices [14]**

    The Author suggests some general methodologies for reducing power consumption of IT systems as follows:

1.) Enabling power management features

2.) Turning off the system when not in use (Dynamic power management)

3.) Using screensavers

4.) Using thin-client computers

5.) Greening data centers

6.) Energy conservation

7.) Eco-friendly design

8.) Virtualization

This paper provides a good overview of the field regarding many different aspects and presents important research areas. It is useful for understanding the term Green IT in general, but details regarding each aspect mentioned will have to be accessed elsewhere.

31. **Kern et al. -2013- Green Software and Green Software Engineering – Definitions, Measurements, and Quality Aspects [23]**

    This paper discusses a classification of green software and green software engineering and provides guidelines for measuring energy efficiency of a software. The authors propose a quality model for green and sustainable software, outlining which metrics should be considered for energy-efficiency.

32. **Barroso and Holzle -2007- The Case for Energy-Proportional Computing [36]**

    This paper identifies the importance of energy proportionality in design purposes in the name of server space motivations. It is a comparative study on server power usage and energy efficiency in varying utilization levels. CPU contribution to total server power is comparatively analyzed.

33. **Ellis -1999- The case for higher-level power management [104]**

    This is an early study from 1999, but Ellis has foreseen the most important points correctly. This study emphasizes the importance of software aspects for energy optimization and illustrates it by using a concrete example. It is also emphasized that there are no adequate and sufficient tools and mechanisms for power management.

After 14 years there is still a lack of appropriate tools and methodologies in order to measure and manage energy efficiency.

34. **Saxe -2010- Power-Efficient Software [105]**

    Saxe classifies software into two familiar ecosystem roles: resource managers (producers) and resource requesters (consumers). Then he examines how each can contribute to (or undermine) overall system efficiency. [105] Some good and bad software engineering practices are presented here with concrete examples, which makes it easier to understand the role of software for overall efficiency of a system. At the end of the article the characteristics of efficient software regarding design, CPU utilization, memory utilization, and I/O utilization are provided.

35. **Govindasamy and Joseph K -2011- Optimization of Operating Systems towards Green Computing [51]**

    This paper focuses mainly on operating system optimizations and also outlines briefly some of the other important aspects through which energy efficiency can be pursued in computers. This study also provides useful advice for users to aid green computing. Operating system optimizations are analyzed elaborately and classified into the following categories:

    1. Virtualization
    2. Terminal servers
    3. Shared memory
    4. Power management
    5. Storage Management
    6. Video card
    7. Display
    8. Computer multitasking
    9. Parallel Processing in Computers
    10. Parallel computing
    11. Amdahl's law and Gustafson's law
    12. Software Pipeline

36. **Vereecken et al. -2010- Overall ICT footprint and green communication technologies [19]**

    In this paper the authors give an overview of the environmental issues related to communication technologies and they present an estimation of the overall ICT footprint. Additionally they present some approaches on how to reduce this footprint and how ICT can assist in other sectors reducing their footprint.

37. **Dick et al. -2011- Measurement and Rating of Software Induced Energy Consumption of Desktop PCs and Servers [106]**

    Dick et al. provide a black box method to measure software-induced energy consumption of stand-alone applications on desktop computers as well as interactive transaction-based applications on servers. As a proof of concept, two exemplary measurements are described, showing the influence of software use on power consumption.

38. **Belady et al. -2008- Center Power Efficiency Metrics PUE and DCIE**

    In this paper tow metrics are defined in order to measure efficiency of data centers. The first one is called "Power Usage Effectiveness (PUE)", the second one "Data Center Infrastructure Efficiency (DCIE)". The PUE is defined as: PUE = Total Facility Power/IT Equipment Power (1), and its reciprocal, the DCIE, is defined as: DCiE = 1/PUE = IT Equipment Power/Total Facility Power x 100%. (2)

    The authors also introduce a new metric to measure the efficiency of the components of a data center as a future work, which could be helpful to identify the bottlenecks. This way it would be possible to focus on these critical points with the potential of improvements.

39. **Li and Absar -1997- Minimizing Memory Access By Improving Register Usage Through High-level Transformations**

    Li and Absar propose a method to reduce memory access related power consumption by reducing the number of data transfers between processor and memory, or between a higher (closer to processor) level of memory and a memory at a lower level using

source program transformation. They confirm their method by illustrating experimental results on a number of benchmarks.

7.2 White Box Measurement Example

Johann et al. present the following experiment using white-box measuring: [20] "As a second example we chose a web-based information system. Usually there are parts such as browser, web-server and database involved. Therefore, we chose a client/server application, which is used in productive operations at the Environmental Campus Birkenfeld. Its purpose is to collect, process and visualize consumption data of the campus buildings (like: energy, water, etc.). The server stores the data in a database that is managed by a web interface. For demonstration purposes only a cut out of the software is used. One of its capabilities is to visualize the consumption data as a line graph over a time period. The charts are dynamically generated for each request.

When looking at the description of the chart feature, a useful metric is: **Joules / delivered charts**. This metric can be expanded to other parts of the chart generation, e.g. database queries, the image generation, and the complete request/response cycle. We assume that this is a good way to represent the energy consumption of this part of the software. It must be kept in mind that there is no 'one size fits all' metric and applications can be split up into more or less fine-grained modules, which have their own metrics.

The load for the measurement will be simulated and is generated by a fixed number of users, which send multiple requests to the server. Therefore, the source code must be instrumented at the point where the request comes in and where the response goes out. The counters must increase with an incoming request and decrease when the response is sent. Thus, we know how many requests are currently in progress. In addition to the two systems described in the set-up, one can use a third system that

generates the load. Here, we will try to demonstrate how the energy efficiency changes when serving different amount of users.

In a first testrun a load of 15 users was generated. The result is shown in figure 5. After the ramp-up period the users that are waiting for their response (blue) even out at approximately six users. In this phase the average power rating of the server is about 110 Watts (green), while simultaneously serving 15 users with a waiting time way under one second. In one hour the system with 15 simultaneous users consumes 110 Wh = 396kJ.

The same measurement is performed with 25 simultaneous served users, whereas the average queued users is approximately 17, which means that users have to wait a little bit longer than in the measurement with 15 users, but still get their response under one second. The energy consumed is also 110 Watts and users can still be served at a satisfactory time. As one can see, the claimed response time plays a key role: the higher the demand for fast response, the lower is the energy efficiency." [20]

7.3 Advanced Configuration and Power Interface (ACPI) Power States

The ACPI specification defines the four Global "Gx" states and six Sleep "Sx" states for an ACPI-compatible computer-system:[56][58]

- G0 (S0): Working. "Awaymode" is a subset of S0, where monitor is off but background tasks are running.
- G1, Sleeping. Divided into four states, S1 through S4:
    - S1: All the processor caches are flushed, and the CPU(s) stops executing instructions. The power to the CPU(s) and RAM is maintained. Devices that do not indicate they must remain on, may be powered off.
    - S2: CPU powered off. Dirty cache is flushed to RAM.
    - S3: Commonly referred to as Standby, Sleep, or Suspend to RAM (STR). RAM remains powered.

    - S4: Hibernation or Suspend to Disk. All content of the main memory is saved to non-volatile memory such as a hard drive, and is powered down.
- G2 (S5), Soft Off: G2/S5 is almost the same as G3 Mechanical Off, except that the PSU still supplies power, at a minimum, to the power button to allow return to S0. A full reboot is required. No previous content is retained. Other components may remain powered so the computer can "wake" on input from the keyboard, clock, modem, LAN, or USB device.
- G3, Mechanical Off: The computer's power has been totally removed via a mechanical switch (as on the rear of a PSU). The power cord can be removed and the system is safe for disassembly (typically, only the real-time clock continues to run - using its own small battery).

ACPI defines the device-dependent device states D0–D3:[56]

- D0 Fully On is the operating state.
- D1 and D2 are intermediate power-states whose definition varies by device.
- D3 Off has the device powered off and unresponsive to its bus.
    - D3 Hot & Cold: The D3 state is further divided into D3 Hot (has aux power), and D3 Cold (no power provided). A device in D3 Hot state can assert power management requests to transition to higher power states.

ACPI defines the CPU power states C0–C3 as follows:[56]

- C0 is the operating state.
- C1 (often known as Halt) is a state where the processor is not executing instructions, but can return to an executing state essentially instantaneously. All ACPI-conformant processors must support this power state. Some processors, such as the Pentium 4, also support an Enhanced C1 state (C1E or Enhanced Halt State) for lower power consumption.[11]

- C2 (often known as Stop-Clock) is a state where the processor maintains all software-visible state, but may take longer to wake up. This processor state is optional.
- C3 (often known as Sleep) is a state where the processor does not need to keep its cache coherent, but maintains other state. Some processors have variations on the C3 state (Deep Sleep, Deeper Sleep, etc.) that differ in how long it takes to wake the processor. This processor state is optional.
- Manufacturers for some processors define additional states. For example, Intel's Haswell platform has states up to C10, where it distinguishes core states and package states

Processor Package and Core C-States

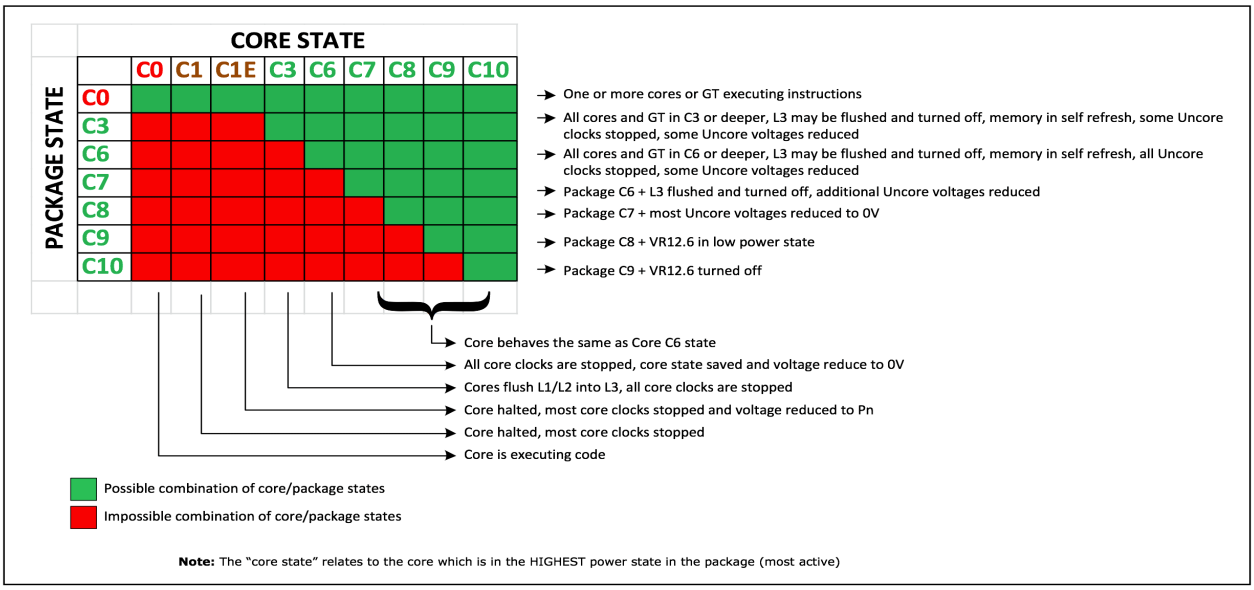

Figure 14: Processor package and Core C-States [107]