Vienna University of Technology
Faculty of Informatics
Institute of Software Technology and Interactive Systems
Favoritenstr. 9-11/188
1040 Vienna, Austria

SS2009

Seminar-Thesis:
Lecture Series on Sustainable Development and Information and Communication Technology

Hard- and Software Strategies for Reducing Energy Consumption in Embedded Systems

Johannes Inführ
Peter Jahrmann

June 29, 2009

Supervised by Alexander Schatten

Abstract

Due to the ongoing increase in earth's population, adequate supply of resources is going to be a major issue. One basic resource in rising demand is energy and in particular electrical energy. In this paper we analyse the contributions of the scientific community toward the goal of sustainability with regard to energy consumption of embedded systems. We consider the energy requirements during operation of embedded systems and the influence of the whole design process of embedded systems on this energy requirements, starting with design methodologies for energy efficient embedded system architectures, continuing with special hardware technologies and concluding with software techniques covering both operating system level software used for example for scheduling and user level software running on those embedded systems.

1 Introduction

At the time of writing, 6.7 billion people live on earth [16]. It is estimated that the world population will reach 9 billion by 2040. This increase alone will result in a rapidly rising energy consumption. This problem is amplified by rising living standards in developing countries. The current world electricity production is about 20 trillion kilowatt-hours. By 2030, it will reach 30 trillion kilowatt-hours, mostly through coal and natural gas [38]. Coal is a very dirty form of energy, especially when its emissions are not properly filtered, which is mostly the case in developing countries. This will result in serious repercussions for the environment. It stands to reason to try to limit those consequences. One approach is sustainable development, which means to consume resources to meet ones needs in such a way that future generations have the ability to meet their needs in the available environment. Producing energy in ways that destroy the environment obviously contradicts this goal. In theory, there is more than enough solar energy available to supply the entire world. In practise, harvesting this energy in a cheap and efficient way is not easy. Therefore it is important to put the created energy to good use instead of wasting it. One surefire way to waste energy is using the standby mode on electric appliances. It is estimated that between 5 and 10 percent of domestic power is used solely to power devices in standby-mode. There are power controlling devices which are specifically developed to disconnect devices on standby like the one reported in [35]. The interesting thing to note is that the inventor of this device states that "this unique power controller design analyses power consumption using an artificial intelligence algorithm implemented on a high-end micro-controller". The key point here is the "high-end micro-controller", which is a nice example for an embedded system and how they can be present without being really noticed, often in surprising quantities. This leads to the main topic of this term paper, the energy consumption of embedded systems and the various strategies available to reduce it. Besides helping to save the environment, reducing energy consumption of embedded systems can lead to immediate monetary rewards for their producers, for example increased sales of the mobile phone with the longest standby time, which we suspect is the main reason behind the efforts to minimise power consumption. In this term paper, we consider three possible avenues for reducing power consumption in embedded devices. In section 2 we discuss some methods that can be used to design energy efficient embedded systems from scratch. In section 3, implementation issues of energy intensive hardware components such as caches are covered. Section 4 gives an overview of software methods available to reduce energy consumption, like efficient scheduling of tasks in a variety of settings or using specialised compiling techniques. It should be pointed out that this separation is not as clear as one might believe, because of various interactions between topics. For example, software methods are useless if the hardware does not support some sort of energy saving mechanism like dynamic voltage scaling or hardware design methods that make extensive use of simulation software. In section 5 we draw some conclusions about the covered methods.

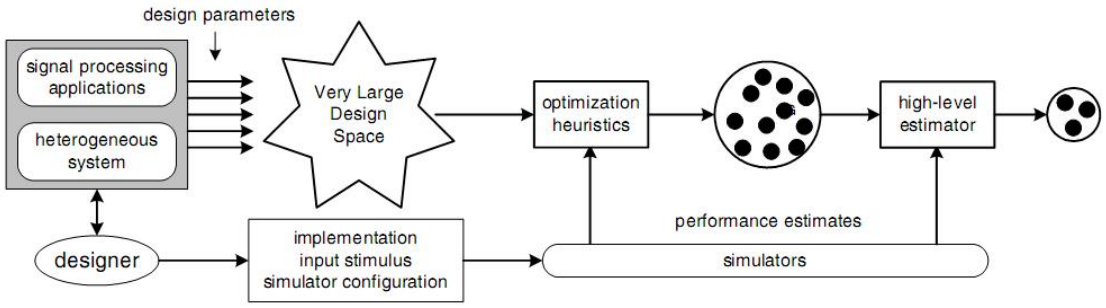


Figure 1: Hierarchical design approach as defined in [23]

2 Design Methods

Every embedded systems starts off as a design. During the design-phase, the important characteristics of the system get fixed. Therefore it is very important to get the design right in order achieve energy efficiency because every further action to reduce energy consumption is futile if the design does not support it. Design can be interpreted as a search in the space of possible system configurations. Various design possibilities concerning efficiency, like using one or multiple computing cores, reconfiguration capabilities of components, low power states, idle states, execution in hard- or software and many more add to the design space which needs to be searched for the optimal design. A hierarchical approach for designing energy efficient embedded systems is discussed in [23], which focuses on heterogeneous embedded system. Heterogeneous embedded systems consist of multiple processing components such as general purpose processors, specifically designed microcontrollers for realtime-systems called digital signal processors *(DSPs)*, field programmable gate arrays *(FPGAs)*, which can be configured by the designer after manufacturing and peripheral components like memory and sensors.

The hierarchical design space exploration works in two steps as seen in Fig. 1. First, a set of designs is created that meet given constraints by using a suitable optimisation heuristic. In the second step, the performance of those designs is evaluated by a high-level performance estimator. Since this performance estimator only works on a preselected set of candidate designs, it can include design parameters not covered by the optimisation heuristic and therefore find more efficient solutions. The capabilities of this method depend on its robustness against approximation errors introduced by the performance estimator. In an experiment that considered a linear array of tasks and a configurable device and varying performance estimation accuracy it was found that this method delivered the same or better designs than dynamic programming at nearly every level of accuracy and problem complexity. The hierarchical design method was demonstrated on two problems, a beam-forming application and a personnel detection application. The beam-forming application had a small search space of approximately 320 designs. Nonetheless, only 8 of those needed to be evaluated by the performance estimator to detect the most energy efficient design that meets latency constraints. For the second

problem, the task was to select energy efficient hardware and map tasks on to them as to meet a hard real time constraint (finish the personnel detection in one image before the camera delivers the next image) and minimise energy dissipation. From the initial design space of 73000, only 16 were selected by the optimisation heuristic and subsequently tested by the performance estimator. This process took less than a minute, while evaluating all 73000 designs with the performance estimator took approximately 10 hours.

The run-times stated in [23] underscore the importance of fast energy and performance evaluation of candidate designs. In [24] a method for automatic energy/performance macro modelling of software is presented to speed up the evaluation of designs, which is also very important if the overall system is to be energy efficient. The key observation in [24] is that software consists mostly of reused libraries "glued" together by a bit of custom code. So it is possible to develop a detailed energy consumption and performance model for the libraries and use them together with some high level estimates of the glue-code to get a accurate picture of energy needs of the whole software package. In this paper, the developed model for some often used libraries is accurate to one percent both for performance and energy consumption and using this model the authors could reduce the runtime of sample code evaluation from a day to a minute.

One important special case for designing embedded systems is the design of *FPGAs*, because they have high computational power and are very flexible. One drawback of *FPGAs* is, that it is not possible to optimise the embedded system design on the gate level, e.g. the most basic level in digital circuits, because the gate-level is implemented by the producer of the *FPGAs*. On the other hand, it is stated in [26] that optimisations on the algorithmic level have about 20 times more impact on the total power dissipation than optimisations on gate level. The flexibility of *FPGAs* is somewhat problematic in the sense that there are a lot of choices on how implement the required functions and again we see an enormous search space which needs to be traversed efficiently. In [25] a design method is proposed to handle this problem, which is illustrated in Figure 2.

The design method consists of four steps. The first step is the selection of the domain. A domain is a set of closely related architectures and algorithms suitable for those architectures. Human intelligence is required to identify domains that will eventually lead to superior designs, but tools are available to quickly obtain preliminary trade-off results between power, size and speed to support the decision process. After the domain is selected, a domain specific high level energy model is developed. At this stage, power management techniques are considered and with the help of the energy model parameters like operation frequency and number of computing components are chosen after an energy consumption optimisation. In the third step of the design methodology, the design space of each considered domain is traversed to find designs that meet certain selection criteria like low energy dissipation. With the developed energy models it is possible to perform trade-off analyses without resorting to low level simulations. This results in a small set of designs that satisfy the selection criteria. In the fourth step of this design method, those designs are subjected to low level simulations in order to

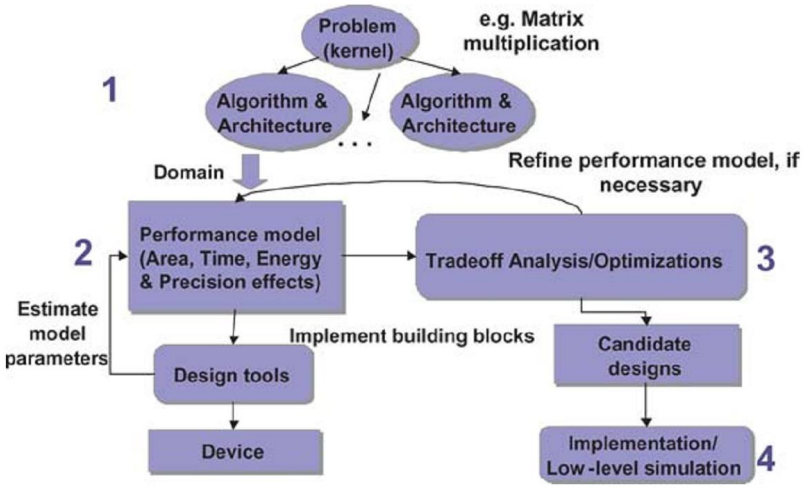


Figure 2: Design methodology as defined in [25]

validate the high level energy models and to distinguish between the candidate designs with a higher degree of certainty. If the differences between the high level energy model and the simulation results are intolerable, the high level model needs to be updated and step three repeated. For more details, especially on domain specific energy modelling, see [25]. With this design method, improvements of about 45 percent in energy efficiency compared to state of the art implementations of an adaptive beam forming application are reported. For other problem domains like matrix multiplication or fast Fourier transform, 10 to 78 percent increased energy performance where achieved.

As could be seen up to this point, the design of energy efficient embedded systems is a difficult task. One further complication is that an embedded system usually doesn't have only one function, which can be analysed and considered during system design, but multiple functions that need to be performed, with different requirements on computing capabilities and so on. Those systems are called multi-mode embedded systems, because they operate in various modes. One possible approach to designing such a system could be to design every function separately and then put the pieces together into one system. This approach, though elegant in its simplicity has major drawbacks which make it unacceptable in practise. For one, the different functions that need to be performed often need the same components as other functions and including them once for each function is a waste of resources. Therefore great opportunities for energy efficient designs are missed. So the better way is to design the system as a whole. Because of this, one key observation can be exploited while designing how the functions are mapped to hardware or software: the execution probabilities of the different functions are different. How this can be done is explained in [30], where a design methodology for energy-efficient multi-mode embedded systems is introduced. This method starts by specifying the embedded system as operational mode state machine. This describes the behaviour of the embedded system at a very high level. Figure 3 gives an example for an operational

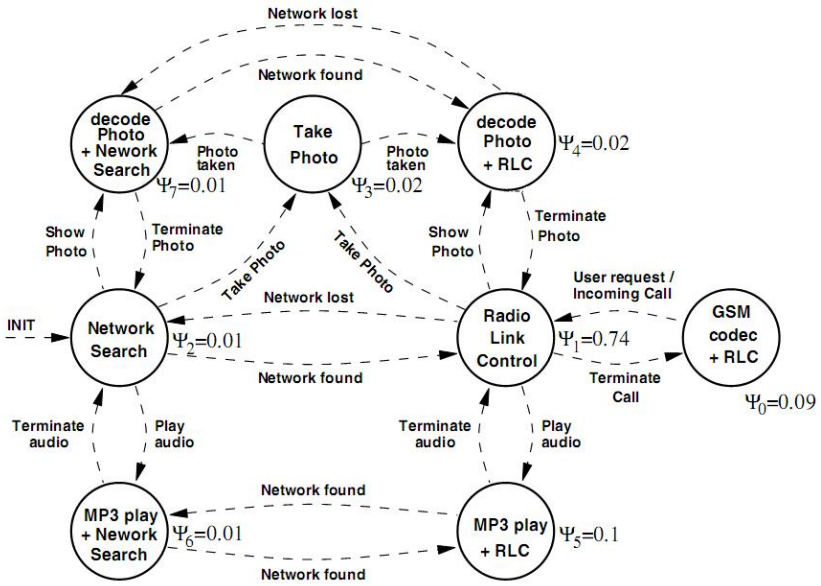


Figure 3: Operational mode state machine of a cell-phone application as defined in [30]

mode state machine. The description is augmented by the probabilities with which the machine will be in some state and detailed descriptions of the necessary computation steps in each state (the task graph, which is not shown in figure 3.

The problem now is to find a mapping between computational units (like a multi purpose *CPU* or a *FPGA*) and the tasks in the operational mode state machine, a task schedule for the computational units and a voltage schedule if dynamic voltage scaling *(DVS)* for power saving is used, that satisfies all timing constraints and needs the least possible energy. Ironically, it might be a good idea to implement the same function on multiple computational units to allow power saving by turning off currently unused units. In [30] a genetic algorithm with specialised mutation operators is used to solve this problem. Computational results show a decrease of 67 percent of the average power on the cell-phone example with dynamic voltage scaling and 30 percent without *DVS*.

We have seen that multiple computational elements in embedded systems are difficult to handle, but this still misses an important aspect. The communication needed between those computational elements is not negligible. The power needed for communication can amount to 20 to 30 percent of the total chip power. One can now use a technique similar to dynamic voltage scaling, namely adjusting the speed of the communication links to save electrical energy. There are two ways to scale the communication speed. One is an online technique, where the speed is adjusted dynamically depending on the current traffic on the link. The other way is an offline technique, were each communication link has a fixed speed based on the communication patterns of the target application. In [34] a method for determining the link speeds in an offline setting for real time systems is presented. In order to determine the optimal link speed assignment, this method first

needs to determine how the communication is actually routed between the computational elements (routing path allocation). To do that, the method also determines the assignment of computing tasks to processing elements (task assignment) and schedules their execution (task scheduling), because they are both needed to calculate the volume of communication that has to be handled. The method works by nesting three genetic algorithms inside of each other. The outer most genetic algorithm computes the optimal task assignment. Each task assignment in the population also holds the second genetic algorithm which determines the task scheduling given the task assignment. Each scheduling in this population also holds the third genetic algorithm which determines the routing path allocation given the task assignment and task scheduling. Based on the routing path allocation the link speed is assigned (for details see [34]) and the total energy requirements are calculated. This nesting of genetic algorithms ensures that the lower levels always correspond to the higher levels (e.g. every time a task assignment is changed and then evaluated, the evaluation starts a new genetic algorithm based on the current assignment). Additionally, special crossover and mutation operators have to be used, especially for the routing path allocation. The experimental results show that indeed every one of the three genetic algorithms is important for overall energy savings. The energy consumption of the on-chip network could be decreased by 39 percent on average. For communication between computational units from a more hardware-centric point of view, see section 3.3.

3 Hardware Methods

Designing energy efficient hardware for embedded systems is, after the design of the system, the second important ingredient for overall energy efficiency. This section introduces some hardware methods to decrease energy consumption of computational units. First, we will discuss a complete processor architecture tuned to reduce energy dissipation [2]. In the subsequent subsections, methods of reducing the power requirements of parts of computational units are discussed. This covers address translation in section 3.1, cache architectures in section 3.2, communication in section 3.3 and probabilistic arithmetic in section 3.4. While those sections cover important parts of computing elements, there are a lot of pieces missing and subsequently only serve as an overview of the possibilities of constructing energy efficient hardware. As already mentioned, the most powerful form of constructing energy efficient computing elements is to design the whole element from scratch. This is done for example in [2], where a complete energy-efficient processor architecture for embedded systems is introduced. In this paper it is stated that the energy requirements of memory and busses to transfer data and instructions to the functional units can account for more than 70 percent of the total energy dissipation of a processor. Therefore, the main focus lies in keeping the needed data and instructions close to those functional units as to minimise the need to access and copy memory. For the data supply, the proposed processor architecture features a small operand register file which is located directly at the input of the functional units to reduce the cost of

transferring operands and results between registers and functional units. The second level of the register hierarchy are indexed register files, used to capture data locality on a higher level than is possible with the operand register files. It costs 10 times more energy to read from an indexed register file than an operand register file. The third level in the data hierarchy is the ensemble memory, which houses a memory bank for each processor. Reading from the ensemble memory costs 30 times more energy than reading from the operand register file. A data management unit inside each processor manages the movement of data between the three levels in the data hierarchy. Besides the operand register file, every functional unit also contains a shallow instruction register file for local storage of instructions. Instructions are cached in the ensemble memory and managed by the instruction memory unit in each processor. The routing of operands and instructions is exposed to the compiler to allow finer control over the data and instruction flow inside the processors. In a direct comparison with a RISC architecture, the RISC processor consumed 23 times more energy than the proposed processor architecture. The best improvement were the instruction registers, which reduced the cost of supplying instructions by 49x. Compared to an ASIC, the proposed processor architecture uses 1.5 times more energy, but then again it is a multi purpose processor rather than a fixed digital circuit.

3.1 Address Translation

Memory management is an important characteristic of embedded processors, since it has a lot of influence on the overall performance and power consumption. Virtual memory has been introduced to hide the complications of instruction and data relocation, memory sharing and protection. Every application has its own virtual memory. Parts of the virtual memory are mapped onto the real memory. To access data in the virtual memory, an address translation has to be performed. A translation look-aside buffer *(TLB)* caches the most requested translations, but it consumes about 20 percent of the total cache power. This energy consumption problem and the unpredictability of execution times (which is important for real time applications) seriously impedes the implementation of virtual memory in embedded systems. In [43] and [44] a method called arithmetic address translation is introduced, that aims at greatly reducing the power consumption of the translation look-aside buffer and making access times predictable for real-time systems. In traditional *TLBs*, no connection between consecutive virtual addresses is assumed and every memory translation request has to be handled either by the buffer or explicitly calculated in the operating system kernel. The key change in arithmetic address translation is, that it is assumed that consecutive data access in virtual memory results in consecutive data accesses in physical memory. With this assumption it is possible to simply add an offset constant to the virtual address to get a physical address. This not only reduces the power consumption of the *TLB* unit because no lookup has to be performed, but it also has beneficial influence on the overall performance, because consecutive reads do not evict other, possibly useful translations from the look-aside buffer. To take care that the central assumption of arithmetic address translation holds,

special care is needed by the compiler and operating system. The compiler has to work together with a profiler to find hot spots in the code, since those are the locations where code will be executed 90 percent of the time and where arithmetic address translation has the most effect. Then the compiler has to add code to notify the operating system that a hot spot is being entered (or left), so that the operating system can ensure that arithmetic address translation will be valid. Additionally, the operating system has to initialise the registers that save the offset between virtual and physical memory addresses. Even more care has to be taken in the presence of multitasking. For an in depth discussion of the requirements on compiler and operating system, see [44]. Validation with real world examples like jpeg, mpeg and mp3 encoding showed, that the address translation energy consumption could be reduced consistently by more than 80 percent.

3.2 Cache

As already seen in the introduction to hardware methods and the efficient processor architecture, up to 70 percent of the total energy requirements are due to the memory and busses used to deliver data and instructions to the functional units. The data for energy consumption of parts of the processor given in [2] shows that accessing cache data needs an order of magnitude more energy than actual computations. So the first step for energy efficiency is to make sure that if the cache is accessed, it delivers the needed data and no further memory read is needed. One possibility is also to only access the part of the cache which holds the needed data. This is possible for n-way set associative cache, which consists of n cache partitions which are accessed at the same time. In [37], a predictor is developed which allows to first check only one of those cache partitions for the requested data. This works by saving the last accessed partition and trying it at the next access. Validating this method showed an average hit rate of 90 percent and energy savings of 64 percent compared to a traditional cache structure. Unfortunately, this kind of prediction can not be used for L2 cache systems, which need different methods for partition prediction. One possible method is presented in [22].

Dynamic Voltage Scaling has already been mentioned as method to reduce power consumption of embedded processors. The full energy saving potential of this method can not be fully realised, because of differences in voltage requirements between functional units an caches. Caches generally need a larger supply voltage to reach speed and reliability targets. So one possible avenue to save more energy is the design of caches that operate at lower voltages, which is explored for example in [11]. This paper presents an reconfigurable energy efficient near threshold cache architecture. The “near threshold” corresponds to a reduced supply voltage in vicinity of the operating point with minimal energy consumption. If one would reduce the supply voltage even further, the energy consumption would rise again because the leakage current starts to increase. The memory storage cells can be made to tolerate “near threshold” supply voltages with acceptable reliability, but it requires a significant increase in cell size, which means

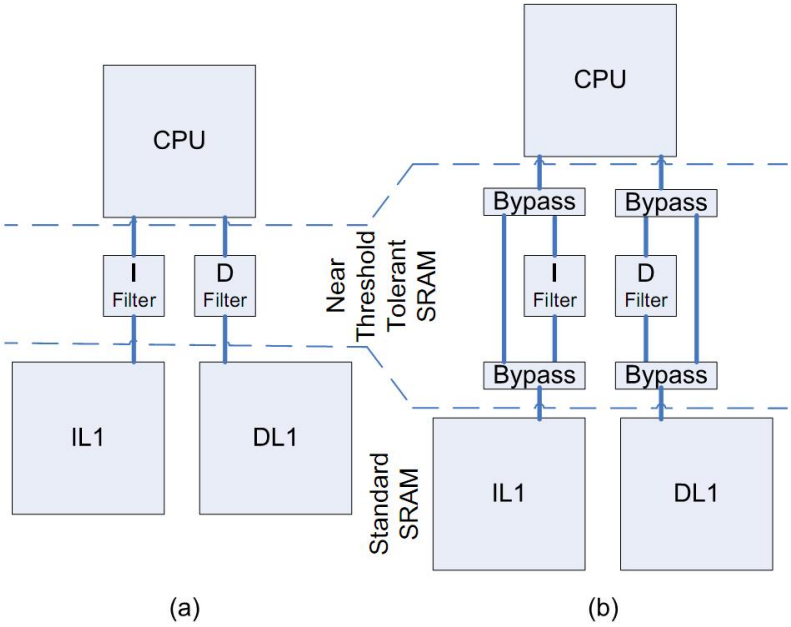


Figure 4: Near threshold filter cache architectures without (a) and with (b) bypass networks as presented in [11].

increased power consumption per cell at full supply voltage and less storage capacity if the space occupied by the memory is to remain constant. In order to overcome those problems, [11] develops two near threshold filter cache architectures as can be seen in 4.

The general idea is to use a filter cache. A filter cache is a small, energy efficient cache between the cpu and the L1-cache. If the cpu issues a data request, the filter cache first checks its contents before the request is sent to the L1-cache in case of a cache miss. Near threshold memory cells are now an obvious choice for this filter cache. However, this introduces additional latency, because a L1-cache lookup is only initiated after an filter cache miss. So an bypass network can be introduced, which bypasses access to the filter cache if the miss rate is to high. This introduces new problems (when to activate the bypass, how to keep the bypassed filter cache current) and so the design is refined further. The cache is redesigned to work in two modes, the conventional mode and the filtered mode. In the conventional mode, the filter cache and L1-cache are accessed in parallel, which does not introduce new latency, but also does not save any power. In the filtered mode, the filter cache is accessed first and the L1 cache only if there is a cache miss. From filtered to conventional mode is switched when the miss rate of the filter cache get too high, and the switch from conventional to filtered mode occurs when the filter cache would produce a sufficiently high hit count. This design produces 86 percent lower energy requirements in low power mode and 2 percent increase in runtime in high performance mode compared to a standard design.

Special operating environments for embedded systems can impose special challenges for the caches. One example are mobile computing environments, where it it difficult but very important to efficiently cache data, because the energy costs of requesting data over a wireless connection are high. In [32] an energy efficient data caching and prefetching mechanism for mobile devices is introduced. This is based on a analytically derived utility function, which takes into account how recently the data was used, but also how certain it is that the data is still current. This is very important in mobile settings, because it makes no sense to cache a news webpage for two days in a mobile internet device. If data is requested that is in the cache but not very certain, a validation request is sent to the server. For details see [32]. With this method it is possible to achieve more than 10 percent energy saving. A different way of dealing with stale data in caches of mobile embedded systems is presented in [3]. It is based on servers which periodically send invalidation reports, which specify which data was updated and should no longer be considered valid in the caches of the mobile devices. This method is not very energy efficient, because the mobile devices need to listen to the whole invalidation report to get the information they are looking for. In this paper three different methods for cache invalidation are considered, which result in decreased energy consumption for the mobile embedded systems.

As we have seen, communication can be a very energy consuming process. This is not only true for wireless communication, but also for on chip communication. In combination with multiple computational element with caches things get complicated. Then every computational element has to listen on the communication bus for memory access to memory that the processor has cached. If access occurs to such memory, the processor has to request the updated data to keep his cache current. This produces a massive communication and energy overhead. In [42] a method is described to reduce this overhead. It depends on in-depth knowledge of consumer-producer relationships between computational elements to optimise the bus snooping. In the field of embedded systems the requirement of in-depth knowledge is not very problematic, because those systems are usually locked and the developer has full control over them. With the proposed method, average power reductions of more than 80 percent were achieved. More information on communication without connection to caches will be given in section 3.3.

3.3 Communication

Communication between computational components on System-on-Chip designs can account for up to 50 percent [20] of the energy dissipation of the whole system. Therefore it is paramount that the communication hardware of the embedded system is designed in an energy efficient way. An overview of techniques for energy efficient communication in embedded systems is given in [27]. These techniques cover all levels of the design hierarchy, from the circuit level to the architecture level as can be seen in in figure 5. Advanced techniques are particularly important because the use of dynamic voltage scaling reduces noise margins which makes communication more error prone. Circuit level

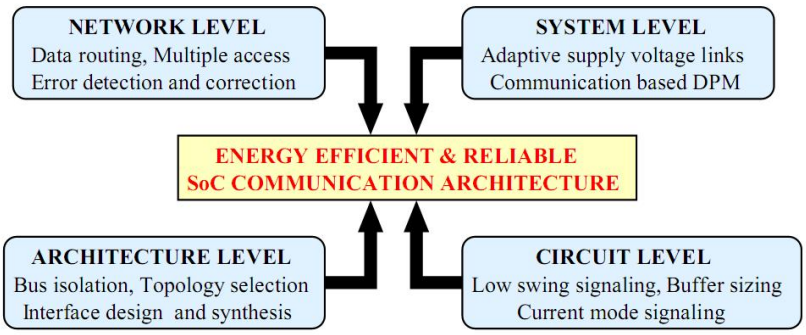


Figure 5: Design levels involved in optimising the communication architecture of SoCs as presented in [27]

techniques are concerned with getting a voltage signal from the beginning of a wire to the end of it. The classical configuration is a two inverter rail to rail signal swing where a CMOS inverter connects the signal-wire either to ground or the supply voltage, which in turn operates another inverter at the receiving end to restore the original signal. This signalling mode wastes power, mostly because of the capacitance on the signalling wire. Reducing the swing voltage, eg. the supply voltage, leads to quadratic improvements of energy efficiency, but also reduces noise margins. Low voltage differential signalling can be used as remedy for this problem. For the communication architecture selection, multiple options are available to reduce energy efficiency. The first technique is bus splitting. Bus splitting cuts the connecting bus into smaller pieces which can operate separately but can be reconnected if needed. The cut can be "horizontal" or "vertical", meaning the bus can be split in multiple busses of the same length but with reduced bandwidth or shorter busses with the normal bandwidth. This can yield energy savings between 16 and 50 percent over an unsplit bus. As alternative to busses a router based communication architecture could be used. This has multiple advantages over a bus architecture, also concerning the energy consumption. One advantage is that the components are only connected to the router, and any communication will only need to overcome the capacitance on the connections from sender and receiver to router and not a bus that connects all components. This enables further energy saving techniques like disabling or reducing supply voltage of unused communication links. Also, the router need not be fully interconnected, because in most cases not all components communicate with all other components, which again saves power. The third technique for on-chip communication is the network on chip. This consists of tiles of computing elements, surrounded by network paths. An example for optimising such structures was already given in the discussion of [34] in section 2. System level techniques to reduce power consumption are communication based power management and adaptive supply voltage links. In communication based power management, the power management is done by the communication hardware, since it is already connected to all components and can collect information to regulate the power consumption of the whole system. This eliminates the need for an extra power controlling element. Using this technique one can

achieve a tradeoff between power consumption and performance, 3x longer battery life with half of the performance. Adaptive supply voltage links are dynamic voltage scaling applied to communication links, which can decrease power consumption by 3 times on average. For further information on those techniques see [27] for an extensive list of references.

3.4 Probabilistic Arithmetic

Correctness is considered a binary property of conventional digital circuits, the circuit either produces correct results all the time or not. Probabilistic arithmetic in embedded systems weakens this constraint by allowing computations to be correct only most of the time. In [12], probabilistic arithmetic is used to achieve massive energy efficiency gains. Conventional voltage scaling techniques can only be used to some extent, because if the supply voltage is lowered too much, the signal noise makes correct computations impossible, which adds a probabilistic effect to the computations. One other way to achieve probabilistic behaviour is using a digital circuit beyond its specified frequency, where the delay time between transistors causes nondeterminism. One could now ask why it should be beneficial to reduce the power beyond the barrier set by correctness. The answer is given in [6]. Trading 0.22 percent in probability of correctness results in 23 percent saving in power consumption per switching step. With a reduction of 1.4 percent of probability one can achieve energy savings of 39 percent. The problem lies in utilising this relationship correctly, for example for designing a 32-bit adder. If one operates every bit with the same reduced correctness probability, the adder will have an average error magnitude of about 200 billion [12], which is too much for practical applications. Therefore a technique called biased voltage scaling is developed in [12]. With biased voltage scaling, the entire adder is not operated on the same supply voltage level. The least significant bits are operated with lower voltage, the most significant bits with higher voltage, which produces a distribution of correctness of calculation from 0.8 to 1 from lsb to msb. Biasing the supply voltage has the effect that the average error will be about 55 thousand, with exactly the same energy consumption as in the uniform case. Extending this difference to a practical application, in this case satellite image data reconstruction, one can see that dynamic voltage scaling produces a factor 2.5 reduction in energy consumption, while achieving a signal-to-noise ratio of 0 dB, which means the reconstructed image is unusable noise. Reconstructing the same image with biased voltage scaling results in a factor 5.6 reduction of used energy while achieving a signal-to-noise ration of 28dB, which corresponds to a usable picture. This example shows the usefulness of biased voltage scaling in inherently error tolerant application domains, for example image or audio reconstruction or radar signal interpretation. For a theoretical foundation of probabilistic arithmetic see [4].

4 Software Methods

Energy consumption is not only a design or hardware problem, it also depends on the software running on the hardware, since it makes no sense to for example construct an embedded system with dynamic voltage scaling if the software is so badly designed that the system never enters states of reduced energy consumption. Scientific effort in this area has focused primarily on two things: Energy aware scheduling algorithms for the operating system and energy aware compilers for the code that runs on the embedded system. The scheduling methods concentrate on assigning the right job at the right time to the right processing element at the right speed to meet execution time constraints while minimising energy consumption and switching to and from battery-friendly sleep states. Compiling methods are concerned with transforming user specified code into machine code for the embedded system in such a way as to minimise the power requirements for executing this code. Of course, scheduling and compiling are not the only topics of interest for energy efficient embedded systems. One example is [10], where energy efficient data structures for a gaming application on an internet based embedded system are studied. Nevertheless, the following subsections will cover scheduling and compiling techniques aimed at reducing energy consumption.

4.1 Scheduling

Within the scope of scheduling, two main approaches can be distinguished for reducing the power consumption of embedded systems [17]. They play a significant role in the improvement of the system's energy efficiency by utilising their special mechanisms. The first approach is called dynamic power management. In this context the idle periods of a system, where the device consumes less power, are managed in such a way that the system really requires the minimum amount of energy during those periods. The key observation is, that it does not suffice to simply switch into a low power state every time the processing elements are idle. This is because the switching between normal and low power states itself costs energy and rapidly switching between those states costs more energy than staying in the normal operating state. The second approach is dynamic voltage scaling or dynamic frequency scaling. If the time requirements of a job are known when it is started (or can at least be accurately estimated) and the timing constraints allow it, the job can be executed in a low power state to conserve energy. So, in a way, one can not only schedule jobs but also idle time and power states. Additionally, one can schedule with real time applications, non constant energy sources, task rejection, distributed systems, multi-process or architectures or other parameters in mind. These parameters define a vast array of possible combinations for scheduling considerations, which is reflected by the extensive amount of literature on the subject [39, 7, 41, 46, 14, 1, 29, 13, 40, 8, 5, 45, 9, 31, 33, 15]. All of this approaches can not possibly be discussed in the scope of this term paper, instead we will focus on the techniques introduced in [28] to give just one, but more detailed example of the

possibilities in scheduling. [28] focuses on scheduling techniques for battery powered embedded systems. In this paper it is shown that scheduling in a battery powered environment needs special care, because due to the discharge characteristics of batteries, even scheduling the same jobs at the same speeds but in a different sequence can result in a different battery lifetime. Switching from scheduling power hungry tasks first to scheduling them last costs 1/6th of battery lifetime. Therefore scheduling at different times has different costs, which needs to be reflected by a charge-based cost function. This cost function is then optimised during the scheduling process. The optimisation of the charge-based cost function for scheduling tasks sticks to three key aspects, these are (1) dependency constraints, (2) delay constraints and (3) endurance constraints. The starting point for the optimisation process contains the prediction of the battery lifetime, given a varying load profile. The solution presented in [28] is an adapted battery model, which combines physical justification and analytical simplicity so that a cost function for analytical use can be constructed easily. This approach treats battery-specific load profiles in a formal form, which makes it possible to fine tune the arrangement of tasks and insertion of idle periods to maximise charge recovery or select the best candidate task for voltage reduction. The concrete optimisation strategies are (1) charge minimisation, (2) voltage down-scaling based on highest-power initial solution and (3) voltage up-scaling based on lowest-power initial solution.

Charge minimisation approach: The goal of this strategy is to minimise energy consumption, which corresponds in the regarded cases to the total charge consummation during task execution. The first step is to assign the highest possible voltage to every task . Under this condition the delay constraint is satisfied for sure. Next step is the generation of a corresponding knapsack problem and calculation of an exact solution by performing the MultipleChoiceKnapsack(.). As a result the task for execution are weighted with voltages still meeting the overall delay constraint. Now in the next step the tasks are queued in such a way that the heaviest-weight task and his successor tasks are scheduled first, next the second heaviest-weight task and his successor tasks and so on. Unfortunately energy minimisation is not the guarantee for a maximisation of battery lifetime, since battery lifetime depends not only on task charges, but also on task ordering in time. So it might be possible that the battery doesn't survive the completion of all tasks. If this is foreseeable, a task repair is performed. Task repair checks if there is a failing task and fixes it by voltage down-scaling in correspondence with the insertion of idle periods. After that the procedure LatencyReduction(.) is called to perform voltage-upscaling and assure that the delay constraint is satisfied again.

Voltage down-scaling based on highest-power initial solution: This strategy uses voltage-downscaling to generate a low-cost load profile. First step is the assignment of the maximum voltage to each task, in order to minimise the profile duration. The procedure TaskSequence(.) is called to generate an initial ordering of the tasks. Afterwards TaskRepair(.) in combination with SlackUtilizationMinCharge(.) and AlterSlackUtilization(.) is executed to reduce the voltage costs and improve the solution costs.

Voltage up-scaling based on lowest-power initial solution: This method operates just the other way round as the voltage down-scaling based on highest-power initial solution. First all tasks are assigned the lowest voltage in order to minimise the energy consumption as much as possible. In the next step the battery lifetime is calculated for the corresponding load profile and an initial sequence of tasks is computed via TaskSequence(.). Now there are two possible scenarios. First one is that the scheduled profile meets the delay constraint, but exceeds the battery lifetime. In this case additional idle periods are added to increase battery lifetime within the delay constraint. Second scenario is that the battery survives the task execution, but unfortunately the delay constraint is violated. In this case some tasks must be assigned to a higher voltage by calling LatencyReduction(.).

For an in-depth discussion of the described method see [28].

4.2 Compiling

The compiler is an important component in determining the types, order and number of instructions executed for an application which enables him to have a significant influence on the power consumed by the system. Early studies [36] in the field of compilation based energy reduction have observed some basic mechanisms, which were a first signpost to a deliberate energy policy. One improvement strategy is the reordering of instructions to reduce switching. Central mechanism is the different energy consumption of an instruction depending on the previously executed instruction. A reordering of several sequences of instructions showed an improvement of the overall energy expenditure. Another strategy is the reduction of memory operands. An inspection of the energy cost of memory operands showed that memory instructions cost nearly twice as much as instructions with register operands. Reduction of the number of memory operands can be achieved by suitable compiler policies. The most effective way to do so is to use a better utilisation of registers by obtaining optimal register allocation.

Present works on this field deal with the development of new frameworks supporting the compiler regarding energy efficiency [18] [19] or with intelligent compilation techniques called architecture-aware compilation [21]. The Energy-Aware Compilation Framework (EAC) can estimate and optimise energy consumption of a certain code in combination with the architectural and technological parameters, energy models and energy/performance constraints. A short overview of the EAC-Framework is given in figure 6.

The framework supports the programmers from single energy cost calculations downward to performance optimisation of the whole program on compilation level. At the beginning an analysis of the code is performed to figure out constructs like loops, nested loops, assignment statements, array references and scalar variable references. In a next step the different constructs are weighted and grouped by their energy consumption and similar functionality. These correlations between high-level code and low-level instructions

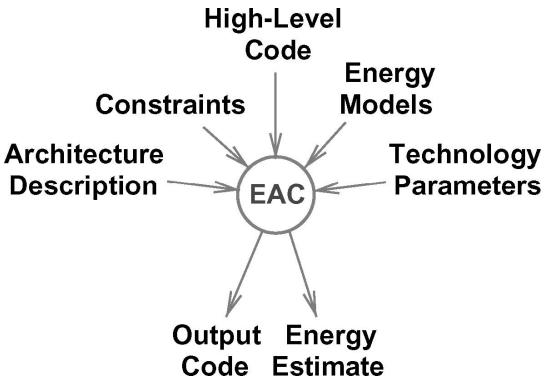


Figure 6: Energy-aware compilation framework as presented in [18]

give a first impression of the energy efforts for a single analysis. Energy optimisation within the EAC-Framework is treated in two different ways. On the one hand there is loop level optimisation and on the other hand also procedure-level energy optimisation. Presented loop-level optimisation can be used in terms of thermal-constrained optimisation, battery capacity-constraint optimisation and multi-constraint optimisation. Picking multi-constraint optimisation, using EAC-Framework makes it possible to generate a table of different tiling strategies with their energy costs for given code constructs. Now individual constraints e.g. limitation of off-chip memory energy or restriction of execution cycles flow in the calculation and lead to an energy-aware code compilation by picking the best instructions under given constraints. Applying such a table strategy, EAC-Framework can also compile a code under a given energy-delay product value.

The other optimisation field of EAC-Framework concentrating on procedure-level energy optimisation is accomplished by employment of integer-linear programming-based procedure-level optimisation. The specific version of ILP implemented by the framework is zero-one ILP, where each integer variable is restricted to be either zero or one during the optimisation process.

The benefits of the EAC-Framework are the extensible problem-specification, by taking description and technology parameters and various constraint into account and the accurate results in connection with an error margin of 6%. The framework can be applied on a wide spectrum of computing environments from high-end servers down to resource-constraint embedded and mobile systems. It provides a rapid evaluation of source-level algorithmic optimisation or it enables software architects to experiment with architectural alternatives to investigate the full capabilities of energy-aware compilation.

5 Conclusion

In the previous sections we have given an overview of the possibilities in designing energy efficient embedded systems. It was shown that in special cases reductions of over 90 percent in energy consumption are possible with the right choices is design methodology, hardware and software. This is an impressive feat, but there is still bad aftertaste, because this whole technological survey misses something: the energy requirements in producing and recycling embedded systems. This is not an accident, there is virtually no literature available on these topics. One possible explanation might be, that it is either difficult to investigate (for optimising the energy requirements of the production of embedded systems some kind of production facility is needed) or not very "interesting" (there is no immediate value in low energy recycling, one can always just throw the system away, though it is bad for the environment, but then again who cares). Reducing the power consumption of embedded systems while they are operating produces an immediate increase in value, because long runtime is an important feature for mobile embedded systems, which translates into increased sales figures. For stationary devices in industrial environments, power consumption equals cost, which reduces profit and, as we all know, companies don't like that. One other facet is, that researchers are not concerned about sustainability when they develop energy efficient algorithms or design methods. This is best illustrated in [42], where the only mentioned downside of high energy consumption is bulky and costly heat dissipation. Energy costs and effect on the environment are not important enough to be mentioned. Then again, this is not necessarily bad, reducing energy consumption is a good thing from a sustainability perspective and the motivation for the research on this topic is only secondary. A problem only arises when the original motivation is gone, for example if power is too cheap to care. Then other sources of motivation for energy efficiency research need to be found, if sustainability is not a concern. Some sorts of regulation by a government might be necessary to generate the necessary incentives, like forbidding energy wasting electrical devices or sensitising the public and corporate management for sustainability issues. Of course, channelling scientific effort toward development of energy efficient embedded systems will always miss one point: the embedded system might not even be needed in the first place. One can argue that an embedded system inside a refrigerator that notifies the user when the milk is running out is a waste, no matter how energy efficient it is. The human mind is quite capable of deducing the need for new milk when it notices the empty milk cartons, no artificial system is needed. All in all, embedded systems need special attention regarding their energy efficiency because of their low profile nature. End users can buy it, plug it in somewhere and forget about it, while it will waste power for years to come. Concerning own ideas for constructing energy efficient embedded systems, we, the authors, cannot contribute much, because we are no experts in the field of hardware design. But there is one thought that should be in the back of the head of every engineer and scientist who just achieved an increase in energy efficiency: Performing calculations is not inherently a process that requires energy like heating or electrolysis. Field-effect transistors, which are the main components of digital circuits,

just require voltage for their operation, not current. Energy is only needed because of the increasing effect of parasite capacities at high switching frequencies. Scientists should always have that in mind and never be satisfied by any increase in energy efficiency.

To sum it all up, more effort is still needed to bring sustainability into the field of embedded systems.

References

[1] Susanne Albers and Hiroshi Fujiwara. Energy-efficient algorithms for flow time minimization. *ACM Trans. Algorithms*, 3(4):49, 2007.

[2] James Balfour, William Dally, David Black-Schaffer, Vishal Parikh, and JongSoo Park. An energy-efficient processor architecture for embedded systems. *IEEE Comput. Archit. Lett.*, 7(1):29–32, 2008.

[3] Jun Cai and Kian-Lee Tan. Energy-efficient selective cache invalidation. *Wirel. Netw.*, 5(6):489–502, 1999.

[4] Lakshmi N.B. Chakrapani, Kirthi Krishna Muntimadugu, Avinash Lingamneni, Jason George, and Krishna V. Palem. Highly energy and performance efficient embedded computing through approximately correct arithmetic: a mathematical foundation and preliminary experimental validation. In *CASES '08: Proceedings of the 2008 international conference on Compilers, architectures and synthesis for embedded systems*, pages 187–196, New York, NY, USA, 2008. ACM.

[5] Ho-Leung Chan, Wun-Tat Chan, Tak-Wah Lam, Lap-Kei Lee, Kin-Sum Mak, and Prudence W. H. Wong. Energy efficient online deadline scheduling. In *SODA '07: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 795–804, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics.

[6] Suresh Cheemalavagu, Pinar Korkmaz, Krishna V. Palem, Bilge E. S. Akgul, and Lakshmi N. Chakrapani. A probabilistic cmos switch and its realization by exploiting noise. *Proc. of IFIP international Conference on VLSI SoC*, 2005.

[7] Jian-Jia Chen, Kazuo Iwama, Tei-Wei Kuo, and Hseuh-I Lu. Flow time minimization under energy constraints. In *ASP-DAC '07: Proceedings of the 2007 conference on Asia South Pacific design automation*, pages 866–871, Washington, DC, USA, 2007. IEEE Computer Society.

[8] Jian-Jia Chen and Tei-Wei Kuo. Procrastination for leakage-aware rate-monotonic scheduling on a dynamic voltage scaling processor. In *LCTES '06: Proceedings of the 2006 ACM SIGPLAN/SIGBED conference on Language, compilers, and tool support for embedded systems*, pages 153–162, New York, NY, USA, 2006. ACM.

[9] Jian-Jia Chen, Tei-Wei Kuo, Chia-Lin Yang, and Ku-Jei King. Energy-efficient real-time task scheduling with task rejection. In *DATE '07: Proceedings of the conference on Design, automation and test in Europe*, pages 1629–1634, San Jose, CA, USA, 2007. EDA Consortium.

[10] E. G. Daylight, T. Fermentel, C. Ykman-Couvreur, and F. Catthoor. Incorporating energy efficient data structures into modular software implementations for internet-based embedded systems. In *WOSP '02: Proceedings of the 3rd international workshop on Software and performance*, pages 134–141, New York, NY, USA, 2002. ACM.

[11] Ronald G. Dreslinski, Gregory K. Chen, Trevor Mudge, David Blaauw, Dennis Sylvester, and Krisztian Flautner. Reconfigurable energy efficient near threshold cache architectures. In *MICRO '08: Proceedings of the 2008 41st IEEE/ACM International Symposium on Microarchitecture*, pages 459–470, Washington, DC, USA, 2008. IEEE Computer Society.

[12] J. George, B. Marr, B. E. S. Akgul, and K. V. Palem. Probabilistic arithmetic and energy efficient embedded signal processing. In *CASES '06: Proceedings of the 2006 international conference on Compilers, architecture and synthesis for embedded systems*, pages 158–168, New York, NY, USA, 2006. ACM.

[13] Lee Kee Goh, Bharadwaj Veeravalli, and Sivakumar Viswanathan. Design of fast and efficient energy-aware gradient-based scheduling algorithms heterogeneous embedded multiprocessor systems. *IEEE Trans. Parallel Distrib. Syst.*, 20(1):1–12, 2009.

[14] Bita Gorjiara, Nader Bagherzadeh, and Pai H. Chou. Ultra-fast and efficient algorithm for energy optimization by gradient-based stochastic voltage and task scheduling. *ACM Trans. Des. Autom. Electron. Syst.*, 12(4):39, 2007.

[15] Shaoxiong Hua, Gang Qu, and Shuvra S. Bhattacharyya. Energy-efficient embedded software implementation on multiprocessor system-on-chip with multiple voltages. *ACM Trans. Embed. Comput. Syst.*, 5(2):321–341, 2006.

[16] International database - world population. http://www.census.gov/ipc/www/idb/worldpopinfo.html, June 2009.

[17] Sandy Irani, Sandeep Shukla, and Rajesh Gupta. Algorithms for power savings. *ACM Trans. Algorithms*, 3(4):41, 2007.

[18] I. Kadayif, M. Kandemir, G. Chen, N. Vijaykrishnan, M. J. Irwin, and A. Sivasubramaniam. Compiler-directed high-level energy estimation and optimization. *ACM Trans. Embed. Comput. Syst.*, 4(4):819–850, 2005.

[19] I. Kadayif, M. Kandemir, N. Vijaykrishnan, M. Irwin, and A. Sivasubramaniam. Eac: A compiler framework for high-level energy estimation and optimization. In *DATE '02: Proceedings of the conference on Design, automation and test in Europe*, page 436, Washington, DC, USA, 2002. IEEE Computer Society.

[20] Dake Liu and C. Svensson. Power consumption estimation in cmos vlsi chips. *Solid-State Circuits, IEEE Journal of*, 29(6):663–670, Jun 1994.

[21] Peter Marwedel, Lars Wehmeyer, Manish Verma, Stefan Steinke, and Urs Helmig. Fast, predictable and low energy memory references through architecture-aware compilation. In *ASP-DAC '04: Proceedings of the 2004 conference on Asia South Pacific design automation*, pages 4–11, Piscataway, NJ, USA, 2004. IEEE Press.

[22] Rui Min, Wen-Ben Jone, and Yiming Hu. Location cache: a low-power l2 cache system. In *ISLPED '04: Proceedings of the 2004 international symposium on Low power electronics and design*, pages 120–125, New York, NY, USA, 2004. ACM.

[23] Sumit Mohanty and Viktor K. Prasanna. A hierarchical approach for energy efficient application design using heterogeneous embedded systems. In *CASES '03: Proceedings of the 2003 international conference on Compilers, architecture and synthesis for embedded systems*, pages 243–254, New York, NY, USA, 2003. ACM.

[24] Anish Muttreja, Anand Raghunathan, Srivaths Ravi, and Niraj K. Jha. Automated energy/performance macromodeling of embedded software. In *DAC '04: Proceedings of the 41st annual Design Automation Conference*, pages 99–102, New York, NY, USA, 2004. ACM.

[25] Viktor K. Prasanna. Energy-efficient computations on fpgas. *J. Supercomput.*, 32(2):139–162, 2005.

[26] Anand Raghunathan, Niraj K. Jha, and Sujit Dey. *High-Level Power Analysis and Optimization*. Kluwer Academic Publishers, Norwell, MA, USA, 1998.

[27] Vijay Raghunathan, Mani B. Srivastava, and Rajesh K. Gupta. A survey of techniques for energy efficient on-chip communication. In *DAC '03: Proceedings of the 40th annual Design Automation Conference*, pages 900–905, New York, NY, USA, 2003. ACM.

[28] Daler Rakhmatov and Sarma Vrudhula. Energy management for battery-powered embedded systems. *ACM Trans. Embed. Comput. Syst.*, 2(3):277–324, 2003.

[29] Youlin Ruan, Gan Liu, Jianjun Han, and Qinghua Li. An energy-efficient scheduling algorithm for real-time tasks. In *ICCS '07: Proceedings of the 7th international conference on Computational Science, Part IV*, pages 965–968, Berlin, Heidelberg, 2007. Springer-Verlag.

[30] Marcus T. Schmitz, Bashir M. Al-Hashimi, and Petru Eles. A co-design methodology for energy-efficient multi-mode embedded systems with consideration of mode execution probabilities. In *DATE '03: Proceedings of the conference on Design, Automation and Test in Europe*, page 10960, Washington, DC, USA, 2003. IEEE Computer Society.

[31] Marcus T. Schmitz, Bashir M. Al-Hashimi, and Petru Eles. Iterative schedule optimization for voltage scalable distributed embedded systems. *ACM Trans. Embed. Comput. Syst.*, 3(1):182–217, 2004.

[32] Huaping Shen, Mohan Kumar, Sajal K. Das, and Zhijun Wang. Energy-efficient data caching and prefetching for mobile devices based on utility. *Mob. Netw. Appl.*, 10(4):475–486, 2005.

[33] Dongkun Shin and Jihong Kim. A profile-based energy-efficient intra-task voltage scheduling algorithm for hard real-time applications. In *In Proceedings of the International Symposium on Low-Power Electronics and Design*, pages 271–274, 2001.

[34] Dongkun Shin and Jihong Kim. Power-aware communication optimization for networks-on-chips with voltage scalable links. In *CODES+ISSS '04: Proceedings of the 2nd IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*, pages 170–175, New York, NY, USA, 2004. ACM.

[35] Intelligent energy standby saver. http://www.reuters.com/article/pressRelease/idUS191572+02-Apr-2009+PRN20090402, June 2009.

[36] Vivek Tiwari, Sharad Malik, and Andrew Wolfe. Compilation techniques for low energy: An overview. pages 38–39, 1994.

[37] Chia-Ying Tseng and Hsin-Chu Chen. The design of way-prediction scheme in set-associative cache for energy efficient embedded system. In *CMC '09: Proceedings of the 2009 WRI International Conference on Communications and Mobile Computing*, pages 3–7, Washington, DC, USA, 2009. IEEE Computer Society.

[38] World energy outlook 2009. http://www.eia.doe.gov/oiaf/ieo/index.html, June 2009.

[39] Haisang Wu, Binoy Ravindran, E. Douglas Jensen, and Peng Li. Cpu scheduling for statistically-assured real-time performance and improved energy efficiency. In *CODES+ISSS '04: Proceedings of the 2nd IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*, pages 110–115, New York, NY, USA, 2004. ACM.

[40] Haisang Wu, Binoy Ravindran, E. Douglas Jensen, and Peng Li. Energy-efficient, utility accrual scheduling under resource constraints for mobile embedded systems. *ACM Trans. Embed. Comput. Syst.*, 5(3):513–542, 2006.

[41] Chuan-Yue Yang, Jian-Jia Chen, and Tei-Wei Kuo. An approximation algorithm for energy-efficient scheduling on a chip multiprocessor. In *DATE '05: Proceedings of the conference on Design, Automation and Test in Europe*, pages 468–473, Washington, DC, USA, 2005. IEEE Computer Society.

[42] Chenjie Yu and Peter Peter Petrov. Aggressive snoop reduction for synchronized producer-consumer communication in energy-efficient embedded multi-processors. In *CODES+ISSS '07: Proceedings of the 5th IEEE/ACM international conference*

on Hardware/software codesign and system synthesis, pages 245–250, New York, NY, USA, 2007. ACM.

[43] Xiangrong Zhou and Peter Petrov. Arithmetic-based address translation for energy-efficient virtual memory support in low-power, real-time embedded systems. In *SBCCI '05: Proceedings of the 18th annual symposium on Integrated circuits and system design*, pages 86–91, New York, NY, USA, 2005. ACM.

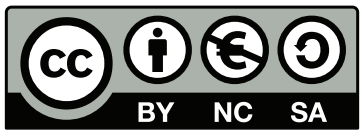
[44] Xiangrong Zhou and Peter Petrov. Direct address translation for virtual memory in energy-efficient embedded systems. *ACM Trans. Embed. Comput. Syst.*, 8(1):1–31, 2008.

[45] Jianli Zhuo and Chaitali Chakrabarti. An efficient dynamic task scheduling algorithm for battery powered dvs systems. In *ASP-DAC '05: Proceedings of the 2005 conference on Asia South Pacific design automation*, pages 846–849, New York, NY, USA, 2005. ACM.

[46] Jianli Zhuo and Chaitali Chakrabarti. Energy-efficient dynamic task scheduling algorithms for dvs systems. *ACM Trans. Embed. Comput. Syst.*, 7(2):1–25, 2008.

About this Document

This thesis is part of the lecture series on *Sustainable Development and ICT*[1] held at Vienna University of Technology[2], Faculty for Informatics[3]. This thesis is part of a selection of submitted student-papers. This paper as well as all other papers[4] are published under the *Attribution-Noncommercial-Share Alike* Creative Commons License[5] to provide a summary/impression from the students-perspective of the seminar for all that are interested in the topic, but could not participate.



[1] http://www.informatik.tuwien.ac.at/events/studium/archiv/161
[2] http://www.tuwien.ac.at
[3] http://www.informatik.tuwien.ac.at
[4] http://bitbucket.org/sdit/sd-ict
[5] http://creativecommons.org/licenses/by-nc-sa/3.0/