Proceedings of the 8th International Conference on Applied Informatics
Eger, Hungary, January 27–30, 2010. Vol. 1. pp. 413–420.

Survey of Dynamic Voltage Scaling Methods for Energy Efficient Embedded Systems

Áron Csendes

University of Szeged, Institute of Informatics, Szeged, Hungary
e-mail:csaron@inf.u-szeged.hu

Abstract

The power consumption of embedded devices is becoming more and more important, thus the energy efficiency needs to be optimized. Today's embedded hardware components (CPU, memory etc.) make it possible to scale both their voltage level and their frequency dynamically in order to achieve optimal energy consumption and meet computation time limitations at the same time.

In this paper we have collected some algorithms that use dynamic voltage and frequency scaling and have set comparison criteria to compare them. The comparison of the methods is done using XEEMU: an improved XScale power simulator.

Keywords: Embedded System, Energy Efficiency, DVFS

1. Introduction to Embedded Systems

1.1. About Embedded Systems

An *embedded system* is a general computer system that is embedded into a special hardware environment that is dedicated to handle special tasks. Quite often some sort of real-time performance constraints apply. The development of the hardware and the software of embedded systems is difficult because special hardware and software tools are necessary for the development.

The embedded systems have increasing importance as the main component of electronic devices. Since for many of these appliances cannot be connected to the mains, the power source can only come in the form of a battery. It is therefore of utmost importance to optimize the battery life of such devices. Given the battery of maximum capacity that can be used with an appliance, the energy consumption of the device determines how long it can be operated without recharging it.

1.2. Embedded System Requirements and Constraints

Embedded systems are in use in many different domains, thus very diverse constraints have to be met.

- *Physical dimensions and circumstances* (size, weight, heat, vibration etc.). These are the most inevitable constraints. The systems have to fit into various equipments and have to withstand adverse conditions. Mobile phones and on-board electronic control units (ECUs) are good examples for this.

- *Reliability.* Embedded systems that are used to control dangerous industrial processes, human heartbeats or automotive systems are among the most reliable ones, and reliability is the major issue.

- *Time frame* (hard or soft real-time operation). The value of the answer or the response that is given by the embedded system may become less after the deadline. If the response has to be delivered before the time limit is reached and it will turn absolutely useless after that time then we have a hard real time constraint. (E.g. airbag controller.) If the answer will have less value after the time limit, but will not be completely useless then that is a soft real time constraint. For example a temporary jitter in case of a mobile phone.

- *Costs.* Even a very small decrease of the manufacturing cost of the hardware will generate a great amount of savings on the long run because of the huge scale of the production.

- *Energy efficiency* is a very important issue because it also has an effect to how a system meets the other criteria. If an embedded system has good energy efficiency it will have better physical dimensional characteristics (smaller battery is needed), it will meet real-time deadlines more easily and will cost less money.

1.3. Some Examples from the Embedded World

- **Industrial control devices**
  Used to control various industrial processes like the assembly of machines, etc. E.g. robot controller.

- **Avionics, on-board electronics**
  Modern vehicles are equipped with lots of electronics: flight instruments, on-board electronic controllers like ABS or airbag controllers are all embedded systems.

- **Aerospace and other vehicles**
  There are many very specialized equipments which are being used in the outer space: satellites, rockets, and landing units all have the properties of the embedded systems.

- **Hand-held devices**
  Mobile phones, personal navigational devices, PDAs and *digital cameras* are good examples of embedded systems of daily use. Good energy efficiency and thus high stand-by times are crucial.
- **IT & multimedia devices**
  Examples are set-top-boxes, modems, even hard disk electronics, PCI cards, video consoles.
- **Medical devices**
  Pacemakers, diagnostic instruments, etc...

2. Power Consumption Measurement

If we would like to optimize the energy consumption of a device, the correct measurement of the energy being used has to be achieved. There are two different approaches to power consumption measurement:

- Measurement using special hardware.
- Simulation using a software.

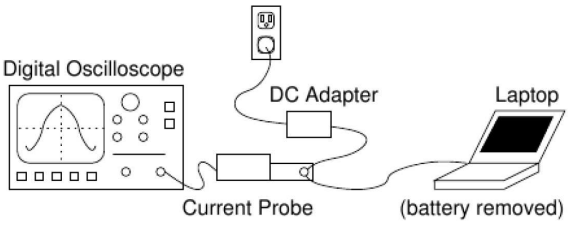

Figure 1: Measurement done by special hardware

The usage of a special measurement device might be very accurate but also laborious. In this case the real target (embedded system) or a PC that simulates it is being used with a measuring equipment that meters electric current that is fed to the target.

On the other hand, if the internal structure and behavior of the CPU and memory of the target is known, it is possible to create an emulation software into which the transistor-level knowledge of the embedded system is built in. This emulation software is capable of counting how many CPU and memory clock-cycles are necessary to complete the computations. By using the specifications of the hardware, the emulator can also calculate how much energy is needed for the computations.

These power usage simulators have to be validated against the real hardware to prove their correct behavior in terms of measuring the energy consumption.

2.1. Power Simulators

A power simulator is a software that has the transistor-level knowledge of the simulated hardware and that can perform a cycle-accurate simulation of the hardware as if that would run the same program code. The energy consumption that the real actual hardware would need, can be calculated, because the need of energy is known for every CPU instruction.

Using power simulators to measure (or rather to estimate) the power consumption of an embedded system has many advantages:

- *No noise.* Unlike real measurements the simulation is not affected by noise and side-effects.
- *Flexibility.* With a power simulator software many different hardware configurations might be tested easily. (E.g. different cache models and sizes or other CPU architectures.)
- *Automation.* Large amount of measurements and experiments with different setups can be made automatically, collecting lots of data.

2.2. XEEMU

The XEEMU is a fast and cycle-accurate power simulator for the Marvel (formerly Intel) *XScale* architecture.

The XScale CPU is an implementation of the ARM v5 processor architecture which is a platform with widespread use for embedded systems. XScale microprocessors are used in many consumer electronics like the popular Blackberry handheld device, Sharp Zaurus, and many other PDAs.

In order to exactly and cycle-accurately model the behavior of a processor pipeline, it is necessary to do the simulation of the CPU on the microarchitectural level. There are already available open source generic processor core simulators e.g. *SimpleScalar*. These kind of processor simulators need to be enhanced to provide dynamic power consumption data. Such power simulators are also already existing, SimplePower, Wattch and XTREM are good examples.

In [1] the XTREM power simulator was validated using the ADI 80200EVB, an XScale-based evaluation board made by ADI Engineering. In the article [1] the authors have addressed some of the weaknesses of XTREM, mainly by providing a more accurate model of the instruction pipeline and an improved simulation of the memory subsystem. Thus XEEMU was developed having energy estimates being accurate within 1.6% in the average case (and within 4.5% in worst case).

3. Voltage and Frequency Scaling

Dynamic voltage and frequency scaling is a widely used technique for managing the CPU performance and throttling power consumption. The term *dynamic* refers to the scaling being done real-time during the execution of the program. By scaling the

frequency of the CPU it is possible to achieve more or less completed instructions per second thus the speed of the CPU can be controlled. However, the processor frequency and the core voltage that is necessary to operate it are proportional:

$$f \propto V$$

so higher processor throughput needs higher core voltage. (The phenomenon behind this relation can be summarized as follows: the processor core has capacitors that need to be filled with charge. If the frequency is higher, then there will be less time to fill in the same amount of charge, so a higher voltage has to be used.)

Frequency (MHz)	**Voltage (V)**
333	0.91
400	0.99
466	1.05
533	1.12
600	1.19
666	1.26
733	1.49

Table 1: Possible frequency and voltage combinations (XScale).

The consumed energy is proportional to the CPU cycles, the capacitance, the voltage squared and the frequency:

$$Energy \propto Cycles \times Capacitance \times Voltage^2 \times Frequency.$$

There is a trade-off to be made: either higher performance or longer battery life. Scaling the frequency and the voltage can be made *at or before compilation time* (static) or *at runtime* (dynamic – DVFS). By regularly polling and setting the core frequency to the minimum level that enables the program to meet its deadlines will give a low power consumption profile.

3.1. DVFS Algorithms

There are several different approaches to this problem, and the algorithm to be used also depends on constraints of the application: e.g. whether it has to run real-time or not.

3.1.1. Non-real-time Algorithms

Workload decomposition

One set of approaches to DVFS is based on the decomposition of the workload of the CPU. By using some heuristics and measuring certain values, it can be

determined dynamically during run-time whether the section of the program that is currently being executed is memory- or CPU-intensive. If the current section is CPU intensive, a higher voltage/frequency setting will be a better compromise in the performance-endurance trade-off.

This approach is used in the algorithm introduced in [2]. The translation look-aside buffer (TLB) hit/miss ratio is used as a heuristic. (The TLB is used in modern CPUs to serve as a cache of the memory management unit to speed up virtual-to-physical memory address translation.) A higher number of TLB misses compared to TLB hits indicates a memory-intensive section of the program: high processor core frequency is disadvantageous. Algorithms using such heuristics need support from the hardware to provide these measurements. On the other hand, the overhead is very small.

Machine learning

A completely different solution is to use machine learning to regularly select the expert that has the best performance. If this approach is used, a set of experts is defined so that each expert sets the CPU frequency/voltage in a different way. Every expert is evaluated at a regular time interval, and the expert that would have had the best result during the last interval will be used next. If the length of the interval is short enough, this algorithm (introduced in [3]) will converge to the optimum.

3.1.2. Real-time Algorithms

The DVFS algorithms are able to make significant energy savings, while enabling the system to provide maximum performance if it is necessary. However, they need to be enhanced to provide real-time guarantees in order to be used in real-time applications. In the paper [4] the authors suggest two real-time scheduling algorithms that use DVFS technique to provide energy efficiency while meeting deadlines:

- cycle-conserving rate monotonic (RM)
- cycle-conserving earliest deadline first (EDF)

Since the activities in a real-time system have to be made at regular intervals, every task has a period. The RM scheduling algorithm always chooses the task to be executed on the CPU that has the shortest period. The minimum frequency (and voltage) is chosen that passes the schedulability test:

$$(\forall T_i \in \{T_1, \ldots, T_n | P_1 \leq \cdots \leq P_n\}) \quad \left\lceil \frac{P_i}{P_1} \right\rceil * C_1 + \cdots + \left\lceil \frac{P_i}{P_i} \right\rceil * C_i \leq P_i$$

where T_i, P_i, C_I are the ith task, period, worst computation time respectively.

The EDF algorithm is different, because it always chooses the task which has its deadline in the nearest future (closest in time). Again the voltage is set to the

lowest possible value that meets the schedulability test of EDF:

$$\frac{C_1}{P_1} + \cdots + \frac{C_n}{P_n} \leq 1$$

These algorithms are called cycle-conserving, because they use the smallest possible core frequency hence they consume the least amount of processor cycles.

3.2. Comparison of DVFS Algorithms

The aforementioned algorithms are based on quite different principles, and also their targeted usage scenarios are somewhat diverse. Thus it makes no sense to make a *complete, pairwise* comparison. Some are not easily or usefully interchangeable, or no comparison by a power simulator can be carried out. However, the comparison of the last two algorithms was done using XEEMU.

3.2.1. Comparison Criteria

As criteria for the comparison, the runtime and the amount of consumed energy have been selected. The deadlines have to be met anyhow. The runtime includes both the overhead of the algorithm and the effective "payload" of the application.

3.2.2. Results

The program that was executed comes with XEEMU as a reasonable example. This was extended to run as virtually multiple tasks so that the RT-DVFS scheduling algorithm can be included.

Algorithm	**RT-DVFS CC-RM**	**RT-DVFS CC-EDF**
Runtime (sec)	0.360	0.370
Energy (Joule)	0.016	0.014

Table 2: Simulation results

The same amount of periods have been made, resulting in slightly different computation times and power usage. These results are in accordance with those described in [4].

4. Conclusions and future work

The energy-efficient processor usage of embedded systems is a topic of increasing importance. In this paper it was shown which methods can be used to measure the power consumption and some of the most relevant algorithms have been presented.

As a possible future work some other comparable algorithms could be searched for, and also the already introduced ones could be compared on real hardware.

References

[1] Zoltan Herczeg, Akos Kiss, Daniel Schmidt, Norbert Wehn, Tibor Gyimothy, XEEMU: An Improved XScale Power Simulator

[2] Kihwan Choi, Ramakrishna Soma, Massoud Pedram, Dynamic Voltage and Frequency Scaling based on Workload Decomposition, *ISLPED'04, August 9-11, 2004, Newport Beach, California, USA*

[3] Gaurav Dhiman, Tajana Simunic Rosing, Dynamic Power Management Using Machine Learning, *ICCAD'06, November 5-9, 2006, San Jose, CA*

[4] Padmanabhan Pillai, Kang G. Shin, Real-Time Dynamic Voltage Scaling for Low-Power Embedded Operating Systems, *SOSP'01, October 21-24, 2001, Chateau Lake Louise, Banff, Canada*

[5] Kihwan Choi, Ramakrishna Soma, Massoud Pedram, Fine-Grained Dynamic Voltage and Frequency Scaling for Precise Energy and Performance Trade-off based on the Ratio of On-chip Access to On-chip Computation Times, *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE'04)*

[6] Gaurav Dhiman, Tajana Sirmunic Rosing Dynamic Voltage Frequency Scaling for Multi-tasking Systems Using On-line Learning, *ISLPED'07, August 27-29, 2007, Portland, Oregon, USA*

Áron Csendes
H-6720 Szeged, Árpád tér 2.