Dynamic Voltage Scaling on a Low-Power Microprocessor

Johan Pouwelse* Koen Langendoen Henk Sips
Delft University of Technology, The Netherlands
{pouwelse,koen,sips}@ubicom.tudelft.nl

Abstract

Power consumption is the limiting factor for the functionality of future wearable devices. Since interactive applications like wireless information access generate bursts of activities, it is important to match the performance of the wearable device accordingly. This paper describes a system with a microprocessor whose speed can be varied (frequency scaling) as well as its input voltage. Voltage scaling is important for reducing power consumption to very low values when operating at low speeds. Measurements show that the energy per instruction at minimal speed (59 MHz) is 1/5 of the energy required at full speed (251 MHz). The frequency and voltage can be scaled dynamically from user space in only 140 µs. This allows power-aware applications to quickly adjust the performance level of the processor whenever the workload changes.

1 Introduction

Today's bulky portable computers will be replaced by small wearable devices in the near future. Wearable devices introduce several challenges that are significantly different from the traditional computing model. From our point of view wearable devices need advanced functionality, low weight, long battery life, and also wireless Internet connectivity. Currently, devices that are sufficiently low weight to be wearable and have a long battery life only offer limited services such as paging and cellular telephony. The goal for the next-generation of wearable devices is to extend the services beyond mere voice, address book, e-mail, and limited computation [9]. The Ubiquitous Communications (UbiCom) project at the Delft University of Technology aims at a wearable device equipped with a long-range 10 Mbps wireless link and an augmented-reality display [24]. The applications within UbiCom are based on wireless access to text, audio, video, and 3D graphics.

*Supported by the Dutch organization for Applied Scientific Research (TNO), Physics and Electronics Laboratory.

Power consumption is becoming the *limiting factor* for the functionality of wearable devices, because advances in battery technology are progressing slowly whereas computation and communication demands are increasing rapidly. It is important to utilize the available energy as efficient as possible. Energy preservation, or energy management, is further translated into a low power consumption of all parts of a wearable device. The initial response to the low-power demand was to lower the supply voltage. For example, reducing the supply voltage from standard 5.0 V to 3.3 V reduces power by 56%. Sophisticated electronic design tools can now deliver optimizations for low power consumption on the transistor level, logical level, or register transfer level [10].

Lowering the supply voltage requires all components to operate at low voltage. Additional reductions can be obtained by selectively lowering the input voltage of specific parts. An obvious candidate is the processor since it is responsible for 10 to 30% of the power consumption [11]. In 1995 Intel introduced the first x86 processor that operated at a lower voltage (2.9 V) than the PC motherboard (3.3 V).

The dynamic approach to low-power is using power down features to minimize the power consumption of unused hardware. For portable computers this means turning off the hard disk, processor, screen, modem, sound, etc. Re-activation of hardware can take some time, which affects performance (e.g., response times). Using simple power-down-when-idle techniques the processor's power consumption can be significantly reduced. Depending on the usage pattern, the power savings can amount to a 66% reduction [12]. A refinement is to make continuous trade-offs between performance and cost. The user demand (performance) must be supplied at the lowest cost (power consumption). Performance can be expressed as the response time for interactive applications, and as spatial/temporal resolution, color depth, and distortion level for video.

The Ubicom system is designed to support balancing continuously power consumption for its wearable components: wireless link (data rate), processor (instruction rate), and rendering (frame rate). In the case of the processor we will apply *voltage scaling* to trade-off power consumption

and performance. Experiments show that in our case it is possible to lower the processor's input voltage from 1.5 V to as low as 0.8 V, dramatically reducing power consumption. This power reduction comes at the price of reduced performance, because proper operation is only guaranteed at a reduced clock frequency. We anticipate a considerable reduction in power consumption since typical Ubicom applications cause a fluctuating processor demand; wearable devices are used interactively by a single user and activities like word processing, reading e-mail, and web browsing generate bursts of processing.

This paper investigates the trade-off between power consumption and performance of processors supporting voltage scaling in detail. Unlike previous studies in voltage scaling that rely on simulations we have realized an actual implementation being part of a wearable computer. This allows us to present raw measurements of the performance/power trade-off in voltage scaling. From these numbers we are able to derive the potential gain in a system where applications assist the operating system in adjusting the voltage by specifying their (bursty) requirements.

2 Voltage scaling

This section introduces the basic principles behind power consumption and the effects of voltage scaling. For digital CMOS circuits the power consumption can be modeled accurately with simple equations [1, 7]. Digital CMOS circuits are used in the majority of microprocessors. CMOS circuits have both dynamic and static power consumption. Static power consumption is caused by bias and leakage currents and is insignificant in most designs that consume more than 1 mW.

The dominant power consumption for CMOS microprocessors is the dynamic component. Every transition of a digital circuit consumes power, because every charge or discharge of the digital circuit's capacitance drains power. The dynamic power consumption is equal to

$$P_{dynamic} = \sum_{k=1}^{M} C_k \cdot f_k \cdot V_{DD}^2 \quad (1)$$

where M is the number of gates in the circuit, C_k the load capacitance of gate g_k, f_k the switching frequency of g_k, and V_{DD} the supply voltage. It is clear from Equation (1) that reduction of V_{DD} is the most effective mean to lower the power consumption. Lowering V_{DD}, however, creates the problem of increased circuit delay. An estimation of circuit delay is given by

$$\tau \propto \frac{V_{DD}}{(V_G - V_T)^2} \quad (2)$$

where τ is the propagation delay of the CMOS transistor, V_T the threshold voltage, and V_G the input gate voltage [1]. The propagation delay restricts the clock frequency in a microprocessor. From Equations (1) and (2) we can see that there is a fundamental trade-off between switching speed and supply voltage. Processors can operate at a lower supply voltage, but only if the clock frequency is reduced to tolerate the increased propagation delay. When we assume the dynamic power is the most dominant one, and the gates g_k of the microprocessor form a collective switching capacitance C with a common switching frequency f, we obtain

$$P = C \cdot f \cdot V_{DD}^2 \quad (3)$$

Equation (3) shows that clock frequency reduction linearly decreases power and voltage reduction results in a quadratic power reduction. The critical path of a processor is the longest path a signal can travel. The implicit constraint is that the propagation delay of the critical path τ must be smaller than $\frac{1}{f}$. In fact, the processor ceases to function when V_{DD} is lowered and the propagation delay becomes too large to satisfy internal timings at frequency f.

2.1 Power versus performance

To put the above formulas in perspective Table 1 gives the relation between frequency, voltage and power consumption for the recently announced Transmeta TM5400 or 'Crusoe' processor [16]. Crusoe is one of the few processors today that actually support voltage scaling. It is interesting that its specifications give insight in the practical constraints (e.g., propagation delays) that relate frequency and voltage. At the lowest frequency (200 MHz) the Crusoe processor operates at 29% of the maximum speed for less than 13% of the maximum power. Without voltage scaling the power would be reduced to only 29%. Therefore, voltage scaling effectively more than halves (13/29) the energy per processor instruction. This result looks better than it actually is, however, since performance of a (wearable) system is not determined by the processor clock speed alone.

First, the processor is only a part of the total system. Consequently, the benefits of voltage scaling are to be weighted against the power consumed by the remainder of the system. This issue will be addressed in Section 4.2. Second, application performance is a combination of processor speed and memory access latency. The performance of the memory subsystem is not linearly related to clock frequency, so application performance does not scale linearly too. Third, certain applications like video players have real-time deadlines to meet. This limits the possibilities to apply voltage scaling by reducing the clock frequency, because the time needed to complete a task increases proportionally. A simple heuristic suggested by Weiser et al. [2] is to distinguish foreground tasks, background tasks, and periodic

Frequency f (MHz)	Voltage V_{DD} (V)	Relative power (%)
700	1.65	100
600	1.60	80.59
500	1.50	59.03
400	1.40	41.14
300	1.25	24.60
200	1.10	12.70

Table 1. Clock frequency versus supply voltage for the Transmeta Crusoe processor.

tasks, and to only apply voltage scaling when all foreground tasks are blocked by I/O or run idle.

In a wearable system power consumption is the obvious cost measure since only a small amount of energy can be carried in batteries. This suggests that reducing power consumption, even for small periods of time, is always a good thing. Unfortunately, batteries do not operate efficiently when the power demand has high peaks. These peak occur when for example the processor is frequently switched on and off to 'reduce' power consumption. An equal demand that is evenly spread out in time can increase battery life by about 25% according to [23]. Therefore, in some cases it may be more efficient to slow down the clock, instead of switching to sleep mode when idle. Consequently, the popular believe that "*slowing the clock is useless if voltage is kept constant*" [13] no longer holds.

3 Implementation

Voltage scaling has been primarily studied through simulation, see the related work discussed in Section 6. The simulation results are promising, so time has come to validate the technique. Within UbiCom we assembled a wearable computer that supports voltage scaling to do just that, but also to experiment with application-directed voltage-scaling policies (Section 5) and to make voltage scaling ready for wide spread use. Therefore, we needed a processor capable of voltage scaling, but very few actually exist; the ARM8 implementation by Berkeley [6] is not readily available, and the Crusoe processor [16] has only been introduced very recently. This is remarkable since Weiser et al. already in 1994 showed the potentials of voltage scaling for general purpose processors [2]. We selected the embedded StrongARM 1100 processor [17] that does support frequency scaling, but only operates at 1.5 V according to the datasheet. Operating outside the specifications is a risk, but experiments show that a range from 0.8 V to 2.0 V is feasible.

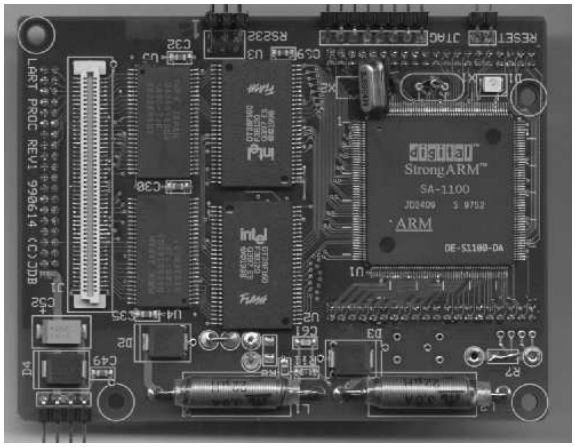

Figure 1. Low-power StrongARM embedded Linux platform (LART).

3.1 Experimental platform

The embedded StrongARM processor board displayed in Figure 1 forms the heart of the wearable augmented-reality terminal that we are developing within UbiCom [20]. The board, which we call LART, has a size of 10x7.5 cm, a weight of 50 gr, 32 MB of volatile memory, 4 MB of non-volatile memory, a SA-1100 190 MHz processor, and various I/O capabilities. The LART has a programmable voltage regulator for the processor voltage.

The LART runs under control of the Linux operating system (Version 2.2.12), which has been modified to support frequency and voltage scaling. We added a kernel module that changes the clock frequency and subsequently recalibrates the kernel's internal delay routines, in particular those that busy-wait by counting instruction cycles. In addition, the kernel module adjusts the memory parameters that control the read/write cycles on the external bus. The code has been structured such that may be interrupted and it does not depend on external memory, which is temporarily unavailable during a clock frequency change.

All the LART design schematics and kernel modules are openly available [21].

3.2 Measurement setup

To measure the power consumption of the LART, we used the configuration shown in Figure 2. The unregulated power of a battery is converted into a fixed 3.3 V for all the components on the board, except the processor. The processor voltage is supplied by a variable regulator. The accuracy of the measurements is within 2%.

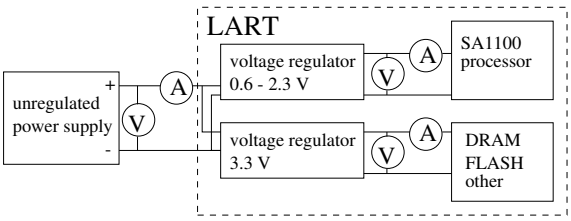

Figure 2. Measurement setup.

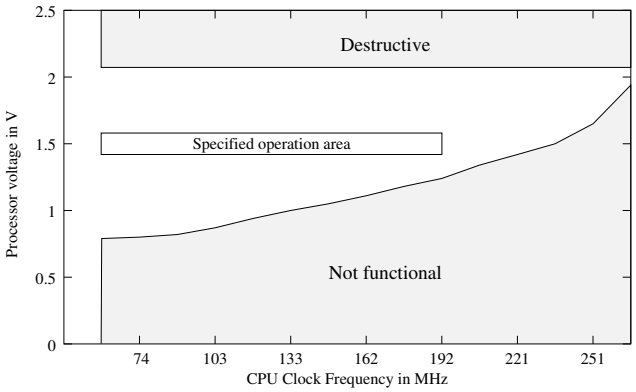

Figure 3. Processor envelope.

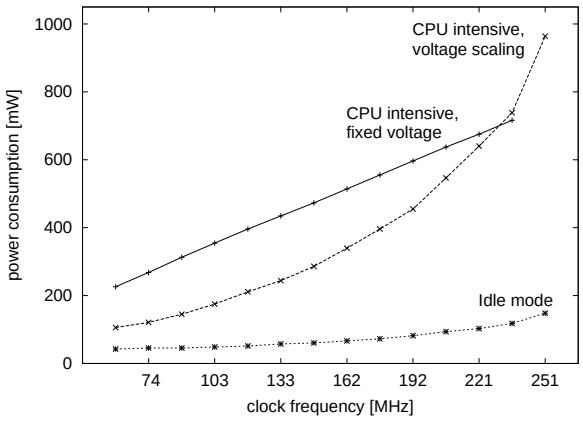

Figure 4. Total power consumption for idle and cpu-intensive workloads.

4 Results

This section describes the effects of frequency and voltage scaling on the power consumption, processor performance, and memory performance of the LART. We ran several micro benchmarks (dhrystone, lmbench) as well as a full-blown H.263 decoder. During each run the clock frequency and voltage were kept constant; dynamic voltage scaling is discussed in Section 5.

4.1 Required voltage

The first experiments determine for each clock frequency the minimum required voltage at which the SA-1100 processor still functions proper. We used the H.263 decoder to check if the processor functions at a given frequency and voltage combination. The resulting processor envelope is shown in Figure 3. Although the SA-1100 is specified for 190 MHz and a V_{DD} of 1.5 V, it can be over-clock up to a frequency of 251 MHz with a V_{DD} of 1.65 V. The minimum clock frequency at which the processor functions correctly is 59 MHz with a V_{DD} of 0.79 V. Voltage scaling really pays off: at the lowest clock frequency the processor consumes 1/5 of the energy per instruction that is required at peak performance. Note that the voltage range and, hence, the efficiency gain of the SA-1100 is much larger than that of the Crusoe processor (see Table 1).

4.2 System performance

The next experiment determines the impact of voltage scaling on the power consumption of the complete LART (processor, memory, etc). Figure 4 shows the total power consumption of the LART under two different workloads: idle and cpu-intensive.

The idle workload was used to measure the background power consumption of the LART, which is always spent regardless of the processor load. The Linux scheduler puts the processor into idle mode when no processes are active. Idle mode stalls the CPU clock, but other services of the embedded processor such as the memory controller and OS-timer are still functional [17]. All these services are driven by the processor clock, which explains why the power consumption in idle mode increases with the frequency. The SA-1100 also supports a more efficient sleep mode, but this mode interrupts DMA transfers, stops the LCD controller, blocks memory access, etc. and the wake-up sequence takes longer than in idle mode.

The cpu-intensive workload consists of the dhrystone benchmark exercising the CPU and cache that operate on the variable core voltage. We first measured the effect of scaling the clock frequency while keeping the voltage constant at 1.5 V. In this case the power consumption increases roughly linear with the frequency, as is expected. Next, we measured the power consumption if the core voltage was set to the minimal value reported in Figure 3. The resulting curve shows the expected quadratic increase of power consumption when the frequency is varied from 59 to 251 MHz.

From the power consumption at 59 MHz (105.8 mW) and at 251 MHz (963.7 mW) it follows that an instruction at peak performance consumes a factor 2.1 more power than at lowest performance. When we only look at the processor

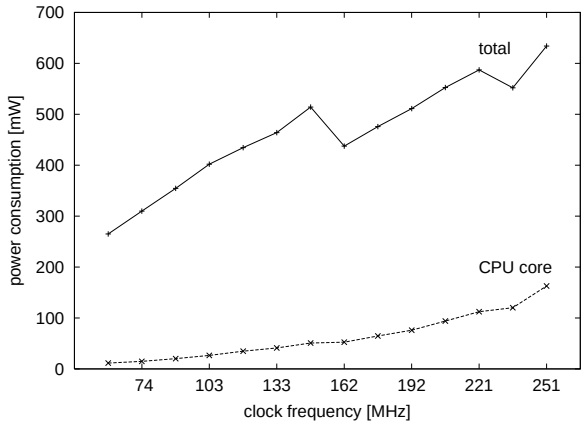

Figure 5. Power consumption for read.

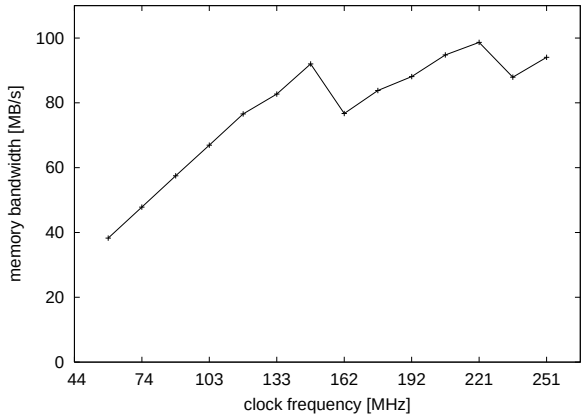

Figure 6. Memory bandwidth for read.

power consumption, instead of the total LART, the difference is a factor 4.94 (33.1 versus 696.7 mW). This observation only holds for cpu-intensive applications; memory references introduce other effects as discussed next.

4.3 Memory performance

The impact of memory references on power consumption is difficult to predict since memory always operates at 3.3 V, while the processor core operates at a variable voltage. The LART has 32 MB of EDO-DRAM with an access time of 60 ns. We used the "lmbench" toolkit [19] to measure the power consumption while reading randomly from memory, circumventing the cache. The power consumption numbers in Figure 5 include voltage scaling, as is the case in all following figures. Note the general trend of a linear increase in power consumption, but with break downs occurring at 162 MHz and 236 MHz.

These break downs are not caused by the processor; the lower curve in Figure 5 plots the power consumption of the

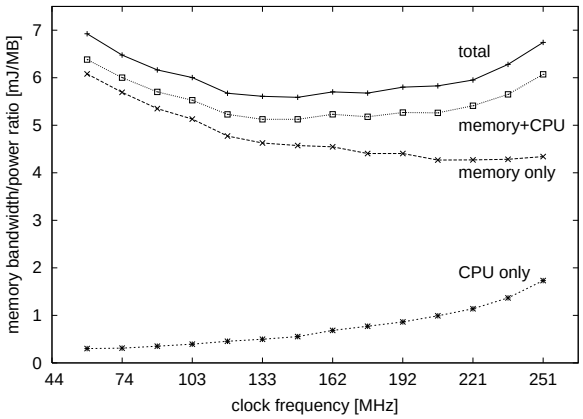

Figure 7. Energy breakdown for memory read.

processor core only. Figure 6 shows that the obtained bandwidth at each frequency has similar dips. The explanation of this phenomenon is the limited capability of the StrongARM to generate high resolution DRAM timing waveforms. Moreover the DRAM waveform generator is driven by the system clock. The memory timing waveforms must be programmed by specifying a bit sequence that is used at subsequent clock pulses. Since the length of a clock pulse is not too fine compared to the (constant) DRAM timings, an optimal waveform can not be generated at each frequency.

It is instructive to combine the bandwidth and power consumption curves to show the relative cost at each frequency. Figure 7 gives the energy required to read one megabyte from memory. The "memory only" curve represents the energy drawn from the fixed 3.3 V, and shows that reading memory becomes relatively cheaper when the frequency increases. The "total" curve is the energy drawn from the batteries and includes both the memory, CPU activities, and voltage regulators. Initially the total energy drops, just as the "memory only" curve, but at higher frequencies the power consumed by the CPU increases considerably and forces the total energy to rise again. The difference between the "memory+cpu" curve and the "total" curve is the constant loss factor of the two voltage regulators. The best result is obtained at 148 MHz, where the bandwidth is 92 MB/s and the power consumption is 514.2 mW, with a cost of 5.6 mJ/MB.

4.4 Application performance

The power consumption of applications depends on the ratio between instructions and memory references. Figure 8 shows the power consumed by a publicly available Telenor H.263 video decoder in relation to the clock frequency. It also gives a breakdown in processor and memory power-consumption. At low frequencies the decoder is memory bound; at high frequencies the processor dominates.

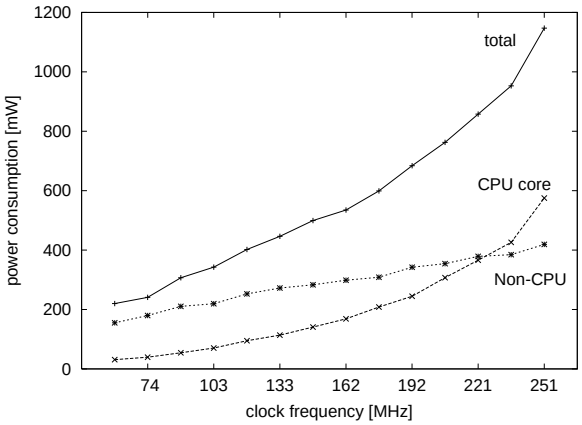

Figure 8. Power breakdown for H.263 decoder.

5 Dynamic voltage scaling

The results from the previous section were obtained under static conditions. In a real system, however, the frequency and voltage have to be set dynamically to match the changing demands for processing power. This is the responsibility of the Operating System (OS). The difficulty is that the OS has no direct knowledge about the workload generated by (bursty) applications, and must derive the optimal settings from external observations, for example, by monitoring the system load and estimating the future demand. This is a non-trivial task; for the Mac OS it is already hard to determine when no useful computation is occurring [12].

Predicting the future workload from the current situation is difficult, and errors can seriously reduce the gains of voltage-scaling as observed in several simulation studies [2, 3, 4, 5]. These simulations use an interval-based scheduler with a time window of 5 to 100 ms. When idle time is detected within a window the clock speed is reduced proportionally in the next window. This reduces the idle-time to zero by running the applications as slow as possible. Whenever some tasks do not meet a deadline, the speed is increased to accommodate for the unfinished work in the next window. With this scheduling policy the speed always lags behind the demand. This is no problem for fairly constant workloads, but poor schedules result for bursty applications. For example, simulations with an MPEG decoder show that an additional 36 % energy reduction remains possible with a better workload prediction [5].

5.1 Power-aware applications

To overcome the above problems we propose that applications provide additional information about their future demands to the OS, so it does not need to work with questionable predictions. The drawback of this approach is that it requires modifications of the application source. This is, however, only required for bursty applications and, more importantly, applications running on a resource-scarce wearable device must be modified anyway. It is generally accepted that limitations imposed by low weight, small size, extensive battery life, wearable user interfaces, and wireless connectivity all have a profound effect on applications [9]. Without extensive adaptation of the applications to the wearable environment, *no* valuable service can be given, therefore modification of applications is inevitable.

Applications must be made aware of their processing demands and inform the OS about it, so the optimal processor speed can be selected that minimizes power consumption and still meets the application's deadlines. As a first step the application could indicate the required number of clock cycles (instructions) to the next deadline (absolute time). Combining the cycle count with the power-consumption curve in Figure 4 allows the OS to compute the lowest speed at which this application could meet its deadline. When multiple applications are time sharing the processor, the OS should take all constraints (deadlines) into account. A refinement is to have the applications express their demands in both instructions and memory references, which would yield better power consumption approximations.

Many real-time applications are periodic. In that case the applications can do better than informing the OS about the next deadline, by giving the time between successive tasks and their length. This is in line with the Quality of Service specifications used to state the required service from a network connection like ATM. QoS specifications do well in stable environments, but applications running on wearable devices will adapt to external conditions and change their behavior, hence, their processing demands will fluctuate. Nevertheless, applications should be able to estimate the workload in the near future.

As an example consider the MPEG decoder studied in [5]. The decoding time of the video frames varies between 8 and 116 ms. Fortunately, it is quite easy to estimate the processing requirements of a specific frame using a model like the one described in [15]. This allows for a setup where the MPEG decoder informs the OS about the requirements of each frame, and the OS sets the processor speed accordingly.

5.2 System support

How much additional energy can be saved by making applications power aware depends, amongst others, on the ability of the system to quickly follow the changes in demand. To assess the responsiveness of the LART we added voltage scaling to the kernel module already providing frequency scaling. Whenever the frequency is changed, the module now also sets the input voltage to the minimum level

reported in Figure 3. The required frequency is read from /dev/proc, which is writable from user space. The kernel module was also modified to generate a pulse on an output pin just *before* changing the frequency and voltage, and another pulse on another output pin just *after* these changes.

We used a digital oscilloscope to measure the time required to change to a new frequency/voltage. The two upper lines on the scope image in Figure 9 show the pulses marking the beginning and the end of a clock/voltage change. The time difference is 140 μs. This time is insensitive to the frequency, and is needed to stabilize the internal clock at the new frequency.

The bottom line on the scope (labeled 3) shows how the processor-core voltage regulator responds to a sharp increase, followed by a similar decrease in clock speed. The benchmark starts with 59 MHz at 0.8 V, then jumps to 221 MHZ at 1.5 V, stays at this level for about 1 ms, and finally returns back to 59 MHz at 0.8 V. The required processor-voltage increase is rapidly handled (40 μs), but the decrease takes a long time (5.5 ms). This is caused by the high capacitance of the regulator and the low power demand of the processor at 59 MHz. The long delay, however, does not consume much power since the current drawn from the voltage regulator is so little.

Since a frequency/voltage change takes little time (140 μs) power-aware applications can change performance levels quite frequently without incurring a large overhead. For example, an MPEG player displaying a video with 30 frames per second will not notice the overhead when setting the processor speed at each frame (i.e., once every 33 ms); the overhead is well below 1%.

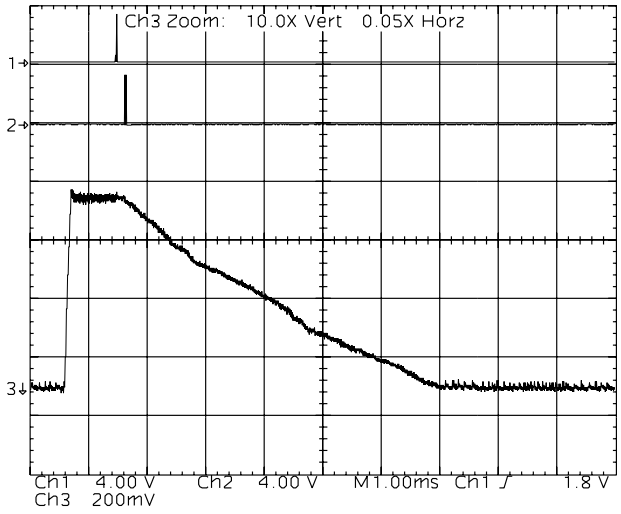

Figure 9. Scope image of voltage scaling.

6 Related work

Low-power is an important issue in traditional laptops and, consequently, is mentioned in industry standards like the Advanced Configuration and Power Interface (ACPI) standard [22]. ACPI, however, does not mention frequency or voltage scaling as such, but uses the term "*clock throttling*". The extensive standard is vague about the exact meaning, but a compliant implementation could use voltage scaling. For now, laptops simply switch off hardware.

The opportunities of varying CPU speed (frequency) and voltage scaling for reducing power consumption in general purpose processors have mainly been explored by theoretical analysis [1] and simulation [2, 3, 4, 5]. Practical experience, as can be obtained with our LART, is needed to validate their results. For example, the experimental research by Martin showed that a basic assumption underlying many simulation studies is wrong: reducing power consumption is not always the best strategy when taking battery effectiveness into account [23]. We are not aware of any practical implementation of voltage scaling including operating system support, except maybe the recently announced Crusoe processor [16]. An interesting approach to dynamic voltage scaling is discussed in [8], where the voltage is automatically adapted when the processor speed is changed. At the Compaq laboratories the Itsy system is under development [18]. It is quite similar to our LART, but the use of linear voltage regulators inside the Itsy will seriously limit the power savings that can be obtained.

The effectiveness of dynamic voltage scaling critically depends on the operating system being able to determine the optimal processor speed. Predicting the future demands from the current load is not effective for bursty applications as demonstrated by the simulations mentioned above. We require applications to be power aware; using applications hints is also suggested, but not explored in [5]. Noble et al [14] suggest to extend the notion of Quality of Service from the traditional communication networks to processor scheduling: applications performance is to be measured in SPECint95 units and the cost in battery minutes.

7 Conclusions and future work

Power consumption is the limiting factor for the functionality of future wearable devices. Since interactive applications like wireless information access generate bursts of activities, it is important to match the performance of the wearable device accordingly. A popular approach is using power-down modes to minimize the power consumption of unused hardware like disks, screen, etc. In the case of the processor, better results can be obtained by scaling the speed and voltage to match the required performance level, since power consumption is quadratically related to the input voltage. Simulations have pointed out the potentials of voltage scaling.

Within the UbiCom project we have assembled a wearable system that supports dynamic voltage scaling. It is based upon the low-power embedded SA-1100 processor, whose frequency can be varied from 59 MHz to 251 MHz. The required input voltage varies from 0.8 V to 2.0 V. Measurements show that the power consumption of the processor at 59 MHz is 33.1 mW, while it consumes 696.7 mW at 251 MHz. It follows that the energy per instruction at minimal speed is 1/5 of the energy required at full speed. This result confirms the importance of voltage scaling.

We have added kernel support to Linux that allows running applications to scale the frequency and voltage from user space in only 140 μs. This allows power-aware applications to quickly adjust the performance level of the processor whenever the workload changes. For example, a bursty application like an MPEG decoder can use a model to estimate the processing requirements for each frame. Calculations indicate that with the small delay (140 μs) to set the performance level, the overhead can be less than 1%.

Our future plans are to specify a processor-independent dynamic voltage-scaling framework as an enhancement of the ACPI standard, so that the voltage scaling technique can become available to a broad audience. Within UbiCom we are developing a QoS framework such that the processor can export its performance/power-consumption trade-off to higher layers. When other components, such as the radio transceiver, also export their performance/power-consumption trade-offs, intelligent decisions can be made at the application level. For example, an audio-streaming application can choose between little compression (low processor load) and a large data stream over the radio, and the alternative scenario of heavy compression and low bit rate. Which is best depends on the current external circumstances that influence the radio.

Acknowledgements

We would like to thank Jan-Derk Bakker and Erik Mouw for providing us with an excellent low-power platform, and assisting us with the measurements and their interpretation. We thank Hylke van Dijk for commenting on the draft version of this paper.

References

[1] T. Ishihara, H Yasuura, “Voltage scheduling problem for dynamically variable voltage processors”, ISLPED, Aug. 1998.

[2] M. Weiser, B. Welch, A. Demers, S. Shenker, “Scheduling for reduced CPU energy”, OSDI, Nov. 1994.

[3] K. Govil, E. Chan, H. Wasserman, “Comparing algorithms for dynamic speed-setting of a low-power CPU”, Mobicom, Nov. 1995.

[4] Y. Lee, C.M. Krishna, “Voltage-clock scaling for low energy consumption in real-time embedded systems”, 6th Int. Conf. on Real-Time Computing Systems and Applications, 1998.

[5] T. Pering, T. Burd, R. Brodersen, “The simulation and evaluation of dynamic voltage scaling algorithms”, ISLPED, Aug. 1998.

[6] T. Pering, T. Burd, R. Brodersen, “Dynamic voltage scaling and the design of a low-power microprocessor system”, ISCA, 1998.

[7] T. Burd, R. Brodersen, "Processor design for portable systems" Journal of VLSI Signal Processing, Aug/Sept 1996.

[8] T. Kuroda, et.al., “Variable supply-voltage scheme for low-power high-speed CMOS digital design”, IEEE Journal of Solid-State Circuits, Vol. 33, No. 3, March 1998.

[9] O. Angin, A.T. Campbell, M.E. Kounavis, R.R.F. Liao, “The MobiWare Toolkit: Programmable Support for Adaptive Mobile Networking”, IEEE Personal Communications Magazine, Aug. 1998.

[10] J. Frenkil, “Tools and Methodologies for Low Power Design”, Proc. Design Automation Conference, 1997

[11] J. Lorch, “A complete picture of the energy consumption of a portable computer”, Masters Thesis, University of California at Berkeley, Dec. 1995.

[12] J.R. Lorch, A.J. Smith, “Scheduling techniques for reducing processor energy use in MacOS”, Wireless Networks, No. 3, 1997.

[13] J.R. Lorch, A.J. Smith, “Software strategies for portable computer energy management”, IEEE Personal Communications, Jun 1998.

[14] B.D. Noble, et.al., “Agile application-aware adaptation for mobility”, 16th ACM Symposium on OS principles, Saint-Malo, France, Oct. 1997.

[15] D. Raychaudhuri, D. Reininger, M. Ott, “Multimedia processing and transport for the wireless personal terminal scenario”, VCIP 95, SPIE Int. Society for Optical Engineering, taipei, Taiwan, May. 1995

[16] Transmeta corporation, “TM5400 processor specifications”, Jan. 2000.

[17] Intel, “Intel strongarm 1100 microprocessor developers manual”, Aug 1999.

[18] Compaq research, “Itsy V2 overview slides”, Jan 2000.

[19] L. McVoy, “Lmbench: portable tools for performance analysis”, USENIX, Jan. 1996.

[20] J.A. Pouwelse, K. Langendoen, H.J. Sips, “A feasible low-power augmented-reality terminal”, 2nd International Workshop on Augmented Reality, Oct. 1999.

[21] J.D. Bakker, J.A.K. Mouw, “Linux Embedded Radio Terminal design page”, http://www-ict.its.tudelft.nl/~erik/open-source/LART/

[22] Intel, microsoft, Toshiba, “Advanced Configuration and Power Interface Specifications”, revision 1.0b, Feb. 1999.

[23] T. Martin, “Balancing batteries, power, and performance: system issues in CPU speed-setting for mobile computing”, PhD. dissertation, Carnegie Mellon University, Aug. 1999.

[24] R.L. Lagendijk, “UbiCom Technical Annex”, Delft University of Technology, Jan. 2000, http://www.ubicom.tudelft.nl.