Energy Consumption Estimation in Embedded Systems

Vasilios Konstantakos, Alexander Chatzigeorgiou, *Member, IEEE*, Spiridon Nikolaidis, *Member, IEEE*, and Theodore Laopoulos, *Senior Member, IEEE*

***Abstract*—This paper presents an energy consumption modeling technique for embedded systems based on a microcontroller. The software tasks that run on the embedded system are profiled, and their characteristics are analyzed. The type of executed assembly instructions, as well as the number of accesses to the memory and the analog-to-digital converter, is the required information for the derivation of the proposed model. An appropriate instrumentation setup has been developed for measuring and modeling the energy consumption in the corresponding digital circuits.**

***Index Terms*—Current measurement, embedded systems, energy complexity, power estimation.**

I. Introduction

THE MARKET for embedded systems in microelectronics continues to increase. Many applications are executed on platforms that incorporate embedded microprocessors, memory units, and peripherals. Although performance still remains a basic design target, energy consumption has become a critical factor for such systems, particularly after the explosion of the portable devices market. This situation has resulted in the requirement for an accurate estimation of the energy consumed by the system for executing certain tasks. Although appropriate methods have been developed and certain models have been provided for the estimation of the energy consumption of embedded processors [1], [2], there have been few efforts published up to now on the modeling of the energy consumption of a complete system, and practically none of them is directly related to low-power instrumentation applications.

In [1], the problem of modeling the energy consumption of a processor is examined for the first time. An appropriate energy budget is assigned to each instruction based on measuring the average current of the processor when it executes the specific instruction. The energy overhead (inter-instruction effect) that results from the changing of the processor state (as it executes one instruction after the other) is also modeled. The energy consumption for the execution of a software task is calculated by summing the individual budgets of each executed instruction and the inter-instruction effects. Another approach to modeling the energy consumption in embedded pipelined processors based on measuring the instantaneous current of the processor has been developed in [2]. This approach is found to present better accuracy and uses an instrumentation setup [3] that can be applied to create models for other components, like memories and peripherals.

An energy consumption model for a system composed of a processor, an instruction, and a data memory has been presented in [4]. This approach aims to define the energy complexity of a program in a way that is analogous to the computational complexity. A polynomial expression of the number of executed assembly instructions (approximately mapping the number of primitive operations in the algorithm) and the number of memory accesses to the data and instruction memory is extracted by analyzing the program under study. Since the energy consumption of an independent assembly instruction can be measured, and each access to the memory has a known energy cost, an estimate of the system's energy consumption is obtained as a single polynomial with appropriate coefficients.

A similar approach is used in [5], where the program whose energy consumption is to be analyzed is represented by its unique control flow graph. Since the control flow graph of any structured program can be constructed by simply sequencing or nesting simpler graphs, an overall energy metric can be obtained by a hierarchical measure. By defining measures of the executed instruction count and the memory access count for primitive graphs and the operations of sequencing and nesting, a relatively simple software energy metric is extracted for any graph. Such a metric can be further used to compare algorithms in terms of their energy consumption.

In [6], a complete methodology for estimating the energy consumption of a processor that executes software tasks has been proposed. This methodology refers to in-order pipelined processors that correspond to the most popular architecture used in embedded applications. A model has been derived for the ARM7TDMI processor with an accuracy of up to 5%. This approach is based on instruction-level energy models. The energy budget of each instruction and the energy overhead observed from executing one instruction after the other (the inter-instruction effect) are estimated based on real measurements of the instantaneous current of the processor. To make the model reasonable, an appropriate grouping of the instructions has been applied. Hence, instructions with similar characteristics that contribute the same amount to the energy budget are located in the same group.

A more generic approach for estimating the energy consumption of component-based distributed systems is presented in [9], specifically for estimating the energy of Java-based software components that run on heterogeneous mobile devices. A theoretical approach that studies methods of algorithms for

Manuscript received July 15, 2007; revised November 7, 2007.

V. Konstantakos, S. Nikolaidis, and T. Laopoulos are with the Electronics Laboratory, Department of Physics, Aristotle University of Thessaloniki, 54124 Thessaloniki, Greece (e-mail: vkonstad@auth.gr).

A. Chatzigeorgiou is with the Department of Applied Informatics, University of Macedonia, 54006 Thessaloniki, Greece.

Digital Object Identifier 10.1109/TIM.2007.913724

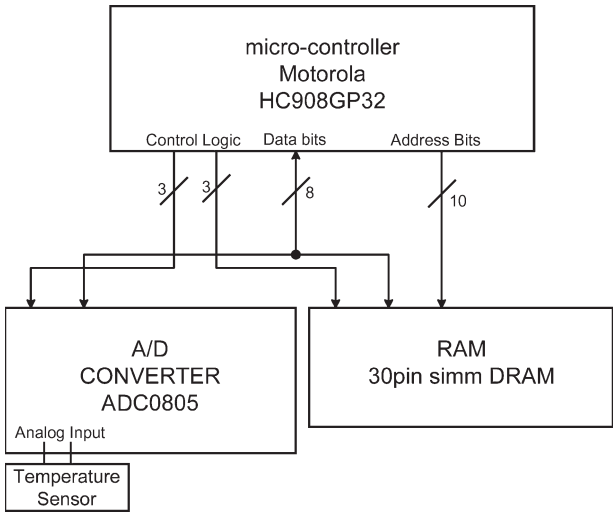

Fig. 1. Target platform.

energy purposes is presented in [10]. In [11], a methodology is presented to perform automated characterization-based high-level software macromodeling. This technique aims at a significantly reduced simulation duration for performance/energy estimation with the difficulty of creating and using accurate software macromodels. Another technique is presented in [12], which is related to the architecture level on the development of a system. The work in [12] is based on detecting sections of the code that are characterized by a high temporal locality in the execution profile of the program being run on a target processor. Moreover, in [13], the design exploration of a system-on-chip simulation-based power-aware architecture, which is dedicated to the implementation of the HIPERLAN/2 communication protocol, is presented. For this purpose, an *ad hoc* C++ simulation environment has been used (which includes proper power models for CPUs and peripherals and incorporates software profiling capabilities). In addition, another power characterization effort of embedded systems is presented in [14].

The proposed approach (which is an extended version of [8]) is differentiated in the sense that it targets a general-purpose microcontroller that is usually employed for logging applications in low-power instrumentation systems. The microcontroller is connected to an A/D converter for input retrieval and to an external RAM for storing data. As such, the architecture, as well as the operation of the microcontroller, is expected to have power consumption that is not analogous to that of typical embedded processing units. Moreover, the proposed model is not solely based on the number of executed instructions but rather on their number of cycles, the activity of the A/D converter, and the number of memory accesses. Due to the nature of the microcontroller, the energy consumption is estimated not by simply profiling the program to be executed but by parsing the program and analyzing the segments that initiate read/write accesses to the memory module and access to the A/D converter.

II. System Architecture

The target platform (on which measurements have been performed) is shown in Fig. 1. The general-purpose microcontroller utilizes a data bus of 8 bits (the number of bits used for read/write is programmable) and an address space of 1024 $\times$ 1024 B. The control logic consists of six signals, i.e., three for controlling the memory and three for the A/D converter.

This platform is a basic configuration of a wide range of instrumentation systems that are used in low-power applications. Similar components can be used in these applications, for example, D/A converters, input components like matrix keyboards, and output components like seven-segment digits or liquid-crystal-display monitor screens. Generally, this is a basic configuration of any system used for data-logging applications. The characteristics of these applications are the nonintensive use of peripheral components. Usually, scheduled measurements are conducted with the use of the A/D converter. These measurements are processed by the microcontroller, and the results of the processing are stored in the memory at the end. Hence, neither the A/D converter nor the memory is intensively used. Furthermore, the memory module is not used during the processing period because the microcontroller has a small amount of internal memory that may be used for the duration of the calculations. The aforementioned configuration is quite simple but is also quite indicative of such applications. The method that will be presented here may be easily applied to any microcontroller-based application.

III. Proposed Modeling Approach

In this paper, we propose an approach for modeling the power consumption of an embedded system and an instrumentation setup for the evaluation of its accuracy. According to the system components presented in the previous section, the total energy consumed by the system when it executes a software task can be expressed by

$$E_{\text{system}} = E_{\text{microcontroller}} + E_{\text{RAM}} + E_{\text{ADC}} \qquad (1)$$

where $E_{\text{microcontroller}}$ corresponds to the energy consumed by the microcontroller while executing a program, E_{RAM} corresponds to the energy consumed by the memory, and E_{ADC} corresponds to the energy consumed by the A/D converter. Each component of the total energy will be analyzed below. The energy model for each system component has been based on separate physical measurements of the drawn current.

For the evaluation of the accuracy of the method, the measurement of the actual energy consumption is needed. Every component has its power supply pins connected to the measuring device so that the drawn current is available for measuring.

This is something that cannot be done by using a simple ampere meter, e.g., like the one used in some instruction-level modeling approaches [1]. In addition, a measuring system that is capable of accurately measuring the energy consumed between certain clock cycles, which correspond to the beginning and the end of the execution of the software task, is required. Such a circuit has been proposed in [7] and is illustrated in Fig. 2.

This measuring device is based on the integration of the current drawn by the system. The current of the system, as it is reflected by a current mirror, is accumulated in a capacitor pair.

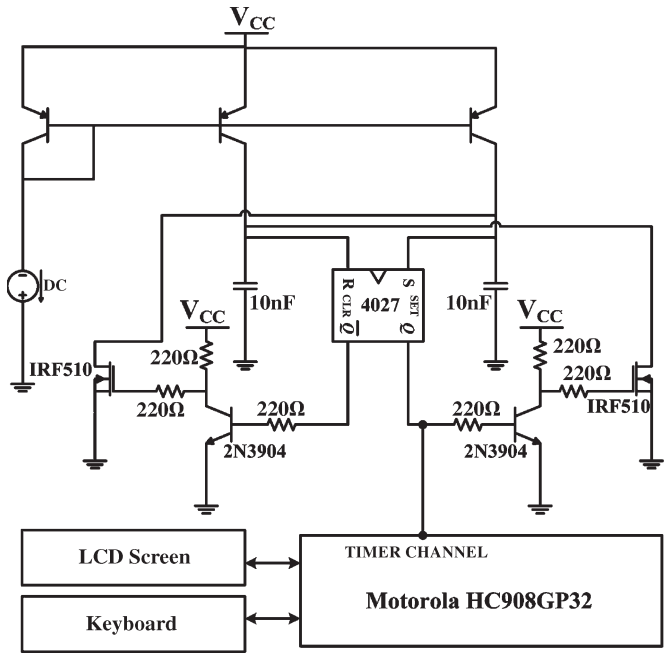

Fig. 2. Energy meter.

When the first of the two capacitors is full, it starts discharging, and the current is directed to the second one, which begins charging. This way, no current amount is lost. The capacitor fills are counted for a specific time window, and the overall energy consumption can be easily calculated. This specific time window is defined by the system under evaluation. A set/clear signal can be controlled by the system under evaluation that will indicate to the measuring system the beginning and the end of a specific instruction execution. Thus, the measuring system will be able to know when an instruction is executed and monitor the capacitors only for this time window. When the energy cost of an instruction is required, then this instruction can be placed to repeatedly run so that an adequate time window will be available for the measuring system. Except for the evaluation of the methodology accuracy, actual measurements are necessary to assign appropriate values to the coefficients that will be mentioned in the proposed model. Experimental measurements were conducted to investigate the power behavior of the mentioned components.

A. Modeling the Energy Consumption of the Microcontroller

Concerning the energy consumption of the microcontroller, the measurements have shown that the energy does not depend that much on the instruction type but rather on the number of required instruction cycles. The typical measurement results are summarized in Table I. This table was created by measuring many different instructions and by taking into account different addressing modes and various arguments. In each situation, an average value is calculated by taking into consideration measurements with the same number of cycles, and the average deviation for each group of instructions is presented. The average deviation is indicative of the difference of the energy consumption between instructions that appear within the same group. The energy variation between the different groups that appear on this table seems to be small. Nevertheless, these energy values (and variations) are actually referring to the energy per cycle consumption. When considering multicycle instructions, the estimated energy will be a product of this value by the number of cycles of each instruction.

TABLE I
AVERAGE ENERGY CONSUMPTION OF MICROCONTROLLER INSTRUCTIONS

Instruction Group	Coefficient	Average Energy per cycle (μJ)	Average Deviation
1 – cycle	cp1	0.0535	0.53%
2 – cycles	cp2	0.0566	2.85%
3 – cycles	cp3	0.0585	5.50%
4 – cycles	cp4	0.0580	3.19%
5 - cycles	cp5	0.0576	1.34%

TABLE II
AVERAGE ENERGY CONSUMPTION DURING MEMORY OPERATIONS AND STEADY-STATE CURRENT VALUE

Memory operation	Coefficient	Average Energy (μJ)	Average Deviation
read access	cm1	16.2	0.29%
write access	cm2	16.6	0.29%
refresh	cm3	3.40	
steady-state current: 8.43 mA			

According to this observation, the proposed model is essentially a function of the number of executed instructions of each instruction group with a different number of cycles, i.e.,

$$\begin{aligned} &E_{\text{microcontroller}} \\ &= f(\#\text{instructions1-cycle}, \#\text{instructions2-cycles}, \\ &\qquad \#\text{instructions3-cycles}, \#\text{instructions4-cycles}, \\ &\qquad \#\text{instructions5-cycles}) \\ &= \text{cp1}.\#\text{instructions1-cycle} + \text{cp2}.\#\text{instructions2-cycles} \\ &\quad + \text{cp3}.\#\text{instructions3-cycles} + \text{cp4}.\#\text{instructions4-cycles} \\ &\quad + \text{cp5}.\#\text{instructions5-cycles} \end{aligned} \tag{2}$$

where the constants $\text{cp1}, \ldots, \text{cp5}$ correspond to the average energy for each instruction group.

B. Modeling the Energy Consumption of the Memory

The energy dissipation of the memory consists of the active energy cost for each read/write access, the standby energy consumed due to the steady-state current flowing through memory cells, and the energy dissipated during each memory row refresh. According to the measurements that have been performed, constant average values for each read and write access can be considered without significant loss in accuracy. Each refresh operation can also be characterized by a standard energy cost. Concerning the steady-state current, its contribution to the overall energy should be calculated as the integral of power during the execution time of the software tasks after subtracting the periods during which read/write accesses are performed. However, for the sake of simplicity, and because the memory accesses in typical logging applications are rare compared to the total execution time, the proposed model integrates the standby power for the complete period of execution. The characterization measurement results are summarized in Table II. For the creation of this table, many different memory

TABLE III
AVERAGE ENERGY CONSUMPTION DURING A/D CONVERTER ACCESS AND IDLE CURRENT

A/D conv. access	Coefficient	Avg. Energy (μJ)
read access	ca	0.3
idle current: 0.912 mA		

accesses were measured by taking into account various addresses and data to be exchanged with memory. Again, the average deviation is calculated by taking into consideration many different accesses and is indicative of the difference of the energy consumption between these accesses.

The corresponding energy model can be formulated as

$$\begin{aligned} E_{\mathrm{RAM}} &= f(\#\text{read\_accesses}, \#\text{write\_accesses}, \#\text{refreshes}) \\ &= \mathrm{cm1}.\#\text{read\_accesses} + \mathrm{cm2}.\#\text{write\_accesses} \\ &\quad + \mathrm{cm3}.\#\text{refreshes} \\ &\quad + V_{dd}.i_{\text{steady-state}}.T_{\text{clock-period}}.\#\text{cycles} \end{aligned} \tag{3}$$

where $\mathrm{cm1}, \mathrm{cm2}$, and $\mathrm{cm3}$ correspond to the average energy for the read, write, and refresh operations, respectively, V_{dd} is the power supply voltage, $T_{\text{clock-period}}$ is the selected clock period, and #cycles is the total number of cycles that correspond to the execution of the software tasks. It should be mentioned that a certain piece of code is used for memory access, and to locate the segments of the code in which the read/write/refresh operations are performed, an appropriate parser should be used.

C. Modeling the Energy Consumption of the A/D Converter

Finally, the energy consumption of the A/D converter also consists of an active component that corresponds to the energy consumed for an access to retrieve data and a standby component that corresponds to the current flowing through the converter when it is idle. Since the access periods are a very small portion of the overall execution time (for the type of applications that have been considered) and because of the relatively limited time of an access (29 μs), the energy due to the idle current is calculated by integrating the power for the total execution time. It should be noted that the energy consumption for an access heavily depends on the transition that has to be made on the A/D converter (from the previously sampled value to the current value). However, as already explained, due to the small number of accesses (compared to the total execution time of the microcontroller), a constant energy value has been considered, as shown in Table III.

The corresponding energy model is

$$\begin{aligned} E_{\mathrm{ADC}} &= f(\#\text{accesses}) \\ &= \mathrm{ca}.\#\text{accesses} + V_{dd}.i_{\mathrm{idle}}.T_{\text{clock-period}}.\#\text{cycles} \end{aligned} \tag{4}$$

where ca corresponds to the average energy for one access to the A/D converter.

For deriving the energy consumption of a software task, someone has to simply sum the aforementioned energy budgets for each system component, which means that the application trace file is required to make estimations. However, the use of the trace file is not always convenient because of its large size and because of the need to compile and run the whole application using a processor simulator. By contrast, the profiling methods (static and dynamic) should be used at the assembly code or, even better, at the higher language description level (e.g., C) of the application to extract appropriate information to guide the estimation procedure. Considering the models presented, such information corresponds to the number of executed instructions (for each instruction group), the number of memory accesses and refresh operations, and the number of A/D converter accesses. The proposed methodology flow for the estimation of the energy consumed in a system executing a given software task is shown in Fig. 3.

IV. EXPERIMENTAL MEASUREMENTS

The next step is to try to measure an actual microcontroller-based system. For this purpose, we implemented the platform that is illustrated in Fig. 1. A temperature sensor is connected to the input of the A/D converter. The microcontroller asks for a temperature measurement once every second and stores this measurement to the external memory. Furthermore, once this measurement is taken, it calculates the average values for the time duration of the last 5 min, 10 min, 30 min, and 1 h.

The microcontroller stands idle until 1 s is completed, and then, a routine that does all the actions mentioned above is executed. When this routine begins to execute, a specific pin of the microcontroller is switched from 0 to 1, and when this routine ends, the same pin is switched from 1 to 0. This is a way of informing the measuring system of the precise execution time window of the routine in each repetition. We must note here that this duration is not the same at each execution of the loop because the duration of the calculation of the average values is not the same in each repetition. Hence, the time duration of each loop is measured to have comparable measurements. This time duration is obtained with the help of the measuring microcontroller's timers.

Since the measurements are performed by monitoring the power supply of each desired module, different measurements can be realized by monitoring a different combination of the power supplies of the modules in the target platform. Each measurement provides an average current value that can help us characterize the consumption in each different repetition. Four different waveforms were taken for a typical case of operation and are shown in Fig. 4.

The first waveform provides the average current value that is measured for the routine executed in each second by monitoring the power supply of the complete platform. Specifically, the first 800 s are being measured because we want to observe at least the first 600 s where additional average calculations are being added as the time passes. The second waveform is the average current value for each second for the memory module only, the third waveform for the microcontroller module only, and the fourth waveform for the A/D module only.

At the first 60 s, the loop takes one measurement in each execution and stores it in the memory. This is done by making one A/D access and one RAM write access. After 60 s, it begins to calculate the average value for the last 1 min by making one more RAM read access. At the first waveform in Fig. 4, we

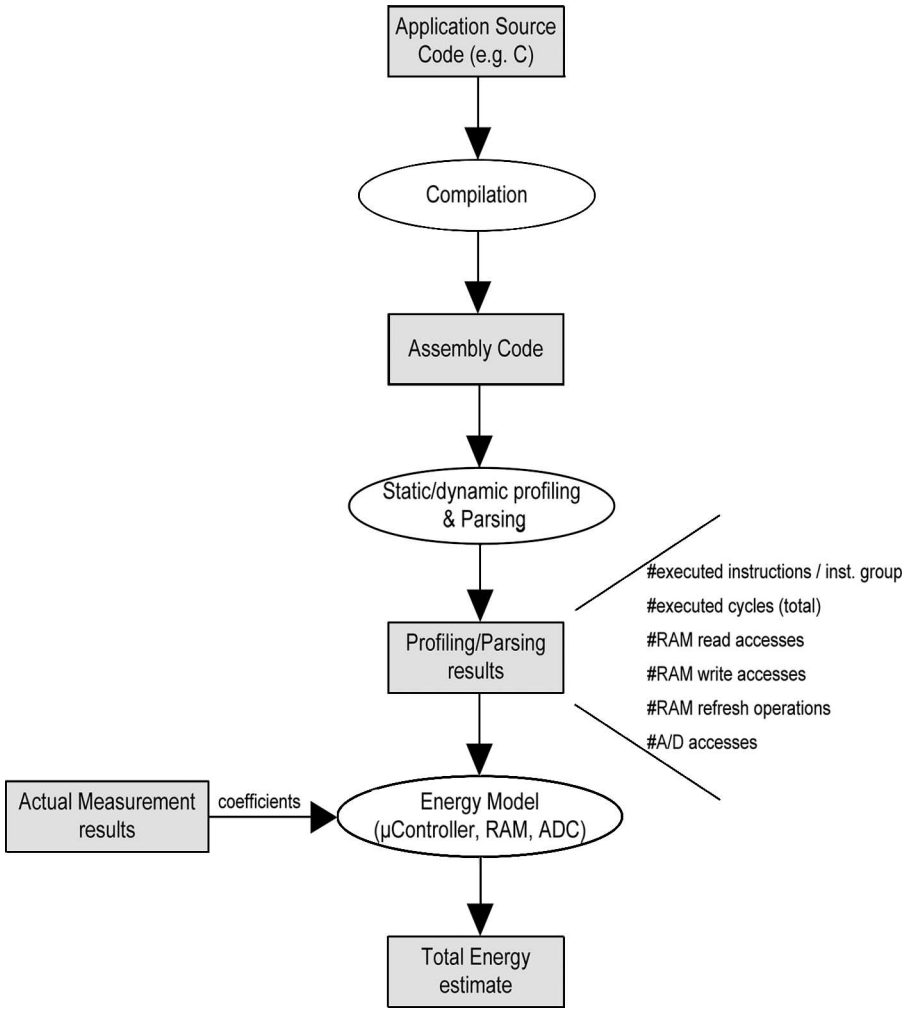

Fig. 3. Energy consumption estimation flow for embedded systems based on a microcontroller.

observe a small increase step at the 60th second that is related to the one more read RAM access being held. A same step is obvious in the 300th second, where one more read access is made to calculate the 5-min average. The same behavior is also obvious in the second waveform of Fig. 4, where only the memory module is being measured. We observe the same small steps at the same seconds. The third figure is quite different as the microcontroller is being evaluated. Here, we observe an almost steady current that is slightly increasing in time because of the more intensive processing that takes place as the time passes. At the fourth waveform of Fig. 4, an almost-steady current is measured for the A/D converter since the variation shown is less than 1%.

These measurements provide us with useful information about the power behavior of this logging system. There is a significant contribution of the memory module and of the number of accesses on this module. The microcontroller shows small variations as time passes, and so does the A/D converter. The behavior of these modules, as described until now, is a characteristic one for such systems that are based on a microcontroller and are built for data-logging instrumentation applications.

This information can also provide built-in self-test abilities to the same system. The evaluation of the aforementioned values can indicate some boundaries concerning the maximum and the minimum power consumption while specific tasks are being executed. After that, if a power measurement on these same tasks results in a number outside these boundaries, then a possible faulty operation can be detected. The ability to measure specific software routines or each module separately can indicate at the end not just a simple faulty operation but also specific information about the cause that is responsible for this faulty operation.

Another calculation is made to verify the good prediction of the energy consumption according to the proposed model. A measurement is made for the first second of the aforementioned algorithm, and according to the average current that is calculated, an energy consumption estimate is extracted for this specific second. Then, the code executed on this second is profiled, and specific parameters like the number of instructions that last a specific number of cycles or the number of RAM write and read accesses are extracted. The idea is to define all the parameters that appear on (2)–(4) to be able to calculate an energy estimate with the help of these equations. The measurement resulted in an energy consumption of 1281 μJ, while the calculations resulted in an energy consumption of 1274 μJ. The deviation between these two values is below 1%, which is quite satisfactory, despite the fact that the proposed model is rather simple.

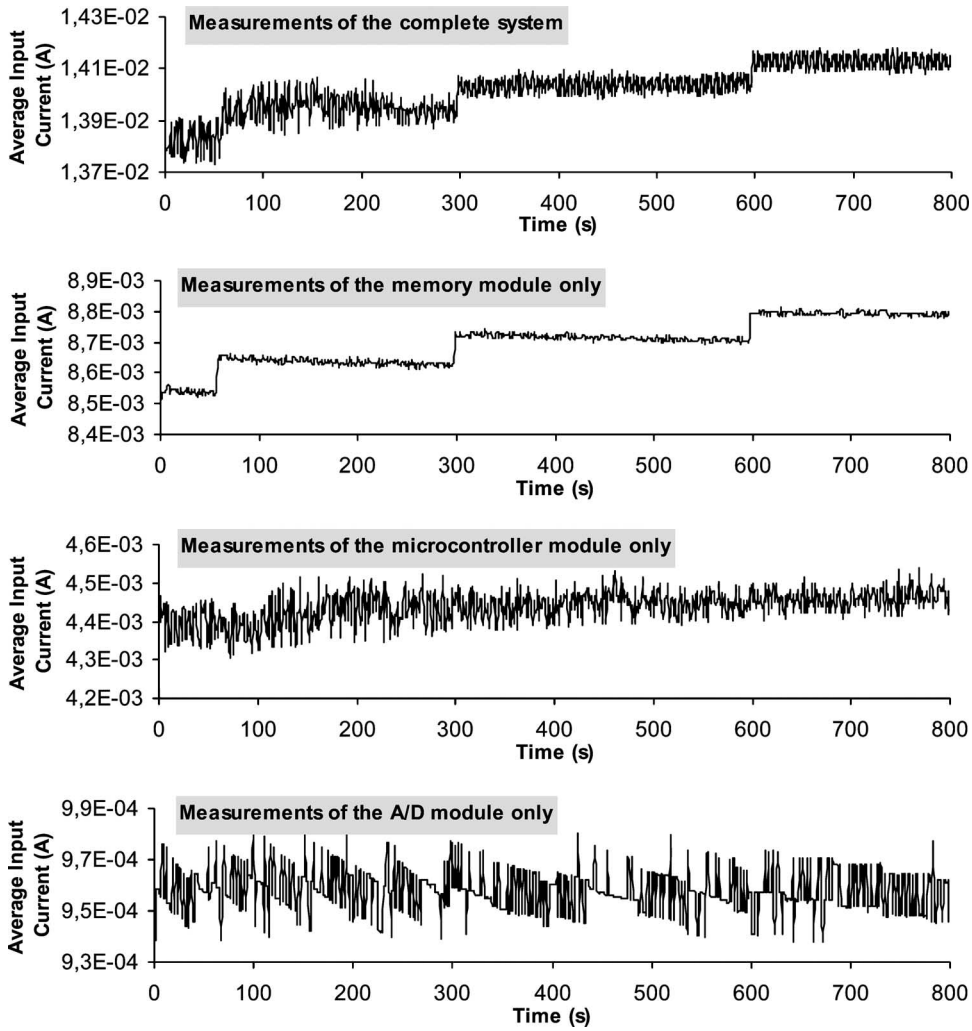

Fig. 4. Measurements on a data-logging application.

TABLE IV
POWER ESTIMATION AND MEASUREMENTS ON THE TARGET PLATFORM

Target Module	Estimated energy by model (uJ)	Measured energy (uJ)	Deviation
Complete system	1274	1281	0.5%
Memory	797.4	794.5	0.4%
Microcontroller	391.7	410.2	4.5%
A/D converter	85.3	87.6	2.6%

If we separately take into consideration each module, then similar results can be obtained, as seen in Table IV. According to the measurements, the energy consumed for this same second by the memory module is 794.5 μJ (it is 410.2 μJ for the microcontroller and 87.6 μJ for the A/D converter), while according to the proposed model for each of these modules, the estimated energy consumption is 797.4 μJ for the memory module, 391.7 μJ for the microcontroller, and 85.3 μJ for the A/D converter. These results indicate a smaller error in the memory module and in the A/D converter, where the model is rather simple, and a bigger error in the microcontroller, where the simplified proposed power model is rather not adequate enough for creating more accurate, results. Nevertheless, the model has provided values that are close to the real measurements. The resulting deviations are a good indication that the creation of power models based on measurements can produce satisfactory results.

The aforementioned measurements and calculations are related only to the first execution of the routine. In a different execution, after the 60th second, this routine will be changed to some degree. If we take a look at the executed code, then we will notice that some functions are expected to last the same duration, and some functions are about to change in duration within each loop execution. The software being measured contains several functions that initiate the modules, which are specific in duration. Moreover, the functions that access the A/D and memory modules do not change in duration. The only part of the program that changes in duration is the part that calculates the average values. Some of these changes can be completely known. For example, the number of memory accesses may change in time (at the maximum of one write access and three read accesses), but this number can be known at the profiling procedure for every loop execution. It should also be noted that one access is being made to the A/D converter in every loop execution, and this does not change in time.

Nevertheless, the microcontroller is only an 8-bit system that conducts complicated calculations (like divisions) with the help of multiple additions and subtractions. Hence, in each

calculation of an average value, the duration of the calculation changes because different numbers are being processed. Moreover, because each calculation depends on numbers obtained at runtime, which cannot be known in the profiling procedure, the calculation routine duration may vary at some degree. If we are to define some upper and lower limit values for this part of the program that changes in each repetition, the difference will not be significant because the number of cycles of this changing part is small compared to the total number of cycles (about 150 cycles of difference over the 23 000 cycles of the total routine duration).

Similar calculations like the ones presented for the first second can be held for each of the rest of the seconds. On each occasion, a profiling procedure is being held on the software, and different accesses to the peripheral modules, as well as different numbers of instructions executed on the microcontroller, are extracted.

V. Conclusion

In this paper, an approach for modeling the energy consumption of embedded systems built around a microcontroller, a memory, and an A/D converter has been proposed. The software tasks that are executed by the microcontroller are analyzed by a profiling procedure, and appropriate information about the activations of the system components and the instructions used is obtained. An appropriate instrumentation setup is employed for modeling the energy behavior of each component in the system both during its active state and during its standby operation. The total energy consumption is obtained by adding the individual energy amounts of these components. An energy meter circuit has been designed for the evaluation of the accuracy of the proposed approach. The measurements indicated that in such systems, the most important energy factor is the number of accesses to memory, while the contribution to the total energy consumption of the A/D converter and the microcontroller itself is less important. The proposed approach can be characterized as a useful tool for analyzing software routines in low-power applications and to a wide range of similar implementation systems.

Acknowledgment

This paper has been accomplished within a bilateral collaboration project between Greece and Ukraine supported by the Greek General Secretariat of Research and Technology.

References

[1] V. Tiwari, S. Malik, and A. Wolfe, "Power analysis of embedded software: A first step towards software power minimization," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 2, no. 4, pp. 437–445, Dec. 1994.

[2] N. Kavvadias, P. Neofotistos, S. Nikolaidis, K. Kosmatopoulos, and T. Laopoulos, "Measurement analysis of the software-related power consumption in microprocessors," *IEEE Trans. Instrum. Meas.*, vol. 53, no. 4, pp. 1106–1112, Aug. 2004.

[3] T. Laopoulos, P. Neofotistos, K. Kosmatopoulos, and S. Nikolaidis, "Measurement of current variations for the estimation of software-related power consumption," *IEEE Trans. Instrum. Meas.*, vol. 52, no. 4, pp. 1206–1212, Aug. 2003.

[4] K. Zotos, A. Litke, A. Chatzigeorgiou, S. Nikolaidis, and G. Stephanides, "Energy complexity of software in embedded systems," in *Proc. IASTED Int. Conf. Autom., Control Appl. (ACIT-ACA)*, Novosibirsk, Russia, Jun. 20–24, 2005.

[5] A. Chatzigeorgiou and G. Stephanides, "Energy metric for software systems," *Softw. Qual. J.*, vol. 10, no. 4, pp. 355–371, Dec. 2002.

[6] S. Nikolaidis, N. Kavvadias, T. Laopoulos, L. Bisdounis, and S. Blionas, "Instruction level energy modeling for pipelined processors," *J. Embed. Comput.*, vol. 1, no. 3, pp. 317–324, Aug. 2005.

[7] V. Konstantakos, K. Kosmatopoulos, S. Nikolaidis, and T. Laopoulos, "In-chip configuration for monitoring power consumption in microprocessing systems," in *Proc. IEEE Int. Workshop Intell. Data Acquisition Adv. Comput. Syst.: Technol. Appl.*, Sep. 2005, pp. 156–161.

[8] V. Konstantakos, A. Chatzigeorgiou, S. Nikolaidis, and T. Laopoulos, "Energy consumption estimation in embedded systems," in *Proc. Instrum. Meas. Technol. Conf.*, Sorrento, Italy, Apr. 2006, pp. 235–238.

[9] C. Seo, S. Malek, and N. Medvidovic, "A generic approach for estimating the energy consumption of component-based distributed systems," Univ. Southern California, Center Softw. Eng., Los Angeles, CA, Tech. Rep. USC-CSE-2005-506, Apr. 2005.

[10] A.-F. Wang, X. Li, T. Lei, and X.-H. Zhou, "Study on methodology of algorithms for energy," *Comput. Eng. Appl.*, vol. 42, no. 29, pp. 100–102, 2006. 106.

[11] A. Muttreja and A. Raghunathan, "Automated energy/performance macromodeling of embedded software," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 26, no. 3, pp. 542–552, Mar. 2007.

[12] V. S. P. Rapaka and D. Marculescu, "Pre-characterization free, efficient power/performance analysis of embedded and general purpose software applications," in *Proc. Des., Autom. Test Eur. Conf.*, Mar. 2003, pp. 504–509.

[13] F. Menichelli, M. Olivieri, L. Benini, M. Donno, and L. Bisdounis, "A simulation-based power-aware architecture exploration of a multiprocessor system-on-chip design," in *Proc. Des., Autom. Test Eur. Conf. Exhib.*, Paris, France, Feb. 2004, vol. 3, pp. 312–317.

[14] A. Mohsen and R. Hofmann, "Characterizing power consumption and delay of functional/library components for hardware/software co-design of embedded systems," in *Proc. 15th IEEE Int. Workshop Rapid Syst. Prototyping*, Jun. 2004, pp. 45–52.

Vasilios Konstantakos received the Diploma in radioelectrology in 2005 from Aristotle University of Thessaloniki, Thessaloniki, Greece, where he is currently working toward the Ph.D. degree.

He has published a small number of papers in conference proceedings. His main field of research is on the techniques for power estimation measurements as well as the estimation of the good operating condition of digital processing systems. He is familiar with measuring systems that combine the use of a personal computer along with the use of modern digital equipment connected to it.

Alexander Chatzigeorgiou (M'95) received the Diploma in electrical engineering and the Ph.D. degree in computer science from Aristotle University of Thessaloniki, Thessaloniki, Greece, in 1996 and 2000, respectively.

From 1997 to 1999, he was a Telecommunications Software Designer with Intracom S.A., Greece. He is currently an Assistant Professor of software engineering with the Department of Applied Informatics, University of Macedonia, Thessaloniki. His research interests are in software metrics, object-oriented design, and low-power hardware/software design.

Dr. Chatzigeorgiou is a member of the IEEE Computer Society.

Spiridon Nikolaidis (M'88) received the Diploma and Ph.D. degrees in electrical engineering from the University of Patras, Patras, Greece, in 1988 and 1994, respectively.

He is currently an Assistant Professor with the Electronics Laboratory, Department of Physics, Aristotle University of Thessaloniki, Thessaloniki, Greece. He is also involved in research projects funded by European and national resources. He is the author or coauthor more than 110 papers in international scientific journals and conference proceedings. His research interests include the design of low-power high-speed reconfigurable processor architectures (ASIP and RISP), CMOS gate propagation delay and power consumption modeling, high-speed and low-power CMOS circuit techniques, and power estimation techniques.

Theodore Laopoulos (SM'89) received the B.Sc. degree in physics and the Ph.D. degree in control electronics from Aristotle University of Thessaloniki, Thessaloniki, Greece, in 1980 and 1989, respectively.

He is an Associate Professor with the Electronics Laboratory, Department of Physics, Aristotle University of Thessaloniki. He has published over 100 papers in international scientific journals and conference proceedings and has supervised one Ph.D. work and many Master-level theses in related subjects. He has served as leader (coordinator) in 11 Greek and European research projects and as a Senior Researcher in certain others. His interests are in the fields of instrumentation circuits and systems, measurement systems and techniques, sensor interfacing, automation, applications of microcontroller systems, and education on electronic instrumentation.

Dr. Laopoulos is an Associate Editor of the IEEE TRANSACTIONS ON INSTRUMENTATION AND MEASUREMENT, an Academic Coordinator of the Socrates/Erasmus program of the Physics Department, and the Chair of the Advisory Board of the "Intelligent Data Acquisition and Advanced Computing Systems" International Workshop.