ELECTRONICS AND ELECTRICAL ENGINEERING

ISSN 1392 – 1215 *2008. No. 8(88)*

ELEKTRONIKA IR ELEKTROTECHNIKA

SYSTEM ENGINEERING, COMPUTER TECHNOLOGY

T 120

SISTEMŲ INŽINERIJA, KOMPIUTERINĖS TECHNOLOGIJOS

Embedded Program Specialization for Multiple Criteria Trade-offs

R. Damaševičius, V. Štuikys

Department of Software Engineering, Kaunas University of Technology,
Studentų str. 50, LT-51368, Kaunas, Lithuania, phone: +370 37 300399, fax: +370 37 300352,
email: robertas.damasevicius@ktu.lt, vytautas.stuikys@ktu.lt

J. Toldinas

Department of Computers, Kaunas University of Technology,
Studentų str. 50-51368, Kaunas, Lithuania, phone: +370 37 300389, email: eugenijus.toldinas@ktu.lt

Introduction

As a result of ever-growing processing energy, larger storage space and faster communication channels, modern embedded systems (ES) are predominantly pervasive and mobile. However, there are heavy constraints on software that limit the functioning of ES, e.g., algorithms that are used in embedded software have to be executed in predefined time with a given accuracy. The time and accuracy constraints always were at the focus for evaluating functionality and efficiency of ES. However, with the advent of mobile computing platforms, other characteristics, such as energy consumption, became an important issue, too.

In most embedded applications, such as cellular phones or portable electronic devices, energy is an important constraint. By decreasing energy consumption, battery life is extended and more capabilities can be included in a device for the same battery capacity. However, there are few tools available to help ES designers to evaluate their designs in terms of energy usage. Currently accurate energy estimation tools are available only for lower levels of design – at the circuit level and to a limited extent at the gate level [1, 2]. For an embedded processor, circuit- or gate-level simulation is slow and impractical to evaluate energy consumption of embedded software.

Although energy consumption has always been a critical concern for mobile computing, other characteristics (execution time, accuracy, memory) remain as important as ever. Current energy analysis methodologies are simply not compatible with the progress in performance evaluation and optimization. They either neglect performance efficiency or isolate energy consumption from other performance metrics such as execution time or memory utilization. It is believed that energy consumption is a major cost when running some large scale applications [3]. This means during an overall cost analysis, energy consumption should be taken into account and should eventually be integrated into performance characterization and evaluation. Thus a systematic approach is needed to evaluate multiple characteristics in embedded software.

The aim of the paper is to introduce a systematic approach for evaluating multiple characteristics and their trade-offs in embedded software. At the core of the approach is a thorough analysis of the problem and solution domains, explicit representation of the domains using *feature diagrams* (FDs) [4] and identification of relationships among various characteristics (features) using analytical and modeling methods. Our contribution is a framework of the methodology for managing embedded software development trade-offs (execution time, energy, accuracy, memory), which use FDs for describing trade-off evaluation models at a high level of abstraction.

Related works

1) *Energy management.* Several techniques have been proposed for energy management in portable and embedded computer systems [5, 6]. Some processor manufacturers have adopted dynamic voltage scaling [7]: the processor clock frequency and supply voltage can be changed on-the-fly to make a trade-off between speed and energy consumption. A method described in [8] relies on application-level observations of *battery dissipation* for a representative set of benchmarks. Another approach for optimizing embedded software energy consumption using instruction reordering and generation of energy-efficient code is proposed in [9]. It is based on an instruction level model that quantifies the energy cost of individual instructions and of various inter-instruction effects.

2) *Profiling-based power optimization* is carried out using profiling tools and is applied at several levels of abstraction: user [10], operational [11], algorithmic, data and instruction-level [12]. The profiler utilizes a cycle-accurate power consumption simulator and relates energy consumption and performance of underlying hardware to the given source code. Using layer abstraction allows developers to focus first on a very abstract view of the problem, and then lower the abstraction level and perform

optimization at a narrower scope. Application-level profiling can be used for dynamically modifying application's behavior to conserve energy [13]. Energy profiling, automated data representation conversion, derivation of polynomial representation and symbolic algebra is combined by [14], where energy profiling is used to identify critical sections of code that need to be optimized. For complex functions, the symbolic algebra techniques decompose the polynomial representation of the basic blocks of a program into a set of embedded processor instructions and automate energy and performance optimization of the arithmetic sections of source code.

3) *Transformation-based optimization.* In [15], power consumption is optimized using *loop unrolling*, where the number of processor cycles is reduced by eliminating loop overheads, and *loop blocking*, where large arrays are broken into several pieces and reused without self interference. Compiler optimizations such as linear loop transformation, tiling, unrolling, fusion, fission and scalar expansion are also considered by [16]. However, only loop unrolling is shown to decrease the consumed energy.

4) *Trade-offs of power vs. other criteria.* For embedded software, other attributes of algorithms, such as execution time, is also important [17]. In most cases minimizing execution time also means minimizing energy consumption. Therefore, additionally to transformation-based optimization, instruction reordering, instruction packing, operand reordering, register allocation, and memory assignment can result in power savings.

A framework of the methodology for assessing multiple criteria

The framework analyzes problem and solution domains at program construction time. The problem domain identifies influential factors, awareness factors, problem type, criteria, their trade-offs and relationships. Analysis of the problem domain consists of the following phases:

- Identification of influential and awareness factors;
- Decomposition of an application into two parts: calculation-intensive and communication–intensive (in this paper we consider only the first part);
- Identification of the criteria: calculation accuracy (A); performance (P); memory (M) and energy (E);
- Identification the following trade-offs: 1) P-A; 2) P-M; 3) P-E; 4) P-A-E;
- Selection of a method (analytic or modeling-based) for trade-off evaluation.

Analysis of the solution domain consists of:

- Identification of calculation-intensive algorithms for embedded applications;
- Selection of representative algorithms for embedded applications;
- Specialization of the representative algorithm(s) for different criteria;
- Description of analytic and modeling-based methods with respect to the identified criteria trade-offs.

What is common for a majority of the calculation-intensive algorithms is that they can be reduced to simpler algorithms such as cosine function. The final reduction leads to the identification of a *representative algorithm*. Calculation of the Taylor series using the Horner's scheme is a representative algorithm because of a variety of functions that can be expressed and calculated using the scheme. The Horner's scheme as a representative algorithm has yet another important property: it can be specialized to a variety of criteria and their trade-offs. We summarize the analysis phase by creating high-level domain models described using feature diagrams (FDs). FDs [18] are tools for representing essential domain features, their dependencies and relationships.

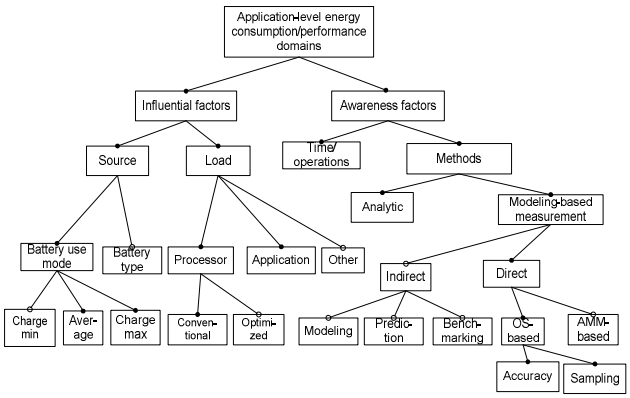


Fig. 1. FD model for representing application-level energy consumption and performance

Fig. 1 explains a general framework that specifies high-level features and possible contexts of the formulated problem. The average energy consumed by a processor while running a certain program is given by $E = I \cdot V_{dd}$, where I is the current and V_{dd} is the supply voltage. The energy consumed by a program, E_p, is given by $E_p = E \cdot T$, where T is the execution time of a program.

Energy consumption values can be obtained physically using the ammeter (AMM). The advantage of the Operating System (OS)-based measurement, which is usually expressed by the battery voltage drop, is that long program execution loops can be introduced to achieve higher accuracy through automatic measurements.

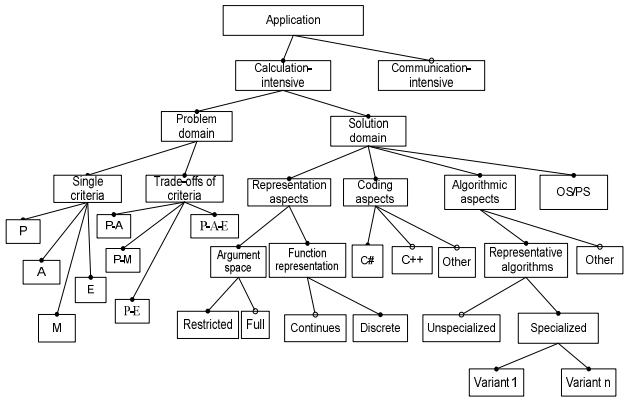


Fig. 2. Application model to evaluate trade-offs of program characteristics

Fig. 2 explains the features of the problem and solution domains. Note that black circles denote obligatory features that were taken into account in our investigation.

White circles denote obligatory features that can be analyzed in other contexts. The models (Fig. 1–2) represent the analyzed domains in the general form and contribute to the explicit description and better understanding of the essential domain features in a whole.

Algorithm and program specialization

Program efficiency can be improved by using 1) more efficient algorithms that solve the same computation problem, or 2) approximate computation algorithms that sacrifice accuracy for gain in other characteristics. More specifically, there are two methods for solving this problem: *data specialization* and *program specialization*.

Data specialization [19] aims at encoding results of early computations in data structures. The execution of a program is divided into two stages. First, a part of the algorithm is pre-computed in advance and the results are saved in a data structure such as look-up table (LUT). A LUT usually is an array (cache), which replaces a runtime computation with a simpler memory access operation. The speed gain can be significant, because retrieving a value from memory is faster than undergoing an expensive computation. Specialization of the algorithm is performed as follows. (1) Analyze the application source code to identify references to the computation costly functions. (2) Generate a LUT for the specialized function using the metaprogramming techniques. (3) Replace all references to the function by the reference to its LUT. A more detailed description of the methodology can be found in [20].

Program specialization [21] aims at improving the efficiency of programs by exploiting known information about the input of a program. An example of such specialization can be computation of the Taylor series specialized for its length.

Case study for cosine function

Trigonometric functions *sine*, *cosine* and *tangent* as well as their inverse functions play important roles in many fields like acoustics, astronomy, DSP, computer graphics, electrical engineering and electronics, mechanical engineering, optics, etc. [22]. For the application of trigonometric functions in real-time ES, typically both the numerical precision and the resource demands are relevant.

The cosine calculation (Eq. 1) has been chosen as a representative algorithm of the calculation-intensive application. According to the *Amdahl's law*, the most effective way to improve performance of a program is to speed-up the most time-consuming part of it. If we speed-up the calculation of cosine values, we can achieve significant gains in program execution times and power usage. Such fine-grained customization is very typical to embedded software development [23]. We analyze three variants of the representative algorithm: Taylor series, cosine LUT and cosine LUT with linear interpolation.

$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \ldots = 1 - \frac{x^2}{2}\left(1 - \frac{x^2}{12}\left(1 - \frac{x^2}{30}(\ldots)\right)\right). \quad (1)$$

Approximation of a function using simpler operations (e.g. addition and multiplication) as in Taylor series of a cosine function enables achieving higher performance and lower power consumption at a cost of accuracy.

Evaluation using the monomial form of an n-degree polynomial requires at most n additions and $(n^2+n)/2$ multiplications, if powers are calculated by repeated multiplication and each monomial is evaluated individually. Using the Horner's scheme representation we need only n additions and n multiplications.

For even better performance, *data specialization* can be applied: known cosine values can be stored in a generated LUT. The trade-off here is that accuracy of the result may depend upon the size of the table. However, in many applications such as JPEG, the results of DCT are rounded-off to the integer values anyway. The complexity of the LUT based method is constant. It requires only 1 multiplication for the calculation of a LUT index and does not depend upon the size of a LUT.

In a simple LUT, the value of a function argument is rounded to the nearest value for which a function value in a LUT exists. Thus the accuracy of this approach is not fine. A more complex approach includes a LUT with linear interpolation of the function values for these arguments of a function, which are not available in the LUT. The complexity of the LUT with linear interpolation is also constant. It requires 2 multiplications and 4 additions, and does not depend upon the size of a LUT.

Measurement method

For estimation of energy, two methods can be used: energy meter-based and OS-based (Fig. 3). The first is better suited for system-level, while the second is more relevant for program level measurements. The OS-based method evaluates voltage drops in battery over program execution time using calls to a specific OS function in different time slots. The display of the mobile device is switched off to avoid unnecessary consumption of power. The measurement result is automatically written to the file before and after calculations that are repeated cyclically in order to ensure the needed accuracy (see Fig. 3).

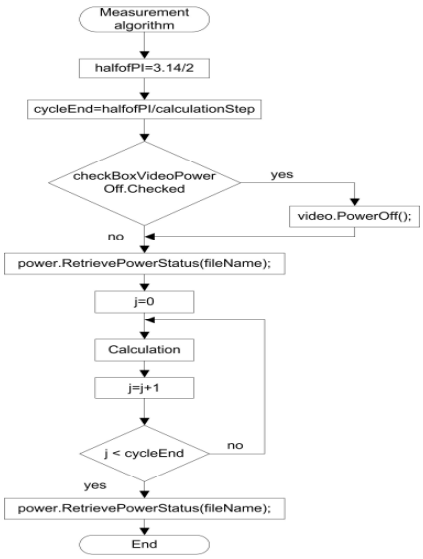


Fig. 3. Energy measurement algorithm at program level

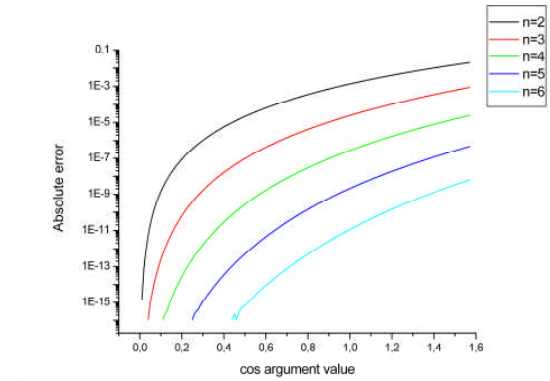
The precision of voltage measurements is 1 mV, and of performance measurements is 1 ms. The calls to cosine functions were put into loops and repeated 10^8 times, while argument values were taken from (0, π/2) range.

Results of experiments

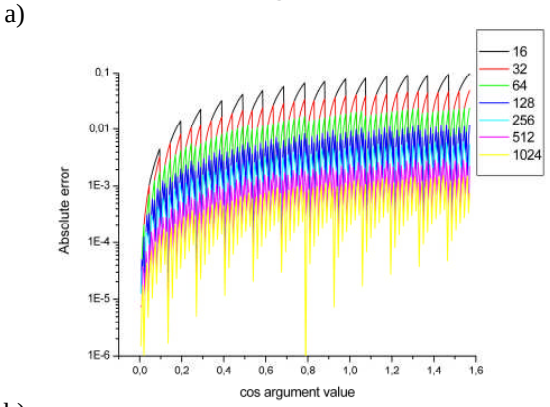
Our investigation corresponds to that part of the general framework, which is described by obligatory features in feature diagrams (see Fig. 1 and 2).

The experiments were performed on a Compaq iPAQ H3900 (Pocket PC platform, Intel PXA250 400 MHz CPU, 32 MB RAM, Windows CE 3.0 OS).

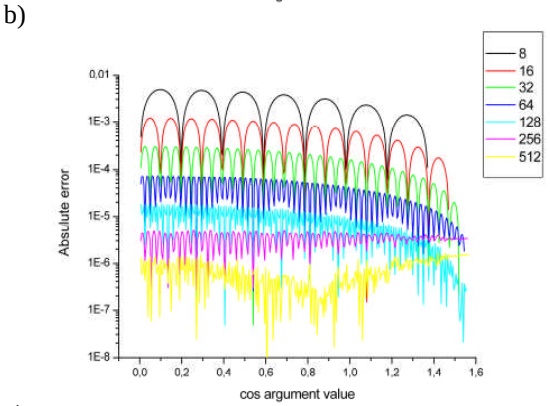
The absolute errors of the *cos* specialization methods are shown in Fig. 4. Taylor series have the best accuracy, while a LUT without interpolation has the worst. Accuracy of the reference function *Math.Cos*() complies to the IEEE 754 standard and is 0.5 ULP (units in the last place), i.e. $0.5 \cdot 10^{-24}$ for single precision floating point arithmetic.



a)



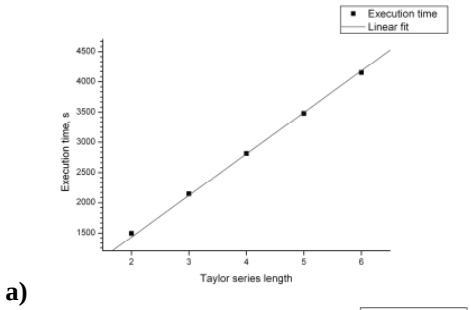
b)



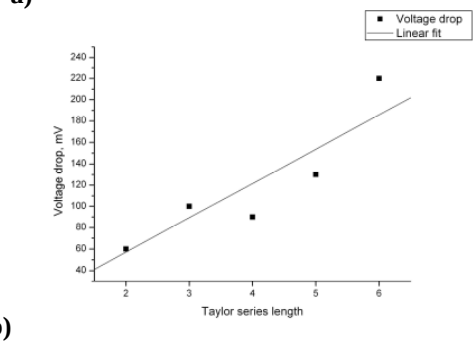
c)

Fig. 4. Accuracy (abs. error) of *cos* computation using a) Taylor series, b) simple LUT and c) LUT with linear interpolation

For the Taylor series, execution time and voltage drop of the *cos* function grows linearly with series length (Fig. 5), whereas for the LUT-based approximation execution time and voltage drop is flat (Fig. 6). The LUT without interpolation has lowest power consumption and best performance. The LUT with linear interpolation has worse power consumption and performance, but higher accuracy. The Taylor series have the worst results both in terms of power and performance (except for n = 2 case, which however, has worst accuracy). The results are presented for the entire experiment (10^8 calls to the measured function).

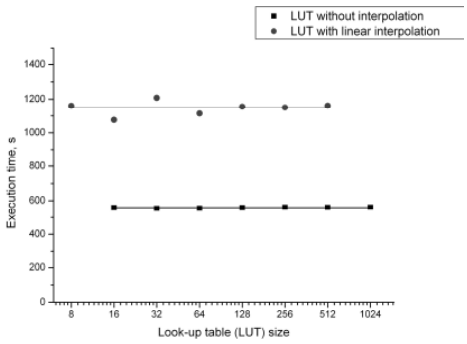


a)

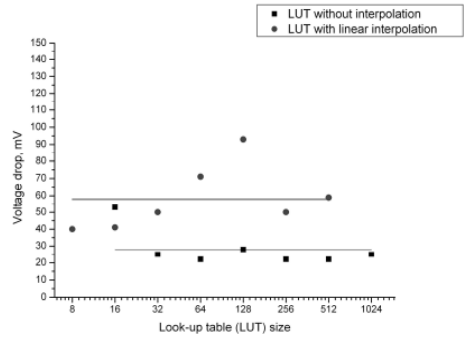


b)

Fig. 5. Execution time (a) and voltage drop (b) of the Taylor series of *cos* function



a)



b)

Fig. 6. Execution time (a) and voltage drop (b) of LUT

The results are summarized in Table 1. The data memory size for LUT methods is $8m$, where m is a size of LUT; 8B is a size of a *double* type used for computations.

Table 1. Experimentally and analytically established complexity of *cos* function specialization methods

Specialization method	Time, ms	Voltage drop, mV	Data memory size, B
Taylor series	$688n - 57$	$32n - 7$	0
LUT without interpolation	560	280	$8m$
LUT with linear interpolation	1150	580	$8m$

The trade-offs between power consumption (expressed via battery voltage drop), execution time and calculation accuracy (expressed via Mean Absolute Percentage Error - MAPE) parameters are shown in Fig. 7.

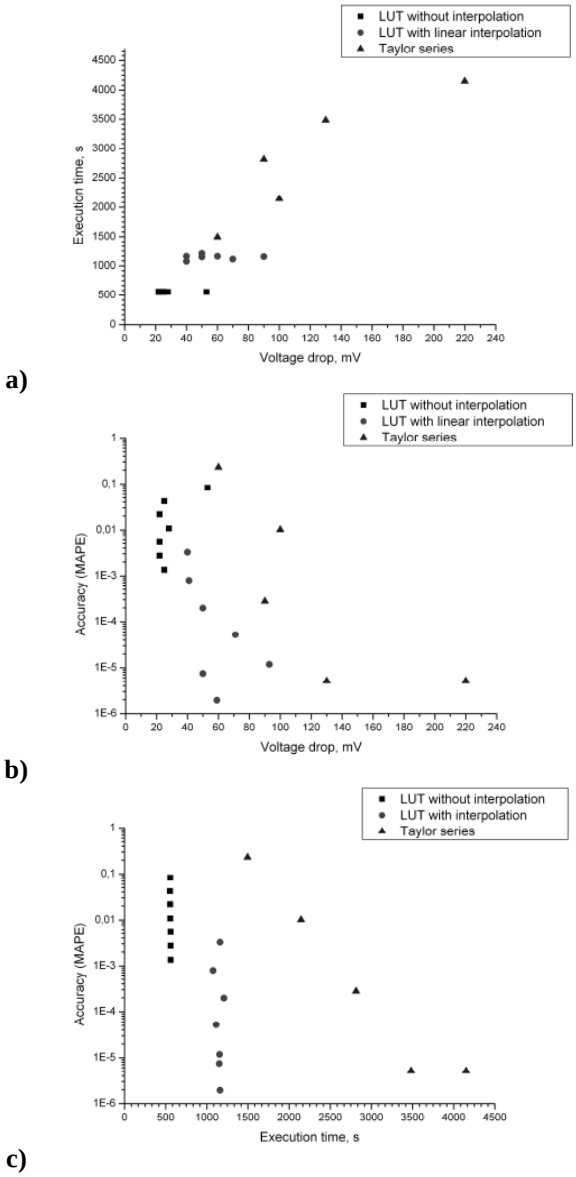


Fig. 7. Trade-offs: a) power/execution time, b) power/accuracy, c) execution time/accuracy

The statistical reliability of complexity estimation using linear fitting is presented in Table 2. The results for voltage drop using simple LUT and execution time using a LUT with linear interpolation could not be fitted satisfactorily due to large spread of measurement results, thus these relationships were established analytically. Such results indicate the difficulties of performing measurements on mobile devices, which were not considered in our experiments. The following factors had significant influence on the results of the experiments: 1) lithium battery discharge behavior deviates significantly from the behavior of an ideal energy source. We have observed that sometimes the battery voltage may increase marginally due to internal chemical effects. Such anomalous behavior may account for large dispersion of voltage drop measurement results. 2) There is a lack of reliable performance profiling tools in Pocket PCs, therefore the OS processes may influence the results of performance measurement.

Table 2. Statistical reliability of results

Measurement	No. of measu-rements	Corre-lation	Std. deviation
Execution time (Taylor series)	5	0.999	47.9
Voltage drop (Taylor series)	5	0.945	26.9
Execution time (LUT)	7	0.775	78.9
Voltage drop (LUT)	7	0.163	18.9
Execution time (LUT + linear interpolation)	7	0.193	41.6
Voltage drop (LUT + linear interpolation)	7	0.691	2.80

Conclusions

We have presented a framework of the methodology for managing embedded software development trade-offs (speed, power, accuracy, memory), which uses Feature Diagrams for describing trade-off models at a high level of abstraction. Such models can include the influential factors, the awareness factors, the type of the problem, single performance, accuracy, memory, and energy criteria, trade-offs of the criteria and relationships. Trade-off models can be used to represent embedded program design trade-offs in energy-speed and energy-accuracy dimensions, and to select a program implementation that best matches system requirements and/or constraints.

References

1. **Jayaseelan R., Mitra T., Li X.** Estimating the Worst-Case Energy Consumption of Embedded Software // IEEE Real Time Technology and Applications Symposium (RTAS 2006). – San Jose, CA, USA. – 2006. – P. 81–90.
2. **Damaševičius R., Štuikys V.** Estimation of Power Consumption at Behavioral Modeling Level Using SystemC // EURASIP J. on Embedded Systems. – 2007. – ID 68673.
3. **Wong P. Y. H.** An Investigation in Energy Consumption Analyses and Application-Level Prediction Techniques / MSc. Theses. – University of Warwick, UK. – 2006.
4. **Kang K., Lee K., Lee J., Kim S.** Feature Oriented Product Line Software Engineering: Principles and Guidelines // In Domain Oriented Systems Development – Practices and Perspectives. – Gordon Breach Science Pub. – 2002.

5. **Lorch J., Smith A.** Software Strategies for Portable Computer Energy Management // IEEE Personal Communications Magazine. – 1998. – Vol. 5(3). – P. 60–73.
6. **Manninger M.** Power management for portable devices // Proc. of 33rd European Solid State Circuits Conf., ESSCIRC. – 2007. – P. 167–173.
7. **Ghazaleh N. A., Childers B., Mosse D., Melhem R., Craven M.** Energy management for real-time embedded applications with compiler support // Proc. of Conf. on Language, Compiler, and Tool for Embedded Systems, LCTES'03. – San Diego, CA, USA. – 2003. – P. 284–293.
8. **Krintz Ch., Wen Y., Wolski R.** Application-level prediction of battery dissipation // Symp. on Low Power Electronics and Design ISLPED'04. – New York, USA. – 2004. – P. 224–229.
9. **Tiwari V., Malik S., Wolfe A.** Compilation techniques for low energy: an overview // Symp. on Low Power Electronics ISLPED'94. – San Diego, USA. – 1994. – P. 38–39.
10. **Ravi N., Scott J., Iftode L.** Context-aware battery management for mobile phones // Conf. on Pervasive Computing and Communications. – Hong Kong, China. – 2008.
11. **Sagahyroon A.** Power Consumption in Handheld Computers // IEEE Asia Pacific Conf. on Circuits and Systems, APCCAS 2006. – Singapore. – 2006. – P. 1721–1724.
12. **Simunic T., de Micheli G., Benini L., Hans M.** Source code optimization and profiling of energy consumption in embedded systems // Proc. of Int. Symp. on System Synthesis (ISSS'00). – Washington, DC, USA. – 2000. – P. 193–198.
13. **Flinn J., Satyanarayanan M.** Energy-aware adaptation for mobile applications // Proc. of the 17th ACM Symp. on Operating Systems Principles (SOSP). – 1999. – P. 48–63.
14. **Peymandoust A., Simunic T., de Micheli G.** Low power embedded software optimization using symbolic algebra // Proc. of the Conf. on Design, Automation and Test in Europe DATE '02. – Washington, DC, USA. – 2002. – P. 1052–1057.
15. **Chung E.-Y., Benini L., De Micheli G.** Source code transformation based on software cost analysis // Proc. of the 14th Int. Symp. on Systems Synthesis ISSS '01. – New York, NY, USA. – 2001. – P. 153–158,
16. **Kandemir M., Vijaykrishnan N., Irwin M., Ye W.** Influence of compiler optimizations on system power // IEEE Trans. on VLSI Systems. – 2001. – Vol. 9(6). – P. 801–804.
17. **Nakutis Z.** Embedded Microcontrollers Benchmarking using Sliding Window Algorithm // Electronics and Electrical Engineering. – 2007. – No. 3 (75). – P. 53–56.
18. **Kang K., Lee K., Lee J., Kim S**. Feature Oriented Product Line Software Engineering: Principles and Guidelines // In Domain Oriented Systems Development – Practices and Perspectives. – Gordon Breach Science Pub. – 2002.
19. **Knoblock T., Ruf E.** Data Specialization // ACM SIGPLAN Notices. – 1998. – Vol. 31(5). – P. 215–225.
20. **Štuikys V., Damaševičius R.** Metaprogramming techniques for designing embedded components for ambient intelligence // Ambient Intelligence: Impact on Embedded System Design. – Kluwer Academic Publishers. – 2003. – P. 229–250.
21. **Jones N. D., Gomard C. K., Sestoft P.** Partial Evaluation and Automatic Program Generation. Prentice Hall Int. – 1993.
22. **Kirner R., Grossing M., Puschner P.** Comparing WCET and resource demands of trigonometric functions implemented as iterative calculations vs. table-lookup // Workshop on Worst-Case Execution Time Analysis. – Dresden, Germany. – 2006.
23. **Beuche D., Spinczyk O., Schroeder-Preikschat W.** Fine-grain application specific customization for embedded software // Design and Analysis of Distributed Embedded Systems. – Montréal, Canada. – 2002. – P. 141–151.

Received 2008 04 29

R. Damaševičius, V. Štuikys, J. Toldinas. Embedded Program Specialization for Multiple Criteria Trade-offs // Electronics and Electrical Engineering. – Kaunas: Technologija, 2008. – No. 8(88). – P. 9–14.

Improvement of characteristics of embedded system programs is important, because of limited resources available for applications in mobile devices. Decreasing power consumption is especially important because of limited battery life and slow growth of battery capacities. Here we analyze a problem of program specialization with multiple criteria (execution time, power, memory, accuracy) in mind. We propose a framework of the methodology for assessing multiple criteria in the problem domain and describe high-level models for evaluating embedded program characteristics using Feature Diagrams. To improve the characteristics of embedded software algorithms, we use function approximation and data specialization (look-up tables). In a case study we analyze the characteristics and trade-offs of the implementations of cosine function. Ill. 7, bibl. 23 (in English; summaries in English, Russian and Lithuanian).

Р. Дамашевичюс, В. Штуйкис, Е. Толдинас. Специализация вложенных программ для множественных критериев // Электроника и электротехника. – Каунас: Технология, 2008. – № 8(88). – С. 9–14.

Улучшение характеристик вложенных программ важно из-за ограниченных ресурсов в мобильных устройствах. Уменьшение расхода энергии вложенного программного обеспечения особенно важно из-за ограниченного срока службы аккумулятора и медленного роста мощностей батареи. Мы анализируем проблему специализации программ с множеством критериев (время выполнения, расход энергии, обьем памяти, точность). Мы предлагаем контуры методологии для оценки множественных критериев в проблемной области и описываем модели высокого уровня для оценки характеристик программ используя диаграммы характеристик. Для улучшения характеристик алгоритмов, мы используем аппроксимацию функций и специализацию данных. Мы анализируем характеристики алгоритмов функции косинуса, которая широко используется в области цифровой обработки сигналов. Ил. 7, библ. 23 (на английском языке; рефераты на английском, русском и литовском яз.).

R. Damaševičius, V. Štuikys, J. Toldinas. Daugiakriteris įterptinių programų specializavimas // Elektronika ir elektrotechnika. – Kaunas: Technologija, 2008. – Nr. 8(88). – P. 9–14.

Įterptinių sistemų programų charakteristikas svarbu gerinti dėl ribotų mobiliųjų įrenginių išteklių. Energijos suvartojimą mažinti yra ypač svarbu, nes akumuliatorių eksploatavimo trukmė yra ribota, o jų talpa didėja lėtai. Analizuojama daugiakriterio (pagal vykdymo trukmę, energijos suvartojimą, naudojamos duomenų atminties dydį, skaičiavimų tikslumą) programų specializavimo problema. Siūlomi daugiakriterio įvertinimo probleminėje srityje metodikos kontūrai ir aprašomi aukšto lygmens modeliai, skirti įterptinių programų charakteristikoms nustatyti sprendimo srityje naudojant bruožų (požymių) diagramas. Algoritmų charakteristikoms gerinti naudojami funkcijų aproksimavimo ir duomenų specializavimo (lentelės) metodai. Tiriamas kosinuso funkcijos skaičiavimo algoritmų charakteristikų įvertinimas. Il. 7, bibl. 23 (anglų kalba; santraukos anglų, rusų ir lietuvių k.).