Predictive Models for Multimedia Applications Power Consumption based on Use-Case and OS Level Analysis

Patrick Bellasi*, William Fornaciari*, and David Siorpaes†
*Dipartimento di Elettronica e Informazione
Politecnico di Milano, P.zza Leonardo da Vinci, 32. 20133 - Milano, Italy
Email: {bellasi,fornacia}@elet.polimi.it
†Advanced System Technology
STMicroelectronics, Via C. Olivetti, 2. 20041 - Agrate Brianza, Italy
Email: david.siorpaes@st.com

***Abstract*—Power management at any abstraction level is a key issue for many mobile multimedia and embedded applications. In this paper a design workflow to generate system-level power models will be presented, tailored to support quantitative run-time power optimization policies to be implemented within an operating system. The approach we followed to derive power models is strongly use-case oriented. Starting from a comprehensive general and accurate model of a representative architecture for embedded applications (including a multi core MPSoC, accelerators, interfaces and peripherals), a methodology to derive compact models is presented, based upon the distinctive characteristics of the selected use cases. The methodology to generate such model, whose exploitation is foreseen within a power manager working at the OS level, is the focus of the paper. The value and accuracy of the approach is quantitatively and statistically justified through extensive experiments carried out on a development board designed for multimedia applications.**

I. Introduction

Mobile devices are nowadays complex and full featured systems, with multiple functionalities embedded into few components, usually realized using SoC based embedded platforms. The shortening of the overall design time of such systems is enforcing more comprehensive design approaches where system-wide power control is exposed also at the Operating System (OS) level. Moreover, to better fit different and frequently changing system usage profiles, both dynamic and adaptive power management policies are required.

The problem of building policies and models to support effective implementation of static and dynamic power managers has been addressed since at least a decade [1], comparisons can be found in [2][3][4]. Such a general goal has been considered in multiple ways, focusing on some architecture components, as well as at higher levels of abstraction.

Many techniques focus on the microprocessor power, tuning the computational power to the application demand, both in the domain of real time applications [5][6], where off-line profiling data are used to optimize scheduling algorithms, and in less time constrained applications exploiting on-line collected measurements [7][8][9]. Other approaches take into account the modeling and optimization of peripheral devices such as memories [10][11], consider some effects related to the technology scaling trends, such as leakage power [12][13], or address specific application contexts [14] and architectures [15]. Only recently appeared some concrete proposals paying particular attention to system-wide power management at the OS level [9][16].

One of the main needs of any policy concerning power management is the availability of some model for the power consumption of the application and possibly of the operating system. Using a model is generally time consuming and a balance between accuracy and overhead must be achieved.

One of the first approaches towards including the OS within the analysis loop, is SimOS, which extended the idea of ISS with those services of the peripherals necessary to simulate the behavior of an entire operating system. A step ahead has been done by SoftWatt [17], EMSIM [18] and SimBed [19] which introduced informations for the simulation related both to the instructions and the connected peripherals in order to enable full-system simulations. However, the problem of long run times when using full-featured configurations is still present, as well as the loss of per-cycle information.

Modeling of timing/power at the operating system level has been carried out in [20] where, starting from a call-tree of some applications, a proper clustering of power and execution time is derived to provide guidelines for application optimization. Other operating system characterizations have been presented in [21], where data collected from simulation are grouped to extract statistical models useful for performance prediction; in [19] this type of data is used at run-time to manage and optimize the power consumption.

Other valuable proposals affording the characterization of the operating systems by using complex dedicated hardware or measurement boards, or very time consuming procedures are presented in [22][23][24], while a complete characterization working at OS system call granularity is presented in [25].

Even if the focus of our research is significantly overlapped with the above literature, our final goal is to work on the development of a lightweight framework to be used for driving

978-3-9810801-5-5/DATE09 © 2009 EDAA

mainly run-time optimization policies. Moved by the aim to better support system-wide power management, we propose a methodology to build use-case based energy estimation models, using OS and device state informations, while still being completely application independent.

The methodology described in the following is conceived to provide a predictive model for the energy consumption depending on the different system configurations, based on observations carried out at OS level and with low run-time overhead. We decided to focus on a quantitative analysis of the use-cases: (i) to simplify the creation of models, by reducing the design space to be explored, and (ii) to increase the accuracy/flexibility of the predictions for a given application context.

The aim of our work is to suggest a new methodology to build system-wide energy consumption models, that are suitable to be used at run-time to support power management decisions, taking into considerations the application context. Thanks to the well defined system configuration, already available at the platform design time, it is possible off-line models generation and optimization. The use-case approach, will allow to be more adaptive with respect to system usage conditions changes and better balance between the control overhead and its accuracy.

This work is the foundation for an on-going study aimed at developing a dynamic and adaptive system-wide power management system. Due to lack of space, this paper will present only the main stages to create a predictive system-wide energy model, disregarding its actual usage to implement run-time power management policies. More details on these aspects can be found in [26].

The rest of this paper is organized as follow: Section II describes the use-case oriented approach to the building of power models. In section III we report some experimental results, verifying our models fitting with power data from real-world applications. Concluding remarks and a brief outline of the work in progress are drawn in Section IV.

II. Energy estimation based on use-cases

One of the aims of our system energy estimation models is to efficiently support the power control decision process. Such models allow on-line estimation, with a sufficient accuracy, of power consumption of a target platform when it is running in a given system state. To better exploit available optimizations, our estimation models refer to use-cases instead of applications, considering the system state as a snapshot of both hardware and software configurations. Let us focus on how such models are built. A view of the whole process is reported in Fig 1. Three main steps can be identified: (A) the training set generation, (B) a profiling phase and, finally, (C) the data processing and model creation.

A. Training set generation

A *system observer module* defines the set of parameters that can be identified in the target platform. Disregarding the nature

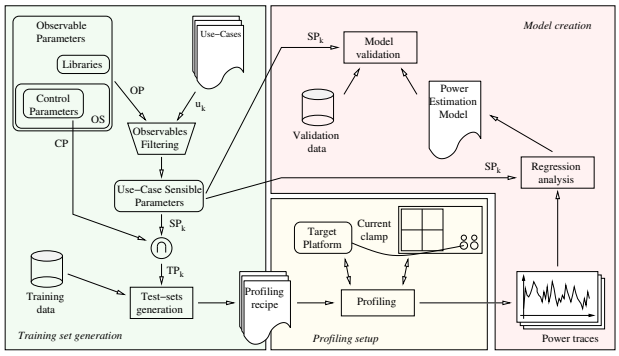


Fig. 1. Workflow for the generation of power estimation models

TABLE I
SOME PARAMETER PROPERTIES

Property	Description
name	Property name
min_val	Minimum allowed value
max_val	Maximum allowed value
min_step	Minimum change step value
controllable	TRUE if its value is configurable

of these parameters, we simply define the vector of *observable (system) parameters*:

$$OP = \{p_1, p_2, \ldots, p_n\}$$

Each entry p_i has an associated structure specifying some of its properties, as shown in Tab I. Some of the OP entries represent control points of the target platform (e.g. voltages and frequencies): the power manager can change the value of these parameters to tune system behaviors. We define the vector of *control parameters* as:

$$CP = \{c_1, c_2, \ldots, c_m\} \subseteq OP$$

The definition of control parameters, along with their properties, should be provided by platform developers during the implementation of the architecture specific part of the system observer module.

The set of *use-cases* is similarly represented by the vector:

$$UC = \{u_1, u_2, ..., u_l\}$$

Each use-case comes with a description of the application context in terms of the system parameters that could be somehow affected by running the use-case. We define the *use-case sensible parameters* as:

$$SP_k = \{p_{jk} \,|p_{jk} \in OP \wedge \mathit{affect}(u_k, p_{jk})\}$$

this represents the collection of the system parameters influenced by the use-case u_k. Of course, $SP_k \subseteq OP$ so that focusing on a specific use-case, the OP set can be filtered to shrink the analysis only on few parameters.

Moving towards the generation of models, we define the *training-set generation* parameters as:

$$TP_k = SP_k \cap CP$$

This is the set of control parameters that we will consider to generate all the possible training traces to build our model.

The training set is a collection of system configurations that are meaningful for the considered use-case. From another

perspective, the training set is the complete enumeration of all the possible states the system can assume while executing the considered use-case. Such exploration is typically time consuming, in terms of both trace generation and model identification; however it is performed off-line and it has to be done only once.

The complete set of configurations to be considered for a specific use-case is given by:

$$TS_k = TP_k \times TP_k$$

The cardinality of this set corresponds to the number of experiments we have to perform in the following profiling step. Such value is defined by:

$$Experiments_{number} \leq \prod_{p \in TP_k} |p|$$

Actually this is an upper bound, since there exist dependencies among the parameters: e.g. voltage and frequency cannot be modified independently. Parameters constraints shrink the search space significantly, avoiding the explosion of its dimensions, as we have observed in the experimental phase, while working on some real use-cases.

B. Configuration profiling

In this step a power consumption trace is generated for each system configuration. Power profiling can also be done by simulation, using a suitable architectural simulator with a sufficiently accurate power consumption model. This approach can be useful to support the development of a power manager, starting from the early design stages of a new platform. Alternatively, as we did in this paper, if the target platform is already available as tangible hardware, it is necessary to organize a profiling environment, e.g. using a digital multimeter (DMM).

Current consumption can be measured using a current clamp on board's power lines. In this case a suitable software toolset has to configure the target platform for a test, synchronize the test execution with DMM sampling and, finally, collect and store the power traces on a host PC. The availability of a software tool allows to simplify the profiling process: it can automate the translation of TS_k's training-set generation parameters into a predefined-syntax "recipe file", which once parsed will describe the tests to execute and how to collect measurements data.

C. Data processing and model generation

The last step of our workflow deals with processing power traces in order to build a sufficiently accurate power estimation model of the considered use-case. The proposed solution for model identification is based on regression analysis [27]. In the implementation of our workflow, regression analysis is supported by "R"[1], which has been selected thanks to its functionalities, which are explicitly oriented to statistical data analysis and model fitting.

Regression is one of the most widely used statistical analysis technique for fitting a quantitative response variable y with a function of one or more predictor variables $x_{1,}x_{2,}\ldots,x_n$. This kind of data analysis is widely adopted in experimental contexts with poor knowledge about the underlying system, and thus it is acceptable to use empirical models to describe reality. Moreover, under some assumptions and constraints that we verified our approach complies with, regression is a powerful tool which can also be used to predict behaviors. This technique fits very well our needs: we have a set of experimental data (power traces) and we have to identify a predictive model related to the system configuration used to produce such data. The values of SP_k are our independent variables (i.e. *explanatory variables*) while the mean value of a power trace is the dependent variable (i.e. *response variable*) we have to predict. The resulting regression model is a function of the independent variables and one or more parameters. These parameters must be tuned in order to achieve the best data fitting.

Note that the input data of such models do not represent a complete coverage of the application context. Only some of the independent variables, namely those belonging to the TP_k set, are changed during profiling. Hence, the model should support *extrapolation*: this is considered to be more risky when regression is used to build the model and therefore we paid particular attention to the validation phase. This is the main motivation for the last step in our workflow (Fig 1): a model validation stage is required using input data different from those considered for the model identification.

The estimation capability of the new predictive model is tested indirectly, by forcing each of the non-controllable parameters in $NP_k = SP_k \setminus TP_k$. For each validation test, if the prediction error can be verified to be within required acceptable margins, the model is assumed to be validated with respect to this non-controllable parameter. Otherwise the considered parameter has to be added to the TP_k set and the profiling stage restarted.

As it will be explained in the next section, each model has to be incrementally refined to balance the number of parameters and its accuracy, defined by residual errors. The statistical significance of parameters will be the guideline to shrink the model. This task can be carried out almost automatically with just some manual fine-tuning to produce the final model.

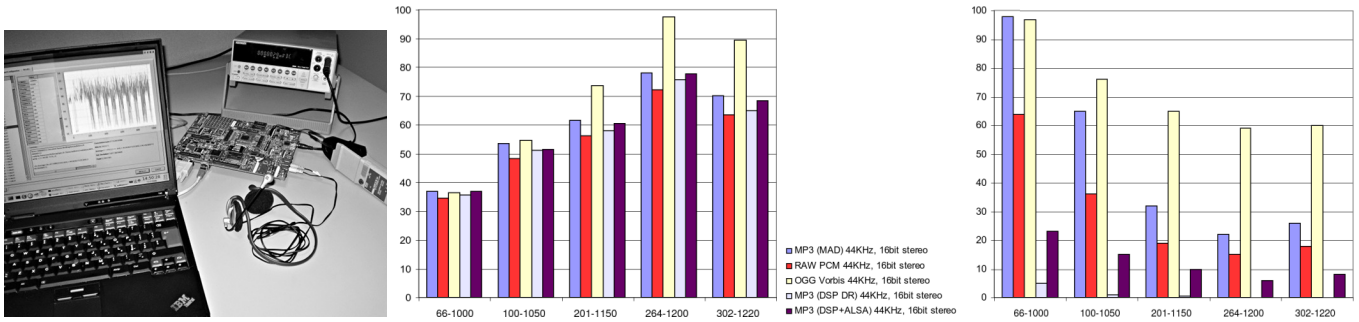
III. Experimental results

The proposed workflow has been validated considering a real-world use-case and computing architecture. We considered audio playback applications running on a Nomadik board equipped with the STn8815 MPSoC[2], which integrates an ARM core and a DSP. This architecture exposes a number of fine grained power-performance controls on different devices, including dynamic frequency and voltage scaling (DVFS).

An implementation of the system observer module has been developed as an extension of the DPM [28] subsystem running on a 2.6.20 Linux kernel. This first implementation exports in userspace all the observable parameters reported in Tab II. The

[1]A language and environment for statistical computing and graphics. http://www.r-project.org

[2]Nomadik: mobile multimedia application processor. STMicroelectronics. http://www.st.com



(a) (b) (c)

Fig. 2. (a) The profiling setup. Pymeasure running on a laptop and collecting data using a current clamp. (b) Consumption data on different system configurations and different playback setups. (c) CPU load data on different system configurations and playback setups. (All data have been normalized due to existing NDA.)

TABLE II
SYSTEM PARAMETERS: OBSERVABLE (OP) AND CONTROLLABLE (CP)

Name	Description	Type	Range
Ch	Number of audio channels	OP	1 or 2
Enc_f	Encoding audio frequency	OP	8000, 22050 and 44100 [Hz]
Bps	Bit per sample	OP	8 or 16
Cpx	Decoding complexity	OP	PCM, MP3, WMA and OGGVorbis
$Proc_{load}$	CPU load	OP	[0;100]
Mem_{load}	Used memory	OP	[0;100]
F_{core}	CPU clock frequency	CP	66, 100, 201, 264 and 302 [MHz]
V_{core}	SoC voltage	CP	1000, 1050, 1150, 1200 and 1220 [mV]
DSP_{load}	DSP load	OP	[0;100]
$HCLK$	System bus clock frequency	CP	66, 100, 133 [MHz]

first three parameters (Ch, Enc_f, Bps) are retrieved by the audio-codec driver configuration. The decoding complexity is a qualitative classification of the computational effort related to the adopted decompression algorithm. $Proc_{num}$, $Proc_{load}$ and Mem_{load} are parameters retrieved directly from some kernel's internal data structures. F_{core} and V_{core} are derived from the control points already exported by DPM to configure the main clock, along with $HCLK$ which represents the clock of the memory and DSPs subsystems. Finally DSP_{load} is read by the on-kernel audio accelerator firmware interface.

It is worth to notice that only three parameters are controllable, while the others are only observable. Anyway, in this particular use-case, it is quite easy to drive the first three observable parameters simply by changing the audio file to decode. Hence, for the profiling of the target use-case we considered:

$$TP_k = \{Ch, Enc_f, Cpx, F_{core}, V_{core}, HCLK\}$$

The range of each parameter is reported in Tab II. In order to compute correctly the size of the training set, it should be observed that some of these parameters cannot be modified independently from the others: e.g. the V_{Core} has a minimum allowed level for each F_{Core} value, as typical in systems supporting DVFS. To evaluate advantages on configuring these two parameters independently, we considered only the combinations of them ensuring system stability, which turned out to be 15. Considering some other architectural constraints, the final training set counts just 134 configurations to be profiled.

For each configuration, a power trace has been generated by collecting 1000 current measures at 50Hz sampling frequency. Each trace corresponds to 20s of audio playback. To skip the perturbations associated with the switching among the configurations, measurements have been taken in a steady condition, by selecting them randomly in the interval of [5..10]s after the starting of the playback. Samples have been collected using a Keithley's Digital Multimeter and a Fluke's current clamp sensing the board's power line. Measures are downloaded from the DMM trough a custom-made application that, for each power trace to be generated, takes care to configure both DMM and the target board, synchronize their behavior, collect the measurement and store the data for the further processing (Fig 2a). The platform average power consumption, for each tested configuration, is immediately available at the end of these highly automated profiling procedures.

An initial analysis of the data collected during profiling has been performed using both numerical (i.e. mean, standard deviation and correlation values) and graphical summaries (e.g. one variable boxplots and two variable scatterplots). We investigated for outliers, data-collection errors and skewed or unusual distributions. This first analysis confirmed that data have been correctly collected and reasonably distributed.

We generated a first model, Model 1 in Tab III, by performing linear regression on these datasets. The obtained model describes quite well data, with a low 2.5 residual standard error, but it still has an excessive number of regressors.

To move towards a simplified version, from the first model we selectively removed regressors with a low statistical significance, using *stepwise regression*, still taking care to keep the residual standard error lower than 10. We obtained Model 2 counting only 10 regressors and still having good fitting properties.

Another improvement in the model has been obtained by exploiting a correlation analysis between the response variables

TABLE III
STATISTICAL PROPERTIES OF THE MODEL

	Model 1	Model 2	Model 3
Residual standard error	2.509	6.466	3.14
Multiple R-Squared	0.9963	0.9514	0.9883
Adjusted R-squared	0.9921	0.9475	0.9876
F-statistic	237.7	242.7	1336
p-value	¡2.2e-16	¡2.2e-16	¡2.2e-16
Regressors count	69	10	8
Average error [%]	6.4	5.15	2.47

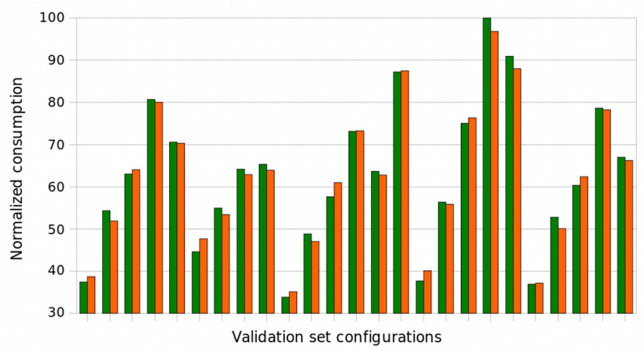


Fig. 3. Comparison between the model forecast (orange) and the actual measurements (green) for a significant validation set

(i.e. mean current drained) and the product of some pairs of input variables. Some of these products, such as $F_{core}V_{core}$ and $HCLKF^2_{core}$, resulted highly correlated and thus have been explicitly introduced into the model before running the stepwise regression. At the end of such optimization stages, Model 3 has been identified, which includes only 8 regressors and still shows good statistical properties.

An extensive model validation has been performed, not only to verify the prediction capability of the Model 3, but also to verify the regression's underlying hypothesis. Results of such analysis[3] is graphically represented by the plots in Fig 4, where we can analyze:

1) *residuals independence:* residual errors are confirmed to be random since the graph does not show clusters.
2) *error linearity:* the graph is balanced around zero, namely the model does not suffer of over or under estimation bias.
3) *error normality:* the residuals distribution is normal since the points on the graph tend to cover the line.
4) *homoschedasticity:* residuals have almost a constant variance.

After model statistical validation, the prediction capabilities have been tested by using different set of traces from those used for profiling and model identification. Such validation set covered a range of different system configurations and playback setups. Differences between the actually measured average power consumption and the model forecasts are shown on Fig 3. As reported in Tab III, the average prediction error of the model is only 2.47%, that is comparable with the accuracy of the adopted measurement equipment.

Due to an NDA with *STMicroelectronics*, only the general structure of Model 3 can be reported:

$$I_{est} = k_1 - k_2\,Cpx + k_3\,Bps + k_4\,DSP_{load} + k_5\,Proc_{load}F_{core} + +k_6\,HCLKV_{core} + k_7F_{core}V_{core} + k_8\,CpxV_{core} - k_9\,HCLKF^2_{core}$$

IV. CONCLUSIONS AND FUTURE WORK

We presented a methodology to produce system-wide energy estimation models suitable to support a power manager run-time policy. Our approach leverages on system knowledge for off-line use-case based platform profiling.

The main benefit of the use-case approach lies in the possibility of identifying fine-grained run-time models, optimized for specific application contexts. Moreover they simplify the off-line profiling stage and improve the power manager adaptability to application contexts changes. Stepwise regression has been used to build and optimize linear models that predict the average power consumption, with a proved accordance with measured data, given the informations corresponding to the system configuration. These informations come from a software implemented observation module, laying on top of an efficient and platform independent OS instrumentation code.

Currently we are working on the run-time composition of energy models, belonging to different use-cases, in order to better support the power manager in more generic utilization scenarios. Others efforts are on: performance estimation models by following an approach similar to that used for the power consumption, and mechanisms to feed these informations to the power manager in order to better support its control decisions.

ACKNOWLEDGMENTS

The authors would like to thank Stefano Galli for the valuable support on developing the measurement tool and contribution on the definition of the described model.

REFERENCES

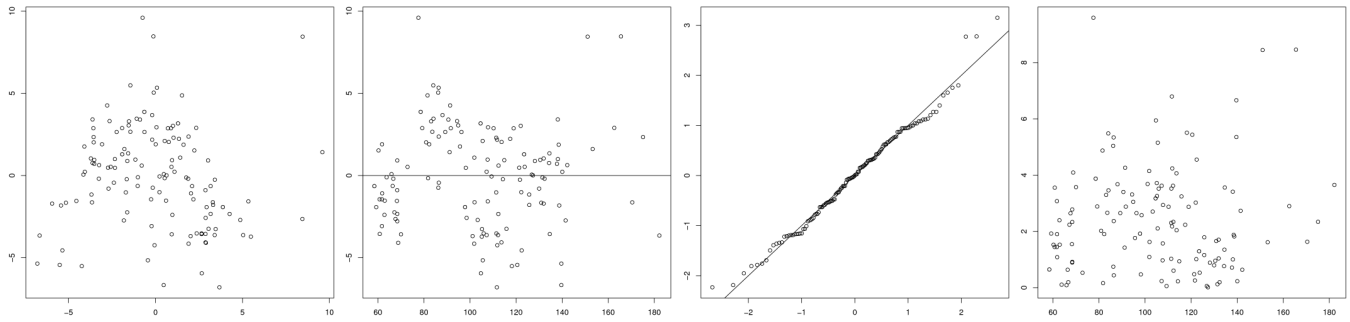
[1] L. Benini, A. Bogliolo, and G. D. Micheli, "A survey of design techniques for system-level dynamic power management," *IEEE Transactions on VLSI Systems*, vol. 8, no. 3, pp. 299–316, Jun 2000.

[2] K. Govil, E. Chan, and H. Wasserman, "Comparing algorithm for dynamic speed-setting of a low-power cpu," in *MobiCom '95: Proceedings of the 1st annual international conference on Mobile computing and networking*. New York, NY, USA: ACM, 1995, pp. 13–25. [Online]. Available: http://doi.acm.org/10.1145/215530.215546

[3] T. Pering, T. Burd, and R. Brodersen, "The simulation and evaluation of dynamic voltage scaling algorithms," in *ISLPED '98: Proceedings of the 1998 international symposium on Low power electronics and design*. New York, NY, USA: ACM, 1998, pp. 76–81. [Online]. Available: http://doi.acm.org/10.1145/280756.280790

[4] Y. Lu, E. Chung, T. Simunic, L. Benini, and G. Micheli, "Quantitative comparison of power management algorithms," 2000. [Online]. Available: citeseer.ist.psu.edu/lu00quantitative.html

[3]Refer to [27] for a detailed description



(a) (b) (c) (d)

Fig. 4. Regression hypothesis verification for Model 3. (a) residuals independence; (b) error linearity; (c) error normality and (d) homoschedasticity.

[5] P. Pillai and K. G. Shin, "Real-time dynamic voltage scaling for low-power embedded operating systems," in *SOSP '01: Proceedings of the eighteenth ACM symposium on Operating systems principles*. New York, NY, USA: ACM, 2001, pp. 89–102. [Online]. Available: http://doi.acm.org/10.1145/502034.502044

[6] A. Manzak and C. Chakrabarti, "Variable voltage task scheduling algorithms for minimizing energy/power," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 11, no. 2, pp. 270–276, 2003. [Online]. Available: http://dx.doi.org/10.1109/TVLSI.2003.810801

[7] W. Yuan and K. Nahrstedt, "Energy-efficient soft real-time cpu scheduling for mobile multimedia systems," in *SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles*. New York, NY, USA: ACM, 2003, pp. 149–163. [Online]. Available: http://doi.acm.org/10.1145/945445.945460

[8] J. Lorch and A. Smith, "Pace: A new approach to dynamic voltage scaling," *IEEE Trans. Comput.*, vol. 53, no. 7, pp. 856–869, 2004. [Online]. Available: http://dx.doi.org/10.1109/TC.2004.35

[9] Y. Cho, N. Chang, C. Chakrabarti, and S. Vrudhula, "High-level power management of embedded systems with application-specific energy cost functions," in *DAC '06: Proceedings of the 43rd Annual Conference on Design Automation*. New York, NY, USA: ACM Press, 2006, pp. 568–573. [Online]. Available: http://doi.acm.org/10.1145/1146909.1147057

[10] Y. Cho and N. Chang, "Memory-aware energy-optimal frequency assignment for dynamic supply voltage scaling," in *ISLPED '04: Proceedings of the 2004 international symposium on Low power electronics and design*. New York, NY, USA: ACM, 2004, pp. 387–392. [Online]. Available: http://doi.acm.org/10.1145/1013235.1013327

[11] Y. Choi, N. Chang, and T. Kim, "Dc-dc converter-aware power management for battery-operated embedded systems," in *DAC '05: Proceedings of the 42nd annual conference on Design automation*. New York, NY, USA: ACM, 2005, pp. 895–900. [Online]. Available: http://doi.acm.org/10.1145/1065579.1065814

[12] R. Jejurikar, C. Pereira, and R. Gupta, "Leakage aware dynamic voltage scaling for real-time embedded systems," in *DAC '04: Proceedings of the 41st annual conference on Design automation*. New York, NY, USA: ACM, 2004, pp. 275–280. [Online]. Available: http://doi.acm.org/10.1145/996566.996650

[13] L. Niu and G. Quan, "Reducing both dynamic and leakage energy consumption for hard real-time systems," in *CASES '04: Proceedings of the 2004 international conference on Compilers, architecture, and synthesis for embedded systems*. New York, NY, USA: ACM, 2004, pp. 140–148. [Online]. Available: http://doi.acm.org/10.1145/1023833.1023854

[14] R. Jejurikar and R. Gupta, "Dynamic voltage scaling for systemwide energy minimization in real-time embedded systems," in *ISLPED '04: Proceedings of the 2004 international symposium on Low power electronics and design*. New York, NY, USA: ACM, 2004, pp. 78–81. [Online]. Available: http://doi.acm.org/10.1145/1013235.1013261

[15] T. Pering, T. Burd, and R. Brodersen, "Voltage scheduling in the iparm microprocessor system," in *ISLPED '00: Proceedings of the 2000 international symposium on Low power electronics and design*. New York, NY, USA: ACM, 2000, pp. 96–101. [Online]. Available: http://doi.acm.org/10.1145/344166.344530

[16] B. Brock and K. Rajamani, "Dynamic power management for embedded systems," *Proceedings on IEEE International SoC Conference*, pp. 416–419, Sept. 2003.

[17] S. Gurumurthi, A. Sivasubramaniam, M. Irwin, N. Vijaykrishnan, and M. Kandemir, "Using complete machine simulation for software power estimation: the softwatt approach," Feb. 2002, pp. 141–150.

[18] T. Tan, A. Raghunathan, and N. Jha, "Emsim: an energy simulation framework for an embedded operating system," *IEEE International Symposium on Circuits and Systems*, vol. 2, pp. II–464–II–467 vol.2, 2002.

[19] K. Baynes, C. Collins, E. Fiterman, B. Ganesh, P. Kohout, C. Smit, T. Zhang, and B. Jacob, "The performance and energy consumption of embedded real-time operating systems," *IEEE Transactions on Computers*, vol. 52, no. 11, pp. 1454–1469, Nov. 2003.

[20] R. Dick, G. Lakshminarayana, A. Raghunathan, and N. Jha, "Analysis of power dissipation in embedded systems using real-time operating systems," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 22, no. 5, pp. 615–627, May 2003.

[21] T. Tan, A. Raghunathan, and N. Jha, "Embedded operating system energy analysis and macro-modeling," 2002, pp. 515–522.

[22] T. Li and L. K. John, "Run-time modeling and estimation of operating system power consumption," in *SIGMETRICS '03: Proceedings of the 2003 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*. New York, NY, USA: ACM Press, 2003, pp. 160–171. [Online]. Available: http://doi.acm.org/10.1145/781027.781048

[23] A. Acquaviva, L. Benini, and B. Riccó, "Energy characterization of embedded real-time operating systems," *SIGARCH Comput. Archit. News*, vol. 29, no. 5, pp. 13–18, 2001. [Online]. Available: http://portal.acm.org/citation.cfm?id=563647.563652

[24] R. Oliver, P. Teller, and W. McGregor, "Accurate measurement of system call service times for trace-driven simulation of memory hierarchy designs," Feb 1998, pp. 239–244.

[25] C. Brandolese and W. Fornaciari, "Measurement, analysis and modeling of rtos system calls timing," Sept. 2008, pp. 618–625.

[26] P. Bellasi, "Dynamicpower for stlinux," STMicroelectronics, Tech. Rep. 76, 2007.

[27] R. Mason, R. Gunst, and J. Hess, *Statistical Design and Analysis of Experiments, 2nd Edition*, Wiley, Ed., 2003.

[28] IBM and M. Software, "Dynamic power management for embedded systems," Tech. Rep., 11 2002.