Energy Estimation for 32-bit Microprocessors

C.Brandolese, W.Fornaciari, F.Salice, D.Sciuto
Politecnico di Milano, DEI
Piazza L. Da Vinci, 32
20133 Milano, Italy
brandole, fornacia, salice, sciuto@elet.polimi.it

ABSTRACT

Estimation of software power consumption is becoming one of the major problems for many embedded applications. The paper presents a novel approach to compute the energy of an Instruction Set, through a suitable *functional decomposition* of the activities involved during instruction execution. One of the main advantages of this approach is the capability to predict the power figures of the overall Instruction-Set starting from a small subset. A formal discussion on the statistical properties of the model is included, together with its application on five commercial 32-bit microprocessors.

1. INTRODUCTION

The presence of software in embedded applications is becoming more and more pervasive. The intrinsic complexity of CPU cores and the lack of low-level details from microprocessor suppliers, makes it hard to predict the overall power consumption from a system-level design perspective. Therefore, there has been a relevant research effort in the past years, to identify possible analysis methodologies capable of evaluating the software power by avoiding the use of EDA tools developed for the hardware to analyze the architectural blocks composing the microprocessor. In fact, this process is time-consuming and produces data difficult to be properly interpreted without further analysis of the module correlations [1; 2; 3]. Recently promising approaches have been proposed, working at the instruction-level, based on the characterization of the Instruction Set via a set of physical measurements when the microprocessor is exercised with long runs of the same instructions [4; 5; 6]. Nevertheless, despite these strategies generally produce good results especially in term of *relative* precision, the validity of the models and the significance of the experimental data are typically not justified under a statistical viewpoint and the need of performing an exhaustive set of measurements represents a major obstacle in extending the analysis towards new microprocessors.

The proposed approach here reported focuses on 32-bit general-purpose microprocessors and is still based on an instruction-level characterization of the software. However, our solution partially overcomes the above limitations since it is independent of the specific microprocessor architecture, the overhead of the physical measurements is dramatically reduced and the significance of each analysis step has been statistically evaluated. These properties have been achieved by analyzing and decomposing the instruction execution in terms of *functionalities*, rather than focusing on the physical blocks composing the microprocessor architecture.

The paper is organized as follows. Section 2 introduces the general theoretical model, namely the strategy to functionally decompose the instruction execution and its use to estimate the static energy associated with the instructions. Section 3 is devoted to the practical application of the methodology, by considering a compared analysis of five commercial microprocessors. Finally, some conclusions are drawn in section 4.

2. MODEL IDENTIFICATION

2.1 Model definition

The problem of the identification of a functional model for the energy consumption at the instruction level is investigated, considering the relation that exists between the processor architecture and a set of *functionalities*.

DEFINITION 1. *A **functionality** is a set of activities aimed at a specific goal and involves, partially or totally, one or more units that can be identified in the structure of a generic microprocessor.*

DEFINITION 2. *Two functionalities F_1 and F_2 are **space-disjoint** if the activities accomplished by F_1 involve different structural units than those that F_2 requires.*

DEFINITION 3. *Two* functionalities *F_1 and F_2 are **time-disjoint** if F_1 accomplishes its activities at a different time than F_2 does.*

According to definitions 1, 2 and 3, the activities associated with an instruction can be modeled as the union of some specific disjoint *functionalities*. More in detail, the problem of model identification consists in determining the set, whose cardinality is k, of disjoint *functionalities* involved in the execution of a generic instruction, the average current absorbed by each *functionality* during its activation (if_j) and the relation between *functionalities* and each instruction ($a_{s,j} \geq 0$) such that the current drawn by each instruction can be approximated with a linear combination of the currents related to the *functionalities*.

DEFINITION 4. *A model is* ***compatible*** *if and only if the current absorbed by each instruction can be expressed as a linear combination of the currents associated with a set of disjoint functionalities.*

As an example, consider a simple decomposition in two *functionalities*: *fetch & decode* and *execute*. These two phases are time-disjoint even if the they are not space-disjoint: some of the activities necessary for *fetch & decode*, in fact, are also performed during the *execute* phase. To verify the compatibility property, the covariance matrix that defines the model must be computed and the principal components analysis should be applied [7]. These data reveal whether or not the identified *functionalities* are reasonably independent, and, in this case, how much each of them contributes to the complete model. A compatible model is said to be *feasible* if the energy associated with any instruction is not less than zero. Furthermore a model is said to be *reliable* if it is both *compatible* and *feasible* and its estimator is *unbiased*.

DEFINITION 5. *Let $d(\boldsymbol{q})$ be some data depending on a set of parameters $\boldsymbol{q}$ and let $\hat{\boldsymbol{q}} = f(d(\boldsymbol{q}))$ be the estimated value of the parameters. The function $f(\cdot)$ is an* ***estimator*** *for a given system, and $\hat{\boldsymbol{q}}$ are the estimated parameters, if and only if it is* ***unbiased*** *that is $E[\hat{\boldsymbol{q}}] = \boldsymbol{q}$, where $E[\hat{\boldsymbol{q}}]$ is the expectation value of $\hat{\boldsymbol{q}}$.*

The adopted model identification procedure is structured on a sequence of steps:

- Identification of a functional decomposition, referring to a generic processor instruction architecture and detecting disjoint *functionalities*.
- Identification of the relation between each instruction and the set of *functionalities* involved. For instance, the instruction `MOV CX,10` (Intel 80486DX2) is characterized by a register writing, while `ADD CX,10` implies a computation and a register writing. This step leads to an over-constrained linear system.
- Estimation of the current associated with each *functionality* using the least square method.

Let m be the cardinality of the instruction set $\mathcal{S}$. The energy associated with instruction $s \in \mathcal{S}$ can be expressed as:

$$e_s = V_{dd} \cdot i_s \cdot n_{ck,s} \cdot \tau \tag{1}$$

where i_s is the average current per clock cycle absorbed by instruction s, V_{dd} is the supply voltage, $n_{ck,s}$ is the total number of clock cycles and τ is the clock period. If if_j is the contribution, in terms of current consumption, of the *functionality* j and $a_{s,j}$ expresses the contribution of the *functionality* j in the execution of instruction s, the total current i_s absorbed by instruction s can be expressed as:

$$i_s \cdot n_{ck,s} = \sum_{j=1}^{k} (if_j \cdot a_{s,j}) + r_s \tag{2}$$

where r_s is a residual due to modeling errors. Taking into account $q \leq m$ instructions whose energy characterization is known, a linear system of q equations in k unknowns (the *functionality* currents) is obtained. In such a system the coefficients $a_{s,j}$ are known since they are derived from the analysis of each instruction in terms of the *functionalities* of the model. The procedure to determine the value of the parameters $a_{s,j}$ is presented in section 3.2.

2.2 Mathematical model

Let $\mathcal{S}$ be the set of all instructions of a given microprocessor, $\mathcal{S}_L \subseteq \mathcal{S}$ be the *learning-set* constituted by the instructions used to tune the model and $\mathcal{S}_G = \mathcal{S} - \mathcal{S}_L$ be the *generalization-set*. Let k be the number of identified *functionalities* and let $m_L > k$ be the cardinality of the energy-characterized *learning-set* $\mathcal{S}_L$. Then, let $\mathbf{A}$ be the $m_L \times k$ matrix whose entries are the activation coefficients $a_{s,j}$, $\mathbf{IF}$ be the $k \times 1$ column vector whose entries are the unknown currents if_j, and $\mathbf{IN}$ be the $m_L \times 1$ column vector whose elements are the known terms $i_s \cdot n_{ck,s}$. The linear system:

$$\mathbf{IN} = \mathbf{A} \times \mathbf{IF} \tag{3}$$

represents the available knowledge on the variables that have to be estimated. Let $\widehat{i_s}$ be an estimate of i_s and $\widehat{\mathbf{IF}}$ be an estimate of the real parameters $\mathbf{IF}$. The minimization of the square error $\|\mathbf{IN} - \widehat{\mathbf{IN}}\|^2$ leads to the equation:

$$\widehat{\mathbf{IF}} = (\mathbf{A}^T \times \mathbf{A})^{-1} \times \mathbf{A}^T \times \mathbf{IN} \tag{4}$$

In order to estimate the model parameters $\widehat{\mathbf{IF}}$, the columns of matrix $\mathbf{A}$ must be linearly independent: this guarantees that the model is *identifiable* with respect to the measurements available. In fact, two columns are linearly dependent if the same two *functionalities* are involved, with the same weight $a_{s,j}$, in the characterization of all the instructions in the *learning-set*. In this case, the two *functionalities* are not disjoint and could be conveniently merged in a single, new, *functionality*. When two or more columns are linearly dependent, there are two possible ways to solve the problem: changing the instructions in the *learning-set* or modifying the functional decomposition either increasing or reducing the model granularity.

2.3 Model characterization

Equation 4 gives the set of estimated parameters based on the relations between current measurements and $a_{s,j}$. This set of parameters constitutes a *reliable* model if its estimator is *unbiased* (definition 5). Combining equations (2) and (3) and indicating with $\mathbf{R} = \{r_s\}$ the residual vector, leads to the complete model, summarized in the equation:

$$\mathbf{IN} = \mathbf{A} \times \mathbf{IF} + \mathbf{R} \tag{5}$$

Solving the system (5) in the least square sense yields:

$$\widehat{\mathbf{IF}} = (\mathbf{A}^T \times \mathbf{A})^{-1} \times \mathbf{A}^T \times \mathbf{IN} \tag{6}$$

Indicating with $\mathbf{A}^*$ the Moore-Penrose pseudo-inverse and substituting the expression for $\mathbf{IN}$, equation (6) becomes:

$$\widehat{\mathbf{IF}} = \mathbf{IF} + \mathbf{A}^* \times \mathbf{R} \tag{7}$$

This equation represents the relation between estimated and actual parameters. The model is completely characterized, from a statistical point of view, when both the expectation value and the variance of its parameters are calculated. Applying the definitions of expectation value and variance to the specific model gives:

$$E\left[\widehat{\mathbf{IF}}\right] = E[\mathbf{IF}] + \mathbf{A}^* \times E[\mathbf{R}] \tag{8}$$

$$VAR\left[\widehat{\mathbf{IF}}\right] = E\left[(\mathbf{A}^* \times \mathbf{R} - \mathbf{A}^* \times E[\mathbf{R}])^2\right] \tag{9}$$

By assuming the residual is a gaussian noise $G(0, \lambda^2)$, where 0 is the expectation value and λ^2 is the variance, the following relations hold:

$$\begin{cases} E[\mathbf{R}] = 0 \\ E\left[\mathbf{R} \times \mathbf{R}^T\right] = \lambda^2 \cdot \mathbf{I} \end{cases} \tag{10}$$

The second relation implies that $E\left[\mathbf{R} \times \mathbf{R}^T\right]$ is a diagonal matrix and thus $E[r_i \cdot r_j]$ must be 0 for $i \neq j$ and λ^2 for $i = j$. Under these assumptions, the estimator is unbiased, that is:

$$E\left[\widehat{\mathbf{IF}}\right] = \widehat{\mathbf{IF}} \tag{11}$$

$$VAR\left[\widehat{\mathbf{IF}}\right] = \lambda^2 \cdot (\mathbf{A}^T \times \mathbf{A})^{-1} \tag{12}$$

Nevertheless, λ^2 is unknown since it depends on the residual vector $\mathbf{R}$. The value of λ^2 has thus to be replaced with an estimate $\hat{\lambda}^2$. By indicating with $\widehat{\mathbf{R}} = \widehat{\mathbf{IN}} - \mathbf{IN}$ the vector of the estimated modeling errors, an estimator of the variance is:

$$\hat{\lambda}^2 = \|\widehat{\mathbf{R}}\|^2/(m-k) \tag{13}$$

where, again, m is the number of samples and k is the number of parameters of the model. The method is applicable if and only if the distribution of the residuals obtained is the gaussian $G(0, \lambda^2)$. The mean value of the residuals:

$$\mu_R = \frac{1}{m} \cdot \sum_{s=1}^{m} r_s \tag{14}$$

depending on a statistical model is a statistical variable and has an expectation value and a variance given by:

$$E[\mu_r] = \mu_r \tag{15}$$

$$VAR[\mu_r] = \lambda^4/m \tag{16}$$

To test the null hypothesis $\mu_r = 0$ with a 95% confidence level, according to a $Z_{0.95}$ test, the following inequalities must be satisfied:

$$-\frac{1.96 \cdot \lambda^2}{\sqrt{m}} \leq \mu_r \leq +\frac{1.96 \cdot \lambda^2}{\sqrt{m}} \tag{17}$$

3. EXPERIMENTAL RESULTS

This section collects the experimental results obtained using the proposed model on a set of five commercial microprocessors. The first paragraph is an analysis of the characteristics of assembly languages aimed at the extraction of the *functionalities*. In the remaining paragraphs, the identified model is applied on a set of processors and its accuracy and generalization capabilities are shown.

3.1 Identification of functionalities

The goal of this first analysis is to extract the *functionalities* into which the execution of a generic instruction on a generic microprocessor can be decomposed. A first simple decomposition leads to two disjoint *functionalities*: *fetch & decode* and *execute*. It is intuitive that the *fetch & decode*, denoted in the following as $F\&D$, can be considered atomic in the sense that the tasks it performs need not to be further differentiated. The *execute*, on the other hand, performs a number of tasks that greatly differ from instruction to instruction and thus a more detailed modeling is necessary.

An accurate analysis [9], supported by the measured power consumption figures reported in the following sections, has led to the following conclusions:

- A single functionality, denoted as $F\&L$, performs arithmetic, logic, comparison, etc. integer operations.
- Data transfer operations may or may not access memory and this, intuitively, affects the power consumption. The functionality $L\&S$ (Load/Store) is responsible of memory read/write and stack operations. The data transfers that operate on registers are accounted for introducing the *functionality* WrR (Write Register). It is worth noting that reading a register does not significantly alter the power consumption with respect to a write operation.
- Conditional or unconditional jumps and procedure calls require some peculiar operations and are thusmodeled with a *functionality*, denoted as Br (Branch).
- Floating-point instructions are usually performed by a specific arithmetic unit. At a functional level of abstraction, this unit is not distinguishable from an integer ALU. For this reason the $A\&L$ is used to model these operations also.
- String operations are usually performed repeating data transfer and/or compare operations with an implicit register used as a counter. Their power consumption can thus be modeled by means of a combination the previously defined *functionalities*.

A generic microprocessor can thus be decomposed in the five *functionalities* listed in table 1.

Functionality	**Activities**
$F\&D$	Fetch and Decode
$A\&L$	Arithmetic and Logic
WrR	Write Register
$L\&S$	Load and Store
Br	Branch

Table 1: A possible functional decomposition

The *functionalities* stimulated by an instruction not only depend on the operation but also on the addressing mode. Table 2 shows the relation between some common addressing modes and the *functionalities* involved. Note that the computation of an address is not functionally associated with $A\&L$ but rather with $L\&S$ or Br. The completion of an instruction requires both executing an operation and accessing zero or more operands. According to the decomposition into op-code and addressing mode, the characterization of each instruction is obtained by computing the union of

Addressing mode	***Functionalities***
Register	[WrR]
Relative	$L\&S$
Indexed	$L\&S$
Auto-increment	$L\&S$, $A\&L$, WrR
Indexed and offset	$L\&S$

Table 2: Addressing modes and related functionalities

the set of *functionalities* associated with the operation and the sets of functionalities relative to the addressing mode of each operand. For instance, consider the instruction `ADD R3,(R2)+`: the `ADD` operation stimulates $F\&L$, the destination operand `R3` uses WrR and the source operand `(R2)+` uses $L\&S$, $A\&L$ and WrR. The *functionalities* stimulated by the complete instruction are thus $\{A\&L\} \cup \{WrReg\} \cup \{Ld\&St, A\&L, WrReg\} = \{Ld\&St, A\&L, WrReg\}$.
According to the previous analysis, the *functionalities* extracted represent a possible partition of the tasks performed by a generic microprocessor. This partition is compliant to definitions 1 and 4 and is an acceptable basis for the mathematical model. However, its correctness from a statistical point of view ought to be verified by computing the covariance matrix and then applying the principal components analysis. This technique aims at showing that the parameters are independent of each other and that all give a significant contribution to the model. For each processor, 500 randomly selected learning-sets of 8 characterized instructions have been used to calculate different estimates of the parameters. The outcome is a matrix with 5 columns (the parameters) and 500 rows (the samples). The first step consists in computing the normalized correlation matrix (a 5×5 matrix) in which the main diagonal elements are all ones. Off-diagonal elements, being much smaller than 1.0, indicate that the selected parameters are nearly independent as the model requires. The principal components are the eigenvectors of the correlation matrix and their relative contribution to the overall model is expressed by the corresponding eigenvalues. The mean values, over all the available processors, of the normalized eigenvalues, represent the relative contribution of each parameter (i.e. *functionality*) and are reported in table 3.

$F\&D$	Br	WrR	$A\&L$	$L\&S$
0.0789	0.0784	0.1336	0.1980	0.5111

Table 3: Relative contribution of the different *functionalities*

According to these results, no functionalities can be neglected without affecting the accuracy of the resulting model. Table 3 points out that the contribution of $F\&D$ and Br is less important compared to the remaining units. Nevertheless, a more detailed analysis indicates that some specific *functionalities*, due to their low impact on the overall model, might be neglected for some processors, but not in general. In the next paragraphs the mathematical model will be built according to the proposed functional decomposition.

3.2 Instruction characterization methodology

Once a functional model has been identified, the instruction set must be characterized by assigning a value to the coefficients $a_{s,j}$ corresponding to the five *functionalities* previously determined. To clarify the procedure adopted for instruction characterization, consider a decomposition in $F\&D$ and $Exec$. In this case, the equation for instruction s is:

$$a_{s,F\&D} \cdot if_{F\&D} + a_{s,Exec} \cdot if_{Exec} = i_s \cdot n_{ck,s} \qquad (18)$$

Characterizing an assembly language on the basis of this decomposition, consists in assigning the two coefficients $a_{s,F\&D}$ and $a_{s,Exec}$ for each instruction s. Since the power consumption depends on the number of cycles taken to fetch, decode and execute the instruction, a reasonable choice is:

$$\begin{cases} a_{s,F\&D} = n_{ck,s,F\&D} \\ a_{s,Exec} = n_{ck,s,Exec} \end{cases} \qquad (19)$$

where $n_{ck,s,F\&D}$ and $n_{ck,s,Exec}$ indicate the number of clock cycles needed for the Fetch & Decode and the Execution functionalities of instruction s, respectively. In a more complex model, constituted by $F\&D$ and $k-1$ disjoint execution *functionalities* ($F\&L$, WrR, etc.) the sum of these $k-1$ coefficients should equal the number of clock cycles needed for execution. By indicating with if_1 the $F\&D$ and with $if_2, \ldots, if_k$ the $k-1$ execution *functionalities*, the following relations must be satisfied:

$$\begin{cases} a_{s,1} = n_{ck,s,F\&D} \\ \sum_{j=2}^{k} a_{s,j} = n_{ck,s,Exec} \end{cases} \qquad (20)$$

Based on the analysis presented in the previous paragraph, it is straightforward to state whether or not a *functionality* is involved in the execution of a given instruction. As a consequence, instructions can be characterized by means of *activation coefficients* $b_{s,j} \in \{0,1\}$, where $b_{s,j} = 1$ indicates that the j-th *functionality* is involved in instruction s. To satisfy equation (19), the activation coefficients $b_{s,j}$ and the coefficients $a_{s,j}$ must be related by the equation:

$$a_{s,j} = \begin{cases} b_{s,j} \cdot n_{ck,s,F\&D} & j = 1 \\ b_{s,j} \cdot w_s & j \neq 1 \end{cases} \qquad (21)$$

where the weight w_s is given by:

$$w_s = \begin{cases} 0 & \sum_{j=2}^{k} b_{s,j} = 0 \\ n_{ck,s,Exec} / \sum_{j=2}^{k} b_{s,j} & \text{otherwise} \end{cases} \qquad (22)$$

In the following paragraphs, this methodology is applied to a set of microprocessors and its adaptability and generalization properties are shown.

3.3 Estimation

The Intel i80486DX [8] is used to present the approach. Table 4 shows the average currents drawn per clock cycle for a subset of instructions along with the number of clock cycles, the total current (which is proportional to the total energy) and the characterization in terms of the five *functionalities* identified. The coefficients $a_{s,j}$, are obtained by using equations (21) and (22). The results, relative to 18 energy characterized instructions of the sample microprocessor, are shown in figure 1. The value of the *functionalities* currents and standard deviations are reported in table 5. To verify the correctness of the gaussian noise hypothesis, residuals have to be analyzed. The $Z_{0.95}$ test confirms that the gaussian noise assumption holds: in fact, the mean estimated error $\mu_R = 9.94 \times 10^{-11}$ falls in the range $Z_{0.95} = \pm 40.75$. The accordance between actual data and estimated data is thus satisfactory and similar results, reported in table 6 have been obtained for all other microprocessors analyzed.

3.4 Generalization

To verify the generalization capabilities of the model this procedure has been repeated a significant number of times:

1. select a *learning-set* such that the least square problem is non-singular and well-conditioned;
2. solve the problem;

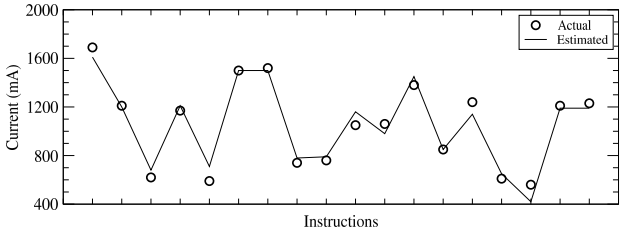

Figure 1: Intel i80486DX power estimates

Instruction	n_{ck}	$i_s \cdot n_{ck}$	$b_{s,j}$ F&D	Br	WrR	A&L	L&S
`ADD DX,BX`	2	313.6	1	0	1	1	0
`CMP [BX],DX`	3	776.0	1	0	0	1	1
`JMP label`	4	1119.0	1	1	0	0	0
`JZ label`	2	355.9	1	1	0	0	0
`MOV [BX],DX`	2	521.7	1	0	0	0	1
`NOP`	2	275.7	1	0	0	0	0

Table 4: Instruction characterization of Intel i80486DX

3. estimate the currents for the *generalization-set*;

4. measure the average learning and generalization errors.

It is important, to ensure the accuracy of the results, that the *learning-set* yields to a well-conditioned system. A problem is well-conditioned if:

$$rcond(\mathbf{A}_L^T \times \mathbf{A}) \geq 0.01 \tag{23}$$

where $\mathbf{A}_L$ is the sub-matrix of $\mathbf{A}$ associated to the *learning-set*, $rcond(\cdot)$ denotes the reciprocal of the condition number in the 1-norm and 0.01 is a reasonable, arbitrary, threshold. The procedure has been repeated varying the *learning-set* cardinality. For each *learning-set* size, 100 different, randomly selected, *learning-sets* have been used. Despite the randomness of the *learning-sets* used, the accordance between the actual currents and the estimates is good. This

Functionality	Current (mA)	STD
F&D	421.41	48.43
Br	355.06	26.98
WrR	228.48	46.46
A&L	228.33	38.27
L&S	505.99	39.90

Table 5: *Functionality* currents and estimated STDs

procedure has been repeated for 500 times for each processor leading to a confirmation of the results just reported: the average error obtained (absolute value) is less than 9%. The model allows thus the extrapolation, with a good confidence of the power consumption of an instruction not in the *learning-set*. Concerning the analysis of real-world software, a commercial 16 synchronous/asynchronous links controller adopted as test vehicle in the SEED ESPRIT project[10] has been used to stress the methodology. Two microprocessor implementations have been considered based on the ARM7TDMI and MC68000 cores, producing and average relative error of 2.77% and 4.4%, respectively.

Processor		F&D	Br	WrR	A&L	L&S
ARM7	i_s	5.7	14.3	13.0	18.3	13.9
	Std	1.1	0.7	0.5	0.6	0.7
i960JF	i_s	362.0	261.9	302.2	320.0	380.6
	Std	8.5	13.1	8.0	8.1	8.9
i960HD	i_s	970.4	692.8	804.8	775.4	1026.3
	Std	17.9	28.4	18.5	18.6	46.5
MB86934	i_s	218.2	0.0	194.7	175.9	190.6
	Std	9.0	0.0	19.2	18.7	21.4

Table 6: Currents and standard deviations

4. CONCLUSIONS

The paper presented a model to characterize instructions energy consumption of 32-bit microprocessors. As confirmed by the experiments performed, the proposed modeling approach shows good generalization properties and allows the extrapolation of the power consumption of uncharacterized instructions. The adopted approach deals with the static aspects of the power consumption, ignoring dynamic effects such as pipeline flush/refills and caching, whose effects in many cases are hidden by the measurement error. The proposed model can be thus used as the basis for a more comprehensive power estimation methodology.

5. REFERENCES

[1] E.Macii, M.Pedram and F.Somenzi, *"High-Level Power Modeling, Estimation, and Optimization,"* IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 17, n. 11, pp. 1061–1079, 1998.

[2] T.Sato, M.Nagamatsu and H.Tago *"Power and performance simulator: ESP and its application for 100MIPS/W class RISC design,"* Proc. of IEEE Symposium on Low Power Electronic, pp. 46–47, San Diego, CA, October 1994.

[3] P.Landman and J.Rabey *"Black-box capacitance models for architectural power analysis,"* Proc. of International Workshop on Low Power Design, pp. 165–170, Napa, CA, April 1994.

[4] V.Tiwari, S.Malik and A.Wolfe *"Power Analysis of Embedded Software: a First Step towards Software Power Minimization,"* IEEE Transactions on VLSI Systems, vol. 2, n. 4, pp. 437–445, 1994.

[5] V.Tiwari and M.T.C. Lee *"Power analysis of a 32-bit Embedded Microcontroller,"* VLSI Design Journal, 1996.

[6] J.Russell, M.F.Jacome *"Software Power Estimation and Optimization for High Performance, 32-bit Embedded Processors,"* Proc. of International Conference on Computer Design, Austin, TX, October 1998.

[7] A.Papoulis *"Probability, Random Variables and Stochastic Processes,"* McGraw-Hill, 1984.

[8] V.Tiwari, S.Malik, and A.Wolfe *"Power Analysis of the Intel 486DX2,"* Internal report CE-M94-5, Princeton University, June 1994.

[9] J.L.Hennessy and D.A.Patterson *"Computer Architecture - A Quantitative Approach, II ed.,"* Morgan Kaufmann Publishers, 1996.

[10] A.Allara, M.Bombana, W.Fornaciari and F.Salice *"A Case Study in Design Space Exploration: The TOSCA Environment Applied to a Telecom Link Controller,"* IEEE Design & Test of Computers, to appear, 2000.