Static Power Modeling of 32-bit Microprocessors

Carlo Brandolese, Fabio Salice, *Member, IEEE*, William Fornaciari, *Senior, IEEE*, Donatella Sciuto, *Member, IEEE*

Abstract— **The paper presents a novel strategy aimed at modelling instruction energy consumption of 32-bits microprocessors. Differently from former approaches, the proposed instruction-level power model is founded on a functional decomposition of the activities accomplished by a generic microprocessor. The proposed model has significant generalization capabilities. It allows estimation of the power figures of the entire instruction-set starting from the analysis of a subset, as well as to power characterize new processors by using the model obtained by considering other microprocessors. The model is formally presented and justified and its actual application over five commercial microprocessors is included.This static characterization is the basic information for system-level power modelling of hardware/software architectures.**

Keywords— **Power Modelling and Estimation, Low-power design, Embedded systems, Modelling, Estimation**

I. Introduction

The use of cooperating hardware and software components is a popular solution for embedded applications to concurrently meet time-to-market, low cost and flexibility. The increasing relevance of power requires to predict with reasonable confidence the power consumption of both hardware and software. While methodologies for hardware power estimation are well established [1], those for software components require further insights. The power estimation approaches for microprocessors reported in literature, fall in two main classes, working at the architectural and instruction level. The first class exploits the main strategies identified for the hardware to characterize blocks of the microprocessor architecture [2] [3] [4]. A power cost is assigned to each architectural module composing the system by considering the average capacitance to be charged when the module is stimulated. Statistical power models have also been proposed. These models are simulation-based and the activity factors are computed over typical input streams. This solution takes full advantage of assessed techniques and EDA tools but exhibits some limitations, due to the lack of details on the internal structure of the microprocessor cores. Furthermore, this fine- grain analysis is time-consuming and the correlation among module activities is hard to take into account and may produce inaccurate results. To overcome these problems, instruction-level measurement-based models have been proposed [5] [6] [7]. The key point lays in measuring the current drawn by the processor as it executes a long sequence of the same instruction and considering the average current absorbed as representative of such an instruction. This procedure has to be repeated for all instructions to completely characterize the microprocessor model. However, these approaches are processor-dependent by construction: for this reason, they do not exhibit any generalization capability over different microprocessor architectures. In fact, to model an alternative CPU core, a new costly analysis of the entire instruction set has to be carried out. Furthermore, the confidence of the estimations is also seldom considered under a formal viewpoint: the statistical significance of the model of consumption is usually neither considered nor justified.

The approach here proposed belongs to the instruction-level class, focuses on 32-bit general-purpose microprocessors and overcomes the above limitations. In fact, it proposes a general methodology, independent of the specific processor, allowing to accurately estimate the static energy of an instruction set, as the basic information necessary to perform a more detailed system-level power consumption characterization, that requires dynamic information to be evaluated. The methodology abstracts from the architectural level and focuses on the *functionalities* involved in the instruction execution. The resulting functional model exhibits generalization capabilities and allows covering a broad range of 32-bits microprocessors architectures. As detailed in section II the static energy consumption of each instruction is obtained as linear combination of independent contributions corresponding to a set of disjoint functionalities. The analysis of the statistical properties of the model confirms two types of generalizations:

Intra-processor: a model built on a suitable subset of instructions allows the extrapolation of the static energy characterization of the whole instruction set;

Inter-processor: a model constructed on a set of even partially characterized microprocessors allows the extension of the results to cover new architectures. The absolute energy consumption of the whole instruction set can be obtained when a single reference instruction is characterized.

The main motivation for such a model, derives from the need of power-simulating at instruction level the software-bound section of embedded applications. Unfortunately, only a few microprocessor manufacturers have a detailed power characterization of their cores and even fewer are willing to disclose this information. One of the most attracting advantages offered by this methodology is that it allows an early *virtual* prototyping of the system on different target processors. In fact, power estimation of a software application running on a microprocessor requires detailed information on both the static power consumption of the single instructions executed, and by the dynamic information associated with the code execution with the actual input data applied. The overall power consumption will be obtained by suitably combining both these factors, as proposed in different approaches in literature [8] [5]. Therefore, the basic static characterization of the instruction set can be valuable within today's co-design methodologies to enable a fast and quantitative evaluation of alternative embedded processors and system architec-

tures, especially during the early stages of the design process. To validate the proposed methodology, as reported in section III, experiments have been performed on five commercial microprocessors. Section III shows all the different phases of the analysis applied to actual cases: identification of the functionalities, characterization of the instruction energy model, estimation of the power consumption of a single microprocessor and its generalization to power figures of generic microprocessors. Particular attention has been paid to show that all the assumptions are well founded under a statistical viewpoint. In section IV some conclusions are drawn to summarize the value of the proposed approach and to outline some future research efforts.

II. Model identification

The proposed approach aims at gathering the most significant computational properties of a set of 32-bit processors in order to define a simple and accurate static energy consumption model. It is worth emphasizing that the proposed model is built on an *a priori* knowledge of both the energy characterization of a set of instructions (of one or more processors) and the relevant functional characteristics of each instruction involved in the model identification. The obtained model produces a static estimation (data independent) of the energy consumed in average by each instruction whose functional characterization is unknown (generalization effect). In other words, starting from a significant subset of instructions of a given processor (e.g. i80486DX2), it is possible to extract a static estimation, reasonably accurate, of power characteristics of instructions that not belonging to the generation set (learning set) of the model. The model accuracy and its generalization capability depend on three factors: the number of instructions, the typological variety (RISC/CISC, arithmetic, branch, logical, etc.) of the instructions and, the model granularity. In particular, it is necessary to consider that the more the model is specialized the less it is exhibits generalization properties. As a consequence, the model generation involves a complex trade-off between accuracy and generalization. The procedure for model generation, its statistical properties and its generalization capability are discussed in the following sections. Section II-A introduces some definitions and outlines the properties that the model must satisfy; section II-B introduces the mathematical formalism; section II-C presents the complete model for the single-processor case and gives its statistical characterization; finally, section II-D generalizes the single-processor model to the multi-processor case.

A. Model definition

The adopted general model is based on a *functional decomposition* of the activities carried out by a generic microprocessor. The problem of the identification of a functional model for the energy consumption at the instruction level is investigated considering the relation that exits 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 modelled as the union of some specific functionalities. In order to estimate the static power consumption of an instruction, it is necessary that each functionality is characterized in terms of its average current absorption per clock cycle and that the functionalities are disjoint. In other words, the decomposition of a processor into functionalities has to be a partition. It is worth noting that the set of functionalities of the model does not represent a structural partition, but a purely functional partition. In this framework, the current absorbed by each instruction can be expressed as a sum of the currents associated with each functionality involved in the instruction execution.

More in detail, the problem of the model identification consists in determining the set, whose cardinality is k, of independent 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 associated with each instruction can be approximated with the linear combination of the currents absorbed by the functionalities. Consequently, the model of a generic instruction s is:

$$i_s = \sum_{j=1}^{k} if_j \cdot a_{s,j} \tag{1}$$

where i_s is the estimated value for the current consumption associated with instruction s.

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.

The concept expressed in this definition indicates that the groups of active functionalities, instruction by instruction, are either time-disjoint or space-disjoint or both, so that the total energy can be obtained by summing up the energy corresponding to each functionality. 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. For this reason, the above decomposition is a partition and thus the additive property on the energy (or, equivalently, on the currents) is applicable. To verify the compatibility property, the covariance matrix must be computed and the principal components analysis should be applied [9] [10]. 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 *feasible* if the energy consumed by any instruction is not less than zero.

Definition 5: Let $\mathcal{S}$ be the set of all instructions of a given processor, $\mathcal{S}_\mathcal{L} \subseteq \mathcal{S}$ be the *learning-set* composed by the instructions used to tune the model and $\mathcal{S}_\mathcal{G} = \mathcal{S} - \mathcal{S}_\mathcal{L}$ be the *generalization-set*. A model is *feasible* if and only if the estimated current absorbed by each instruction, in the learning- and generalization-sets, is greater than zero.

This trivial hypothesis has to be verified to make the proposed model adherent to the physical reality.

Theorem 1: A sufficient condition for a model to be feasible is that the current associated with each functional unit is greater than or equal to zero.

Proof: Since $a_{s,j} \geq 0$ and $if_j \geq 0$ by hypothesis, the product $if_j \cdot a_{s,j} \geq 0$ and thus the sum $if_1 \cdot a_{s,1} + if_2 \cdot a_{s,2} + \ldots + if_k \cdot a_{s,k} \geq 0$. ■

Unfortunately, it is not sufficient that the model is *compatible* and *feasible*: it also has to provide a realistic evaluation of data. To this purpose, the following definitions are introduced:

Definition 6: Let $d(\mathbf{q})$ be some data depending on a set of parameters $\mathbf{q}$ and let $\hat{\mathbf{q}} = f(d(\mathbf{q}))$ be the estimated value of the parameters. The function $f()$ is an estimator for a given system, and $\hat{\mathbf{q}}$ are the estimated values, if and only if it is *unbiased* that is $E[\hat{\mathbf{q}}] = \mathbf{q}$, where $E[\hat{\mathbf{q}}]$ is the expected value of $\hat{\mathbf{q}}$.

Definition 7: A model is *reliable* if and only if it is both *compatible* and *feasible* and the estimator used is *unbiased*.

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

In the first step, a functional decomposition is identified. This subdivision is obtained referring to a generic processor instruction-set architecture and detecting disjoint functionalities whose absorbed currents are independent of -or weakly correlated to- each other.

The second step consists in identifying the correspondence between each instruction in the learning-set, considering both the operating code and the addressing modes, 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 system of linear equartions.

The third step consists in computing the estimates of the current associated with each functionality. Since the number of available measures is larger than the number of parameters, the estimator used is the least square method.

Let m be the cardinality of the considered instruction set $\mathcal{S}$. The energy associated with the instruction $s \in \mathcal{S}$ is expressed by the relation:

$$e_s = \sum_{j=1}^{k} e_{s,j} = V_{dd} \cdot i_s \cdot n_{ck,s} \cdot \tau \qquad (2)$$

where $e_{s,j}$ is the energy absorbed by the j-th functionality involved in the instruction s, k is the number of functionalities considered, V_{dd} is the supply voltage, $n_{ck,s}$ is the total number of clock cycles of the instruction s and τ is the clock period. If if_j is the current consumption of the functionality j, and $a_{s,j}$ expresses the contribution of the functionality j in the execution of instruction s, the above relation can be rewritten as:

$$e_s = \sum_{j=1}^{k} e_{s,j} = V_{dd} \cdot \left[\sum_{j=1}^{k} (if_j \cdot a_{s,j}) + r_s \right] \cdot \tau \qquad (3)$$

where, $a_{s,j}$ is known for each instruction s and r_s is a residual. Comparing relations 2 and 3, for each instruction s, the following relation has to be verified:

$$\sum_{j=1}^{k} (if_j \cdot a_{s,j}) + r_s = i_s \cdot n_{ck,s} \qquad (4)$$

Taking into account $q \leq m$ instructions (m being the cardinality of $\mathcal{S}$) 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. As an example, consider the simplest possible decomposition in a *fetch & decode* (*F&D*) functionality and an *execute* (*Exec*) functionality. The equation for each instruction is thus:

$$if_{F\&D} \cdot a_{s,F\&D} + if_{Exec} \cdot a_{s,Exec} + r_s = i_s \cdot n_{ck,s} \qquad (5)$$

The procedure to determine the value of the parameters $a_{s,F\&D}$ and $a_{s,Exec}$ is detailed in section III.

B. Mathematical model

In this section the mathematical properties of the statistical model are investigated and a criterion to validate the correctness and generality of the static model is derived.

Let k be the number of identified functionalities and let $m_L > k$ be the cardinality of the energy-characterized instruction set $\mathcal{S}_\mathcal{L}$. 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:

$$\mathbf{IN} = \mathbf{A} \times \mathbf{IF} \qquad (6)$$

represents the available knowledge on the variables i_s 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 $\|\mathbf{IN} - \widehat{\mathbf{IN}}\|^2$ yelds:

$$\widehat{\mathbf{IF}} = (\mathbf{A}^T \times \mathbf{A})^{-1} \times \mathbf{A}^T \times \mathbf{IN} \qquad (7)$$

To estimate the model parameters $\widehat{\mathbf{IF}}$, the columns of matrix $\mathbf{A}$ must be linearly independent, otherwise the problem has infinitely many solutions and the model is *not identifiable* with respect to the measurements available.

Theorem 2: A model is **identifiable** with respect to a set of measurements if it admits a single solution. A necessary and sufficient condition for a model to be identifiable is that the columns of $\mathbf{A} = \{a_{s,j}\}$ are linearly independent.

Proof: If the columns of $\mathbf{A}$ are linearly dependent, the matrix $\mathbf{A}^T \times \mathbf{A}$ is singular, i.e. $|\mathbf{A}^T \times \mathbf{A}| = 0$, and the pseudo-inverse $\mathbf{A}^* = (\mathbf{A}^T \times \mathbf{A})^{-1} \times \mathbf{A}^T$ cannot be calculated. ■

Theorem 2 has the following meaning: 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. When two or more columns are linearly dependent, there are two possible ways to solve the problem.

A first solution consists in suitably changing the instructions in the learning-set. For example, if the model only consists of *Exec* and *F&D*, the instructions in the learning-set must differ with respect to these two functionalities. The learning-set should thus be composed of instructions with one-cycle and multi-cycles fetches and/or instructions with executions distributed over one or more cycles.

The second solution requires the modification of the functional decomposition either by increasing or by reducing the model granularity. Considering the functional decomposition of the previous example, a possible solution would be splitting the *Exec* functionality into more specific functionalities such as arithmetic, registers etc.

C. Single-processor model characterization and generalization properties

Equation 7 gives the set of estimated parameters based on the known relations between current measurements and weights $a_{s,j}$. This set of parameters is a *reliable* model if its estimator is not biased (definitions 6 and 7). Starting from the preliminary problem description (where $\mathbf{R}$ is the residual vector of the r_s):

$$\mathbf{IN} = \mathbf{A} \times \mathbf{IF} + \mathbf{R} \tag{8}$$

and solving the system in the least square sense:

$$\widehat{\mathbf{IF}} = (\mathbf{A}^T \times \mathbf{A})^{-1} \times \mathbf{A}^T \times \mathbf{IN} \tag{9}$$

Letting $\mathbf{A}^* = (\mathbf{A}^T \times \mathbf{A})^{-}1 \times \mathbf{A}^T$, the previous equation, representing the relation between estimated and actual parameters, becomes:

$$\begin{aligned} \widehat{\mathbf{IF}} &= \mathbf{A}^* \times \mathbf{IN} = \\ &= \mathbf{A}^* \times (\mathbf{A} \times \mathbf{IF} + \mathbf{R}) = \\ &= \mathbf{A}^* \times \mathbf{A} \times \mathbf{IF} + \mathbf{A}^* \times \mathbf{R} = \\ &= \mathbf{IF} + \mathbf{A}^* \times \mathbf{R} \end{aligned} \tag{10}$$

The model is completely characterized, from a statistical point of view, when the expectation value and the variance of its parameters are given. Applying the definitions of expectation value and variance to the specific model, formal expression for both are derived. The expectation value $E[\widehat{\mathbf{IF}}]$ of the model parameters $\widehat{\mathbf{IF}}$ is, in fact:

$$\begin{aligned} E[\widehat{\mathbf{IF}}] &= E[\mathbf{IF} + \mathbf{A}^* \times \mathbf{R}] = \\ &= E[\mathbf{IF}] + \mathbf{A}^* \times E[\mathbf{R}] \end{aligned} \tag{11}$$

and the variance $VAR[\widehat{\mathbf{IF}}]$ is given by the equation:

$$\begin{aligned} VAR[\widehat{\mathbf{IF}}] &= E\Big[\Big(\widehat{\mathbf{IF}} - E[\widehat{\mathbf{IF}}]\Big) \times \Big(\widehat{\mathbf{IF}} - E[\widehat{\mathbf{IF}}]\Big)^T\Big] = \\ &= E\Big[\Big(\widehat{\mathbf{IF}} - \mathbf{IF}\Big) \times \Big(\widehat{\mathbf{IF}} - \mathbf{IF}\Big)^T\Big] = \\ &= E\Big[\big(\mathbf{IF} + \mathbf{A}^* \times \mathbf{R} - \mathbf{IF} - \mathbf{A}^* \times E[\mathbf{R}]\big)^2\Big] \end{aligned} \tag{12}$$

Expanding the square in the last equation, eventually gives:

$$\begin{aligned} VAR[\widehat{\mathbf{IF}}] = \mathbf{A}^* \times \big(&E\Big[\mathbf{R} \times \mathbf{R}^T\Big] + \\ &+ \mathbf{R} \times E\Big[\mathbf{R}^T\Big] + \\ &+ E[\mathbf{R}] \times \mathbf{R}^T + \\ &+ E\Big[E[\mathbf{R}] \times E[\mathbf{R}^T]\Big]\big) \times (\mathbf{A}^*)^T \end{aligned} \tag{13}$$

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 are satisfied:

$$\begin{cases} E[\mathbf{R}] = 0 \\ E[\mathbf{R} \times \mathbf{R}^T] = \lambda^2 \cdot \mathbf{I} \end{cases} \tag{14}$$

Note that the gaussian noise hypothesis needs to be verified. While the first of these relations is straightforward, the second implies that $E[\mathbf{R} \times \mathbf{R}^T]$ is a diagonal matrix and thus:

$$E[r_i \cdot r_j] = \begin{cases} 0 & i \neq j \\ \lambda^2 & i = j \end{cases} \tag{15}$$

Under these assumptions, the estimator is *unbiased*, thus:

$$E[\widehat{\mathbf{IF}}] = \widehat{\mathbf{IF}} \tag{16}$$

and its variance is:

$$\begin{aligned} VAR[\widehat{\mathbf{IF}}] &= \mathbf{A}^* \times (\lambda^2 \cdot \mathbf{I}) \times (\mathbf{A}^*)^T = \\ &= \lambda^2 \cdot (\mathbf{A}^T \times \mathbf{A})^{-1} \end{aligned} \tag{17}$$

Unfortunately, λ^2 is unknown since it depends on the residual vector $\mathbf{R}$. For this reason the value of λ^2 has to be substituted by its estimation $\hat{\lambda}^2$. By indicating with $\widehat{\mathbf{R}} = \widehat{\mathbf{IN}} - \mathbf{IN}$ the vector of the estimated modeling errors, an estimator $\hat{\lambda}^2$ of the variance is:

$$\hat{\lambda}^2 = \frac{\|\widehat{\mathbf{R}}\|^2}{m - k} \tag{18}$$

where, again, m is the number of samples and k is the number of parameters of the model. The method described in the preceding paragraphs is applicable if and only if the distribution of the residuals obtained by using the proposed linear model is the gaussian $G(0, \lambda^2)$. The mean value of the residuals μ_R, depending on a statistical model, is in turn a statistical variable and has an expectation value and a variance:

$$\begin{cases} E[\mu_r] = \mu_r \\ VAR[\mu_r] = \lambda^4/m \end{cases} \tag{19}$$

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:

$$-\frac{1.96 \cdot \lambda^2}{\sqrt{m}} \leq \mu_r \leq +\frac{1.96 \cdot \lambda^2}{\sqrt{m}} \tag{20}$$

D. Multi-processor model characterization and generalization properties

Data collected from measures on five microprocessors (Intel i80486DX2 [11], SPARClite MB86934 [12], Intel i960JF, Intel i960HD [7], [13] and ARM7 TDMI [14]) shows that the same type of instruction has an energy consumption that strongly differs from processor to processor. Nevertheless, it has been observed that the *relative* current consumption of instructions of the same type (i.e. same operation and addressing mode) are of the same order of magnitude, that is, the ratio between the current absorbed by a generic instruction and the current absorbed by a reference instruction is nearly independent of the microprocessor. This suggests that by using the relative currents instead of absolute ones, a single general model for all processors can be extracted.

Let $\mathcal{P}$ be the set of available characterized processors, whose cardinality is p. By using the methodology outlined in the previous section, the parameters characterizing each processor can be extracted. These parameters not only represent the available knowledge on each processor, but also constitute the basis for the construction of a general model, that is a model capable of fitting and generalizing the currents absorbed by the instructions of a processor not in $\mathcal{P}$. The key idea is, on one hand, the use of relative currents, on the other hand, the use of a set of processors for learning. The relative current is defined as:

$$\begin{aligned} i_{rel,s} = \frac{i_s}{i_{ref}} = \frac{\sum_{j=1}^{k} if_j \cdot a_{s,j}}{i_{ref}} = \sum_{j=1}^{k} \frac{if_j}{i_{ref}} \cdot a_{s,j} = \\ = \sum_{j=1}^{k} if_{rel,j} \cdot a_{s,j} \end{aligned} \tag{21}$$

and, according to the equations for absolute currents, the total current is $i_{rel,s} \cdot n_{ck,s}$. For the generic q-th processor of the set $\mathcal{P}$, characterized by $\mathbf{A}_q$ and $\mathbf{IN}_{rel,q} = \{i_{rel,s,q} \cdot n_{ck,s,q}\}$, the following equation holds:

$$\mathbf{IN}_{rel,q} = \mathbf{A}_q \times \mathbf{IF}_{rel,q} + \mathbf{R}_{rel,q} \tag{22}$$

where $\mathbf{R}_{rel,q}$ is the residual vector of the $r_{rel,s,q}$. Solving the system in the least square sense yields:

$$\widehat{\mathbf{IF}}_{rel,q} = \mathbf{A}_q^* \times \mathbf{IN}_{rel,q} \tag{23}$$

The general model should depend on a single, general, set of parameters $\mathbf{IF}_{rel}$, rather than the processor-specific $\mathbf{IF}_{rel,q}$, i.e. it should be expressed by an equation of the form:

$$\mathbf{IN}_{rel,q} = \mathbf{A}_q \times \mathbf{IF}_{rel} + \mathbf{R}_{rel,q} \tag{24}$$

Combining equations (23) and (23) gives:

$$\begin{aligned} \widehat{\mathbf{IF}}_{rel,q} = \mathbf{A}_q^* \times \mathbf{IN}_{rel,q} = \\ = \mathbf{A}_q^* \times (\mathbf{A}_q \times \mathbf{IF}_{rel} + \mathbf{R}_{rel,q}) = \\ = \mathbf{IF}_{rel} + \mathbf{A}_q^* \times \mathbf{R}_{rel,q} \end{aligned} \tag{25}$$

Adding up these relations for all indices q corresponding to the p available processors, leads to the following relation:

$$\begin{aligned} \sum_{q=1}^{p} \widehat{\mathbf{IF}}_{rel,q} = \sum_{q=1}^{p} \mathbf{IF}_{rel} + \sum_{q=1}^{p} \mathbf{A}_q^* \times \mathbf{R}_{rel,q} = \\ = p \cdot \mathbf{IF}_{rel} + \sum_{q=1}^{p} \mathbf{A}_q^* \times \mathbf{R}_{rel,q} \end{aligned} \tag{26}$$

that, dividing by p, becomes:

$$\frac{1}{p} \cdot \sum_{q=1}^{p} \widehat{\mathbf{IF}}_{rel,q} = \mathbf{IF}_{rel} + \frac{1}{p} \cdot \sum_{q=1}^{p} \mathbf{A}_q^* \times \mathbf{R}_{rel,q} \tag{27}$$

Equation (27) indicates that, apart from an error whose statistical properties will be detailed in the following, an estimate of the parameters characterizing the general model can be the mean value of the estimated parameters of each processor in the set $\mathcal{P}$. To prove that the *mean* estimator is adequate, it is necessary to verify that it is *unbiased*. To do this, the expectation value and the variance of the parameters have to be calculated. The expectation values of the two sides of the previous equation are:

$$\begin{aligned} E\left[\frac{1}{p} \cdot \sum_{q=1}^{p} \widehat{\mathbf{IF}}_{rel,q}\right] = E[\mathbf{IF}_{rel}] + E\left[\frac{1}{p} \cdot \sum_{q=1}^{p} \mathbf{A}_q^* \times \mathbf{R}_{rel,q}\right] = \\ = \mathbf{IF}_{rel} + \frac{1}{p} \cdot \sum_{q=1}^{p} \mathbf{A}_q^* \times E[\mathbf{R}_{rel,q}] \end{aligned} \tag{28}$$

Under the gaussian noise hypothesis for each processor, the expectation values $E[\mathbf{R}_{rel,q}]$ are known to be zero, thus:

$$E\left[\frac{1}{p} \cdot \sum_{q=1}^{p} \widehat{\mathbf{IF}}_{rel,q}\right] = \mathbf{IF}_{rel} \tag{29}$$

An estimate of $\mathbf{IF}_{rel}$ is thus:

$$\widehat{\mathbf{IF}}_{rel} = \frac{1}{p} \cdot \sum_{q=1}^{p} \widehat{\mathbf{IF}}_{rel,q} \tag{30}$$

This general model is completely characterized from a statistical point of view only if the expectation value and the variance of its parameters $\widehat{\mathbf{IF}}_{rel}$ are known. The expectation value is:

$$E[\widehat{\mathbf{IF}}_{rel}] = E\left[\frac{1}{p} \cdot \sum_{q=1}^{p} \widehat{\mathbf{IF}}_{rel,q}\right] = \mathbf{IF}_{rel} \tag{31}$$

and the variance, obtained applying the same method used

for the single-processor case, is:

$$
\begin{aligned}
VAR[\widehat{\mathbf{IF}}_{rel}] &= E\left[\widehat{\mathbf{IF}}_{rel} - E[\widehat{\mathbf{IF}}_{rel}] \times \widehat{\mathbf{IF}}_{rel} - E[\widehat{\mathbf{IF}}_{rel}]\right]^T = \\
&= E\left[\left(\frac{1}{p}\cdot\sum_{q=1}^{p}\mathbf{A}_q^* \times \mathbf{R}_{rel_q}\right) \times \left(\frac{1}{p}\cdot\sum_{q=1}^{p}\mathbf{A}_q^* \times \mathbf{R}_{rel_q}\right)^T\right] = \\
&= \frac{1}{p^2}\cdot E\left[\sum_{q=1}^{p}\sum_{r=1}^{p}\mathbf{A}_q^* \times \mathbf{R}_{rel,q} \times \mathbf{R}_{rel,r}^T \times (\mathbf{A}_r^*)^T\right] = \\
&= \frac{1}{p^2}\cdot\sum_{q=1}^{p}\sum_{r=1}^{p}\mathbf{A}_q^* \times E\left[\mathbf{R}_{rel,q} \times \mathbf{R}_{rel,r}^T\right] \times (\mathbf{A}_r^*)^T
\end{aligned}
\tag{32}
$$

According to the previous hypothesis, the residuals $\mathbf{R}_{rel,q}$ and $\mathbf{R}_{rel,r}$ are gaussian noises and thus:

$$
E[\mathbf{R}_{rel,q} \times \mathbf{R}_{rel,r}^T] = \begin{cases} \mathbf{0} & q \neq r \\ \lambda_{rel,q}^2 \cdot \mathbf{I} & q = r \end{cases} \tag{33}
$$

The expression for the variance simplifies to:

$$
\begin{aligned}
VAR\left[\widehat{\mathbf{IF}}_{rel}\right] &= \frac{1}{p^2}\cdot\sum_{q=1}^{p}\mathbf{A}_q^* \times E\left[\mathbf{R}_{rel,q} \times \mathbf{R}_{rel,q}^T\right] \times (\mathbf{A}_q^*)^T = \\
&= \frac{1}{p^2}\cdot\sum_{q=1}^{p}\lambda_{rel,q}^2 \cdot \mathbf{A}_q^* \times (\mathbf{A}_q^*)^T = \\
&= \frac{1}{p^2}\cdot\sum_{q=1}^{p}\lambda_{rel,q}^2 \cdot \left[\mathbf{A}_q^T \times \mathbf{A}_q\right]^{-1} = \\
&= \frac{1}{p^2}\cdot\sum_{q=1}^{p} VAR\left[\widehat{\mathbf{IF}}_{rel,q}\right]
\end{aligned}
\tag{34}
$$

The meaning of this last equation is that by increasing the number p of considered processors, the variance of the parameters of the general model decreases.

III. Experimental results

This section collects the experimental results obtained using the proposed 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* needed for the proposed model. In the remaining paragraphs, the identified model is applied on a single processor first, and then on a set of processors and its accuracy and generalization capabilities are shown.

A. 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 functionality*, denoted in the following as *F&D*, can be considered atomic in the sense that the tasks it performs need not to be differentiated. The *execute functionality*, on the other hand, performs a number of tasks that greatly differs from instruction to instruction and thus a more detailed analysis is necessary. Table I shows the classes of operations provided by the majority of assembly languages [15].

TABLE I
Operation classes of assembly languages

Class	Operation
Arithmetic & logic	add, subtract
	and, or, not, exor
	multiply, divide
	compare
	shift
Data transfer	registers
	memory
	stack
Control	unconditional jumps
	conditional jumps
	calls and returns
System	exception handling
	interrupts
	system calls
Floating-point	add, subtract, compare
	multiply
Decimal	divide
	BCD arithmetic
	conversion
String	transfer
	compare
	search

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

- a single *functionality*, denoted in the following simply as *A&L*, performs arithmetic, logic, comparison, etc. integer operations;
- data transfer operations may or may not access memory and this, intuitively, affects the power consumption. A *functionality* to read and write memory is necessary. This unit will be denoted as *Ld&St* (Load/Store) and includes stack operation as well. The data transfers that operate on registers are accounted for including in the model the functionality *WrReg* (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 operation to be performed, such as destination address calculation, and thus their execution need to be modeled with the *Br* (Branch) *functionality*. This *functionality*is also involved in interrupt handling and system call instructions;
- 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, a specific *functionality* for floating-point arithmetic is not included in the model. The *A&L functionality* is used to model these operations;
- string operations are usually performed repeating data transfer and/or compare operations with an implicit register used as a counter. These operations can thus be modeled by means of the previously defined *functionalities*.

Functionally, a microprocessor can thus be decomposed in five*functionalities*. These are listed in Table II, while the relations between classes of operations and the involved *functionalities* are summarized in Table III.

TABLE II
A POSSIBLE FUNCTIONAL DECOMPOSITION

Functionality	**Activities**
F&D	Fetch and Decode
A&L	Arithmetic and Logic
WrReg	Write Register
Ld&St	Load and Store
Br	Branch

TABLE III
OPERATIONS AND RELATED FUNCTIONALITIES

Class	**Functionalities**
Arithmetic & logic	*F&D*, *A&L*
Data transfer	*F&D*
Control	*F&D*, *Br*
System	*F&D*, *Br*
Floating-point	*F&D*, *A&L*
Decimal	*F&D*, *Br*
String	*F&D*, *A&L*, *WrReg*

The *functionalities* stimulated by an instruction not only depend on the operation but also on the addressing mode. Table IV shows the relation between the most common addressing modes and the *functionalities* involved. Note that the calculation of an address is not functionally associated with the *A&L functionality* but with *Ld&St* or *Br*.

TABLE IV
ADDRESSING MODES AND RELATED FUNCTIONALITIES

Addressing mode	**Sample**	**Functionalities**
Register	`R2`	[*WrReg*]
Immediate	`#3`	-
Relative	`10(R2)`	*Ld&St*
PC Relative	`#100`	-
Indirect	`(R2)`	*Ld&St*
Indexed	`(R2+R3)`	*Ld&St*
Memory	`(100)`	*Ld&St*
Indirect Memory	`@(R2)`	*Ld&St*
Auto-increment	`(R2)+`	*Ld&St*, *A&L*, *WrReg*
Indexed and offset	`10(R2)[R3]`	*Ld&St*

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 computing the union of the set of *functionalities* relative to 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 the *A&L functionality*, the destination operand `R3` uses the *WrReg functionality* and the source operand `+(R2)` uses the *Ld&St*, *A&L* and *WrReg functionalities*. The *functionalities* stimulated by the complete instruction are thus:

$$\begin{aligned}\{A\&L\} \cup \{WrReg\} \cup \\ \cup \{Ld\&St, A\&L, WrReg\} = \\ = \{Ld\&St, A\&L, WrReg\}\end{aligned} \tag{35}$$

According to the previous analysis, the extracted *functionalities* represent a possible partition of the tasks performed by a generic microprocessor. It is worth noting that the number of functionalities should represent a trade-off between the available knowledge on the architectures being modeled and the accuracy obtainable. As a limiting situation consider a model with a single functionality: the only architectural knowledge required is the number of cycles taken to fetch, decode and execute each instruction. With such a trivial model, both the accuracy obtained in fitting and the generalization capabilities are reduced with respect to a more complex model. Figure 1 represents a comparison between a 1-functionality and a 5-functionalities models in terms of average errors.

Fig. 1. Comparison of models based on 1 and 5 functionalities

The proposed 5-functionalities partition, though arbitrary, 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 calculating the covariance matrix and then carrying out the same analysis steps of the principal components analysis [9]. This technique aims at showing that the parameters are independent from 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 calculating the normalized correlation matrix (a 5×5 matrix) in which the main diagonal elements are all ones. Off-diagonal elements of the covariance matrix, being much smaller than 1.0, indicate that the selected parameters are nearly independent as the model and the principal components analysis require. 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 V.

TABLE V
NORMALIZED CONTRIBUTION OF FUNCTIONALITIES TO THE MODEL

Functionality	*F&D*	*Br*	*WrReg*	*A&L*	*Ld&St*
Contribution	0.078	0.078	0.133	0.198	0.511

According to these results, no *functionalities* can be neglected without affecting the accuracy of the resulting

model. Nevertheless, as table V points out, the contribution of *F&D* and *Br* is less important compared to the remaining units. A more detailed analysis, summarized in table VI, qualitatively indicates the significance of each *functionality* for all processors. It is worth noting that a

TABLE VI
RELATIVE IMPORTANCE OF THE *functionalities*. THE SYMBOLS ↓, — AND ↑ INDICATE A LOW, MEDIUM AND HIGH IMPACT, RESPECTIVELY.

Processor	*F&D*	*Br*	*WrReg*	*A&L*	*Ld&St*
SPARClite	↓	—	—	↑	↑
Intel i486DX	—	—	↓	↑	↑
ARM7TDMI	↓	↓	—	↑	↑
Intel i960JF	—	—	↓	↑	↑
Intel i960HD	↓	—	—	—	↑

specific *functionalities*, due to its low impact on the overall model, might be neglected for some processors, but none of the *functionalities* can be ignored when the whole set of processors is considered.

In the next paragraphs the mathematical model will be built according to the proposed functional decomposition, which has proven correct.

B. 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 each instruction has the form:

$$a_{s,F\&D} \cdot if_{F\&D} + a_{s,Exec} \cdot if_{Exec} = i_s \cdot n_{ck,s} \tag{36}$$

The task of 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. Intuitively, since the power consumption depends on the number of cycles taken to fetch, decode and execute the instruction, a reasonable choice is:

$$\begin{cases} a_{s,F\&D} = n_{ck,s,F\&D} \\ a_{s,Exec} = n_{ck,s,Exec} \end{cases} \tag{37}$$

that is $a_{s,F\&D}$ is the number of clock cycles $n_{ck,s,F\&D}$ needed for fetch and decode, and $a_{s,Exec}$ is the number of clock cycles $n_{ck,s,Exec}$ needed for the execution phase of instruction s. In a more complex model, constituted by a *F&D functionality* and $k-1$ disjoint execution *functionalities* (*A&L*, *WrReg*, etc.) the sum of the $k-1$ coefficients associated with the execution *functionalities* should equal the number of clock cycles needed for the execution. By indicating with if_1 the *F&D functionality* and with $if_2, \ldots, if_k$ the $k-1$ execution *functionalities*, the following relations must be satisfied:

$$\begin{cases} a_{s,1} = n_{ck,s,F\&D} \\ \sum_{j=2}^{k} a_{s,j} = n_{ck,s,Exec} \end{cases} \tag{38}$$

Based on the analysis presented in the previous paragraph, it is possible to determine whether or not a *functionality* is involved in the execution of a given instruction. As a consequence, the involvement of a *functionality* is represented by means of an *activation coefficient* $b_{s,j} \in \{0,1\}$, where $b_{s,j} = 1$ indicates that the j-th *functionality* is involved in instruction s. The activation coefficients $b_{s,j}$ and the coefficients $a_{s,j}$ are related by the equation:

$$a_{s,j} = \begin{cases} b_{s,j} \cdot n_{ck,s,F\&D} & j = 1 \\ b_{s,j} \cdot w_s & j = 1 \ldots k \end{cases} \tag{39}$$

where the weight w_s is given by:

$$w_s = \begin{cases} 0 & \sum_{j=2}^{k} b_{s,j} = 0 \\ n_{ck,s,Exec} / \sum_{j=2}^{k} b_{s,j} & otherwise \end{cases} \tag{40}$$

In the following paragraphs, this methodology is applied to a set of microprocessors to verify its suitability in terms of adaptability and generalization properties.

C. Estimation on a single processor

The first step to verify the suitability of the model is to ensure that it adequately fits power consumption data on a single processor and that the gaussian noise hypothesis is satisfied. The processor used here to show the validity of the approach is the Intel i80486DX [11]. Tables VII and VIII show the average currents drawn per clock cycle for a subset of instructions, reported in [11], 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.

TABLE VII
AVERAGE CURRENTS AND CLOCK CYCLES OF INTEL I80486DX

Instruction	i_s	$n_{ck,F\&D}$	$n_{ck,Exec}$	$i_s \cdot n_{ck}$
`ADD DX,BX`	313.6	1	1	313.6
`CMP [BX],DX`	388.0	1	2	776.0
`JMP label`	373.0	1	3	1119.0
`JZ label`	355.9	1	1	355.9
`MOV [BX],DX`	521.7	1	1	521.7
`NOP`	275.7	1	1	275.7
`SAL BX,CL`	306.5	1	3	919.5

TABLE VIII
INSTRUCTION CHARACTERIZATION OF INTEL I80486DX ($b_{s,j}$)

Instruction	*F&D*	*Br*	*WrReg*	*A&L*	*Ld&St*
`ADD DX,BX`	1	0	1	1	0
`CMP [BX],DX`	1	0	0	1	1
`JMP label`	1	1	0	0	0
`JZ label`	1	1	0	0	0
`MOV [BX],DX`	1	0	0	0	1
`NOP`	1	0	0	0	0
`SAL BX,CL`	1	0	1	1	0

The characterization, in terms of the coefficients $a_{s,j}$, is obtained by using equations (39) and (40). Let $\mathbf{A} = \{a_{s,j}\}$ be the instruction characterization matrix for the considered processor and $\mathbf{IN} = \{i_s \cdot n_{ck,s}\}$ the column vector of

the total currents. The solution **IF** is obtained by solving the linear problem in the least square sense. The results, relative to 18 energy characterized [11] instructions of the sample microprocessor, are shown in figure 2.

Fig. 2. Intel i80486DX power estimates

The value of the *functionalities* currents and estimated standard deviations are reported in table IX.

TABLE IX
FUNCTIONAL UNIT CURRENTS AND ESTIMATED STANDARD DEVIATION

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

To verify the correctness of the gaussian noise hypothesis, residuals have been analyzed. The errors, measured as the difference between actual and estimated currents, give an estimate of the input residual $\mathbf{R}$. The $Z_{0.95}$ test is satisfied and thus the gaussian noise assumption holds (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 and estimated data is also satisfactory for all other processors analyzed, as shown in Tables X and XI.

TABLE X
FUNCTIONALITY CURRENTS AND STANDARD DEVIATIONS

Processor		if_1	if_2	if_3	if_4	if_5
ARM7TDMI	if	5.7	14.3	13.0	18.3	13.9
	σ^2	1.1	0.7	0.5	0.6	0.7
i960JF	if	362.0	261.9	302.2	320.0	380.6
	σ^2	8.5	13.1	8.0	8.1	8.9
i960HD	if	970.4	692.8	804.8	775.4	1026.3
	σ^2	17.9	28.4	18.5	18.6	46.5
SPARClite	if	218.2	0.0	194.7	175.9	190.6
	σ^2	9.0	0.0	19.2	18.7	21.4

TABLE XI
$Z_{0.95}$ TEST FOR GAUSSIAN NOISE

Microprocessor	m	μ_R	STD	$Z_{0.95}$
ARM7TDMI	32	1.00×10^{-10}	2.99	± 1.05
Intel i960JF	92	1.00×10^{-10}	56.79	± 11.87
Intel i960HD	92	1.00×10^{-10}	119.75	± 25.28
SPARClite MB86934	28	1.02×10^{-10}	30.44	± 12.21

D. Generalization on a single processor

In the previous section, it has been shown that the model adequately fits actual data on a range of microprocessors. It is now essential to prove that the proposed model exhibits generalization capabilities. To verify this, the following procedure has been repeatedly applied: i) from the set of available instructions, select a *learning-set*, whose cardinality is m_L such that the least square problem is non-singular and well-conditioned; ii) solve the problem; iii) estimate the currents for the instructions in the *generalization-set*; iv) measure the learning and generalization errors. It is important, to ensure the accuracy of the results, that the *learning-set* yields to a well-conditioned system. Let $\mathbf{A}_L$ be the sub-matrix of $\mathbf{A}$ relative to the *learning-set* and $\mathbf{A}_G$ the sub-matrix relative to the *generalization-set*. A problem is well-conditioned if:

$$rcond(\mathbf{A}_L^T \times \mathbf{A}_L) \geq 0.01 \tag{41}$$

where $rcond(\cdot)$ denotes the reciprocal of the condition number in the 1-norm [16] and 0.01 is a reasonable, arbitrary, threshold. The procedure has been repeated with the learning-set cardinality varying from a minimum of 6 samples to a maximum of 16 samples, that is the maximum number of samples available for all processors. For each *learning-set* size, 100 different, randomly selected, *learning-sets* have been used for the Intel i80486DX leading to the results shown in the graphs of Figure 3. Similar results have been obtained for the other microprocessors.

Fig. 3. Generalization error and standard deviation for i80486DX

Despite the random choice of the 100 subsets used in the analysis, the accordance between the actual currents and estimated currents is good. This procedure has been repeated for 500 times for each processor leading to the results summarized in figure 4.

Fig. 4. Learning and generalization relative error

Note that, since the errors tend to compensate, their mean value is very close to zero ($\approx 10^{-10}$): for this rea-

son the absolute value of the errors has been used to assess the accuracy of the methodology. The graph reports the average, computed over all processors and over all 500 different estimates, of the mean error on the *learning-set* and *generalization-set*, plotted against the size of the *learning-set*. Figure 4 shows that the mean of the absolute value of the errors computed on the *generalization-set* is less than 9%, confirming the generalization capability of the model. In other words, the model is capable of extrapolating, with a good confidence, the power consumption of an instruction not included in the used *learning-set*.

E. General processor model

The aim of this section is to show that the model is capable of fitting and generalizing power data, coming from different microprocessors, without any change to the rationale behind the functional characterization of each instruction.

The absolute current absorbed by an instruction strongly varies from a microprocessor to another. In general, an arbitrary current per clock cycle (e.g. the average or the maximum current of an arbitrary number of instructions, not necessarily the same for each processor) might be used as reference. In this approach, the current per clock cycle of a single, specific instruction is used as reference value since this choice exhibits two major advantages. On one hand, the accuracy of the estimates produced by using an arbitrary subset of instructions to compute the reference value is not better than that obtained by using a single and specific instruction. On the other hand, the actual power consumption of only one instruction, rather than a subset, has to be experimentally measured whenever the absolute—rather than relative—power characterization of a generic processor is required. The advantage of using the current per clock cycle of a single reference instruction is thus twofold. Table XII shows the ratios for three instructions on three microprocessors. The reference instruction selected is the `ADD Reg, Reg`.

TABLE XII
CURRENTS RELATIVE TO THE ADD REG,REG REFERENCE INSTRUCTION

Instruction	SPARClite	i80486DX	i960JF
`MOVE Reg,Reg`	0.985	0.964	0.941
`LOAD (Reg) Reg`	1.070	1.366	0.990
`SHIFT Reg,Imm`	0.977	0.959	0.993

A general processor-independent model is derived in the following using the relative currents. When a single processor at a time is considered, the behavior of such a model is the same obtained with absolute currents. In fact, in this case the actual data is simply divided by a constant factor: the reference instruction current. The aim of this section is to collect the results showing that a model derived using a set of processors is capable of modelling the power consumption of the instructions of other processors.

Let $\mathcal{P}$ be the set of available processors whose cardinality is $p = 5$ and $\mathcal{P}_L \subseteq \mathcal{P}$ be the *processors-learning-set*. Let $\mathcal{P}_{L,h}$ be a generic *processors-learning-set* with cardinality p_h and $\widehat{\mathbf{IF}}_{rel,q}$ be the estimated model parameters computed using all available instructions of the q-th processor. The general model parameters computed on the basis of the *processors-learning-set* $\mathcal{P}_{L,h}$ are given by:

$$\widehat{\mathbf{IF}}_{rel,h} = \frac{1}{p_h} \cdot \sum_{q=1}^{p_h} \widehat{\mathbf{IF}}_{rel,q} \tag{42}$$

The parameters $\widehat{\mathbf{IF}}_{rel,h}$ have been extracted for all $2^5 - 1$ possible *processors-learning-sets* and have been used to estimate the current consumption on all processors in the *processors-generalization-sets* $\mathcal{P} - \mathcal{P}_{L,h}$. Figure 5 reports the trend of the mean error, conservatively computed as the absolute value of the difference between the actual data and the estimates, for all *processor-learning-sets* with cardinality p_h from 1 to 5.

Fig. 5. General model mean error

Each point in the graph represents the mean of average errors calculated over all the processors in the corresponding *processor-generalization-set*. The first point of the series $p_h = 1$ corresponds to the learning set $\mathcal{P}_{L,1} =$ { MB86934 }, the second point corresponds to $\mathcal{P}_{L,2} =$ { i80486DX }, etc. The first point in the second series, $p_h = 2$, corresponds to the set $\mathcal{P}_{L,6} =$ { MB86934, i80486DX } and so on. The plot confirms that the model is adequately precise and general and that the variance of the mean error decreases as the number of processors in the *processor-learning-set* increases. According to Section II, the variance of the parameters of the general model is:

$$VAR\Big[\widehat{\mathbf{IF}}_{rel}\Big] = \frac{1}{p^2} \cdot \sum_{q=1}^{p} VAR\Big[\widehat{\mathbf{IF}}_{rel,q}\Big] \tag{43}$$

which is confirmed by the actual data depicted in figure 6.

Fig. 6. General parameters mean variance

As an example, consider a general model built using the parameters of the SPARClite MB86934 and ARM7TDMI architectures to estimate the power consumption of the Intel i80486DX. The accordance between actual data and estimated data, shown in figure 7, is satisfactory, with errors falling in the range between ±20%.

Fig. 7. Generalization of Intel i80486DX power consumption

IV. Conclusions

An approach to model the static instructions energy consumption of 32-bit microprocessors has been proposed. Differently from the strategies in literature, such as architectural-level and measurement-based instruction-level approaches, the proposed *functionality-based instruction-level* method estimates the current for each instruction by means of a linear combination of values associated with a set of five disjoint *functionalities*. As confirmed by the experiments performed, the proposed modeling approach shows notable accuracy and good generalization properties, both intra-processor and inter-processor, and allows extrapolating the power consumption (absolute or relative) of uncharacterized instructions.

The following table shows the estimated current values and standard deviations for the five *functionalities* of the model extracted by using the reference instruction `ADD reg, reg` and the five not completely characterized processors ARM7TDMI [14], Intel i960JF, Intel i960HD [7], [13], SPARClite MB86934[12] and Intel i80486DX.

TABLE XIII
Relative functionality currents and standard deviations

Functionality	*F&D*	*Br*	*WrReg*	*A&L*	*Ld&St*
Current	0.51	0.40	0.47	0.52	0.61
STD	0.007	0.004	0.006	0.006	0.007

By construction, the adopted approach allows the definition of the static aspects of the power consumption of each instruction. In this context, the *inter-instruction* effects are deliberately neglected. Two main considerations have driven this choice. First of all, the effects corresponding to the measure errors and the modeling inaccuracy mask the contribution related to the state of the processor. Second, other effects, such as those connected with the pipe and the memory hierarchy, and in particular the cache, are related to the dynamic components of the relation between instructions (in general, more than two) and they have to be considered by suitably characterizing the computation and the target architecture. Thus, the proposed model, given such assumptions, can be considered as the basic component of a more general power estimation framework, which includes the dynamic aspects related to the code execution, the input data and the architectural characteristics of the embedded system, i.e. memory hierarchy, pipeline effects and I/O sub-system.

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, no. 11, pp. 1061–1079, November 1998.

[2] T. Sato, M. Nagamatsu, and H. Tago, "Power and performance simulator: Esp and its application for 100mips/w class risc design," in *Proceedings of IEEE Symposium on Low Power Electronic*, San Diego, CA, October 1994, pp. 46–47.

[3] P.W. Ong and R.H. Yan, "Power-conscious software design: a framework for modeling software on hardware," in *Proceedings of IEEE Symposium on Low Power Electronic*, San Diego, CA, October 1994, pp. 36–37.

[4] P. Landam and J. Rabaey, "Black-box capacitance models for architectural power analysis," in *Proceedings of the IEEE International Workshop on Low Power Design*, Napa, CA, April 1994, pp. 165–170.

[5] 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, no. 4, pp. 437–445, December 1994.

[6] V. Tiwari and M.T.-C. Lee, "Power analysis of a 32-bit embedded microcontroller," in *Proceedings of the Asian and South Pacific Design Automation Conference*, Chiba, Japan, September 1995, pp. 141–148.

[7] J. Russel and M.F. Jacome, "Software power estimation and optimization for high performance, 32-bit embedded processors," in *Proceedings of the IEEE International Conference on Computer Design*, Austin, TX, October 1998, pp. 328–333.

[8] T.M. Conte, K.N. Menezes, S.W. Sathaye, and M.C. Toburen, "System-level power consumption modeling and tradeoff analysis techniques for superscalar processor design," *IEEE Transactions on VLSI Systems*, vol. 8, no. 2, pp. 129–137, April 2000.

[9] StatSoft, *http://www.statsoftinc.com/textbook/stathome.html*, StatSoft, Inc., Electronic Testbook.

[10] A. Papoulis, *Probability, Random Variables and Stochastic Processes*, McGraw-Hill, New York, NY, 1984.

[11] V. Tiwari, S. Malik, and A. Wolfe, "Power analysis of the intel 486dx2," Tech. Rep. CE-M94-5, Princeton University, June 1994.

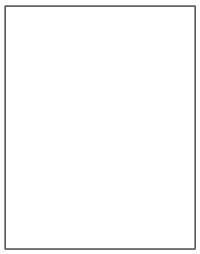
[12] V. Tiwari, M.T.C. Lee, M. Fujita, and D. Maheshwari, "Power analysis of the sparclite mb86934," Tech. Rep. FLA-CAD-94-01, Fujitsu Labs of America, October 1994.

[13] J. Russell, "Instruction Comparison Experiment," *http://www.ece.utexas.edu/~jrussell/power_instr/general_exp2/index.html*

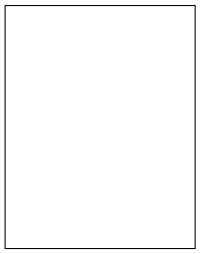
[14] C. Brandolese, W. Fornaciari, F. Salice, and D. Sciuto, "People (power estimation for fast exploration of embedded systems)," Tech. Rep. D3.3.1, ESPRIT-ESD project n.26769, September 1999.

[15] J.L. Hennessy and D.A. Patterson, *Computer Architecture - A Quantitative Approach, II Edition*, Morgan Kaufmann Publishers, San Mateo, CA, 1996.

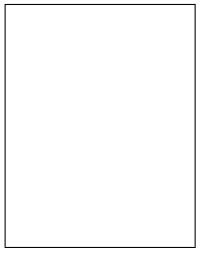
[16] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery, *Numerical Recipes in C, the Art of Scientific Computing*, Cambridge University Press, Cambridge, MA, 1989.



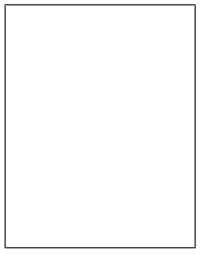
Carlo Brandolese received his laurea degree in Electronic Engineering in 1995 from the Politecnico di Milano, Italy, working on floorplanning of analog integrated circuits. He has been working from 1995 to 1997 at Italtel Central R&D Labs as a CAD engineer and focused on the FPGA design flows and methodologies. He got his MS in Information Technology in 1997 from CEFRIEL, Politecnico di Milano, with a thesis on hardware/software co-design. Since 1997 he is studying the problem of power estimation and optimization of the software components of embedded systems. He received his Ph.D. in Information Technology in 2000 with a thesis on the power modelling of embedded systems. He published 15 papers and received a best paper awards at IEEE-ICCD'98. He is currently post-doc researcher at Politecnico di Milano.



William Fornaciari received his laurea and PhD degrees in Electronic Engineering from the Politecnico di Milano, Italy. Currently he is associate professor at the Politecnico di Milano and also supervises the Embedded Systems Design (ESD) Unit at the CEFRIEL research center in Milano. He published over 90 papers and received the best paper awards during IEEE-IJCNN'92, IEEE-ICONIP'95 and IEEE-ICCD'98, and in 1996 the Certification of Appreciation from the IEEE Circuits and Systems Society. He is IEEE senior member. His main effort is currently in the field of design automation for embedded systems, HW/SW co-design and low-power system-level analysis and design.



Fabio Salice is associate professor at the Departement of Electronics and Information (DEI) of Politecnico di Milano. His research interests include design and synthesis methodologies of self-checking VLSI systems, hardware-software co-design and low-power systems design. He has published more than 70 papers on CAD for VLSI. He received the best paper awards during IEEE-IJCNN'92, IEEE-ICONIP'95 and IEEE-ICCD'98. He received his laurea degree in electronic engineering and his PhD in Electronic and Communication, both from the Politecnico di Milano. He is a member of IEEE and the Computer Society.



Donatella Sciuto received her Laurea in Electronic Engineering in 1984. She received her PhD in Electrical and Computer Engineering in 1988 from University of Colorado, Boulder. She has been an Assistant Professor at the University of Brescia, Dipartimento di Elettronica per l'Automazione until 1992. She is currently a Full Professor at the Dipartimento di Elettronica e Informazione of the Politecnico di Milano, Italy. She is member IEEE, IFIP 10.5, EDAA. She is member of different program committees of EDA conferences: DAC, DATE, CODES/CASHE, ISSS, DFT, FDL, member of the executive committee of ICCAD and DATE, and associate Editor of the IEEE Transactions on Computers, the Journal Design Automation of Embedded Systems, Kluwer Academic Publishers and the Journal of System Architecture, North Holland. She has been Guest editor of a special issue of IEEE Design & Test Magazine in 2000. Her research interests cover mainly the following areas: methodologies for co-design of embedded systems, including system verification, design for low power consumption, HW/SW partitioning and test generation techniques.