# An Instruction-level Functionality-based Energy Estimation Model for 32-bits Microprocessors

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

## ABSTRACT

The paper presents a novel strategy aimed at modeling the instruction energy consumption of 32-bits microprocessors. The proposed *instruction-level* power model is founded on a *functional* decomposition of the activities accomplished by a generic microprocessor and exhibits 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.

## 1. INTRODUCTION

The increasing relevance of power requires to predict with reasonable confidence the power consumption of both hardware and software. The approaches for software 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 power-characterize blocks of the microprocessor architecture [1; 2; 3]. This solution suffers the lack of details on the internal structure of processor cores and, being typically simulation-based, is time-consuming. To overcome these problems, instruction-level measurement-based models have been proposed [4; 5; 6]. The key point lays in measuring the average current drawn by the processor as it executes a long sequence of the same instruction. This procedure has to be repeated for all instructions to completely characterize the processor model. These approaches are processor-dependent by construction and do not exhibit any generalization capability over different architectures. 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 processors and overcomes the above limitations. In fact, it proposes a general methodology, independent of the specific processor, allowing to accurately estimate the energy of an instruction set. The methodology abstracts from the architectural level and focuses on the *functionalities* involved in the instruction execution. The analysis of the statistical properties of the model, confirms two types of generalization:

- Intra-processor: a model built on a suitable subset of instructions allows the extrapolation of the energy characterization of the whole instruction set;
- Inter-processor: a model built on a set of even partially characterized processors allows the extension of the results to new architectures.

To validate the proposed methodology, as reported in section 3, experiments have been performed on five commercial microprocessors. Particular attention has been paid to show that all the assumptions are well founded under a statistical viewpoint. In section 4 conclusions are drawn to summarize the value of the proposed approach and to outline some future research efforts.

## 2. THE MODEL IDENTIFICATION

The proposed model is built on an *a priori* knowledge of both the energy characterization of a set of instructions and the relevant functional characteristics of each instruction of the set. The model, starting from a subset of power-characterized instructions of a given processor, allows derivation of a static—data independent—estimate for the instructions that do not belong to the set on which the model is built. The model accuracy and its generalization capability depend on three factors: the number of instructions, their type (RISC/CISC, arithmetic, branch, etc.) and the model granularity. As the specialization of the model increases, its generalization capability decreases and thus the model is a trade-off between accuracy and generalization.

### 2.1 Model definition

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 $F1$ and $F2$ are *space-disjoint* if the activities accomplished by $F1$ involve different structural units than those of $F2$. Two functionalities $F1$ and $F2$ are *time-disjoint* if $F1$ accomplishes its activities at a different time than $F2$ does.

According to definitions 1 and 2, the activities associated with an instruction can be modeled as the union of some specific functionalities not representing a structural partition, but rather a purely *functional* partition. 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 ($if_j$) and, the relation between functional units 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 $s$ can be expressed as:

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

where $i_s$ is the estimated value for the current consumption of instruction $s$.

*Definition 3.* 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.

Definition 3 indicates that the groups of active functionalities, instruction by instruction, are either time-disjoint, 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 they are not space-disjoint: some of the activities necessary for fetch & decode, in fact, are also performed during execution. 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. 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 an instruction is positive.

*Definition 4.* Let $\mathcal{S}$ be the set of all instructions of a processor, $\mathcal{S}_L \subseteq \mathcal{S}$ 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*. A model is *feasible* if and only if the estimated currents of instructions in $\mathcal{S}$ is nonnegative.

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

*Definition 5.* Let $\mathbf{d}(\mathbf{w})$ be some data depending on a set of parameters $\mathbf{w}$ and $\hat{\mathbf{w}} = f(\mathbf{d}(\mathbf{w}))$ be the estimated value of the parameters. The function $f(\cdot)$ is an estimator, and $\hat{\mathbf{w}}$ are the estimated values, if and only if it is *unbiased* that is $\mathrm{E}[\hat{\mathbf{w}}] = \mathbf{w}$, where $\mathrm{E}[\hat{\mathbf{w}}]$ is the expected value of $\hat{\mathbf{w}}$.

*Definition 6.* A model is *reliable* if and only if it is both *compatible* and *feasible* and the estimator used is *unbiased*.

The 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 equations. The third step consists in computing the estimates of the current associated with each functionality by means of 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:

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

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 of the core, $n_{ck,s}$ is the total number of clock cycles of the instruction $s$ and $\tau$ is the clock period. If $if_j$ is the current drawn by functionality $j$ and $a_{s,j}$ is the contribution of the same functionality in the execution of instruction $s$, equation 2 becomes:

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

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 equality must hold:

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

Taking into account $q < m$ 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:

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

The procedure to determine the value of the parameters $a_{s,F\&D}$ and $a_{s,Exec}$ is detailed in section 3.

## 2.2 Mathematical model

Let $k$ be the number of identified functionalities and let $m_L > k$ be the cardinality of the energy-characterized instruction 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 $mL \times 1$ column vector whose elements are the known terms $i_s \cdot n_{ck,s}$. The linear system:

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

represents the available knowledge on the variables is 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 $\mathbf{IF}$. Minimizing the square error $\|\mathbf{IN} - \widehat{\mathbf{IN}}\|^2$ leads to the equation:

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

To estimate the model parameters $\widehat{\mathbf{IF}}$, the columns of matrix $\mathbf{A}$ must be linearly independent, otherwise two possible strategies can be envisaged. 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 by either increasing or 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.

## 2.3 Single-processor model

Equation 7 gives the set of estimated parameters based on the known relations between current measures and weights $a_{s,j}$. This set of parameters is a reliable model if its estimator is not biased (definitions 5 and 6). 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 yields:

$$\widehat{\mathbf{IF}} = \mathbf{IF} + \mathbf{A}^* \times \mathbf{R} \tag{9}$$

where $\mathbf{A}^* = (\mathbf{A}^T \times \mathbf{A})^-1 \times \mathbf{A}^T$. Equation 9 expresses the relation between estimated and actual parameters. The model is completely characterized from a statistical point of view when the expectation value and the variance of its parameters are given. The expectation value $\mathrm{E}[\widehat{\mathbf{IF}}]$ of $\widehat{\mathbf{IF}}$ and its variance $\mathrm{VAR}[\widehat{\mathbf{IF}}]$ are:

$$\mathrm{E}[\widehat{\mathbf{IF}}] = \mathbf{IF} + \mathbf{A}^* \times \mathrm{E}[\mathbf{R}] \tag{10}$$

$$\mathrm{VAR}[\widehat{\mathbf{IF}}] = \mathrm{E}[(\mathbf{IF} + \mathbf{A}^* \times \mathbf{R} - \mathbf{IF} - \mathbf{A}^* \times \mathrm{E}[\mathbf{R}])^2] \tag{11}$$

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

$$\mathrm{E}[\mathbf{R}] = 0; \quad \mathrm{E}[\mathbf{R} \times \mathbf{R}^T] = \lambda^2 \cdot \mathbf{I} \tag{12}$$

While the first of these relations is straightforward, the second implies that $\mathrm{E}[\mathbf{R} \times \mathbf{R}^T]$ is a diagonal matrix and thus:

$$\mathrm{E}[r_i \cdot r_j] = \begin{cases} 0 & i \neq j \\ \lambda^2 & i = j \end{cases} \tag{13}$$

Under these assumptions the estimator is unbiased, that is $\mathrm{E}[\widehat{\mathbf{IF}}] = \mathrm{E}[\mathbf{IF}]$ and $\mathrm{VAR}[\widehat{\mathbf{IF}}] = \lambda^2(\mathbf{A}^T \times \mathbf{A})^{-1}$. Unfortunately, $\lambda^2$ is unknown since it depends on the residual vector $\mathbf{R}$. For this reason the value of $\lambda^2$ 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 of the variance is $\hat{\lambda}^2 = \|\widehat{\mathbf{R}}\|^2/(m-k)$, 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 $\mu_R$, depending on a statistical model, is in turn a statistical variable and has an expectation value and a variance:

$$\mathrm{E}[\mu_r] = \mu_r; \quad \mathrm{VAR}[\mu_r] = \lambda^4/m \tag{14}$$

To test the null hypothesis $\mu_R = 0$ with a 95% confidence level, according to a $\mathrm{Z}_{0.95}$ test, the inequality $|\mu_R| \leq 1.96\lambda^2/\sqrt{m}$ must be satisfied.

## 2.4 Multi-processor model

Data collected from measures on five microprocessors, shows that instructions of the same type (i.e. same operation and addressing mode) have strongly different absolute current absorption, while the relative currents are of the same order of magnitude. This suggests that by using relative currents and a set $\mathcal{P}$ of $p$ processors for learning, a single general model for all processors can be derived. The relative current is defined as:

$$i_{rel,s} = \frac{i_s}{i_{ref}} = \sum_{j=1}^{k} \frac{if_j}{i_{ref}} \cdot a_{s,j} = \sum_{j=1}^{k} if_{rel,j} \cdot a_{s,j} \tag{15}$$

For the generic $q$-th processor of the set $\mathcal{P}$ characterized by $\mathbf{A}_q$ and $\mathbf{IN}_{rel,q} = \{i_{s,rel} \cdot n_{ck,s}\}$ the following equation holds:

$$\mathbf{IN}_{rel,q} = \mathbf{A}_q \times \mathbf{IF}_{rel,q} + \mathbf{R}_{rel,q} \tag{16}$$

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

$$\widehat{\mathbf{IF}}_{rel,q} = \mathbf{A}_q^* \times \mathbf{IN}_{rel,q} \tag{17}$$

The general model should depend on a single, general, set of parameters $\mathbf{IF}_{rel}$, rather than the processor-specific parameters $\mathbf{IF}_{rel,j}$, and thus the model becomes:

$$\mathbf{IN}_{rel,q} = \mathbf{A}_q \times \mathbf{IF}_{rel} + \mathbf{R}_{rel,q} \tag{18}$$

Combining equation 17 and 18 gives:

$$\widehat{\mathbf{IF}}_{rel,q} = \mathbf{A}_q^* \times \mathbf{IN}_{rel,q} = \mathbf{IF}_{rel} + \mathbf{A}_q^* \times \mathbf{R}_{rel,q} \tag{19}$$

Adding up these relations for all indices $q$ corresponding to the $p$ available processors and dividing both sides by $p$, leads to the following relation:

$$\frac{1}{p}\sum_{q=1}^{p} \widehat{\mathbf{IF}}_{rel,q} = \mathbf{IF}_{rel} + \frac{1}{p}\sum_{q=1}^{p} \mathbf{A}_q^* \times \mathbf{R}_{rel,q} \tag{20}$$

Equation 20 indicates that an unbiased estimator of the parameters of the general model can be the *average* of the estimated parameters of each processor in the set $\mathcal{P}$. The statistical properties of the residuals $\mathbf{R}_{rel,q}$ and the chosen estimator are discussed in [11]. By applying the same method used for the single-processor case, the expression for the variance is:

$$\mathrm{VAR}[\widehat{\mathbf{IF}}_{rel}] = \frac{1}{p^2}\sum_{q=1}^{p} \mathrm{VAR}[\widehat{\mathbf{IF}}_{rel,q}] \tag{21}$$

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.

# 3. EXPERIMENTAL RESULTS

This section collects the experimental results obtained on a set of five power-characterized commercial microprocessors.

## 3.1 Identification of functionalities

A first simple decomposition leads to two disjoint functionalities: *fetch & decode* and *execute*. It is intuitive that the *fetch & decode* needs not to be further differentiated, while the *execute* performs a number of tasks that greatly differs from instruction to instruction and thus a more detailed decomposition is necessary. An accurate analysis, supported by the available measures, revealed that:

- a functionality, denoted as $A\&L$, performs arithmetic and logic operations;
- data transfer operations may or may not access memory and this affects the power consumption. The functionality denoted as $Ld\&St$ performs load, store and stack operations. Register write operations are accounted for by the $WrReg$ functionality, while register reads can be neglected;
- jumps and procedure calls require specific operations and thus their execution is modeled with the $Br$ functionality (branch). This functionality is also involved in interrupt handling and system calls 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. The $A\&L$ functionality is thus used to model these operations;
- string operations can be be modeled using the previously defined functionalities.

A microprocessor can thus be decomposed in the five functionalities: $F\&D$, $A\&L$, $WrReg$, $Ld\&St$ and $Br$. The functionalities stimulated by an instruction not only depend on the operation but also on the addressing modes. Table 1 shows the relation between some 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 $Br$. The completion of

| **Addressing mode** | **Sample** | **Functionalities** |
|---|---|---|
| Register | `R2` | $[WrReg]$ |
| Relative | `10(R2)` | $Ld\&St$ |
| Indexed | `(R2+R3)` | $Ld\&St$ |
| Memory | `(100)` | $Ld\&St$ |
| Auto-increment | `(R2)+` | $Ld\&St$, $A\&L$, $WrReg$ |

**Table 1: Addressing modes and functionalities**

an instruction requires both executing an operation and accessing zero or more operands. According to a decomposition into op-code and addressing mode, the characterization of each instruction is obtained by 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 $\{A\&L\} \cup \{WrReg\} \cup \{Ld\&St, A\&L, WrReg\} = \{Ld\&St, A\&L, WrReg\}$. The five extracted functionalities are a possible partition of the tasks performed by a generic processor and represent a trade-off between the necessary knowledge on the architectures being modeled and the accuracy obtainable. This partition is compliant to definitions 1 and 3 and is an acceptable basis for the mathematical model. Its correctness, from a statistical point of view, extensively discussed in [11], is confirmed by the values of the relative normalized contribution of each parameter (i.e. functionality) to the overall model (see table 2). These results show that no functionalities can be neglected without affecting the accuracy of the model.

| Functionality | $F\&D$ | $Br$ | $WrReg$ | $A\&L$ | $Ld\&St$ |
|---|---|---|---|---|---|
| Contribution | 0.08 | 0.08 | 0.13 | 0.20 | 0.51 |

**Table 2: Relative contribution to the overall model**

## 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 $a_{s,j}$. To clarify the procedure adopted for instruction characterization, consider a decomposition two functionalities $F\&D$ and $Exec$. In this case, equation 5 expresses the model for each instruction. Intuitively, since the power consumption depends on the number of cycles taken to fetch, decode and execute the instruction, a reasonable choice is $a_{s,F\&D} = n_{ck,s,F\&D}$ and $a_{s,Exec} = n_{ck,s,Exec}$, 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. 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 of the execution functionalities should equal the number of clock cycles needed for the complete execution. By indicating with $if_1$ the $F\&D$ current and with $if_2, \ldots, if_k$ the $k-1$ execution functionality currents, these relations must hold:

$$a_{s,1} = n_{ck,s,F\&D}; \quad \sum_{j=2}^{k} a_{s,j} = n_{ck,s,Exec} \tag{22}$$

Based on the analysis presented in section 3.1, it is possible to determine whether or not a functionality is involved in the execution of a given instruction. This information can be represented by means of an *activation coefficient* $b_{s,j} = \{0, 1\}$, such that $b_{s,j} = 1$ indicates that the $j$-th functionality takes part in the execution of instruction $s$. The $b_{s,j}$ and the model coefficients $a_{s,j}$ are related by the equation:

$$a_{s,j} = \begin{cases} b_{s,j}\, n_{ck,s,F\&D} & j = 1 \\ b_{s,j}\, w_s & j = 1 \ldots k \end{cases} \tag{23}$$

where the weight $w_s$ is given by:

$$w_s = \begin{cases} 0 & \sum_{j=2}^{k} b_{s,j} = 0 \\ n_{ck,s,Exec} \Big/ \sum_{j=2}^{k} b_{s,j} & \text{otherwise} \end{cases} \tag{24}$$

## 3.3 Estimation on a single processor

The processor used here to show the validity of the approach is the Intel i80486DX. Tables 3 and 4 show the average currents drawn per clock cycle for a subset of instructions along with the number of clock cycles, the total current and the activation coefficients. The coefficients $a_{s,j}$ are obtained using equations 23 and 24. Let $\mathbf{A} = \{a_{s,j}\}$ be the instruction characterization matrix and $\mathbf{IN} = \{i_s n_{ck,s}\}$ the column vector of the total currents. The solution **IF** is computed

| Instruction | $i_s$ | $n_{ck,F\&D}$ | $n_{ck,Exec}$ | $i_s n_{ck}$ |
|---|---|---|---|---|
| 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 |
| 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 3: Currents and clock cycles of 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 |
| MOV [BX],DX | 1 | 0 | 0 | 0 | 1 |
| NOP | 1 | 0 | 0 | 0 | 0 |
| SAL BX,CL | 1 | 0 | 1 | 1 | 0 |

**Table 4: Activation coefficients of i80486DX**

by solving the linear problem in the least square sense. The results, relative to the available 18 energy characterized [7] instructions of the sample microprocessor, are shown in figure 1. The value of the functionalities currents are reported

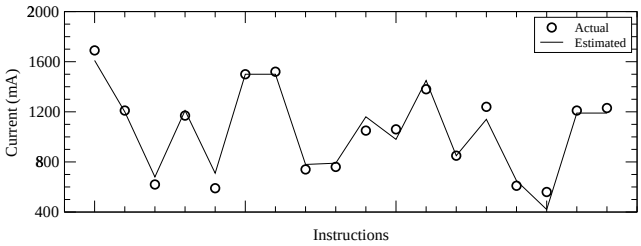


**Figure 1: Intel i80486DX power estimates**

in 5. 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, therefore, the gaussian noise assumption holds (the mean estimated error $\mu_R = 9.94 \times 10^{-11}$ falls in the range $Z_{0.95} = \pm 40.75$). Similar results, reported in table 6, have been obtained for all the other processors analyzed.

## 3.4 Generalization on a single processor

To investigate the generalization capabilities of the model, the following procedure has been adopted: *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 and estimate the currents of instructions in the generalization-set; *iii*) measure the learning and generalization errors. The procedure has been repeated varying the learning-set cardinality. For each learning-set size, 100 different, randomly selected, learning-sets have been used. The mean learning and generalization errors are shown in the graph of figure 2 for the Intel i80486DX processor. Similar results have been obtained for the other processors. Despite the random choice of the 100 subsets used in the analysis, the accordance between actual currents and estimated currents is good. This procedure has been repeated for 500 times for each processor

| Functionality | $F\&D$ | $Br$ | $WrReg$ | $A\&L$ | $Ld\&St$ |
|---|---|---|---|---|---|
| Current(mA) | 421 | 355 | 228 | 228 | 505 |

**Table 5: Functionality currents of i480486DX**

| Processor | | $if_1$ | $if_2$ | $if_3$ | $if_4$ | $if_5$ |
|---|---|---|---|---|---|---|
| ARM7 | mA | 5.7 | 14.3 | 13.0 | 18.3 | 13.9 |
| | STD | 1.1 | 0.7 | 0.5 | 0.6 | 0.7 |
| i960JF | mA | 362.0 | 261.9 | 302.2 | 320.0 | 380.6 |
| | STD | 8.5 | 13.1 | 8.0 | 8.1 | 8.9 |
| i960HD | mA | 970.4 | 692.8 | 804.8 | 775.4 | 1026.3 |
| | STD | 17.9 | 28.4 | 18.5 | 18.6 | 46.5 |
| SPARC | mA | 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**

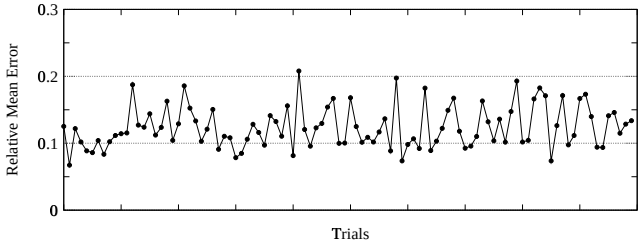


**Figure 2: Generalization error for i80486DX**

leading to the results summarized in figure 3. Note that, since the errors tend to compensate, their mean value is very close to zero ($\approx 10^{-10}$): for this reason 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 3 shows

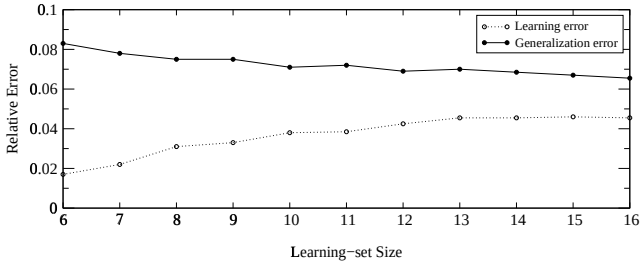


**Figure 3: Learning and generalization relative error**

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.

## 3.5 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. 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:

$$\widehat{\mathbf{IF}}_{rel,h} = \frac{1}{p_h}\sum_{q=1}^{p_h}\widehat{\mathbf{IF}}_{rel,q} \qquad (25)$$

The parameters $\widehat{\mathbf{IF}}_{rel,h}$ have been computed for all $2^5 - 1$ possible processors-learning-sets and have been used to estimate the current consumption on all processors in the *processor-generalization-set* $\mathcal{P}_G = \mathcal{P} - \mathcal{P}_L$. Figure 4 reports

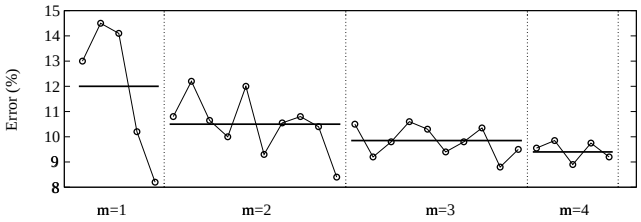


**Figure 4: General model mean error**

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 $p_h$ ranging from 1 to 5. Each point in the graph represents the mean of average errors calculated over the processors in $\mathcal{P}_G$. The first point of the series 1 corresponds to the set $\mathcal{P}_{L,1}$ = { SPARClite MB86934 }, the second point corresponds to $\mathcal{P}_{L,2}$ = { Intel i80486DX }, etc. The first point in the second series corresponds to the set $\mathcal{P}_{L,6}$ = { SPARClite MB86934, Intel 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 learning set increases. As an example, consider a general model built using the parameters of the Intel i80486DX and ARM7TDMI architectures to estimate the power consumption of the SPARClite MB86934. The accordance between actual data and estimated data, shown in figure 5, is satisfactory.

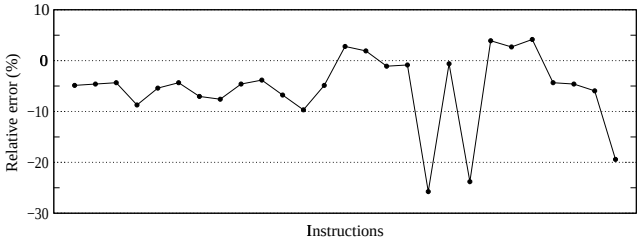


**Figure 5: Generalization error of SPARClite**

# 4. CONCLUSIONS

An approach to model the instruction energy consumption of 32-bit microprocessors has been proposed. Differently from the strategies in literature, this method estimates the current for each instruction by means of a linear combination of values associated with a set of disjoint functionalities. As confirmed by the experiments performed, the proposed model, based on five functionalities, shows notable accuracy and good generalization properties, both intra-processor and inter-processor, and allows extrapolating the power consumption of uncharacterized instructions. The following table shows the values and the estimated standard deviations of the five functionalities of the general model obtained on the five available processors [6; 7; 8; 9]. 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 measurement errors and the modeling inaccuracy mask the contribution related to the state of the processor. Other effects, such as those connected with pipes and caches, 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. The proposed model can thus be considered a basis for a general power estimation framework considering also higher-level aspects of code execution; its main value lays in the possibility to easily model and compare different processors with limited measurement effort.

| 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 |

**Table 7: Relative functionality currents**

# 5. REFERENCES


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

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

[3] P.Landam and J.Rabaey, *"Black-box capacitance models for architectural power analysis,"* Proc. of Int. 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, No. 4, pp. 437-445, Dec. 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 ICCD'98, International Conference on Computer Design, Austin, Texas, USA, October, 1998.

[7] V. Tiwari, S. Malik, and A. Wolfe, *"Power Analysis of the Intel 486DX2,"* Computer Engineering Technical Report No. CE-M94-5, Princeton University, Jun. 1994.

[8] V. Tiwari, M.T.C. Lee M.Fujita and D. Maheshwari, *"Power Analysis of the SPARClite MB86934,"* Technical Report FLA-CAD-94-01, Fujitsu Labs of America, Oct. 1994.

[9] J. Russell, *"Power Consumption and Execution Time for the Various Instructions: JF and HD processors results,"* http://www.ece.utexas.edu/~jrussell/power_instr

[10] PEOPLE (Power Estimation for Fast Exploration of Embedded Systems) ESPRIT-ESD project n.26769, Technical Report D3.3.1.

[11] C. Brandolese, W. Fornaciari, F. Salice, D. Sciuto *"Fast Software-Level Power Estimation for Design Space Exploration,"* Politecnico di Milano, Tech. Report 99.62, 1999.