

An Instruction-Level Energy Model for Embedded VLIW Architectures

Mariagiovanna Sami, *Member, IEEE*, Donatella Sciuto, *Member, IEEE*, Cristina Silvano, *Member, IEEE*, and Vittorio Zaccaria, *Student Member, IEEE*

Abstract—In this paper, an instruction-level energy model is proposed for the data-path of very long instruction word (VLIW) pipelined processors that can be used to provide accurate power consumption information during either an instruction-level simulation or power-oriented scheduling at compile time. The analytical model takes into account several software-level parameters (such as instruction ordering, pipeline stall probability, and instruction cache miss probability) as well as microarchitectural-level ones (such as pipeline stage power consumption per instruction) providing an efficient pipeline-aware instruction-level power estimation, whose accuracy is very close to those given by RT or gate-level simulations. The problem of instruction-level power characterization of a K -issue VLIW processor is $O(N^{2K})$ where N is the number of operations in the ISA and K is the number of parallel instructions composing the very long instruction. One of the advantages of the proposed model consists of reducing the complexity of the characterization problem to $O(K \times N^2)$. The proposed model has been used to characterize a four-issue VLIW core with a six-stage pipeline, and its accuracy and efficiency has been compared with respect to energy estimates derived by gate-level simulation. Experimental results (carried out on a set of embedded DSP benchmarks) have demonstrated an average error in accuracy of 4.8% of the instruction-level estimation engine with respect to the gate-level engine. The average simulation speed-up of the instruction-level power estimation engine with respect to the gate-level engine is of four orders of magnitude approximately.

I. INTRODUCTION

LOW POWER is an increasingly relevant requirement for many classes of embedded systems [1]. A reasonably efficient approach to power estimation at the highest levels of abstraction is of fundamental importance for successful system-level design [2], [3]. Starting from the pioneer work of Landman and Rabaey [4] on the high-level power modeling of data path modules, many research works appeared in literature to face the problem of increasing the level of abstraction of the power estimation. Particular attention has been devoted to the problem of power estimation for high-performance microprocessors, where pipelining and instruction-level parallelism have to be accounted for in the early design phases. However, many issues are still open in this area, and an efficient and accurate framework for system-level power estimation has not been developed yet.

Manuscript received April 3, 2001; revised November 15, 2001. This work was supported in part by CNR (Project MADESS II) and by ST Microelectronics. This paper was recommended by Associate Editor R. Gupta.

M. Sami, D. Sciuto, and V. Zaccaria are with the Dipartimento di Elettronica e Informazione, Politecnico di Milano, Italy (e-mail: sami@elet.polimi.it; sciuto@elet.polimi.it; zaccaria@elet.polimi.it).

C. Silvano is with the Dipartimento di Scienze dell'Informazione, Università degli Studi di Milano, Italy (e-mail: silvano@dsi.unimi.it).

Publisher Item Identifier 10.1109/TCAD.2002.801105.

The overall goal of our work is to define a power-oriented exploration methodology at the system-level suitable for very long instruction word (VLIW) embedded cores. A VLIW processor is a (generally) pipelined processor that can execute, in each clock cycle, a set of explicitly parallel *operations*; this set of operations is statically scheduled to form a VLIW. The approach that we propose in this paper is an extension of the work we previously proposed in [5] and [6] targeting an instruction-level power model to evaluate the software energy consumption associated with a pipelined VLIW core. The model is quite general and parametric for the exploration of the power budget by considering different compiler-level and architectural-level solutions for the VLIW core, given a specific application.

A. Background

In this section, we discuss some relevant research results that recently appeared in literature related to software-level power estimation methods.

A software-level power estimation tool is an application that receives as inputs an HDL or functional representation of a processor and a binary program to be executed. The tool produces as output an estimate of the power consumed by the processor during the execution of the program. The power values can be represented as an average overall power consumption or a trace of instantaneous power consumption values (*run time power profiling*), depending on the capabilities of the tool.

Power estimation tools for processor cores can be classified as based on gate-level or RT-level simulation [7]–[9], microarchitectural-level simulation [10]–[14], and instruction-level simulation.

Instruction-level power estimation tools are based on a functional simulation of the instruction set of the processor. During the instruction set simulation (ISS), an instruction-level model associates a black-box cost with each instruction by considering also circuit state effects, pipeline stalls, and cache misses produced during its execution. Research on instruction-level power analysis (*ILPA*, for brevity) has started only recently; thus, only a few proposals have been made on this subject and it lacks a rigorous approach.

The main contributions found in literature are empirical approaches based on physical measurements of the current drawn by the processor during the execution of embedded software routines. Tiwari *et al.* [15], [16] proposed to estimate the energy consumption of a processor by an instruction-level model that assigns a given power cost (namely the *base energy cost*) to each single instruction of the instruction set. However, during the execution of a software routine, certain *interinstruction effects*

occur, whose power cost is not taken into account if only the base energy costs are considered. The first type of interinstruction effects (namely the *circuit state overhead*) is associated with the fact that the cost of a pair of instructions is always greater than the base cost of each instruction in the pair. The remaining interinstruction effects are related to resource constraints that can lead to stalls (such as pipeline stalls and write buffer stalls) and cache misses, which imply a penalty in terms of energy.

Globally, the model in [15] and [16] expresses the average energy as

$$E = \sum_i (B_i \times N_i) + \sum_{i,j} (O_{i,j} \times N_{i,j}) + \sum_k E_k \quad (1)$$

where E is the total energy dissipation associated with the execution of a given program, which is computed as the sum of three components. The first component is the sum of the energy base cost B_i of each instruction i multiplied by the number of times N_i that the instruction is executed. The second component is the sum of the additional energy cost $O_{i,j}$ due to the elementary instruction sequence (i, j) multiplied by the number $N_{i,j}$ of executions of each sequence; such a contribution is evaluated for all instruction pairs. The third contribution E_k takes into account any additional energy penalties due to resource constraints that can lead to pipeline stalls or cache misses.

The basic idea of this approach is to measure B_i as the current drawn by the processor as it repeatedly executes a loop whose body contains a sequence of identical instructions i . The term $O_{i,j}$ takes into account the fact that the measured cost of a pair of instructions has been always observed to be greater than the sum of the base costs of each single instruction. These terms must be stored in a matrix whose size grows with the square of the number of instructions in the ISA. This turns out to be impractical for CISC processors with several hundreds of instructions, and the storage problem becomes exponentially complex when considering VLIW architectures, in which long instructions are composed of arbitrary combinations of parallel operations. A possible solution to this problem is presented in [16] where the authors propose to reduce the spatial complexity of the problem by clustering instructions based on their cost. The results of the analysis motivated several software-level power optimization techniques, such as instruction reordering to exploit the power characteristics of each instruction.

The instruction-level power model proposed in [17] considers an average instruction energy equal for all instructions in the ISA. More specifically, this model is based on the observation that, for a certain class of processors, the *energy per instruction* is characterized by a very small variance. Note that, while in complex architectures like CISCs, the heavy use of caches and microcode ROMs produces a relevant increase of the average instruction energy and a reduction of the energy variance, this is not always true for RISC and VLIW architectures for which these effects can become particularly evident.

Another attempt to perform instruction-level power estimation is proposed in [18] where the authors consider several possible interinstruction effects as well as statistics of data values used within the processor. Although the developed power model is quite accurate, it lacks general applicability, being developed only for a specific embedded processor.

In [19], interinstruction effects have been measured by considering only the additional energy consumption observed when a generic instruction is executed after an NOP (the proposed power model is also called the *NOP model*). As a matter of fact, the model reduces the spatial complexity proper of instruction-level power models.

More recently, a function-level power estimation methodology has been developed in [29] to estimate the average power of an embedded software. The method is based on a “power data bank” that stores the power information of built-in library functions and basic instructions for a given microprocessor. The power estimation tool does not require power simulation once the “power data bank” is built, so this method outperforms other simulation-based approaches in terms of efficiency. However, this higher efficiency must be paid for in terms of lack of accuracy. If we are only interested in terms of the average power consumption and execution time for a given embedded program, the level of accuracy is acceptable (within 3%).

B. Main Contributions of This Work

The main limitation of the instruction-level power analysis is that it does not provide any insight on the instantaneous causes of power consumption within the processor core, which is seen as a black-box model. Similarly, these methods do not analyze the power contributions due to the single operations that are active in each pipeline stage. These two factors can affect the capability to perform a realistic run-time power profiling of a software application since overlapping of instructions within the pipeline is neglected. Moreover, this can exclude some specific compiler-level optimization techniques, such as creating instruction schedules whose power consumption is always under a given upper bound or with reduced variation rates (to optimize the battery usage).

Concerning VLIW processors, traditional instruction-level power modeling poses new challenges to be afforded, since the number of long instructions that can be created by combining N operations into a K -wide VLIW is N^K . It becomes evident that traditional instruction-level methods, that need to store the energy parameters for each pair of instructions, are not applicable to VLIW machines.

To address these limitations, our main focus is the definition of an energy model for VLIW architectures that combines the efficiency and flexibility of instruction-level models with the accuracy of microarchitectural level models. The main goals are: 1) to provide accurate information on the energy consumed by the processor core during an instruction-level simulation of a VLIW embedded application and 2) to provide to the compiler a fine grained power model of the instructions overlapped in the pipeline. The long term goal of our analysis is to give support to higher level compilation-level and software-level power optimization techniques.

Our analysis mainly targets the data path of VLIW cores. The energy model is based on an analytical decomposition of the energy consumed by a software instruction into the contributions of few microarchitectural components (the active processor function units and the pipeline stages). These microarchitectural contributions are associated with the multiple parallel operations flowing through the pipeline stages combined with

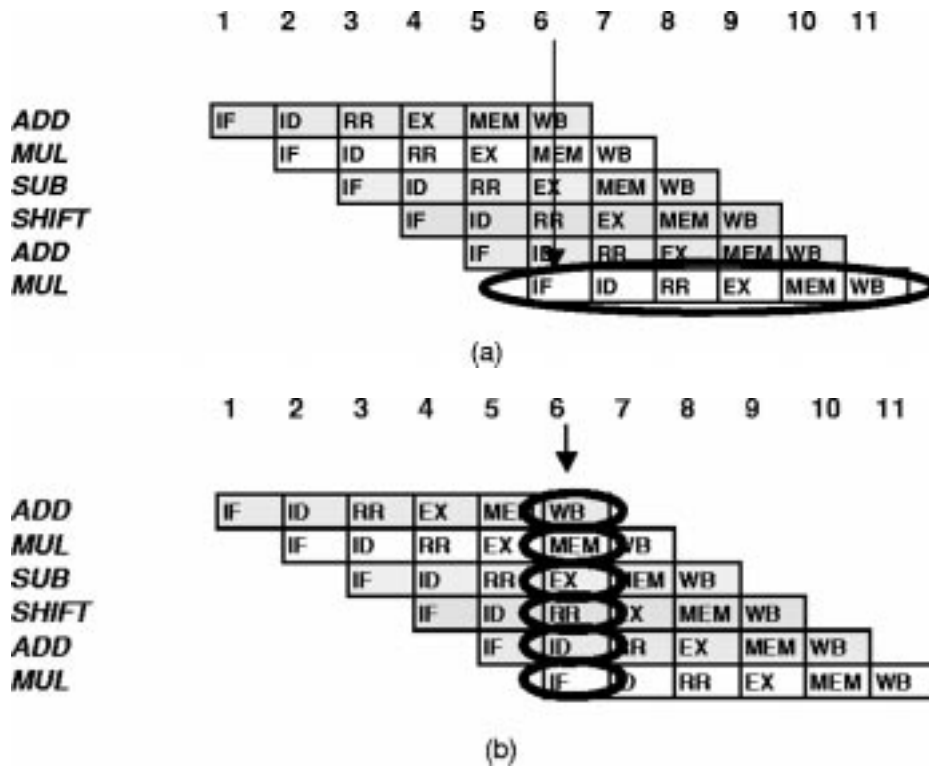


Fig. 1. (a) Traditional and (b) pipeline-aware instruction-level energy models.

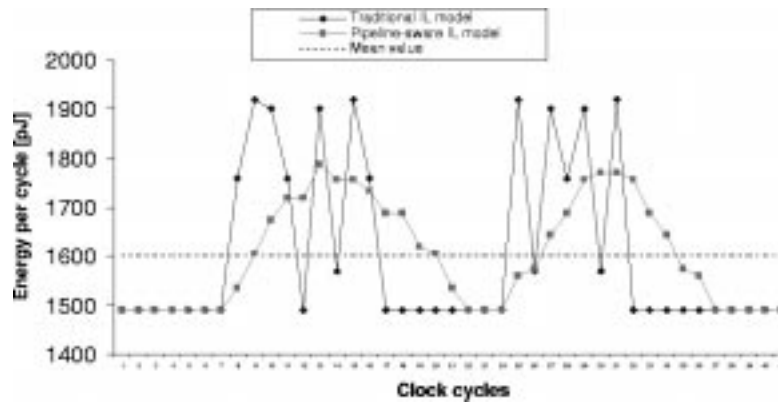


Fig. 2. Comparison between traditional and pipeline-aware instruction-level energy models.

other instruction-level contributions (such as the probability and the latency per instruction of a miss in the memory hierarchy). Run-time accuracy cannot be achieved by using a traditional black-box instruction-level model since, such types of models neglect the overlapping of instructions within the pipeline.

Fig. 1(a) shows how a traditional energy model can be used to compute (e.g., within an ISS) the energy for a given clock cycle; the traditional model associates with each clock cycle, the instruction that is being currently fetched from the cache, without considering (totally or mostly) the effective presence in the pipeline of other instructions and by modeling cache misses as a fixed power cost to be added to the contribution of the current instruction. Fig. 1(b) shows our pipeline-aware model in which the power consumption is modeled on a *per-cycle basis* by splitting the energy of each instruction in the energy contributions of the corresponding stages and by summing up all the

different contributions at any given clock cycle. In particular, in the given example, the pipeline-aware power model considers a pipeline composed of six stages with a uniform decomposition of the energy of each instruction over the single pipeline stages. Of course, such types of models can be used within an ISS that is aware of the pipeline structure of the processor (i.e., cycle-based ISS). The value of the *average* energy per cycle for a given sequence of instructions is identical for both types of models as shown in Fig. 1.

Anyway, the pipeline-aware power model is closer to the real pipeline behavior, since it effectively considers the instructions overlapping within the pipeline. Fig. 2 shows a comparison of the results derived from the application of a traditional instruction-level power model and the pipeline-aware power model to the same sequence of instructions. This short sequence, extracted from a real application, is characterized by two bursts of

instructions separated by a sequence of NOPs inserted by the compiler (energy values are in the range from 1490 to 1919 pJ). As shown in Fig. 2, even if the average energy per cycle is equal for both models, the energy behavior is quite different. The “traditional” instruction-level power model considers in each clock cycle the energy contribution associated with the instruction in the fetch stage. The “pipeline-aware” model has a more precise view on the different instructions present in the pipeline at any given clock cycle. This becomes particularly evident during the phases of filling up (flushing) of the pipeline after (before) a sequence of NOP instructions. Moreover, the pipeline-aware model reproduces a “smoother” energy behavior with respect to the traditional model. The traditional model has some peaks, not actually found in the real-world energy behavior, outlining the differences in terms of energy cost for each instruction.

The pipeline-aware model can add value either to an instruction-level simulation to realistically model the energy behavior of a processor or to a power-aware compiler that can modify power figures (i.e., spikes, upper bounds, and rates of variation) by means of instruction scheduling. An application example of such a type of compilation can be the rescheduling for the minimization of variation rates in the energy use to extend the battery life. Another application could be a more power-conscious instruction scrambling algorithm that tries to make uniform the energy consumption for a given cryptography application, to avoid security attacks based on the observation of the power behavior.

To enter into more detail regarding our energy model, we consider that the execution of an instruction can be projected on two axes: a *temporal* one and a *spatial* one.

The *temporal projection* traces the progress of a very long instruction through the pipeline stages. Our model exploits the temporal projection to compute the energy of an instruction as the sum of the energy contributions due to each pipeline stage involved in its execution. In this case, we say that the energy is *temporal additive*.

The *spatial projection* traces the use of the resources of the processors within a single pipeline stage. Our model exploits the spatial projection to decompose the energy of a pipeline stage into the sum of the energy contributions due to each operation involved in the execution of the current and the previous long instruction. This can be done by assuming that mutual influence between two different operations of the same very long instruction can be considered as an irrelevant effect with respect to the mutual influence between operations of two successive very long instructions. This characteristic of the model, introduced as the *spatial additive property* of the energy of a VLIW processor, has been experimentally verified and it is fundamental to deal with the complexity of the characterization of the instruction-level power model for VLIW cores.

By using both temporal and spatial additive properties, the problem complexity has been reduced from $O(N^{2K})$ to $O(K \times N^2)$, where K is the number of parallel operations in the very long instruction and N is the number of operations in the ISA. Thus, a more accurate estimation than a black-box model is achieved with a reasonable computational complexity.

The main difference of our approach with respect to ILPA-based approaches is that our method gives better insight on the

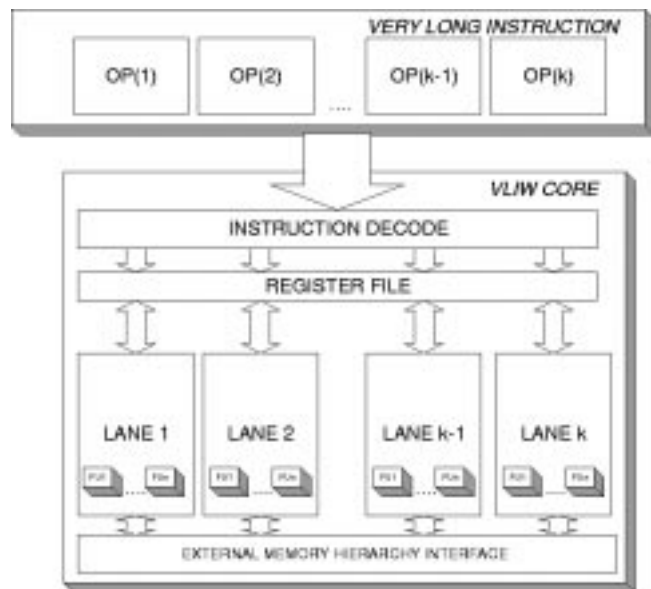


Fig. 3. Architecture of the target VLIW pipelined core.

power bottlenecks during software execution because it is based on a detailed microarchitectural model of the processor core and it analyzes the energy behavior on a per-cycle basis. In particular, the proposed analytical model defines a mapping between the long instructions and microarchitectural functional units involved during the instruction execution. This instruction-to-unit mapping is used to retrieve RT-level energy estimates for each unit that is combined with stall and latency energy contributions to obtain instruction-level power estimates.

The overall goal of our work is to offer an accurate power estimation tool for the end user to evaluate the power consumption associated with a software code for an embedded system. Furthermore, we can envision the use of the proposed tool for system-level power exploration during the compilation phase (through a power-oriented scheduler) as well as during the microarchitectural design optimization phase of the processor.

The paper is organized as follows. The overall architecture of the target system is described in Section II. Section III introduces the general instruction-level energy estimation methodology for pipelined VLIW architectures. Experimental results derived from the application of our method to a four-issue VLIW architecture are reported in Section IV. Finally, Section V concludes the paper by outlining some research directions originated from this work.

II. TARGET SYSTEM ARCHITECTURE

In this section, we introduce the processor architecture, the instruction set architecture, and the processor requirements for the target system.

A. Target Processor Architecture

The target processor architecture is a general load/store VLIW core processor (see Fig. 3). A VLIW processor is a pipelined CPU that can execute, in each clock cycle, a set of explicitly parallel *operations*. This set of operations is called a VLIW. In the rest of the paper, we abbreviate the term very long

instruction as *instruction*, while maintaining the use of the term *operation* to identify the single components of an instruction. The position of an operation within an instruction is called *slot*. A slot usually identifies the execution path (also called *lane*) of the operation within the VLIW processor. A K -issue VLIW processor can fetch an instruction composed of one to K parallel operations and it executes them on K parallel lanes (see Fig. 3). In a pipelined VLIW, each lane executes a single operation in a pipelined fashion. Generally, the decode stage decodes an instruction in several control words associated with each operation and these words are then used to control the pipeline stages of each lane. Ideally, even the decode stage can be decomposed into lanes that decode each single operation of the long instruction.

Furthermore, the target VLIW architecture does not support predication, since this mechanism could waste power while executing instructions that will not be committed back to the registers.

We assume also that the target system architecture includes separate instruction and data caches.

When a miss in the instruction cache occurs, we assume the instruction cache propagates to the core an “IC-MISS” signal, that is used, within the core, to “freeze” the inputs of the major part of the pipeline registers, avoiding useless switching activity. The macrobehavior is similar to the execution of NOP instructions inserted in the code at compile time. The only difference between the two cases is that explicit NOP instructions generate additional switching activity passing through the pipeline registers.

When a miss in the data cache occurs, we assume that the core gates the clock to all the pipeline registers until the miss is served.

B. Target ISA

We define a very long instruction (instruction for brevity) \mathbf{w} of a K -issue VLIW as a vector of K elements

$$\mathbf{w} = \begin{bmatrix} w^1 \\ \dots \\ w^k \\ \dots \\ w^K \end{bmatrix} \quad (2)$$

where $w^k \in ISA$ is the operation associated with slot k of the instruction \mathbf{w} . When power is concerned, it has been observed in [16] that the operations most commonly used in a target application can be categorized into classes, such that the operations in a given class are characterized by very similar power costs. Operations with similar functionality activate similar parts of the processor core, thus they have similar power behavior. The spatial additive property (discussed in detail in Section III) implies that the different lanes will be considered mutually independent with respect to power consumption, so that power requirements of single operations can be evaluated without considering the relationships with other operations in the same instruction.

In this work, we consider two operations in the ISA as “not equal” if they differ either in terms of functionality (i.e., opcode) or in terms of addressing mode (immediate, register, indirect, etc.). Even without considering data differences (either

in terms of register names or immediate values), the number of operation’s pair to be considered could become too large to be characterized with a transistor-level or even gate-level simulation engine. For example, in the case of an ISA composed of about 70 operations, considering also the possible addressing modes for each operation, we would need to characterize 6126 pairs of operations to completely define the interinstruction effect.

To sensibly reduce the number of experiments to be generated during the characterization phase, we can apply the cluster analysis we proposed in [33] to the operations of the ISA for VLIW architectures. The basic idea of the cluster analysis consists of grouping in the same cluster the operations showing similar power costs. The power cost of an operation is defined as the power consumed by the processor when it executes only that operation.

In the rest of the presentation of our model, we assume that the ISA of the target VLIW processor is clustered into classes of operations that have similar power behavior.

C. Target Processor Requirements

Our approach requires a gate-level or (at most) RT-level description of the target processor, so that accurate energy values can be derived for the characterization of the model here proposed. Moreover, in order to record the information needed at run-time for the proposed energy model, it is necessary to make use of an ISS suitably instrumented.

Besides, in our design, we assume that two identical modules M_i and M_j in the *logic* architecture belonging either to different stages of the pipeline or to different lanes, map into distinct modules in the *physical* architecture as well. Such synthesis is particularly useful for our experiments, since power contributions are easily associated with physical modules during program execution. Still, this does not detract from the generality of our approach; we assume in fact that M_i and M_j map into the same physical module M_p (as it may occur in commercial architectures, to minimize area). Obviously, at any given clock cycle, M_p supports the execution of one only of the two logical modules (otherwise, no correct execution would ensue); therefore, its power contribution will be correctly associated with the operation or the stage involving the active logical module.

III. VLIW ENERGY MODEL

This paper represents a main extension of the work we proposed in [5] and [6] that addresses instruction-level energy estimation for the data-path of VLIW embedded cores based on an analytical energy model that considers the processor microarchitecture. In particular, the proposed model defines the energy of a very long instruction as the sum of a set of energy contributions due to the individual pipeline stages involved in the instruction execution and to the individual operations composing the instruction.

In a VLIW processor, instructions are composed of one or more explicitly parallel operations. After an instruction is fetched from the I-cache, each operation is dispatched to a specific functional unit (see Fig. 3). Unlike a superscalar processor, a VLIW processor does not check for any data

or control dependencies, since the compiler guarantees that the flow of operations does not present any type of *intra-* or *interbundle* dependency.

As already noted in [19], the challenge of an instruction-level power model for microprocessors derives from the complexity due to the spatial and temporal correlation of the instructions in the execution trace. As noted before, for each interinstruction effect we must use an n -dimensional array where n depends on the type of interinstruction effects that we are considering. For example, if we need to take into account the interinstruction effect between adjacent instructions only, it is necessary to store this information in a bidimensional array whose size is proportional to the number of instructions in the ISA. In the case of a RISC or CISC machine, the problem complexity can be reduced by clustering instructions [16], thus reducing the parameter's array length.

In contrast, for K -issue VLIW architectures, the power model should account for all possible combinations of operations in an instruction, thus the problem complexity grows exponentially with K and the number of operations in the ISA. As for the RISC case, instruction clustering can be effective to reduce the problem complexity by trading power estimation accuracy and problem complexity, but the problem complexity can be further simplified as shown in the rest of the paper.

Let us consider a stream \mathcal{W} composed of N very long instructions

$$\mathcal{W} = \langle \mathbf{w}_1, \dots, \mathbf{w}_{n-1}, \mathbf{w}_n, \dots, \mathbf{w}_N \rangle \quad (3)$$

where \mathbf{w}_n is the n th instruction [see (2)] of the stream.

In this work, we assume the energy associated with \mathbf{w}_n to be dependent on the properties of \mathbf{w}_n (e.g., class of the instruction, data values involved in its evaluation and so on) as well as on its *execution context*, i.e., the set of instructions contained in \mathcal{W} near to \mathbf{w}_n . The *execution context* of \mathbf{w}_n can be split in two contributions. The *first-order* contribution is due to the previous instruction \mathbf{w}_{n-1} , while the *second-order* contribution is given by the other instructions affecting the pipeline during the execution time of \mathbf{w}_n . In our model, we directly consider only the first-order contribution of the execution context, while the second order contribution is indirectly accounted for mainly in terms of additional stall/latency cycles introduced during the execution of instruction \mathbf{w}_n . Besides, as usual, we consider the design of the core as partitioned in a control unit and a data-path unit.

The estimation of the energy consumed by the processor during the execution of the stream \mathcal{W} is

$$E(\mathcal{W}) = \sum_{\forall n \in N} E(\mathbf{w}_n | \mathbf{w}_{n-1}) + E_c + c_0 \quad (4)$$

where the term $E(\mathbf{w}_n | \mathbf{w}_{n-1})$ is the energy dissipated by the data path associated with the execution of instruction \mathbf{w}_n dependent on the properties of \mathbf{w}_n , \mathbf{w}_{n-1} as well as their *execution context*, E_c is the total energy dissipated by the control unit, and the term c_0 is an energy constant associated with the startup sequence of the processor (in which the sequential elements of the processor are reset to initialize the state of the processor).

In this work, we assume that the control is explicitly designed outside the pipeline and it is relatively invariant in power

consumption with respect to the type of instructions executed. Since, for control units, accurate power estimates can be derived by using power models developed for finite state machines (such as those presented in [23]), we target our research on the characterization at the instruction level of the power consumption of the data path.

We assume that the pipeline stages enjoy the *temporal additive property*, i.e., the energy consumption associated with an instruction flowing through the pipeline stages corresponds to the sum of the energy contributions of each pipeline stage. The property assumes that the processor modules associated with each pipeline stage are independent to each other.

Therefore, given the processor pipeline composed of a set S of stages, the energy consumption associated with \mathbf{w}_n actually corresponds to the sum of the energy consumption of the modules that elaborate \mathbf{w}_n in each one of the pipeline stages

$$E(\mathbf{w}_n | \mathbf{w}_{n-1}) \approx \sum_{\forall s \in S} A_s(\mathbf{w}_n | \mathbf{w}_{n-1}) + I(\mathbf{w}_n | \mathbf{w}_{n-1}) \quad (5)$$

where

- 1) the term $A_s(\mathbf{w}_n | \mathbf{w}_{n-1})$ is the average energy consumed *per stage* s when executing instruction \mathbf{w}_n after instruction \mathbf{w}_{n-1} ;
- 2) the term $I(\mathbf{w}_n | \mathbf{w}_{n-1})$ is the energy consumed by the connections between pipeline stages (interstage connections).

Note that modules used in more than one pipeline stage, such as the register file, can be considered only once per instruction with an average value that must be characterized separately.

Let us detail the two terms A_s and I appearing in (5). The execution of two consecutive instructions can be influenced by two type of events. The first type of event is a data cache miss occurring during the execution of an instruction. In this case the processor stalls all the instructions in the pipeline until the write or read data cache miss is resolved. Fig. 4 shows an example of a data cache miss occurring during the execution of instruction E and another one during instruction G (the source of the stall occurs in the MEM stage). The other possible event is an instruction cache miss occurring during the fetch of an instruction: when this event occurs, the instruction cache generates to the core an "IC-MISS" signal, that is used, within the core, to freeze the inputs of the majority of the pipeline registers, avoiding useless switching activity. Fig. 4 shows an example of an instruction cache miss occurring during the fetch of instruction F and another one during the fetch of instruction G . In both cases, the penalty for such misses consists of 1 cycle. The macrobehavior of the pipeline is similar to the execution of a sequence of NOP operations, with the difference that, in case of NOPs, we have some energy dissipated due to the flowing of the NOP within the pipeline registers. In the proposed model, in fact, the presence of a NOP instruction generated at compile-time is treated as a normal instruction and it is characterized similarly to the other instructions in the ISA.

In our model, the average energy consumption *per stage* can be expressed linearly as

$$A_s(\mathbf{w}_n | \mathbf{w}_{n-1}) = U_s(\mathbf{w}_n | \mathbf{w}_{n-1}) + \sigma_s^n + \mu_s^n \quad (6)$$

the overall cost associated with \mathbf{w}_n for each stage would be

$$U_s(\mathbf{w}_n|\mathbf{w}_{n-1}) = U_s(\mathbf{0}|\mathbf{0}) + \nu_s(\text{ALU}|\text{LS}) + \nu_s(\text{NOP}|\text{ALU}) \quad (10)$$

given, by definition that $\nu_s(\text{NOP}|\text{NOP}) = 0$.

In general, for an ISA composed of C operation classes, each ν_s is a lookup table composed of $(C^2+C)/2$ elements. Note that in the case of standard instruction-level power characterization we would require a matrix of C^{2K} elements to characterize the interinstruction effects between \mathbf{w}_n and \mathbf{w}_{n-1} .

Let us now analyze the terms σ_s^n and μ_s^n that consider the energy consumed by stage s whenever the number of cycles to execute \mathbf{w}_n exceeds one, since there is a miss either in the data or in the instruction cache.

B. Average Energy per Stage Due to a Data Cache Miss σ_s^n

The energy contribution σ_s^n accounts for the core energy consumption due to a data cache miss. In this case, the processor stalls all the instructions in the pipeline until the write or read data cache miss is resolved. This term is quite important since it depends on the probability per instruction that the processor stalls the pipeline

$$\sigma_s^n = m_s^n * p_s^n * S_s \quad (11)$$

where m_s^n is the average number of additional cycles (stall cycles) occurred due to a data cache miss during the execution of the \mathbf{w}_n in s , p_s^n is the probability that this event occurs, and S_s is the energy consumption per stage of the processor modules that are active due to a data cache miss. In particular, our model considers the energy cost of a data cache miss on the processor core, but we do not consider the energy cost of the cache and memory arrays when this event occurs.

C. Average Energy per Stage Due to an Instruction Cache Miss μ_s^n

Similarly, the energy contribution μ_s^n accounts for the additional energy spent by instruction \mathbf{w}_n due to an instruction cache miss. When this event occurs, the instruction cache generates an IC-MISS signal as explained before. This contribution can be written as

$$\mu_s^n = l_s^n * q_s^n * M_s \quad (12)$$

where l_s^n is the average number of additional cycles (IC-MISS penalty cycles) occurred after the execution of the \mathbf{w}_n in s due to an instruction cache miss, q_s^n is the probability that this event occurs, and M_s is the energy consumption per stage of the processor modules that are active due to an instruction cache miss. In particular, our model considers the energy cost of an instruction cache miss on the processor core, while we do not target the energy cost of the cache and memory arrays.

As an example, let us consider an example of computation of $A_s(\mathbf{w}_n|\mathbf{w}_{n-1})$ in the case in which an instruction cache miss occurs after \mathbf{w}_n . Effects of such event must be taken into account because the IC-MISS signal is propagated within the pipeline and this affects the energy behavior of \mathbf{w}_n . If we refer to the example of Fig. 4, this case corresponds to the evaluation of $A_s(E|D) = U_s(E|D) + l * q_s^E * M_s + m * p_s^E * S_s$ where $q_s^E = 1 \forall s$, and $p_s^E = 1$ for $s \in \{MEM, EX, RR, ID, IF\}$ and $p_{WB}^E = 0$.

D. Reduction of the Model to a Scalar Pipeline

The proposed energy model for VLIW pipelined processors can be mapped to *scalar* pipeline processors, where the spatial dimension is reduced to one. In this case, our model complies with the model of Tiwari *et al.* summarized in (1)

$$B_i = \sum_{s \in S} [U_s(\mathbf{0}|\mathbf{0}) + \nu_s(i|i)] \quad (13)$$

$$O_{i,j} = \sum_{s \in S} [\nu_s(i|j) - \nu_s(i|i)] \quad (14)$$

$$\sum_k E_k = \sum_{1 \leq n \leq N, s \in S} [\mu_s + \sigma_s]. \quad (15)$$

E. Model Characterization

The proposed model is built upon a set of energy parameters ($\mathbf{U}_s(\mathbf{0}|\mathbf{0})$, ν_s , M_s , and S_s) that must be characterized once and for all by the manufacturer of the VLIW core.

Both $\mathbf{U}_s(\mathbf{0}|\mathbf{0})$ and ν_s can be characterized by reformulating the model of (8) in the following matrix form:

$$\mathbf{U}_s = \mathbf{X}\beta_s \quad (16)$$

where \mathbf{U}_s is a vector of E elements representing the observed energy consumption for E experiments where each experiment e is a program composed of a sequence of the same couple of instructions ($\mathbf{w}_{e,a}$, $\mathbf{w}_{e,b}$).

Vector β_s has the following structure:

$$\beta_s = \begin{bmatrix} U_s(\mathbf{0}|\mathbf{0}) \\ \Theta_s \end{bmatrix} \quad (17)$$

where $U_s(\mathbf{0}|\mathbf{0})$ is the scalar value representing the base cost of the instruction while Θ_s is a vector of T elements that linearly stores the elements of the upper diagonal part of the ν_s (since it is a symmetrical matrix) except for the term $\nu_s(\text{NOP}|\text{NOP})$. If C is the number of classes in the ISA, then $T = ((C^2+C)/2) - 1$.

\mathbf{X} is an $E \times (T+1)$ matrix with the following structure:

$$\mathbf{X} = \begin{bmatrix} 1 & \eta_1 \\ \dots & \dots \\ 1 & \eta_e \\ \dots & \dots \\ 1 & \eta_E \end{bmatrix} \quad (18)$$

where η_e is a row vector of T elements associated with each experiment e such that

$$\eta_e \times \Theta_s = \sum_{\forall k \in K} \nu_s(w_{e,a}^k | w_{e,b}^k) \quad (19)$$

where “ \times ” represents the scalar product. As a matter of fact, each element $\eta_e(i)$ stores the number of times that the i th term of Θ_s must be taken into account into the computation of (19). This vector can be directly derived from the experimental setup chosen for experiment e .

At this point, the vector of parameters β_s can be estimated by means of a linear regression. In order to perform the linear regression, the rank of \mathbf{X} must be of $((C^2+C)/2)$, that is, the design of the experiments must generate at least $((C^2+C)/2)$ experiments whose pairs ($\mathbf{w}_{e,a}$, $\mathbf{w}_{e,b}$) are linearly independent.

Given this lower bound, for a K -issue VLIW machine with S pipeline stages with an ISA composed of C classes of operations, the problem complexity of calculating the energy parameters is thus reduced from $O(SC^{2K})$ to $O(SC^2)$, where C is the number of operation classes derived by clustering the ISA.

The remaining energy parameters (M_s and S_s) are simply estimated by measuring the power consumption of the processor during, respectively, an I-cache miss and a pipeline stall.

F. Plug-In of the Model Into an ISS

Application-dependent variables composing the model must be estimated or computed during an instruction set simulation. These contributions can be gathered either in accurate mode or in statistically approximated mode.

In the accurate mode, operations ordering and cache miss length are dynamically computed by the simulator while the event of a cache miss is predicted either statistically ($p_s^n, q_s^n \in [0, 1]$) or deterministically ($p_s^n, q_s^n \in \{0, 1\}$). The accurate mode is used by an instruction set simulator that elaborates the power model to compute on-the-fly the power consumption of the program.

In the statistically approximated mode, operations ordering, cache miss length, and operation latencies are statistically averaged values and they are used to elaborate an average off-line power profile of the application. In this case, while p_s^n and q_s^n become average values \bar{p}_s and \bar{q}_s , respectively, the term $\nu_s(w_n^k | w_{n-1}^k)$ is substituted by a weighted average value $\bar{\nu}_s$

$$\bar{\nu}_s = \sum_{o_j^k, o_h^k \in ISA} p(o_j^k | o_h^k) \nu_s(o_j^k | o_h^k)$$

where $p(o_j^k | o_h^k)$ is the probability, estimated by the ISS, that operation o_j^k follows o_h^k in lane k .

IV. EXPERIMENTAL RESULTS

In this section, we describe how the energy model has been successfully used to characterize at the instruction-level the energy consumption of a VLIW-based in-house embedded system.

A. Experimental Environment

To validate the proposed energy model, we developed a VLIW-based system-level design environment (see Fig. 5) composed of

- 1) four-issue VLIW processor with six-stage pipeline characterized by an ISA composed of five operation classes (see Table I);
- 2) configurable instruction cache;
- 3) configurable data cache;
- 4) 72×32 -bit register file;
- 5) main memory containing the object code to be executed by the VLIW processor.

The processor pipeline is constituted by six stages [fetch (IF), decode (ID), register read (RR), execute (EX), memory access (MEM), and write back (WB)] and four lanes; each lane can execute an integer operation (with a dedicated ALU), a multiply operation (with a dedicated multiplier), or a load/store operation

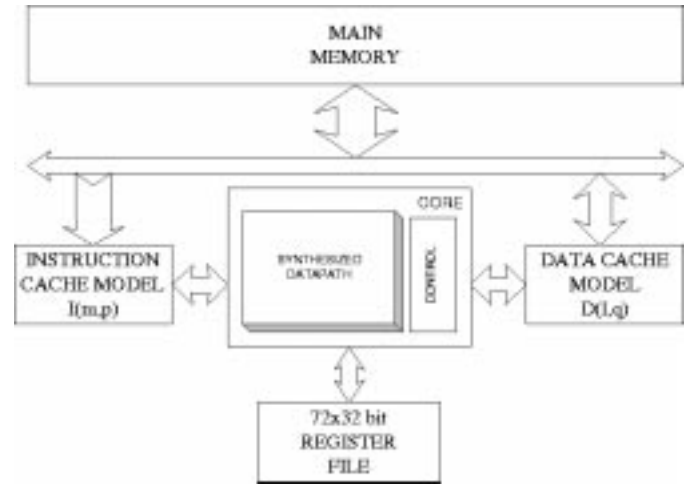


Fig. 5. The system-level test-bench used to validate the model.

through a load/store unit common to all the lanes. The VLIW core is provided with data forwarding and operation latency that varies between one and two cycles (in the case of load/store operation) with in-order issue and in-order completion.

The configurable instruction cache $I(l, q)$ produces, within a normal stream of execution, a miss signal for l cycles with an average probability q for each normal instruction fetched.

The configurable data cache $D(m, p)$ can generate a miss with a probability p for each access and, during a miss, it stalls the VLIW pipeline for m cycles.

All design modules have been described in Verilog-HDL. The data path of the processor has been described at RT-level and it has been synthesized as multilevel logic by using Synopsys Design Compiler and mapped into a commercial $0.25\text{-}\mu\text{m}$ and 2.5-V technology library. Gate-level simulations of the core within the Verilog test-bench have been performed by using Synopsys VCS 5.2. We used PLI routines from Synopsys to retrieve toggle information about the core cells and we used the resulting toggle file as input to Synopsys Design Power tool. Energy estimates have been obtained with Synopsys Design Power tool by assuming a 100-MHz clock frequency.

B. Design of Experiments

The experimental design flow has been decomposed in two main phases: the first phase is the characterization of the model to derive the energy parameters and the second phase is the validation of the characterized model against the execution of real embedded programs.

1) *Experiments for Model Characterization:* Following (16), an automatic tool has been used to generate a set of 250 experiments where each experiment e is a program composed of a sequence of 1000 pairs of instructions ($w_{e,a}, w_{e,b}$). The automatic tool satisfies the requirements imposed on the rank of the matrix of experiments \mathbf{X} [see (16)]. In fact, given $C = 5$ we can easily reach $\text{rank}[\mathbf{X}] = 15$.

The experiments have been generated by varying the following parameters:

- 1) number and type of operations within the pair ($w_{e,a}, w_{e,b}$);

TABLE II

CHARACTERIZATION RESULTS FOR THE MATRIX ν SUMMED OVER THE PIPELINE STAGES FOR ALL PAIRS OF OPERATIONS WHEN NEITHER PIPELINE STALLS NOR I-CACHE MISSES OCCUR. ENERGY VALUES ARE GIVEN IN pJ

$\sum_{s \in S} \nu_s$	NOP	AL	MUL	DT	CNT
NOP	0	339	350	349	172
AL	-	270	363	450	385
MUL	-	-	429	424	467
DT	-	-	-	411	467
CNT	-	-	-	-	80

- 2) registers used within the instructions (registers are addressed through seven-bit addresses);
- 3) values of the nine-bit immediate operands used within the instructions.

Finally, the energy consumption of the stages of the processor during stalls and I-cache misses have been measured by directly gathering toggle information.

2) *Experiments for Model Validation*: The model validation phase has been decomposed in four main steps.

- Step 1 consists of the validation of the model when neither I-cache misses nor D-cache misses occur. To perform this step, we generated a set of 250 experiments, where each experiment e is a program composed of a sequence of 1000 pairs of instructions ($w_{e,a}, w_{e,b}$).
- Step 2 represents the model validation during pipeline stalls. In this case, a set of experiments have been generated such that the average stall probability per instruction [forced by $D(m, p)$] ranges from 1% to 14% (according to [20], this is the typical miss rate range for a generic cache). The parameter m , in this case, has been fixed to 25 cycles.
- Step 3 consists of the validation of the model during I-cache misses. In this case, another set of experiments has been generated such that the average miss rate of $I(l, q)$ ranges from 1% to 14%. The parameter l , in this case, has been fixed to 36 cycles.
- Step 4 consists of the model validation against a set of kernels derived from embedded DSP algorithms. These kernels are characterized by several behaviors in terms of instructions composition and I- and D-cache misses.

C. Model Characterization

The proposed model for the core has first been characterized to derive the energy parameters. The measured energy base cost $U_s(0|0)$ summed over all the pipeline stages is 1490.57 pJ. Table II reports the characterization results for the matrix ν summed over the pipeline stages for all pairs of operations when neither pipeline stalls nor I-cache misses occur. The value of M_s and S_s parameters summed over the pipeline stages are 1400 and 110 pJ, respectively.

D. Model Validation

The experimental results derived during Step 1 have been reported in Table III and Fig. 6. Table III reports the comparison results of measured with respect to estimated power values for each pipeline stage and for the interconnection network.

TABLE III

COMPARISON RESULTS OF MEASURED VERSUS ESTIMATED POWER VALUES FOR EACH PIPELINE STAGE AND FOR THE INTERCONNECTION

Pipeline Stage	Avg.Err.	Std. Dev.	Max. Err.
IF	4,24%	3,15%	13,69%
ID	1,27%	1,13%	8,06%
RR	3,54%	2,02%	11,17%
EX	6,75%	4,16%	28,26%
MEM	5,43%	5,81%	26,19%
INTERC.	13,59%	14,33%	116,91%

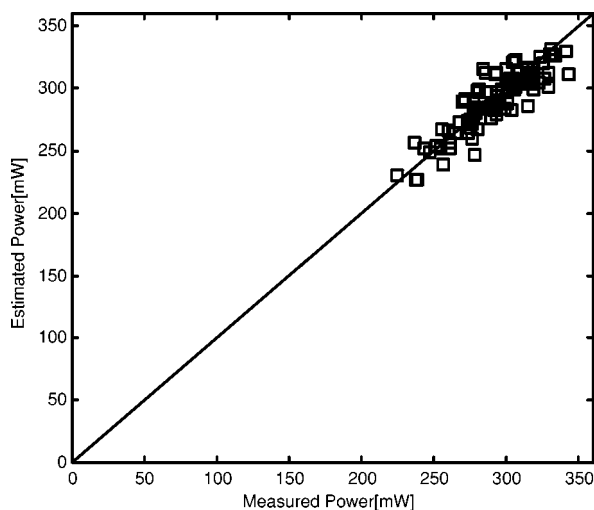


Fig. 6. Agreement between measured and estimated power values when neither I-cache misses nor D-cache misses occur. (Absolute maximum error 11.4%.)

Note that, since the RF power consumption is not considered here (being the RF outside of the core) and the pipeline latch MEM/WB is contained in the module of pipeline stage MEM , the power consumption due to the WB stage is considered zero and therefore it has been omitted. Fig. 6 reports the agreement between measured and estimated power values for the whole in-house VLIW core, when neither I-cache misses nor D-cache misses occur. For the set of experiments, the absolute maximum error is 11.4%, the absolute average error is 3.19%, and the standard deviation is 2.07%.

For Step 2, Fig. 7 shows the agreement between measured and estimated power values when pipeline stalls occur. For the given experiments, the absolute maximum error is 20.2%, the absolute average error is 3.1%. The standard deviation of the error is 2.7%. Note that the lower values of power consumption are associated with experiments with a higher data cache miss probability. In fact, as the miss probability increases, the power consumption approaches the stall power consumption S_s summed over the pipeline stages.

Concerning Step 3, Fig. 8 reports the agreement between measured and estimated power values when I-cache misses occur. The absolute maximum error and the absolute average error are 9.6% and 3.2%, respectively. The standard deviation of the error is 2.2%. Note that, in this case, the lower values of power consumption are associated with experiments with a higher instruction cache miss probability. In fact, as the miss probability increases, the power consumption approaches the

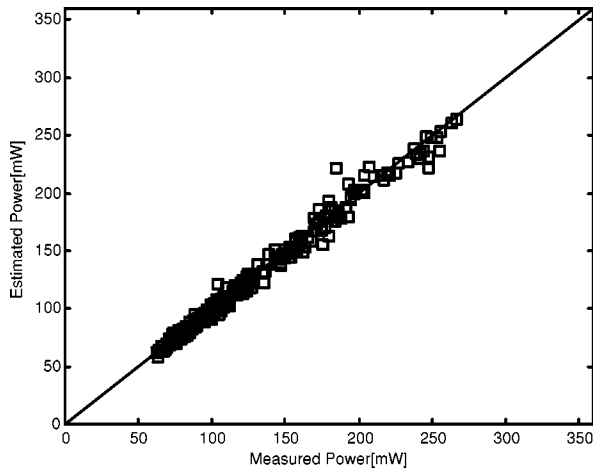


Fig. 7. Agreement between measured and estimated power values when D-cache misses occur. (Absolute maximum error 20.2%.)

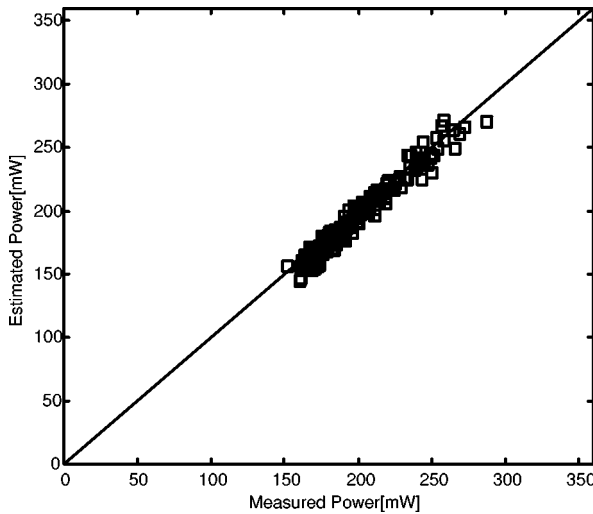


Fig. 8. Agreement between measured and estimated power values when I-cache misses occur. (Absolute maximum error 9.6%.)

instruction cache miss power consumption M_s summed over the pipeline stages.

Finally, for Step 4, Fig. 9 reports the agreement between measured and estimated power values for the given set of DSP benchmarks, where both pipeline stalls and I-cache misses parameters can vary. The given kernels (written in assembly code for the target processor) are

- 1) an unoptimized DCT and IDCT transform applied to a 64 half-word block;
- 2) an optimized version of the DCT;
- 3) an optimized version of the IDCT;
- 4) a finite-impulse response (FIR) routine (32-tap, 16-bit);
- 5) an optimized (unrolled) FIR routine (32-tap, 16-bit);
- 6) a Gaussian elimination algorithm.

The maximum absolute error between the observed and the estimated power is 10%. The average absolute error is 4.8%. Full core estimation at the gate-level has been shown four orders of magnitude slower than instruction-level simulation. In our experiments, an underestimation has been shown that can be due to the approximation used in our characterization process to capture the data behavior of single instructions.

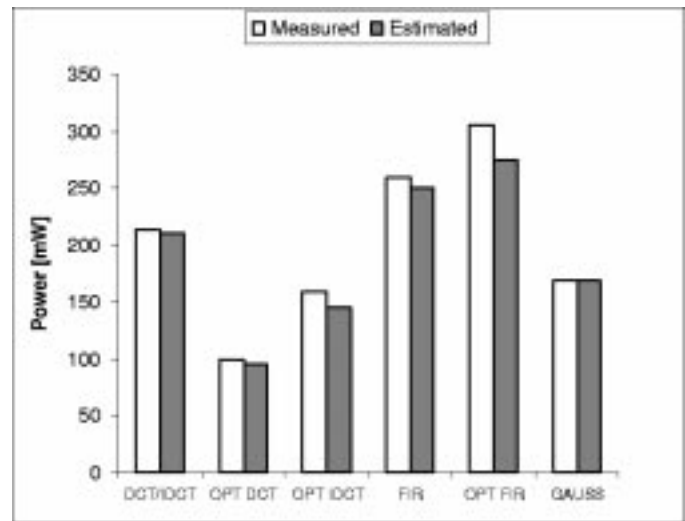


Fig. 9. Comparison between measured and estimated power values for the given set of DSP benchmarks. (Absolute maximum error 10%.)

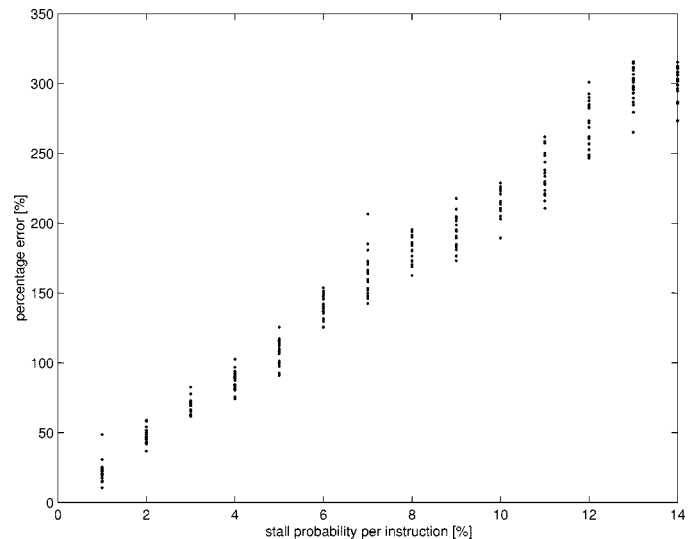


Fig. 10. Errors generated not considering D-cache miss events.

As a final example, let us consider the error estimates generated when both D-cache and I-cache misses are not considered in the model. In this case, the energy for each instruction A_s becomes the ideal energy U_s , thus we would expect underestimation on the power consumption. However, the total energy computed is divided by the ideal execution time that is much less than the real execution time giving an overestimation of the actual power value.

Fig. 10 shows the percentage errors between the predicted power values and the measured power values when stall events are not considered. Miss probability values are the same used for the characterization and validation of the model. Note that if stall events are not included in the model, the predicted length of one instruction is ideal, and the predicted instruction energy consumption (even not considering the energy lost during stalls) is concentrated in only one cycle giving an overestimation on power consumption almost proportional to the stalls per instruction rate.

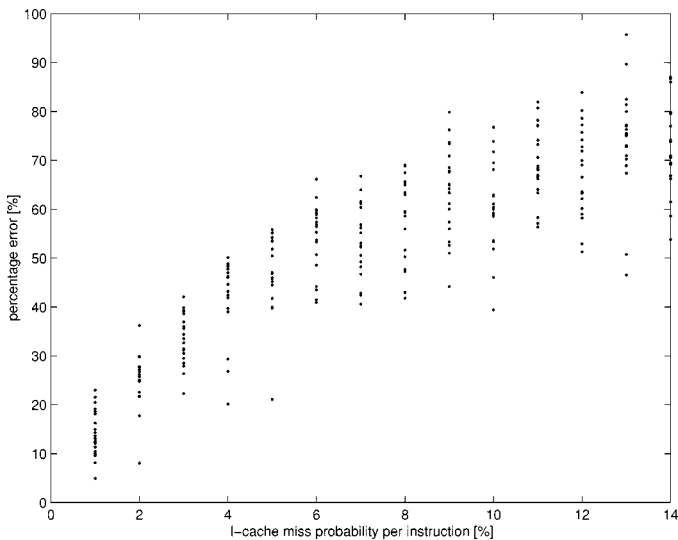


Fig. 11. Errors generated not considering I-cache miss events.

Fig. 11 shows the percentage errors between the predicted power values and the measured power values when instruction cache miss events are not considered. Miss probability values are the same used for the characterization and validation of the model. If I-cache miss events are not included in the model, the predicted length of one instruction is ideal, and the predicted instruction energy consumption is concentrated in only one cycle giving, even here, an overestimation on power consumption almost proportional to I-cache misses per instruction rate.

V. CONCLUDING REMARKS AND FUTURE WORK

In this paper, an instruction-level energy model for VLIW pipelined processors have been proposed and validated over a given set of DSP-based benchmarks. The choice of this particular set of benchmarks, with respect to general-purpose ones, is justified by our analysis targeting high-performance multimedia embedded processing. The proposed model addresses the estimation of the power budget for VLIW-based single-cluster architectures, where all operations in a long instruction access the same set of functional units and the same register file. Future works aim at exploring multiclustered VLIW architectures. Other future directions of the work aim at: 1) integrating the proposed energy model in a system-level exploration methodology to evaluate the impact of architectural parameters on energy-performance tradeoffs; 2) using the energy model results to define microarchitectural optimizations in the processor; and 3) using the instruction-level energy model to assess compilation optimizations such as instruction scheduling and register labeling.

REFERENCES

- [1] A. Chandrakasan and R. Brodersen, "Minimizing power consumption in digital CMOS circuits," *Proc. IEEE*, vol. 83, pp. 498–523, Apr. 1995.
- [2] P. Landman, "High-level power estimation," in *Proc. ISLPED96: Int. Symp. Low Power Electronics and Design*, 1996, pp. 29–35.
- [3] E. Macii, M. Pedram, and F. Somenzi, "High-level power modeling, estimation and optimization," *IEEE Trans. Computer-Aided Design*, vol. 17, pp. 1061–1079, Nov. 1998.

- [4] P. Landman and J. Rabaey, "Architectural power analysis: The dual bit type method," *IEEE Trans. VLSI Syst.*, vol. 3, pp. 173–187, June 1995.
- [5] M. Sami, D. Sciuto, C. Silvano, and V. Zaccaria, "Instruction-level power estimation for Embedded VLIW cores," in *CODES 2000: Eighth Int. Workshop on Hardware/Software Codesign*, San Diego, CA, May 3–5, 2000, pp. 34–38.
- [6] M. Sami, D. Sciuto, C. Silvano, and V. Zaccaria, "Power exploration for embedded VLIW architectures," in *Proc. ICCAD-2000: IEEE/ACM Int. Conf. Computer-Aided Design*, San Jose, CA, Nov. 5–9, 2000, pp. 498–503.
- [7] F. N. Najm, "A survey of power estimation technique in VLSI circuits," *IEEE Trans. VLSI Syst.*, vol. 2, pp. 446–455, Dec. 1994.
- [8] T. Chou and K. Roy, "Accurate estimation of power dissipation in CMOS sequential circuits," *IEEE Trans. VLSI Syst.*, vol. 4, pp. 369–380, Sept. 1996.
- [9] C. Hsieh and M. Pedram, "Microprocessor power estimation using profile-driven program synthesis," *IEEE Trans. Computer-Aided Design*, vol. 17, pp. 1080–1089, Nov. 1999.
- [10] D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: A framework for architectural-level power analysis and optimizations," in *Proc. ISCA-27*, June 2000, pp. 83–94.
- [11] W. Ye, N. Vijaykrishnan, M. Kandemir, and M. J. Irwin, "The design and use of SimplePower: A cycle-accurate energy estimation tool," in *Proc. DAC2000: IEEE/ACM Design Automation Conf.*, Los Angeles, CA, June 2000, pp. 340–345.
- [12] T. Sato, Y. Ootaguro, M. Nagmatsu, and H. Tago, "Evaluation of architecture-level power estimation for CMOS RISC processors," in *ISLPED-95: ACM/IEEE Int. Symp. Low Power Electronics and Design*, Oct. 1995, pp. 44–45.
- [13] P. W. Ong and R.-H. Yan, "Power-conscious software design: A framework for modeling software on hardware," in *Proc. ISLPED-94: ACM/IEEE Int. Symp. Low Power Electronics and Design*, San Diego, CA, Oct. 1994, pp. 36–37.
- [14] D. M. Brooks, P. Bose, S. E. Schuster, H. Jacobson, P. N. Kudva, A. Buyuktosunoğlu, J.-D. Wellman, V. Zyuban, M. Gupta, and P. W. Cook, "Power-microarchitecture: Design and modeling challenges for next-generation microprocessors," *IEEE Micro.*, vol. 20, no. 6, pp. 26–44, Nov./Dec. 2000.
- [15] V. Tiwari, S. Malik, and A. Wolfe, "Power analysis of embedded software: A first step toward software power minimization," *IEEE Trans. VLSI Syst.*, pp. 437–445, Dec. 1994.
- [16] M. T.-C. Lee, V. Tiwari, S. Malik, and M. Fujita, "Power analysis and minimization techniques for embedded DSP software," *IEEE Trans. VLSI Syst.*, vol. 5, pp. 123–135, Mar. 1997.
- [17] J. T. Russel and M. F. Jacome, "Software power estimation for high performance 32-bit embedded processors," in *Proc. Int. Conf. Computer Design: VLSI in Computers and Processors 1998*, pp. 328–333.
- [18] D. Sarta, D. Trifone, and G. Ascia, "A data dependent approach to instruction-level power Estimation," in *Low Power Design, 1999 Proc. IEEE Alessandro Volta Memorial Workshop on*, Como, Italy, Mar. 1999, pp. 182–190.
- [19] B. Klass, D. E. Thomas, H. Schmit, and D. F. Nagle, "Modeling inter-instruction energy effects in a digital signal processor," in *Power-Driven Microarchitecture Workshop, in Conjunction With Int. Symp. Computer Architecture*, June 1998.
- [20] J. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*, 2nd ed. San Mateo, CA: Morgan Kaufmann, 1996.
- [21] K. Masselos, C. E. Goutis, F. Catthoor, and H. DeMan, "Interaction between sub-word parallelism exploitation and low power code transformations for VLIW multi-media processors," in *Low Power Design, 1999 Proc. IEEE Alessandro Volta Memorial Workshop on*, Como, Italy, Mar. 1999, pp. 52–60.
- [22] H. Mehta, R. M. Owens, M. J. Irwin, R. Chen, and D. Ghosh, "Techniques for low energy software," in *Proc. ISLPED-97: ACM/IEEE Int. Symp. Low Power Electronics and Design*, Monterey, CA, Aug. 1997, pp. 72–75.
- [23] E. Macii, "Sequential synthesis and optimization for low power," in *Low Power Design in Deep Submicron Electronics, NATO ASI Series, Series E: Applied Sciences*. New York: Kluwer, 1997, vol. 337, pp. 321–353.
- [24] G. Goossens, J. Van Praet, D. Lanneer, W. Geurts, A. Kifli, C. Liem, and P. G. Paulin, "Embedded software in real-time signal processing systems: Design technologies," *Proc. IEEE*, vol. 85, pp. 436–454, Mar. 1997.
- [25] M. Kamble and K. Ghose, "Analytical energy dissipation models for low power caches," in *Proc. ISLPED-97: ACM/IEEE Int. Symp. Low Power Electronics and Design*, Monterey, CA, Aug. 1997, pp. 143–148.

- [26] H. Corporaal, *Microprocessor Architectures From VLIW to TTA*. Chichester, U.K.: Wiley.
- [27] N. Bellas, I. N. Hajj, C. D. Polychronopoulos, and G. Stamoulis, "Architectural and compiler techniques for energy reduction in high-performance microprocessors," *IEEE Trans. VLSI Syst.*, vol. 8, pp. 317–326, June 2000.
- [28] Y. Li and J. Henkel, "A framework for estimating and minimizing energy dissipation of embedded HW/SW systems," in *Proc. DAC-35: ACM/IEEE Design Automation Conf.*, June 1998, pp. 188–193.
- [29] G. Qu, N. Kawabe, K. Usami, and M. Potkonjak, "Function-level power estimation methodology for microprocessors," in *Proc. DAC 2000: ACM/IEEE Design Automation Conf.*, June 2000, pp. 810–813.
- [30] 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 Trans. VLSI Syst.*, vol. 8, no. 2, pp. 129–137, Apr. 2000.
- [31] W. Ye, N. Vijaykrishna, M. Kandemir, and M. Irwin, "The design and use of SimplePower: A cycle-accurate energy estimation tool," in *Design Automation Conf.*, June 2000, pp. 340–345.
- [32] K. Roy and M. C. Johnson, "Software design for low power," in *Low Power Design in Deep Submicron Electronics, NATO ASI Series, Series E: Applied Sciences*. New York: Kluwer, 1997, vol. 337, pp. 433–460.
- [33] A. Bona, M. Sami, D. Sciuto, C. Silvano, V. Zaccaria, and R. Zafalon, "Energy estimation and optimization of embedded VLIW processors based on instruction clustering," in *Design Automation Conf.*, New Orleans, LA, June 2002.

Mariagiovanna Sami (M'86) obtained the Dr.Eng. degree in electronic engineering from Politecnico di Milano in 1966 and the Libera Docenza (computing and switching theory) in 1971.

Since graduation, she has been with the Department of Electronics and Information, Politecnico di Milano, where she obtained various research and teaching positions and where she has been a Full Professor since 1980. Her research interests include the areas of fault tolerance, highly parallel VLSI and WSI architectures, and high-level synthesis. She has published more than 150 papers in international journals and conference proceedings and she is co-author of the book *Fault Tolerance Through Reconfiguration in VLSI and WSI Arrays* (Cambridge, MA: MIT Press).

Dr. Sami is member of EUROMICRO. She was editor-in-chief of the EUROMICRO Journal and she was in the board of editors of the IEEE TRANSACTIONS ON COMPUTERS.



Donatella Sciuto (M'90) received the Laurea in electronic engineering in 1984. She received the Ph.D. degree in electrical and computer engineering, in 1988, from University of Colorado, Boulder.

She was 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. Her research interests include 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.

Dr. Sciuto is a Member of IFIP 10.5, EDAA and a member of different program committees of EDA conferences: DAC, ICCAD, DATE, CODES/CASHE, DFT, and FDL. She is a member of the executive committee of ICCAD and DATE, and an Associate Editor of the IEEE TRANSACTIONS ON COMPUTERS, the *Journal Design Automation of Embedded Systems*, and the *Journal of System Architecture*. She was a Guest Editor of a special issue of IEEE DESIGN AND TEST MAGAZINE in 2000.

Cristina Silvano (M'93) received the Dr.Eng. degree in electronic engineering from the Politecnico di Milano, Italy, in 1987, and the Ph.D. degree in information engineering from the University of Brescia, Italy, in 1999.

From 1987 to 1996, she was with R&D Labs in Pregnana M., Italy, of Bull HN Information Systems, where she held the position of Senior Design Engineer. From 1996 to 1998, she was with the Dipartimento di Elettronica per l'Automazione, University of Brescia. From 1998 to 1999, she was a Consultant Researcher in the Electronic Design Automation Area at Center for the Research and the Education in Information Engineering (CEFRIEL), Milan. Currently, she is an Assistant Professor at the Dipartimento di Scienze dell'Informazione, University of Milano. Her primary research interests include the areas of complex computer architectures based on microprocessors and design automation methodologies of VLSI circuits and systems, with particular emphasis on low-power design techniques, high-level power estimation methodologies, and hardware/software codesign methodologies for embedded systems.

Vittorio Zaccaria (S'98) received the Dr. Eng. degree and the Ph.D. degree in computer engineering from the Politecnico di Milano, Italy, in 1998 and 2002, respectively.

Since December 2001, he has held a research contract as Assistant Professor at the Dipartimento di Elettronica e Informazione, Facoltà di Ingegneria, Politecnico di Milano. His research interests include algorithms, methodologies, and tools for the design of low-power digital systems with particular emphasis on microprocessor based embedded architectures.