Instruction-Level Power Estimation for Embedded VLIW Cores

M. Sami D. Sciuto C. Silvano V. Zaccaria

Politecnico di Milano
Dip. di Elettronica e Informazione
Milano, ITALY 20133

ABSTRACT

In this paper, a power estimation methodology operating at the instruction-level is proposed. The methodology is tightly related to the characteristics of the system architecture, mainly in terms of one or more target processors, the memory sub-system, the system-level buses and the co-processors. In this system-level framework, our main goal is to define a power model for CPU cores at the instruction-level. First, the proposed power model deals with a general five-stage pipeline processor architecture, then, the model is extended to VLIW processors. The derivation of a VLIW instruction-level power model results to be intractable from the point of view of spatial complexity (which grows exponentially w.r.t. the number of possible operations in the ISA). In order to tackle this complexity, a new kind of simplification, based on the original concept of *separability* of processor functional units, is introduced. The proposed system-level methodology is the first step toward a more general framework to support the design of power-oriented applications through hardware/software co-design. [1]

1. INTRODUCTION

Low power is an increasingly relevant requirement for ever wider classes of embedded systems [1]. A reasonably efficient approach to power estimation at the higher levels of abstraction is of fundamental importance for system design. Early availability of such figures will make it possible to guide the subsequent design choices, from hardware/software partitioning down to technology mapping.

Aim of our work is the definition of a power-oriented methodology suitable for embedded systems based on VLIW cores. The system-level methodology includes power models for each module composing the target system architecture, mainly one or more processors, the memory sub-system, the system-level buses and the co-processors. The target processor architecture is a pipelined VLIW core in both single-cluster and multi-cluster configuration. In this system-level scenario, the main focus of the present paper is the definition of an instruction-level power estimation model for VLIW cores based on a micro-architectural model of the processor. The goal is to provide information on the power consumed by the processor core during code execution. At the same time, the micro-architectural model should provide data to the programmer/compiler on the hot spots where most power is consumed within the core micro-architecture. We assume this level of detail cannot be achieved by using a simple black-box instruction-level power model such as those presented in the literature so far.

The proposed model is quite general and parametric, to allow us the possibility of exploring the power budget by considering different architectural solutions for a VLIW core, given a specific application. The model is parametric with respect to: number of clusters, instruction formats, number of registers, number and type of functional units, interconnection buses, the type and number of operations in the long instruction, and so on.

In this paper, first we propose a new type of instruction-level power estimation model for pipelined processors that aims at unifying the micro-architectural view of the processor with its power behaviour. Then, we propose a power characterization model of a general VLIW pipelined architecture based on the model derived for pipelined processors. We introduce the concept of *separability* of the processor functional units, to deal with the spatial complexity of the instruction-level power model for VLIW cores, which grows exponentially with the number of possible operations in the ISA.

In our model, we define a mapping between very-long instructions (bundles) and micro-architectural functional units involved during the bundle execution. The bundle-to-unit mapping provides power estimates during the instruction-level simulation. The instruction-set simulator (ISS) transfers information to the micro-architectural power model on which units are involved in execution, and, at the same time, it obtains from the model detailed estimates on the power consumption on a cycle-by-cycle basis. The main goal of the integration of the micro-architectural power model and the bundle-to-unit mapping is the definition of an efficient interaction mechanism for forward and backward transfer of information between the proposed model and the ISS. The forward flow of information from ISS to the power estimation engine and the backward flow of power estimates from the power model to the ISS, require minimum overhead on

[1] This work is partially supported by CNR (Project MADESS II) and ST Microelectronics.

the standard instruction-level simulation.

As further evolution of this work, the model will be refined by the assessment of the relationships between elementary sequences of very long instructions and relative power requirements. This includes both the choice of operations inserted in a single long instruction or distributed over a sequence of long instructions (i.e. spatial versus temporal inter-operations effects and dependencies) and the assessment of policies for register management. The information acquired can be passed to the user/compiler to provide further power optimization figures.

The paper is organized as follows. Previous literature on instruction-level power estimation has been summarized in Section 2. Section 3 introduces the general instruction-level power estimation methodology for pipelined processors aiming at modeling the energy associated with a single stage of the pipeline. The model has been extended in Section 4 to consider the power contributions of the different pipeline stages. Section 5 describes how the proposed model can be generalized to VLIW architectures, for which the complexity of the power estimation problem is reduced based on the concept of *separability* of the internal functional units. Experimental results carried out to demonstrate the validity of the proposed approach applied to a simplified VLIW architecture have been reported in Section 6. Finally, Section 7 concludes the paper by outlining some research directions originated from this work.

2. PREVIOUS WORK

Instruction-level power estimation [2] is a problem afforded only recently. As a consequence, only few proposals have been made on this subject, that yet lacks of a mathematical approach. The major contributions found in literature are based on empirical approaches. Tiwari *et al.* [2][3] explore instruction level power estimation using simple models such as *average energy per instruction*, which is derived from experimental measurements. They partially include an average *inter-instruction effect* energy.

The instruction-level power model proposed in [5] 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* has very small variance. In [4], the authors propose a new processor power model by considering possible inter-instruction effects as well as data statistics. Although the developed power model is quite accurate, it lacks general applicability, being developed only for a specific embedded processor. In [8], the authors propose to measure inter-instruction effects by considering only the additional energy consumption observed when a generic instruction is executed after a NOP (the power model derived is also called the *NOP model*). This model could be an effective solution to the problem of the spatial complexity proper of instruction-level power models.

In general, inter-instruction effects play an important role being highly correlated to the gate-level *switching activity* of the functional units of processors. In RISC and VLIW architectures [6], these effects can become particularly evident. Conversely, in complex architectures like CISCs, the heavy use of caches and microcode ROMs effectively increases the average instruction energy, but reduces its variance, masking inter-instruction effects.

3. INSTRUCTION-LEVEL ENERGY MODEL

Up to now, the existing approaches to software-level power estimation do not take into account the power behaviour of each pipeline stage of the processor. To characterize energy consumption of a single instruction, the processor is considered as a whole without analyzing the power behavior during the flow of the instructions through the pipeline stages.

This paper presents the first results of our work on software-level power estimation based on an accurate characterization of the pipeline stages. To describe our energy model, we introduce a five-stage load/store pipelined processor architecture (see Fig.1). [2] The stages composing the processor pipeline are:

- Instruction Fetch Stage (IF);
- Instruction Decode Stage (ID);
- Register Read Stage (RR);
- Execute Stage (EX);
- Write Back Stage (WB).

Let w_n be the n-th instruction of a program execution trace W. We assume the energy associated with w_n to be strictly dependent on the properties of w_n (e.g., type of instruction, registers accessed and data dependencies) as well as on its *execution context*, i.e., the set of instructions contained in W near to w_n. The *execution context* can be split in two major contributions:

- The preceding instruction w_{n-1}.
- The instructions w_k present in the pipeline during the execution time of w_n.

Given the processor pipeline composed of a set S of stages (where S={IF, ID, RR, EX, WB}), we propose to estimate the average energy E associated with w_n as follows:

$$E(w_n) \approx \sum_{s \in S} A_s(w_n|w_{n-1}) \quad (1)$$

where A_s is the energy consumed by stage s when executing instruction w_n after w_{n-1}. Then, we decompose the average energy consumption *per stage* in two terms:

$$A_s(w_n|w_{n-1}) \approx U_s(w_n|w_{n-1}) + T_s(w_n) \quad (2)$$

where term U_s is the energy consumption of stage s during an *ideal* execution of w_n in the absence of any hazards or exceptions, thus assuming *one cycle* execution per stage. Term U_s depends on the current instruction word executed and on the preceding one in the trace. U_s represents the contribution of the individual stage only and therefore can be evaluated independently. Term T_s is the incremental average energy consumed by stage s whenever either the number of cycles needed by stage s to elaborate w_n (*latency cycles*) exceeds one or stage s stalls while executing w_n because there is a data-path conflict (*stall cycles* due to resource conflicts, data hazards, or control hazards). In this case, w_n has to wait in s until the conflict is resolved and the average energy consumed depends only on w_n. In the rest of the paper, we denote by *stall/latency probability* the probability that stage s is in one of the above stall/latency states. This parameter is quite important because term T_s depends on the probability that the processor stalls the pipeline:

$$T_s(w_n) \approx m_s(w_n) * p_s(w_n) * S_s(w_n) \quad (3)$$

[2] The pipeline is modified with respect to the MIPS architecture presented in [6]; in particular, the memory access stage has been included in the EX stage, for which an ad-hoc power model has been developed.

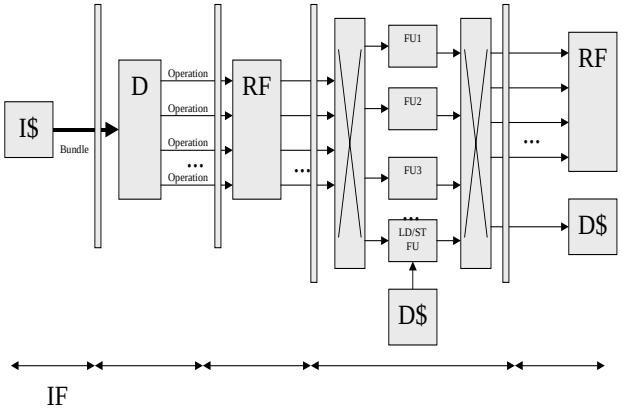

Figure 1: A general VLIW Pipelined Processor Architecture.

where m_s is the typical number of stall/latency cycles occurred while executing w_n, p_s is the stall/latency probability while executing w_n and S_s is the stage energy consumption for each of these stall/latency cycles. Note that p_s and m_s have to be experimentally measured by observing the behaviour of the pipeline during several executions of a given application or by simulating it cycle-by-cycle. Considering our target architecture, this is not a major issue because existing VLIW simulators (such as Trimaran [9]) can produce instrumented VLIW code to record this type of statistical values. Finally, S_s can be derived in the same manner as U_s, i.e., by simulating a gate level description of the processor.

4. ANALYSIS OF PIPELINE STAGES

Referring to the pipelined processor architecture in Fig. 1, in this section we analyze the contributions of the different pipeline stages.

4.1 IF Stage

The energy consumption of stage IF, A_{IF}, can be decomposed in two parts: *(i)* I-cache energy contribution $A_{IF,ic}$; *(ii)* energy contribution $A_{IF,logic}$ due to the fetch logic, such as issue buffers. I-cache energy contribution is strictly dependent on the I-cache hit ratio. Thus we propose, for $A_{IF,ic}$, the following approximation:

$$H * E_{hit} + (1 - H) * E_{miss} + T_{IF,ic}(w_n) \quad (4)$$

where H is the I-cache hit ratio and E_{hit} and E_{miss} are the average cache hit and miss energy consumption respectively; $T_{IF,ic}(w_n)$ is the stall energy consumption for the I-cache due to stalls of the current instruction caused by other stages and can be approximated as follows:

$$T_{IF,ic}(w_n) \approx m_{IF}(w_n) * p_{IF}(w_n) * S_{IF,ic}(w_n) \quad (5)$$

where m_{IF} and p_{IF} are the average stall/latency cycles and stall/latency probability not due to a cache miss, respectively, and $S_{IF,ic}$ is the relative cache energy consumption. Finally, fetch logic power consumption $A_{IF,logic}$ can be approximated by following the general power model:

$$U_{IF,logic}(w_n|w_{n-1}) + m_{IF}(w_n) * p_{IF}(w_n) * S_{IF,logic}(w_n) \quad (6)$$

where $U_{IF,logic}$ and $S_{IF,logic}$ represent, respectively, the single-cycle ideal execution and the stall/latency average power consumption of the fetch logic.

4.2 ID Stage

The average energy consumption $A_{ID}(w_n|w_{n-1})$ can be approximated by:

$$U_{ID}(w_n|w_{n-1}) + m_{ID}(w_n) * p_{ID}(w_n) * S_{ID}(w_n) \quad (7)$$

where U_{ID} is the single-cycle ideal execution energy consumption, m_{ID} is the average number of stall/latency cycles in the ID stage, p_{ID} is the average stall/latency probability and S_{ID} is the average stall/latency energy consumption of the decoding logic. The ideal (U_s) and stall (S_s) average energy are also strongly affected by the switching activity produced by the control word flow through the pipeline. This power consumption is due to the fact that the control word pipeline buffers drive other pipeline buffers and the controlled units.

4.3 RR Stage

In this stage, operand values are selected from the following sources:

- A multi-ported register file with $2n$ read ports and n write ports (where n is the number of slots in a bundle);
- A register by-pass network that presents result values of the current instruction in the EX stage. This forwarding network is used to reduce the latency of operation to 1 cycle.

We propose to approximate A_{RR} with the following expression:

$$\bar{R}_{RR} * (1 + m_{RR}(w_n) * p_{RR}(w_n)) \quad (8)$$

where $\bar{R}_{RR}$ is the average energy consumption per cycle of the entire pipeline stage. We take an average value because the register file power consumption is influenced by inter-stage effects (such as writes generated by the WB stage) which are very complex to model.

4.4 EX Stage

During this stage, instructions are dispatched to the dedicated FUs through a switching network. After the instruction execution, if the instruction is a simple operation between registers, the results are redirected to a register file write port. Conversely, if the instruction is a load operation, the data are fetched from the data-cache (stalling the pipeline in the case of a cache miss). On the other hand, a store instruction moves data towards the data-cache buffers in the WB stage. The energy consumption ($A_{EX}(w_n|w_{n-1})$) associated to stage EX can be modeled as:

$$U_{EX}(w_n|w_{n-1}) + m_{EX}(w_n) * p_{EX}(w_n) * S_{EX}(w_n) \quad (9)$$

where U_{EX} and S_{EX} take into account energy consumption due to: *(i)* functional units; *(ii)* switching networks; *(iii)* D-cache (including incoming reads from EX stage and incoming writes from WB stage).

The energy consumption of the functional units is strongly dependent on both the switching activity induced by inter-instruction effects and data correlation. Due to the inherent complexity and number of alternative implementations for switching networks, functional units and D-caches, their energy consumption is modeled statistically by taking an average value over all the executed instructions.

4.5 WB Stage

Finally, we model the WB stage energy consumption by:

$$U_{WB}(w_n|w_{n-1}) + m_{WB}(w_n) * p_{WB}(w_n) * S_{WB}(w_n) \quad (10)$$

Note that energy consumption due to register file writes and D-cache writes has been already considered in the RR and EX stages, thus WB energy parameters must account only for the remainder of the logic (e.g., stage buffers).

5. EXTENDING THE ENERGY MODEL TO VLIW ARCHITECTURES

In a VLIW processor, long instructions (or *bundles*) are composed of one or more explicitly parallel operations. After a bundle is fetched from the I-cache, each operation is dispatched to a specific functional unit (See Fig. 1). Unlike a superscalar processor, which can be considered an *atomic instruction architecture* [10], a VLIW processor does not check for any data or control dependency, thus the compiler must assure that the flow of operations does not present any type of *intra* or *inter*-bundle dependency.

As already noted in [8], the challenge of every instruction-level power model derives from the complexity of the spatial and temporal correlation of the instructions in the execution trace W. In fact, for each parameter we must maintain an n-dimensional array where n depends on the type of inter-instruction effects that we are considering. For example, the parameter $U_s(w_n|w_{n-1})$ takes into account only the inter-instruction effect between adjacent instructions and must be maintained in a bi-dimensional array whose dimension is proportional to the number of instruction in the ISA. In the case of a RISC machine this problem can be solved by clustering instructions [3], thus reducing the parameter's array length.

In contrast, a VLIW machine presents each bundle as composed by a set of operations [6]. The derived power model must account for all the allowable combinations of operations in a bundle, thus the problem complexity grows exponentially with the size of the bundle. As for the RISC case, instruction clustering can be effective to reduce the problem complexity by trading off power estimation accuracy with spatial complexity.

Power-per-stage analytical regression models based on few parameters, such as the average Hamming distance between operand addresses, can also be envisioned to reduce spatial complexity. In this case, the size of the problem would be reduced to the number of parameters considered (hopefully less than VLIW issue width). This approach would also support energy reduction through instruction scheduling for all the pipeline stages.

In this paper, we propose an approach aiming at reducing the problem complexity by introducing the concept of *separability*. Seemingly, since operations in a long instruction are by definition not-conflicting with each other (being executed simultaneously) they should be considered as *separable*, i.e. their contribution to the energy model should be simply additive. In reality, even by referring to a very simple architectural model (such as that in Fig. 1), it becomes evident that *real* separability fully applies only to the set of functional units, therefore to stage EX.

In that sense, the functional units are *separable* since each one of them is dedicated to execute one particular operation of the bundle. Therefore, no VLIW processor is entirely separable. In fact, although the interference between operations of the same bundle is minimized, there exist always some portions of the processor that are influenced by the concurrent execution of more than one operation at the same time. We define these portions of the processor as *not separable*. They are:

- The IF stage: We assume the processor fetches the instructions from a common I-cache, thus the fetch logic is not separable.
- The ID stage: Complex switching networks that redirect operations to the functional units are not separable.
- The RR stage: The structure of the multi-port register file used to support concurrent read and write operations is not separable.
- The WB stage: Same as the RR stage.

For not separable stages we propose to substitute parameters $U_s(w_n|w_{n-1})$ and $S_s(w_n)$ by their average, i.e., $\bar{U_s}$ and $\bar{S_s}$. In this way, the expression of $A_s(w_n|w_{n-1})$ becomes easier to compute:

$$A_s(w_n|w_{n-1}) \approx \bar{U_s} + m_s(w_n) * p_s(w_n) * \bar{S_s} \quad (11)$$

Besides, the standard deviation of A_s can be calculated to characterize the model accuracy.

Suppose now that stage s is separable, i.e., it is decomposable in D independent partitions, each one executing a particular stream of operations. In the case of S_s (that depends on only one instruction), we do the following approximation:

$$S_s(w) \approx \sum_{d=1}^{D} \sigma_s(o_d) \quad (12)$$

where o_d is the operation issued to partition d by w and $\sigma_s(o)$ is the average energy consumed by the stage while executing a long instruction composed only of an operation o (all other operations in the bundle being NOPs and thus not activating functional units). In the case of a regular VLIW architecture [6], the derivation of "o_d"s from a given bundle w is straightforward because each slot of the bundle can be conceptually assigned to one and only one partition. In this way, we reduce the complexity due to S_s from $O(N^d)$ to $O(d*N)$ (where N is the number of operations in the ISA) which is much more tractable. In the case of $U_s = U_s(w_1|w_2)$ (s being separable) we can apply the same approximation:

$$U_s(w_1|w_2) \approx \sum_{d=1}^{D} \nu_s(o_{1,d}|o_{2,d}) \quad (13)$$

where $o_{1,d}, o_{2,d}$ are the operations issued to partition d by the first and the second bundle respectively, and $\nu_s(o_1|o_2)$ is the average energy consumed by the stage while executing o_1 after o_2 on the same partition (assuming the rest of partitions being issued with NOPs).

6. EXPERIMENTAL RESULTS

To validate the proposed energy model, we have considered a simplified VLIW architecture ($SVLIW$ for brevity) provided with data forwarding and operation latency equal to one cycle. The $SVLIW$ architecture (shown in Fig. 2) can execute a two operation bundle $< o_1; o_2 >$ without any stall. The operations taken into account in our experiments are ADD and MUL functional operations and NOP operation. The processor architecture is composed of:

- A register file composed of 64 registers with 4 read-ports and 2 write-ports;
- 2 FUs (adder and multiplier) each one statically linked to 2 register file read ports and 1 write port;

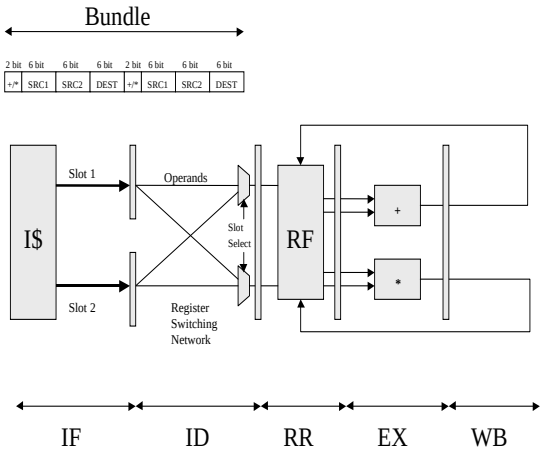

Figure 2: A simplified VLIW Pipelined Processor Architecture.

- A direct mapped 16-bundle I-cache with 1-bundle line size and 100% hit rate.

The $SVLIW$ processor has been described in VHDL. This description has been synthesized as a multi-level logic by using Synopsys Design Compiler and mapped onto the 0.35μm and $3.3V$ technology library supplied by ST Microelectronics. Energy consumption estimates have been obtained with Synopsis Design Power by assuming a $50MHz$ clock frequency.

A first set of experiments have been carried out to demonstrate the validity and the accuracy of the energy model proposed in equation (1), when the $SVLIW$ processor executes a sequence of instructions of the same type. In this case, the term A_s of equation (1) becomes:

$$A_s(w_n|w_{n-1}) = A_s(< o_1; NOP > | < o_1; NOP >) \qquad (14)$$

where $o_1 \in \{NOP, ADD, MUL\}$.

The columns of Table I report the corresponding power consumed by each stage s of the pipeline. Note that the energy consumed by the WB stage accounts only for stage buffers, while the energy due to write operations in the register file are taken into account in the energy associated with the RR stage. The row $\tilde{P}(w_n)$ shows the power estimated by applying equation (1), that is by adding the single contribution of each pipeline stage, while row $P(w_n)$ reports the average processor power associated with the instruction w_n. The value of $P(w_n)$ has been estimated by considering the processor as a black-box executing a trace of identical instructions. The percentage error of $\tilde{P}(w_n)$ with respect to $P(w_n)$ is equal to 4.94% for NOP, 12.62% for ADD, and 3.12% for MUL. These errors are justified by the fact that our model allows us to account for fine-grained switching activity and capacitive loads, while the black-box model inevitably masks some details and thus may be, in some cases, less accurate.

A second set of experiments have been carried out to demonstrate the validity of the separability assumption for the FUs of the $SVLIW$ processor. In this case:

$$A_s(w_n|w_{n-1}) = A_s(< o_1; o_2 > | < NOP; NOP >) \qquad (15)$$

where $o_1, o_2 \in \{NOP, ADD, MUL\} \wedge o_1 \neq o_2$.

From Table II, we can observe that the 39.0072 mW power consumption of the EX separable stage in the case of $< ADD; MUL >$ can be approximated (as stated by equation (13)), with the sum of the power figures associated with the $< ADD; NOP >$ and $< MUL; NOP >$ with a percentage error of 15.55% in excess.

Pipeline Stage	$o_1 = NOP$ $[mW]$	$o_1 = ADD$ $[mW]$	$o_1 = MUL$ $[mW]$
IF-cache	2.3818	3.3832	3.3832
IF-fetch	2.4725	2.4725	2.4725
ID	1.4967	2.4891	1.5853
RR	19.0749	22.2322	22.3636
EX	6.0191	11.6928	32.626
WB	2.9439	4.5638	4.5642
$\tilde{P}(w_n)$	34.3889	46.8336	66.9948
$P(w_n)$	36.1777	41.584	64.9657
% Error	4.94%	12.62%	3.12%

Table 1: Power consumption results for the $SVLIW$ processor executing instructions of the same type

Pipeline Stage	$o_1 = ADD$ $o_2 = NOP$ $[mW]$	$o_1 = MUL$ $o_2 = NOP$ $[mW]$	$o_1 = ADD$ $o_2 = MUL$ $[mW]$
EX	12.0712	33.0017	39.0072

Table 2: Power consumption results for the $SVLIW$ processor to prove the separability assumption

7. CONCLUDING REMARKS

We have presented a new instruction-level power estimation method which, for the first time, accounts for several architectural parameters such as number and typology of the pipeline stages as well as the stall probability and the latency of operations. We applied it to a simple VLIW processor to show its effectiveness and to demonstrate how to reduce the spatial complexity associated with this type of processor architectures. Our work is now oriented to validate this methodology by measuring the accuracy (absolute and relative) obtained on real processors while executing real instruction traces.

8. REFERENCES

[1] A. Chandrakasan and R. Brodersen, "Minimizing Power Consumption in Digital CMOS Circuits," *Proc. of IEEE*, 83(4), pp. 498-523, 1995.

[2] V. Tiwari, S. Malik and A. Wolfe, "Power Analisys of Embedded Software: A First Step Towards Software Power Minimization," *IEEE Trans. VLSI Systems*, pp. 437-445, Dec. 1994.

[3] M. T.-C. Lee, V. Tiwari, S. Malik and M. Fujita, "Power Analisys and Minimization Techinques for Embedded DSP Software," *IEEE Trans. VLSI System*, pp. 123-135, Mar. 1997.

[4] D. Sarta, D. Trifone and G. Ascia, "A Data Dependent Approach to Instruction Level Power Estimation," *Proc. of Volta '99*, Como, Italy, pp. 182-190, Mar. 1999.

[5] J. T. Russel and M. F. Jacome, "Software Power Estimation for High Performance 32-bit Embedded Processors," *Proc. of ICCD '98.*

[6] J. Hennessy and D. A. Patterson, *"Computer Architecture: A Quantitative Approach,"* Morgan Kaufmann Publishers, San Mateo, CA, Second Edition, 1996.

[7] C. Chakrabarti and D. Gaitonde, "Instruction Level Power Model of Microcontrollers," *Proc. of ISCAS '99.*

[8] B. Klass, D. E. Thomas, H. Schmit and D. F. Nagle "Modeling Inter-Instruction Energy Effects in a Digital Signal Processor," *Proc. of ISCAS '98.*

[9] Trimaran Home Page, http://www.trimaran.org

[10] H. Corporaal, *"Microprocessor Architectures from VLIW to TTA,"* John Wiley and Sons, Chichester, England.