

Power Exploration for Embedded VLIW Architectures

Mariagiovanna Sami Donatella Sciuto Cristina Silvano Vittorio Zaccaria

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

{sami,sciuto,silvano,zaccaria}@elet.polimi.it

Abstract

In this paper, we propose a system-level power exploration methodology for embedded VLIW architectures based on an instruction-level analysis. The instruction-level energy model targets a general pipeline scalar processor; several architectural parameters such as number and type of pipeline stages as well as average stall/latency cycles per instruction and inter-instruction effects are taken into account. The application of the proposed model to VLIW processors results intractable from the point of view of both spatial and temporal complexity (which grow exponentially w.r.t. the number of possible operations in the ISA). To reduce this complexity, the basic model has been extended by assuming that the energy associated with a long instruction is given by the sum of the energy associated with the single operations of the long instruction and the single pipeline stages. The instruction-level energy model has been applied to a simplified VLIW architecture to demonstrate the validity of the proposed approach.¹

1 Introduction

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

The overall goal of our work is to define a general power-oriented system-level optimization methodology suitable for embedded systems based on VLIW cores. We envision the possibility to introduce power optimization and architectural exploration during the high-level phases of system design: the compilation phase (through a power optimizing compiler), the definition of the target system architecture, the definition of the processor micro-architecture and the mapping of instructions and data to the memory address space. This system-level methodology is based on power models for each module composing the target system architecture, mainly one or more processors, the memory subsystem, the system-level buses and the co-processors.

This approach requires the definition of a suitable instruction-level power model [2] that enables us to evaluate the software power consumption for a pipelined VLIW

core in both single-cluster and multi-cluster configuration. The model must be general and parametric to allow the exploration of the power budget by considering different architectural solutions for the VLIW core, given a specific application. Typical parameters are: number of clusters, instruction formats, number of registers, number and type of functional units, interconnection buses, type and number of operations in the long instruction, and so on.

Research on the basic problem of instruction-level power estimation started only recently. As a consequence, only few proposals have been made on this subject, that yet lacks a rigorous approach. The major contributions found in literature are based on empirical approaches. Tiwari *et al.* [3][4] 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* is characterized by very small variance. In [6], 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 [7], inter-instruction effects are measured by considering only the additional energy consumption observed when a generic instruction is executed after a NOP (the proposed power model is also called the *NOP model*). This model could be an effective solution to the problem of 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 [8], 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, reducing inter-instruction effects. Other authors [9], [10] introduced a power optimization methodology from a user perspective, by operating purely at the software level, by pre-processing and restructuring the source code to reduce the power consumption of the executed code.

In this paper, our main focus is the definition of an instruction-level energy model to guide system-level exploration. The goal is to provide information on the energy consumed

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

by the processor core during an instruction-level simulation with minimum overhead. The simulation is based on a description of the micro-architecture of the processor, that can be either accurate or statistically approximated.

In the accurate simulation mode, stalls length and operations latency are dynamically computed by using a processor model functionally equivalent to the target one. In the statistical approximation mode, the stalls length as well as the operations latency are statically averaged values derived from an accurate off-line profiling of the application. In both cases, instruction-level power is accurately calculated by looking at the constituent blocks of the processor by taking into account the effect that the single instruction can produce on them. This level of detail cannot be achieved by using a simple black-box instruction-level energy model such as those presented in the literature so far.

Considering scalar processors, the model decomposes the energy associated with a single instruction into the energy contributions due to each processor macro-block involved into its execution. These macro-blocks are the pipeline stages, the inter-stage connections and the instruction and data caches.

For a VLIW architecture, the analysis can be pushed further by decomposing the energy contributions of a single macro-block in the energy contribution of the macro-block functional units that work separately on each operation of the long instruction (bundle). This property, introduced as the spatial additive property, has been fundamental to deal with the complexity of the instruction-level power model for VLIW cores, which grows exponentially with the number of possible operations in the ISA. The proposed decomposition provides a way to create a mapping between bundles and micro-architectural functional units involved during the simulation. This instruction-to-unit mapping is used to retrieve energy information for each unit that is elaborated together with the stall and latency information to obtain run-time power estimates.

The paper is organized as follows. The overall architecture of the system is described in Section 2. Section 3 introduces the general instruction-level energy estimation methodology for pipelined processors, while Section 4 describes how the proposed model can be extended to VLIW architectures. Experimental results demonstrating the validity of the proposed approach applied to a simplified VLIW architecture are reported in Section 5. Finally, Section 6 concludes the paper by outlining some research directions originated from this work.

2 Target System Architecture

A power-oriented methodology at system-level is mandatory for architectural exploration during the first phases of a HW/SW co-design flow. The methodology should be tightly related to the characteristics of the system architecture, mainly in terms of target processors, memory sub-system, system-level buses and co-processors.

Figure 1 shows the block diagram of our target system architecture, which is a shared memory multi-processor system that can be implemented by using either the System-On-Chip approach or the multi-chip approach. The system includes one or more processors, the instruction caches (I-caches), the data caches (D-caches), the memory controller, the main memory, the I/O controllers, the peripheral units, and the co-processors to support specific applications (such

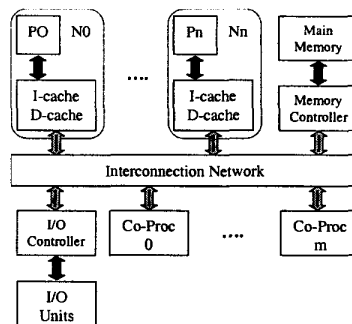


Figure 1: The target system architecture.

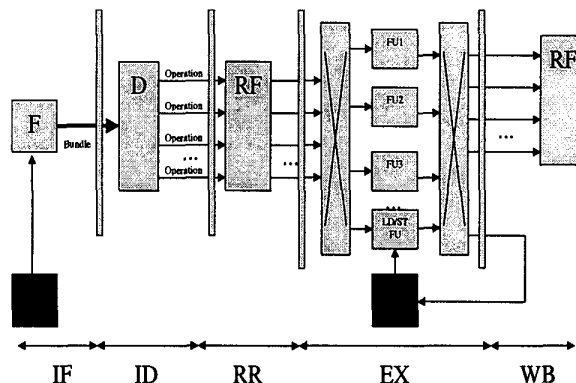


Figure 2: Architecture of a node including the VLIW pipelined processor, the I-cache and the D-cache.

as MPEG encoding).

All these building blocks are connected through address, data and control buses implemented by using different topologies. HW/SW communication occurs either on the sub-system-level buses, such as the processor-to-cache buses (which have been coloured in dark grey in Figure 1) or on the system-level buses (which have been coloured in light grey in Figure 1). A modified version of the proposed architecture may include a D-cache for each node, but a common I-cache for the different nodes.

The target processor architecture included in each node is a pipelined load/store VLIW core processor (see Figure 2). The processor pipeline is composed of five stages ²:

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

In our VLIW architecture, we assume that the branch prediction scheme adopts a *predict-untaken* scheme, i.e., the processor continues to fetch sequential instructions from memory until a branch, decoded in the ID stage, is taken. A taken branch nullifies the instruction in the fetch stage by

²The pipeline has been modified with respect to the MIPS architecture presented in [8]; in particular, the memory access stage has been included in the EX stage.

substituting it with a NOP and redirects the execution flow to the branch target.

Furthermore, the considered VLIW architecture does not support predication, because this mechanism could waste power while executing instructions that will not be committed back to the architectural state.

3 Energy Model for Pipelined Architectures

The existing approaches to software-level power estimation do not analyze the contribution of each pipeline stage. In fact, characterization of a single instruction is performed by considering the processor as a black box, without analyzing the power behavior during the flow of the instructions through the pipeline stages. This paper presents an extension of the work proposed in [2] addressing instruction-level power estimation based on an accurate characterization of the *data-path* of the pipeline stages. The proposed instruction-level energy model accounts for the energy contributions due to each pipeline stage, inter-stage connections and I- and D- caches.

The control blocks of the processor are not modeled explicitly because of their relative invariance in power consumption w.r.t. the type of instruction executed. For these blocks, accurate power estimates can be derived using power models developed for finite state machines [11].

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* of w_n can be split in two contributions. The first-order contribution is due to the preceding instruction w_{n-1} , while the second-order contribution is given by other instructions w_k present in the pipeline during the execution time of w_n . We consider here only the first-order contribution of the execution context, while the second order contribution is indirectly taken into consideration in terms of stall/latency cycles introduced during the execution of instruction w_n .

As basic assumption, we consider the data stationary instruction code type, which has been proposed for the compilation of ASIP cores [12]. Furthermore, we assume 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.

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 associated with w_n as follows:

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

where A_s is the energy consumed by stage s when executing instruction w_n after w_{n-1} , $I(w_n)$ is the energy consumed by the connections between pipeline stages (inter-stage connections) and $\Gamma(w_n)$ is the energy consumed by the I- and D- caches. In turn, 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 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. U_s depends on the current instruction word executed and on the preceding one in the trace; it represents the contribution of the individual stage only and therefore can be evaluated independently. U_s includes an *energy base cost*, independent of instruction type and switching activity; it mainly depends on the clock buffers contained in the processor sequential elements (registers and latches) which imply an energy cost at each clock cycle, independently of the switching activity of the register outputs.

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:

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

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 profiling a given application on a cycle-by-cycle basis. Existing simulators (such as Trimaran [13]) 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.

A special remark concerns the overall energy of the RR stage. We propose to approximate A_{RR} with the following expression:

$$\bar{R}_{RR} * (1 + m_{RR}(w_n) * p_{RR}(w_n)) \quad (4)$$

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 energy consumption is influenced by inter-stage effects (such as writes generated by the WB stage) which are very complex to model. Obviously, by considering all register file energy consumption here, WB energy parameters must account only for the remainder of the WB logic (e.g., stage buffers).

The model for the energy consumption due to inter-stage connections $I(w_n)$ is given by:

$$I(w_n) \approx 0.5 * V_{dd}^2 * f_{clk} * C_L * \alpha_{ave} \quad (5)$$

where V_{dd} is the power supply, f_{clk} is the pipeline clock frequency, C_L is the capacitive load of the inter-stage buses and α_{ave} is the switching activity of the inter-stage buses averaged with respect to the number of pipeline stages and the number of bit per stage N_s . Term α_{ave} can be estimated by the following expression:

$$\alpha_{ave} = \frac{\sum_{s \in S} H_s(\mathbf{B}^{(n)}, \mathbf{B}^{(n-1)}) / N_s}{S} \quad (6)$$

where H_s is the Hamming distance between the inter-stage bus lines B at time n and $n - 1$.

The cache energy contribution $\Gamma(w_n)$ depends on the cache hit and miss ratio of both the I- and D-caches. During

execution of w_n , the I-cache is accessed once, while the number $\delta(w_n)$ of D-cache accesses depend on instruction type and pipeline structure. In our pipeline model, the value of $\delta(w_n)$ can be either 0 or 1 depending on the instruction type. The cache energy contribution $\Gamma(w_n)$ can be approximated as:

$$\Gamma(w_n) \approx (H_I * E_{I,hit} + (1 - H_I) * E_{I,miss}) + \delta(w_n) * (H_D * E_{D,hit} + (1 - H_D) * E_{D,miss}) \quad (7)$$

where H_I , $E_{I,hit}$ and $E_{I,miss}$ are, respectively, the hit ratio, the average hit energy and the average miss energy consumption per instruction of the I-cache. Similarly for the D-cache. The average hit and miss energy for caches have been derived by applying the model proposed in [14].

4 Energy Model for Pipelined 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 Figure 2). Unlike a superscalar processor, which can be considered an *atomic instruction architecture* [15], 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. Thus the stall cycles in pure VLIW processors are only due to control hazards.

As already noted in [7], the challenge of each 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, parameter $U_s(w_n|w_{n-1})$ takes into account only the inter-instruction effect between adjacent instructions and can be maintained in a bi-dimensional array whose size is proportional to the number of instructions in the ISA. In the case of a RISC machine this problem can be solved by clustering instructions [4], thus reducing the parameter's array length.

In contrast, for VLIW architecture, the power model should account for all possible combinations of operations in a bundle, thus the problem complexity grows exponentially with the bundle size. As for the RISC case, instruction clustering can be effective to reduce the problem complexity by trading off power estimation accuracy and spatial complexity.

In this paper, we propose to reduce the problem complexity by introducing the *temporal and spatial additive* property of VLIW pipelined architectures. The *temporal additive* property is the same property stated in the previous section for pipelined processors. The *spatial additive* property states that the energy consumption of a single pipeline stage corresponds to the sum of the energy contributions due to the single operations in the current bundle, given the operations in the previous bundle. This property enables us to decompose the energy associated with a pipeline stage into the sum of the energy contributions due to each functional unit active in the stage. In our approach, we apply this property not only to the EX stage, for which this concept seems to be more intuitive, but to all pipeline stages. An *energy base cost*, common to all executable bundles, has also been considered, as introduced in the pipeline energy model shown in the previous section.

Let us consider a simplified VLIW architecture with an ISA composed of three operations (ADD, MUL and NOP), a set of two FUs (an adder and a multiplier) and a bundle size equal to two. We derive the model to estimate the energy consumption of the EX stage while executing a $\langle ADD, NOP \rangle$ bundle that follows a $\langle MUL, NOP \rangle$ bundle. The first component is the energy base cost: this contribution can be estimated by measuring U_{EX} during an execution of NOPs (namely, $U_{EX}(NOPs|NOPs)$). The second contribution is due to the behavior of the ADD operations in the current and in the previous bundle. In our example, the current bundle uses the adder, while the previous one does not use it. We indicate this case as $(ADD|\emptyset)$ and the energy contribution associated to it as $\nu_{EX}(ADD|\emptyset)$. The third contribution is due to the behavior of the MUL operations in the current and in the previous bundle. In this case, we see that the previous bundle used the multiplier, while the current one does not use it. We indicate this case as $(\emptyset|MUL)$ and the energy consumption associated to it as $\nu_{EX}(\emptyset|MUL)$. We assume these two cases can be considered orthogonal to each other, thus the spatial additive property holds. The total energy consumption:

$$U_{EX}(\langle ADD; NOP \rangle | \langle MUL; NOP \rangle)$$

can be expressed as the sum of the three contributions described above:

$$U_{EX}(NOPs|NOPs) + \nu_{EX}(ADD|\emptyset) + \nu_{EX}(\emptyset|MUL)$$

In the general case, given a sequence of bundles $(w_n|w_{n-1})$ and a stage s , the energy associated with w_n depends on the behavior of each operation d included in one or both the current and the previous bundle: $d \in (w_n \cup w_{n-1})$. The presence of operation d in the current bundle and/or in the previous one can be summarized in three cases:

- $(d|\emptyset)$: $d \in w_n, d \notin w_{n-1}$;
- $(\emptyset|d)$: $d \in w_{n-1}, d \notin w_n$;
- $(d|d)$: $d \in w_n, d \in w_{n-1}$.

Let us indicate one of the above three cases, as $\Theta(d) \in \{(d|\emptyset), (\emptyset|d), (d|d)\}$. The additive property enables us to affirm that:

$$U_s(w_n|w_{n-1}) \approx U_s(NOPs|NOPs) + \sum_{\forall d \in (w_n \cup w_{n-1})} \nu_s(\Theta(d)) \quad (8)$$

where:

- $U_s(NOPs|NOPs)$ is the energy base cost estimated by executing a sequence of bundles composed of NOPs;
- ν_s is the energy contribution due to $\Theta(d)$.

The additive property can also be applied to the per stage stall energy consumption S_s :

$$S_s(w_n) \approx S_s(NOP) + \sum_{\forall d \in w_n} \sigma_s(d) \quad (9)$$

where $S_s(NOP)$ is the stall energy consumption of a NOP and σ_s is the additive energy contribution due to the stalls of operation d .

In this way, the problem complexity has been reduced from $O(N^k)$ to $O(N)$ (where N is the number of operations in the ISA and k is the number of operations in the bundle).

Globally, if we assume temporal and spatial properties hold for the pipeline stages as well as for the operations in the stage, the average energy associated with w_n is given by:

$$E(w_n) \approx \sum_{v_s \in S} \{U_s(w_n|w_{n-1}) + m_s(w_n) * p_s(w_n) * S_s(w_n)\} \quad (10)$$

where U_s is given by equation (8), $m_s(w_n)$ is the typical number of stall/latency cycles occurred while executing w_n , $p_s(w_n)$ is the stall/latency probability while executing w_n and S_s is given by equation (9).

5 Experimental Results

To validate the proposed energy model, we considered a simplified VLIW architecture (*SVLIW* for brevity) provided with data forwarding and operation latency equal to one cycle with in-order issue and in-order completion. The *SVLIW* architecture (shown in Figure 3) can execute a two operation bundle $\langle o_1; o_2 \rangle$ without any stall. The operations taken into account in our experiments are *ADD* and *MUL* functional operations and *NOP* operations. The processor architecture is composed of:

- A register file of 64 registers with 4 read-ports and 2 write-ports;
- Two FUs (adder and multiplier) each one statically linked to 2 register file read-ports and 1 write-port;
- A direct mapped 16-bundle I-cache with 1-bundle line size and 100% hit rate³.

The *SVLIW* processor has been described in VHDL, synthesized as multi-level logic by using Synopsys Design Compiler and mapped into a commercial 0.35 μ m and 3.3V technology library. Energy consumption estimates have been obtained with Synopsys Design Power by assuming a 50 MHz clock frequency. The Design Power tool has been used to obtain a breakdown of the power values into their single contributing elements with a user-defined granularity.

In this section, all reported results are energy values normalized to a clock cycle of 20 ns and given in [mW].

Our main goal is to demonstrate that the proposed instruction-level model achieves results very close to the black-box power estimates while providing major efficiency due to its inherent simplicity.

A first set of experiments have been carried out to demonstrate the validity and the accuracy of the temporal additive property for the pipeline stages of the *SVLIW* processor executing a sequence of bundles of the same type, i.e. $w_n = w_{n-1} = \langle o_1, o_2 \rangle$, where $o_1, o_2 \in \{NOP, ADD, MUL\}$. Due to resource constraints of the *SVLIW* processor, we do not consider bundles $\langle ADD, ADD \rangle$ and $\langle MUL, MUL \rangle$, while for the pairs of energetically and functionally equivalent bundles (e.g. $\langle ADD; NOP \rangle$ and $\langle NOP; ADD \rangle$), a single case has been studied. Globally four cases have been analyzed: $\langle NOP; NOP \rangle$, $\langle ADD; NOP \rangle$, $\langle MUL; NOP \rangle$ and $\langle ADD; MUL \rangle$. Table 1 reports the normalized energy consumed by each stage s of the pipeline, where 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

³ The considered I-cache size is small w.r.t. commercial VLIW processors, thus the related power contribution reported in Tables 1 and 4 is quite reduced compared to the *RR* stage.

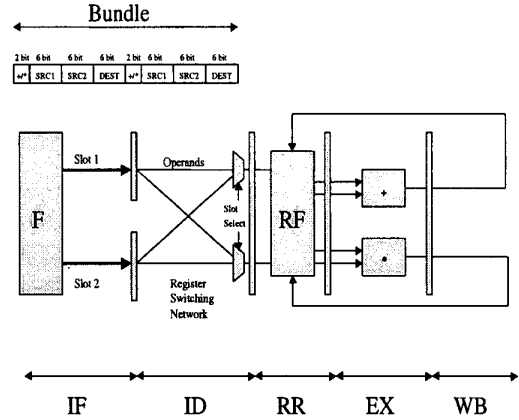


Figure 3: The simplified VLIW Pipelined Processor Architecture.

RR stage. Row $|\tilde{E}(w_n)|$ reports the normalized energy estimated by applying equation (1), while row $|E(w_n)|$ reports the average normalized energy estimated by considering the processor as a black-box executing a trace of identical bundles (computed by Design Power). The percentage error of $|\tilde{E}(w_n)|$ with respect to $|E(w_n)|$ is equal to 0.34% for $\langle NOP; NOP \rangle$, 5.61% for $\langle ADD; NOP \rangle$, 5.05% for $\langle MUL; NOP \rangle$, and 5.99% for $\langle ADD, MUL \rangle$ respectively.

The second set of experiments aims at providing the normalized energy consumed by the *SVLIW* processor by using the temporal additive property. Each possible pair of bundles: $(w_n|w_{n-1})$, has been considered, as reported in Table 2. Each column of Table 2 reports also the mean (or *unconditioned*) normalized instruction energy as well as standard deviation. The results show that the standard deviation is less than 3.68.

A third set of experiments has been carried out to demonstrate the validity of the temporal and spatial additive assumptions for the *SVLIW* processor for a selected set of bundle pairs $(w_n|w_{n-1})$, shown in Table 3. For brevity, we used the following shortcuts to indicate bundles: **A** for $\langle ADD; NOP \rangle$, **M** for $\langle MUL; NOP \rangle$, **AM** for $\langle ADD; MUL \rangle$ and **N** for $\langle NOP; NOP \rangle$. The column $|\tilde{E}|$ represents the energy values estimated by applying equation (10) (i.e., by using both temporal and spatial additive properties), while column $|E|$ represents the energy values estimated by considering the black-box processor model. Note that the percentage error is at worst 1.34%.

A fourth set of experiments has been carried out to validate equation (1) in the presence of stalls. Table 4 shows the percentage errors between effective and estimated power consumption of three types of traces, where control hazards have been inserted to produce stalls ($p_s = 1, m_s = 1Vs$). Note that in a VLIW processor stalls are reduced with respect to a scalar or superscalar processor, being due only to control hazards. For this reason, the percentage errors shown in Table 4 do not represent a main concern.

As a conclusion, the experimental results show that we can advocate the proposed energy model as a methodology for exploring the architectural trade-offs in the processor design from the energy perspective.

	w_n $o_1=NOP$ $o_2=NOP$ [mW]	w_n $o_1=ADD$ $o_2=NOP$ [mW]	w_n $o_1=MUL$ $o_2=NOP$ [mW]	w_n $o_1=ADD$ $o_2=MUL$ [mW]
I-cache	1.59	1.79	1.78	1.99
IF	1.81	1.81	1.81	1.81
ID	1.10	1.40	1.40	1.71
RR	18.47	32.84	32.15	45.05
EX	3.90	6.20	37.91	40.15
WB	2.14	2.34	2.34	2.54
Interconnect	7.77	10.32	10.60	13.15
$ E(w_n) $	36.77	56.70	87.99	106.39
$ E(w_n) $	36.64	53.52	83.54	100.02
Perc. Error	0.34%	5.61%	5.05%	5.99%

Table 1: Normalized energies consumed by the SVLIW processor executing a sequence of bundles of the same type $w_n = w_{n-1} = \langle o_1, o_2 \rangle$.

w_{n-1}	w_n $o_1=NOP$ $o_2=NOP$	w_n $o_1=ADD$ $o_2=NOP$	w_n $o_1=MUL$ $o_2=NOP$	w_n $o_1=ADD$ $o_2=MUL$
$o_3=NOP$ $o_4=NOP$	36.77	57.05	88.35	107.10
$o_3=ADD$ $o_4=NOP$	41.13	56.70	92.80	106.76
$o_3=MUL$ $o_4=NOP$	41.41	61.79	87.99	106.74
$o_3=ADD$ $o_4=MUL$	45.76	61.45	92.44	106.39
$Mean[E(w_n)]$	41.27	59.25	90.39	106.75
$Std[E(w_n)]$	3.68	2.74	2.58	0.29

Table 2: Normalized energies consumed by the SVLIW processor executing the possible pairs of bundles $(w_n | w_{n-1})$.

6 Concluding Remarks

The overall goal of our work is to define a power-oriented system-level optimization methodology suitable for embedded systems based on VLIW cores. We presented a new instruction-level energy estimation method which accounts for several architectural parameters such as number and topology 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 complexity associated with this type of processor architectures.

Up to now, our model supports sequential code execution. Our work is now oriented to extend the model to support control instructions (such as jumps, calls and returns) and to estimate the energy cost of the accesses through the memory hierarchy. The extended model aims at the energy assessment of commercial VLIW processors executing real instruction traces.

References

- [1] A. Chandrakasan and R. Brodersen, "Minimizing Power Consumption in Digital CMOS Circuits," *Proc. of IEEE*, 83(4), pp. 498-523, 1995.
- [2] M. Sami, D. Sciuto, C. Silvano and V. Zaccaria, "Instruction-Level Power Estimation for Embedded VLIW Cores," *CODES 2000: Eighth Int. Workshop on Hardware/Software Codesign*, pp. 34-38, San Diego, CA, May 3-5, 2000.
- [3] V. Tiwari, S. Malik and A. Wolfe, "Power Analysis of Embedded Software: A First Step Towards Software Power Minimization," *IEEE Trans. VLSI Systems*, pp. 437-445, Dec. 1994.

$w_n w_{n-1}$	$\Theta(ADD)$	$\Theta(MUL)$	$ E $ [mW]	$ E $ [mW]	Perc. Error
AM N	ADD \emptyset	MUL \emptyset	108.63	107.23	-1.31%
AM A	ADD ADD	MUL \emptyset	108.29	106.87	-1.33%
AM M	ADD \emptyset	MUL MUL	108.27	106.85	-1.33%
AM AM	ADD ADD	MUL MUL	107.93	106.50	-1.34%
N AM	\emptyset ADD	\emptyset MUL	45.76	45.90	0.31%
A AM	ADD ADD	\emptyset MUL	61.34	61.56	0.36%
M AM	\emptyset ADD	MUL MUL	92.35	92.56	0.23%
A M	ADD \emptyset	\emptyset MUL	61.69	61.91	0.36%
M A	\emptyset ADD	MUL \emptyset	92.71	92.93	0.24%

Table 3: Normalized energies for bundle pairs $(w_n | w_{n-1})$ considered to validate the temporal and spatial additive properties.

Block	Trace of ADD [mW]	Trace of MUL [mW]	Trace of < ADD; MUL > [mW]
I-Cache	1.69	1.68	1.79
IF	1.81	1.81	1.81
ID	1.25	1.25	1.40
RR	25.66	25.31	31.76
EX	5.05	20.90	22.02
WB	2.24	2.24	2.34
Interconnect	9.04	9.19	10.46
$ E(w_n) $	46.74	62.41	71.58
$ E(w_n) $	47.39	71.46	81.85
Perc. Error	-1.38%	-12.67%	-12.54%

Table 4: Normalized energies consumed by the SVLIW processor to validate the model when stalls occur

- [4] M. T.-C. Lee, V. Tiwari, S. Malik and M. Fujita, "Power Analysis and Minimization Techniques for Embedded DSP Software," *IEEE Trans. VLSI System*, pp. 123-135, Mar. 1997.
- [5] J. T. Russel and M. F. Jacome, "Software Power Estimation for High Performance 32-bit Embedded Processors," *Proc. of ICCD '98*.
- [6] D. Sarta, D. Trifone and G. Ascia, "A Data Dependent Approach to Instruction Level Power Estimation," *Low Power Design, 1999 Proc. IEEE Alessandro Volta Memorial Workshop on*, pp. 182-190, Como, Italy, Mar. 1999.
- [7] 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*.
- [8] J. Hennessy and D. A. Patterson, "Computer Architecture: A Quantitative Approach," Morgan Kaufmann Publishers, San Mateo, CA, Second Edition, 1996.
- [9] K. Masselos et al. "Interaction between Sub-word Parallelism Exploitation and Low Power Code Transformations for VLIW Multi-media Processors," *Low Power Design, 1999 Proc. IEEE Alessandro Volta Memorial Workshop on*, pp. 52-60, Como, Italy, March 1999.
- [10] H. Mehta et al. "Techniques for Low Energy Software," *Proc. of ISLPED-97: ACM/IEEE Int. Symposium on Low Power Electronics and Design*, pp. 72-75, Monterey, CA, August 1997.
- [11] E. Macii, "Sequential Synthesis and Optimization for Low Power", in *Low Power Design in Deep Submicron Electronics, NATO ASI Series, Series E: Applied Sciences*, Vol. 337, Kluwer Academic Publisher, pp. 321-353, 1997.
- [12] G. Goossens et al. "Embedded Software in Real-Time Signal Processing Systems: Design Technologies," *Proc. of IEEB*, Vol. 35, No. 3, March 1997.
- [13] Trimaran Home Page, <http://www.trimaran.org>
- [14] M. Kamble and K. Ghose, "Analytical Energy Dissipation Models for Low Power Caches," *Proc. of ISLPED-97: ACM/IEEE Int. Symposium on Low Power Electronics and Design*, pp. 143-148, Monterey, CA, August 1997.
- [15] H. Corporaal, "Microprocessor Architectures from VLIW to TTA," John Wiley and Sons, Chichester, England.