

# Reducing the complexity of instruction-level power models for VLIW processors

A. Bona · M. Sami · D. Sciuto · C. Silvano · V. Zaccaria ·  
R. Zafalon

Received: 31 August 2005 / Revised: 4 May 2006 / Accepted: 4 May 2006  
© Springer Science + Business Media, LLC 2006

**Abstract** Aim of this paper is to propose a high-level power exploration framework based on an instruction-level energy model for VLIW (Very Long Instruction Word) architectures. More specifically, the present paper deals with the reduction of the complexity of the energy model of  $K$ -issue VLIW processors from exponential with respect to the number of operations within the Instruction Set  $O(|ISA|^K)$  to quadratic ( $O(K * |ISA|^2)$ ). The complexity of the energy model has been further simplified by automatically clustering the operations in the ISA with respect to their average energy. Globally, the proposed approach reduces the complexity of the characterization problem for a  $K$ -issue VLIW processor to quadratic ( $O(K * |C|^2)$ ) with respect to the number of operation clusters. In this way, a more efficient characterization of the VLIW core power consumption can be achieved, while preserving the accuracy of the power estimates. The proposed model has been further extended to provide early power figures and energy/performance trade-offs for multi-cluster VLIW architectures composed of multiple data-path units and a single instruction cache control unit. The proposed high-level power estimation methodology has been applied to the Lx 4-issue VLIW pipelined processor provided by STMicroelectronics.

**Keywords** Low-power design · Power modeling · VLIW architectures

## 1. Introduction

High-level power estimation represents a fundamental phase during the design of a complex embedded system. In fact it enables the possibility to perform early power and speed trade-offs which can be of critical importance for market success.

---

A. Bona · V. Zaccaria  
STMicroelectronics N.V., Manno, Switzerland

R. Zafalon  
STMicroelectronics, Agrate B. (MI), Italy

M. Sami · D. Sciuto · C. Silvano (✉)  
Politecnico di Milano, Milano, Italy, e-mail: silvano@elet.polimi.it

Many research works in the area of high-level power estimation appeared in literature in recent years. Particular attention has been devoted to the problem of instruction-level power estimation of high-end processors, where pipelining and Instruction-Level Parallelism (ILP) have to be accounted for.

The overall goal of this work is to propose a methodology for the definition of an instruction-level energy estimation framework for VLIW processors. 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 *Very Long Instruction Word*. For VLIW processors, traditional instruction-level power modelling poses new challenges, since the number of long instructions that can be created by combining  $|ISA|$  operations (where  $|ISA|$  represents the cardinality of the Instruction Set) into a  $K$ -wide VLIW is  $|ISA|^K$ . It becomes evident that the complexity of applying traditional instruction-level methods [1, 2] to VLIW processors might be prohibitively high, since they require the storage of the energy parameters for each pair of instructions. Starting from our work proposed in [3], we introduced the temporal and spatial additive properties to reduce such a complexity.

The approach presented in this paper aims at extending and integrating into a comprehensive power estimation and optimisation framework our previous work [3], targeting an instruction-level energy model to evaluate and to optimize the energy consumption associated with a software execution on a pipelined VLIW core. More specifically, the present paper deals with the reduction of the complexity of the energy model of  $K$ -issue VLIW processors to quadratic with respect to the number of operations within the instruction set ( $O(K * |ISA|^2)$ ), while a black-box model would have a complexity of  $O(|ISA|^K)$ . This leads to a consistent reduction of the characterization effort, since the necessary number of experiments is reduced exponentially.

In [4], we further simplified the complexity of the energy model by introducing a methodology to automatically cluster the operations in the Instruction Set of a VLIW processor with respect to their average energy cost, to converge to an effective design of experiments for the actual characterization task. The main advantage of the proposed approach to cluster the operations in the  $|ISA|$  is the complexity reduction of the characterization effort from  $O(|ISA|^2)$  to  $O(|C|^2)$ , where  $C$  is the number of clusters.

The main contribution of the present paper with respect to [4] consists of proposing an extension of our instruction-level power model to provide early power figures and energy/performance trade-offs to explore *multi-cluster* architectures based on the same target VLIW processor. A multi-cluster VLIW architecture can further improve Instruction Level Parallelism (ILP) by introducing a set of parallel data-path units (clusters) but a single Program Counter and an unified Instruction Cache, so that the clusters can run in lockstep. The proposed architectural exploration is based on two different kinds of analysis: a *non-constrained analysis*, based on the energy-delay product metric, and a *constrained analysis*, that reports the energy/delay Pareto curves of several multi-cluster system configurations. These exploration results can support the designer to assess the configuration that optimizes both the performance and the energy consumption in the early stages of the design of a multi-cluster VLIW architecture.

The long term goal of our research is to provide a high-level power exploration framework to support power optimization techniques operating at the compilation-level and software-level.

The paper is organized as follows. The current state of the art on instruction-level power estimation and design space exploration is presented in Section 2. An overall description of the customization applied to the proposed VLIW power model is given in Section 3, while Section 4 introduces the automatic methodology to cluster operations with the same energy

behavior. The proposed extension of the power model to multi-cluster VLIW processors is described in Section 5. Some experimental results demonstrating the applicability of the proposed general estimation and optimization approach to the industrial Lx VLIW processor are reported in Section 6, while Section 7 shows a possible extension of the model to support power modeling and exploration of the multi-cluster version of the Lx VLIW processor. Finally, Section 8 contains some concluding remarks and outlines some future research directions originated from this work.

## 2. Background on instruction level power analysis and design space exploration

In general, software power estimation is based on a functional simulation of the instruction set of the processor. During the instruction set simulation, 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 has started only recently. 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.* [1, 2, 5] 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 *inter-instruction effects* occur, whose power cost is not taken into account if only the base energy costs are considered. The first type of inter-instruction 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 inter-instruction effects are related to resource constraints that can lead to stalls (such as pipeline stalls and write buffer stalls) and cache misses.

Globally, the model in [1, 2] 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 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 penalty due to resource constraints  $k$  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 is presented in [2], 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 re-ordering to exploit the power characteristics of each instruction.

The instruction-level power model proposed in [6] 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.

Another attempt to perform instruction level power estimation is proposed in [7], where the authors consider several inter-instruction effects as well as statistics of data values used within the processor. Although the developed power model is quite accurate, it lacks of general applicability, being developed only for a specific embedded processor.

In [8], 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*). As a matter of fact, the model reduces the spatial complexity proper of instruction-level power models.

For what concerns system-level power estimation and exploration, several methods have been recently proposed in literature targeting power-performance trade-offs from the system-level standpoint. Among these works, the most significant methods can be divided into two main categories: (i) system-level power estimation and exploration in general, and (ii) power estimation and exploration focusing on cache memories.

In the first category, the SimplePower approach [9] can be considered one of the first efforts to evaluate the different contributions to the energy budget at the system-level. As cited above, the SimplePower exploration framework includes a transition-sensitive, cycle-accurate datapath energy model that interfaces with analytical and transition sensitive energy models for the memory and bus sub-systems. The SimpleScalar estimation framework has been designed to evaluate the effects of some high-level algorithmic, architectural and compilation trade-offs on energy. Although the proposed system-level framework is quite general, the exploration methodology reported in [10] is limited to a search over the space of the following parameters: cache size, block buffering, isolated sense amplifiers, pulsed word lines and eight different compilation optimizations (such as loop unrolling).

The *Avalanche* framework presented in [11] simultaneously evaluates the energy-performance tradeoffs for software, memory and hardware for embedded systems. The approach is based on system-level and takes into consideration the interdependencies between the various system modules. The target system features the CPU, D-cache, I-cache and main memory, while the impact of buses is not accounted for. For this reason, the estimates are less accurate in advanced sub-micron technologies, for which routing capacitances play a significant role in the system's power consumption.

In [12], a system-level technique is proposed to find optimal configurations for low-power high-performance superscalar processors tailored to specific user applications. More recently, the Watch architectural-level framework has been proposed in [13] to analyze power vs. performance trade-offs with a good level of accuracy with respect to lower-level estimation approaches. Low-power design optimization techniques for high-performance processors have been also investigated in [14, 15] from the architectural and compiler standpoints.

A trade-off analysis of power/performance effects of SOC (System-On-Chip) architectures has been recently presented in [16], where the authors propose a simulation-based approach to configure the parameters related to the caches and the buses. The approach is viable only when it is possible to build a model either analytically or statistically (with a high level of accuracy) for the energy and delay behavior of the system's modules. Moreover the reported

results are limited to finite ranges of the following parameters: bus width, bus encoding, cache size, cache line size and set associativity.

A model to evaluate the power/performance trade-offs in cache design has been proposed in [17], where the authors discuss also the effectiveness of novel cache design techniques targeted for low-power (such as vertical and horizontal cache partitioning). An analytical power model for several cache structures has been proposed in [18]. The model accounts for technological parameters (such as capacitances and power supplies) as well as architectural factors (such as block size, set associativity and capacity). The process models are based on measurements reported in [19] for a 0.8  $\mu\text{m}$  process technology.

The previous analytical model of energy consumption for the memory hierarchy has been extended in [20] and [21] where the cache energy model is included in a more general approach for the exploration of memory parameters for low-power embedded systems. The authors consider as performance metrics the number of processor cycles and the energy consumption and show how these metrics can be affected by some cache parameters such as cache size, cache line size, set associativity and tiling size, and off-chip data organizations.

More recently, Lapinskii *et al.* [22] proposed a partitioned design space exploration algorithm for clustered VLIW data-paths. The approach consists of the division of the design space into a set of *slices*, where each slice contains the solutions with the same physical figures of merit such as power, clock rate and/or area. Each solution is associated with a slice by considering the number of functional units in the largest cluster, the number of clusters and the interconnect capacity. The authors then focused on finding the most representative elements within a slice in terms of latency of the schedules (i.e., initiation interval of the main software pipelined loop of the application) and the number of transfers among clusters. Even if the design exploration framework is targeted to find optimal solutions in terms of power/area/delay, no actual measured values are given by the authors.

### 3. VLIW power model

The overall goal of our work is to reduce the complexity of an instruction level power model suitable for VLIW (Very Long Instruction Word) embedded cores. A VLIW processor is a (generally) pipelined processor that can execute, in each clock cycle, a set of  $K$  explicitly parallel *operations*; this set of operations is statically scheduled to form a *Very Long Instruction Word* (or shortly *bundle*) and executed on a set of  $K$  parallel lanes.

Concerning VLIW processors, traditional instruction-level power modelling poses new challenges to be afforded, since the number of long instructions that can be created by combining  $|ISA|$  operations (where  $|ISA|$  represents the cardinality of the Instruction Set) into a  $K$ -wide VLIW is  $|ISA|^K$ .

The problem is increasingly complex since we have to associate an extra energy contribution to each possible sequence of  $y$  instructions to take into account inter-instruction effects of the  $y$ -th order. Thus, concerning  $K$ -issue VLIW processors we have that, considering the Instruction Set composed of  $|ISA|$  operations, and  $y$  inter-instruction effects, the complexity (i.e., the number of coefficients) of the instruction-level energy model is bounded by  $O(|ISA|^{K*(y+1)})$ .

In [3], we introduced the *temporal* and *spatial additive properties*, to deal with such a complexity. The temporal additive property enables the decomposition of the instruction energy into the pipeline stages of the processor. The spatial additive property enables the decomposition of the energy dissipated by the processor in the single energy dissipated by the processor parallel paths followed by the operations.

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 by the ISS (Instruction Set Simulator) to retrieve the energy information *on-the-fly* to eventually compute run-time power estimates.

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

$$\mathcal{W} = \langle \mathbf{w}_1, \dots, \mathbf{w}_{n-1}, \mathbf{w}_n, \dots, \mathbf{w}_N \rangle \quad (2)$$

where  $\mathbf{w}_n$  represents the  $n$ -th very long instruction composed of  $K$  parallel operations carried out by multiple and independent functional units working in parallel:

$$\mathbf{w}_n = [w_n^\Delta \dots w_n^k \dots w_n^K]^T \quad (3)$$

where  $w_n^k$  is the  $k$ -th operation (issued on the  $k$ -th lane of the processor) of the  $n$ -th bundle of the stream.

The energy model proposed in [3] assumes that the energy associated with  $\mathbf{w}_n$  is dependent on its own properties (e.g., class of the instruction, addressing mode and so on) as well as on its *execution context*, i.e., the former instruction  $\mathbf{w}_{n-1}$  and the additional stall/latency cycles introduced during the execution of the instruction. In particular, the energy model can be summarized as follows:

$$E(\mathcal{W}) \approx \sum_{\Delta \leq n \leq N} \sum_{\forall s \in S} \left[ U_s + \sum_{\forall k \in K} v_s(w_n^k | w_{n-1}^k) + m_s^n * p_s^n * S_s + l_s^n * q_s^n * M_s \right] \quad (4)$$

where:

- the term  $U_s$  is the base energy cost that represents the energy consumed by stage  $s$  during an execution of a sequence of bundles constituted entirely by NOPs;
- the term  $v_s(w_n^k | w_{n-1}^k)$  is the energy contribution due to the sequence of operations ( $w_{n-1}^k \rightarrow w_n^k$ ) on the same lane  $k$ ;
- the term  $m_s^n$  ( $l_s^n$ ) is the average number of additional cycles due to a data (instruction) cache miss during the execution of the  $\mathbf{w}_n$  in  $s$ ,  $p_s^n$  ( $q_s^n$ ) is the probability that this event occurs, and  $S_s$  ( $M_s$ ) is the energy consumption per stage of the processor modules that are active due to a data (instruction) cache miss.

Concerning branch prediction, we assume the VLIW processor adopts a static approach. The branch mispredictions have been considered as *NOPs*.

In the present paper, the complexity of the instruction-level energy model expressed by Eq. (4) has been simplified, while preserving its accuracy, by means of the following basic observations.

First, in a pipelined VLIW processor, it is quite difficult to isolate the energy contributions due to the different processor's modules associated with the different pipeline stages. This aspect could represent a limit for the application of Eq. (4) but, as a matter of fact, the model can be tailored to further support these cases. The term  $\sum_s U_s$  corresponds to the energy consumption of the core while it is executing NOPs and it can be substituted by the average base cost  $U$ .

Second, the sum  $\sum_s \sum_k v_s(w_n^k | w_{n-1}^k)$  can be substituted by a cost dependent only on the pair of instructions  $\sum_k v(w_n^k | w_{n-1}^k)$ , that corresponds to the energy consumption of the core while it is executing the same sequence of long instructions ( $\mathbf{w}_{n-1} \rightarrow \mathbf{w}_n$ ).

Third, concerning instruction and data cache misses, we assume that, after a transient state, the probabilities per stage ( $p$  and  $q$ ) and their penalties ( $m$  and  $l$ ) can be averaged for each instruction of the stream as follows:

$$\sum_s (m_s^n * p_s^n * S_s + l_s^n * q_s^n * M_s) \rightarrow (m * p * S + l * q * M) \quad (5)$$

where  $m(l)$  is the average data (instruction) cache miss penalty;  $p(q)$  is the average probability per stage and per instruction that a data (instruction) cache miss can affect one instruction;  $S$  ( $M$ ) is the average energy consumption of the *whole* processor during these events. We will show in the experimental results that this assumption does not involve a significant loss in the accuracy for the considered Lx VLIW processor.

In this way, Eq. (4) can be reduced to:

$$E(\mathcal{W}) \approx \sum_{\Delta \leq n \leq N} \left[ U + \sum_{\forall k \in K} v(w_n^k | w_{n-1}^k) + m * p * S + l * q * M \right] \quad (6)$$

The complexity of the model is now quadratic with respect to the number of operations within the instruction set ( $O(|ISA|^2)$ ), while a black-box model would have a complexity of  $O(|ISA|^K)$ . This leads to a significant reduction of the characterization effort, since the necessary number of experiments has been reduced exponentially.

In the next section, we show how clustering the operations in the *ISA* with respect to their average energy consumption enables a further reduction of the model in terms of a faster and more effective characterization of the core's power consumption, while preserving the model accuracy.

#### 4. Clustering of the operations in the *ISA*

To reduce the number of experiments to be generated during the characterization phase, in this section we introduce the cluster analysis on the operations of the *ISA*. The basic idea of the cluster analysis consists of grouping in the same cluster those operations with energy cost values close to each other. Although the idea to apply the cluster analysis to the instructions of the *ISA* to reduce the complexity of the power characterization process of a processor has been already proposed by Tiwari *et al.* [1, 2, 5], in this paper, for the first time, we apply this concept to a VLIW processor architecture, for which the operation clustering approach can provide further benefits in terms of the power characterization phase.

First of all, let us introduce some basic concepts used in the cluster analysis. 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 (in terms of register names or immediate values), the number of operation's pair to be considered in Eq. (6) would be too large to be characterized with a transistor-level or even gate-level simulation engine.

For example, considering the Lx processor [23], the *ISA* is composed of 70 operations, but the possible addressing modes for each operation would imply the power characterization of 6126 pairs of operations to completely define the model. In particular, this is due to the characterization of the matrix  $v$  (see Eq. 6), while the other parameters do not depend on the particular instruction executed. To give a rough idea of the time required for the characterization, a Linux-based PC at 2.4 GHz performs a gate-level power simulation of

the Lx core within 25 minutes. Given such a simulation time, we would need approximately 108 days to perform the whole characterization phase!

Several clustering algorithms appeared in literature so far (see [24, 25] for a survey); among these we have chosen the *k-mean clustering algorithm*, since it requires a lower computational cost with respect to other methods such as hierarchical clustering.

Given a population  $\Theta = \{e_1 \dots e_t \dots e_M\}$ , where each  $e_t$  is the energy consumption associated with a single operation  $t$ , the *k-mean clustering algorithm* tries to partition  $\Theta$  into a set of  $C$  clusters  $(\theta_1 \dots \theta_C)$  to minimize the following mean-square error:

$$\sum_{j=1}^C \sum_{i=1}^{n_j} (x_{i,j} - c_j)^2 \tag{7}$$

where  $n_j$  is the number of elements of cluster  $\theta_j$ ,  $x_{i,j}$  is the  $i$ -th element of cluster  $j$ ,  $c_j$  is the center of gravity of the  $j$ -th cluster and  $\bigcup_j C_j = \Theta$ .

Given the number  $C$  of classes in which the original population must be partitioned, the *k-mean clustering algorithm* attempts to split randomly the whole set into  $C$  subsets. Then, each element is moved into the subset  $j$  whose center of gravity is closest. This procedure iterates until the stopping criterion is met.

Although we expected that operations using the same functional unit of the processor are characterized by a very similar power cost and, thus, they fall in the same cluster, the process of generating the clusters is fully automatized. For each operation, we performed different experiments created by randomly varying register names and values. Based on the obtained energy measures, the *k-mean algorithm* automatically generates the operation clusters. Once the operations have been clustered into a set of  $\theta_1 \dots \theta_j$  clusters, we compute the matrix  $\nu$  of Eq. (6) in the following way:

$$\nu(w_n^k | w_{n-1}^k) = \begin{cases} E_i & \text{if } [(w_n^k = w_{n-1}^k) \text{ and } (w_n^k, w_{n-1}^k \in \theta_i)] \\ D_{i,j} & \text{if } [(w_n^k \neq w_{n-1}^k) \text{ and } (w_n^k \in \theta_i, w_{n-1}^k \in \theta_j)] \end{cases} \tag{8}$$

Without considering the switching activity due to data dependency, this decomposition tries to model the fact that when operations are equal ( $w_n^k = w_{n-1}^k$ ), they generate the least switching activity possible (first case). In the second case, when the operations are different, they generate an increased switching activity even if they are in the same cluster (accounted for by matrix  $D_{i,j}$ ).

The main advantage of the proposed operation clustering approach is that the complexity of matrix  $\nu$  has been reduced from  $O(|ISA|^2)$  to  $O(|C|^2)$ , where  $C$  is the number of operation clusters. Whenever a new processor generation is introduced (e.g. new ISA or new technology), the population  $\Theta$  composed of the energy consumption associated with each operation should be re-clustered. This phase requires a characterization effort with a complexity equal to  $O(|ISA|)$ . Then the model should be re-characterized by estimating the terms  $E_i$  and  $D_{i,j}$  to build the matrix  $\nu$  (see Eq. 8) requiring a complexity of  $O(|C|^2)$ , where  $C$  is the number of operation clusters.

To apply the clustering algorithm, one of the most relevant parameters to choose is the number of clusters  $C$ . Typically, this is a user-defined parameter and no automatic rules exist for that. As a general rule, the number of clusters is determined by a trade-off between the maximum standard deviation of the elements within the same cluster and the number of experiments that must be performed in order to characterize the model.

The minimum number of experiments is a typical cost function that must be reduced when performing the design of the experiments. In this work, we assume that the linear regression [26] is used to perform the characterization of the model. In this case, the minimum number of experiments is linearly dependent on the number of parameters of the model. Thus, the characterization cost has the following upper bound:

$$O\left(\frac{C \cdot (C - 1)}{2}\right) \quad (9)$$

where  $C$  is the number of clusters.

Section 6 reports the results of the trade-off analysis applied to the *ISA* of the Lx processor.

The instruction-level energy model developed so far, can be applied not only to provide data on the power consumption of the core, but also to statically optimize the power consumption associated with the embedded software applications. In [4], we proposed a *spatial scheduling* algorithm similar to the one presented in [27]. The basic idea consists of rescheduling the parallel operations within the same bundle (*spatially*) based on the power figures identified the power characterization phase, without increasing significantly the overall latency associated with the code. The main difference of the approach proposed here with respect to the approach proposed in [27] is that, in our case, the effectiveness of a specific scheduling policy is quantified based on the actual power model, while in [27] the goodness of a pair of scheduled operations is evaluated in terms of the Hamming distance between the coding of two adjacent operations carried out in the same processor's lane. The results of the proposed re-scheduling approach have been reported in [4]. Briefly, on a selected set of benchmarks, the measured average power consumption decreases by 17%, while the average energy consumption decreases only by 12%, since the re-scheduling technique causes in some cases a slight increase in the latency of the code. This is due to the fact that operation rescheduling could modify the results of the Lx instruction compression mechanism, leading in some cases to an increment of cache misses.

## 5. Extension of power model towards a multi-cluster VLIW architecture

In this section, we show how the proposed energy model can be extended to a multi-cluster version of the VLIW processor to support an early design space architectural exploration. This analysis is performed by taking into account both the energy consumption and the execution time of multi-cluster applications.

In our model, we assume a multi-cluster VLIW processor has a single instruction cache, one instruction fetch issue unit and a set of data-path units (clusters) that are identical, in terms of function units, to the single-cluster core. The role of the *instruction fetch mechanism* is to fetch a multi-cluster bundle from the instruction cache and to dispatch the operations to the appropriate clusters. We assume clusters have a separate register file and communicate through an inter-cluster communication mechanism that supports send and receive operations. Concerning the data cache, the specification for the multi-cluster architecture enables several possible configurations, among which we have chosen a single multi-port data cache shared among all clusters.

The choice of a single-cluster or a multi-cluster solution depends on the application and the power and performance constraints. In this section, we extend the instruction-level energy model proposed in Eq. (6) to support a multi-cluster processor to explore different architectural solutions with respect to the power and performance combined perspective.

This analysis can be done without a detailed description (e.g. an HDL netlist) of the processor, so it can be performed in the early stages of the design of an embedded system. This is of primary importance for the designer, who can assess the suitability of a multi-cluster architecture for the specific embedded application before the implementation phase.

In the specification of a multi-cluster architecture, we can identify two classes of processor blocks: blocks that are in common among all the clusters (*shared blocks*, such as the instruction cache), and blocks that are related to a single cluster (*non-shared blocks*), whose number varies linearly with the number of clusters. Here, we assume that the power consumption of the shared blocks does not increase with the number of clusters; this assumption results in a model whose predicted power consumption is a lower bound of the actual power consumption, but it is sufficiently accurate to derive relative estimates.

To extend the energy model described by Eq. (6), we introduced three additional parameters:

- $D$ : number of data-path units (clusters);
- $P_{ns}$ : percentage of power consumption related to blocks that are non shared ( $ns$ ) in the multi-cluster architecture;
- $P_s = (1 - P_{ns})$ : percentage of power consumption related to blocks that are shared ( $s$ ) in the multi-cluster architecture.

$P_{ns}$ ,  $P_s$  are estimated on the single cluster architecture by considering the power consumption of the non-shared and shared blocks with respect to the total power consumption. These two parameters can be composed to form a *scaling factor*  $\lambda(D) = (P_{ns}D + P_s)$  that multiplies the coefficients  $U$ ,  $M$  and  $S$ .

To simplify the computation of the model, an average value of the function  $v(w_n^k | w_{n-1}^k) = \bar{v}$  can be used. The term  $\bar{v}$  is weighted with the average  $ILP(D)$  of the stream (i.e., the number of syllables per instruction).  $ILP(D)$  depends, obviously, on the number of data-path clusters and on the capability of the compiler to exploit the availability of clusters. Note that the term  $\bar{v}$  is associated with the non-shared blocks of the system.

The model of Eq. (6) can be rewritten by considering the scaling factor  $\lambda(D)$  and  $ILP(D)$  as follows:

$$E(\mathcal{W}, D) \approx N \left[ \lambda(D)U + ILP(D)\bar{v} + \lambda(D)(m * p * S + l * q * M) \right] \quad (10)$$

Results of the application of the proposed multi-cluster VLIW energy model to the Lx processor are shown in Section 7.

## 6. Case study: the Lx VLIW architecture

The Lx architecture is a scalable and customizable processor technology [23] designed for both multimedia and signal processing embedded applications. The Lx processor is a statically scheduled VLIW architecture designed by STMicroelectronics and Hewlett-Packard, that supports a multi-cluster organization based on a single PC and a unified instruction cache. The basic processor (ST210) is a 4-issue VLIW core featuring four 32-bit integer ALUs, two  $16 \times 32$  multipliers, one load/store unit and one branch unit. The cluster also includes 64 32-bit GPRs an 8 1-bit Branch Registers. Lx has an in-order 6-stage pipeline and a very simple integer RISC ISA. For the first generation, the scalable Lx architecture is planned to span from one to four clusters (i.e., from 4 to 16 instructions issued per cycle).

Lx comes with a complete software tool-chain, where no visible changes are exposed to the programmer when the core is scaled and customized. The tool-chain includes a sophisticated *ILP* compiler technology (based on trace scheduling), GNU tools and libraries.

A gate-level netlist of the core processor has been used to perform the characterization needed for the power model presented in this work. The experiments have been carried out by using Synopsys VCS 5.2 and a set of PLI routines to elaborate toggle statistics over the whole gate-level netlist. PowerCompiler (by Synopsys) has been used to combine the toggle statistics with the power models of the standard cells library provided by STMicroelectronics, to compute the power figures of the entire core.

In the rest of this section, we will describe the characterization and validation flow of the power model. The validation phase aims at determining the model accuracy under realistic benchmarks.

### 6.1. Power characterization

The power characterization phase is, in turn, composed of two steps; the first one consists of clustering the operations in the ISA in order to reduce the number of experiments (needed by the linear regression), while maintaining a good accuracy level. The second step consists of generating the appropriate set of experiments necessary to perform the estimation of the coefficients in the power model.

The first step of the Lx ISA power characterization, is to group the operations in clusters by the  $k$ -mean clustering algorithm. For each operation  $o$ , we generate a set of assembly programs composed of repeated cycles of  $o$  operations (by varying register names and values) and we measured the energy consumption of the core at the gate-level. Then we applied the  $k$ -means clustering algorithm for several values of the number of clusters. In order to determine the most suitable number of clusters, we analyzed the minimum number of experiments that would be needed to characterize the model with  $C$  clusters and we performed a trade-off analysis with respect to the maximum variance within the clusters. As mentioned above, the minimum number of necessary experiments depends linearly on the number and size of the parameters involved in the model. In this case, the shape of the curve defining the number of experiments is quadratic, due to the quadratic dependence of the size of the matrix  $\nu$  with respect to the number of clusters  $C$ .

In general, a small number of operation clusters implies a high variance within them (i.e., poor accuracy of the model), though it also implies a small number of experiments during the regression. On the contrary, a large number of clusters implies good accuracy, but would result in a huge number of experiments.

Figure 1 shows the results concerning the maximum variance within the operations clusters and the number of experiments required to characterize each corresponding model. We can note how, for a number of clusters equal to 11, we can obtain a good tradeoff between the maximum variation that drops to 13% and the minimum number of experiments (that reaches 78). For this reason, we selected 11 clusters to characterize the Lx's power model.

The number of experiments found is a minimum value because, for each cluster, we can perform multiple experiments by varying the operations within the cluster, the register names and the immediate values.

To provide experimental evidence on the accuracy of the clustering algorithm applied on the instructions of the Lx ISA, Fig. 2 shows the mean and the variance of the energy values associated with each operation in the ISA and the clusters in which they have been grouped.

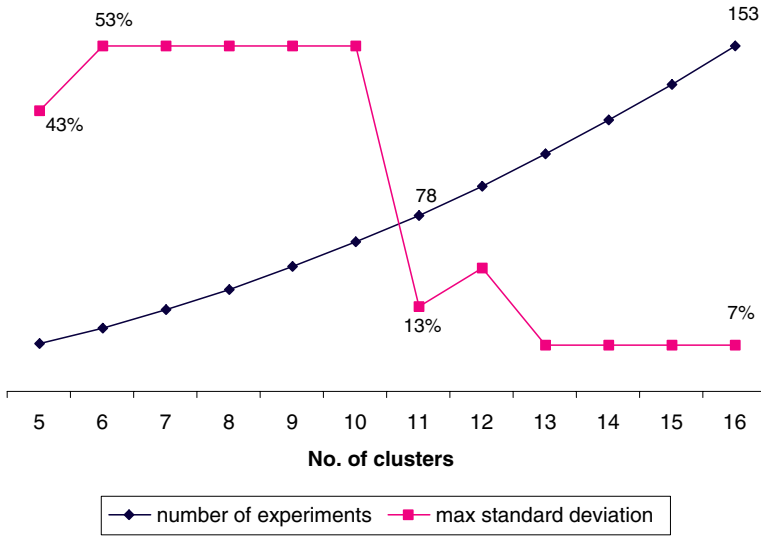


Fig. 1 Tradeoff between the number of experiments required to characterize the model and the maximum variance within each operation cluster

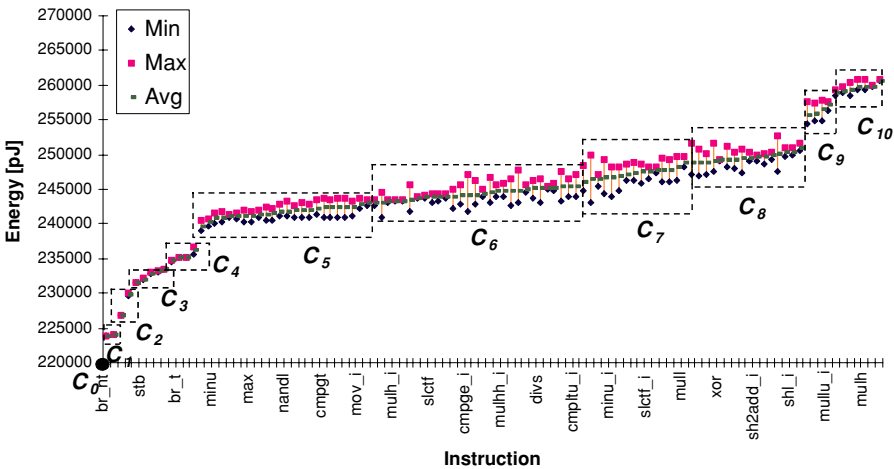
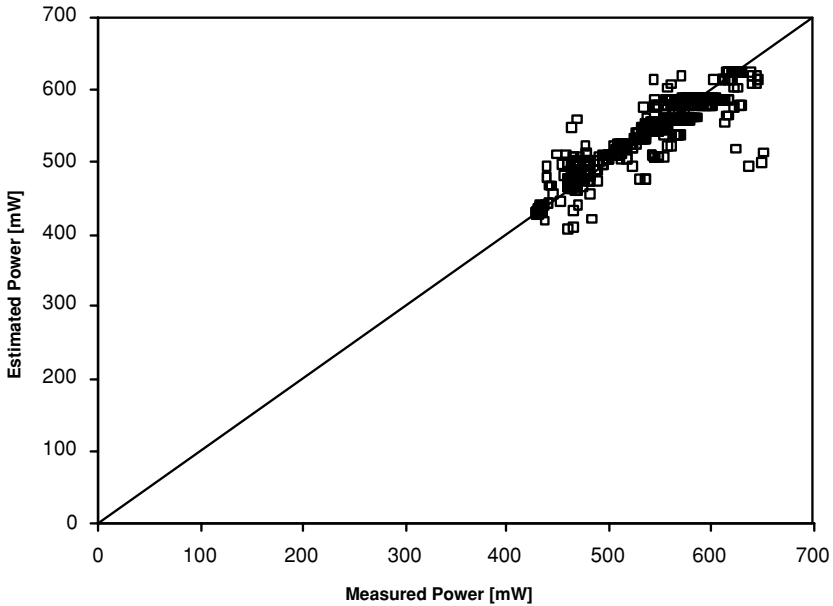


Fig. 2 Energy values associated with each operation and the corresponding operation clusters (where cluster C<sub>0</sub> corresponds to the NOP). Note that only a subset of operation names is shown

Once selected the number of clusters and, therefore, the number of coefficients in the model (i.e., the size of the  $v$  matrix), we realized a set of experiments in which each possible pairs of clusters could be generated more than once, by varying register names and values.

The experiments are characterized by a value of  $p$  and  $q$  (the data and instruction cache miss probabilities per instruction respectively) as small as possible to accurately characterize by regression only  $U$  and  $v(w_n^k | w_{n-1}^k)$ . The values of  $M$  and  $S$  (the energy consumed in a cycle during an I-cache and D-cache miss respectively) have been characterized by generating a set of few experiments with a large number of data and instruction cache misses, and then



**Fig. 3** Scatter plot of the measured average power values with respect to the estimated power values for the characterization experiments

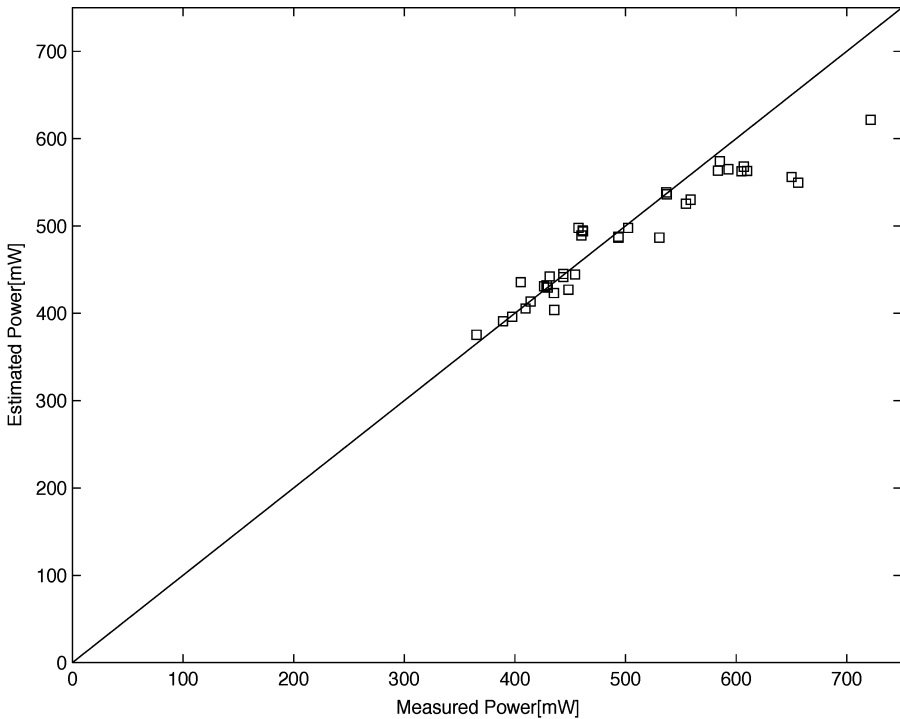
by measuring the power only during these events. Finally, the  $m$  and  $l$  miss penalties have been extrapolated by observing the behavior of the micro-architecture during these events.

Figure 3 shows the scatter plot of the predicted and measured power values, derived during the characterization process. The average absolute error is 2% while the standard deviation is 3%.

## 6.2. Model validation

The model has then been validated by using a set of benchmarks (obviously different from the benchmarks used to perform the characterization) derived from the Mediabench applications [28] (namely, G721 encoder and decoder, EPIC encoder and decoder, MPEG2), a set of finite impulse response filters, discrete cosine transforms and matrix elaboration algorithms. Figure 4 shows the agreement between the gate-level power values and the power values estimated with the proposed model. Here, the parameters for the instruction level model, i.e., probability per instruction per lane as well as data and instruction cache miss probabilities, have been gathered by RT-level simulation so they are the *actual* (or *ideal*) values. In these conditions the power model has shown, on the validation benchmarks, a average absolute error of 1.9% and a standard deviation on the error of 5.8%. The multiple correlation coefficient, that explains the percentage of the total variation explained by the model, has been computed as in [26] and is equal to:

$$\sqrt{\frac{SSR}{SSR + SSE}} = 90\% \quad (11)$$



**Fig. 4** Scatter plot of the measured average power values with respect to the estimated power values for the validation experiments

where SSR is the regression sum of squares and SSE is the sum of square error as defined in [26]. As a matter of fact, there are only three benchmarks whose error on the prediction is in the neighborhood of 10% presumably because of the high switching activity of the data consumed by instructions that is not captured by the model. The other benchmarks show much closer correlation. For high-level/instruction-level power macro models this can be considered an acceptable value in terms of accuracy. In addition, the hook-up of the model into an ISS can introduce some error due to the inaccuracy of the ISS in modelling the exact behavior of the lanes and cache misses.

## 7. Exploration of multi-cluster Lx VLIW architecture

The Lx is a customizable processor architecture suitable for a multi-cluster extension [23]. In this section, we show how the multi-cluster VLIW power model can be applied to the Lx architecture in order to perform an early design space exploration. This analysis is performed by taking into account both the power consumption and the execution time of multi-cluster applications. To show the effectiveness of the extended power model, in this section, we have considered some case studies based on the optimization of a selected set of multimedia applications on the Lx architecture.

The multi-cluster Lx architecture has a *single instruction cache*, a *single instruction fetch issue unit* and a set of identical Lx data-path units (clusters). Clusters have a separate Register

File and communicate through an *inter-cluster communication mechanism* that supports send and receive operations. Concerning the *data cache*, the multi-cluster Lx assumes a single multi-port data cache shared among all clusters.

The Lx processor modules have been divided in two classes: blocks that are in common among all the clusters (*shared blocks*, such as the instruction cache), and blocks that are related to a single cluster (*non-shared blocks*), whose number varies linearly with the number of clusters.

For multi-cluster Lx, we measured the percentage ( $P_s$  and  $P_{ns}$ ) of power related respectively to shared or non-shared blocks, resulting in the following values:

$$P_s = 0.337 \quad (12)$$

$$P_{ns} = 0.663 \quad (13)$$

As we will show in the following, the model of Eq. (10) can be used to early analyze a specific embedded application, so assessing the suitability of the multi-cluster implementation. To perform the design space exploration, we use our model combined with the CACTI configurable power and performance model of the caches [29]. The CACTI-based cache model has been re-targeted to the 0.25  $\mu\text{m}$  STMicroelectronics technology library. The authors of the CACTI Model claim an accuracy of 6% with respect to HSPICE. The parameters depending on the execution trace of the program (that is the number of instructions  $N$ ), the *ILLP* and the cache miss probabilities are estimated by using an experimental version of the Lx tool-chain, that allows the designer to compile and to simulate programs for the multi-cluster configuration.

The design space is characterized by the following sub-spaces:

- Number of data-path clusters: 1, 2, 4
- I-cache sizes: 2048, 4096, 8192, 16384, 32768, 65536 [byte]
- D-cache sizes: 2048, 4096, 8192, 16384, 32768, 65536 [byte]

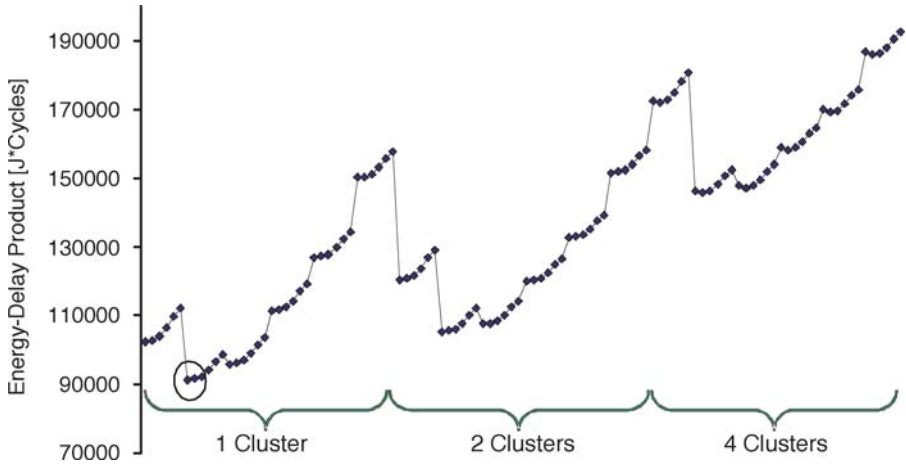
The other cache parameters have been set to the values of the single-cluster configuration: I-cache (block = 64 Bytes, associativity = 1), D-cache (block = 32 Bytes, associativity = 4).

### 7.0.1. Exploration based on energy-delay product

The first analysis consists of searching for the optimum configuration, that minimizes the *energy-delay* product (i.e., the product between the total energy consumed by the system and the number of execution cycles). In this case, this analysis is called *unconstrained* analysis, since no constraints on the maximum delay or the maximum energy of the system are set. Usually, this analysis leads to a configuration whose energy and delay represents a trade-off between the two solutions.

The design space has been explored over a set of 32 multimedia applications written in C language, such as a set of FIR filters, fast cosine transforms, AES encryption/decryption and a subset of the public domain *Mediabench* suite (ADPCM encode and decode, EPIC decode and encode, JPEG encode and decode, MPEG2 encode and decode).

Globally, over the 32 applications analyzed, only three of them benefit from the multi-cluster configuration. As an example, Fig. 5 shows the *ADPCM Encoder* behavior of the energy-delay product, when clusters and caches related parameters have been varied within the specified range. For the *ADPCM Encoder*, it is not suitable to use a multi-cluster



**Fig. 5** Energy-Delay product for the *ADPCM Encoder* benchmark by varying the number of data-path units (clusters) and the cache sizes

configuration, since the minimum energy-delay product (highlighted by a circle) is associated with a single-cluster architecture. Since the energy computed by the model represents a lower bound on the energy consumption of the processor, we can reasonably assume that the real multi-cluster implementation will follow the same trend.

### 7.0.2. Exploration based on energy-delay pareto curves

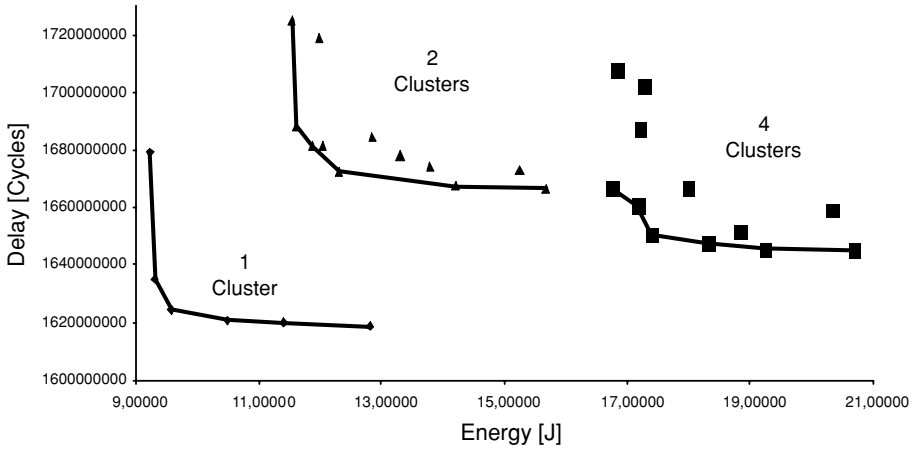
The second analysis consists of the construction of the Pareto curves of the energy and delay metrics of the system. A two-dimensional Pareto curve is defined [30] as the set of points that are not dominated in both dimensions by any other points. Pareto curves are often used to perform a constrained analysis of the design space. In fact, they are useful to assess the best configuration for a particular metric, given a constraint on the other one(s).

Figures 6 and 7 show the energy-delay scatter plots and the corresponding Pareto curves for two significant benchmarks among the 32 used for the analysis by varying the number of clusters and the cache sizes. Figure 6 represents the scatter plot and the Pareto curve for the *MPEG2 Decoder* benchmark: the mono-cluster Pareto curve represents the Pareto curve of the global system exploration. In other words, the single-cluster configuration dominates the benefits (both in terms of performance and energy) of the two-cluster and four-cluster configurations.

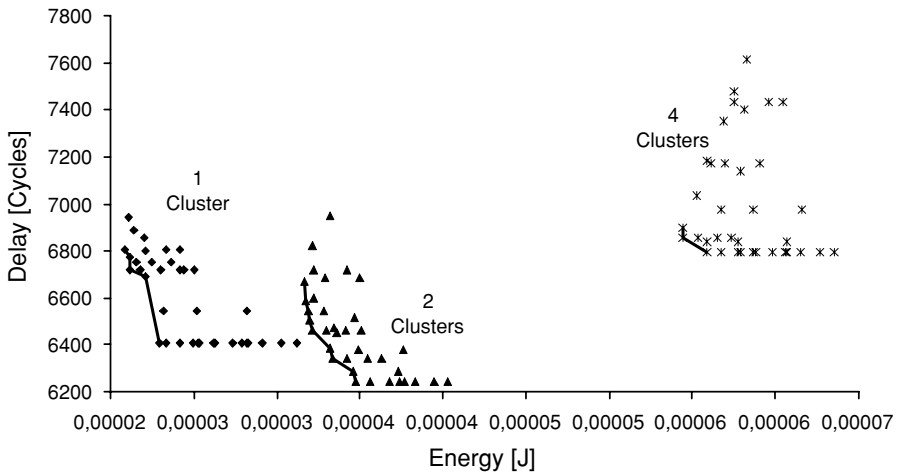
Figure 7 represents the scatter plot and the Pareto curve for the *AES* benchmark (encode and decode with a key of 128 bits). The Pareto curve of the global exploration consists of some points of the mono-cluster and two-cluster Pareto curves. The two-cluster configuration can be chosen if the performance constraints are not satisfied by the single-cluster configuration, providing lower energy dissipation.

## 8. Conclusions

In this paper, an instruction-level methodology has been proposed to estimate the energy consumption in embedded systems based on VLIW architectures. First of all, we have shown



**Fig. 6** Energy-Delay scatter plot for the *MPEG2 Decoder* benchmark by varying the number of data-path units (clusters) and cache sizes



**Fig. 7** Energy-Delay scatter plot for the *AES Encoder and Decoder* benchmark by varying the number of data-path units (clusters) and cache sizes

the reduction of the complexity of the energy model for VLIW cores, while preserving a good level of accuracy. Then we have shown how the proposed energy model can be further simplified by introducing a methodology to automatically cluster the operations in the ISA, based on the average energy behavior of the operations. The power model has then been extended to a multi-cluster VLIW architecture composed of multiple data-path units and a single instruction cache control unit. The power estimation methodology has then been applied to the Lx VLIW architecture provided by STMicroelectronics. The long-term goal of our work is to provide support to higher-level compilation-level and software-level power optimization techniques.

## References

1. Tiwari, V., S. Malik, and A. Wolfe. Power Analysis of Embedded Software: A First Step Towards Software Power Minimization. In *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 2, no. 4, pp. 437–445, 1994.
2. Lee, T., V. Tiwari, and S. Malik. Power Analysis and Minimization Techniques for Embedded DSP Software. In *IEEE Transactions on VLSI Systems*, vol. 5, no. 1, pp. 123–135, 1997.
3. Sami, M., D. Sciuto, C. Silvano, and V. Zaccaria. An Instruction-Level Energy Model for Embedded VLIW Architectures. In *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 21, no. r92, September 2002, pp. 998–1010.
4. Bona, A., M. Sami, D. Sciuto, C. Silvano, V. Zaccaria, and R. Zafalon. Energy Estimation and Optimization of Embedded VLIW Processors Based on Instruction Clustering. In *Proceedings of the 39th Design Automation Conference DAC'02*, June 2002, pp. 886–891.
5. Tiwari, V., S. Malik, A. Wolfe, and M. Lee. Instruction Level Power Analysis and Optimization of Software. In *J. VLSI Signal Processing*, pp. 1–18, 1996.
6. Russell, J., and M. Jacome. Software Power Estimation and Optimization for High Performance, 32-bit Embedded Processors. In *International Conference on Computer Design: VLSI in Computers and Processors*, pp. 328–333, 1998.
7. Trifone, D., D. Sarta and G. Ascia. A Data Dependent Approach to Instruction Level Power Estimation. In *Proc. IEEE Alessandro Volta Memorial Workshop on Low Power Design*, Como, Italy, March 1999, pp. 182–190.
8. Klass, B., D. Thomas, H. Schmit, and D. Nagle. Modeling Inter-Instruction Energy Effects in a Digital Signal Processor. In *Power-Driven Microarchitecture Workshop*, June 1998.
9. Ye, W., N. Vijaykrishnan, M. Kandemir, and M. Irwin. The Design and Use of Simplepower: A Cycleaccurate Energy Estimation Tool. In *Proc. 37th Design Automation Conference*, Los Angeles, CA, June 2000.
10. Vijaykrishnan, N., M. Kandemir, M.J. Irwin, H.S. Kim, and W. Ye. Energy-Driven Integrated Hardware-Software Optimizations Using Simplepower. In *ISCA 2000: 2000 International Symposium on Computer Architecture*, Vancouver BC, Canada, June 2000.
11. Li, Y., and J. Henkel. A Framework for Estimating and Minimizing Energy Dissipation of Embedded HW/SW Systems. In *DAC-35: ACM/IEEE Design Automation Conference*, June 1998.
12. Conte, T. M., K. N. Menezes, S. W. Sathaye, and M. C. Toburen. System-Level Power Consumption Modeling and Tradeoff Analysis Techniques for Superscalar Processor Design. In *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, vol. 8, no. 2, April 2000, pp. 129–137.
13. Brooks, D., V. Tiwari, and M. Martonosi. Wattch: A Framework for Architectural-Level Power Analysis and Optimizations. In *Proceedings ISCA 2000*, pp. 83–94, 2000.
14. Bellas, N., I. N. Hajj, D. Polychronopoulos, and G. Stamoulis. Architectural and Compiler Techniques for Energy Reduction in High-Performance Microprocessors. In *IEEE Transactions on Very Large Scale of Integration (VLSI) Systems*, vol. 8, no. 3, June 2000.
15. Bahar, R. I., G. Albera, and S. Manne. Power and Performance Tradeoffs Using Various Caching Strategies. In *ISLPED-98: ACM/IEEE Int. Symposium on Low Power Electronics and Design*, Monterey, CA, 1998.
16. Givargis, T. D., F. Vahid, and Jrg Henkel. Evaluating Power Consumption of Parameterized Cache and Bus Architectures in System-on-a-Chip Designs. In *IEEE Transactions on Very Large Scale of Integration (VLSI) Systems*, vol. 9, no. 4, August 2001.
17. Su, C. L., and A. M. Despain. Cache Design Trade-Offs for Power and Performance Optimization: A Case Study. In *ISLPED-95: ACM/IEEE Int. Symposium on Low Power Electronics and Design*, 1995.
18. Kamble, M. B., and K. Ghose. Analytical Energy Dissipation Models for Low Power Caches. In *ISLPED-97: ACM/IEEE Int. Symposium on Low Power Electronics and Design*, 1997.
19. Wilton, S. E., and N. Jouppi. An Enhanced Access and Cycle Time Model for On-Chip Caches. In Tech. Rep. 93/5, Digital Equipment Corporation Western Research Lab., 1994.
20. Hicks, P., M. Walnock, and R. M. Owens. Analysis of Power Consumption in Memory Hierarchies. In *ISLPED-97: ACM/IEEE Int. Symposium on Low Power Electronics and Design*, Monterey, CA, pp. 239–242, 1997.
21. Shiue, W.-T., and C. Chakrabarti. Power Estimation of System-Level Buses for Microprocessor-Based Architectures: A Case Study. In *Proc. DAC99: Design Automation Conference*, New Orleans, LU, 1999.
22. Lapinskii, V.S., M.F. Jacome, and G.A. de Veciana. Application-Specific Clustered VLIW Datapaths: Early Exploration on a Parameterized Design Space. In *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. CAD-21, no. 8, August 2002, pp. 889–903.

23. Faraboschi, P., G. Brown, J. Fisher, G. Desoli, and F. Homewood. Lx: A Technology Platform for Customizable Vliw Embedded Processing. In *Proceedings of the International Symposium on Computer Architecture*, June 2000, pp. 203–213.
24. Jain, A. K., M. N. Murty, and P. J. Flynn. Data Clustering: A Review. In *ACM Comp. Surveys*, vol. 31, no. 3, September 1999, pp. 264–323.
25. Gennari, J. H. A Survey of Clustering Methods. In Technical Report ICS-TR-89-38, University of California, Irvine, Department of Information and Computer Science, October 1989.
26. Wu, Q., Q. Qiu, M. Pedram, and C. Ding. Cycle-Accurate Macromodels for RT-level Power Analysis. In *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, vol. 6, December 1998, pp. 520–528.
27. Lee, C., J. K. Lee, and T. T. Hwang. Compiler Optimization on Instruction Scheduling for Low Power. In *Proceedings of The 13th International Symposium on System Synthesis*. September 20–22, 2000, pp. 55–60, IEEE Computer Society Press.
28. Lee, C., M. Potkonjak, and W. H. Mangione-Smith. Mediabench: A Tool for Evaluating Multimedia and Communication Systems. In *Proceedings of Micro 30*, 1997.
29. Wilton, S., and N. Jouppi. CACTI: An Enhanced Cache Access and Cycle Time Model. In *IEEE Journal of Solid-State Circuits*, vol. 31, no. 5, pp. 677–688, 1996.
30. Aho, A., J. Hopcroft, and J. Ullman. *Data Structures and Algorithms*. Addison-Wesley, Reading, MA, USA, 1983.