A Design Framework to Efficiently Explore Energy-Delay Tradeoffs

William Fornaciari[§] Donatella Sciuto[§] Cristina Silvano[◇] Vittorio Zaccaria[§]

[§]Politecnico di Milano
Dip. di Elettronica e Informazione
Milano, ITALY 20133
{fornacia,sciuto,zaccaria}@elet.polimi.it

[◇]Università degli Studi di Milano
Dip. di Scienze dell'Informazione
Milano, ITALY 20135
silvano@dsi.unimi.it

Abstract

Comprehensive exploration of the design space parameters at the system-level is a crucial task to evaluate architectural tradeoffs accounting for both energy and performance constraints. In this paper, we propose a system-level design methodology for the efficient exploration of the memory architecture from the energy-delay combined perspective. The aim is to find a sub-optimal configuration of the memory hierarchy without performing the exhaustive analysis of the parameters space. The target system architecture includes the processor, separated instruction and data level-one caches, the main memory, and the system buses. The methodology is based on the sensitivity analysis of the optimization function with respect to the tuning parameters of the cache architecture (mainly cache size, block size and associativity). The effectiveness of the proposed methodology has been demonstrated through the design space exploration of a real-world example: a MicroSPARC2-based system running the Mediabench suite. Experimental results have shown an optimization speedup of 329 times with respect to the full search, while the near-optimal system-level configuration is characterized by a distance from the optimal full search configuration in the band of 10%.

1. INTRODUCTION

Decreasing power consumption in microprocessor-based systems without significantly impacting performance is a must during the design of a broad range of embedded applications. Evaluation of energy-delay metrics at the system-level is of fundamental importance for embedded applications characterized by low-power and high-performance requirements. Given the application-specific functionality, the design of an embedded system requires the definition of the best architecture in terms of core processor, memory subsystem, and system-level bus topology. Full search of the optimal system architecture with respect to the energy-delay cost function can be computationally very costly due to the simulation time required to explore the wide space of parameters.

Several system-level exploration methods have been recently proposed in literature targeting power-performance tradeoffs from the system-level standpoint [1], [2], [3], [4], [5], [6], [7], [8]. In [5], the authors propose to sacrifice some performance to save power by filtering memory references through a small cache placed close to the processor (namely *filter cache*). Su and Despain [1] proposed a model to evaluate the power/performance tradeoffs in cache design and the effectiveness of novel cache design techniques targeted for low-power (such as vertical and horizontal cache partitioning). Kamble and Ghose ([9] proposed an analytical power model for various cache structures accounting for both technological parameters (such as capacitances and power supplies) and architectural factors (such as block size, associativity and capacity). An analytical model of energy consumption for the memory hierarchy has been provided in [10]. Power and performance tradeoffs in cache architectures have been also investigated in [3]. The *Avalanche* framework presented in [2] evaluates simultaneously the energy-performance tradeoffs for software, memory and hardware for embedded systems. The work in [6] proposes a system-level technique to find low-power high-performance superscalar processors tailored to specific user application. More recently, the Wattch architectural-level framework has been proposed in [7] to analyze power/performance tradeoffs with a good level of accuracy with respect to lower-level estimation approaches. Low-power design optimization techniques for high-performance processors have been investigated in [8] from the architectural and compiler standpoints.

Aim of this paper is to propose a system-level methodology for the efficient exploration of memory architectures for application-specific systems characterized by energy and delay constraints. We are focusing on the class of microprocessor-based embedded systems. The target of our work is to find a near-optimal configuration of the cache architecture without performing the exhaustive analysis of the space of parameters (mainly cache size, block size and associativity). The paper proposes a heuristic method to reduce the time spent during the simulation of different system configurations. The method is based on the sensitivity analysis of the system behavior with respect to the most relevant system-level parameters. In such a way, the resulting design exploration phase cost increases linearly with respect to the design space size. The system-level architecture we are focusing on includes a separate instruction and data L1 caches. To reduce the problem complexity, the I- and D-cache parameters have been optimized independently.

.

The cornerstone of our strategy is the dynamic profiling of the memory references obtained by tracing the software execution in terms of transition activity on system-level buses and by filtering the bus traces with a behavioral model of the caches. Bus traces, derived from the execution of several application programs, are analyzed from the energy-delay combined perspective to evaluate the cost associated with different architectural configurations.

The effectiveness of the proposed methodology has been validated by applying it to the design of a real-world representative example (a MicroSPARC2-based system) running the Mediabench suite [11]. Experimental results have shown how much the design exploration phase can be shortened while achieving either the optimal system-level configuration (obtained by the exhaustive system-level exploration) or a sub-optimal system-level configuration with a maximum error of 9.71%.

The paper is organized as follows. In the next section, the proposed system-level exploration methodology is described, while in Section 3, an application example of the exploration method is discussed. Finally, some concluding remarks and future directions of our work are drawn in Section 4.

2. DESIGN SPACE EXPLORATION

The analysis of goal functions at the system-level, plays a primary role during the design space exploration. Our focus is on processor-to-memory communication through the memory hierarchy. In this section, we describe *(i)* the separate optimization flow based on the analysis of energy-delay metrics; *(ii)* the sensitivity-based optimization; *(iii)* the system-level simulation environment; *(iv)* the energy-delay models used to optimize the system.

2.1 Separate Optimization Flow

The choice of the optimal system configuration, in many cases, is the most important and time consuming activity of the whole design process. Such a task is typically accomplished by considering some goal functions, e.g. taking into account energy consumption and performance. In many cases, the optimization of either energy or performance leads to sub-optimal architectures unable to meet the tight constraints of embedded applications. To be general and flexible, we adopted the $Energy * Delay$ product metric to compare alternative system configurations from both a power and performance standpoint. Despite the choice of a unique and comprehensive goal function to be optimized, the exhaustive analysis of the typical design space still remains an hard task.

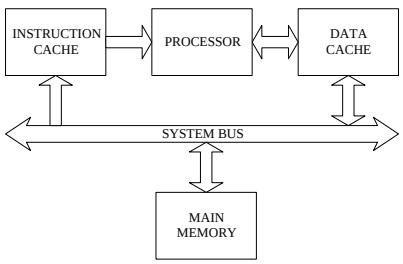

Figure 1: Target system architecture.

Let us consider the target configurable system architecture shown in Figure 1 composed of a processor, a separated I- and D- L1 caches, the main memory and the system buses. We assume to explore the design space of this architecture in terms of six parameters: cache size, block size and associativity of both D- and I- L1 caches. Thus each instance of the configurable architecture is described as a 6-tuple $t =< c_i, b_i, v_i, c_d, b_d, v_d > \in T = C_i \times B_i \times V_i \times C_d \times B_d \times V_d$ where:

- C_i, C_d are the spaces of sizes of I- and D-caches.
- B_i, B_d are the spaces of block sizes of I- and D-caches.
- V_i, V_d are the spaces of associativity of I- and D-caches.

Let us indicate as $I = C_i \times B_i \times V_i$ the space of I-cache parameters and as $D = C_d \times B_d \times V_d$ the space of D-cache parameters.

In this work, we propose to apply a separate analysis to the data and instruction streams in the memory hierarchy in order to compute two sub-optimal configurations, one for the D-cache and one for the I-cache (see Figure 2). The resulting two sub-optimal solutions are composed to build a sub-optimal solution for the entire system. Even if this procedure seems intuitive from the point of view of the system delay (because the delays introduced by the D- and the I-cache are independent to each other), this is not true from the point of view of the ED metric, where energy and delay contributions are merged together to build a complex system-level cost function.

Since the two sub-optimal configurations can be computed independently, we can find a sub-optimal configuration in the $I \times D$ space in a $|I| + |D|$ number of simulation steps instead of a full search requiring $|I| * |D|$ steps.

The sub-optimal configuration for the D-cache is computed by exploring a simplified system model in which the D-cache is kept varying and the I-cache is considered fixed and ideal, i.e., it does not introduce any delay in the system and it does not consume power. We denote this family of architectures (or subspace of exploration) as $\mathcal{M}(i_{ideal}, D) = \{i_{ideal}\} \times D$ where i_{ideal} is the ideal I-cache configuration and D is the D-cache parameter space. Given a family of systems $\mathcal{M}(i_{ideal}, D)$, the optimization (described in the next

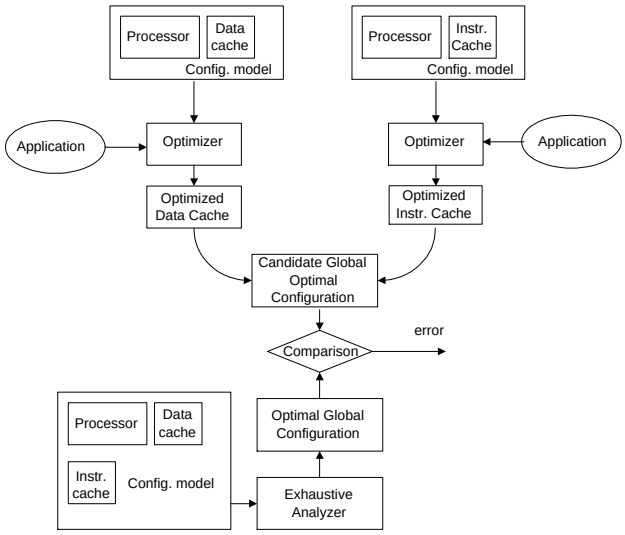

Figure 2: Separated I- and D-cache optimization flow

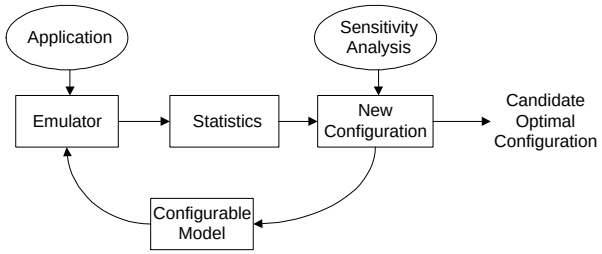

Figure 3: The sensitivity analysis controls the behavior of the sensitivity optimizer by suggesting a smart path in the architectural space exploration.

section) is applied to find a near-optimal D-cache configuration d_{opt}. The dual procedure is applied to a family of architectures in which the D-cache is considered fixed and ideal to find a near-optimal i_{opt} I-cache configuration.

2.2 Sensitivity-Based Optimization

The optimization methodology we are proposing is based on the *sensitivity* analysis of the system to the parameters defining its configuration as well as on the relative independence of some parameters with respect to others.

To perform the sensitivity-based optimization on both families of systems ($\mathcal{M}(I, d_{ideal})$ and $\mathcal{M}(i_{ideal}, D)$), the parameters of each family must be ordered by their *sensitivity*. We define the sensitivity $Sens(p)$ of a parameter p with respect to the full search optimal configuration t_{opt} as the maximum variation of the metric ED from $ED(t_{opt})$ when the component p is varied of the smallest δp possible.

This phase is called the *methodology tuning phase* and it has to be applied only once for a particular system. In order to perform the methodology tuning, a full architectural space exploration for a selected set of benchmarks must be done. In our application example, we selected eight Mediabench programs representative of a class of audio, image and video processing algorithms [11] to characterize the sensitivity of both the system models.

The sensitivity analysis produces *tuning* information that will be used for the fast determination of the optimal configuration for a new, arbitrary application running on the target system. During this optimization (shown in Fig. 3), we exploit the sensitivity analysis results to find the sub-optimal I-cache and D-cache configurations given an arbitrary application. First let us consider the optimization of the family $\mathcal{M}(I, d_{ideal})$ where we have to explore the I-cache configuration space $I = C_i \times B_i \times V_i$.

As a clarifying example, let us assume that, the sensitivity analysis has shown that the ED product is mostly affected first by the cache size, second by the associativity, and third by the block size. Our optimization methodology suggests to find the sub-optimal i_{opt} by optimizing first the most sensitive parameters in the following way:

1. Select a pair of values $(b_{i,0}, v_{i,0})$ to be used as initial values and perform the exhaustive search of the minimum of the function ED on the subspace of the I-cache configurations $\{< c_i, b_{i,0}, v_{i,0} > | c_i \in C_i\}$. In the experimental results we used the mean values of the B_i and V_i spaces. Define as $\hat{c}_{i,opt}$ the *estimated* I-cache size of the near-optimal configuration found.
2. Perform an exhaustive search of the minimum of the function ED on the subspace of the I-cache configurations $\{< \hat{c}_{i,opt}, b_{i,0}, v_i > | v_i \in V_i\}$. Define as $\hat{v}_{i,opt}$ the *estimated* associativity of the near-optimal configuration found.
3. Perform an exhaustive search of the minimum of the function ED on the subspace of I-cache configurations $\{< \hat{c}_{i,opt}, b_i, \hat{v}_{i,opt} > | b_i \in B_i\}$. Define as $\hat{b}_{i,opt}$ the *estimated* block size of the near-optimal I-cache configuration.

The 3-tuple $i_{opt} =< \hat{c}_{i,opt}, \hat{b}_{i,opt}, \hat{v}_{i,opt} >$ represents the estimated ED sub-optimal I-cache configuration for the $\mathcal{M}(I,$-$d_{ideal})$ family. This configuration has been found in $|C_i| + |B_i| + |V_i|$ steps, while the exhaustive search would require $|C_i| * |B_i| * |V_i|$ steps.

The optimization task of the methodology is then applied to the D-cache by searching for a $d_{opt} =< c_d, b_d, v_d > \in C_d \times B_d \times V_d$ that minimizes the ED product for the $\mathcal{M}(i_{ideal}, D)$ family of architectures.

The combined results $< i_{opt}, d_{opt} > \in T$ constitutes the near- optimal configuration for the entire system, found with with $|C_i|+|B_i|+|V_i|+|C_d|+|B_d|+|V_d|$ simulation steps while the full search optimal configuration would have required $|C_i| \times |B_i| \times |V_i| \times |C_d| \times |B_d| \times |V_d|$ steps.

2.3 System-Level Simulation Framework

To apply this general methodology, we require a system-level simulation environment to profile dynamically the behavior of the multi-level memory hierarchy. The system architecture we address is quite general and models system-level buses in terms of the main parameters that affect the energy-delay behavior of the processor-to-memory communication (operating frequency, transition activity, capacitive load, power supply, bus width, cache hit rate, cache and memory access time, etc.).

The system-level simulation environment [12] [13] is mainly composed of the following modules: *(i)* Software Execution Profiler, *(ii)* Configurable Memory Model, *(iii)* Configurable Bus Model. In our approach, the performance and the energy associated to each level of the memory hierarchy, mainly depends on the number of accesses. Given a target software application, our framework enables us to determine the actual number of accesses to each level in the hierarchy and the corresponding hit rates. The method is based on the profiling of the memory references generated by the processor during the execution of the application software in terms of transition activity on system-level buses. The Software Execution Profiler combines a cycle-accurate Instruction-Set Simulator (ISS) to execute software application programs and a dynamic tracer to generate data and address bus streams during the program execution. The bus traces generated by the Software Execution Profiler are given as inputs to the Configurable Memory Model, that includes different levels in the memory hierarchy, such as on- and off-processor $L1$ and $L2$ caches as well as the main memory. Each level of storage can be customized in terms of design parameters (such as cache size, block size, associativity, memory array organization, etc.).

In particular, the bus traces generated by the Software Execution Profiler are filtered by a behavioral model of the first level cache (dependent on cache size, block size, associativity, replacement policy, write strategy, etc.), then they are passed to the behavioral model of the second level cache and finally to the main memory model.

2.4 Energy-Delay Models

In our system-level model, we focus on the energy and the delay associated with the processor-to-memory communication, considering the contributions of the processor core, the system-level buses and each level of the memory hierarchy. While the delay calculation is embedded in the cycle-accurate simulation environment, the statistics of each resource of the system have to be extracted and imported into the analytical energy models to evaluate the overall energy-delay cost functions. These analytical energy models include the following contributions:

- processor core;
- processor I/O pads;
- processor-to-$L1$ on-chip buses;
- on-chip $L1$ I- and D-caches;
- $L1$-to-$L2$ off-chip buses;
- $L2$ unified $SRAM$ cache;
- $L2$-to-MM off-chip buses;
- MM $DRAM$.

All the energies and delays presented in this this work are normalized to the instructions executed. This is done in order to compare the behavior of different applications on the same system architecture and to compare different architectures on the same application. For the processor core, we use a simple model that accounts for an average power consumed by the processor in normal operating conditions and we multiplied it by the average execution time of an instruction. In general, this model depends on the particular processor used. For system-level buses (such as the processor-to-$L1$ buses), the energy model can be simply expressed as: $E \approx (C_{load} V_{dd}^2 n_{trans})/2$, where C_{load} is the bus load capacitance, V_{dd} is the power supply voltage and n_{trans} is the number of transitions on the bus lines. The n_{trans} term is given by:

$$n_{trans} \approx \frac{\sum_{t=0}^{L-1} H(B^{(t)}, B^{(t+1)})}{(L-1)} \qquad (1)$$

where H is the Hamming distance between the bus line B at time t and the bus line at time $(t+1)$ and L is the total length of the bus stream. The actual values of $B^{(t)}$, derived from our simulation environment, are considered in our model.

For on-chip $L1$ caches, our analytical energy model is based on the model developed in [9], that accounts for:

- technological parameters (such as capacitances and power supplies);
- architectural parameters (such as block size and cache size);
- switching activity parameters (such as number of bit line transitions).

The energy model has been used in recent works (such as in [5]), where the switching activity parameters have been calculated either by using application-dependent statistics or by assuming typical values (such as half of the address lines switching during each memory request). In our cache energy model, we directly import the actual values of hit/miss rates and and transitions on cache components, that have been derived by our system-level simulation environment to account for actual profiling information depending on the application software.

3. AN APPLICATION EXAMPLE

In this section, we present the experimental environment setup to explore a system architecture (see Fig. 1) composed of the following modules :

- A 100MHz MicroSPARC2 Processor Core (operating at 3.3V and without I- and D-caches).
- Separated and configurable I- and D-caches implemented in $0.8\mu m$ CMOS technology with one-cycle hit-time. The speed of the processor-to-memory bus is 100MHz.
- A 32Mbyte DRAM composed by 16×16-Mbit blocks and characterized by a 7-cycle latency. The power model for this memory has been derived from [14]. The speed of the bus cache-DRAM is 100MHz.

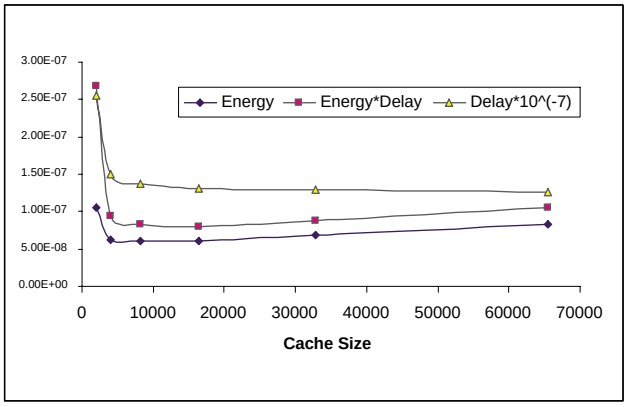

Figure 4: Energy $[JPI]$, normalized Delay $[CPI]$ and Energy*Delay $[JPI * CPI]$ product for *adpcmdec* benchmark with respect to cache size for a 4-way set associative cache with 32B block

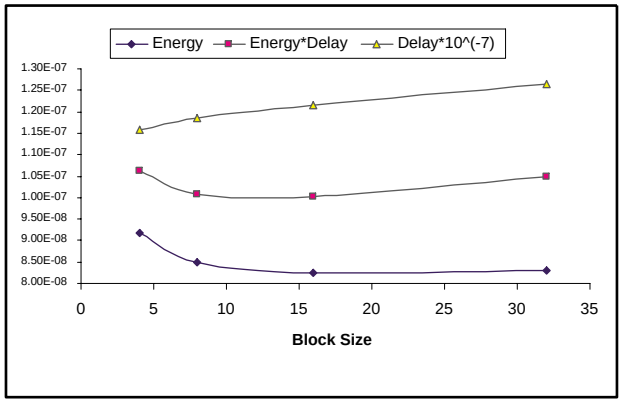

Figure 5: Energy $[JPI]$, normalized Delay $[CPI]$ and Energy*Delay $[JPI * CPI]$ product for *adpcmdec* benchmark with respect to block size for a 64KB 4-way set associative cache.

The cache energy model is related to $0.8\mu m$ caches implemented in $CMOS$ technology, however the equations in the analytical energy model can be easily modified to reflect

Benchmark	*Sens(C)*	*Sens(B)*	*Sens(V)*
adpcmdec	11,6%	0,3%	4,1%
adpcmenc	9,7%	0,1%	3,5
g721decode	109,3%	1,0%	42,5%
gsdec	7,1%	2,8%	0,1%
pegwit	92,8%	0,7%	1,1%
pgp	51,3%	0,9%	6,6%
rasta	29,0%	3,7%	3,1%
Mean	**44,4%**	**1,4%**	**8,7%**

Table 1: Sensitivity of ED w.r.t. cache size, block size and associativity for L1 I-Cache.

Benchmark	*Sens(C)*	*Sens(B)*	*Sens(V)*
adpcmdec	126,1%	8,5%	2,1%
adpcmenc	125,7%	8,8%	2,0%
g721decode	2,1%	0,4%	11,0%
gsdec	9,3%	1,7%	1,5%
pegwit	20,9%	37,1%	3,3%
pgp	86,0%	3,6%	21,3%
rasta	45,8%	0,3%	11,5%
Mean	**59,4%**	**8,6%**	**7,5%**

Table 2: Sensitivity of ED w.r.t. cache size, block size and associativity for L1 D-Cache.

a more updated process technology by simply changing the values of the capacitance parameters to account for technological and layout features.

Each instance of the virtual architecture has been described as a 6-tuple $t =< c_i, b_i, v_i, c_d, b_d, v_d > \in T = C_i \times B_i \times V_i \times C_d \times B_d \times V_d$ where:

- C_i, C_d ={2KB, 4KB, 8KB, 16KB, 32KB, 64KB}.
- B_i, B_d ={4B, 8B, 16B, 32B}.
- V_i, V_d ={1, 2, 4, 8}).

The architecture has been explored by using our in-house developed tool, called **MEX,** that simulates the execution of a program compiled for the Sparc V8 architecture within a configurable memory architecture. **MEX** exploits the Shade [15] library to trace the memory accesses made by a SPARC V8 program and consequently simulates the target memory architecture to obtain accurate memory access statistics. At the end of the simulation, **MEX** reports the following statistics:

- number of accesses generated to the memory hierarchy (on both I- and D-buses);
- average cache miss rate (on both I- and D-buses);
- address sequentiality (on both I and D address buses) [12];
- bus transition activities (on both I and D address buses);
- delay D (average clock cycles per instruction, measured in $[CPI]$);
- energy E dissipated by the architecture during program execution measured in *Joule per Instruction* $[JPI]$.

To give the flavor of the ED trend for the *adpcmdec* benchmark, Figure 4 reports the behavior of the ED metric with respect to the cache size for fixed associativity and block size, while Figure 5 reports the ED metric with respect to the block size for fixed $64KB$ 4-way set associative cache. Table 1 reports the sensitivity index with respect to the three cache parameters (cache size, block size and associativity) for the $\mathcal{M}(I, d_{ideal})$ family. Similarly Table 2 reports the sensitivity data derived for the $\mathcal{M}(i_{ideal}, D)$ family .

	Proposed Method			Full Search			Error
Benchmark	C_{opt}	B_{opt}	V_{opt}	C_{opt}	B_{opt}	V_{opt}	on ED
epic	2KB	8B	2	2KB	8B	2	0,00%
gsmdec	8KB	8B	8	8KB	8B	8	0,00%
gsmenc	16KB	16B	2	16KB	16B	2	0,00%
jpegdec	4KB	4B	4	4KB	4B	4	0,00%
jpegenc	4KB	8B	4	4KB	8B	4	0,00%
mesa	16KB	16B	8	16KB	16B	4	0,21%
mpegdec	8KB	8B	4	8KB	8B	4	0,00%
mpegenc	2KB	8B	4	2KB	8B	4	0,00%
unepic	2KB	8B	2	2KB	8B	2	0,00%
g721encode	8KB	16B	2	8KB	16B	2	0,00%

Table 3: Comparison between the I-cache configurations derived with the proposed method and the full search analysis.

	Proposed Method			Full Search			Error
Benchmark	C_{opt}	B_{opt}	V_{opt}	C_{opt}	B_{opt}	V_{opt}	on ED
epic	16KB	4B	2	16KB	4B	2	0,00%
gsmdec	8KB	16B	2	8KB	16B	2	0,00%
gsmenc	4KB	8B	4	8KB	16B	1	1,92%
jpegdec	32KB	16B	4	32KB	16B	4	0,00%
jpegenc	32KB	8B	2	64KB	16B	1	1,90%
mesa	16KB	4B	4	8KB	4B	8	1,00%
mpegdec	8KB	4B	4	4KB	4B	4	0,13%
mpegenc	4KB	4B	4	2KB	4B	4	0,17%
unepic	32KB	4B	4	8KB	4B	2	2,39%
g721encode	2KB	8B	4	2KB	4B	2	0,23%

Table 4: Comparison between the D-cache configurations derived with the proposed method and the full search analysis.

In the case of the I-cache, the ED metric is affected, first by cache size (44.4%), second by associativity (8.7%) and third by block size (1.4%). For the D-cache, the ED metric is affected, in order, by cache size (59.4%), block size (8.6%) and associativity (7.5%). The results of these two sensitivity analysis are used to optimize the $\mathcal{M}(I, d_{ideal})$ and $\mathcal{M}(i_{ideal}, D)$ families for any new algorithm without a full search of the exploration phase. In our case, this corresponds to 14 simulation steps instead of 96 steps (585% speedup).

To assess the methodology, we performed the separate optimization presented in Section 2.1 for each application of the remaining set of Mediabench applications. Table 3 compares the I-cache configurations estimated by our approach with those found with the full search. The table shows also the percentage errors on the ED metric. For all benchmarks but one, the near-optimal configuration found coincides with the full search one.

To complete the I-cache analysis, we varied the choice of the initial values $< b_{i,0}, v_{i,0} >$ among all the $16 = |B_i| * |V_i|$ possible pairs of initial values and we found an maximum error between the *estimated* sub-optimal configuration and the full search-optimal configuration of 6.10%. After the I-cache analysis, we performed the dual sensitivity optimization on the D-cache by optimizing, in order, D-cache, block size and associativity. The related results are shown in Table 4

Finally, we show how the combined $< i_{opt}, d_{opt} >$ can be used as a near-optimal global configuration. Table 5 compares the full search optimal configurations with the combined near-optimal configurations. The table shows also the percentage errors between the full search optimal ED and the near-optimal ED values. The maximum error is

	Full Search						Proposed Method						Error
	I-cache			D-cache			I-cache			D-cache			
Benchmark	C_{opt}	B_{opt}	V_{opt}	C_{opt}	B_{opt}	V_{opt}	C_{opt}	B_{opt}	V_{opt}	C_{opt}	B_{opt}	V_{opt}	on ED
`epic`	2KB	8B	2	32KB	4B	4	2KB	8B	2	16KB	4B	2	0,07%
`gsmdec`	8KB	8B	8	16KB	32B	2	8KB	8B	8	8KB	16B	2	0,23%
`gsmenc`	16KB	16B	2	8KB	16B	1	16KB	16B	2	4KB	8B	4	0,19%
`jpegdec`	4KB	4B	4	32KB	16B	4	4KB	4B	4	32KB	16B	4	0,00%
`jpegenc`	4KB	8B	4	64KB	8B	1	4KB	8B	4	32KB	8B	2	9,17%
`mesa`	16KB	16B	4	16KB	4B	4	16KB	16B	8	16KB	4B	4	0,53%
`mpegdec`	8KB	8B	4	8KB	4B	4	8KB	8B	4	8KB	4B	4	0,00%
`mpegenc`	2KB	4B	4	4KB	4B	4	2KB	8B	4	4KB	4B	4	0,00%
`unepic`	2KB	8B	2	64KB	8B	4	2KB	8B	2	32KB	4B	4	0,25%
`g721encode`	8KB	16B	2	8KB	8B	2	8KB	16B	2	2KB	8B	4	1,21%

Table 5: Comparison between the I- and D-cache configurations derived with the proposed method and those derived with the full search analysis.

9.17%. Note that the combined optimal configuration has been found with $|C_i|+|B_i|+|V_i|+|C_d|+|B_d|+|V_d| = 28$ simulations while the full search needs $|C_i| \times |B_i| \times |V_i| \times |C_d| \times |B_d| \times |V_d| = 9216$ steps. This means that our methodology enables a speedup in terms of steps of approximately 329 times on the design space exploration phase. As an example, for a generic application requiring a simulation time of 2 minutes for a single simulation steps, the full search optimization would have required approximately 13 days, while our methodology finds a near optimum solution within 56 minutes. A full search step can exploit the information generated by previous steps without performing a real simulation thus reducing the actual evaluation time of an architecture.

4. CONCLUSIONS AND FUTURE WORK

This paper addresses the problem of design space exploration of system architectures where both performance and power consumption are relevant issues. In particular, a methodology to reduce simulation time dramatically while preserving acceptable design accuracy has been proposed and experimentally assessed by considering the design of the memory subsystem of a real-world processor. Experimental results have shown a speed-up in simulation time of 329 times and a distance from the optimal configuration always in the band of 10%. Furthermore, this methodology allows the designer to save analysis time since the number of configuration to be compared is significantly reduced. Work is in progress to validate this methodology on a broader range of processor architectures.

5. REFERENCES

[1] C. L. Su and A. M. Despain, "Cache Design Trade-offs for Power and Performance Optimization: A Case Study," *ISLPED-95: ACM/IEEE Int. Symposium on Low Power Electronics and Design*, 1995.

[2] Y. Li and J. Henkel, "A Framework for Estimating and Minimizing Energy Dissipation of Embedded HW/SW Systems," *DAC-35: ACM/IEEE Design Automation Conference*, June 1998.

[3] R. I. Bahar, G. Albera, and S. Manne, "Power and Performance Tradeoffs using Various Caching Strategies," *ISLPED98: ACM/IEEE Int. Symposium on Low Power Electronics and Design*, Monterey, CA, 1998.

[4] C. A. Mandal, P. P. Chakrabarti and S. Ghose, "A Design Space Exploration Scheme for Data-Path Synthesis," *IEEE Trans. on Very Large Scale Integration (VLSI) Systems* , Vol. 7, No. 3, Sep. 1999, pp. 331-338.

[5] J. K. Kin, M. Gupta and W. H. Mangione-Smith, "Filtering Memory References to Increase Energy Efficiency," *IEEE Trans. on Computers*, Vol. 49, No. 1, Jan. 2000.

[6] T. M. Conte, K. N. Menezes, S. W. Sathaye and M. C. Toburen, "System-Level Power Consumption Modeling and Tradeoff Analysis Techniques for Superscalar Processor Design," *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, Vol. 8, No. 2, Apr. 2000, pp. 129-137.

[7] D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: A Framework for Architectural-Level Power Analysis and Optimizations," *ISCA 2000: 2000 International Symposium on Computer Architecture*, Vancouver BC Canada, pp. 83-94, June 2000.

[8] N. Bellas, I. N. Hajj, D. Polychronopoulos, and G. Stamoulis "Architectural and Compiler Techniques for Energy Reduction in High-Performance Microprocessors," *IEEE Transactions on Very Large Scale of Integration (VLSI) Systems*, Vol. 8, no 3, June 2000.

[9] M. B. Kamble and K. Ghose, "Analytical Energy Dissipation Models for Low Power Caches," *: ISLPED97: ACM/IEEE Int. Symposium on Low Power Electronics and Design*, 1997.

[10] P. Hicks, M. Walnock, and R. M. Owens, "Analysis of Power Consumption in Memory Hierarchies," *ISLPED-97: ACM/IEEE Int. Symposium on Low Power Electronics and Design*, Monterey, CA, August 1997, pp. 239-242.

[11] C. Lee, M. Potkonjak and W. H. Mangione-Smith, "MediaBench: A Tool for Evaluating Multimedia and Communication Systems," *Proc. of MICRO30*, 1997.

[12] W. Fornaciari, M. Polentarutti, D. Sciuto, and C. Silvano, "Power Optimization of System-Level Address Buses based on Software Profiling," *CODES-2000: 8th Int. Workshop on Hardware/Software Co-Design*, San Diego, CA, May 2000.

[13] W. Fornaciari, D. Sciuto, and C. Silvano, "Power Estimation of System-Level Buses for Microprocessor-Based Architectures: A Case Study," *ICCD99: 1999 IEEE Int. Conf. on Computer Design*, Austin, Texas, Oct. 1999.

[14] NEC, "16M-bit Synchronous DRAM Data Sheet," Doc. No. M12939EJ3V0DS00, 3rd Ed., April 1998.

[15] B. Cmelik, D. Keppel, "Shade: A Fast Instruction-Set Simulator for Execution Profiling," *ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, 1994.