

# A Methodology for the Efficient Architectural Exploration of Energy-Delay Trade-offs for Embedded Systems

L. Salvemini<sup>§</sup> M. Sami<sup>‡</sup> D. Sciuto<sup>‡</sup> C. Silvano<sup>‡</sup> V. Zaccaria<sup>‡</sup> R. Zafalon<sup>†</sup>  
<sup>§</sup> ALaRI, Lugano, Switzerland <sup>‡</sup> Politecnico di Milano, Milano, Italy  
<sup>†</sup> STMicroelectronics, Agrate B. (MI), Italy

## ABSTRACT

The main goal of this paper is to identify the best architecture of an embedded system by considering at the same time energy and delay, avoiding the comprehensive analysis of the architectural design space. We adopt the Energy-Delay Product (EDP) as the evaluation metric to compare the alternative architectures of the target system. The paper analyzes an extended adaptive random search algorithm (ADGREED) to efficiently explore the architectural design space. The ADGREED algorithm is a pseudo-random optimisation algorithm that combines the best potentialities of the adaptive random search (ADRAS) and the Greedy deterministic algorithm. The analysis has been carried out through the architectural optimisation of the memory subsystem of a real-word embedded system executing the set of Mediabench benchmarks for multimedia applications. The reported experimental results have shown a reduction up to one order of magnitude of the number of design alternatives analyzed during the exploration phase, while maintaining very high accuracy.

**Keywords:** Design Space Exploration, Embedded Systems.

## 1. INTRODUCTION

Decreasing energy consumption without a relevant impact on 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 mainly in terms of core processor, number of levels in the memory hierarchy, cache-related parameters, system-level bus topology, width of address and data busses, etc.. To achieve this goal, an approach based on the full search

of the optimal architectural parameters at the system-level with respect to the energy-delay cost function can be computationally very costly due to the long simulation time required to explore the wide space of parameters.

Aim of our work is to provide a high level analysis technique for an efficient and flexible exploration of the architectural design space for embedded systems from an energy/delay comprehensive standpoint. For each point of the design space, we estimate the value of the corresponding Energy-Delay Product (EDP), taking into consideration both performance and energy constraints. The target of our work is to find a possibly optimal or near-optimal system configuration without performing the exhaustive analysis of the space of the chosen parameters.

In this paper, we analyze the integration of a heuristic algorithm (namely, ADGREED) to speed up the optimisation phase of the search of the best system-level configuration in terms of the EDP.

The ADGREED algorithm is a pseudo-random optimisation algorithm that combines the best potentialities of the adaptive random search (ADRAS) and the Greedy deterministic algorithm. For the analysis completeness, other optimisation algorithms appeared in literature so far have also been investigated.

Both local search algorithms and global search algorithms have been implemented in the optimizer module of the proposed framework for comparison purposes with respect to the target ADGREED algorithm.

The reported results have shown a reduction of one order of magnitude in the number of alternatives analyzed during the design exploration phase, while achieving either the same optimal system-level configuration obtained by the exhaustive system-level exploration or a sub-optimal system-level configuration. This analysis confirmed that the family of random search algorithms is a viable solution for speeding-up the design space exploration phase.

The paper is organized as follows. The next section presents the most relevant approaches for system-level exploration appeared in literature so far, while Section 3 proposes our design space exploration framework. The ADGREED pseudo-random optimization algorithm is discussed in Section 4. The results derived from the application of the proposed methodology to a case study have been reported in Section 5. Finally, Section 6 summarizes the main contributions of this work and outlines the directions for our future research.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.  
SAC 2003, Melbourne, Florida, USA.

## 2. BACKGROUND

Several system-level estimation and exploration methods have been recently proposed in literature targeting power-performance tradeoffs from the system-level standpoint.

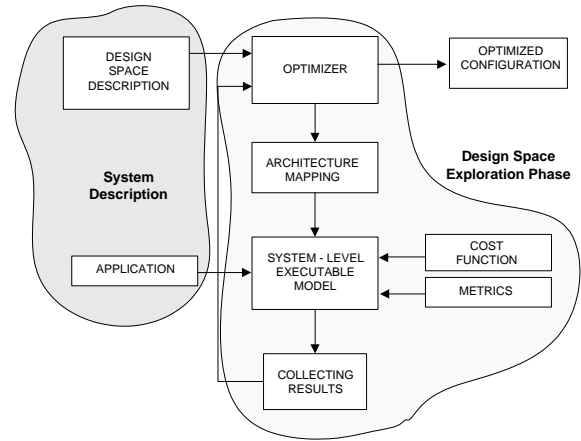
Among these works, the most significant methods related to our approach can be divided into two main categories: (i) system-level power estimation and exploration in general, and (ii) power estimation and exploration focusing on cache memories.

In the first category, the SimplePower approach [1] can be considered one of the first efforts to evaluate the different contributions to the energy budget at the system-level. The *Avalanche* framework presented in [2] evaluates simultaneously the energy-performance tradeoffs for software, memory and hardware for embedded systems. The work in [3] proposes a system-level technique to find low-power high-performance superscalar processors tailored to specific user applications. More recently, the Wattch architectural-level framework has been proposed in [4] to analyze power vs. 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 [5] 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 [6], where the authors propose a simulation-based approach to configure the parameters related to the caches and the buses.

In the second category of approaches dealing with power estimation and exploration for the memory hierarchy, the authors of [7] propose to sacrifice some performance to save power by filtering memory references through a small cache placed close to the processor (namely *filter cache*). A similar idea has been exploited in [8], where memory locations with the highest access frequencies are mapped onto a small, low-energy, and application-specific memory that is placed close to the core processor. Power and performance tradeoffs in cache architectures have been also investigated in [9]. A model to evaluate the power/performance tradeoffs in cache design has been proposed in [10], 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 [11]. 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 [12] for a  $0.8 \mu\text{m}$  process technology. The analytical model of energy consumption for the memory hierarchy has been extended in [13] and [14] where the cache energy model is included in a more general approach for the exploration of memory parameters for low-power embedded systems.

## 3. DESIGN SPACE EXPLORATION

Even considering a simple embedded microprocessor-based architecture composed of the CPU and the memory hierarchy, the identification of the optimal system configuration by trading off power and performance still leads to the analysis of too many alternatives. The overall goal of our work aims at overcoming such problems by providing a methodology and a design framework to drive the designer towards



**Figure 1: The proposed design space exploration flow**

optimal solutions in a cost-effective manner. We achieved such a goal by integrating a fast optimizer within a framework for the design space exploration, as shown in Figure 1. The framework receives as input a description of the possible design configurations (i.e., the *design space*) and an application for which an optimal configuration (with respect to the EDP) must be found.

The 'Optimizer' module is responsible for choosing, from the design space, a set of candidate optimal points to be evaluated in terms of the cost function. Once selected, each point is mapped to the specific simulation framework of the architecture (typically, a configuration file) and it is used to simulate the application by means of the 'Executable Model' to gather information about its *cost*. The results are then collected to give feedback to the 'Optimizer' module which, in turn, can decide whether continuing searching over the design space or stopping and generating an estimate of the optimal configuration. The architecture is modular and flexible; the optimizer module is unaware of the specific implementation of the architectural simulator thus, when a new architecture must be evaluated, only the 'Architecture Mapping' module must be changed.

### 3.1 Definition of the Design Space

The design space of an architecture takes into account all the possible combinations of the configurable parameters. For a modern, embedded architecture, a reasonable set of parameters strongly impacting on the performance and energy consumption can be the following:

- Number of levels in the memory hierarchy and positioning (on-chip, off-chip);
- Cache configuration parameters (cache size, line size, associativity etc.);
- Unified vs split data/instruction cache, at any hierarchy level;
- Width of address and data buses;
- Encoding strategy for the information flowing on both data and address buses;

Micro-architectural-level parameters (e.g., number of function units and topology of system-level busses) and software-level parameters (e.g., compiler optimization flags) can be also included in the design space and simulated within our framework; however, in this paper we will analyze the main contribution given only by the variation of the parameters related to the first level of the memory hierarchy.

In the rest of the paper we will indicate the generic point in the design space as the vector  $a \in \mathcal{A}$  where  $\mathcal{A}$  is the architectural space defined as:

$$\mathcal{A} = S_{p_1} \times \dots \times S_{p_l} \dots \times S_{p_n} \quad (1)$$

where  $S_{p_l}$  is the ordered set of possible configurations for parameter  $p_l$  and "x" is the cartesian product.

A configuration element  $a_{p_l}$  ( $a \in \mathcal{A}$ ) is an integer value representing the position in the ordered set  $S_{p_l}$  of the given parameter value. For example, if we have the set

$$S_i = \{ "2K", "4K", "8K", "16K" \}$$

the value  $a_i = 3$  represents "8K". This is useful to easily perform computations over the space  $S_i$  in terms of distances, e.g., "16K" has distance 3 from "2K".

### 3.2 The Cost Function

The *cost function* of a system provides a quantitative analysis of the system characteristics and provides a mean to optimize the system by discarding non-optimal system configurations and by considering several metrics simultaneously. Here, both performance and power are considered equally relevant for an embedded system, and should be captured by a general and flexible goal function. These metrics have been extensively discussed since last decade [7] and, based on this literature, we adopted the Energy-Delay Product (EDP) metric to compare alternative system designs.

The EDP product for an architectural configuration  $a$  and an application  $k$  is defined as:

$$\text{EDP}(a, k) = \text{EPI}(a, k) \times \text{CPI}(a, k) \quad (2)$$

where  $\text{EPI}(a, k)$  is the energy per instruction dissipated by the system in configuration  $a$  during the execution of application  $k$  and  $\text{CPI}(a, k)$  is the average number of clock cycles necessary to the system in configuration  $a$  to execute an instruction of  $k$ .

To compute such a general goal function, we focus on the energy and the delay associated with the processor and memory parts of a system, considering the contributions of the processor core, the system-level busses and each level of the memory hierarchy. While the delay calculation is embedded in the cycle-accurate simulation environment (the *System Executable Model*, see Figure 1), the use of each system resource has to be extracted and imported into analytical energy models defined to evaluate the overall energy metric. All these statistics are gathered by profiling the functional and memory behavior of the processor during the simulation of the embedded application by means of a cycle-accurate Instruction-Set Simulator (*ISS*).

In our study the analytical energy models include the following contributions: processor core, processor I/O pads, processor-to-L1 on-chip busses, on-chip L1 I- and D-caches, L1-to-L2 off-chip busses, L2 unified SRAM cache; Main memory (DRAM), L2-to-DRAM off-chip busses.

All the energies and delays presented in this work are normalized with respect 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 suitable model able to provide the power consumption at the instruction-level [15]. It consists of a simple instruction-level model that accounts for the average power consumed by the processor in normal operating conditions and we multiplied it by the average execution time of an instruction.

For the system-level busses (such as the processor-to-L1 busses), 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, depending on the Hamming distance between subsequent vectors composing the bus stream.

For on-chip L1 caches, we used the analytical energy model developed in [11], 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).

This cache energy model has been used in recent works (such as in [7]), where the switching activity parameters have been computed 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 framework, we directly import the actual values of hit/miss rates and bit transitions on cache components, that have been derived by the system-level simulation environment to account for the actual profiling information depending on the execution of embedded applications.

Finally the characterization of the main memory has been carried out by directly importing information derived from the data-sheet into our model.

### 3.3 Design Space Exploration

Even if a unique and comprehensive goal function to be optimized is selected, an exhaustive analysis of the typical design space still remains a complex and hard task. Let us consider a simple architecture, to be configured according to three parameters, one with 8 different values: the resulting number of possible configurations to be compared is over 500 for each application to be considered for the system. In such cases, a *brute force* approach could be impractical if the time required by a single simulation is high. Thus some heuristics to perform a near-optimal search must be applied to find out acceptable solutions while avoiding impractical simulation times.

The heuristics that can be used to find an optimal configuration can be divided in two categories: *local search algorithms* and *global search algorithms*. Local search algorithms are typically used for finding local minima with a high convergence speed; however, if the global optimum has to be found, the application of local search algorithms is limited only to convex spaces where the local minimum corresponds to the global one. These algorithms are usually characterized by an iterative process in which each new point to be

analyzed depends on the current analyzed point. In this paper we will compare our heuristic with the following local search algorithms: greedy algorithm, gradient algorithm and sensitivity based algorithm.

Global search algorithms are typically used for finding global minima; they are fundamentally based on rules that allow escaping from local minima by means of random point selection (random search [16]). However, random selection is one of the most important factors impacting the convergence speed.

In this study, we introduce and analyze an extended Adaptive Random Search Algorithm (called ADGREED) to explore the possible design space of microprocessor-based platforms with a configurable memory hierarchy. The heuristic algorithm is implemented in the *optimizer* module of the framework (see Figure 1) that is responsible for the generation/evaluation of new alternatives of the system configuration in order to find the near-optimal one. The proposed algorithm is described in the next section.

#### 4. ADAPTIVE RANDOM SEARCH ALGORITHM

ADGREED is a search algorithm that consists of a combination of the *ADaptive RAndom Search* algorithm (or ADRAS) [16] and a Greedy deterministic algorithm. The choice of such a combination derives from the need for accuracy in the global search (a property of random algorithms) and for high convergence speed in local search (proper of deterministic algorithms).

The pseudo-code of the ADGREED algorithm is shown in Figure 4. The algorithm receives as inputs the maximum total number of simulations  $N_{MAX}$ , the maximum number of simulations per window  $N_w$  and the starting point  $a(0)$ . The algorithm returns the estimate of the optimal configuration  $\hat{a}_{opt}$  within the design space.

The algorithm generates a set of  $N_{MAX}$  different architectures in the neighborhood of the current estimate of the optimum  $\hat{a}_{opt}$ . This is done by means of the function `generate_new_point(A, x, k)`. The function generates a new and feasible configuration vector *tmp* that is guaranteed to be not contained in the set of already visited points *A*, by using the following expression for each element *i* of the configuration vector *tmp*:

$$tmp_i = x_i + \lambda(x_i)(2\theta - 1)^k \quad (3)$$

where  $\theta$  is a randomly generated number,  $\lambda(x_i)$  is the maximum distance between  $x_i$  and the boundary of the subspace  $S_i$  and  $k$  is an odd integer number. As an example, if we assume  $S_i = \{ "2K", "4K", "8K", "16K" \}$ ,  $x_i = 3$  ("8K"),  $k = 1$  and  $\theta = 0.2$ , then  $\lambda(x_i) = 2$  since  $x_i$  is 2 step from the farthest boundary (i.e., "2K"), thus  $tmp_i = 3 - 1.2 \sim 2$  ("4K").

The ADGREED algorithm starts by choosing  $k = 1$  and it generates a *tmp* architectural configuration through the `generate_new_point` function. If *tmp* is better than the currently estimated optimum  $\hat{a}_{opt}$  with respect to the EDP, then *tmp* becomes the new optimum. However, if the number of times  $w$  in which *tmp* is worst than  $\hat{a}_{opt}$  becomes  $N_w$ , then  $k$  is incremented by 2 to reduce the window for the generation of the random configurations (see Eq. 3) and thus the search is localized in the neighborhood of the current point.

After a number of  $N_{MAX}$  simulations has been reached,

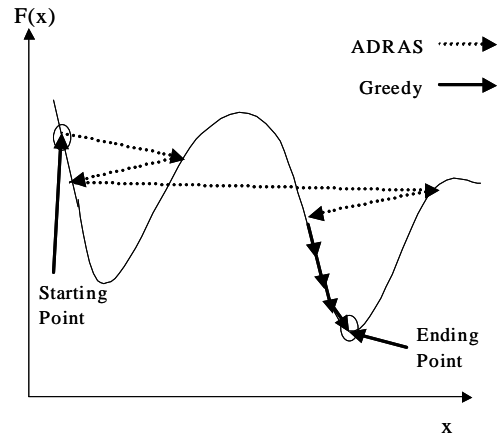


Figure 2: An example of the behavior of the ADGREED algorithm.

the random search is ended and the greedy algorithm starts by looking at all the *tmp* points in the one-step neighborhood of  $\hat{a}_{opt}$ . If a *tmp* configuration better than  $\hat{a}_{opt}$  is found, then *tmp* becomes the new optimum. This process is iterated until the  $\hat{a}_{opt}$  does not change.

As shown in Figure 3, in the best case, the number of simulations required by the ADGREED algorithm is equal to  $(N_{MAX} + 2 * N)$ . The term  $N_{MAX}$  is required by the adaptive random search phase of the algorithm, while the term  $2 * N$  (where  $N$  is the number of parameters of the architecture) is required by the greedy search to analyze the neighborhood region of only the last point.

In the worst case, the number of simulations required depends on the greedy algorithm and could become equal to the full space size, that is  $|\mathcal{A}|$ . However, in our experiments, the average performance is near to the best case.

#### 5. CASE STUDY

In this section we present the experimental results obtained by applying the ADGREED algorithm to the Lx microprocessor, a VLIW architecture jointly designed by STMicroelectronics and Hewlett Packard Labs. Lx [17] is a scalable and customizable 4-issue VLIW processor technology platform jointly designed by Hewlett-Packard and STMicroelectronics.

The system is characterized by I- and D-caches integrated on-chip and an off-chip DRAM memory. The memory hierarchy is connected by means of a point-to-point bus topology; however, in our investigations, we will neglect the power consumption of on-chip buses, since experimental results on power estimation have shown a very little impact on the total power consumption.

For the Lx processor, we considered the variation of the following design parameters: I- and D-cache size ( $ics, dcs$ ), I- and D-cache block size ( $ibs, dbs$ ) and associativity ( $ia, da$ ). The range of considered values are:

- $S_{ics}, S_{dcs} = \{ "2K", "4K", "8K", "16K", "32K", "64K" \}$
- $S_{ibs}, S_{dbs} = \{ 8B, 16B, 32B \}$
- $S_{ia, da} = \{ 1, 2, 4, 8 \}$

```

ADGREED( $N_w, N_{MAX}, a(0)$ )  $\rightarrow \hat{a}_{opt}$ 
/* adaptive random search phase */
 $A = \{\}$ ; /* set of already visited points */
 $\hat{a}_{opt} = a(0)$ ;
 $k = 1; w = 0$ ;
for ( $t=0; t < N_{MAX}; t++$ ) do
   $tmp = \text{generate\_new\_point}(A, \hat{a}_{opt}, k)$ ;
   $A = A \cup \{tmp\}$ ;
  if ( $EDP(tmp) < EDP(\hat{a}_{opt})$ ) then
     $\hat{a}_{opt} = tmp$ ;
  else
     $w++$ ;
    if ( $w == N_w$ ) then
       $w = 0$ ;
       $k = k + 2$ ;
    end if
  end if
end for
/* deterministic greedy search phase */
repeat
  for all ( $tmp \in \text{neighborhood}(\hat{a}_{opt}), tmp \notin A$ ) do
     $A = A \cup \{tmp\}$ ;
    if ( $EDP(tmp) < EDP(\hat{a}_{opt})$ ) then
       $\hat{a}_{opt} = tmp$ ;
    end if
  end for
until ( $\hat{a}_{opt}$  is stable)

```

Figure 3: Pseudo-code of the ADGREED algorithm

So the design space size  $|A|$  is composed of 5184 system configurations for each given application.

The experimental workload we used is based on the Mediabench suite [18], that is composed of the following applications: ADPCM, EPIC, G721, GSM, JPEG, MPEG2, and PEGWIT.

To gather information about the delay, an Lx ISS (Instruction Set Simulator) has been used [17]. The Lx ISS simulator profiles the memory accesses of each application and computes a total execution delay.

The results of the simulations have been used to compute the overall energy consumption by applying the Lx instruction-level energy model [19] and the CACTI cache energy models presented in [20]. The energy dissipated by the main memory has been derived from [21].

We applied a full EDP search analysis to each application of the Mediabench suite to understand the real behavior of the EDP for the target architecture. Figure 4 show the shape of the EDP function versus the instruction cache size for different values of the instruction cache block, while the data cache configuration is fixed.

The resulting EDP curve is convex. This is due to the fact that increasing the cache size, while keeping a constant block size, tends to minimize the delay since more blocks are present in the cache and the miss probability decreases. However, increasing the cache increases also the energy consumption and this implies a trade-off between energy consumption and performance. A similar convex behavior has been observed for the EDP varying I-cache associativity and I-cache size for the same benchmarks.

In this section, we describe the results obtained using ADGREED to find the optimal EDP configuration for the Lx architecture.

We performed different runs of ADGREED by giving different values to its parameters:  $70 \leq N_{MAX} \leq 100$  and  $18 \leq N_w \leq 35$ . We found that increasing the number of

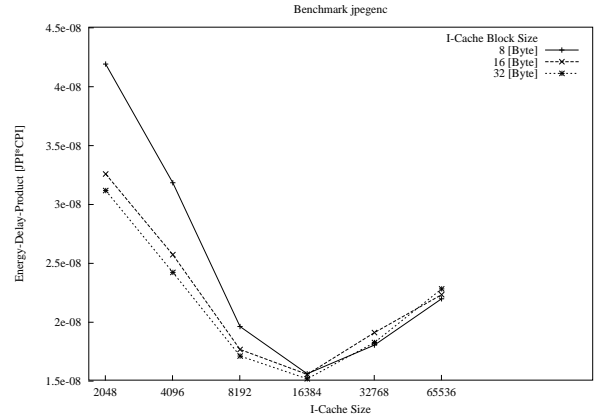


Figure 4: EDP for the Lx system varying I-cache size and I-cache block size for the JPEG encoder.

simulations  $N_{MAX}$  leads to a decreased probability to arrive into a sub-optimal region and, thus, a decreased error. However, for what concerns the  $N_w$ , no significant dependence was found on the global error.

Table 1 shows the percentage error between the global EDP optimal configuration found with the full search analysis and the EDP of the configurations found with ADGREED. The table shows also the number of simulations performed by the ADGREED algorithm to find such a configuration. Each percentage error is an average of the errors obtained with the various values of  $N_{MAX}, N_w$ .

Benchmark	E%	T
adpcmdec	0,16	111
adpcmenc	0,22	111
epic	0,75	113
g721dec	2,26	109
g721enc	1,75	115
gsmdec	0,08	117
gsmenc	1,17	111
jpegdec	0,69	115
jpegenc	1,03	114
mpeg2dec	3,37	116
mpeg2enc	0,79	120
pegwit	0,00	113

Table 1: Average percentage error (E%) with respect to the full search EDP optimum and average number of simulations (T) for each benchmark.

Table 1 shows also that the average speedup of ADGREED with respect to the full search is of more than one order of magnitude ( $\sim 100$  vs  $\sim 5000$ ). Thus, ADGREED finds a good approximation of the global minimum of EDP with very few simulations. It can also be noted that, while the ADRAS part of ADGREED requires an average of 85 simulations, the GREEDY part performs an average of 15 simulations, i.e., it has an overhead of  $\sim 30\%$  over the ADRAS part.

Table 2 represents the percentage error and the number of simulations for the deterministic algorithms (Gradient, Greedy, Sensitivity) and the Random Search algorithm (ADRAS and ADGREED), with varying parameters (represented in the form  $[N_{MAX}, N_w]$ ) for the Lx architecture.

As we can see from Table 2, the deterministic algorithms

such as Gradient, Greedy and Sensitivity provide an average percentage error higher than the random algorithms such as ADRAS and ADGREED. On the other hand, they present a lower average number of simulations than random algorithms.

Although the ADGREED algorithm performs additional simulations with respect to the pure ADRAS (due to the Greedy part), it presents a lower average error. The results relative to the ADRAS algorithm show an evident dependence on the  $N_w$  parameter. On the contrary, for the ADGREED algorithm it is also evident that the parameter  $N_w$  has not a clear influence on the average error. More in detail, Figure 5 shows, for the ADGREED, how the probability to find the optimal configuration varies with  $N_{max}$  and  $N_w$ . It is evident that the average error decreases as the maximum number of simulations ( $N_{max}$ ) increases while there is no significant dependence on  $N_w$ . This would suggest to use a simplified version of ADGREED in which the parameter  $N_w$  is fixed to 1, corresponding to a pure random search algorithm.

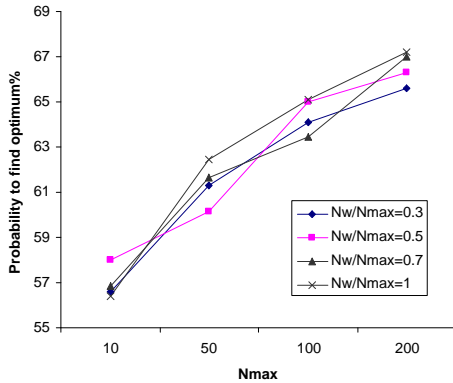


Figure 5: Probability to find optimum by varying  $N_{MAX}$  and  $N_w$  for the ADGREED algorithm

## 6. CONCLUSIONS

In this paper, a modular analysis methodology has been described for the efficient exploration of the architectural parameters at the system-level. The main goal is to find quickly the best architecture of an embedded system by considering at the same time energy and delay, avoiding the comprehensive analysis of the architectural design space. We analyzed an extended adaptive random search algorithm (ADGREED) to efficiently explore the architectural design space. The effectiveness of the proposed exploration methodology has been evaluated through the architectural optimisation of the memory sub-system of a real-word embedded system executing the set of Mediabench benchmarks for multimedia applications. The system is based on a 4-issue 32-bit VLIW (Very Long Instruction Word) processor. The reported experimental results have shown a speed up of up to one order of magnitude of the number of design alternatives analyzed during the exploration phase, with a significant accuracy. We are currently working on the evaluation of different exploration techniques such as genetic algorithms as well as simulated annealing.

ALGORITHM	E%	T
ADRAS [70,18]	3,43	70
ADRAS [70,26]	4,48	70
ADRAS [75,21]	3,29	75
ADRAS [75,30]	4,43	75
ADRAS [85,23]	2,66	85
ADRAS [85,30]	3,40	85
ADRAS [90,25]	2,53	90
ADRAS [90,35]	3,39	90
ADRAS [100,25]	1,73	100
ADRAS [100,35]	2,73	100
ADGREED [70,18]	1,13	100
ADGREED [70,26]	1,04	101
ADGREED [75,21]	1,15	105
ADGREED [75,30]	1,09	105
ADGREED [85,23]	1,00	115
ADGREED [85,30]	1,08	115
ADGREED [90,25]	0,90	120
ADGREED [90,35]	1,01	120
ADGREED [100,25]	0,94	129
ADGREED [100,35]	0,86	129
GREEDY	4,80	62
SENSITIVITY	4,53	47
GRADIENT	32,62	41

Table 2: Algorithms comparison.

## 7. REFERENCES

- [1] N. Vijaykrishnan, 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.
- [2] Y. Li 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.
- [3] 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*, 8(2):129–137, Apr. 2000.
- [4] David Brooks, Vivek Tiwari, and Margaret Martonosi. Watch: a framework for architectural-level power analysis and optimizations. In *Proceedings ISCA 2000*, pages 83–94, 2000.
- [5] 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*, 8(3), June 2000.
- [6] Tony D. Givargis, Frank Vahid, and Jörg Henkel. Evaluating power consumption of parameterized cache and bus architectures in system-on-a-chip designs. *IEEE Transactions on Very Large Scale of Integration (VLSI) Systems*, 9(4), August 2001.
- [7] J. K. Kin, M. Gupta, and W. H. Mangione-Smith. Filtering Memory References to Increase Energy Efficiency. *IEEE Trans. on Computers*, 49(1), Jan. 2000.
- [8] L. Benini, A. Macii, E. Macii, , and M. Poncino. Increasing energy efficiency of embedded systems by

- application-specific memory hierarchy generation.  
*Design and Test of Computers*, 17(2):74–85,  
 April–June 2000.
- [9] R. I. Bahar, 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.
  - [10] C. L. Su 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.
  - [11] M. B. Kamble 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.
  - [12] S. E. Wilton and N. Jouppi. An enhanced access and cycle time model for on-chip caches. Technical Report 93/5, Digital Equipment Corporation Western Research Lab., 1994.
  - [13] P. Hicks, 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*, pages 239–242, Monterey, CA, 1997.
  - [14] W.-T. Shiue 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.
  - [15] L. Benini, D. Bruni, M. Chinosi, C. Silvano, V. Zaccaria, and R. Zafalon. A power modeling and estimation framework for vliw-based embedded systems. In *Proceedings of International Workshop-Power And Timing Modeling, Optimization and Simulation, PATMOS'01*, 26–28 2001.
  - [16] Anatoly A. Zhigljavsky. *Theory of global random search*, volume 65. Kluwer Academic Publishers Group, Dordrecht, 1991.
  - [17] P. Faraboschi, 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*, pages 203–213, June 2000.
  - [18] C. Lee, M. Potkonjak, and W. H. Mangione-Smith. Mediabench: A tool for evaluating multimedia and communication systems. In *Proceedings of Micro 30*, 1997.
  - [19] Andrea Bona, Mariagiovanna Sami, Donatella Sciuto, Cristina Silvano, Vittorio Zaccaria, and Roberto Zafalon. Energy Estimation and Optimization of Embedded VLIW Processors Based on Instruction Clustering. In *Proceedings of the 39th Design Automation Conference DAC'02*, pages 886–891, June 2002.
  - [20] S. Wilton and N. Jouppi. CACTI: An Enhanced Cache Access and Cycle Time Model. *IEEE Journal of Solid-State Circuits*, 31(5):677–688, 1996.
  - [21] Itoh Sasaki Nakagome. Trends in low-power ram circuit technologies. *Proceedings of the IEEE*, 83(4), April 1995.