A System-level Framework for Evaluating Area/Performance/Power Trade-offs of VLIW-based Embedded Systems

Giuseppe Ascia, Vincenzo Catania, Maurizio Palesi, Davide Patti

DIIT - University of Catania, Italy

{gascia,vcatania,mpalesi,dpatti}@diit.unict.it

Abstract—Architectures based on Very Long Instruction Word (VLIW) have found fertile ground in multimedia electronic appliances thanks to their ability to exploit high degrees of Instruction Level Parallelism (ILP) with a reasonable trade-off in complexity and silicon costs. In this case Application Specific Instruction-set Processor (ASIP) specialization may require not only manipulation of the instruction-set but also tuning of the architectural parameters of the processor (e.g. the number and type of functional units, register files, etc.) and the memory subsystem (cache size, associativity, etc.). Setting the parameters so as to optimize certain metrics requires the use of efficient Design Space Exploration (DSE) strategies and also simulation tools (retargetable compilers and simulators) and accurate estimation models operating at a high level of abstraction. In this paper we present a framework for evaluation, in terms of performance, cost and power consumption, of a system based on a parameterized VLIW microprocessor together with the memory hierarchy subsystem following execution of a specific application. The framework, which can be freely downloaded from the Internet, implements a number of multi-objective DSE strategies to obtain Pareto-optimal configurations for the system.

I. Introduction

It is widely accepted nowadays that the use of Application Specific Instruction-set Processors (ASIP) in embedded systems provides much more flexible solutions than an approach based on ASICs and is much more efficient than using standard processors in terms of both performance and power consumption [8]. With ASIPs it is possible to modify some of the hardware parameters of the processor to generate a customized instance for a specific application domain. To guarantee high performance levels, an ASIP has to exploit the instruction level parallelism (ILP) available in the specific application. Architectures based on VLIW processor, in particular, are currently seen as answering the demand for modern, increasingly complex embedded multimedia applications, given their capacity to exploit high levels of ILP while maintaining a reasonable trade-off between hardware complexity and cost.

In this paper we will focus on the optimization of the architectural parameters of a VLIW processor, together with the memory subsystem, with a view to optimizing area, performance and power consumption for a specific application. This requires two tools: 1) an environment to measure the variables to be optimized following any variation in the architecture being examined, and 2) A Design Space Exploration (DSE) strategy. The contribution we intend to make with this paper covers both these points. We propose a framework (that can be freely downloaded from the Internet [11]) which combines a retargetable ILP compiler and simulators and exploits the state of the art in estimation models with a high level of abstraction, making it possible to evaluate, in terms of area, performance and power consumption, any configuration of a system comprising a parameterized VLIW architecture and a parameterized 2-level memory hierarchy. Using the framework we conduct an extensive exploration of the low-power/high-performance/low-cost design space for a set of typical media and communication applications.

The rest of the paper is organized as follows: Section II presents EPIC-Explorer, the simulation and exploration framework used in the experiments and the models used to estimate the performance indexes considered. Section III gives the experimental results and discusses the area/performance/power trade-off for a set of specific applications representative of multimedia embedded systems. Finally, Section IV summarizes our contribution and outlines some directions for future work.

II. Experimentation Framework

To evaluate and compare the performance indexes of different architectures for a specific application, one needs to simulate the architecture running the code of the application. In addition, to make architectural exploration possible both the compiler and the simulator have to be retargetable. Trimaran [1] provides these tools and thus represents the pillar around which we have constructed EPIC-Explorer [3]. EPIC-Explorer is a framework that not only allows us to evaluate any instance of a platform in terms of area, performance and power, exploiting the state of the art in estimation approaches at a high level of abstraction, but also implements various techniques for exploration of the design space.

The tunable parameters of the framework can be classified in two categories: compilation parameters and architectural parameters. Compilation parameters refer to the possibility of enabling or disenabling speculative execution and hyperblock

0-7803-8736-8/05/$20.00 ©2005 IEEE.

formation. The tunable parameters of the architecture can be classified in three main categories: *register files, functional units* and *memory sub-system*. Each of these parameters can be assigned a value from a finite set of values. A complete assignment of values to all the parameters is a *configuration*. A complete collection of all possible configurations is the *configuration space*.

A configuration of the system generates an instance that is simulated and evaluated for a specific application as follows. The application written in C is first compiled. Trimaran uses the IMPACT compiler system as its front-end. The code produced, together with the High Level Machine Description Facility (HMDES) machine specification, represents the Elcor input. The HMDES is the machine description language used in Trimaran. Elcor is Trimaran's back-end for the HPL-PD architecture. It performs three tasks: code selection and scheduling, register allocation, and machine dependent code optimizations. The Trimaran framework also consists of a simulator which is used to generate various statistics such as compute cycles, total number of operations, etc. Together with the configuration of the system, the statistics produced by simulation contain all the information needed to apply the area, performance and power consumption estimation models. The results obtained by these models are the input for the exploration block. This block implements an optimization algorithm, the aim of which is to modify the parameters of the configuration so as to minimize the three cost functions (area, execution time and power dissipation).

A. Estimation Models

The amount of power consumed by the processor was estimated using an adaptation of the Cai-Lim model [4] to the VLIW processor. The contribution to power consumption made by the memory cache was estimated using the analytical model presented in [7] based on the characterisation performed by Wilton and Jouppi in [13]. A fundamental aspect of the model being considered is that it is based on estimation of the number of transitions for the various circuit elements involved in the activity of the cache. These transitions are estimated using the dynamic statistics from the simulations and the equations described in [7]. The main memory energy is based on the model in [12] and assumes a per main memory access energy of $4.95 \times 10^{-9} J$ based on the data for the Cypress CY7C1326-133 memory chip. The energy consumption of the buses depends on the switching activity on the bus lines and the interconnect capacitance of the bus lines (with off-chip buses having much larger capacitive loads than on-chip buses). The contribution towards power consumption made by the interconnection system was calculated by counting the number of transitions on the bus lines and applying the formula $P_{bus} = 1/2 V_{dd}^2 \alpha f C_l$ where V_{dd} is the supply voltage, α is the switching activity, f is the clock frequency and C_l is the capacity of a bus line.

The area occupied by the processor with varying architectural and micro-architectural parameters was estimated using the analytical model proposed by Miyaoka *et al.* in [10]. To estimate the area occupied by the caches we used the model described in [13].

The performance statistics produced by the simulator are expressed in clock cycles. To evaluate the execution time it is sufficient to multiply the number of clock cycles by the clock period. This was set to 200MHz, which is long enough to access cache memory in one single clock cycle.

A more detailed analysis, which discusses adaptation of the models to the system and interfacing with the Trimaran infrastructure can be found in [3].

B. Exploration Strategies

EPIC-Explorer implements various multi-criteria exploration algorithms which provide an approximation of the Pareto-optimal set. The current distribution implements four exploration techniques. The first (*DEP*) is the one proposed by Givargis *et al.* in [6] and consists of clustering dependent parameters and then carrying out an exhaustive exploration within these clusters. The second technique (*GA*), proposed by Ascia *et al.* in [2], uses genetic algorithms as optimization tools. The third (*SA*), proposed by Fornaciari *et al.* in [5] uses sensitivity analysis to reduce the space of exploration from the product of the cardinalities of the sets of variation of the parameters to their sum. Finally, the fourth technique (*PBSA*), is a multi-objective extension of *SA*.

Each of these approaches features a different ratio between the accuracy of the solutions found (distance between the approximated Pareto-front and the Pareto-optimal front) and efficiency (the simulation time required to complete the exploration).

III. Experiments

In the following subsections we will analyze the impact of architectural and compilation parameters on the performance indexes considered. The class of applications being considered belongs to the MediaBench suite and represents quite a broad spectrum of the possibilities of using a VLIW architecture in an embedded multimedia environment.

A. Impact of the Functional Units on Parallelism

Given the principle on which an architecture based on a VLIW processor works, the number and type of functional units affects the way in which the compiler can schedule the operations in each long instruction. The presence of several instance of a certain functional unit, for example, makes it possible to schedule several operations using the unit at the same clock cycle. The experiments carried out, however, show that even when the number of functional units increases there is an inherent limit to the degree of parallelism that can be achieved, which is specific to each of the applications investigated. The presence of conditional branches, for instance, is one of the factors that most limit the possibility of parallelizing instructions by the scheduler. An essential technique to overcome this limit is hyperblock formation [9].

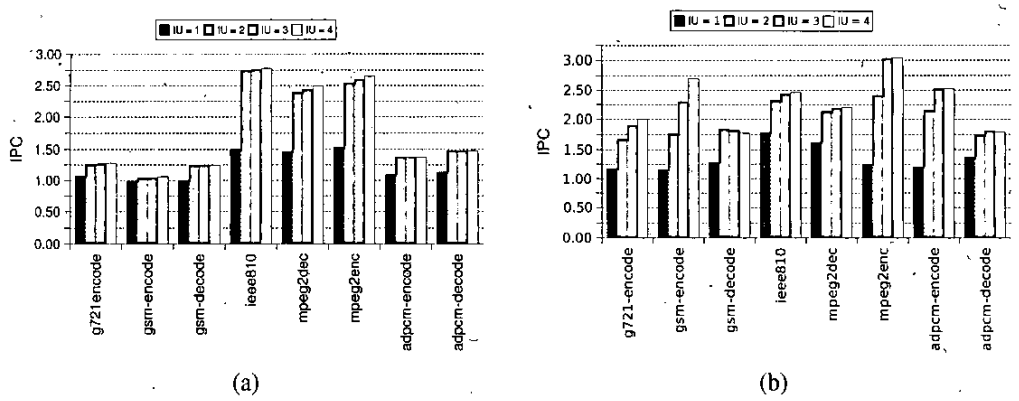

Fig. 1. Instructions per cycle for different number of integer units. Without hyperblock formation (a) and with hyperblock formation (b).

To give a practical example of this effect, we will refer to Figure 1 which gives the IPC for varying numbers of integer units with and without hyperblock formation. As can be seen, a variation in the number of units for integer operations affects the average number of instructions executed per cycle (IPC). As we are focusing on the effect of the functional units, we are momentarily neglecting the impact of a variation in the memory hierarchy on the actual degree of parallelism achieved.

We have confined this example to a variation in the integer units because this type of operation is widely used in each of the benchmarks being investigated. This simple example already shows that the impact on the number of operations executed per cycle is not uniform in the various applications. This and other similar tests have shown that the degree of parallelism that can be achieved strictly depends on the application source code and has to be analyzed on a case-by-case basis. Of course, when we search for optimal configurations for the architecture as a whole, we will also have to take into account variations in the number of units of other types, and the effect of this variation, as we shall see, will be much more complex and less predictable.

B. Design Space Exploration

As said previously, a key issue to be taken into consideration in embedded multimedia applications is that performance optimization is often not the only aim. In the last few years, for example, it has become clear that minimization of power consumption is a critical factor in evaluating the correctness of a design decision. Maximization of performance and minimization of power consumption are two requisites that often clash with each other. There is thus no single optimal configuration: it is necessary to identify a set of Pareto-optimal configurations that will represent the best trade-offs the architecture can offer. The design space considered, as seen previously, is of a totally prohibitive size, which precludes any attempt at exhaustive evaluation of all the possible design alternatives. It is therefore necessary to have techniques that will permit "intelligent" exploration of the space of possible configurations so as to obtain, in the initial design stages, the configurations that are more likely to meet the requirements of the system. As the number of alternatives thus obtained is lower, the designer can then assess them individually using more accurate tools and estimation models at a lower level.

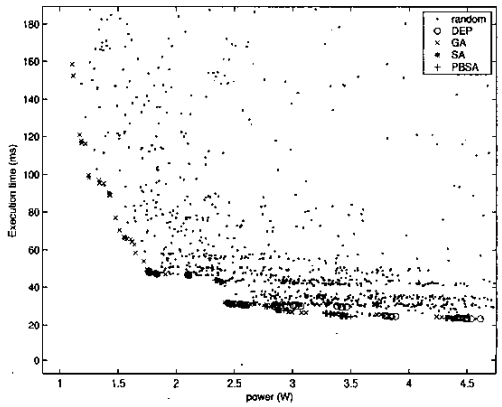

Fig. 2. Exploration of the design space described in Table I. Pareto-fronts obtained with the various exploration approaches.

Figure 2 shows the approximated Pareto-optimal front (power vs. execution time), obtained by application of the various exploration strategies implemented in EPIC-Explorer, relating to the design space described in Table I and the benchmark `adpcm-encode` which implement an Adaptive Differential Pulse Code Modulation algorithm[1]. The figure also shows the points obtained by evaluation of 10,000 configurations obtained from random sampling of the design space. As can be seen, with a limited number of simulations, the solutions obtained by the exploration techniques achieve an excellent approximation of the Pareto-front. Finally, Figure 3 shows the Pareto-surface (area/power/execution-time) for `mpeg2-decode` benchmark.

[1]For reasons of space we cannot show all the results. Please refer to [11] for a complete list of the tests performed.

TABLE I
SPACE OF VARIATION OF THE PARAMETERS.

Parameter	Parameter space
GPR	32,40,64
PR	32,40,64,128
CR	32,40,64,128
BTR	8,12,16,32
Integer Units	1,2,3,4
Memory Units	1,2,3
Branch Units	1,2,3
L1D/I cache size	2k,...,64KB
L1D/I cache block size	16B,32B,64B,128B
L1D/I cache associativity	4,8,16
L2U cache size	128k,...,512KB
L2U cache block size	8B,16B,32B,64B,128B
L2U cache associativity	1,2,4,8,16
Space size	3.627×10^8

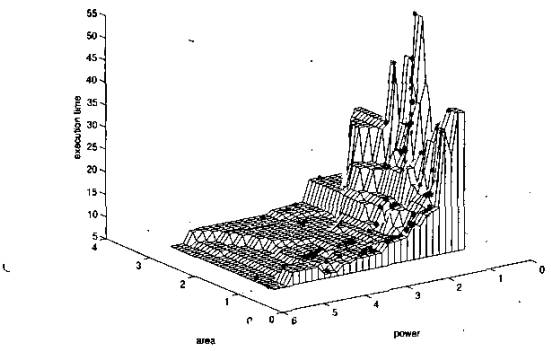

Fig. 3. Exploration of the whole configuration space. Trade-off area/power/execution-time.

IV. CONCLUSIONS

In this paper we have presented EPIC-Explorer, a framework to evaluate a system comprising a parameterized VLIW microprocessor and a parameterized memory hierarchy, in terms of area, performance and power consumption. Since the architecture is parametric as regards both the units of the VLIW processor and the memory hierarchy, it provides a broad spectrum of design alternatives. The price of this is a difficult, if not impossible, search for the optimal configurations based solely on empirical suppositions. For this reason the framework implements a number of design space exploration strategies, each featuring a different trade-off between the computational complexity of completing the exploration and the quality of the solutions obtained. Tests carried out on a set of specific embedded multimedia applications confirmed the flexibility of the platform. Its field of application is considerable: it can be used to evaluate the impact of a VLIW system's architectural parameters on area, power and performance, to test new design space exploration strategies and to investigate the several compilation options offered by retargetable compilers, etc. EPIC-Explorer can be freely downloaded from the internet [11].

REFERENCES

[1] An infrastructure for research in instruction-level parallelism. `http://www.trimaran.org/`.

[2] G. Ascia, V. Catania, and M. Palesi. An evolutionary approach for pareto-optimal configurations in soc platforms. In K. A. Publishers, editor, *SOC Design Methodologies*, 2002.

[3] G. Ascia, V. Catania, M. Palesi, and D. Patti. EPIC-Explorer: A parameterized VLIW-based platform framework for design space exploration. In *First Workshop on Embedded Systems for Real-Time Multimedia (ESTIMedia)*, Newport Beach, California, USA, Oct. 3–4 2003.

[4] G. Cai and C. H. Lim. Architectural level power/performance optimization and dynamic power estimation. In *Cool Chips Tutorial colocated with MICRO32*, pages 90–113, Nov. 1999.

[5] W. Fornaciari, D. Sciuto, C. Silvano, and V. Zaccaria. A sensitivity-based design space exploration methodology for embedded systems. *Design Automation for Embedded Systems*, 7:7–33, 2002.

[6] T. Givargis, F. Vahid, and J. Henkel. System-level exploration for Pareto-optimal configurations in parameterized System-on-a-Chip. *IEEE Transactions on Very Large Scale Integration Systems*, 10(2):416–422, Aug. 2002.

[7] M. B. Kamble and K. Ghose. Analytical energy dissipation models for low power caches. In *IEEE International Symposium on Low Power Electronics and Design*, pages 143–148, Aug. 1997.

[8] K. Keutzer, S. Malik, and A. R. Newton. From ASIC to ASIP: The next design discontinuity. In *IEEE International Conference on Computer Design: VLSI in Computers and Processors*, pages 16–18, Sept. 2002.

[9] S. A. Mahlke, D. C. Lin, W. Y. Chen, R. E. Hank, and R. A. Bringmann. Effective compiler support for predicated execution using the hyperblock. In *International Symposium on Microarchitecture*, pages 45–54, Dec. 1992.

[10] Y. Miyaoka, Y. Kataoka, N. Togawa, M. Yanagisawa, and T. Ohtsuki. Area/delay estimation for digital signal processor cores. In *Asia and South Pacific Design Automation Conference*, pages 156–161, 2001.

[11] D. Patti and M. Palesi. Epic-explorer. `http://epic-explorer.sourceforge.net/`, July 2003.

[12] W.-T. Shiue and C. Chakrabarti. Memory exploration for low power, embedded systems. In *36th ACM/IEEE Conference on Design Automation Conference*, pages 140–145, New Orleans, Louisiana, United States, 1999.

[13] P. Shivakumar and N. P. Jouppi. CACTI 3.0: An integrated cache timing, power, and area model. Technical report, COMPAQ Western Research Lab, Palo Alto, California 94301 USA, 1999.