Power and performance exploration of embedded systems executing multimedia kernels

M. Dasygenis, N. Kroupis, K. Tatas, A. Argyriou, D. Soudris and A. Thanailakis

Abstract: The memory subsystem in modern embedded programmable architectures executing multimedia applications consumes a significant amount of energy. The designer has to take this fact into consideration, together with the system performance, in order to design devices portable or otherwise. An exploration approach for optimising the power and performance of the data-memory hierarchy as well as the instruction memory in the early system-design phase, is introduced. A power- and performance-efficient data-memory hierarchy is obtained by applying data-reuse transformations in a high-level description of the application, whereas the instruction-memory power optimisation, of the selected optimal data hierarchies of the previous step, is achieved by using a suitably selected cache memory. Furthermore, two cache energy models, namely the high-level power model and the architecture-dependent power model, are introduced. The experimental results, obtained with four well known motion-estimation kernels, provide an insight on the trade-offs among algorithm performance and energy consumption, comparing memory hierarchies with and without an instruction cache for the ARM programmable core. Comparisons results are also provided for choosing an optimal cache memory size.

1 Introduction

Embedded systems have reached a significant turning point in the last decade, from low-performance products such as wristwatches and calculators to high-throughput and computation-intensive products such as portable multimedia terminals or personal digital assistants. The introduction of these devices to the consumer market brought to the surface a characteristic, which had previously been ignored: energy dissipation. Gradually, engineers presented methodologies for designing efficient circuits, not only in terms of area and in terms of performance, as in the past, but also in terms of low energy consumption.

Multimedia are data-intensive applications, meaning that large amounts of data are transferred between the large off-chip background memory and the on-chip processing core to be computed and processed. For this reason, the data transfers play a dominant role in the power and area cost of such system realisations. Providing some measurement information as feedback to a multimedia-system designer, at the very early stages of design exploration, permits global algorithmic implementation trade-offs and results into large energy savings. The problem of memory management and optimisation in the embedded systems, as well as the fundamental research contributions in that field, were described extensively by Panda *et al.* [1]. To design power- and performance-efficient embedded systems realising multimedia applications is relatively new. The majority of researchers in the multimedia domain are trying to optimise the data transfers [2–7] which can be achieved using a number of algorithm transformations that can reduce the accesses to off-chip background memory. In addition, few researchers target instruction-memory optimisations [8–12].

Although the processing power of current embedded processor cores is many MIPS, allowing applications of high computational complexity to be executed, a bottleneck is imposed by the memory subsystem, owing to its longer access time. Even contemporary digital signal processors are facing a great challenge to perform the operations of video and audio encoding, in some profiles of the MPEG applications [13]. Moreover, the energy cost per memory access is responsible for the high energy consumption in data-dominated applications [2]. Pioneering work on the application of algorithm transformations to reduce power in data-dominated applications has been done at IMEC [2]. Recently, IMEC researchers [14] have expanded their methodology to include hardware caches and techniques to provide placement of the data elements, ignoring the effect of instruction-memory power on their programmable processors. A methodology to find the optimal data-cache and register configuration, according to specific time and power constraints, was proposed in [15], but their model is based on solving linear equations of the manually extracted dependence graph of the data elements, making it very time consuming for the designer in real-life applications. Masselos *et al.* [3] suggested the use of high-level algorithmic platform-independent transformations which yield significant reductions of data-memory power consumption. Benini *et al.* [16] presented a promising methodology for automatic generation of the

IEE Proceedings online no. 20020468

DOI: 10.1049/ip-cdt:20020468

Paper first received 26th October 2001 and in revised form 25th April 2002

M. Dasygenis, N. Kroupis, K. Tatas, D. Soudris and A. Thanailakis are with the VLSI Design and Testing Center, Department of Electrical and Computer Engineering, Democritus University of Thrace, 67100 Xanthi, Greece

A. Argyriou is with the School of Electrical and Computer Engineering, 325716 Georgia Institute of Technology, Atlanta, GA, 30332-1070, USA

application specific memory (ASM) of low-energy embedded systems, using a novel memory architecture, but they focused only on the power consumption of data memory, without using high-level transformations to achieve more optimal ASMs. In [17], a new technique for allocation of D-cache and scratchpad memories from a common on-chip pool of memory was presented. A software mechanism determined the memory size of each memory, taking into account the characteristics of the application being considered. However, to reach an efficient memory allocation, a number of hardware modifications were necessary. In other words, standard memories cannot be used. Finally, other researchers [18] have presented a mathematical framework for analysing the degree of reusability in nested loops and the appropriate transformations, to exploit this, but this approach is restricted to perfectly nested loops of DSP applications.

The efficient design of the instruction memory is strongly related to performance optimisation and energy consumption. In particular, Kin *et al.* [8] and Bellas *et al.* [9] proposed architectures using a small cache in front of an I-cache hierarchy (named filter cache [8] or loop cache [9]). More specifically, the first technique aims at power-consumption reduction, while the second aims to increase, performance. Also, additional hardware modifications are needed to implement all these schemes. Although they exhibit these advantages, an extra overhead is added to the designer, due to the compiler's and application code's modifications. In [10], a methodology targeting software transformations and driven by analytical equations was proposed, but the tough task of code transformation was done manually. Finally, in [11] a technique based on instruction-code relocation, utilised the direct-mapped L1 (I-cache) efficiently. However, an overhead is needed because of the presence of to additional preprocessing steps.

In this paper, an exploration approach for determining the low-energy memory subsystem for embedded programmable processors, executing multimedia applications, is introduced. To optimise this subsystem, a combination of transformations has to be applied in the high-level description of the application, which gradually refines the code, together with careful selection of an appropriate instruction cache (I-cache). The data-memory hierarchy exploration has, as a result, a pool of candidate on-chip memory hierarchies that will be used in the I-cache exploration, to design power- and performance-optimal architectures. Since our target architecture consists of programmable processors, instruction-memory (I-mem) energy consumption plays a significant part in the overall energy budget. To alleviate this, a number of different I-cache configurations were analysed and the most promising candidates were selected. Applying the steps of the exploration approach, one can specify the optimal cache size for each multimedia kernel, considering two cache-energy models, namely the high-level model and the architecture-dependent one.

2 Proposed exploration approach

The target architecture model consists of:

(i) multiple embedded programmable processing cores N, each of which has its individual on-chip I-mem, and I-cache memory, and
(ii) on-chip and off-chip data memory hierarchy (Fig. 1).

Here, the methodology of reducing memory energy consumption is analysed for an architecture consisting of only one programmable ARM processor [19] ($N=1$), and experimental results for four multimedia kernels are presented to validate it. Since the considered multimedia kernels consist of perfectly-nested loops, it can be partitioned into N ($N>1$) nested loops, each of which is executed by a processor, applying the partitioning technique of locally serial globally parallel (LSGP) [20]. More specifically, dividing the range of values of the outermost loop index into N parts, N partitioned multimedia kernels are derived, each of which is executed by a single core. It has been proved [21] that, using the LSGP partitioning technique, the data-memory energy consumption was reduced, while the instruction-memory energy consumption was increased.

We aim to determine of the optimal data-memory hierarchy for reducing energy, reducing the number of off-chip transfers, and of the optimal I-cache memory for reducing instruction-memory energy consumption. This

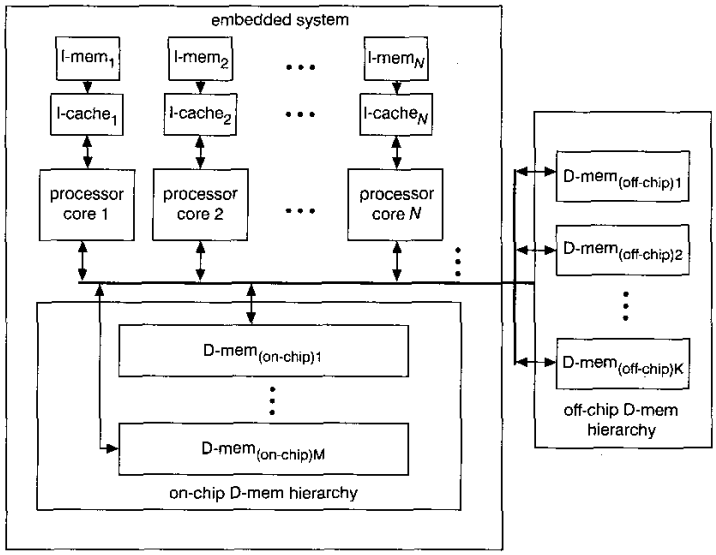

Fig. 1 *Target architecture model*

problem of energy-consumption estimation can be divided into two distinct tasks of equal importance, which a designer has to take into account:

(i) the corresponding data-memory hierarchy (D-mem), and
(ii) the corresponding instruction memory with the presence or absence of I-cache.

The first task has already been studied in detail [7]. In the present paper, the aim is to optimise the whole memory subsystem of the target architecture, and not only the data memory.

The flow of the proposed exploration approach is depicted in Fig. 2. This methodology aims at mapping a behavioural description in a high-level software language (for example C) on an embedded core with a dedicated data-memory hierarchy and a level 1 ($L1$) I-mem cache. The entry point in the methodology is a C-language description of a multimedia kernel. This algorithm is then transformed using high-level software-optimisation techniques, and is fed to the ARMulator [19], an ARM processor emulator. Depending on the transformations used, one can calculate the energy and area of the data-memory hierarchy, or the ARMulator can be used to extract a memory-access trace that can be filtered to extract data-address information, that can be fed to a cache simulator to extract cache statistics. In other words, following the steps of the proposed exploration scheme, the energy consumption of the data-memory hierarchy and instruction memory with and without cache memory are calculated.

2.1 Memory-exploration approach

A typical data-memory optimisation methodology consists of many steps, some of which include a set of software techniques aiming at the optimisation of the algorithm's kernel source code, in terms of power and performance, or acting as enabling transformations for the other types of transformations. After some preprocessing/pruning work on the source code, a first step is to apply a series of high-level transformations, which are: (i) data-flow transformations [22], (ii) loop/control-flow transformations [1, 18] and (iii) data-reuse transformations [1, 23]. In particular, data-flow transformations 'enable' the efficient use of the loop/control-flow transformations as well as the data-reuse transformations. A next step is to map the transformed algorithm to the physical memories, i.e. memory allocation and assignment [1, 2]. Eventually, the application of the data memory-methodology results in a number of energy- and/or performance-optimised expressions of the original algorithm, each of which corresponds to a specific data memory hierarchy.

Although the programmable-processor approach is more flexible for implementations than the custom- (or dedicated-) hardware approaches, its main drawback is the fact that instruction-energy consumption plays a significant role in the total energy budget of the embedded system and it should therefore be reduced. Indeed, after detailed exploration of the impact of data-reuse transformations on a series of multimedia kernels, considering multiple programmable processor platforms, it was concluded that the energy consumption of instruction memory cannot be ignored [6, 7].

Taking into account the target architecture, and using the memory-energy-consumption Landman's model [24], measurements of the efficiency of the algorithm, in terms of energy and performance, related to the ARM core can be taken. In particular, Landman presented a black-box capacitance model for architectural energy analysis. Given its capacitance, the energy consumption is calculated by:

$$E = \frac{1}{2} V_{dd}^2 (C_{read} \times \#readaccesses + C_{write} \times \#writeaccesses) \quad (1)$$

where C_{read} and C_{write} denote the capacitances for read and write operation, and $\#readaccesses$, $\#writeaccesses$ are the number of read and write operations, respectively. The values of C_{read} and C_{write} are specified by the model itself, while the number of accesses depend on the chosen application, and they are calculated by the designer.

To measure the performance, it is assumed that on-chip data-memory accesses need one processor cycle per access, while I-mem accesses need two processor cycles per access. Evaluation of cache statistics requires a cache simulator to account for the dynamic effects caused by the cache-replacement mechanism, the different degrees of associativity, the write policy etc. The next step in the exploration procedure (Fig. 2) is transformation of the memory trace of the ARMulator into a format compatible with the Dinero cache simulator [25], and the collection of statistical data for various cache parameters. These statistics are used as input to cache models (i.e. high-level and architecture-dependent energy-cache models), which have been developed.

2.2 Memory model

To estimate the difference in performance between using and not using an I-cache, a best case scenario is considered, where every access to I-mem has a penalty of two processor cycles, whereas an access to a directly mapped I-cache has no such penalty (i.e. every access to I-cache needs one

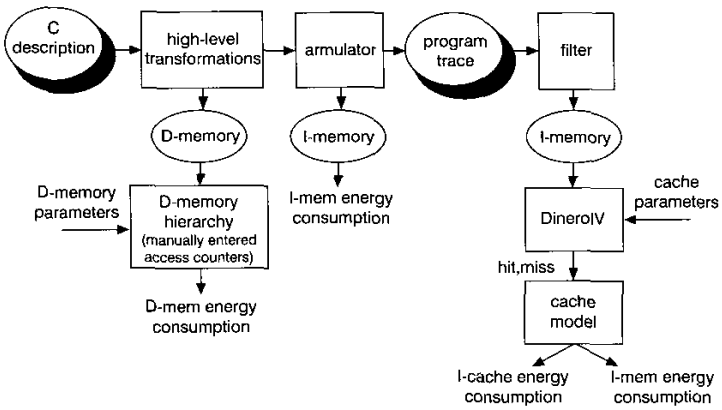

Fig. 2 *Proposed exploration approach: estimation of D-mem, I-mem and I-cache energy consumption and performance*

processor cycle). Direct-mapped cache is used for achieving low power consumption, as reported by [8, 11], because associativity increases the amount of data and control information read out of cache arrays; thus associativity is proportional to the power reduction. The improved hit rate does not usually fully compensate for the increased energy per reference. If an I-cache has degree of associativity 2, 4, or 8, the penalty per access is 1.1, 1.12, and 1.14, respectively [15]. If there is a cache miss, the penalty is four processor cycles, because one cycle is required in the tag-comparison logic, two cycles are needed for access of the I-mem, and one cycle is required for transfer data from the I-mem to the processor core.

To perform measurements in terms of performance and energy consumption, a number of models are assumed for each parameter. More specifically, to measure the performance of an embedded system, accesses to on-chip memories of the data hierarchy were considered to have a penalty of one processor cycle per access, whereas accesses to off-chip data memory have a penalty of 20 processor cycles per access [17]. Hence, to measure the performance of an embedded system consisting of N embedded cores, each one having its own individual instruction memory, instruction cache and data-memory methodology, the following equations are used:

$$core\_performance_i = C_{P_i} + C_{D_i} + C_{I_i} \quad (2)$$

$$system\_performance = \max\{core\_performance_i\}_{i=1\ldots N} \quad (3)$$

where C_{P_i} is the number of programmable processor-core cycles required by the ith programmable core; C_{D_i} is the number of processor cycles required to access the data-memory hierarchy; C_{I_i} is the number of processor cycles required to access the instruction memory and the instruction cache (if it exists); $core\_performance_i$ is the total number of cycles of the ith core; and $system\_performance$ is the total performance of the embedded system.

Energy consumption is the second parameter that will be modelled for measurements. Generally, the total system energy consumption P_{total} during the execution of an application is given by the equation

$$P_{total} = P_{data} + P_{instruction} \quad (4)$$

$$P_{instruction} = P_{I\text{-}mem} + P_{I\text{-}cache} \quad (5)$$

where P_{data} is the energy consumption in the data-memory hierarchy and $P_{instruction}$ is the energy consumption in the instruction-memory hierarchy. The latter consists of the energy consumption in the instruction memory $P_{I\text{-}mem}$ and in the instruction cache $P_{I\text{-}cache}$. In the present model, only the energy consumption in the memory subsystem is considered, because 50–80% of the energy cost in programmable embedded cores executing multimedia applications in real time is due to memory traffic caused by transfers between the processor cores and the memory subsystem [15].

To calculate the energy in the data-memory hierarchy, the access frequency in every level of the data memory hierarchy on-chip and off-chip is used, along with the energy consumption estimate for read and write operation and the number of words and the word length of every level [7]. Using Landman's model [24], the computation of total energy consumption in the data-memory hierarchy can be carried out. To estimate the energy consumption in the instruction memory, two energy models have been developed: (i) the high-level architecture-independent cache model, and (ii) the architecture-dependent cache model, which will be described in Sections 2.3.1 and 2.3.2.

2.3.1 High-level architecture-independent cache model

To estimate the energy consumption in the I-cache according to the high-level model, without any knowledge of the architecture that the application is going to be mapped onto, one uses only the number of accesses in the cache. The number of accesses to the cache memory $I_{cache\_accesses}$ and to the instruction memory $I_{mem\_accesses}$ depends on the miss ratio m and the number of accesses n to the instruction memory, without using the I-cache, and their corresponding values are obtained from

$$I_{cache\_accesses} = (1 + m) \times n \quad (6)$$

$$I_{mem\_accesses} = n \times m \quad (7)$$

The constant '1' in (6) is justified by the fact that the processor core 'requires', for the specific algorithm, n accesses to the instruction memory, regardless of whether an I-cache has been inserted or not. When a cache exist, all the n accesses come from the cache. However, there exists a miss rate of m, which means that $m \times n$ accesses are required by I-mem, resulting in extra $m \times n$ accesses in the I-cache. Summing up, the I-cache accesses are: $1 \times n + m \times n = (1 + m) \times n$, whereas the I-mem accesses are: $m \times n$. Using an appropriate method, the parameters of $I_{cache\_accesses}$ and $I_{mem\_accesses}$ in (1), the energy consumption in I-cache and in I-mem, can be calculated. The model described above does not take into account the energy dissipation in the tag comparator, the steering logic, the cache-control logic, and the sense amplifiers, because they are regarded as minor sources of energy dissipation. Although, the high-level model does not provide accurate estimations, it is very useful for fast estimations.

2.3.2 Architecture-dependent cache model

To validate the previous model, an architecture-dependent model was developed, which is an extension of [26]. This model has been extended by taking into account the impact of total number of accesses and miss/hit rate, during a computation of the motion vectors of two successive frames. The model of [26] was developed for 0.8 μm CMOS process.

The energy consumption $P_{I\text{-}cache}$ of a cache (5) during the computation of the motion vectors of two successive frames is given by

$$P_{I\text{-}cache} = E_{miss}\ m\ n + E_{hit}\ h\ n \quad (8)$$

where E_{miss} and E_{hit} are the energy consumption per miss and hit, respectively, m and h are the cache miss and hit rate, respectively, as calculated by the Dinero tool, and n is the number of instructions executed. Comparing with the formulas provided by [26], a more precise model is proposed by also taking into account the variables n, m and h, which were omitted in [26]. Because m and h are mutually exclusive, one may substitute h with $(1 - m)$, resulting in

$$P_{I\text{-}cache} = n[E_{miss}\ m + E_{hit}\ (1 - m)] \quad (9)$$

To compute E_{miss} and E_{hit}, it is necessary to determine the circuits of cache that are active during a hit or miss state, respectively, calculating their corresponding energy dissipation. The hit energy E_{hit} is the sum of the energy on the

decoding path $E_{decoding\_path}$ of the cache and the energy on cell array E_{cell_array}[26]:

$$E_{hit} = E_{decoding\_path} + E_{cell\_array} \quad (10)$$

$$E_{decoding\_path} = \alpha \times Addr\_bus\_bs \quad (11)$$

$$E_{cell\_array} = \beta \times Word\_line\_size \times Bit\_line\_size \times Bit\_line\_sb \quad (12)$$

The scaling factors α and β have been found by the authors of [26] to be 0.001 and 2 respectively, using heuristic methods. *Addr_bus_bs* is the number of bit switches on address buses per instruction, *Word_line_size* is the number of memory cells in a word line, while *Bit_line_sb* is the number of switching bit lines per instruction.

The miss energy E_{miss} is the sum of the energy on the decoding path $E_{decoding\_path}$, the energy on the cell array E_{cell_array} and the energy on the input/output path E_{i/o_path}, where the scaling factor χ [in (14)] has been found to be 20, for VLSI implementation in 0.8 μm CMOS process:

$$E_{miss} = E_{decoding\_path} + E_{cell\_array} + E_{i/o\_path} \quad (13)$$

$$E_{i/o\_path} = \chi \times Word\_line\_size \times bit\_line\_size \quad (14)$$

Finally, the energy consumption $P_{I\text{-}mem}$ (5) is computed by means of the high-level architecture-independent energy model.

3 Experimental results

To demonstrate the high competence of the combined I-mem and D-mem optimisation, four well known motion-estimation (ME) kernels [27] are used: (i) full search (FS), (ii) hierarchical search (HS), (iii) parallel hierarchical one-dimensional search (PHODS), and (iv) three-step logarithmic search (3SLOG). These algorithms are some of the fundamental multimedia algorithmic cores that are in use in many multimedia systems.

Table 1: Cache-memory-size exploration experimental results and selection of the optimal cache size

Energy consumption (%)			
Algorithm	Cache memory size		
	128	258	512
FS	−38	**−60**	−42
HS	−34	−60	**−67**
PHODS	−18	−33	**−38**
3SLOG	−26	**−87**	−39

Taking into account the original source-code description of FS, HS, PHODS and 3SLOG, first global-loop transformations (merging of imperfect nested loops) were first applied for HS and PHODS kernels to permit the application of data-reuse transformations. Further, for all kernels 21 data-reuse transformations were applied for optimising their data-memory hierarchy. These transformations were presented for FS in [6] and for the remaining kernels in [7].

The experiments were carried out using the luminance component of QCIF frames (144 × 176 pixels) format of a sequence of frames with the name 'Akiyo'. A reference window was selected to include 15 × 15 ($2p+1 \times 2p+1$, where the search space $p=7$) candidate blocks, while B blocks of 16 × 16 pixels were considered. The original

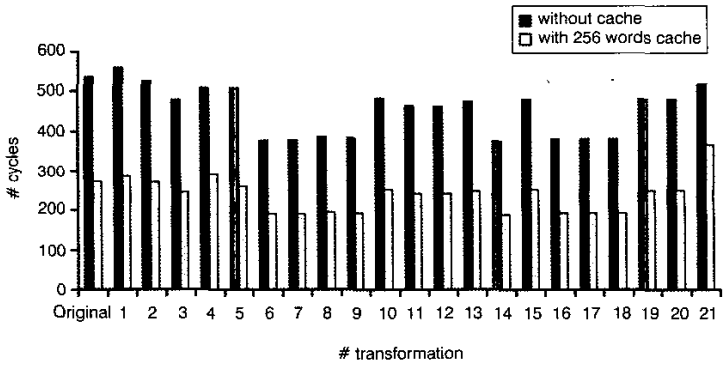

Fig. 3 *Performance of the full-search original algorithm and its 21 transformations with and without an I-cache of 256 words*

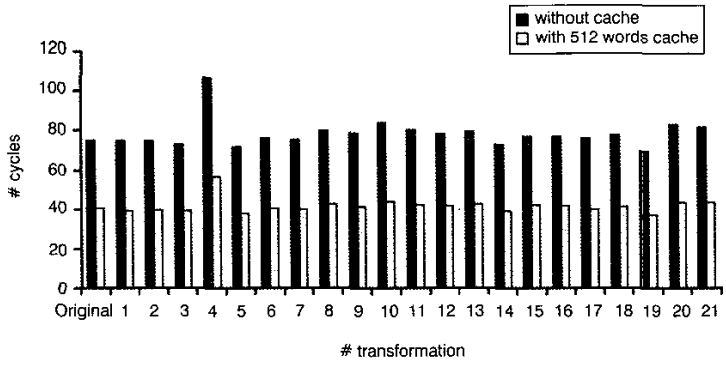

Fig. 4 *Performance of the hierarchical-search original algorithm and its 21 transformations with and without an I-cache of 512 words*

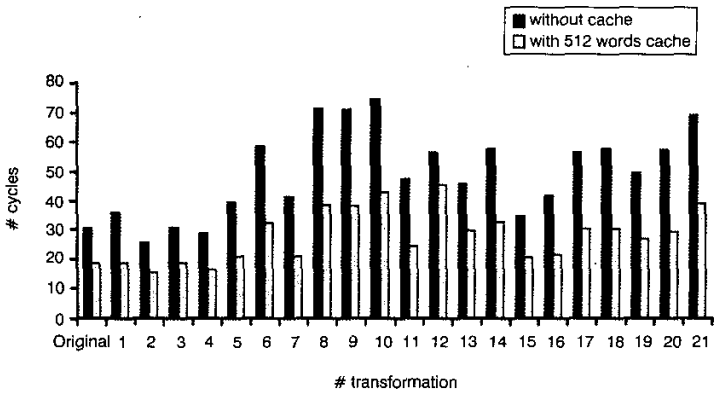

Fig. 5 *Performance of the parallel hierarchical one-dimensional search original algorithm and its 21 transformations with and without an I-cache of 512 words*

algorithmic descriptions and their transformed versions were compiled and simulated with the ARM software development tools v2.1. The clock frequency was set at 33.3 MHz. The data-memory subsystem used was a single read/write-memory port 8 bits wide. The instruction-memory subsystem used was a single read-only-memory port 32 bits wide. To perform design-space exploration, a plethora of measurements was required. Measurements concerning the data memory and performance were acquired using ARMulator combined with the adopted energy models, performed within 30 min per transformed version. For I-mem and I-cache energy consumption, the derivation of the corresponding trace file was a requisite, which was a tedious procedure. Since the trace file normally is huge, in [28] a technique was used to simplify and speed up the measurement procedure (time interval <5 min for each kernel). Having available a trace file, the calculation of miss/hit ratio required some minutes too, using the Dinero tool. All measurements were made using a Pentium III 700 MHz PC with 256 MB.

To determine the optimal cache size for each multimedia kernel, an exhaustive exploration was performed, considering three cache memory sizes: 128, 256 and 512 words. The measurements were taken for various prefetch distances in terms of blocks, with block line $L=2$ words, and for write-back policy. The results are shown in Table 1. More specifically Table 1 shows the average percentage energy increase or decrease due to insertion of an I-cache compared with the architectures without cache. Bold-face numbers denote the maximal power decrease for the selected kernel, which eventually determines the optimal cache size for the kernel considered. The optimal cache sizes are different for each kernel because of the changing code size of the kernel's innermost loop, which contributes significantly to the total power consumption. In other words, the cache size, which corresponds to the code size of the innermost loop, is optimal. This fact implies that there are no capacity misses, which is the case for smaller caches. Further, the larger cache size results in smaller energy savings, despite the fact that the innermost loop code is smaller than the cache size, because there are costly accesses in larger memories.

The experimental results in terms of performance are illustrated in Figs. 3–6. The 21 transformations appearing on these graphs correspond to different data-memory hierarchies, as they have been extracted using the data-memory methodology [7]. The performance figures for the FS, HS, PHODS and the 3SLOG reveal that, by using an I-cache of 256, 512, 512 and 256 words, respectively, an average reduction of 48%, 47%, 43% and 16% in processor cycles can be achieved. The above-optimal I-cache sizes have been chosen after an exhaustive design-space exploration. The fact that, in the 3SLOG, the corresponding reduction is not as high as in the other three algorithms is justified, because transformations 1–3 have a large loop that cannot be effectively mapped onto the I-cache, due to a large number of conflict misses. To alleviate this, one could use special I-cache transformations, or change the degree of associativity.

In Fig. 7*a*, the performance breakdown of the original PHODS algorithm, without a dedicated data memory hier-

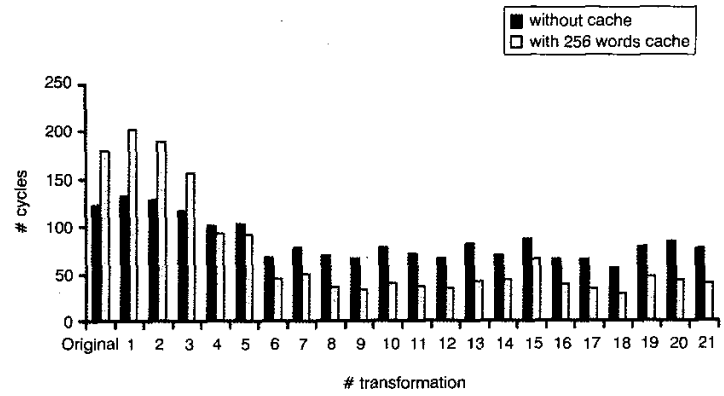

Fig. 6 *Performance of the three-step logarithmic-search original algorithm and its 21 transformations with and without an I-cache of 256 words*

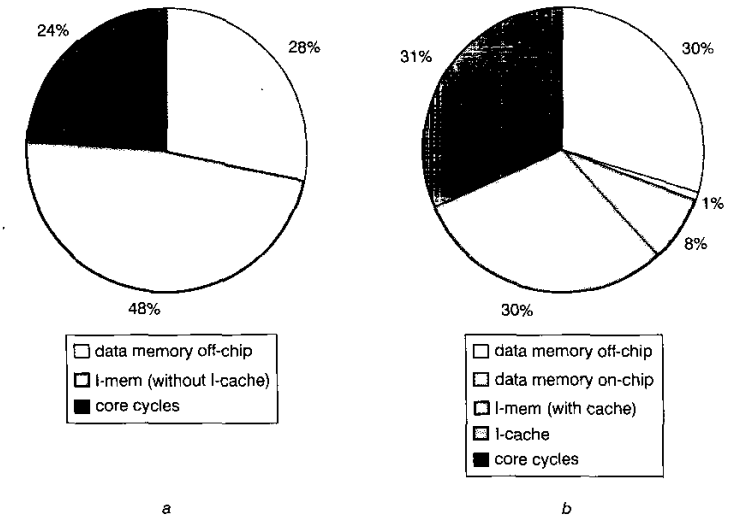

Fig. 7 *Performance breakdown*
a Without the proposed exploration approach being applied
b After the data and instruction-memory exploration approach have been applied

archy and an I-cache, is displayed. The total number of processing cycles required for processing two successive frames is nearly 30×10^6. From this cycle budget, 48% is spent in I-mem accesses, 28% in accesses to background memory and 24% in the actual processing core. It is evident that the I-mem accesses are dominant in the total memory accesses and, consequently, in the total energy consumption. In Fig. 7*b*, the performance breakdown, after application of the proposed methodology in the PHODS algorithm, is illustrated. The total number of processing cycles required for processing two successive frames has been reduced to 15×10^6. Of this cycle budget, 31% corresponds to core cycles, 30% corresponds to I-cache, 30% corresponds to background data memory, only 8% corresponds to I-mem and 1% corresponds to data-memory hierarchy on chip. Also, the energy consumption has been reduced by 22% compared with the original algorithm. Similar results are obtained from the other multimedia kernels.

Detailed experimental results for the optimal cache size are shown in Figs. 8–11 for four multimedia kernels. It is

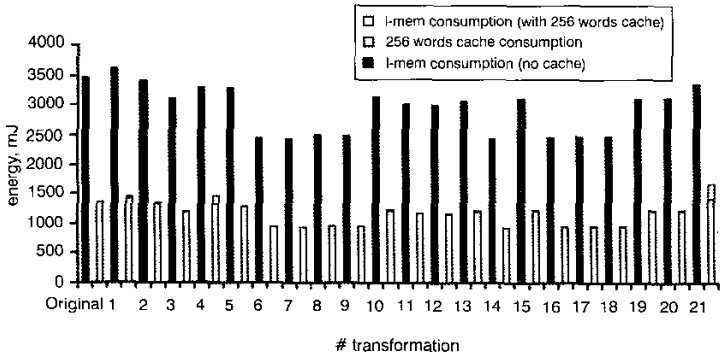

Fig. 8 *Energy consumption of the full-search original algorithm and its 21 transformations with and without an I-cache of 512 words*

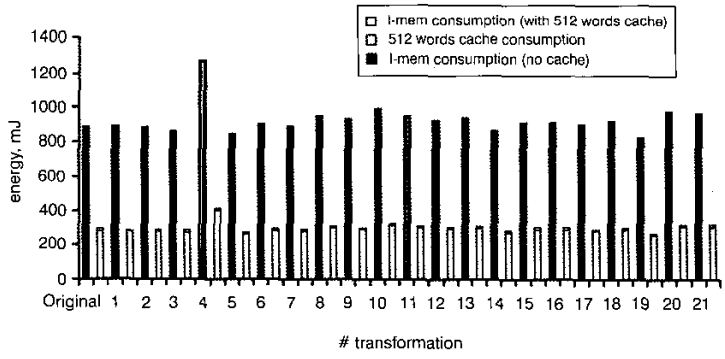

Fig. 9 *Energy consumption of the hierarchical-search original algorithm and its 21 transformations with and without an I-cache of 512 words*

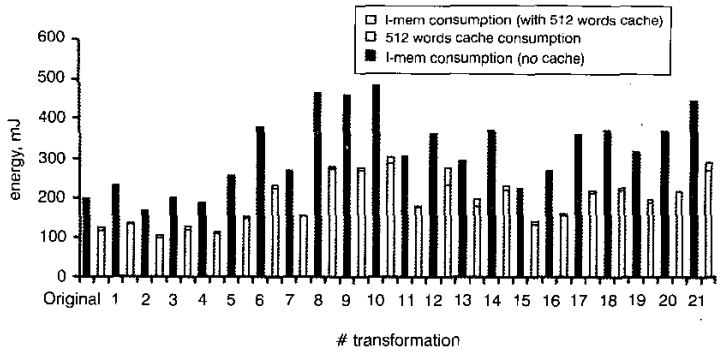

Fig. 10 *Energy consumption of the parallel hierarchical one-dimensional-search original algorithm and its 21 transformations with and without an I-cache of 512 words*

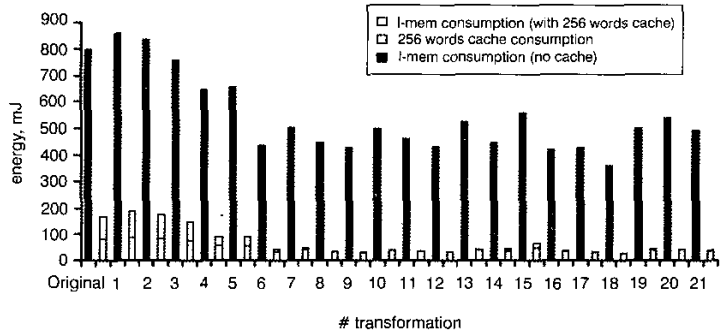

Fig. 11 *Energy consumption of the three-step logarithmic search original algorithm and its 21 transformations with and without an I-cache of 256 words*

concluded that, after the insertion of I-cache, the energy contribution of I-mem is not so important, because the energy consumption in I-cache is over 90% of the total energy consumption of I-mem and I-cache. Using the same optimal cache sizes as in the performance measurements, an average reduction in energy consumption of 59%, 66%, 36% and 83% for all data reuse transformations can be achieved for the FS, HS, PHODS and 3SLOG algorithms, respectively.

Fig. 12 shows a comparison between measurements of the high-level and the technology-dependent model for a sample algorithm (original PHODS). It can be seen that the high-level model follows very closely the architecture-dependent model. The average measured fluctuation between these two models is stable for almost all transformations, and is 34% (Fig. 12). This difference stems from the accuracy level of the adopted energy models. More specifically, the high-level model considers the 'miss' and 'hit' operations as similar processes in terms of energy consumption, and thus uses an average value of the energy consumed for each operations. In contrast, the architecture-dependent model uses the accurate values for 'miss' and 'hit' operations. Comparing (10) and (13), it is concluded that $E_{hit} < E_{miss}$. Consequently, the calculated energy consumption from (8) increases or decreases depending on the measured hit ratio. In Fig. 12, optimised hit ratios were used for the particular application, i.e. they were larger than 0.9. Thus, the calculated values from the high-level model are greater than the corresponding values of the second model. This would not be true, if the hit ratio of

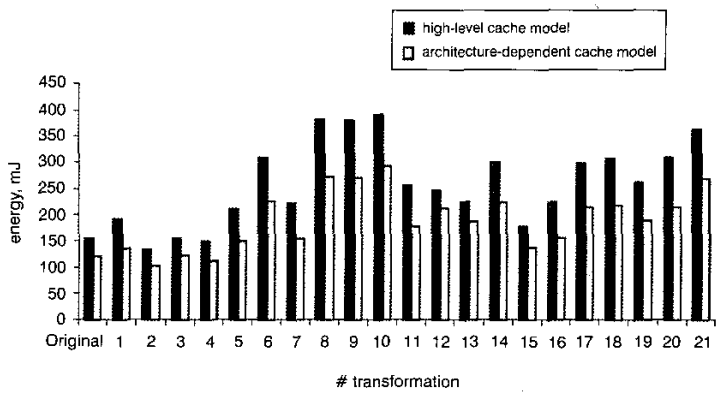

Fig. 12 *Comparison between the high-level architecture-dependent and architecture-independent energy model for the cache*

the 21 different versions of PHODS kernel had not been optimised.

Summarising, it is adequate to carry out measurements using the high-level technology-independent model for fast exploration of the design space, and to use the technology-dependent model for more accurate measurements.

Finally, another conclusion derived from the measurements is that data-reuse transformations, which insert a number of data-memory levels, could give greater savings with the use of an I-cache. This is true because, in order, to exploit the data memory hierarchy, the application code has been rewritten, utilising a number of conditional statements, resulting in a smaller code, in the innermost loop, compared with the original code. The smaller code size of instructions in this loop results in a lower miss rate and, consequently, in lower energy consumption in an I-cache configuration.

4 Conclusions

A framework for designing energy- and performance-efficient embedded systems, executing multimedia kernels, was presented. Application-specific data and instruction-memory hierarchy, as well as embedded programmable processing elements, were considered. Since the instruction-memory power consumption is a significant portion of the total power budget of modern programmable cores, an exploration strategy was developed for optimising first the data memory and secondly the instruction-memory system. An important contribution of this paper is that a fast and efficient design-space exploration for the memory subsystem is feasible, if the designer focuses the I-cache analysis on the most promising candidates of the data-memory part of the technique. Further, the instruction-memory analysis revealed that I-cache has to be evaluated and selected wisely; otherwise a power penalty may occur. Finally, two cache-energy models, a high-level and an architecture-dependent model, were used, to carry out experimental measurements. The experimental results prove that an effective solution in terms of energy, performance or a combination of these two can be detained from the right combination of a processor-core-structure model, high-level algorithmic transformations and a suitable instruction cache.

5 Acknowledgments

This work was partially supported by the project ED 501 PENED '99 funded by G.S.R.T of Greek Ministry of Development. The authors thank Prof. M. Poncino, Dept. Di Automatica é Infromattica, Politechnico di Torino, for his help in the DineroIV Cache simulation application. The authors also thank the reviewers for their suggestions and comments to improve the paper's structure and research results.

6 References

1 PANDA, P., CATTHOOR, F., DUTT, N., DANCKAERT, K., BROCKMEYER, E., KULKARNI, C., VANDECAPPELLE, A., and KJELDSBERG, P.: 'Data and memory optimizations for embedded systems', *ACM Trans. Des. Autom. Electron. Syst.*, 2001, **6**, (2), pp. 142–206

2 CATTHOOR, F., WUYTACK, S., GREEF, E., BALASA, F., NACHTERGAELE, L., and VANDECAPPELLE, A.: 'Custom memory management methodology' (Kluwer Academic Publishers, Boston, 1998)

3 MASSELOS, K., DANCKAERT, K., CATTHOOR, F., GOUTIS, C., and DEMAN, H.: 'A methodology for power efficient partitioning of data-dominated algorithm specifications within performance constraints'. International Workshop on Power and timing modeling, optimization and simulation, PATMOS 1999, Kos Island, Greece, Oct. 1999, pp. 270–272

4 RAMANUJAN, J., HONG, J., KANDEMIR, M., and NARAYAN, A.: 'Reducing memory requirements of nested loops for embedded systems'. Proceedings of 38th Design Automation Conference, Las Vegas, NV, USA, June 2001, pp. 359–364

5 ZERVAS, N., MASSELOS, K., and GOUTIS, C.: 'Code Transformations for embedded multimedia applications: impact on power and performance'. Proceedings of Power driven microarchitecture Workshop, ISCA'98, Barcelona, Spain, June 1998, pp. 70–74

6 ZERVAS, N., MASSELOS, K., and GOUTIS, C.: 'Data-reuse exploration for low-power realization of multimedia applications on embedded cores'. International Workshop on Power and timing modeling, optimization and simulation, PATMOS 1999, Kos Island, Greece, Oct. 1999, pp. 71–80

7 SOUDRIS, D., ZERVAS, N.D., ARGYRIOU, A., DASYGENIS, M., TATAS, K., GOUTIS, C., and THANAILAKIS, A.: 'Data-reuse and parallel embedded architectures for low-power, real-time multimedia applications'. Proceedings of 10th International Workshop, PATMOS 2000, Göttingen, Germany, Sept. 2000, pp. 171–176

8 KIN, J., GUPTA, M., and MANGIONE-SMITH, W.: 'The filter cache: an energy efficient memory structure'. IEEE/ACM International Symposium on Microarchitecture, MICRO'97, Triangle Park, NC, USA, Dec. 1997, pp. 184–193

9 BELLAS, N., HAJJ, I., POLYCHRONOPOULOS, C., and STAMOULIS, G.: 'Architectural and compiler support for energy reduction in the memory hierarchy of high performance microprocessors'. International symposium on Low power electronics & design, ISLPED'98, Monterey, CA, USA, Aug. 1998, pp. 70–75

10 LIVERIS, N., ZERVAS, N., SOUDRIS, D., and GOUTIS, C.: 'A code transformation-based methodology for improving I-cache performance of DSP applications'. Design Automation and Test in Europe, DATE 2002, Paris, France, March 2002, pp. 977–980

11 PARAMESWARAN, S., and HENKEL, J.: 'I-CoPES: fast instruction code placement for embedded systems to improve performance and energy efficiency'. IEEE/ACM International Conference on Computer aided design, ICCAD 2001, San Jose, CA, USA, Nov. 2001, pp. 635–641

12 JAYAPALA, M., CATTHOOR, F., and LAUWEREINS, R.: 'Low energy clustered instruction fetch and split loop cache architecture for long instruction word processors'. Workshop on Compilers and operating systems for low power, COLP'01, Barcelona, Spain, Sept. 2001, pp. 141–148

13 BUDAGAVI, M., WEBB, J., ZHOU, M., LIANG, J., and TALLURI, R.: 'MPEG-4 video and image coding on digital signal processors', *J. VLSI Signal Process.*, 1999, **23**, pp. 51–66

14 KULKARNI, C., MOOLENAAR, D., and NACHTERGAELE, L.: 'System-level energy-delay exploration for multimedia applications on embedded cores with hardware cache', *J. VLSI Signal Process.*, 1999, **22**, pp. 45–57

15 SHIUE, W., and CHAKRABARTI, C.: 'Memory design and exploration for low power, embedded systems'. IEEE Workshop on Signal processing systems: design and implementation, SiPS 99, Taipei, Taiwan, ROC, Oct. 1999, pp. 281–290

16 BENINI, L., MACII, A., MACII, E., and PONCINO, M.: 'Increasing energy efficiency of embedded systems by application-specific memory hierarchy generation', *IEEE Des. Test Comput.*, 2000, **17**, (2), pp. 74–85

17 DEREK, C., JAIN, P., RUDOLPH, L., and DEVADAS, S.: 'Application-specific memory management for embedded systems using software controlled caches'. Design automation Conference, Los Angeles, CA, USA, June 2000, pp. 416–419

18 RAMANUJAM, J., HONG, J., KANDEMIR, M., and NARAYAN, A.: 'Reducing memory requirements of nested loops for embedded systems'. 38th Design automation Conference, Las Vegas, NV, USA, June 2001, pp. 359–364

19 Advanced Risc Machines Ltd Dec. 1994, ARM7 data sheet

20 KUNG, S.Y.: 'VLSI array processors' (Prentice Hall, Eaglewood Cliffs, NJ, 1988)

21 ARGYRIOU, A., DASYGENIS, M., TATAS, T., ZERVAS, N., and SOUDRIS, D.: 'Data-reuse exploration of multimedia applications on programmable processor cores'. IEEE International Conference on Design of circuits and integrated systems, DCIS 2000, Montpellier, France, Nov. 2000, pp. 651–656

22 CATTHOOR, F., JANSSEN, M., NACHRERGAELE, L., and DE MAN, H.: 'System-level data-flow transformations exploration and power-area trade-offs demonstrated on video codes', *J. VLSI Signal Process.*, 1998, **18**, (1), pp. 39–50

23 WUYTACK, S., DIGUET, J., CATTHOOR, F., and DE MAN, H.: 'Formalized methodology for data reuse: exploration for low-power hierarchical memory mappings', *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, 1998, **6**, (4), pp. 529–537

24 LANDMAN, P.: 'Low power architectural design methodologies: August 1994, PhD, U.C. Berkeley

25 Edler, J., Hill, M.D.: 'A cache simulator for memory reference traces', http://www.neci.nk.nec.com/homepages/edler/d4

26 SU, C., and DESPAIN, A.: 'Cache design trade-offs for power and performance optimization: a case study'. International Symposium on Low power electronics and design, 1995, pp. 63–68

27 BHASKARAN, V., and KONSTANTINIDES, K.: 'Image and video compression standards' (Kluwer Academic Publishers, 1998)

28 DASYGENIS, M., KROUPIS, N., ARGYRIOU, A., TATAS, K., SOUDRIS, D., THANAILAKIS, A., and ZERVAS, N.: 'A memory management approach for efficient implementation of multimedia kernels on programmable architectures'. IEEE Computer Society Workshop on VLSI, WVLSI 2001, Orlando, FL, USA, April 2001, pp. 171–176