DATA AND INSTRUCTION MEMORY EXPLORATION OF EMBEDDED SYSTEMS FOR MULTIMEDIA APPLICATIONS

M. Dasigenis, N. Kroupis, A. Argyriou, K. Tatas, D. Soudris

VLSI Design and Testing Center,
ECE Dept., Democritus University of Thrace,
67 100 Xanthi, Greece

N. Zervas

VLSI Design Lab
ECE Dept., University of Patras,
26500, Patras, Greece

ABSTRACT

A methodology for power optimization of the data memory hierarchy and instruction memory, is introduced. The effect of the methodology on a set of widely used multimedia application kernels, namely Full Search (FS), Hierarchical Search (HS), and Parellel Hierarchical One Dimension Search (PHODS), is demonstrated . Three different target architecture models are used. The issues of the data memory power reduction and instruction memory are tackled separately. We find the power optimal data memory hierarchy applying the appropriate data-use transformation, while the instruction power optimization is done using suitable cache memory. Using data-reuse transformations, performance optimizations techniques, and instruction-level transformations, we perform exhaustive exploration of all the possible alternatives to reach power efficient solutions. The experimental results prove the efficiency of the methodology in terms of power for all the multimedia kernels.

1. INTRODUCTION

In multimedia and other applications that make use of large multidimensional array type data structures, a very large amount of memory is required. In the past, the major concerns of the VLSI engineers were designing efficient circuits in terms of area and performance; power considerations were rarely dealt with. In recent years however, designers have begun to study equal the subject of power consumption along with these traditional factors, and have tried to find heuristic approaches for power and area efficient designs. Several factors have contributed to this attitude change. The most important factor was the increasing need for wireless systems or portable multimedia applications. The portability of a device is heavily bound with the power consumption of it, since power consumption affects the battery service life and weight, packaging costs as well as the circuit reliability. For this reasons, power consumption has emerged as a very significant design constraint, that has to be tackled, especially in the high levels of design, where the most significant savings can be achieved[1] .

Generally speaking, two possible implementations exist in order to meet the processing constraint, that is the use of a dedicated hardware architecture or a number of embedded programmable cores. In particular custom hardware designs are area and power efficient but they lack of the flexibility, since it is possible to execute only one algorithm every time. On the other hand, the programmable embedded cores are less efficient in terms of power consumption and chip area, but they are more versatile since it allow us to execute multiple algorithms in the same target architecture. Thus, a broader class of applications can be implemented, reducing the design cycle.

Multimedia applications, are data-dominated applications that need a lot of memory to store the data being processed. The data transfers between the memories and the data paths, have a huge impact on power consumption. As it was demonstrated in recent studies [2], the memory system is the main power consuming unit in multimedia systems. Two are the major reasons for this behavior: (i) the data dominated nature of multimedia applications, as it was stated before, and (ii) the power consumed in accessing off-chip memories, which is significant more than normal arithmetic or logical operations. Thus, it is necessary to perform hardware and software power optimization techniques, in order to design a power efficient system.

The problem of designing power and area efficient embedded systems, is rather new and thus, the bibliography is relatively small [2], [3], [4]. Specifically, Catthour et. al. [2] proposed a systematic methodology for the reduction of memory power consumption in custom architectures. Zervas et. al [3] presented another research work, which target single programmable processor-based systems. All of these approaches do not tackle with the problem of partitioning and thus, they cannot apply on multiprocessor environments. Partitioning approaches were presented in [5], [4], improving the memory utilization. However, these approaches are limited by the two-level memory hierarchy, while the class of algorithms expressed in Weak Single Assignment Code form [6]. Recently, some novel partitioning techniques are presented in [7], [8], stating for the first time the importance of the instruction memory power consumption on embedded systems.

Except from the impact of the data memory, embedded systems are characterized by one additional critical component that has a significant part in the power budget, which is the instruction memory of the programmable memory that stores the algorithm to be executed. From our experiments, the task of designing power and area efficient systems, comprises of two distinct problems: (i) the data memory optimization using small on-chip memories, and (ii) the instruction memory optimization using appopriate caches.

In this paper we perform an exhaustive exploration of data-reuse transformations, performance optimizations and power transformations, in terms of area, power and performance for mul-

This work was supported by the project PENED '99 funded by G.S.R.T of Greek Ministry of Development.

timedia applications executed on embedded cores. After we have find an optimal data memory hierarchy, we perform instruction power optimization by using a suitable cache memory. Experimental results, illustrate the efficiency of our proposed methodology.

2. TARGET ARCHITECTURE AND DEMONSTRATION APPLICATIONS

Our target architecture model is illustrated in Fig. 1, and consists of: (i) multiple processing cores, N, each of which has its individual on-chip instruction memory, (ii) instruction cache memory, and (iii) data memory hierarchy. This architecture is an extension of a previous architecture [7], but is extended to include not only the instruction memory but also it's corresponding cache. The used target architecture models are the Distributed Memory Architecture (DMA), Shared Memory Architecture (SMA) and Share-Distributed Memory Architecture (SDMA)[7], [6]. For experimental purposes, we consider four different algorithms, with two processor cores ($N = 2$), and without any restriction about data memory hierarchy levels.

As demonstration applications we select three well-known Motion Estimation (ME) kernels, used in a great number of video processing applications, which are the FS, the HS, and the PHODS. Our experiments were carried out using the luminance components of QCIF frame (144x176) format. Reference window was selected to include 15x15 candidate blocks, while blocks of 16x16 pixels were considered. All these algorithms calculate the motion vectors of two images, but they differ in the granularity, the precision and the complexity. Specifically, FS is the most computational expensive but guarantees finding the optimal motion vectors, HS is a fast ME scheme that use a combination of search strategies that use both fewer search locations and fewer pixels in computing the motion vectors, while PHODS belongs to the class of very fast algorithms that reduce motion-estimation complexity by reducing the number of search locations that are used in determining the motion vectors.

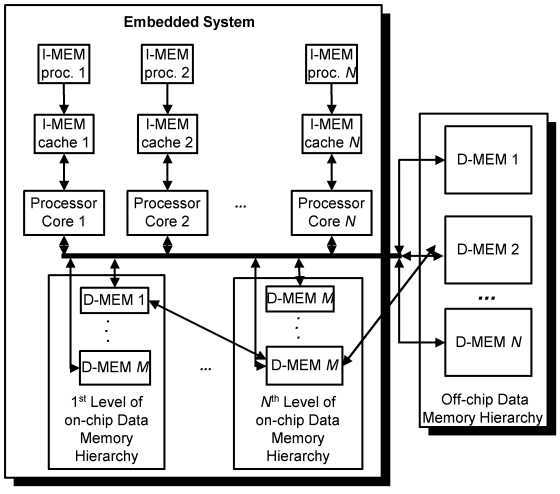

Fig. 1. Target Architecture Model

3. PROPOSED METHODOLOGY

The structure of the proposed methodology is shown in Fig. 2. We aim at the determination of the optimal data memory hierarchy for reducing power due to a number of off-chips transfers, and the optimal cache memory for reducing instruction memory power consumption. Actually, there exist two distinct paths regarding the power consumption estimation, the corresponding data memory hierarchy (D-Memory) and the power estimation of the cache and instruction memory (I-Memory). The first path studied in detailed in [7]. [8]. Here, we will emphasis on the second path. The motivation behind this decision was the results obtained from previous work, that revealed the dominant role of the I-Memory power in the total memory power budget. Of course, this is valid when programmable processor cores are assumed.

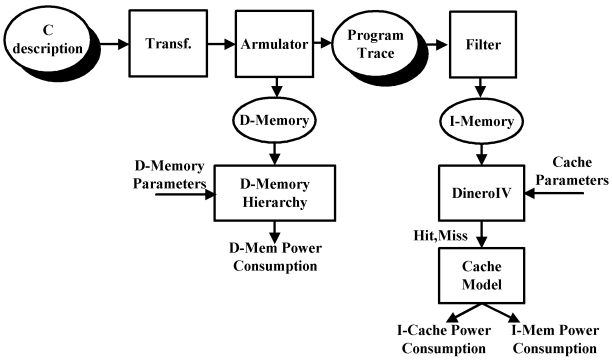

Fig. 2. The proposed Methodology: Estimation of D-Mem, I-Mem, I-Cache power consumption

3.1. Data Memory Methodology

The first step of our data memory methodology consists of three types of high level transformations; namely data-reuse, performance and instruction level transformations. The second step is to map the transformed algorithm to the physical memories.

Employing data reuse transformations [2], we determine the certain data sets, which are heavily re-used in a short period of time. The re-used data can be stored in smaller on-chip memories, which require less power per access. In this way, redundant accesses from large off-chip memories are transferred on chip, reducing power consumption related to data transfers. Of course, data reuse exploration has to decide which data sets are appropriate to be placed in separate memory. Otherwise, we will need a lot of different memories for each data set resulting into a significant area penalty. Here, we applied 21 data-reuse transformations [7] to all target architecture models for the three ME kernels.

Another type of transformations applied was the performance optimizations, like common sub-expression elimination. Of course this kind of transformation has an impact in the instruction power budget. The tradeoff in this case was between the increase in the instructions due to the extra assignments in one hand, and the decrease in the instructions due to sub-expression elimination on the other hand. Sub-expressions are useful to eliminate when they have to be executed in a great number of loops. When the number of loops is small the overhead produced by the assignment retracts the benefits of the elimination.

The third type of transformations are the instruction level transformations, which are processor dependent. Indeed, a program written in high level language, eg. C, can be re-written substituting power hungry instructions with power efficient ones. For example we have found that the multiply operation in the ARM processor could be substituted with summation operations.

The final step is the mapping process. For all the data-memory architectures models a shared background (probably off-chip) memory module is assumed. Thus, in all cases special care must be taken during the scheduling of accesses to this memory, to avoid violating data-dependencies and to keep the number of memory ports as small as possible in order to keep the power per access cost as small as possible. With DMA, a separate data-memory hierarchy exists for each processor. In this way all memories modules of the memory hierarchy are single ported, but also area overhead is possible in cases of large amount of common data to be processed by the N processors. The second data-memory architecture-model (i.e. SMA) implies a common hierarchy of memory levels for the N processors. Since, in the data-dominated programmable parallel processing domain, it is very difficult and very performance inefficient to sequentially schedule all memory accesses, we assume that the number of ports for each memory block equals the maximum number of parallel accesses to it. Finally, SDMA is a combination of the above two models, where the common data to the N processors are placed in a shared memory hierarchy, while a separate data memory hierarchy also exist for the lowest levels of the hierarchy.

3.2. Instruction Cache Methodology

In this point of the methodology, we have reached to some (in our case 21) transformed versions of our algorithm. These transformed algorithms imply the use of a specific data-memory hierarchy. Having this in mind we make measurements in order to evaluate the performance of the algorithms. In this way we create a pool of possible solutions for further study. From this pool of candidate algorithms we choose those that have relatively low execution time and they also have reduced data-memory power. This is a way to reduce the huge search space and help us reach a near-optimal solution more quickly. After having selected some of the candidate algorithms we then run a simulation again to attain their exact instruction trace. This trace is then fed to the DineroIV cache simulator [10] for various cache parameters. At the end, we select the algorithm with a big hit ratio and as small memory size as possible. After this last step we have selected a very good instruction cache that reduces the power produced by the fetching of the instructions from the instruction memory.

The combination of the data and instruction memory exploration/optimization provide a complete approach to the problem of excessive power consumption in an embedded system's memory.

4. EXPERIMENTAL RESULTS

Comparison among the three target architectures DMA, SMA and SDMA in terms of power consumption or three well-known ME algorithms, are shown in Fig. 3-8. Emphasis of this contribution is on the reduction of the instruction memory consumption the use of appropriate cache memory. The issue of data memory, perfomance, and area has been studied detailed manner in [7],[8].

Using the proposed methodology, in the first phase we perform exhaustive exploration applying 21 data re-use transformations to estimate the instruction power without the use of cache and the optimized data memory power consumption. Then, we perform instruction power optimization for the data re-use transformations. Here we provide the results corresponding to the minimum and maximum instruction power consumption. The derived results for all ME kernels and target architectures are shown in Fig. 3 - 5 For each target architecture we perform three pairs of measurements with and without cache memory, that is, (i) original kernel, (ii) transformed kernel (using appropriate data-use transformation) that corresponds to MIN instruction power consumption, and (iii) transformed kernel (using appropriate data-use transformation) that corresponds to MAX instruction power consumption.

From Fig. 6 - 8, we inferred that the existence of cache memory affect significantly the instruction memory power consumption. More specifically, FS exhibits the most computational complexity is dominated by instruction power (Fig. 6). The corresponding cache analysis proves that the power savings almost for all target architecture models varries from 10% to 90%. The remaining two ME kernels have similar instruction power savings. However, the existence of cache memory has smaller impact on the SDMA model than the SMA and DMA model.

The larger is the computational complexity of an algorithm the larger the instruction power savings, for instance FS kernel. On the other hand, HS and PHODS kernels have similar complexity and their corresponding power analysis shows similar power consumption and eventually, similar cache optimization results.

For our experiments we use cache memory size 128 bytes, with block line L=2 bytes, and associativity a=1. These optimal values came after power exploration of cache memory.

The final conclusions after the exhaustive exploration both in data and instruction memory are: (i) for FS kernel the most power efficient model (with cache) is the DMA, (ii) for HS kernel the most power efficient is the SMA, and (iii) PHODS kernel the most power efficient is the SDMA model. Consequently, the power optimized solutions depended on the chosen application and the assumed target architecture model.

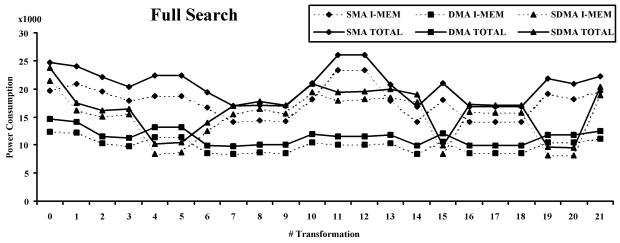

Fig. 3. Total and Instruction power consumption in FS Algorithm

5. CONCLUSIONS-FUTURE WORK

A complete methodology for designing power-efficient embedded systems for ME kernels, was presented. Application specific, data-memory hierarchy and instruction memory, as well as embedded programmable processing elements, were assumed. The proposed methodology had two goals: First, to design of an efficient data memory hierarchy, and second to use of a suitable instruction cache to reduce the heavy impact of the programmable instuction memory on total power consumption. The experimental results

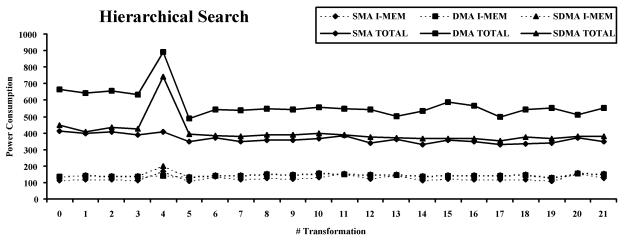

Fig. 4. Total and Instruction power consumption in HS Algorithm

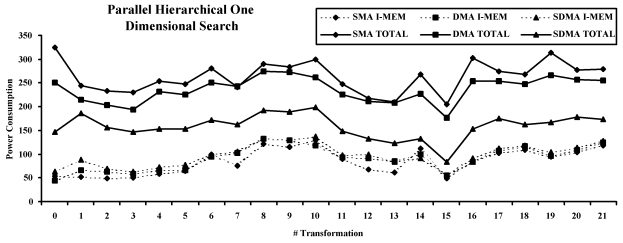

Fig. 5. Total and Instruction power consumption in PHODS Algorithm

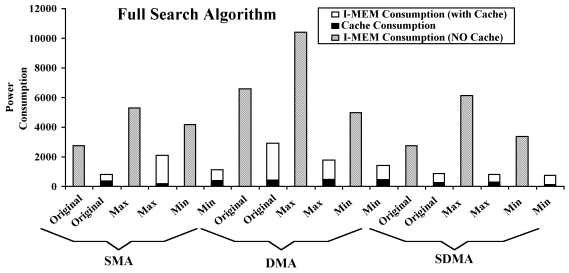

Fig. 6. Cache Analysis for the Full Search Algorithm

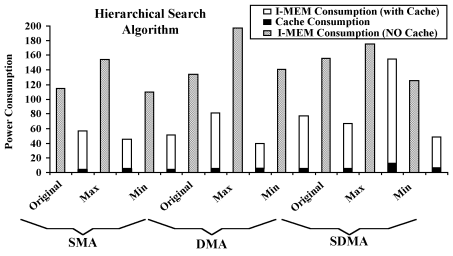

Fig. 7. Cache Analysis for the Hierarchical Search Algorithm

prove that an effective solution either in terms of power, can be acquired from the right combination of processor core structure model, data-reuse transformation and suitable instruction cache.

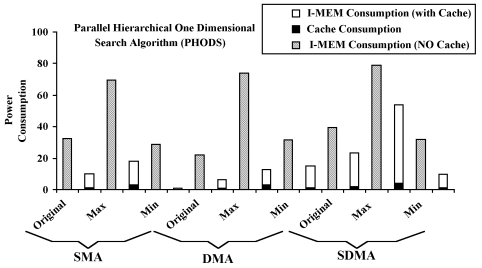

Fig. 8. Cache Analysis for the PHODS Algorithm

Aknowledgments

The authors would like to thank Professors E. Macii and M. Poncino from the "Politechnico di Torino", for their help in DineroIV cache simulation application.

6. REFERENCES

[1] A. P. Chandrakasan, R. W. Brodersen, *"Low Power Digital CMOS Design"*, Kluwer Academic Publishers 1998.

[2] F. Catthoor, et al., *"Custom Memory Management Methodology,"* Kluwer Academic Publishers, Boston, 1998.

[3] N. D. Zervas, K. Masselos, C.E. Goutis, *"Data-reuse exploration for low-power realization of multimedia applications on embedded cores"*, Proc. of PATMOS'99, October 1999, pp. 71-80.

[4] S. Wuytack, J. P. Diguet, F. Catthoor, D. Moolenaar and H. De Man *"Formalized Methodology for Data Reuse Exploration for Low-Power Hierarchical Memory Mappings"*, IEEE Trans. on VLSI Systems, Dec. 1998, pp. 529-537.

[5] U. Eckhardt, R. Merker, *"Hierarchical Algorithm Partitioning at System Level for an Improved Utilization of Memory Structures"*, IEEE Trans. on CAD, pp. 14-23, Jan. 1999.

[6] L. Nachtergaele, B. Vanhoof, F. Catthoor, D. Moolenaar, and H De Man, *"System-level power optimizations of video codecs on embedded cores: a systematic approach,"* Journal of VLSI Signal Processing Systems, Kluwer Academic Publishers, Boston, 1998.

[7] D. Soudris, et. al., *"Data-Reuse and Parallel Embedded Architectures for Low-Power, Real-Time Multimedia Applications"*, Proc. of 10th Int. Workshop PATMOS 2000, Sep. 2000, pp. 243-254.

[8] K.Tatas, et. al., *"Memory Hierarchy Optimization of Multimedia Applications on Programmable Embedded Cores"*, accepted in Int. Symp. on Quality of Electronic Design (ISQED) March 26-28, 2001, San Jose, USA.

[9] V. Bhaskaran and K. Kostantinides, *"Image and Video Compression Standards"*, Kluwer Academic Publishers, 1998.

[10] Jan Edler and Mark D. Hill, *"A cache simulator for memory reference traces"*, http://www.neci.nj.nec.com/homepages/edler/d4