CODE SIZE EFFECTS OF POWER OPTIMIZING CODE TRANSFORMATIONS FOR EMBEDDED MULTIMEDIA APPLICATIONS

K. Masselos[1,2], F. Catthoor[2], C. E. Goutis[1], H. DeMan[2]

[1] VLSI Design Laboratory, Department of Electrical and Computer Engineering, University of Patras, Rio 26500, Greece.
[2] IMEC, Kapeldreef 75, B 3001 Leuven, Belgium.

ABSTRACT

A systematic methodology has been developed for the reduction of the data transfers and storage related power consumption in realizations of multimedia applications on programmable multimedia processors. The methodology is based on the application of code transformations that move the main part of the memory accesses from the large (off-chip) memories to smaller ones (on-chip). Performance (in number of cycles) which is the overriding constraint for multimedia applications is improved in most cases by the application of the power optimizing code transformations as well. The main focus of this paper is on the effect of these power optimizing code transformations on the code size, an important design parameter that implicitly affects both the total system power consumption and the performance. Experimental results from real-life applications demonstrate that contrary to earlier conjectures, if applied well our methodology introduces only small code size penalties that do not offset the significant gains in both performance and power consumption achieved by its application.

1. INTRODUCTION

Rapid advances in the area of programmable (embedded) processors made them an attractive solution for the realization of real time multimedia applications, since they allow implementation of several different applications on the same hardware but also for time-to-market reasons. Power consumption became an important design consideration due to portability as well as packaging and cooling issues [1]. Realizations of data dominated signal processing applications

such as multi-media require a large amount of memory for the storage of the large multi-dimensional array-type data structures that are present in such applications. Advances in memory technology keep decreasing the memory power cost but the ever-increasing storage requirements of data dominated applications offset these gains and retain memory related power consumption as the dominant contribution to the total (board-level) system power cost.

In the area of system level memory management for power and area, the ATOMIUM [2] and the PHIDEO [3] methodologies have been proposed. Both these methodologies mainly target uniprocessor (single thread of control) custom hardware architectures and cannot be directly applied to a programmable processor context in a straightforward manner. This is because extra issues like performance (in terms of number of cycles) which is the overriding constraint in multimedia applications and code size must be taken into account in such systems. Furthermore in many embedded systems the architecture of the memory hierarchy is fixed and includes partly hardware controlled caches offering constraints and different freedom for optimization. No full methodologies exist currently for systematic power-performance exploration in systems with a fixed cache-based general-purpose memory hierarchy. Only some initial results have been recently obtained by us [4].

The overall aim of our research, part of which is described in this paper, is the development of a global methodology for the application of data storage and transfer optimizing code transformations that minimize power consumption and bus load while still meeting all performance constraints. The target domain consists of data-dominated multimedia applications realized on programmable processors. This methodology will be included in the high-level transformation stage of the ACROPOLIS source-to-source multimedia pre-compiler at IMEC [5]. In this paper the focus is on the effect of the power optimizing code transformations on the code size of the target application which affects implicitly both the total system power and the performance.

The rest of the paper is organized as follows: In section 2 the target architecture is descibed while the power model used is briefly discussed in section 3. In section 4 a review of the proposed methodology for the reduction of the data storage and transfers related power consumption is presented. In section 5 the way in which the code size affects the total system power and the performance is

explored. Experimental results from real-life applications are presented in section 6. Finally conclusions are offered in section 7.

2. TARGET ARCHITECTURE

The target architecture for our work is based on (parallel) programmable video or multimedia processors (Philips TriMedia, TMS320C6x but also PENTIUM MMX) as this is the current state-of-the-art in real time multi-dimensional signal processing applications. The target architecture is described in figure 1.

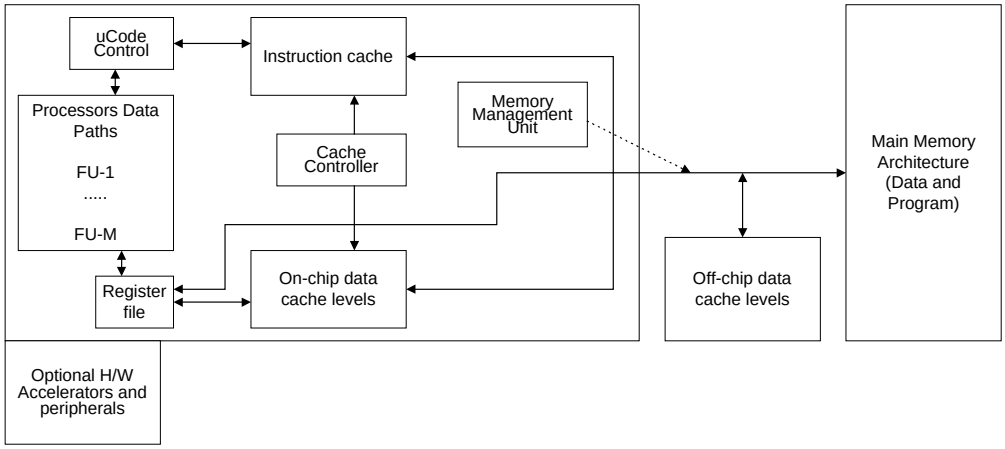

fig.1: General view of the target architecture model

3. POWER MODEL

The power consumption due to accesses to the background memories (caches, main off-chip memory) and due to transfers through on-chip and off-chip busses is only taken into consideration in the proposed context since the power consumption in the registers and in the data paths is much smaller [1]. More details on the used power model can be found in [2].

4. BRIEF REVIEW OF OUR METHODOLOGY

The proposed methodology is based on a number of code transformations that have already been proposed for power optimization in a custom hardware context in ATOMIUM [2]. These transformations are summarized below:

1. **Data flow transformations:** Remove dataflow bottlenecks and reduce the number of accesses to background memories.

2. **Loop/control flow transformations:** Increase the locality and regularity of the accesses enabling the reduction of the number of accesses to the larger background memories in the memory hierarchy.
3. **Data reuse transformations [2]**: Introduce array signals where copies from larger signals that exhibit data reuse are stored.
4. **In-place mapping [2]**: Consists of two steps: a) Intra signal in-place. This step optimizes the storage order inside a single signal. b) Inter signal in-place. It optimizes the storage order between different signals.

The transformations can be regrouped in three categories in our extended script for programmable multi-media processors, depending on the way they operate on the array signals of the initial system specification (see [4] for details).

a) **Access removing transformations:** Reduce the number of accesses to the different array signals present in the algorithm's description.
b) **Size reduction transformations:** Reduce the size of array signals that are present in the initial specification, allowing storage in lower and smaller levels of the memory hierarchy that lie closer to the processing units**.**
c) **Access moving transformations:** Introduce extra array signals where accesses from larger array signals existing in the initial specification are moved to. The new signals are smaller than the existing signals and thus they can be stored in lower levels of the memory hierarchy (even the register file).

In the proposed methodology the code transformations are applied in the order described above to create space in the cache and achieve a better cache performance. In this paper the focus lies on analyzing how these code transformations affect the code size.

5. RELATION OF CODE SIZE TO THE TOTAL SYSTEM POWER AND PERFORMANCE

The code size of the target application realized on a specific processor is related to the number of the instruction cache misses. This is better illustrated by the results presented in figure 2. Specifically the relative changes in the code size and in the instruction cache misses for different versions of several realistic applications are presented. The applications include four motion estimation kernels namely the full search (FS), the logarithmic (Log), the parallel hierarchical one dimensional (Phods) and the three level hierarchical (Hier), the

QSDPCM video compression algorithm, a voice coding algorithm and a cavity (edge) detection algorithm. For each application, the first version corresponds to the original version of the application while the others have been produced after the application of the power optimizing transformations described in section 4. The results are relative for each application on a stand-alone basis and they correspond to mappings on a TriMedia TM1 processor.

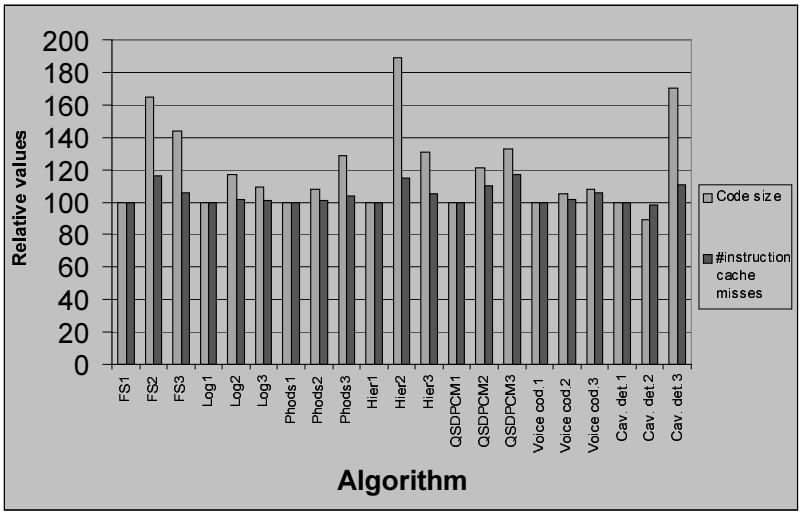

fig. 2: Relation between code size changes and changes in the number of instructrion cache misses

In all cases, changes (increase/decrease) in code size lead to similar but smaller changes (1.5 to 9 times) in the number of instruction cache misses. Instruction cache misses and thus code size are closely related to performance (in terms of number of cycles) and to the total power consumption (since they lead to off-chip transfers which are power expensive) in the target architecture. Thus in the context of our research the application of the power optimizing code transformations should not introduce code size penalties that may (through increase of the instruction cache misses) offset the savings in both power and performance achieved by their application.

6. EXPERIMENTAL RESULTS

In this section the effect of the systematic application of power optimizing code transformations on the code size of the real-life demonstrators on different target processors is explored.

For the case of the QSDPCM application the results for the TriMedia and Pentium MMX processors are shown in figure 3. Transformations 1-4 are size

reduction transformations and transformations 5-9 are access moving transformations. Performance oriented transformations have been applied for the reduction of the complexity mainly of the address and control expressions [6] both before and after the application of the power oriented transformations. The application of the power oriented transformations leads to a reasonable increase of the code size (31% for TriMedia - 30% for Pentium MMX). This code size increase is relatively small taking into consideration that the applied power oriented transformations cover the whole application code. Moreover, the transformations that introduce the larger code size penalties (trafos 1, 8, 9) are those that apply relatively extensive loop unrolling (in order to achieve a smaller address overhead and simpler register storage). Thus it is the unavoidable unrolling step which introduces the major code size penalties and not the high-level power optimizing transformations themselves. Note though that also the traditional instruction-level compiler stage would anyhow apply this unrolling, but then mainly for arithmetic efficiency, with a similar penalty. The actual power oriented transformations either slightly increase the code size or decrease it (trafos 3, 7: 2-3% small decrease for TriMedia - trafo 3: 50% large decrease for Pentium MMX). The size reduction code transformations applied are mainly loop merging transformations. In principle, the latter does not introduce extra code. It must be noted that also transformation 1 which is a global transformation (it covers the whole application code) is a loop merging. Access moving transformations and especially data re-use transformations introduce a limited amount of new code. The applied access moving transformations are mainly data re-use transformations which are local and affect only small parts of the application code. The pure performance oriented transformations (last steps) increase the code size significantly since they affect the whole application code. The effect of the combined application of power and performance oriented transformations on code size is an increase by 42% for the case of TriMedia and by 45% for the case of Pentium MMX. Taking into consideration the results of the previous section, it is expected that this increase of the code size will only lead to a minor increase of the instruction cache misses (and of the related power and performance penalties), namely less than 5%. This is largely compensated by the increased performance due to DTSE transformations, which means that

the processor is active for a smaller amount of cycles to execute a given application.

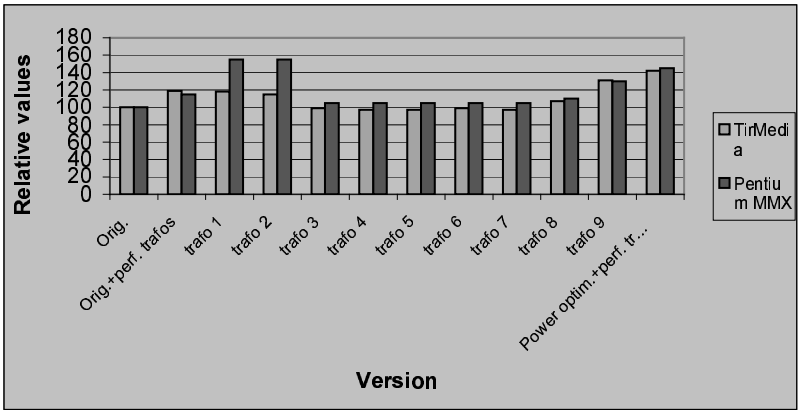

fig. 3: Effect of power optimizing code transformations on code size for the QSDPCM

The results for the voice coder application are shown in figure 4. The effect of the power-oriented transformations on the code size is even positive mainly due to the efficient application of register storage (without using loop unrolling). The code size is reduced after the application of the size reduction step in this case for both processors. This is possible because the size reduction code transformations which have been applied are in-place and loop merging transformations that do not introduce new code and even simplify the original code. The applied access moving transformations are data re-use transformations. Their effect in code size in this case is small since their application is very limited and local. Furthermore the signals introduced by the re-use transformations are scalars and thus the introduced code is small (in comparison to the cases where the introduced signals are arrays). It must be noted that for the voice coder application, power optimizing transformations do not cover the whole application code. Actually, the transformations affect a smaller percentage of the application code in comparison to the other two applications. For the case of the TriMedia, the code size is decreased by 6% while the decrease is even larger for Pentium MMX (14 %).

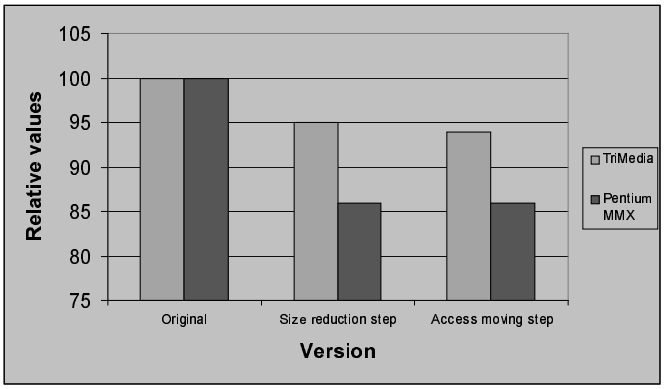

fig. 4: Effect of power optimizing code transformations on code size for the voice coder

Finally the results for a cavity detection application are presented in figure 5. Transformations 1-5 are power optimizing transformations while transformation 6 is a performance oriented transformation. In this case, the code not related to our data transfer oriented approach is quite small, because it is only the algorithm kernels that have been included. So it can be expected that the increase of the code size after the application of the code transformations is larger in comparison to the previous applications for both processors. Significant code size penalty is introduced after the application of the data re-use transformation (trafo. 4). For the case of the TriMedia the code size is increased by 102% while the penalty for the case of Pentium MMX is 64%. Some power optimizing code transformations reduce the code size (trafo 1, trafo 3) while in this case the performance oriented transformations (trafo 6) do not significantly increase the code size (they even slightly decrease it for the case of TriMedia).

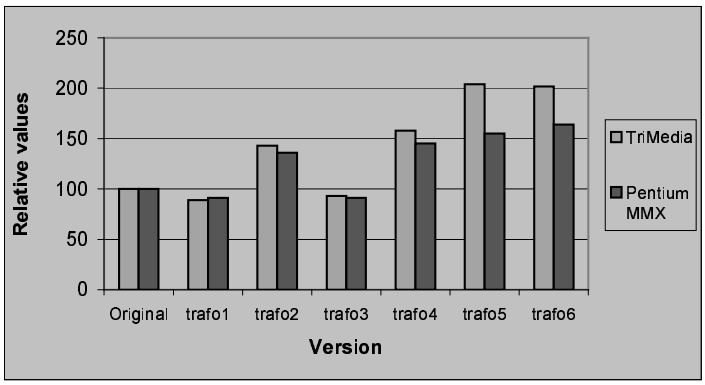

fig. 5: Effect of power optimizing code transformations on code size for the cavity detection

The effect of the application of the power optimizing transformations on the data storage and transfer related power, on instruction power and on the system performance is presented for the case of the TriMedia in figure 6. Data and instruction related power has been estimated using the model presented in section 3 and the output of the TriMedia simulator.

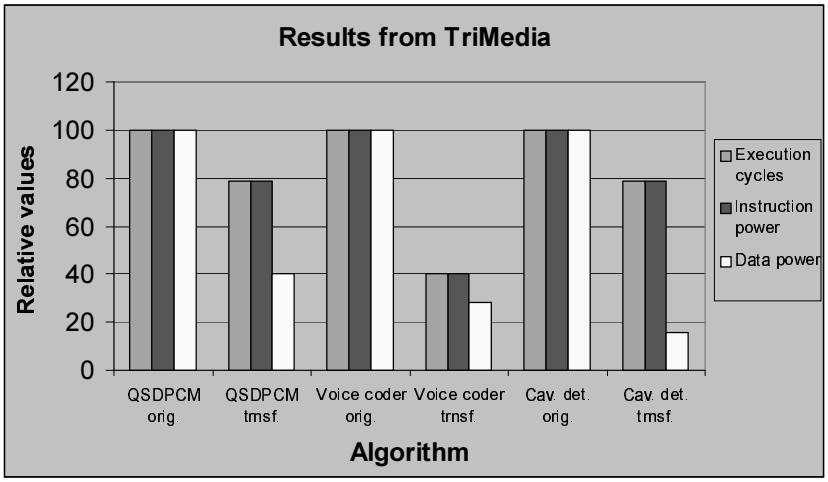

fig. 6: Overall effect of power optimizing transformations on data storage and transfer power, instruction power and on performance for TriMedia

From the results of figure 6 it is clear that the application of the power optimizing code transformations (plus the performance oriented transformations in some cases) leads to important gains in data power, instruction power and performance. The overall performance and instruction power gains prove that the expected increase in the number of the instruction cache misses due to the code size increase (which is almost solely the result of the performance oriented transformations and not of the power optimizing class) has been small enough not to offset the gains achieved by the application of the proposed transformations themselves. This is true even for the case of the cavity detection application where the larger code size penalties are introduced.

7. CONCLUSIONS

A methodology for the performance efficient reduction of the data transfer and storage related power consumption in realizations of data dominated applications on multimedia processors has been developed. The methodology is based on the application of a number of code transformations in a specific order. The application of these code transformations sometimes introduces significant penalties in the code size which is an important design parameter implicitly

affecting the total system power and the performance. Experimental results have shown however, that the large gains achieved by the application of the proposed transformations in both system power and performance are hardly affected by the side effects of the code size penalties introduced, even when these penalties are relatively significant (which is the case only in smaller kernels).

REFERENCES

[1] J. M. Rabaey, M. Pedram, “Low Power Design Methodologies”, Kluwer Academic Publishers 1995.

[2] F.Catthoor, S.Wuytack, E.De Greef, F.Balasa, L.Nachtergaele, A.Vandecappelle, "Custom Memory Management Methodology - Exploration of Memory Organisation for Embedded Multimedia System Design", ISBN 0-7923-8288-9, Kluwer Acad.\ Publ., Boston, 1998.

[3] J. Van Meerbergen, P. Lippens, W. Verhaegh, A. Van Der Werf, “PHIDEO: high-level synthesis for high throughput applications”, Journal of VLSI signal processing, special issue on "Design environments for DSP" (eds. I. Verbauwhede, J. Rabaey), Vol. 9, No. 1/2, Kluwer, Boston, pp. 89-104, Jan. 1995.

[4] K. Masselos, F. Catthoor, C. E. Goutis, H. DeMan, "A Performance-Oriented Use Methodology of Power Optimizing Code Transformations for Multimedia Applications Realized on Programmable Multimedia Processors", IEEE Workshop on Signal Processing Systems, (Taiwan), November 1999.

[5] http://www.imec.be/vsdm/projects/mm_comp/. The IMEC multi-media compilation project–ACROPOLIS.

[6] M. Miranda, F. Catthoor, M. Janssen, H. DeMan, "High-level address optimisation and synthesis techniques for data transfer intensive applications", IEEE Trans. on VLSI systems, Dec. 1998.