DATA-REUSE EXPLORATION FOR LOW-POWER REALIZATION OF MULTIMEDIA APPLICATIONS ON EMBEDDED CORES

Nikos D. Zervas, Kostas Masselos and C.E. Goutis
Dept. of Electrical and Computer Engineering, University of Patras, Greece

ABSTRACT

Exploitation of data re-use in combination with the use of custom memory hierarchy that exploits the temporal locality of data accesses may introduce significant power savings especially for data-intensive applications. In this paper the effect of the data-reuse decisions on the power dissipation but also on area and performance of multimedia applications realized on embedded cores is explored. Experimental results prove that power savings of about 35% can be achieved through the exploitation of data re-use without introducing performance penalties in comparison to reference designs.

1 INTRODUCTION

In data-intensive applications like multimedia applications the power related to data storage and transfers forms a significant part of the total system's power budget. This statement holds for custom architectures as well as for programmable processor architectures [2,3]. In such applications significant power savings can be obtained by the joint exploitation of data re-use and the introduction of a custom data memory hierarchy that exploits the temporal locality of data memory references [4,5]. The basic idea is to introduce layers of smaller memories where copies of data from larger memories that exhibit high re-use are stored. The greater part of the accesses is moved to the smaller memories. This way power savings can be obtained since the accesses to smaller levels of memory hierarchy are less power costly.

In [5] the trade-off, according to which on the one hand data-memory power decreases because data are accessed from smaller memories but on the other hand the same power component increases because extra memory transfers are introduced and additionally area increases due to the extra memories and interconnect, was explored. However when targeting architectures based on embedded programmable processors extra issues have to be taken into account such as performance, code size and instruction memory power consumption. In [6] it was illustrated that the instruction memory power consumption of such systems cannot be neglected since in most cases is larger than the power consumed in the data memory hierarchy. The work presented here targets architectures based on a processor core and custom data memory hierarchy. This

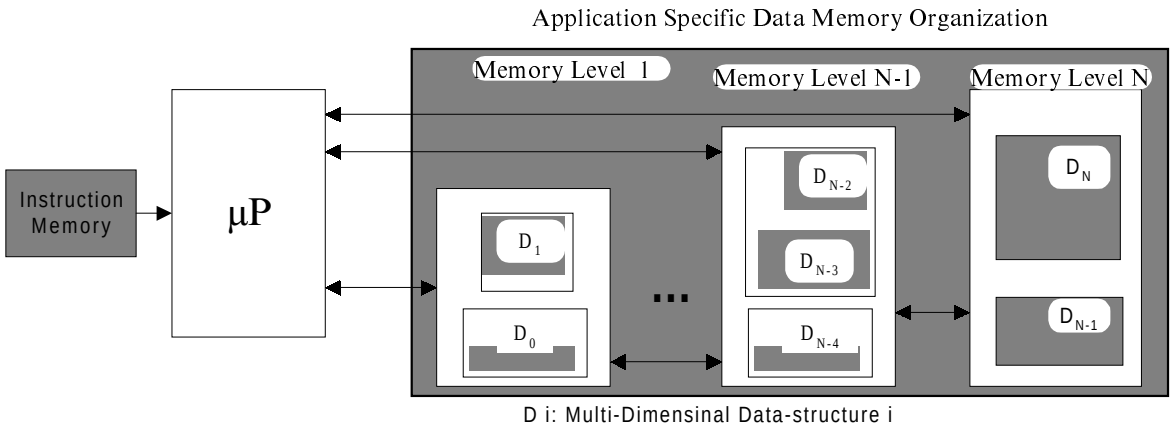

Figure 1: Target architecture

architecture allow for great optimization freedom of the data memory power consumption. In this paper the impact of data-reuse transformations on instruction and data memory power consumption and on performance is explored.

The rest of the paper is organized as follows: In section 2 the target architecture is defined. In section 3 the power model used is described. In section 4 the effect of the data-reuse transformations on power, area and performance is analyzed, while in section 5 the demonstrator application is described and the experimental results are presented. Finally in section 6 some conclusions are offered.

2 TARGET ARCHITECTURE

The target architecture (fig. 1) is based on one or multiple processor cores each one of communicating with its own instruction memory. For the organization of the data memory an application-specific architecture as the one proposed in [4] is used.

This organization can be derived after detailed analysis of the target application or set of applications. It may consist of a number of levels with different number of memory blocks per level. The levels may reside on-chip or off-chip depending on their size. Each memory block can be connected to the processor-core either directly or through the blocks of the levels that are closer to the processor. A global bus can be used for connecting the memory hierarchy to the processor core. This architecture can lead to low power consumption for a given application since it allows great freedom for optimization of the data memory power consumption. A separate memory hierarchy exists for the application code. The instructions are transferred to the core from a single instruction memory (preferably on-chip).

3 POWER MODEL

In this work only the power due to memory accesses and data transfers is evaluated since this power component dominates upon the total power budget of embedded

multimedia applications [2,3]. Additionally the power due to accesses to the background memories is only taken into account, since the power due to the accesses to the register files is much smaller [1]. The power consumed on the memory hierarchy is approximated by the power consumption due to on-chip memory accesses plus the power consumption due to off-chip memory accesses.

$$P_{MEM} = \sum_i P_{ON\ -\ CHIP}\_i + \sum_j P_{OFF\ -\ CHIP\ \_\ j} \quad (1)$$

The power cost of an on-chip memory transfer is approximated by the power cost of the memory access itself, since the power consumed upon on-chip busses is much smaller than the internal power consumption of on-chip memories The power consumed on on-chip memory accesses is estimated using Landman's model [8]. According to this, the power consumed on memory accesses is a function of the memory size, the access frequency, the technology, the number and the type (R or R/W) of ports and the number of bits of the accessed word. In this paper all memories are assumed to have a single read-write port. It is assumed that the power is linearly proportional to the access frequency. This assumes that the memories are in power-down mode when no transfer is going on, which is true for most modern memories [9]. So the power consumption due to memory accesses is given as:

$$P_{ON\ -\ CHIP} = P_{ACCESS} = f_{ACCESS} \cdot F(Word\_length, \#Words, V_{DD}) \quad (2)$$

For a given supply voltage the function F of equation *(2)* determines the relation between the memory power consumption and the memory size and depends only on technology. Such a function is described in [8] and is used here for the estimation of memory access power cost.

The bus driver, the chips' I/O pins (bonding wires and pads), the bus wires and the memory banks consume power during an off-chip memory access. High-level accurate estimation of the effective capacitance corresponding to each one of the above sources of power consumption is very difficult to be made. However a rough but still useful (for the aim of comparison) estimate can be acquired by considering typical values for the capacitance corresponding to each one of the above factors [10]. More precisely the off-chip memories are assumed to be the most power conscious ones i.e. low-power SRAM [9]. For the I/O pins, the bus driver and the bus wires a capacitance of 24pF per bus line is assumed. It is also assumed that in average half of the off-chip bus (bit) lines make a transition per off-chip memory access. The internal power consumption of the off-chip memories is also modeled by equation *(2)*. Thus:

$$P_{OFF\ -\ CHIP} = \left(Word\_lenght \cdot \frac{24 \cdot 10^{-12}}{2} \cdot {V_{DD}}^2 + F(Word\_length, \#Words, V_{DD}) \right) \cdot f_{ACCESS} \quad (3)$$

For the evaluation of the area occupied by the memories the Mulder model [7] is used.

4 THE DATA-REUSE TRANSFORMATIONS

As stated earlier in this paper in data-intensive applications the power related to data storage and transfers form a significant part of the total system's power budget. In such applications significant power savings can be obtained by exploiting data re-use through the introduction of a custom data memory hierarchy that exploits the temporal locality of data memory references [5]. In order to develop a power-efficient custom memory hierarchy scheme the first step is the *data-reuse exploration/decisions* [4,5]. More precisely sets of data that are reused often in a short fragment of time are identified; for each of them a smaller memory can be introduced, from which the data can then be accessed. The aim of the data reuse exploration is to decide for which of the data sets, it is power and area effective to introduce an extra memory.

The introduction of an extra memory usually comes with an area penalty, not only due to the area of the memory itself but also due to the additional interconnections required. However there are cases that area decreases when introducing extra memory layers as shown in [5]. This is due to the limited bandwidth that characterizes a memory. More precisely if a certain memory is referred above its maximum access frequency then this memory is split into two memories as of half the size. This splitting introduces an area overhead.

In terms of power several parameters have to be taken into account in order to decide whether or not an extra memory should be introduced. At first the power-effectiveness of memory hierarchy is strongly related to the data set reusability. In general the number of read operations from a copy of a data set in smaller memory must be greater than the number of read operations from the data set in the larger memory on the next hierarchical level. Additionally it is always preferable to move accesses from off-chip to on-chip memories, because the later accesses dissipate less power than the first. The above statements concerning area and power effect of the data-reuse transformations correspond to systems with custom implementation of the processing element(s).

The same statements are valid for programmable architectures as well, but in this case extra issues must be taken under consideration. The main reason for that is that in such case a significant part of the total power budget is consumed for the storage and transfer of the instructions. High-level parameters related to this component of energy dissipation are the code size and the number of executed instructions of the application's code. Code size determines the size of the instruction memory, and the number of instructions determines how many times this memory is accessed. For core-

based architectures when an extra memory is introduced the processor has to handle the additional transfers between the memories. Also addressing, and control expressions may be modified. This implies that a different number of instructions have to be executed and additionally the code size may also differentiate. An increase to the number of instructions means a performance overhead and that for the certain task more accesses to the program memory will be made and thus more energy will be dissipated. The code size is a design parameter for the target domain described in section 2, since it actually defines the size of the instruction memory. So according to the power models used here, an increase of the code size (memory size) leads also to an increase of the instruction memory power consumption. It is possible for the number of instructions and the code size to have contradictory effect on energy dissipation (i.e. the one increases while the other decreases).

The cost function used for the data reuse exploration during which the data-reuse choices are evaluated in terms of power and area should include parameters related to both data and instruction memory. Such a cost function is here proposed, a description of which follows:

$$Power\_\cos t = \sum_{c \in CT} [\mathrm{P_r}(word\_length(c), \#words(c), f_{read}(c)) + \\ + P_w(word\_length(c), \#words(c), f_{write}(c))] + P(instr\_word\_length, \#code\_size, f) \quad (4)$$

$$Area\_\cos t = \sum_{c \in CT} Area(\#word\_length(c), \#words(c) + Area(instr\_word\_length, code\_size) \quad (5)$$

$$Cost = a \cdot Power\_\cos t + b \cdot Area\_\cos t \quad (6)$$

Where: *a, b* are weighting factors for area/power tradeoffs, $P_{r/w}$ is the power estimate for the read/write operations, *Area* is the area estimate and c is a member of the set of all copy candidates (copy tree [5]) CT.

5 DATA-REUSE EXPLORATION FOR A REAL-LIFE APPLICATION

In this section the proposed cost function is used for the data-reuse exploration concerning a real-life application. The results show that maximum power-gains in the data-memory hierarchy are not always compatible with maximum power-gains in the total power dissipation. Also the significance of the power consumed in the instruction memory is illustrated and the close relation between this power-component and performance is shown. The role of code size as a parameter to the instruction memory power dissipation is underlined.

As demonstrator a typical video application was chosen namely the full-search motion estimation kernel. A detailed description of the algorithm can be found in [11]. Simulations where carried out using the luminance component of QCIF frames. The

reference window was selected to include 15×15 candidate-matching blocks. Blocks of 16x16 pixels were considered. The power consumption was estimated using the model described in section 3. In order to evaluate the performance the different version of codes were simulated on ARM 7 [12], which was assumed to be the embedded core. The latency for all memory accesses was assumed to be the same. This is a worst-case assumption for the effect of code transformations on performance, because the transformations introduce small buffers, which are close to the core and can be accessed faster than the larger memories in the higher levels of the data memory hierarchy. The estimation of the power consumption of the program memory was based on the results obtained by ARMulator [12]. For the motion-estimation kernel, twenty-five data-reuse transformations were examined. They involve memories for a line of reference windows (RW line), a reference window (RW), a line of candidate blocks (PB line), a candidate block (PB), a line of current blocks (CB line) and a current block (CB). In figure 2 the copy tree for the full-search kernel is illustrated. The dashed lines indicate a memory hierarchy level. Each rectangle is annotated with the number of the corresponding data-reuse transformation.

The data-reuse transformations usually make the code more complex and thus increase

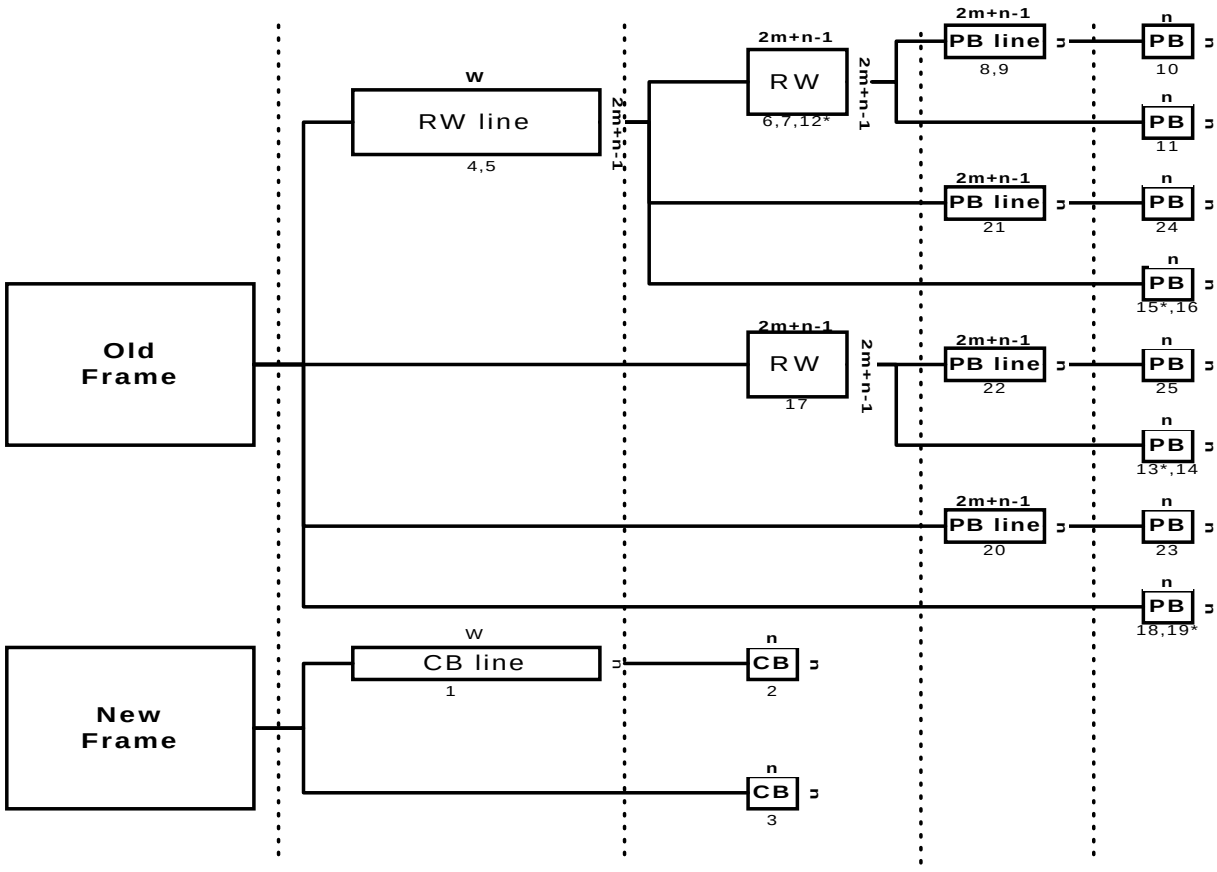

Figure 2: The copy tree for the full search motion estimation kernel

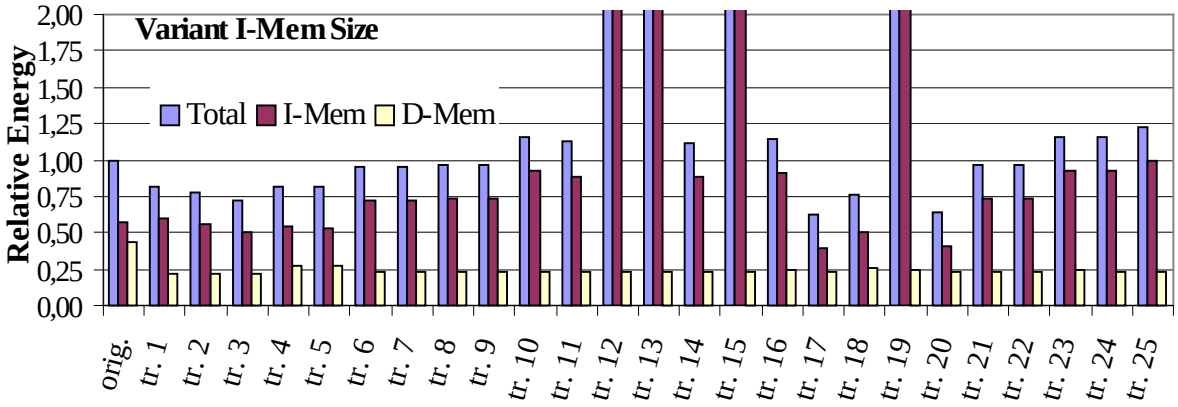

Figure 3: The effect on of the data-reuse transf. on energy for variant I-Mem size

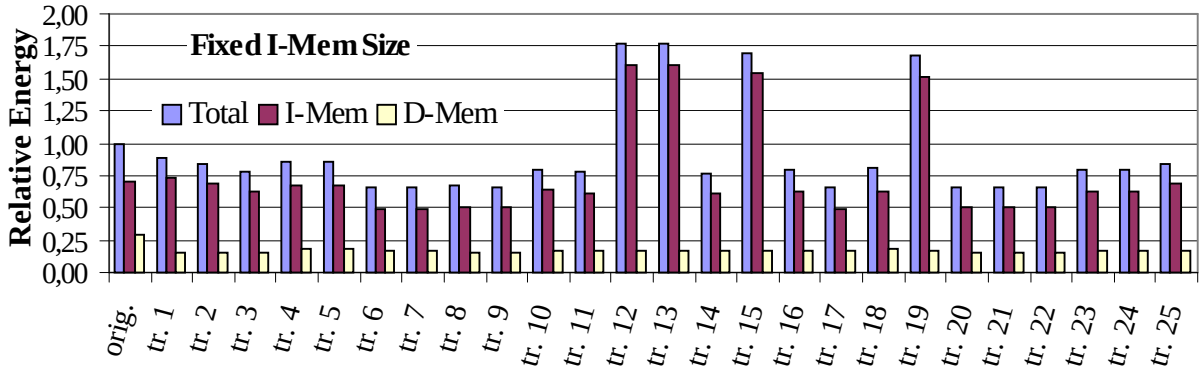

Figure 4: The effect on of the data-reuse transf. on energy for fixed I-Mem size

the code size. This implies an indirect effect on the system's power consumption since increase of the code size leads to an increase of the program memory. Increased size of the program memory leads to increased effective capacitance per accesses i.e. capacitance per instruction fetching. In figures 3 and 4 the total power consumption and its components (data-memory and instruction-memory power consumption) are illustrated.

In figure 4 the instruction memory size is fixed for all the data-reuse scenaria, while in figure 3 the instruction memory varies based upon the code size. The size of the instruction memory in the case of figure 3 is chosen equal to the smallest multiple of 1 Kwords that is larger or equal to the maximum of all the code sizes that correspond to all the data-reuse scenaria. Comparing the two figures shows the crucial effect that code size can have to power consumption. For instance transformations 10, 11, 14, 16, 23, 24 and 25 reduce power when the effect of code size is ignored (fig. 4). However, when code size is taken into consideration (fig. 3) the same transformations introduce a power penalty. Thus code size cannot be ignored when evaluating the power effectiveness of the data-reuse transformations, since an increase to the code size may overcome the power savings introduced in the data-memory.

Another interesting observation made is that the instruction memory power consumption is greater than the power consumed in the data memory hierarchy, even in

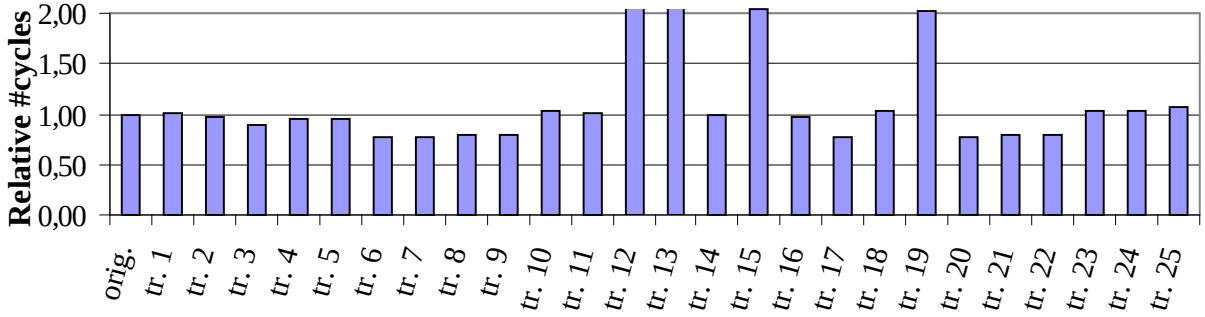

Figure 5: The effect of the data-reuse transformations on performance

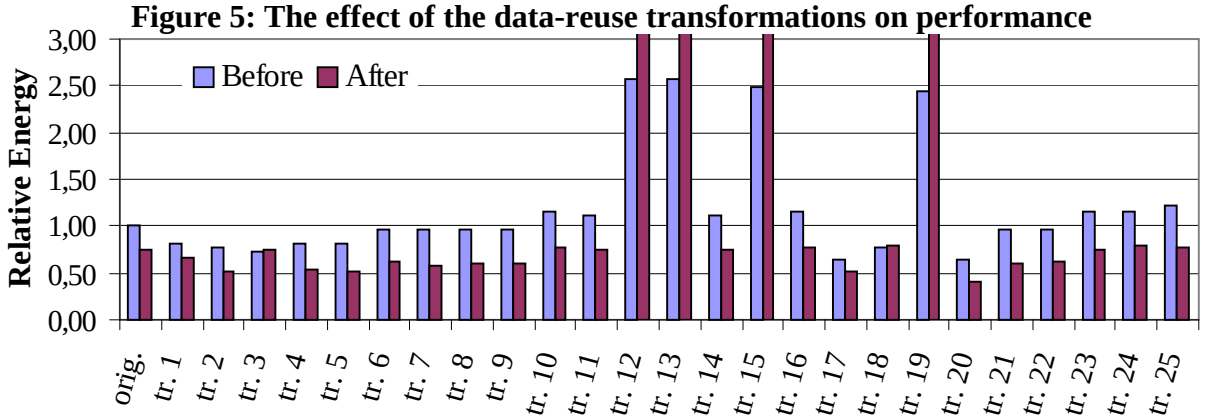

Figure 6: The effect of the performance optimizing transformations on power

this data dominated application. This is because the instruction word-length is usually larger than the data word-length. Thus according to the power model described in section 3 an instruction transfer is more power costly than a data transfer. Additionally the instruction memory is accessed with a much higher frequency than the data memory hierarchy. The difference in the access frequency is due to the fact that one instruction concerning a data-memory access goes with several non-memory access instructions (for indexing the array, loop conditions etc). Moreover the centralized program memory architecture is not the most power-efficient one. However the use of a distributed instruction memory hierarchy may lead to performance penalty. Performance is of critical importance in such systems and this is the reason why distributed instruction memory architectures are not followed in practice.

In figure 5 the effect of the applied data-reuse transformations on performance is illustrated. The data-reuse transformations also affect performance since they introduce extra memory accesses and modify address and control expressions. There is a relation between program-memory energy dissipation and performance (in terms of machine cycles), since both of them depend on the number of executed instructions. Experiments have shown that gains in performance usually come with energy savings in the program memory. In cases that gains in performance are obtained by paying a code size overhead, energy dissipated in the program memory is reduced due to decreased number of executed instructions but on the other hand the same energy component is increased due to increase of the size of the program memory. In such cases weather or

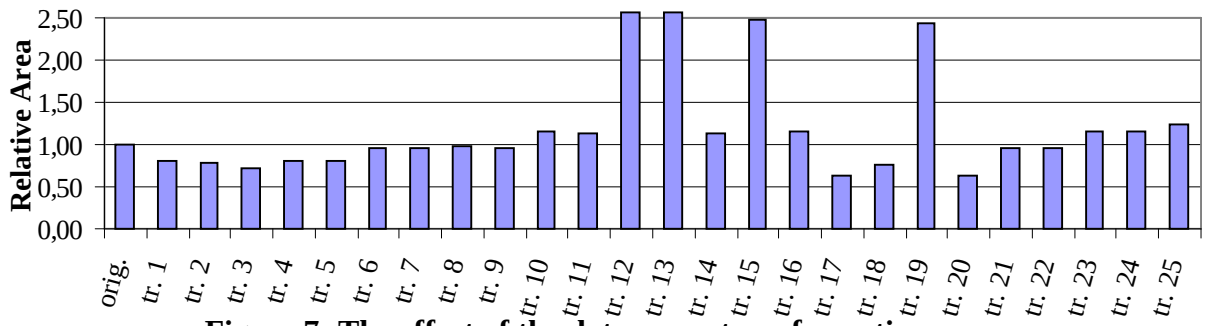

Figure 7: The effect of the data-reuse transformation on area

not energy savings are introduced depends on which of the contradictory effects dominates. Some data-reuse transformations introduce performance penalties while some other reduces the number of machine cycles. In any case the "traditional" performance optimizing code transformations (loop invariant code motion, common sub-expression elimination etc) can be applied in order to further improve the speed of the application execution.

The effect of the application of performance-optimizing code transformations on energy dissipation is illustrated in figure 6. This category of transformations is considered up to now as energy optimizing code transformations also, since they reduce the number of required instructions for the completion of a certain task, and thus reduce the energy dissipated in the program memory by this task. However the effect of the performance-optimizing transformations on energy is not easily predictable, since they affect both number of executed instructions and code size. Additionally all the codes cannot be evenly optimized. Thus the comparison between the different data-reuse schemes must be made after the application of all possible additional transformations such as performance optimizing transformation. An example is given in figure 6: Before the application of the performance optimizing transformation the power optimal data-reuse scheme is transformation 17 while after their application the optimal data-reuse scheme is transformation 20. In [5] for the same application it was inferred that the power-optimal scenario for the data memory is the data-reuse of the line of reference windows, reference window and candidate block from the previous frame (tr. 11) and the data-reuse of the current block from the current frame (tr. 3). However the reuse of the line of reference windows, reference window and candidate block from the previous frame (tr. 11) leads to a power penalty when instruction-memory power is taken into consideration (fig. 4).

The area corresponding to the data-reuse scenaria is illustrated in figure 7. As mentioned in section 4 there are cases where the introduction of an extra memory also optimizes area. Specifically the splitting of memories that are initially accessed above their maximum access frequency is avoided with the introduction of extra memories.

This is due to the fact that accesses are moved from the initial memories to the introduced extra memories, resulting this way the reduction of access frequencies that can become lower than the maximum access frequency for all memories. In this paper it is assumed that the maximum access frequency for all memories is 50 MHz.

6 CONCLUSIONS

In this paper the effect of the data-reuse decisions on the power consumption, area and performance of systems based on embedded cores was explored. The necessity to take into consideration instruction-memory (except from data-memory) for the evaluation of the effect of the data-reuse decisions on total system's power was shown. The effect of high-level parameters, such as code size, number of instructions and number of memory accesses on memory power consumption, area and performance were analyzed and a cost function that can be used during data-reuse exploration was proposed. The experimental results showed the significant power savings that can be introduced by exploiting data re-use in combination with a custom memory hierarchy.

REFERENCES

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

[2] T. H. Meng, B. Gordon, E. Tsern, A. Hung, "Portable video-on-demand in wireless communications", in *Proc. of the IEEE*, vol. 83, No 4, pp. 659-680, April 1995.

[3] J.-M. Puiatti, C. Piquet, E. Sanchez, J. Llosa, "Low-Power VLIW Processors: A High-Level Evaluation", in *proc. of PATMOS' 98*, Octomber 1998, pp. 399-408.

[4] 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, Boston, 1998.

[5] S. Wuytack, J.–P. Diguet, F. Catthoor, "Formalized Methodology for Data Reuse Exploration for Low-Power Hierarchical Memory Mappings", in *IEEE Transactions on VLSI Systems*, Vol. 6, No. 4, pp. 529-537, Dec 1998.

[6] N.D. Zervas, K. Masselos, C.E. Goutis, "Code Transformations for Embedded Multimedia Applications: Impact on Power and Performance", in Proc. of *Power Driven Microarchitecture Workshop, ISCA '98*, June 1998, Barcelona, Spain, pp 20-24.

[7] J. M. Mulder, N. T. Quach, M.J. Flynn, "An Area Model for On-Chip Memories and its Application", *IEEE Journal of Solid-State Circuits*,Vol. SC26,No.1,pp.98-105, Feb. 1991

[8] P. Landman, Low power architectural design methodologies, Doctoral Dissertation, U.C. Berkeley, Aug. 1994.

[9] K. Itoh, K. Sasaki and Y. Nakagome, "Trends in Low-Power RAM Circuit Technologies", *Proceedings of the IEEE*, volume 83, pp. 524-543, April 1995.

[10] F. Catthoor, S. Wuytack et al., Custom Memory Management Memthodology, Kluwer Academic Publishers, Boston, 1998

[11] V. Bhaskaran and K. Konstantinides, Image and Video Compression Standards, Kluwer Academic Publishers, 1994.

[12] ARM software development toolkit, v2.11, Copyright 1996-7, Advanced RISC Machines.