MEMORY HIERARCHY EXPLORATION FOR LOW POWER ARCHITECTURES IN EMBEDDED MULTIMEDIA APPLICATIONS

N. Kavvadias, A. Chatzigeorgiou, N. Zervas[1], *S. Nikolaidis*

Electronics and Computers Div., Department of Physics
Aristotle University of Thessaloniki, 54006 Thessaloniki, Greece

[1]VLSI Design Laboratory, Department of Electrical Engineering
University of Patras, Patras 26500, Greece

ABSTRACT

Multimedia applications are characterized by an increased number of data transfer and storage operations due to real time requirements. Appropriate transformations can be applied at the algorithmic level to improve crucial implementation characteristics. In this paper, the effect of the data-reuse transformations on power consumption, area and performance of multimedia applications realized on embedded cores is examined. As demonstrators, widely applicable video processing algorithmic kernels, namely the row-column decomposition DCT and its fast implementation found in MPEG-X, are used. Experimental results prove that significant improvements in power consumption can be achieved without performance degradation by the application of data-reuse transformations in combination with the use of a custom memory hierarchy.

1. INTRODUCTION

The popularity of multimedia systems used for computing and exchanging information is rapidly increasing. With the emergence of portable multimedia applications (mobile phones, laptop computers, video cameras, etc) the power consumption has been promoted to a major design consideration due to the requirements for long battery life, large integration scale and the related cooling and reliability issues [1][2]. Consequently, there is great need for power optimization strategies, especially in higher design levels, where the most significant savings are achieved.

A number of code transformations can be applied to any algorithm aiming at a memory hierarchy where copies of data from larger memories that exhibit high data-reuse are stored to additional layers of smaller memories. In this way, exploiting the temporal locality of data memory references [1], the greater part of the accesses is moved to smaller memories. Since accesses to smaller levels of the memory hierarchy are less power costly, significant power savings can be obtained [3].

Multimedia applications require increased processing power for manipulating large amounts of data in real time. Two general implementation approaches exist, to meet this demand. The first is to use application specific integrated circuits (ASICs). This solution leads to high performance, small area and power consumption. However it completely lacks flexibility since only a specific algorithm can be mapped on the system. The second solution is to use embedded instruction set processors. This solution requires increased area and power compared to the ASIC solution while achieving lower performance. The main advantage of such systems compared to the first solution is programmability. For multimedia applications realized on ASICs, the dominant factor in power consumption is the one related to data storage and transfers [3]. However in programmable platforms, the power consumption due to instruction memory accesses usually dominates the power related to data storage and transfers [4].

In this work, a formalized data storage and transfer exploration methodology [1][3] is followed for the exploration of the effect of data-reuse transformations on the implementation of two well-known DSP algorithms. These transformations lead to the determination of appropriate memory hierarchy schemes with a number of intermediate on-chip small memory set between the processor unit and the off-chip main memory. A comparative study concerning power, performance and area, indicates that the most effective solution can be achieved from the right combination of data-reuse decisions and the development of an appropriate data memory hierarchy.

2. DATA REUSE TRANSFORMATIONS

In data-dominated applications such as multimedia algorithms, significant power savings can be achieved by developing a custom memory organization that exploits the temporal locality in memory accesses [1]. According to the proposed methodology, data sets that are often being accessed in a short period of time are identified and placed into smaller memories leading to a new memory hierarchy. Hence, power savings can be obtained by accessing heavily used data from smaller foreground memories instead of large background memories. Such an optimization requires architectural transformations that consist of adding layers of smaller memories to which frequently used data can be copied. Consequently, there is a trade off here; on the one hand, power consumption is decreased because data is now read mostly from smaller memories, while on the other hand, power consumption is increased because extra memory

This work was supported by the ED 501 PENED'99 project funded by G.S.R.T. of the Greek Ministry of Development and the European Union.

transfers are introduced. An exploration of all architectural alternatives is required for finding the optimum solution.

This data-reuse exploration is performed by applying a number of code transformations to the original code, which are determined by the group of data sets that are being used in the algorithm. For the DCT algorithms the possible data-reuse transformations together with the introduced levels in the memory hierarchy, which correspond to reused data sets, are shown in Fig. 1. These transformations are extracted according to the methodology described in [3]. The parameters for these algorithms are: the size of the current frame ($N \times M$) and the current block size ($B \times B$). These transformations involve memories for a line of current blocks (CB line) of size $M \times B$, a current block (CB) of size $B \times B$ and a line of current block pixel line (CP line) of size $B \times 1$. The number of the corresponding transformation and the size of the introduced memory, given parametrically, annotate each rectangle in the figure. Transformation #7 will not be taken into account due to its proven inefficiency [3].

3. TARGET ARCHITECTURE

An appropriate target architecture is considered for the implementation of the multimedia algorithms, to exploit all possibilities for power consumption reduction resulting by the application of the data-reuse transformations. The target architecture is based on an embedded processing unit (programmable or not programmable) communicating with data memory and optionally with an instruction memory (ROM) depending on whether the system is programmable or not. The data memory hierarchy is fully customizable in terms of selecting the number of memory layers and the size of the memory modules at each layer. A global bus is considered for the communication of memory blocks with the processor. Memory blocks reside on-chip except for the first memory layer, which is an off-chip memory.

In the proposed approach only the power due to accesses to foreground and background memories is taken into account since they constitute the higher percentage of the power dissipation of the implementation [3]. According to the power model that has been used, the power consumed on memory accesses is a function of the memory size, the access frequency, the technology, the number and the type (R and R/W) of ports. For example, assuming for a given technology that all memories have a single read-write port and that the power is linearly proportional to the access frequency (f) [4], the power consumption is given as:

$$P_{access} = f_{access} \cdot F(Word\_length, \#words, V_{dd}) \quad (1)$$

where the word length and the number of words define the used memory size.

For the evaluation of the performance for the transformed algorithms compared to the corresponding original codes, simulations on a cycle-accurate RISC prosessor simulator (ARMulator [7]) are performed. The performance metric is given in the following equation:

$$Performance = (Num\_of\_cycles)^{-1} \quad (2)$$

where performance denotes the execution speed and is inversely proportional to the number of the required cycles to execute the object code.

The corresponding area is estimated with the use of Mulder's model [5] characterized for an industrial part library.

4. APPLICATION DEMONSTRATOR: THE ROW-COLUMN DECOMPOSITION DCT

As test vehicles, a typical and a fast Discrete Cosine Transform (DCT) algorithm [6] will be used. The DCT algorithm, which can exploit spatial redundancy, plays an important role in multimedia data compression standards such as JPEG, MPEG-X and H.26X. The usefulness of the DCT arises from its sparseness due to the fact that a set of coefficients is produced that typically has a lower dynamic range than the input pixel data. The fast DCT is an optimized version of the typical one and runs with a significantly higher computational speed.

The description of the row-column decomposition DCT algorithm is given in C pseudo-code form in Fig. 2. In the row-column decomposition version of the DCT, the transformation is separable into 1-D DCTs, and the 2-dimensional algorithm results, by performing row-wise 1-D transform followed by column-wise 1-D transform with intermediate transposition.

The algorithm structure basically consists of a double nested loop including a triple nested one. For the calculation of the coefficients, a frame size of $N \times M$ and blocks of $B \times B$ pixels are considered. In the specific example the luminance component of QCIF frames (N=144, M=176) is DCT coded in 8x8 blocks (B=8) in a scanner-like way.

5. DESIGN EXPLORATION OF THE DCT ALGORITHMS

As already mentioned, two design approaches exist for the implementation of multimedia applications, Application Specific Integrated Circuits (ASICs) or Application Specific Instruction set architecture Processors (ASIPs) and general-purpose processors. Since the instruction memory related power component is different for each approach, each case has to be studied separately.

In the case of ASICs the total power budget is dominated by the power component due to accesses to data memory. Application specific processors (ASIPs) with a custom instruction set suited to the target algorithm are also considered to belong in this case. Since their instruction set is tailored to the target algorithm to achieve fast execution, the resulting code size and the number of executed program cycles is small. These in turn lead to a smaller number of accesses to the instruction memory and reduced memory size reducing significantly the power component due to instruction memory accesses. In Fig. 3, the energy consumption due to accesses to data memory layers for the processing of one frame is presented for both algorithms (originals) and all the transformations. As it can be seen the fast version of the DCT algorithm presents almost five times lower power consumption compared to the typical one. The transformations affect the power component in an analogous way for the two algorithms. However, the percentage reduction of power in case of the typical algorithm is much higher. In both cases the most power efficient data-reuse transformation is #3. A

power reduction factor about of 10 and 3 is achieved for the typical and fast DCT, respectively. It can be noticed that transformation #5 (as well as #7, which is not given) corresponds to an inefficient memory architecture. The requirements in memory area (excluded the main off-chip memory) for both algorithms and all the transformations are given in Fig. 4. The introduction of additional data memory layers comes with an inevitable area penalty. However due to the small size of the CB buffer, transformations #3 and #4 impose acceptable area occupation increase. As it is shown, the case for the fast DCT presents slightly higher requirements.

If a general-purpose processor is used, the power component due to instruction memory accesses is not negligible and has to be taken into account for the estimation of the total power consumption. In order to prove the dominant role of instruction memory in the overall power consumption, simulations using the ARMulator have been performed [7]. In Fig. 5 the instruction memory energy consumption is shown as part of the total power consumption, for both algorithms. The power consumption due to instruction memory is about 1.5 (6) and 5 (30) times higher than that of data for the fast and typical DCT original (transformed) algorithms, respectively. Transformation #3 remains the most power efficient solution for both algorithms. It has to be mentioned, that in the results shown in Fig. 5, the power consumption due to instruction memory accesses is overestimated. That is because no instruction caching was taken into account, which for data dominated applications (where cache misses do not occur frequently) would result in a smaller number of accesses to the instruction memory.

The code size graph depicted in Fig. 6, shows that the implementation of the fast DCT on a programmable platform is more area demanding than the standard DCT, since area is a function of code size. In addition to that, in some cases the application of transformations can even reduce code size in benefit of instruction memory area demands.

Almost no performance penalty is introduced by the application of the data-reuse transformations, as shown in Fig. 7. This is due to: (a) The small number of additional control operations introduced in the transformed algorithm. This is because the algorithm presents high regularity since no conditional statements exist within loops. (b) The relative reduction of data addressing and arithmetic operations in the transformed algorithm as a result of simplified addressing equations. This conclusion has been verified by profiling trace data obtained from executing the object code by the instruction-level profiler software IPROF reported in [8]. Consequently, the selection of the most appropriate memory hierarchy should be based only on power criteria, whereas in some cases area occupation could be considered.

6. CONCLUSIONS

In this paper the effect of data-reuse transformations on the energy consumption, area and performance of embedded systems implementing multimedia applications has been presented. A number of data-reuse transformations have been applied on DCT algorithms to achieve significant power savings by moving a large amount of background accesses to smaller foreground memory banks. The obtained results indicated a reduction in data memory power consumption of about 70-90% when custom memory hierarchy is used compared to the case without memory hierarchy. If a general-purpose processor is used, the instruction memory power dissipation has to be considered, since it often dominates total power consumption. It has been proven that with this methodology, the hardware-software designer is given an additional degree of freedom for implementing an energy-efficient architecture for multimedia applications.

REFERENCES

[1] F. Catthoor, S. Wuytack et al, *Custom Memory Management Methodology*, Kluwer Academic Publishers, Boston, 1998.

[2] A. Chandrakasan and R. Brodersen, *Low Power Digital CMOS Design*, Kluwer Academic Publishers, Boston, 1995.

[3] S. Wuytack, J-P. Diguet, F. Cathhoor, H. De Man, "Formalized Methodology for Data Reuse Exploration for Low-Power Hierarchical Mappings", *special issue of IEEE Transactions on VLSI Systems on low power electronics and Design*, Vol. 6, No. 4, pp. 529-537, December 1998.

[4] N. D. Zervas, K. Masselos, C.E. Goutis, "Data-reuse exploration for low-power realization of multimedia applications on embedded cores", *Proc. Of 9th Int. Workshop on Power and Timing Modeling, Optimization and Simulation* (PATMOS'99), pp. 71-80, October 1999.

[5] J.M. Mulder, N.T. Quach, N.J. Flynn, "An area model for on-chip memories and its application", *IEEE Journal of Solid-State Circuits*, Vol. SC-26, pp. 98-105, Feb 1991.

[6] K.R. Rao, P. Yip, *Discrete Cosine Transform - Algorithms, Advantages, Applications*, Academic Press Inc., London, 1990.

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

[8] P. Kuhn, *Algorithms, Complexity Analysis and VLSI Architectures for MPEG-4 Motion Estimation*, Kluwer Academic Publishers, Boston, 1999.

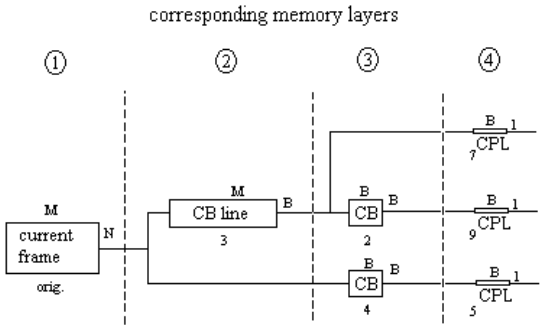

Figure 1. Search tree for data reuse decision for the DCT algorithm

```
/* Transposition of the input data array */
for(i=0; i<N/B; i++)
  for(j=0; j<M/B; j++)
{
      /* Transposition of the block being processed */
}

 /* First 1-D DCT for the rows of the initial data array */
   for(i=0; i<N/B; i++)        /* For all blocks in a frame */
     for(j=0; j<M/B; j++)
{
      for(k=0; k<B; k++){   /* For all pixels in a block */
        for(m=0; m<B; m++){
        temp=0;
          for(l=0; l<B; l++)
          temp+=coeff[m][l]*image[B*i+l][B*j+k];

        output1[B*i+m][B*j+k]=temp;
       }
     }
}
/* Same steps follow for transposition of the 1st output data
array and the application of 1-D DCTs for the columns of the 1st
output data array */
```

Figure 2. The row-column decomposition DCT algorithm

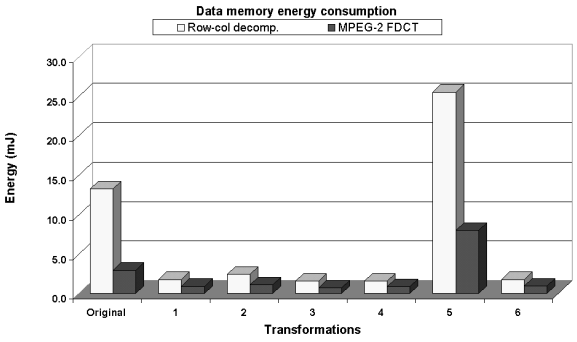

Figure 3. Data Memory Energy Consumption

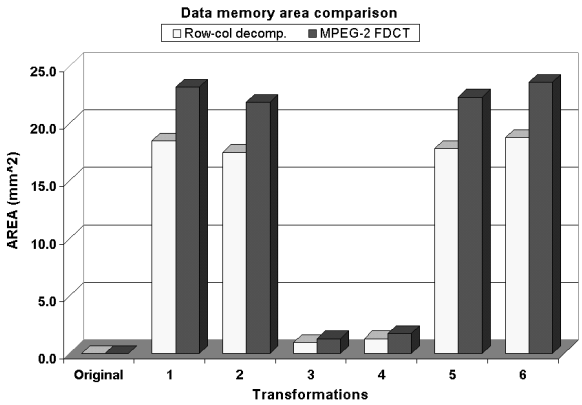

Figure 4. Data Memory Area Occupation

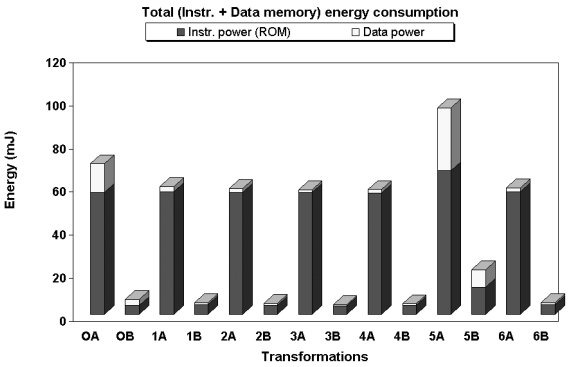

Figure 5. Instruction Energy Consumption vs Total Energy Consumption (A→typical, B→fast)

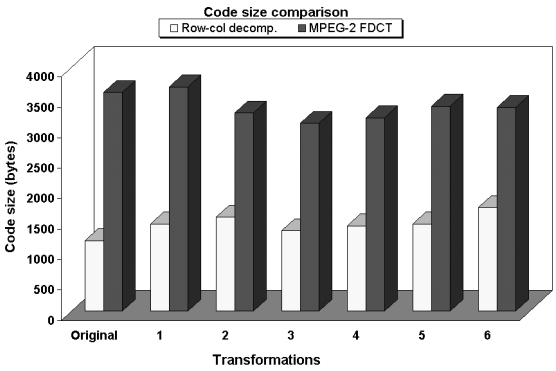

Figure 6. Code size for various transformations

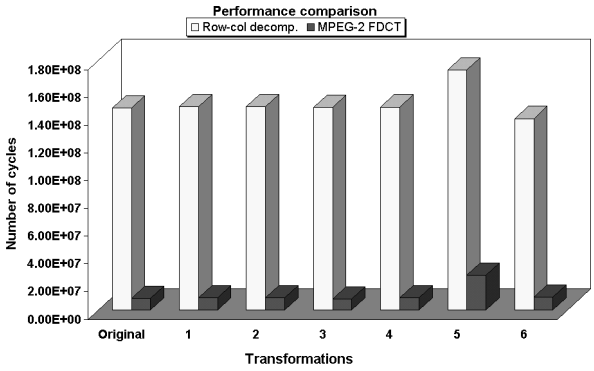

Figure 7. Code performance for various transformations