IEEE International Conference on Electronics, Circuits and Systems (ICECS'2001), September 2001, Malta

POWER REDUCTION FOR MULTIMEDIA APPLICATIONS THROUGH DATA-REUSE MEMORY EXPLORATION

S. Kougia, A. Chatzigeorgiou, S. Nikolaidis

Electronics and Computers Division, Department of Physics
Aristotle University of Thessaloniki, 54006 Thessaloniki, Greece
{mkoug, achat}@skiathos.physics.auth.gr, snikolaid@physics.auth.gr

ABSTRACT

Power consumption of multimedia applications executing on embedded cores is heavily dependent on data transfers between system memory and processing units. In this paper, a power optimizing methodology based on data-reuse decisions and the development of a custom memory hierarchy is extended in order to determine the optimal solution in a rapid and reliable way. Data-reuse transformations are applied on a typical motion estimation algorithm in order to reduce the data-related power consumption by moving background memory accesses to smaller foreground memories, which are less power costly. Fast exploration of the design space is achieved by extracting analytical expressions for the number of accesses to data and instruction memories.

1. INTRODUCTION

Most embedded applications in the multi-media and the telecommunication domain turn out to be data-dominated with the data-related power consumption affecting heavily the total power budget [1]. At the same time an increasing number of real time applications such as image and video processing is being available on portable devices. Low power consumption is of primary importance for such systems because of the requirements for long battery life and large integration scales, and the related cooling and reliability issues [2].

In order to confront the performance requirements of multimedia applications two general implementation choices exist: The first is to use dedicated hardware (non-programmable or partially programmable platforms), which offers maximum performance since such systems are tailored to each targeted application. However, this solution comes at a high cost while it completely lacks flexibility. The alternative is to use embedded instruction set processor cores (fully programmable platforms), which offer increased flexibility, smaller time-to-market and opportunities for reuse at the cost of significantly lower performance than the previous solution.

Cathoor et al [1] suggested that 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 blocks of smaller memories. In this way, the greater part of the accesses is moved to smaller on-chip memories. Since accesses to smaller memory blocks are less power costly, significant power savings can be obtained [1]. A formalized methodology for data-reuse exploration has been proposed in [3] where a systematic way on how to decide on the optimal memory hierarchy is developed. However, instruction related power consumption is not taken into account since only custom architectures are considered. In [4] it was illustrated that the instruction memory power consumption in embedded programmable processors cannot be neglected since in most cases is larger than the power consumed in the data memory hierarchy.

All previous work adopt a simulative approach in order to determine the optimal solution from a pool of possible implementations which is the end result of the data-reuse exploration and decision methodology [4]. Each alternative has to be implemented, compiled and executed on a processor simulator in order to evaluate its power efficiency. Unfortunately, since the design space can be extremely large [3], the previous approaches can be very time and effort consuming. Moreover, this process cannot be automated prohibiting the development of appropriate EDA tools that could be used early in the design process.

In this paper, an analytical approach is proposed for data-reuse exploration, during which power efficient code transformations are examined using as demonstrator application the three-step logarithmic search motion estimation algorithm [5]. The proposed methodology offers the possibility to explore a pool of possible design decisions and to determine in short times the optimum solution, considering the trade-off between power, performance and area.

2. DATA-REUSE TRANSFORMATIONS

The target architecture is based on embedded processing units, each one communicating with one or more data memory layers and optionally with its own instruction memory, depending on whether the system is programmable or not. Instruction memories are considered to be on-chip single-port ROMs. Data memory blocks communicate with the processor over a global bus and are considered to reside on chip except for the first memory layer that holds the previous and the current frame, which is an off-chip 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 European Union"

According to the power model that has been used, the power consumed due to accesses to *i*-th memory layer, is directly proportional to the number of accesses and depends on the size and the number of ports, of the memory, the power supply and the technology.

In data-dominated applications such as multimedia algorithms, significant power savings can be achieved by developing a custom memory organization [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 memory blocks. In this way, the total number of accesses increases while the average power cost of each access decreases since smaller memory blocks have a lower energy cost per access. An exploration procedure has to be established for determining the optimum, in terms of energy consumption, memory hierarchy.

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 motion estimation algorithms the possible data-reuse transformations together with the introduced levels in the memory hierarchy, are shown in Fig. 1. The parameters for these algorithms are: the size of the current and previous frame ($N \times M$), block size ($B \acute{} B$) and p which determines the search region ($p \acute{} p$) around the location of a specific block in the current frame. These transformations 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). Capital letters *C*, *P* indicate current and previous frame respectively, in which the transformation takes place.

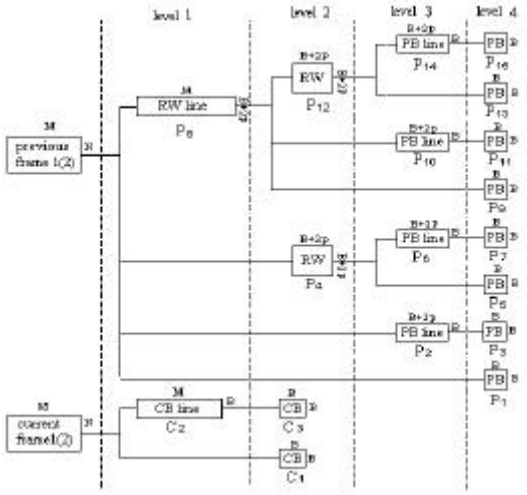

Fig. 1: Copy tree for a motion estimation kernel

3. DESIGN EXPLORATION: AN ANALYTICAL APPROACH

The application of the existing methodology [1] for the Data Transfer and Storage Exploration (DTSE) step, including its extension for the case of embedded processor cores as proposed by [4], [6] requires the evaluation of each code transformation by executing the corresponding code on a simulator, such as the ARMulator [7]. This process can be extremely time consuming, leading to prohibitive requirements in terms of time and effort. In contrast to previous methodologies, the total number of accesses to each memory layer and the number of executed instructions are analytically calculated in the proposed approach. In this way it is possible, without having the transformed code simulated, to evaluate the power consumption for each transformation at very short times. It will be proved that it is not necessary to implement all possible transformations in order to evaluate their power efficiency. Rather, only a few key transformations have to be studied to extract the required information.

As already mentioned, there are two general approaches for the implementation of multimedia applications: Application specific and general purpose systems. Since the power component due to instruction memory accesses is different for each approach, each case has to be treated separately:

3. 1 Application Specific Units

In the case of an application specific system the total power budget is dominated by the power component due to accesses to data memory, since either there is no instruction memory or the number of executed program cycles is very small.

The first step of the proposed methodology is to define a set of key transformations from which the required information for the number of data accesses can be extracted. In the next step the number of data accesses is analytically calculated enabling the use of automated tools. Finally, in the last step, the number of accesses is fed to the power model that is being used in order to calculate the data-related power consumption.

STEP 1 (*Definition of key transformations*)

The number of data accesses to each memory layer is the sum of the accesses, which are made in order to update this memory from its previous memory layer, and the accesses which are made in order to update the next memory layer. However, the number of accesses, which are made in order to update a memory layer from a previous one, is independent of the previous layer from which data are read and depends only on the following memory layer.

According to the above, in order to calculate the number of accesses for each data transfer between memory layers a reference table [8], has to be built. The dimension of the Table is $(n+1)\times(n+1)$ where n are the memory levels of the applied hierarchy. The contents of the cells provide the total number of accesses for the memory layer which is indicated by the corresponding column, when it is followed by the memory layer that is indicated in the corresponding line. Only transformations for the previous frame are considered. That is because for the current frame transformation C_1 yields always better results, since

current blocks have no overlap and thus no advantage of a line of current blocks can be made. To calculate the total power consumption, all involved memory layers are defined and the power cost of accesses between them is summed.

We define as 'key' transformations the minimum set of transformations, which are required in order to fill the Table with the necessary information. For the case of a memory hierarchy with four levels (Fig. 1), all possible transformations are 15 while the key transformations are only 8, leading to a ratio of 0.53.

STEP 2 (*Calculation of memory accesses*)

Usually motion estimation algorithms are constructed upon loops. According to the proposed methodology analytical expressions for the number of memory accesses are extracted using the loop hierarchy, without the need to compile or simulate the corresponding code.

We assume a general structure of loops like the one in Fig. 2 where n_i is the number of iterations of i loop and *array*(l) is the array (memory) for which the data accesses have to be obtained. By parsing the code serially, it is detected whether a loop is started or finished, which is the number of iterations of the loop, how many times data are being read or written from/to the array within a loop. Three temporary variables are used, one (x) for holding the depth of the nested loops and two (*read_array*(l)) and (*write_array*(l)) for holding the number of times data are being read or written to *array*(l). In the beginning of each loop, x is multiplied by the number of iterations of this loop (Fig. 2) while at the end of the loop x is divided by the number of iterations in order to keep track of the nested loops. Variables *read_array*(l) and *write_array*(l) are updated each time a read or write operation is encountered taking into account the depth of the corresponding loop.

According to the above, it is possible to extract analytical expressions for the number of data accesses to each memory layer.

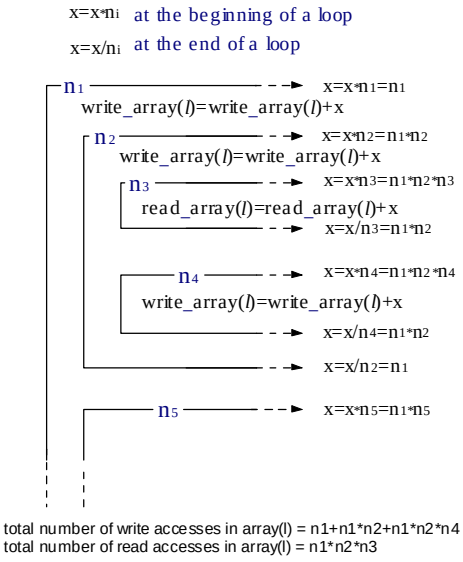

Fig. 2: General loop structure

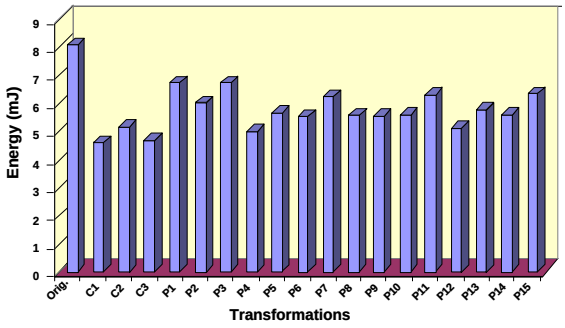

Fig. 3: Data Memory Energy Consumption

In the case of conditional accesses to a memory layer, the number of accesses to an array should be equal to the number of times the corresponding conditions are fulfilled.

The proposed analytical expressions have been validated with comparisons to simulation results, proving that the analytical calculations are error free leading to an accuracy of 100 % in all cases. In Fig. 3 the energy consumption due to accesses to data memory layers is presented for all transformations and compared to that corresponding to the original code.

Except for the fast calculation of the power consumption, these analytical expressions allow for the exploration of the whole design space by varying parameters such as the frame size (N,M), the size of the search space (p) and the block size (B).

3.2 General Purpose Processors

A general purpose processor obviously is not optimally designed for any algorithm resulting in larger programs and consequently instruction memory and in higher number of executed cycles. As a result, the power component due to instruction memory accesses is no longer negligible [4]. In order to prove the dominant role of instruction memory in the power consumption the ARMulator has been used [5]. In Fig. 4 the power consumption due to instruction memory accesses is shown as part of the total power consumption for the original and the transformed codes.

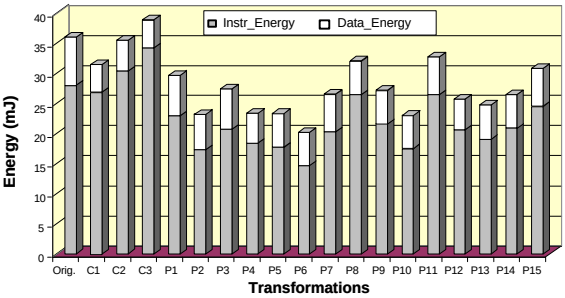

Fig. 4: Instruction memory energy consumption over total energy consumption

According to the proposed methodology the total number of executed instructions is calculated parametrically, similarly to data accesses. Only the same key transformations have to be examined and a

corresponding table for the accesses in instruction memory has to be derived.

This is in contrast to previous works, which adopt an experimental approach requiring the compilation and execution of each code transformation on a processor simulator in order to obtain the total number of executed instructions, which correspond to accesses to the instruction memory. The number of assembly instructions in the proposed methodology is obtained according to the number of iterations for the nested loops that implement each of the applied motion estimation algorithms. In its general form, each double nested loop containing k instructions of the form :

for(i=0; $i<n_0$; i++)
  for(j=0; $j<n_1$; j++)
    { #k instructions }

corresponds to :

$\#instr = c_1 + c_2\, n_0 + n_0\,[c_1 + (c_2 + k)\, n_1] = f(n_0, n_1, k)$

assembly instructions. A double loop is selected for the definition of function f due to the two-dimensional nature of motion estimation algorithms. Constants c_1, c_2 are fixed for each loop independently on the number of iterations (for the ARM processor c_1=4, c_2=5).

Starting from the most inner loop, the number of executed assembly instructions is calculated and the result is added to the number of instructions between nested loops. Since the indices of each loop are determined by the algorithmic parameters M, N, B and p, the total number of instructions is obtained as a polynomial function of these parameters. The proposed expressions calculate accurately the absolute number of executed assembly instructions except for a small deviation which is due to high-level statements that are being compiled to different number of assembly instructions according to the processor state (i.e. number of available registers) and due to program parts whose execution depends on the image content.

It should be mentioned that in the results shown in Fig. 4, 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.

Since the introduction of additional memory layers comes with an area penalty, this parameter has also to be considered. Obviously, code transformations affect the processor performance, i.e. the number of cycles required for the execution of the code. In Fig. 5 the effect of the proposed code transformations on performance is shown. It becomes clear, that the decision of the optimal solution should considered the trade-off between power savings, area and performance penalty.

4. CONCLUSIONS

A novel methodology for the evaluation of power efficient data-reuse decisions, which aim at the reduction of the total power consumption for embedded processor implementing multimedia applications, has been presented. The transformations achieve power reduction by moving background memory accesses to smaller foreground memories. Analytical expressions for the number of accesses to each memory layer and the corresponding executed instructions are derived, allowing a fast exploration of the design space. Experimental results prove that for data dominated applications, the optimal solution in terms of power, performance and area can result by the right combination of high-level decisions for the adaptation of a certain data memory architecture and the application of high-level data-reuse transformations.

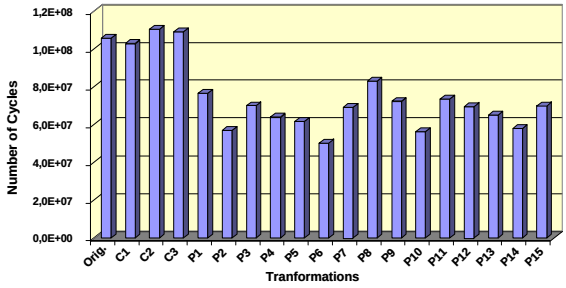

Fig. 5: Code Performance for all Transformations

REFERENCES

[1] F. Cathoor, 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. Cathoor, "Formalized Methodology for Data Reuse Exploration for Low-Power Hierarchical Memory Mappings", *IEEE Trans. on VLSI Systems*, vol. 6, No. 4, pp. 529-537, Dec. 1998.

[4] N. D. Zervas, K. Masselos and 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), October 1999, pp. 71-80.

[5] V. Bhaskaran, K. Konstantinides, *Image and Video Compression Standards: Algorithms and Architectures*, 2nd ed., Kluwer Academic Publishers.

[6] D. Soudris et al, "Data-Reuse and Parallel Embedded Architectures for Low-Power, Real-Time Multimedia Applications", Proc. *of 10 Int. Workshop on Power and Timing Modeling, Optimization and Simulation* (PATMOS'2000), September 2000.

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

[8] S. Kougia, A. Chatzigeorgiou, S. Nikolaidis, "Analytical Exploration of Power Efficient Data-Reuse Transformations on Multimedia Applications", *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing* (ICASSP'2001), May 2001.