VLSI Design, 2002 Vol. 14 (3), pp. 273–286

Taylor & Francis
Taylor & Francis Group

Energy Minimization Under Area and Performance Constraints for Multimedia Applications Realized on Embedded Cores

N. D. ZERVAS*, K. MASSELOS, Y. A. KARAYIANNIS and C. E. GOUTIS

VLSI Design Laboratory, Department of Electrical and Computer Engineering, University of Patras, Rio 26500, Greece

(Received 21 August 2000; Revised 18 January 2001)

A systematic methodology for energy dissipation reduction of multimedia applications realized on architectures based on embedded cores and application specific data memory organization is proposed. Performance and area are explicitly taken into account. The proposed methodology includes two major steps: A high-level code transformation step that reorganizes the original description of the target application. The second major step includes the determination of the processor, memory and bus organization of the system and is briefly described. Experimental results from several real-life demonstrators prove the impact of the high level step of the proposed methodology.

Keywords: Low-power; Embedded systems; Programmable cores; Performance; Area

INTRODUCTION

Portability as well as packaging and cooling issues made power consumption an important design consideration [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 [3], however, 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 system power cost [5]. This statement is true for different architecture platforms [6–11].

Rapid advances in the area of programmable embedded processor cores made them an attractive solution for the realization of real time multimedia applications, due to the flexibility they offer but also for time-to-market reasons. Especially the combination of processor cores with an application specific memory and bus organization is very promising since it allows combination of the processor cores advantages described above, with the optimization freedom offered by the application specific memory and bus organization.

In the area of data transfer and storage exploration for power and area costs, the ATOMIUM [4] and the PHIDEO [12] methodologies have been proposed. Both these methodologies mainly target uni-processor (single thread of control) custom hardware architectures and cannot be applied to a programmable processor context in a straightforward manner. This is because extra issues like power consumed due to instruction storage and transfers, performance (in terms of number of cycles) and code size must be taken into account in such systems. No systematic methodologies currently exist for high-level data transfer and storage energy optimization for systems realized on programmable processors. Some initial results for programmable processors with predefined fixed memory hierarchy have been recently presented [13,14]. For the case of applications realized on systems consisting of embedded cores and application specific data memory and bus organization the application of ATOMIUM in such a context is only described in Ref. [15]. However, no systematic methodology for data transfer and storage optimization is proposed and only the power-performance trade-off is evaluated. Furthermore, issues such as the power consumption due to transfer and storage of instructions and the code size effect on total system's power have not been explored.

Our previous work [26] has shown that the power consumed on instruction storage and transfers can be heavily affected by the application of code transformations that targets the minimization of power consumed on

*Corresponding author. Tel.: +30-61-997324. Fax: +30-61-994798. E-mail: zervas@ee.upatras.gr

ISSN 1065-514X print/ISSN 1563-5171 online © 2002 Taylor & Francis Ltd
DOI: 10.1080/10655140290011078

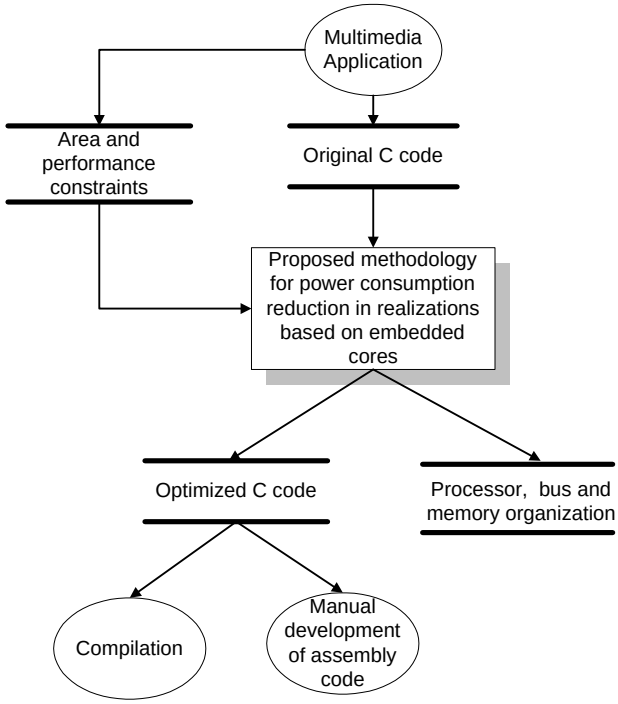

FIGURE 1 Context of application of the proposed methodology.

data storage and transfers. In the same work it was also pointed that the instruction memory-related power consumption of such systems can be comparable or even greater than the power consumed due to data storage and transfers. These observations lead to the following conclusions: First, the cost function that should drive a methodology for the application of high-level power-optimizing code transformations must take into account the power consumed on storage and transfers of both instructions and data. Second, in order for such a methodology to be power efficient, it must also include transformations that aim at optimizing the power consumed by instruction accesses and transfers. Clearly, these are not considered as the main contribution of this work, but they are distinguishing points between our approach and the approaches of Refs. [13–15], as far as the application of high-level code transformations is concerned.

The aim of the proposed research, part of which is described in this paper, is the development of a systematic methodology for the reduction of the power consumption related to storage and transfers of both instructions and data in realizations of multimedia applications on systems including embedded cores and applications specific data memory and bus organization. This is shown in Fig. 1. Inputs to the proposed methodology are an original high-level description of the target application and a set of area and performance constraints. The proposed methodology produces an optimized description of the application and a detailed processor, memory and bus organization. The optimized code produced can be either passed to the target processor compiler or can be used as input for the manual development of assembly code. This paper focuses on the high-level part of the proposed methodology and specifically on a systematic approach for the application of a number of high-level code transformations. There are two main categories of transformations included: The first category aims at optimizing in a high-level the power consumed in data memory accesses and data transfers, while the second category aims at optimizing the power consumed due to storage and transfers of instructions.

The rest of the paper is organized as follows: In the second section, the target architecture is described. The energy model and the cost functions are discussed in the third section. The proposed methodology is presented in the fourth section. The experimental results from several demonstrators are presented in the fifth section while conclusions are offered in the sixth section.

TARGET ARCHITECTURE

The general view of the target architecture is shown in Fig. 2. The main points of the target architecture are discussed below:

Processing units: Processing is performed on a number of instruction-set processor cores. The number of cores allocated is determined based on the performance requirements of the target application. The registers present in the cores can be exploited by the proposed methodology. Each processor core present is programmed by its private program memory.

Program memory and related bus organization: The application code is stored in on-chip program memories. On-chip storage of the application code is the case in most embedded systems since this reduces the total system's power consumption and improves performance as well. The size of the program memory is fixed in most cases. However, there are cases in which the size of the program memory is assumed to be application dependent, i.e. determined by the application's code size. Each processor core present is directly coupled through a dedicated bus to a program memory that holds the code executed by the specific core.

Data memory organization: Application data are stored in an application-specific data memory hierarchy. Most levels lie on-chip while off-chip levels are also possible. Storage of the major part of the data on-chip favors significantly power consumption reduction. Each level of the data memory hierarchy may be divided to a number of different blocks (for power and performance reasons). Data organization is fully compile-time determined.

Bus organization: The processor cores communicate with the different blocks of the data memory hierarchy over a number of global data, address and control buses. For simplification, the buses are merged to one in Fig. 2. The address space of each core is divided to parts assigned

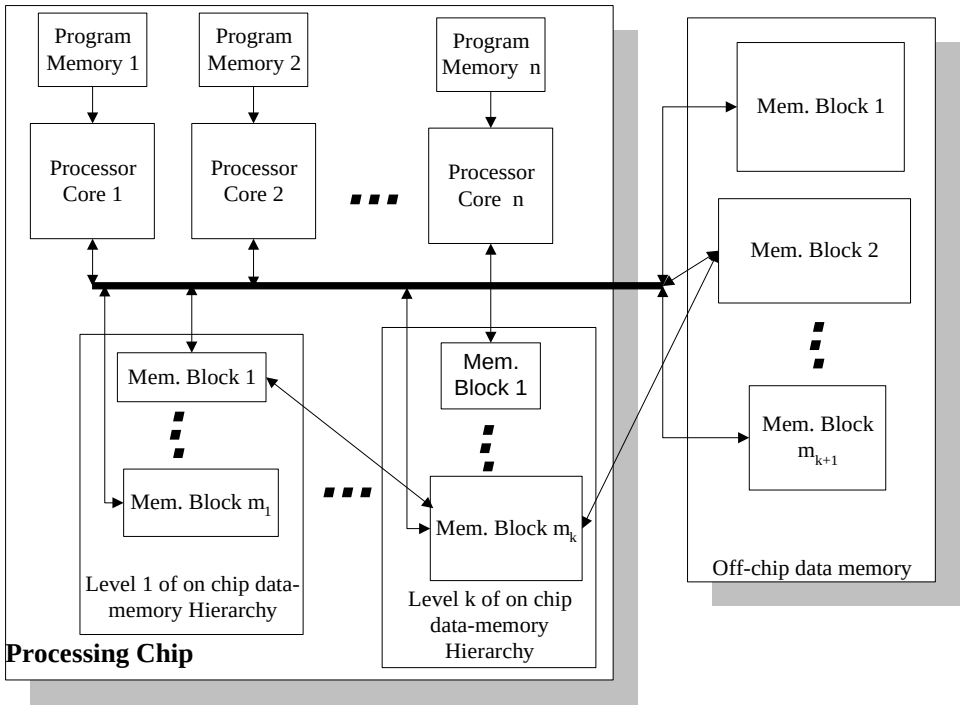

FIGURE 2 General view of the target architecture.

to the memory blocks used by the specific processor. The fact that each memory block has a single address range must be taken into consideration during the assignment of the address spaces to memory blocks when multiple cores use the same memory. All memory blocks are directly connected to the data bus. Transfers between memory blocks belonging to different levels of the hierarchy are performed through dedicated buses connecting the corresponding levels. Data transfers from memory blocks to the cores but also between blocks are performed in a fully deterministic way (no hardware control mechanism is present as in the case of instruction set processor caches) controlled by the cores.

ENERGY MODELS—COST FUNCTIONS

The main cost function that is the target for optimization of the proposed methodology is the energy dissipation related to data and instructions storage and transfers. For multimedia applications, this energy dissipation component dominates upon the total energy budget of the system. This cost is heavily related to the (external) bus traffic, which is also a crucial performance measure of the global system. In the proposed context only the energy consumption in background memories and in interconnect buses is taken into account. The energy consumed in the functional units, in glue logic and in foreground storage (registers—register files) is much smaller [1] and thus it is neglected.

As far as the storage related energy dissipation is concerned the energy model used for the on-chip memories depends upon the number of accesses, memory size, number of read/write ports and the number of bits that can be accessed in every access. The model is valid for any type of memory (SRAM, DRAM, etc.). The model is linear with respect to the number of accesses while the dependence on the memory size is determined by a sub-linear polynomial function f. This function is completely dependent on technology and the specific vendor used. Thus the power consumed by an on-chip memory is given by the following equation:

$$E = (\#\,\text{accesses}) \times f(\text{size}) \tag{1}$$

The function f that has been used to produce the energy figures that will be presented in the rest of the paper as described in Ref. [2]. For the estimation of the energy consumption of the off-chip memories the low-power 1 MB SRAM presented in Ref. [3] is assumed as in Ref. [4] since no other figures for power consumption of off-chip memories are currently available by the vendors. This assumption leads to an energy dissipation of 2.6 nJ per off-chip memory accesses. In this way, it is assumed that the off-chip energy dissipation depends on the number of accesses only and not on the memory size. For both on-chip and off-chip memories it is assumed that the memory is in power-down mode when not accessed since this is the case for the state-of-the-art memories currently available.

In terms of interconnect energy the ever increasing technology scaling makes the energy dissipated in the on-

chip buses more important contributor to the total system's energy dissipation than the energy dissipated in the functional units. The energy dissipation per line of on-chip interconnect during a transfer is given by Eq. (2) and depends on the energy dissipated in the related driver and the energy dissipated in the wire capacitance driven.

$$E_{\text{line\_on\_chip\_interconnect}} = E_{\text{driver}} + E_{\text{wire\_capacitance}} \quad (2)$$

The energy dissipated per line of off-chip interconnect during a data transfer to/from an off-chip memory is given by Eq. (3). It is approximated as the sum of the energy dissipated in the I/O pins involved (in both the memory and the processing chip), the energy dissipated in the corresponding I/O drivers (pads) and the energy dissipation in the wire capacitance driven.

$$E_{\text{line\_off\_chip\_interconnect}} = E_{\text{I/O\_pins}} + E_{\text{I/O\_drivers}} + E_{\text{wire\_capacitance}} \quad (3)$$

For the results presented in this paper the energy dissipated per line of on-chip interconnect has been estimated assuming a total load of 6 pF (including all the components described above). The energy dissipated in one line of an off-chip bus has been estimated assuming a total load of 30 pF (including all the related components). Both the load values have been obtained using the data presented in Ref. [4].

As already mentioned the proposed methodology aims at reducing the energy dissipation due to data and instruction storage and transfers. Assuming a data memory hierarchy with M blocks on-chip, N blocks off-chip, K on-chip bus lines and L off-chip bus lines the cost function that is optimized by the proposed methodology is given by Eq. (4).

$$E_{\text{data\_related}} = \sum_{m=1}^{M} E_{\text{on\_chip\_mem\_}m} + \sum_{n=1}^{N} E_{\text{off\_chip\_mem\_}n} + \sum_{k=1}^{K} E_{\text{on\_chip\_bus\_line\_}k} + \sum_{l=1}^{L} E_{\text{off\_chip\_bus\_line\_}l}. \quad (4)$$

Energy dissipation due to accesses to the program memory and due to the corresponding transfers is also a major contributor (except from the power consumption due to data storage and transfers) to the global system's energy dissipation. The energy dissipation due to the accesses to the program memory can be evaluated in the same way as described above. The number of executed instructions (directly proportional to the number of execution cycles) is used to approximate the number of access to the program memory. Thus the energy dissipation in the program memory is given by the following equation:

$$E_{\text{program\_memory}} = \text{executed instructions} \times f(\text{program\_memory\_size}) \quad (5)$$

The size of the program memory can be either fixed (by the processor's core used) or can be adapted to the target application offering more freedom (design parameter). In such a case the program memory should be large enough to store the complete application code. It is assumed that in all cases the application code can fit in on chip program memory. The program related energy dissipation is given by Eq. (6).

$$E_{\text{program\_related}} = E_{\text{program\_memory}} + E_{\text{instruction\_bus}} \quad (6)$$

The (major part of) global energy dissipation of the system is given by the following equation:

$$E_{\text{global}} = E_{\text{data\_related}} + E_{\text{program\_related}} \quad (7)$$

PROPOSED METHODOLOGY

In this section, the complete methodology for the reduction of the power consumption due to data and instructions storage and transfers under performance and area constraints is briefly described. The high level part of the methodology is described in detail.

Global Approach

The proposed methodology is presented in Fig. 3. The inputs to the methodology are an original high-level description (e.g. C/C++ code) of the target application and a set of area and performance constraints. A flow in which the processor's compiler will be used in the lower level is assumed. The methodology consists of two major steps: (a) A high level step that includes the application of code transformations and (b) A lower step in which detailed decisions on processor, data memory and bus organization are made.

In the first step (that is described in detail in the following sub-section) power optimizing code transformations are applied. A transformed description of the application is produced at the end of this step. The performance of this description is evaluated on the cores selected for the realization. Taking into consideration the performance constraints a number of cores that will be used for the realization is allocated. This ensures that performance constraints are met. In the next step, the area of the implementation is estimated using the procedure described in Fig. 4. If area constraints are not met then the application's description must be changed again by removing certain transformations. Transformations that

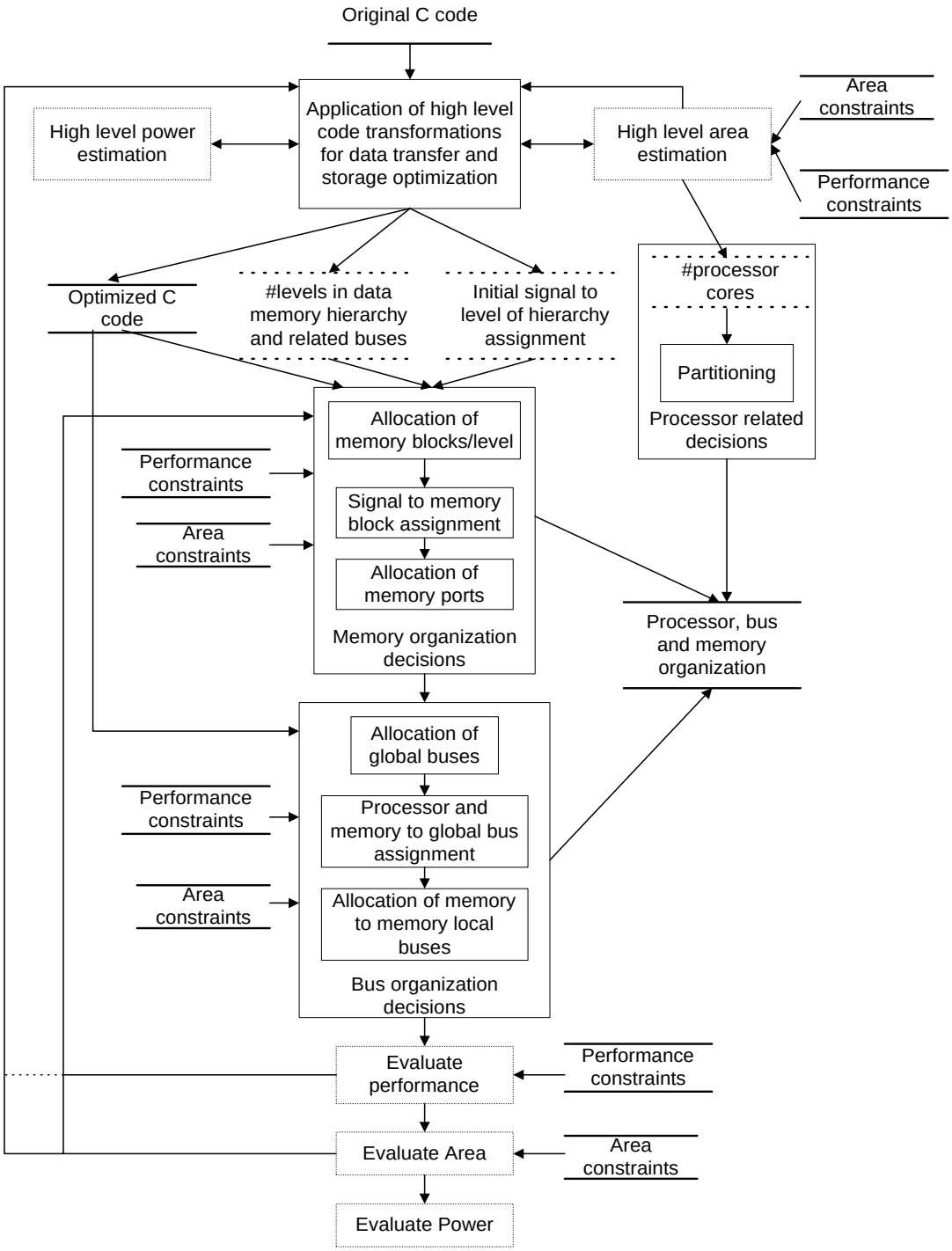

FIGURE 3 Proposed global methodology.

are candidates to be removed are:

i) Transformations that introduce performance penalties (in comparison to the original description). Such transformations negatively affect area since more cores and related buses will be required to execute code within the performance constraints.
ii) Transformations that introduce large number of new data signals with increased size. Such transformations increase the data memory area and the area of the related buses.
iii) Transformations that increase code size thus increasing the program storage requirements.

This procedure is iterated until both performance and area constraints are met. If both types of constraints cannot be satisfied then priority is given to the performance constraints (overriding issue in the target domain) while

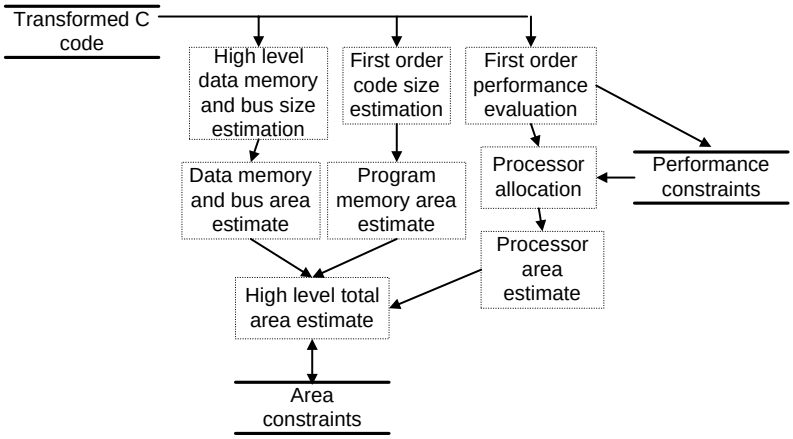

FIGURE 4 High level area estimation procedure.

area must be kept as small as possible. The output of the first step includes an optimized description of the application that will be compiled in the end as well as a set of constraints in relation to the number of levels of the data memory hierarchy and some initial signal-to-level of the hierarchy assignment decisions. As far as the processor organization is concerned the high level step determines the number of processor cores that will be used for the realization of the application.

The second step of the proposed methodology includes three main sub-steps:

(a) Determination of processor organization. This step mainly includes partitioning of the final code to the number of processors allocated.
(b) Determination of memory organization. Includes allocation of memory blocks/level of the data memory hierarchy, assignment of signals to memory blocks and allocation of ports to the memory blocks.
(c) Determination of the bus organization. The number of global and local buses is determined in this step and processors and memories are assigned to the buses.

All these steps are power oriented and take performance and area constraints into account. After the application of the second step performance and area constraints are evaluated and if not met local (over the second step) and global (over both steps) feedback loops can be performed. The output of the second step is a detailed processor, memory and bus organization of the system.

The Memory Management Graph

To better explain the energy minimization procedure and the effect of the power optimizing code transformations on power the concept of the Memory Management Graph (MMG) [27] is used. The memory management graph, $G_{\mathrm{MM}}\{V, E\}$ is a directed graph, whose vertex set $V\{v_i; i = 1, 2\ldots, n_{\mathrm{tasks}}\}$ is in one-to-one correspondence with the set of tasks. The directed edge set $E\{(v_i, v_j);\ i, j = 1, 2, \ldots, n_{\mathrm{tasks}}\}$ is in correspondence with data arrays passed from the one task to another.

The edges of the graph are annotated with two costs:

(a) The size of the data arrays $S_{ij}(i, j = 1, 2, \ldots, N)$ passed from the source task (i) to the sink task (j) of the edge.
(b) The number of accesses $A_{ij}(i, j = 1, 2, \ldots, N)$ to the arrays implied by the edges.

It must be noted that the edges connecting primary inputs (PIs) or outputs (POs) to tasks of the graph, correspond to input or output arrays of the algorithm. The corresponding edges are annotated with the same information as explained above. An edge connecting the k_{th} primary input to the task i is denoted as (PI_k, i). In the same way an edge connecting the task j to the lth primary output is denoted as (j, PO_l).

Each vertex v_i is annotated with two costs:

(a) The number of instructions $I_i(i = 1, 2, \ldots, N)$ executed by task
(b) The corresponding code size. $C_i(i = 1, 2, \ldots, N)$

The graph in its general form is hierarchical, i.e. each task represents a sub-graph divided to smaller subtasks and having the same characteristics. Usually the sizes of the data arrays between the subtasks inside a task are smaller in comparison to those of the arrays present between main tasks. A general view of the Memory Management Graph is shown in Fig. 5.

Using the Memory Management Graph concept the problem of power minimization under performance and area constraints in the proposed context can be expressed

as described by the following equation:

$$\text{Minimize} \sum_{i=1}^{N} \left(I_i \times f(C_i) + \sum_{\text{for all edges } (i,j)} (A_{i,j} \times f(S_{i,j})) \right) \quad (8)$$

within performance and area constraints.

Construction of the memory management graph of the target application allows easier identification of the main power consumption bottlenecks. Power optimizing code transformations can then focus at those points.

Proposed Methodology for the Application of High-level Code Transformations

The high level step of the proposed methodology that includes the application of a number of (mainly power oriented) code transformations is discussed in this section. The methodology proposes a detailed order in which several data transfer and storage optimizing transformations included in ATOMIUM [4] methodology are applied to an initial specification of the target application. Additionally, the proposed methodology includes transformations for the optimization of the energy dissipated in instruction storage and transfer. The new methodology extensions are meant in particular for the context of systems based on programmable cores and application specific memory and bus organization where performance, power and area constraints are crucial.

The main categories of data storage and transfer power optimizing code transformations used by the proposed methodology are:

(1) Data flow transformations [17]: Remove dataflow bottlenecks and reduce the number of accesses to background memories (especially focusing on inherently redundant accesses which are not found by conventional code clean-up approaches).
(2) Loop/control flow transformations [18]: 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 [19]: Introduce array signals where copies from larger signals that exhibit data reuse are stored.

In-place mapping [20]: Consists of two steps which heavily depend on the execution order of the array accesses: (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. At the highest levels in the script, the execution order is not yet fixed at all so only a high-level in-place estimate (lower and upper bounds) can be derived.

The full ATOMIUM methodology [4] also contains additional steps after the global transformations discussed above. Those steps mainly handle memory organization decisions. The transformations described above are in principle applied by ATOMIUM in the order presented.

In our approach the transformations that aim the reduction of the energy dissipated due to data accesses and transfers are grouped in three categories depending on the way they operate on the array signals of the initial system specification.

(a) *Access removing transformations*: Reduce the number of accesses to the different array signals present in the algorithm's description. From the transformations belonging to this category only data flow transformations will be considered in this paper. The effect of the access removing transformations can be described by the following equation using the memory management graph notation.

$$\sum_{\text{for all edges } (i,j)} A_{i,j}\text{initial} > \sum_{\text{for all edges } (i,j)} A_{i,j}\text{transformed} \qquad i,j : \text{PI or PO or task} \quad (9)$$

(b) *Size reduction transformations*: Reduce the size of array signals that are present in the initial specification allowing storage in lower levels of the memory hierarchy that lie closer to the processing units and thus reducing power consumption. In place transformations, always belong to this category. Loop transformations such as loop merging may operate in this way as well. The effect of the access removing transformations can be described by the following equation using the memory management graph notation. The effect of the size reduction transformations can be described by the following equation using the memory management graph notation.

$$\sum_{\text{for all edges } (i,j)} S_{i,j}\text{initial} > \sum_{\text{for all edges } (i,j)} S_{i,j}\text{transformed} \qquad i,j : \text{PI or PO or task} \quad (10)$$

(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. Data re-use transformations are always transformations of this kind but also loop transformations may operate in this way. The effect of the access moving transformations can be described by the following equations using the memory management graph notation.

$$\sum_{\text{for all edges } (i,j) \text{ initially present}} A_{i,j}\text{initial} > \sum_{\text{for all edges } (i,j) \text{ initially present}} A_{i,j}\text{transformed}$$

$$i,j : \text{PI or PO or task}$$

$$\sum_{\text{for all edges } (i,j) \text{ introduced}} S_{i,j} \lll \sum_{\text{for all edges } (i,j) \text{ initially present}} S_{i,j} \quad (12)$$

Additionally our approach includes a group of transformations that targets the reduction of the energy

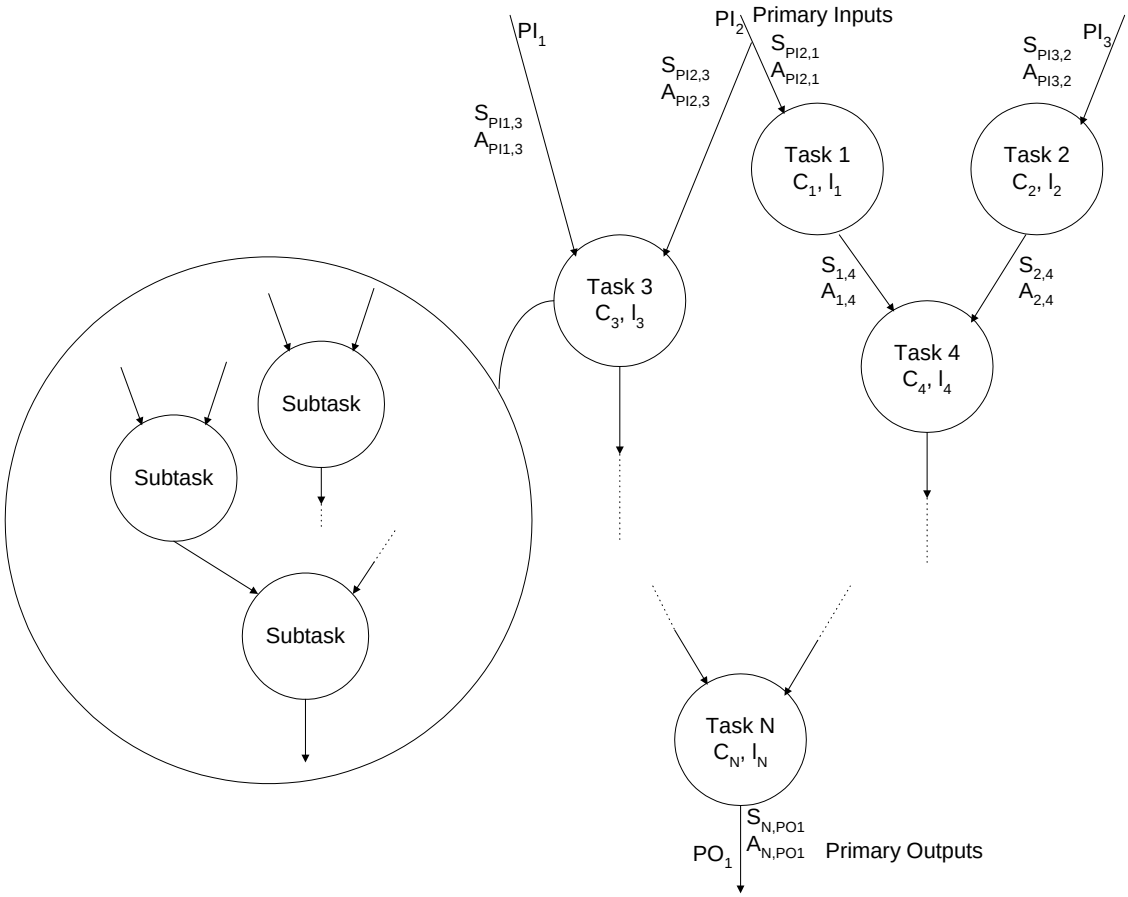

FIGURE 5 General view of the memory management graph.

dissipated due to instruction accesses and transfers. These transformations do not affect the way data are accessed or stored.

(d) *Instructions reduction transformations*: Mainly target the reduction of the number of executed instructions but may also affect the code size. Number of instruction is given higher priority than code size, since energy is linearly proportional to instructions and sub-linearly proportional to code size, according to the energy model described in the "Energy models—cost functions" section. Common sub-expression elimination, constant expression elimination, loop invariant code motion and algebraic transformations belong to this category. The effect of the instruction reduction transformations can be described by the following equations using the memory graph notation:

$$\sum_{i=1}^{n_{\text{taskw}}} I_i \times f(C_i)\_\text{initial} > \sum_{i=1}^{n_{\text{taskw}}} I_i \times f(C_i)\_\text{transformed} \quad (13)$$

The claim here is that these categories should be applied in the specific order above in order to better control the energy-area trade-off. Transformations that enable the application of all the above-described energy optimizing transformations may also be applied. Data flow transformations that remove data dependencies and loop transformations that improve locality and regularity are the most important enabling transformations.

Another important issue in our approach is that the array signals in an algorithm description are classified based on their functionality to the following categories [13,27]:

(a) *Array signals with long lifetimes*: These signals are "alive" either throughout or for long periods in algorithm's code. Typical examples are the input and output signals of an algorithm, coefficient arrays and look-up tables.
(b) *Intermediate array signals*: These signals are produced as intermediate results of the algorithm and usually have shorter lifetimes in comparison with the signals of the first category.

The detailed order of application that is the main contribution of this paper is described in Fig. 6. In the first step, data flow transformations are applied to reduce accesses to the array signals of the description (see Ref. [17]). Then the size reduction step is applied. The long lifetime signals are mainly optimized by applying in-place mapping mainly intra-signal but also inter-signal

especially between input and output signals. Intermediate signals can be mainly optimized by applying loop and in-place mapping [20]. At the same time, also enabling transformations are applied. In the next step, access-moving transformations are applied targeting mainly the long lifetime signals. Since the application of the access moving transformations usually introduces new arrays (increasing the memory requirements) an energy-area trade-off must be explored in this point since the data memory hierarchy is not fully predefined. Moreover, in-place mapping needs to be applied for the signals introduced by the access moving transformations.

The transformations targeting the data-related energy dissipation (access removing, size reduction and access moving) may affect performance in a programmable processor context, since they may increase the computational complexity of address and control expressions. For this reason the instruction reduction transformations are applied in the last step since they mainly aim at optimizing the computational complexity of all (address, control and arithmetic) expressions present in the application's description. Since both energy related to program memory and performance are proportional to the number of executed instructions, the instructions reduction transformations can be viewed as energy and performance optimizing transformations. The instructions reduction transformations almost always increase the code size. Thus because here it is assumed that the program memory size can be adapted to the size of a specific code, an energy-area trade off exist at this point and must be explored.

Clearly, the proposed order of application of the code transformations is different than the order proposed by ATOMIUM. Main points of differences are: (a) the four types of transformations (data-flow, loop, data re-use and in-place) are mixed in the proposed methodology while ATOMIUM applies them independently and in distinct steps, (b) detailed in-place mapping even in the early steps is applied by the proposed methodology while detailed in-place in ATOMIUM is applied as a last step (even after the final memory organization), (c) loop transformations are included in both the main steps of the proposed methodology while in ATOMIUM all loop transformations are applied together whether they introduce new signals or not. The main idea behind the proposed order of application of the high-level code transformations is that the application of the code transformations that do not introduce new signals in a different step than those introducing new signals allows better control of the crucial power-area trade-off. Size reduction transformations operate on signals existing in the initial description that will be in the finally transformed description as well and try to reduce their size. After the application of the size reduction step the minimum storage requirements of the application are determined. Then the access moving transformations that introduce new signals are applied and thus the area of the finally transformed description can be better controlled.

EXPERIMENTAL RESULTS

In this section, experimental results from the application of the high level part of the proposed methodology to a number of real life demonstrators are presented. The demonstrators include the full search motion estimation kernel [21], the QSDPCM algorithm for video compression [22], the SGLDM texture analysis algorithm [23] and a voice coder application [24]. For the realization of the demonstrators the ARM 7 processor core has been used. For the case of the QSDPCM application realization on 14 cores was assumed. The best in terms of power hybrid (task-data level) partitioning of the QSDPCM described in Ref. [16] was assumed. For the rest applications single core realizations were assumed.

As far as the data memory organization is concerned the number of levels is assumed to be the minimum required to make all the access moving transformations meaningful in terms of power, i.e. to ensure that no data transfers beginning and ending in the same level of the data hierarchy exist in the transformed code (for example if the maximum number of new levels introduced by all the access moving transformations in one application is three then the data memory hierarchy should at least include three levels). It is also assumed that the levels of the data memory hierarchy are not divided into blocks (centralized architecture). These assumptions will be used for the high-level evaluation of the data storage and transfer related power consumption and data memory area.

For the program memory it is assumed that its size is not fixed, i.e. it can be adapted to the size of a specific code and thus it is an important design parameter. No other implementation specific assumptions (number of memory ports, bus organization) are made. An important point is that the ARM debugger that has been used for the simulations does not evaluate cache related effects, i.e. it assumes that all the accesses to the array signals present in the application code have the latency required to access the (off-chip) main memory. The proposed methodology moves certain signals in levels of the data memory hierarchy lying on-chip and closer to the cores where accesses have shorter latencies than those to the main memory. Thus the performance results presented in this paper are underestimates of the real performance that can be achieved if the proposed memory organization is adopted.

The effect of the proposed methodology on the data transfer and storage related power consumption (main cost targeted for optimization by the proposed methodology) and on memory area are presented in Figs. 7 and 8. The power models described in the "Energy models—cost functions" section have been used. For the area estimation the model of Mulder [25] has been used. The results are relative with respect to each application independently. Clearly the application of the proposed methodology reduces significantly the data related power consumption. The smaller savings (46%) are achieved in the case of the full search motion estimation where the small code restricts

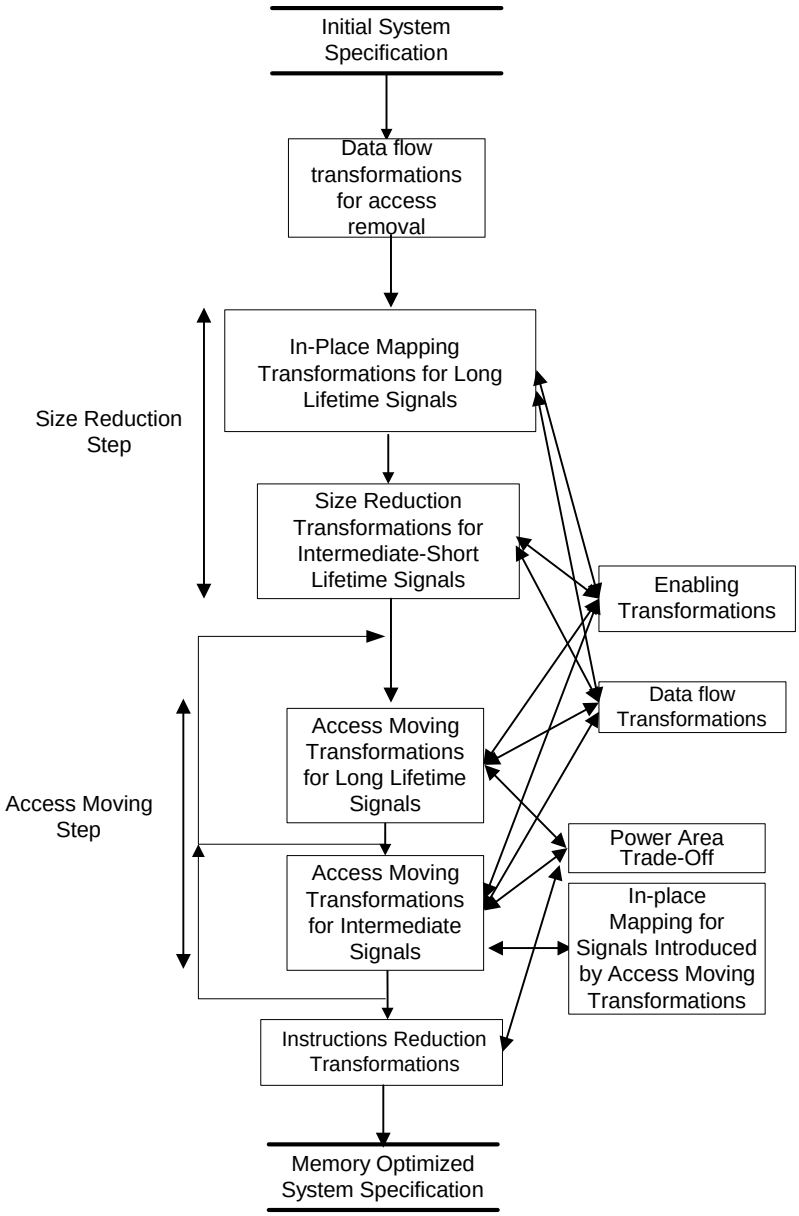

FIGURE 6 Proposed systematic methodology for the application of high-level code transformations.

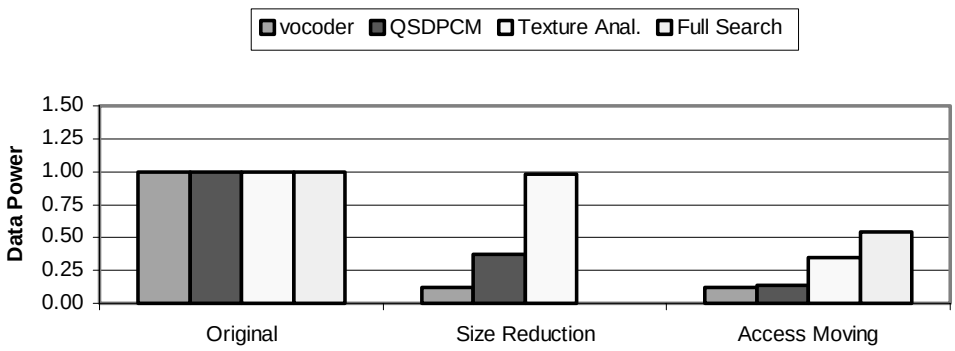

FIGURE 7 Data transfer and storage related power results.

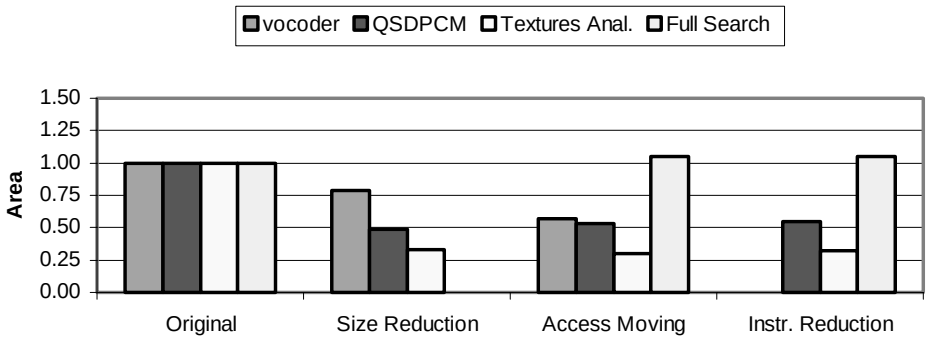

FIGURE 8 Memory area results.

the opportunities for optimization (this is also the reason why only access moving transformations are applied). For the rest applications larger savings are achieved (voice coder 89%, QSDPCM 86%, texture analysis 65%). For the voice coder and the QSDPCM cases the size reduction step has a larger impact while for the texture analysis application the access-moving step contributes more to the data related power consumption reduction.

In terms of area, the effect of the proposed methodology is positive when both the size reduction and the access moving steps are applied. This is very significant and proves that the proposed order of application of the code transformations allows indeed better control of the area-power trade-off in a high level. For the case of the full search motion estimation where only access moving transformations are applied small area penalties are introduced.

The effect of the proposed methodology on execution time (in terms of number of cycles) is shown in Fig. 9. It is clear that the combined application of the data-related energy optimizing transformations with the instructions reduction transformations leads always to better performance in comparison to reference descriptions. In all cases, except from the voice coder application, even the application of the data-related energy optimizing transformations on a stand-alone basis improves performance. This is because a large number of accesses to the data memories is moved to the register file of the ARM cores after the application of the power oriented transformations.

Possible reasons behind the performance penalties for the voice coder application are the increase of the address expressions complexity (not removed since no instructions reduction transformations are applied) and the increased number of scalar variables introduced (mainly by data reuse transformations) that cannot be all stored in the register file of the ARM core. There are cases where either the instructions reduction transformations are not applied (as for the voice coder case) or they are applied and cannot remove the performance penalties introduced by the data-related energy optimizing transformations. In such cases more processing power is required to meet the performance constraints (i.e. larger number of cores, higher performance cores or accelerator data paths). In such cases area and power overheads are possible. Clearly, the application of power oriented transformations should not introduce performance penalties.

The effect of the proposed methodology on code size is shown in Fig. 10. Since the size of the program memory is not fixed the code size becomes a design parameter offering another dimension for optimization in the design space (since it directly affects program related power and total system area). In two cases the application of the proposed methodology improves the code size (texture analysis and voice coder) while in the rest cases it introduces code size penalties. The application of the

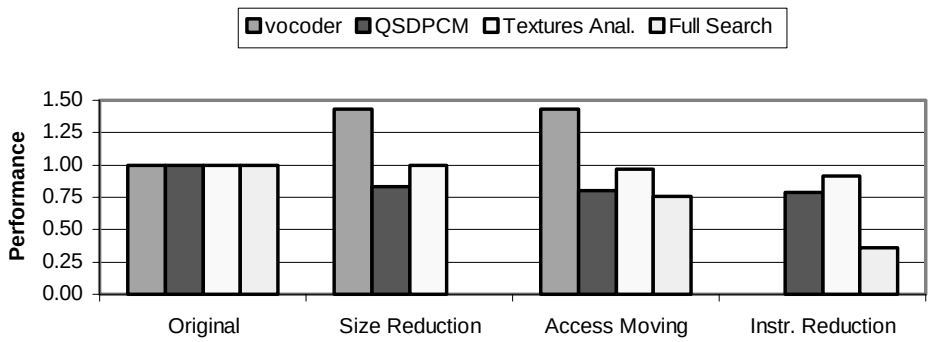

FIGURE 9 Effect of the proposed methodology on execution time.

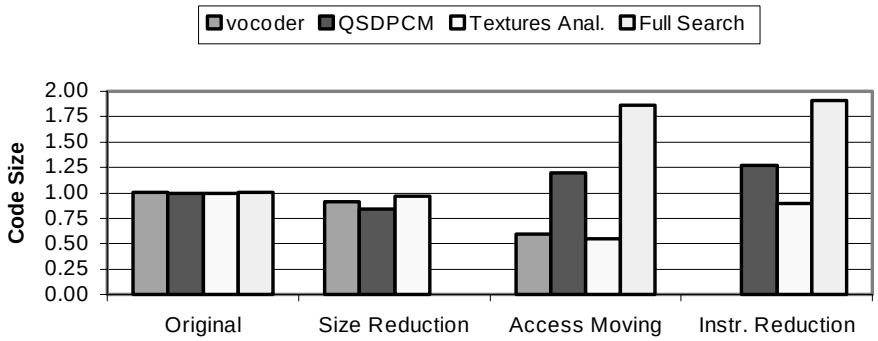

FIGURE 10 Effect of the proposed methodology on code size.

instructions reduction transformations increases the code size in all cases. Extensive application of access moving transformations and specifically data re-use transformations normally increases the code size (as for the QSDPCM and the full search motion estimation cases) since extra loops are introduced. Introduction of data re-use levels including only scalar variables does not introduce code size penalties (since no extra loops are introduced) and may even reduce the size of the code (as for the voice coder and the texture analysis cases). Size reduction transformations do not introduce code size penalties and may even reduce it since in certain cases they reduce the number of loops.

The effect of the proposed methodology on the energy due to instruction storage and transfers is presented in Fig. 11. These energy results mainly follow the performance results described above but are also affected by the code size results. This mainly due to the fact that the number of instruction transfers and accesses to the program memory is given by the number of executed instructions (according to the model of the "Energy models—cost functions" section). Furthermore the size of the program memory that also affects the program related power consumption depends on the code size.

The total system power consumption results (including both data related and program related contributions) are presented in Fig. 12. Clearly, total power follows the program related power. Data power appears to have minor contribution to the total system's power. Thus the proposed methodology should be applied in such a way that both performance (directly related to the number of executed instructions) and code size penalties are avoided since (even small) such penalties may offset the data related energy savings leading to total system power penalties.

CONCLUSIONS

A systematic methodology for the reduction of the data and instruction memory and bus energy dissipation in realizations of multimedia applications on systems including embedded cores and application specific data memory and bus organization has been proposed. The methodology includes two main steps. The high level step reorganizes the initial description of the target application through the application of a number of code transformations. The transformations aim at the minimization of the energy dissipated in data and instructions storage and transfers. The second step produces a detailed processor, memory and bus organization of the system. A detailed description of the order of application of the high-level power oriented code transformations has been also presented. Experimental results from different real-life applications illustrate that significant power consumption reduction can be achieved through the application of the

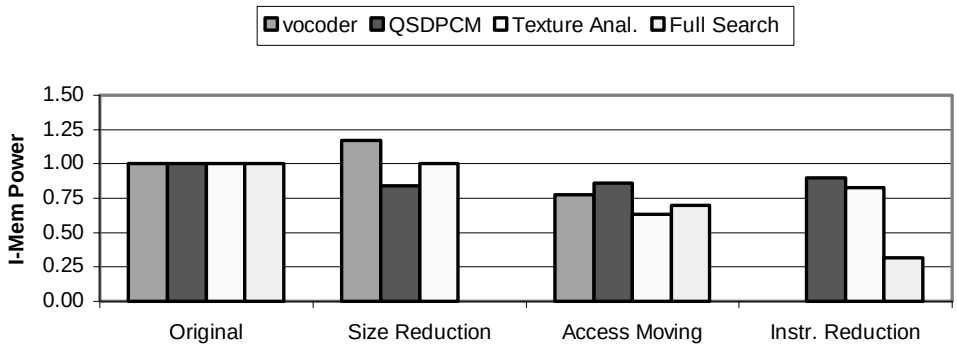

FIGURE 11 Effect of the proposed methodology on program memory related power.

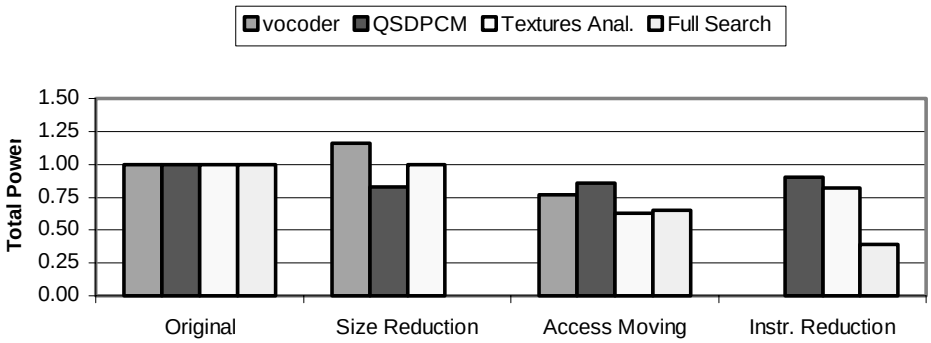

FIGURE 12 Total power results.

proposed methodology while meeting performance and area constraints.

References

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

[2] Landman, P., "Low power architectural design methodologies" Doctoral Dissertation, U.C. Berkeley.

[3] Seki, T., Itoh, E., Furukawa, C., Maeno, I., Ozawa, T., Sano, H. and Suzuki, N. (1993) "A 6-ns 1-Mb CMOS SRAM with latched sense amplifier", *IEEE J. Solid State Circuits* **28**(4), 478–483.

[4] Catthoor, F., Wuytack, S., De Greef, E., Balasa, F., Nachtergaele, L. and Vandecappelle, A. (1998) *Custom Memory Management Methodology—Exploration of Memory Organisation for Embedded Multimedia System Design* (Kluwer Academic Publishers, Boston), ISBN 0-7923-8288-9.

[5] Wuytack, S., Catthoor, F., Nachtergaele, L. and DeMan, H. (1996) "Power Exploration for Data Dominated Video Applications", *Proc. IEEE Int. Symp. Low Power Des., Monterey CA* **August**, 359–364.

[6] DeMan, H., Catthoor, F., Goosens, G., Vanhoof, J., Van Meerbergen, J., Note, S. and Huisken, J. (1990) "Architecture-driven synthesis techniques for VLSI implementation of DSP algorithms", *Proc. IEEE* **78**(2), 319–335, special issue on The future of Computer-Aided Design.

[7] Lippens, P., Van Meerbergen, J., Verhaegh, W. and Van Der Werf, A. (1993) "Allocation of multiport memories for hierarchical data streams", *Proc. IEEE Int. Conf. Comput.-Aided Des., Santa Clara CA* **November**.

[8] Danckaert, K., Catthoor, F. and DeMan, H. (1996) "System-level memory management for weakly parallel image processing", *EuroPar Conf., Lyon* **August**.

[9] Danckaert, K., Catthoor, F. and DeMan, H. (1997) "System-level memory optimization for hardware–software co-design", *Proc. IEEE Int. Workshop Hardware/Software Co-des., Braunschweig, Germany* **March**, 55–59.

[10] Meng, T.H., Gordon, B., Tsern, E. and Hung, A. (1995) "Portable video-on-demand in wireless communication", *Proc. IEEE* **83**(4), 659–680, special issue on Low Power Design.

[11] Tiwari, V., Malik, S. and Wolfe, A. (1994) "Power analysis of embedded software: a first step towards software power minimization", *IEEE Trans. VLSI Syst.* **2**(4), 437–445.

[12] Van Meerbergen, J., Lippens, P., Verhaegh, W. and Van Der Werf, A. (1995) "PHIDEO: high-level synthesis for high throughput applications", *J. VLSI Signal Process.* **9**(1/2), 89–104, Special issue on Design environments for DSP (Eds. I. Verbauwhede, J. Rabaey).

[13] Masselos, K., Danckaert, K., Catthoor, F., Zervas, N., Goutis, C.E. and De Man, H. (2000) "A specification refinement methodology for power efficient partitioning of data-dominated algorithms within performance constraints", *J. VLSI Signal Process. Systems* **26**(3), 291–317.

[14] Kulkarni, C., Catthoor, F. and DeMan, H. (1998) "Code transformations for low power caching in embedded multimedia processors", *Proc. Int. Parallel Process. Symp. (IPPS), Orlando, FL* **April**, 292–297.

[15] Nachtergaele, L., Vanhoof, B., Catthoor, F., Moolenaar, D. and De Man, H. (1998) "System-level power optimizations of video codecs on embedded cores: a systematic approach", *J. VLSI Signal Process. Syst.*.

[16] Masselos, K., Danckaert, K., Catthoor, F., Goutis, C.E. and De Man, H. (1999) "A methodology for power efficient partitioning of data-dominated algorithm specifications within performance constraints", *IEEE Int. Symp. Low-Power Electronics Design*, pp 270–272.

[17] Catthoor, F., Janssen, M., Nachtergaele, L., DeMan, H. "System-level data-flow transformations for power reduction in image and video processing". In: *Proc. of ICECS'96*, pp. 1025–1028.

[18] Kulkarni, D., Stumm, M. "Loop and data transformations: a tutorial", Technical Report CSRI-337, Computer Systems Research Institute, University of Toronto, June 1993.

[19] Diguet, J.P., Wuytack, S., Catthoor, F. and DeMan, H. (1997) "Hierarchy exploration in high-level memory management", *Proc. Int. Symp. Low Power Electronics Des., Monterey, CA* **August**, 18–20.

[20] De Greef, E., Catthoor, F. and DeMan, H. (1997) "Memory size reduction through storage order optimization for embedded parallel multimedia applications", *Int. Parallel Process. Symp. (IPPS) in Proc. Workshop Parallel Process. Multimedia* **April**, 84–98.

[21] Bhaskaran, V. and Konstantinides, K. (1994) *Image and Video Compression Standards* (Kluwer Academic Publishers, Dordrecht).

[22] Strobach, P. (1988) "A New Technique in Scene Adaptive Coding", *Proc. 4th Eur. Signal Processing Conf., EUSIPCO-88, Grenoble, Franc* (Elsevier, Amsterdam), pp 1141–1144.

[23] Conners, R.W. and Harlow, C.A. (1980) "A theoretical comparison of texture algorithms", *IEEE Trans. Pattern Analysis Machine Intelligence* **3**, 204–222.

[24] Rabiner, L.R. and Schafer, R.W. (1988) *Digital signal processing of speech signals* (Prentice Hall International, Englewood cliffs, NJ).

[25] Mulder, J.M., Quach, N.T. and Flynn, M.J. (1991) "An area model for on-chip0 memories and its application", *IEEE J. Solid-State Circuits* **SC-26**(1), 98–105.

[26] Zervas, N.D., Masselos, K., Koufopavlou, O.G. and Goutis, C.E. (1999) "Power exploration of multimedia applications realized on embedded cores", *Proc. Int. Symp. Circuits Syst.* **4**, 378–381.

[27] Masselos, K., Catthoor, F. "Performance efficient application of power optimizing code transformations for programmable multimedia processors", IMEC Internal Report, June 1999.

Authors' Biographies

Nikos D Zervas received a Diploma in Electrical and Computer Engineering from University of Patras, Greece in 1997. Since 1998, he has been working towards a PhD degree in the Department of Electrical and Computer Engineering of the same University. His research interests are in the area of high-level, power optimization

techniques and methodologies for multimedia and telecommunication applications. He has received an award from IEEE Computer Society in the context of Low-Power Design Contest of 2000 IEEE Computer Elements Mesa Workshop. Mr Zervas is a member of the IEEE and of the Technical Chamber of Greece.

Dr Konstantinos Masselos received a first degree and a PhD in Electrical Engineering from University of Patras, Greece in 1994 and 2000 respectively, and an MSc degree in VLSI Systems Engineering from University of Manchester Institute of Science and Technology (UMIST), United Kingdom in 1996. His research interests concern system-level low-power design methodologies for multimedia applications and includes data storage and transfer optimization, design of low complexity algorithms and efficient implementations for image and video processing, high-level synthesis and compilation techniques. He is associated as a visiting researcher to the System Exploration for Memory and Power (SEMP) group of the VLSI Systems Design Methodologies (VSDM) division of the Inter-university Micro Electronics Centre (IMEC) in Leuven, Belgium. He is a member of the IEEE and of the Technical Chamber of Greece.

Yorgos A Karayiannis was born in 1966, in Nafplio, Greece. He received a Diploma in Electrical Engineering from University of Patras, Greece in 1991. Since 1992 he has been working towards a PhD degree in the Department of Electrical and Computer Engineering of the same University. His PhD research concerns multiresolution image processing, wavelet transforms, texture analysis, image coding. He is a member of the Technical Chamber of Greece.

Dr Costas Goutis was a Research Assistant and Research Fellow in the Department of Electrical and Electronic Engineering at the University of Strathclyde, UK, from 1976–1979, and Lecturer in the Department of Electrical and Electronic Engineering at the University of Newcastle upon Tyne, UK, from 1979–1985. Since 1985, he has been Associate Professor and Full Professor in the Department of Electrical and Computer Engineering, University of Patras, Greece. His recent research interests focus on VLSI Circuit Design, Low Power VLSI Design, Systems Design, Analysis and Design of Systems for Signal Processing and Telecommunications. He has been awarded a large number of Research Contracts from ESPRIT, RACE and National Programs.