Power Exploration for Data Dominated Video Applications

Sven Wuytack, Francky Catthoor, Lode Nachtergaele, Hugo De Man

IMEC, Kapeldreef 75, B-3001 Leuven, Belgium

Abstract

In this paper we present our power exploration methodology for data dominated video applications. This formalized methodology is based on the observation that for this type of applications the power consumption is dominated by the memory architecture. Hence, the first exploration stage should be to come up with an optimized memory organisation. Other important observations are that the power consumption of the address generators is of the same magnitude as that of the data-paths and that the address generators are better optimized using specialized techniques.

1 Introduction

In video applications large amounts of data have to be handled in real-time. This usually results in high power consumption in both data transfers over communication channels and in data storage in large background memories. Therefore it is important to optimize the power consumption and required memory storage as much as possible. Our power exploration methodology is based on the observation that in this type of data-dominated applications, the system power consumption is dominated by the power consumed in the transfers and storage related to the main memory organisation [20]. So, the first stage in our power exploration methodology, is to come up with an optimized memory architecture. The derivation of an optimal memory architecture is done in a number of steps. The first step is the optimization of the control-flow to increase the regularity and locality in the algorithm. The next step is to decide on the memory hierarchy, to allocate the memories and to assign every signal to one of the allocated memories. Finally, there is an in-place mapping step that minimizes the size of each memory by calculating a storage scheme that allows to overwrite as much as possible data that is no longer alive.

These basic steps have been described by us before for an area oriented system exploration (see [14] and its refs). The systematic approach for the combination of power and area is however new. This script and its effects will be discussed further in section 5.

ISLPED 1996 Monterey CA USA
0-7803-3571-8/96/$5.00 © 1996

All of these steps are included in our High-level Memory Management methodology, which is partly supported in our ATOMIUM environment [14]. In this paper we will illustrate this methodology on a 2D motion estimation kernel.

In the experiments it has been assumed that the application works with frames of $W \times H$ pixels, processed at F frames/s. For example, in the 2D motion estimation kernel for the QCIF standard, which we use as a test-vehicle to illustrate the general methodology, this means 176×144 pixel frames in a video sequence of 30 frames/s. This results in an incoming pixel frequency of about 0.76 MHz.

This paper is organized as follows. Section 2 describes the related work. Section 3 introduces the test-vehicle on which the methodology is illustrated. Section 4 describes the power models we have used for the power estimation. Section 5 explains and illustrates the different steps in the power exploration experiments. Section 6 summarizes the conclusions of the paper.

2 Related Work

Most designs which have already been published on motion estimation in related MPEG video coders [5, 13, 18] are based on a systolic array type approach because of the relatively large frame sizes involved, leading to a large computational requirement on the DCT. However, in the video conferencing case, this is not needed. An example of this is discussed in [4]. As a result, it will be shown that a power and area optimized architecture is not so parallel (even partly multiplexed). Hence, also the multi-dimensional (M-D) signals should be stored in a more centralized way and not fully distributed over a huge amount of local registers. This storage organisation then becomes the bottle-neck[1].

As we have shown earlier, in principle much power can be gained by reducing the number of accesses to large frames or buffers [20]. Also other groups have made similar observations [12] for video applications. Up to now however no systematic approach has been published to target this important field. Indeed, most effort up to now has been spent, either on data-path oriented work (e.g. [2]), on control-dominated logic or on programmable processors (see [17] for a good overview).

[1] Note that the transfer between the required frame memories and the systolic array is also quite power hungry and usually not incorporated in the analysis in previous work

3 Test-vehicle

The motion estimation algorithm [11] is used in moving image compression algorithms. It allows to estimate the motion vector of small blocks of successive image frames. We will assume that the images are gray-scaled (in practice, for color images only the luminance is considered). The version we consider here is the kernel of what is commonly referred to as the "full-search full-pixel" implementation [9].

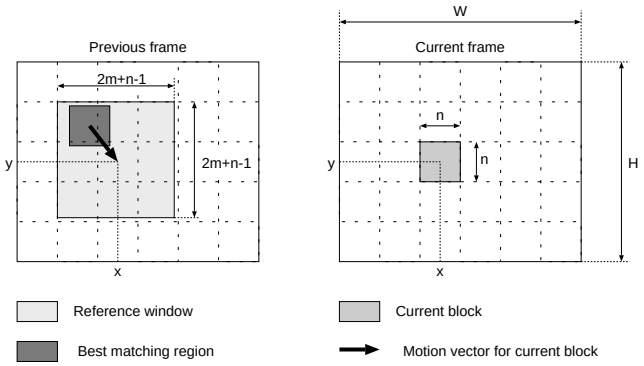

Figure 1: The motion estimation algorithm and its parameters

The algorithm is typically executed in 6 nested loops, except for the implicit frame loop. The choice of the nesting for these loops is partially open, and there is quite a lot of room for parallelisation and (loop) reordering. The basic operation at the inner loop consists of an accumulation of pixel differences, while the basic operation two levels higher in the loop hierarchy consists of the calculation of the new minimum and its location.

4 Power Models

The libraries used in the power models have been uniformly adapted for a 0.7μ CMOS technology, operating at 5V. If some figures where not available in that specific technology, they were scaled with experimental weights. Note that if a lower supply voltage can be allowed by the process technology, the appropriate scaling has to be taken into account. It will however be (realistically) assumed that V_{dd} is fixed in advance within the process constraints, and that it cannot be lowered any further by architectural considerations.

For the data-paths and address generation units (which were realized as custom data-paths), a standard cell technology was assumed where the cells were *NOT* adapted to low power operation. As a result, the power figures for these data-paths are very high compared to a macro-cell design with power-optimized custom cells. The power estimation itself however has been accurately done with the PowerMill tool of EPIC, based on gate-level circuits which have been obtained from behavioural specifications using IMEC's Clash/Dolphin custom data-path synthesis environment followed by the Synopsys RT-synthesis Design Compiler. The resulting VHDL standard cell net-list was supplied with reasonable input stimuli to measure average power.

For the memories two power models were used:

- For the large off-the-shelf units on a separate chip, SRAMs have been assumed because of the lower power consumption for the currently available RAMs [6]. For the SRAMs, we have used the model of a Fujitsu low-power memory [16] which is not yet commercially available. It leads to 0.26 W for a 1 Mbit SRAM operating at 100 MHz at 5V [2]. Because this low-power RAM is however internally partitioned, the power will not really be reduced by considering a smaller memory (as we will require further on). The power budget [16] clearly shows that about 50% of the power is consumed anyhow in the peripheral circuitry which is almost independent of the size. Moreover, we have no figures on the power consumed in the chip-to-chip communication so that will be ignored. This issue would more than compensate for the potential power gains by having smaller off-chip memories available. Hence, we will use a power budget of 0.26 W for 100 MHz operation in all off-chip RAMs further on. For lower access frequencies we will scale linearly, which is a reasonable assumption.

- For the embedded background RAMs, we have used the single-port memory power model developed at U.C.Berkeley by Paul Landman [10]. For the parameters in the model, we have scaled down the parameters to a 0.7 μ technology. Different values for read and write accesses were available and have been used. The input for this model are 3 essential parameters: number of bits, number of words and rate (frequency) at which the RAM is really accessed. The appropriate capacitance formula is then evaluated. Power is then equal to $F_{real} \cdot C \cdot V_{dd}^2$, where V_{dd} is assumed to be 5V. For dual-port memories we have multiplied this value by 2 [3].

Note that the real access rate F_{real} should be provided and not the maximum frequency F_{cl} at which the RAM can be accessed. The maximal rate is only needed to determine whether enough bandwidth is available for the investigated array signal access. This maximal frequency will be assumed to be 100 MHz [4]. It should be stressed however that results further on will show that in practice (after optimisation) a much lower access frequency is required so this maximum is never met and hence a single-port memory would suffice for all the allocated frame memories. If the background memory is not accessed, it will be in power-down mode[5].

[2] Currently, vendors do not supply much open information, so there are no better power consumption models available to us for off-chip memories.

[3] Values ranging from 1.6 to 2.3 were experimentally found for different types and parameters.

[4] Most commercial RAMs have a maximal operating frequency between 50 and 100 MHz

[5] This statement is true for any modern low-power RAM [6]

The real access F_{real} is the number of read or write accesses per frame multiplied by the maximal number of frames per s (which is 30 fr/s for many video conferencing applications). This is a very important consideration, because it means that the maximal clock frequency is not that crucial in memory related power optimizations.

A similar reasoning can apply however for the data-paths, if we carefully investigate the power formula. Also here the maximal clock frequency is not needed in most cases. Instead, the actual number of activations F_{real} should be applied, in contrast with common belief which is based on an oversimplification of the power model. During the cycles for which the data-path is idle, all power consumption can then be easily avoided by any power-down strategy. A simple way to achieve this is the cheap gated-clock approach for which several realizations exist (see e.g. [19]). In order to obtain a good power estimate, it is crucial however to obtain a good estimate of the average energy per activation by taking into account the accurately modeled weights between the occurrence of the different modes on the components. For instance, when a data-path can operate in two different modes, the relative occurrence and the order in which these modes are applied should be taken into account, especially to incorporate correlation effects. Once this is done, also here the maximal F_{cl} frequency is only needed afterwards to compute the minimal number of parallel data-paths of a certain type (given that V_{dd} is fixed initially).

5 Power Experiments

For the combined power and area exploration approach, we consider a target architecture as in Fig. 2.

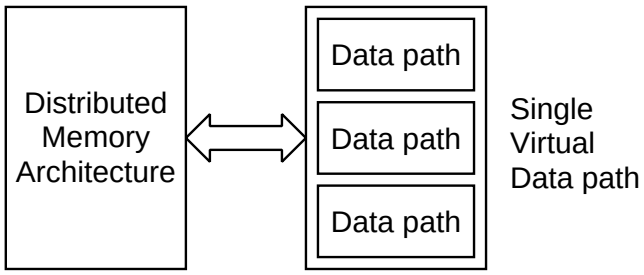

Figure 2: Architecture consisting of a distributed memory architecture that communicates with a data-path consisting of a number of parallel data-paths.

Depending on the parameters, a number of parallel data-paths are needed. In particular, for the 2D motion estimation this is $2m \times 2m \times W \times H \times F/F_{cl}$ processors for a given clock rate F_{cl}. However, this number is not really important for us because we consider an architecture in which the parallel data-paths with their local buffers are combined into one large data-path which communicates with the distributed frame memory. This is only allowed if the parallelism is not too large (as is the case for the motion estimator for the QCIF format). Otherwise, more systolic organisations, with memory architectures tuned to that approach, would lead to better results. In practice, we will assume that a maximal F_{cl} of 48.66 MHz is feasible for the on-chip components, which means that we need 4 parallel data-path processors.

We will now discuss a power optimized architecture exploration for the motion estimation, as illustration of the more general methodology.

5.1 Memory Organisation

For background memories, experiments have been performed to go from a non-optimized applicative description of the kernel in figure 1 to an optimized one for power, tuned to an optimized allocation and internal storage organisation. In the latter case, the accesses are heavily reduced. These accesses take up the majority of the power as we will see later.

Control-flow optimization. The first optimisation step in our methodology [14], is related to data-flow and loop transformations. For the 2D motion estimation, we will focus on the effect of loop transformations. It is clear that reordering of the loops in the kernel will affect the order of accesses and hence the regularity and locality of the frame accesses. In order to improve this, it is vital to group related accesses in the same loop scope. This means that all important accesses have to be collected in one inner loop in the 2D motion estimation. The latter is usually done if one starts from a C specification for one mode of the motion estimation, but it is usually not the case if several modes are present. Indeed, most descriptions will then partition the quite distinct functionality over different functions which are not easily combined. Here is a first option to improve the access locality by reorganize the loop nest order and function hierarchy amongst the different modes.

Another important class of loop transformations is related to reversal and interchanging the loop iterators in one loop nest. For instance, the 4 loops corresponding to the window and block traversal in figure 1 can be ordered either with the window based ones as the outer or with the block based ones as outer. In this relatively simple case, a straightforward analysis of the required signal storage and the related number of transfers, shows that we have a trade-off. If the traversal over the block is put in the outer loops, the advantage is that for each pixel in the block, we can directly use it to compute all related contributions for all block locations in the window. This avoids a large amount of redundant frame accesses. However, we then need to store the resulting intermediate accumulation for the motion error for the different locations. This buffer will be quite large (16×16 words) and hence, this is not a good option. The best alternative is to put the block traversal as inner loops, surrounded by the window loops. In that case, the motion error can be directly accumulated in foreground registers, eliminating the costly background (buffer) access.

Such experiments on loop transformations are supported by our interactive loop transformation environment SynGuide in ATOMIUM. It allows to remove the tedious and error-prone steps in the transformation, while the designer can still fully control the desired manipulations.

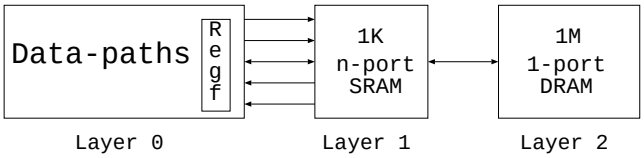

Figure 3: Memory hierarchy illustration: the foreground memory next to the data-paths (level 0) consists of registers and register files; the intermediate buffers (level 1) typically consist of fast embedded synchronous SRAMs with many ports to the lower level where potential "signal copies" from the higher levels are stored; the top layer (level 2) in this example consists of a slower mass storage unit with very low access bandwidth.

Memory architecture decision. In a second step, we have to decide on the memory hierarchy, allocation and signal-to-memory assignment. Here, the search space for possible memory configurations meeting the cycle budget is quite large. Important considerations here are the frequency of access and the size of each resulting memory. Obviously, the most frequently accessed memories should be the smallest. This can however only be fully optimized if we introduce extra memory hierarchy. The steering for this is driven by estimates on bandwidth and high-level in-place cost. Based on this, the background transfers are partitioned over several hierarchical memory levels (within a range of 1 to a maximal memory depth $MaxDepth$), to reduce the power and/or area cost. A simple illustration of this is shown in Figure 3, but also more than 3 layers may be present.

An important task at this step is to perform transformations which introduce extra transfers between the different memory levels and which are mainly reducing the power cost. In particular, these involve adding temporary values – to be assigned to a "lower level" – wherever a signal in a "higher" level is read more than once. This involves clearly a trade-off between the power lost by adding these transfers, and the power gained by having less frequent access to the larger memories in the higher layer.

Based on these considerations, an optimized memory organization has been obtained for the frame memories and the different data-path processors for 2D motion estimation. In many applications, this memory organization can be assumed to be identical for each of the parallel processors (data-paths) because the parallelism is usually created by "unrolling" one or more of the loops and letting them operate at different parts of the image data. In order to obtain a good overall memory organization, the number of processors required should however also be relatively low. Otherwise inter-processor memory sharing and optimization has to take place which is not currently supported in ATOMIUM. For the QCIF standard, the number of processors is relatively low when a reasonable clock rate is assumed. For larger search neighbourhoods or image frames this is however not true. For larger parameters, it will probably be better to go to a systolic array type solution [3, 8, 9, 15] even though much power is then spent on letting all the data "flow" through the array and on accessing the still required frame memories.

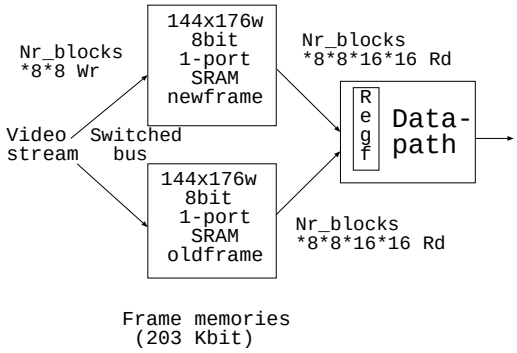

Figure 4: Data routing for the straightforward memory architecture. The formulas at the arrows indicate the amount of words that are read (Rd) from the memories or written (Wr) to the memories per frame.

The original architecture for the optimized loop order, if we assume only 1 layer of background memories is present, is shown in Fig. 4. Note that the required access rate of about $396\,blocks \times 8 \times 8 \times 16 \times 16 \times 30 fr/s = 195$ MHz in this case, is too high for the available frame memories, so two of them should be accessed in parallel (for both frames so 4 in total). Each of these can then however be half the size. The memory access power budget related to this is $4 \times 0.26 = 1.04W$.

After introduction of 1 extra layer of buffers, both for the block and the window accesses, we arrive at Fig. 5. A direct implementation leads to a switched frame memory of $2 \times H \times W \times 8$ bit, a neighbourhood buffer of $(2m + n - 1) \times (2m + n - 1) \times 8$ bit and a block buffer of $n \times n \times 8$ bit. The power budget then becomes 580 mW. In principle, the buffers need to have two ports because they have to supply data every cycle to the data-path and a second port is needed for writing the updates. As the latter are however performed at a much lower rate and as the two-port memory is very area and power hungry, it is better to increase the cycle budget per data-path a little bit to use 1-port memories instead. The best way to achieve this is by providing a slightly larger maximal clock frequency i.e. 48.86 MHz i.s.o. 48.66 MHz. A more costly alternative would be putting 1 more parallel data-path. This further optimization leads to a reduced power budget of 300 mW.

We can do even better however, if we realize that a large amount of the window pixels can be reused from the "previous" block processing. By exploiting this overlap, we can reduce the number of write transfers per block for the window buffer to $(2m + n - 1) \times n \times 8$ bit with a corresponding reduction also in read accesses to the $oldframe$ memory. The result is shown in Fig. 6. The power budget now becomes 260 mW.

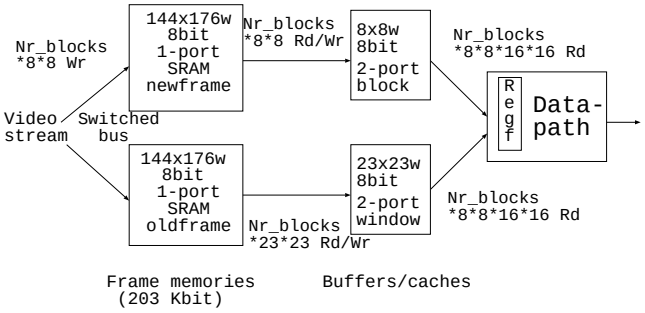

Figure 5: Data routing for the partly optimized memory architecture. The formulas at the arrows again indicate the transfers.

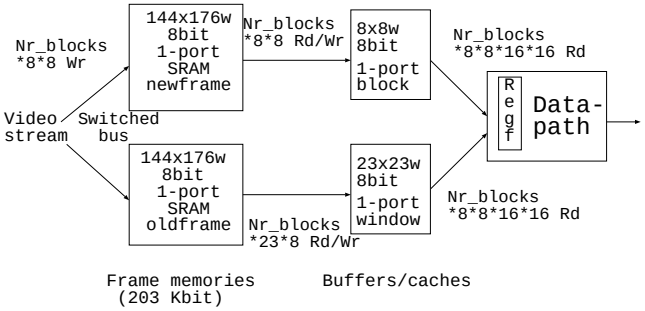

Figure 6: Data routing for the fully optimized memory architecture.

In-place mapping optimization. In a final step, each of the memories – with the corresponding multi-dimensional (M-D) signals assigned to it – should be optimized in terms of storage size applying so-called in-place mapping for the M-D signals. Instead of the two frames $oldframe$ and $newframe$ used in the initial architecture, it is possible to overlap their storage based on detailed "life-time analysis" but then extended to array signals where the concept becomes much more complex. Life-times do not just overlap anymore when the part of the array is still in use! Instead, a polyhedral analysis is required to identify the part of the M-D storage domains which can be reused "in-place" for the different signals. This analysis is depicted in Fig. 7.

Because of the operation on a 8×8 block basis and assuming the maximal span of the motion vectors to be 8, the overhead in terms of extra rows in the combined frame buffer is then only $8 + 8 = 16$ lines, by using a careful in-place compaction. This leads to a common frame memory of about $(H + 16) \times W \times 8$ bit, in addition to the already minimal neighbourhood buffer of $(2m + n - 1) \times n \times 8$ bit and a block buffer of $n \times n \times 8$ bit. For the parameters used in the example, the frame memory becomes about 0.225 Mbit. The result of the optimized memory architecture is shown in Fig. 7.

In practice, however, the window around the block position in the "old active" frame is buffered already in the window buffer so the mostly unused line of blocks on the boundary between "new" and "active old" (indicated with hashed shading in Figure 7) can be removed also. This leads to an overhead of only 8 lines in the combined $new/oldframe$ (the maximal span of the motion vectors), namely 1408 words, with a total of 26752 i.s.o $2 \times 25344 = 50688$ words (47% storage reduction). A very small power penalty is paid for this large storage area saving because the access count is the same but the storage size accessed is a little higher. Because of the relatively small frame size in QCIF, we can now however consider putting this combined frame memory of 214 kbit on chip. If low-power embedded RAMs are used[6], this will reduce the power budget further because the expensive off-chip communication is totally avoided.

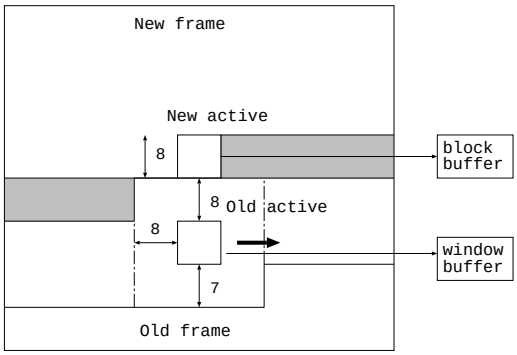

Figure 7: In place storage organisation for the optimized frame organisation.

Taking into account all the memory transfers to the frame memory, from the combined frame memory to the two buffers, and from these two buffers to the data-paths, leads to a total power budget for the memory architecture of about 260 mW using the memory power models discussed in section 4. A breakdown of this final power figure over the different buffers is shown in Fig. 8. This corresponds to a substantial saving compared to the initial situation of Fig. 4 (factor 4).

5.2 Address Generators

In addition to the memory architecture optimization, we have also explored the address generation for the original architecture of Figure 4. The necessary address generation units, based on custom data-paths, have been automatically optimized and generated from a behavioural specification with indexed signals. This has been done using the ATOMIUM address exploration and optimization environment [14]. A large number of different address architectures have been explored with this.

The results will be discussed in another paper. The main conclusion is that the best solution is neither the fully parallel (N ACUs), nor the fully sequential one (1 ACU) when a

[6] Due to the low access speed, very low-power circuitry should be feasible

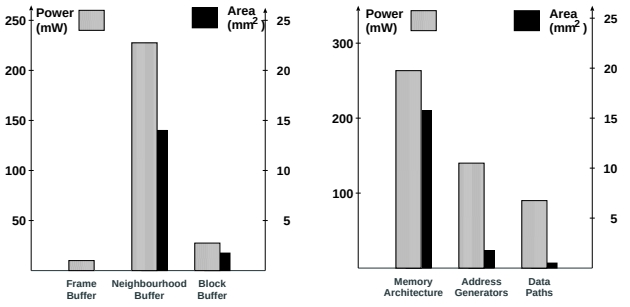

Figure 8: A breakdown of memory power and area (left) for the three memories types and (right) for the different architecture modules in the optimized memory architecture.

combined Power-Area trade off is taken into account. Hence exploration supported by a design environment is definitely needed. Moreover, we have shown that the range of accumulated power is of the same magnitude as the data-path power, which is discussed next.

5.3 Data-path and Local Control

In order to compare the power consumed in the memory architecture and the address generators with the power consumed in the data-path, we have estimated the power of the data-paths as well. However, not very much optimization was done here.

In our experiments with the 2D motion estimation kernel we have used 4 parallel data-paths, corresponding to an F_{cl} of 48.66 MHz. A block diagram of one of the data-paths can be seen in Fig. 9.

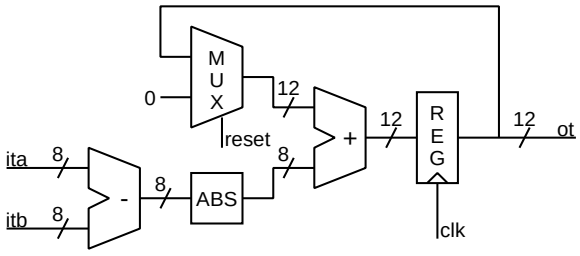

Figure 9: Block diagram of a data-path.

The widths of the blocks in the data-paths have been assumed to be 8 bit in the compare and absolute values operators and 12 bit in the accumulator. This block has been defined in VHDL and synthesized using Synopsys' Design Analyzer. The power estimation itself is done using PowerMill with random input vectors.

This has lead to 22.4 mW per data-path, including some local control but excluding the connections and external buffering. The total power consumed by the unoptimized data-paths is therefore about 90 mW.

6 Conclusion

It can be concluded from these experiments that after optimization of the memories, data-paths and address generation units, the power which goes into the memory accesses (260 mW) dominates the other contributions, which are both comparable (less than 90 mW for all the data-paths and about 140 mW for the optimized address generators). This is true even when low-power circuits are used in the memories and when power hungry standard cells are used in the data-paths and address units. Moreover, the current figures do not yet include the power of the data transfers themselves which will also consume much power (especially the off-chip ones). These transfers are also within the focus of ATOMIUM and they are equally optimized by reducing the background accesses.

References

[1] F.Catthoor, W.Geurts, H.De Man, "Loop transformation methodology for fixed-rate video, image and telecom processing applications", *Proc. Intnl. Conf. on Applic.-Spec. Array Processors*, San Francisco, CA, pp.427-438, Aug. 1994.

[2] A.Chandrakasan, M.Potkonjak, R.Mehra, J.Rabaey, R.W.Brodersen, "Optimizing power using transformations". *IEEE Trans. on Comp.-aided Design*, Vol.CAD-14, No.1, pp.12-30, Jan. 1995.

[3] L. De Vos, M. Stegherr, "Parameterizable VLSI Architectures for the full-search block-matching algorithm", IEEE Transactions on Circuits and Systems, vol. 36, pp. 1309-1316, Oct. 1989.

[4] M.Harrand, M.Henry, P.Chaisemartin, P.Mougeat, Y.Durand, A.Tournier, R.Wilson, J.Herluison, J.Langchambon, J.Bauer, M.Runtz and J.Bulone, "A single chip videophone encoder/decoder", *Proc. IEEE Int. Solid-State Circ. Conf.*, pp.292-293, Feb. 1995.

[5] K.Ishihara et al. "A half-pel precision MPEG2 motion-estimation processor with concurrent three-vector search", *Proc. IEEE Int. Solid-State Circ. Conf.*, pp.288-289, Feb. 1995.

[6] K.Itoh, K.Sasaki, Y.Nakagome, "Trends in Low-Power RAM Circuit Technologies", *Proc. of the IEEE*, Vol. 83, No. 4, pp.524-543, Apr. 95.

[7] J.M.Janssen, F.Catthoor, H.De Man, "A Specification Invariant Technique for Operation Cost Minimization in Flow-Graphs", *Proc. of the 7th ACM/IEEE Intl Symp. on High-level Synthesis*, pp.146-157, 1994.

[8] Y. Jehng, L. Chen, T. Chiueh, "An efficient and simple VLSI tree architecture for motion estimation algorithms", IEEE Transactions on Signal Processing, vol. 41, pp. 889-900, Feb. 1993.

[9] T. Komarek, P. Pirsch, "Array Architectures for Block Matching Algorithms", IEEE Transactions on Circuits and Systems, vol 36, Oct. 1989.

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

[11] C.Lin and S.Kwatra, "An adaptive algorithm for motion compensated colour image coding", *Proc. IEEE Globecom*, pp.47.1.1-4, 1984.

[12] T.H.Meng, B.Gordon, E.Tsern, A.Hung, "Portable video-on-demand in wireless communication", special issue on "Low power design" of the *Proceedings of the IEEE*, Vol.83, No.4, pp.659-680, April 1995.

[13] T.Miyazaki, I.Kuroda, and M.Imanishi, "A low-cost MPEG1 video encoder based on a single chip DSP", *Proc. DSPX*, pp.136-142, 1994.

[14] L.Nachtergaele, F.Catthoor, F.Balasa, F.Franssen, E.De Greef, H.Samsom, H.De Man, "Optimisation of memory organisation and hierarchy for decreased size and power in video and image processing systems", *Proc. Intnl. Workshop on Memory Technology, Design and Testing*, San Jose CA, pp.82-87, Aug. 1995.

[15] J.Rosseel, F.Catthoor, H.De Man, "The systematic design of a motion estimation array architecture", *Proc. Intnl. Conf. on Applic.-Spec. Array Processors*, Barcelona, Spain, pp.40-54, Sep.91.

[16] T.Seki, E.Itoh, C.Furukawa, I.Maeno, T.Ozawa, H.Sano, N.Suzuki, "A 6-ns 1-Mb CMOS SRAM with Latched Sense Amplifier", *IEEE Journal of Solid-State Circuits*, Vol.28, No.4, pp.478-483, Apr. 1993.

[17] D.Singh, J.Rabaey, M.Pedram, F.Catthoor, S.Rajgopal, N.Sehgal, T.Mozdzen, "Power conscious CAD tools and methodologies: a perspective", special issue on "Low power design" of the *Proceedings of the IEEE*, Vol.83, No.4, pp.570-594, April 1995.

[18] M.Toyokura et al., "A video DSP with a macroblock-level-pipeline and a SIMD type vector-pipeline architecture for MPEG2 codec", *IEEE J. Solid-state Circ.*, Vol.SC-29, pp.1474-1480, Dec. 1994.

[19] P.Van Oostende, G.Van Wauwe, "Low power design: a gated-clock strategy", *Low Power Workshop*, Ulm, Germany, Sep. 1994.

[20] S.Wuytack, F.Catthoor, F.Franssen, L.Nachtergaele, H.De Man, "Global communication and memory optimizing transformations for low power systems", *IEEE Intnl. Workshop on Low Power Design*, Napa CA, pp.203-208, April 1994.