Memory Centric Design of an MPEG-4 Video Encoder

Kristof Denolf, Christophe De Vleeschouwer, Robert Turney, Gauthier Lafruit, and Jan Bormans

***Abstract*—The cost-efficient implementation of video codecs requires a set of methodologies and decision taking at different levels in the design flow. We combine upfront algorithmic tuning with memory centric optimizations to transform the video application into a system consisting of functional blocks with localized data processing and a tailored memory hierarchy. This memory optimized functional description is the leverage for the cost-efficient mapping of the system on integrated multimedia platforms. It closely reflects the real implementation constraints and consequently allows for steering the architecture selection in a correct way. The proposed approach is demonstrated on a MPEG-4 video encoder and leads to its implementation as a pipelined system. Hardware development of the motion estimation validates that the high-level memory centric concepts are applicable and realizable at the lowest level. The motion estimation kernel supports up to 30 CIF f/s with minimized processing element requirements and data input rates.**

***Index Terms*—Cost-efficient design, design methodology, memory optimization, video coding.**

I. Introduction

THE quest for improved video compression performance leads to algorithms with a multitude of complex coding tools that aggressively exploit the spatial and temporal correlations characterizing moving pictures. This continuous evolution in video compression schemes results in designers opting for implementation platforms that offer a certain degree of flexibility. Software solutions on generic processors provide the highest flexibility, but this option is not viable in situations where cost and/or energy consumption is limited. Hence, low-cost microprocessors combined with specialized units in a hybrid architecture matching the diverse processing requirements of the tasks in a multimedia application are often used [1]–[4]. These solutions apply various implementation strategies offering different degrees of flexibility and parallelism [5], [6].

The typical design flow to map a video coding specification on a flexible platform starts from an initial profiling analysis [7] gained from software simulations on a generic computer platform [8] to indicate a possible system setup [e.g., hardware/software (HW/SW) partitioning] using the computational requirements as complexity representative [1]. Then, platform specific optimizations techniques, like the usage of direct memory access (DMA) and processor-specific functions are combined with the design of hardware accelerators to achieve the required throughput.

In this paper, we discuss the use of our previously proposed memory centric design methodology [9], completing the computational cost with a detailed analysis of data transfers and storage [10], applied on a simple profile MPEG-4 video encoder. We show how algorithmic optimizations are complimentary to and an enabling factor for this methodology. The memory focus is motivated by the data dominated nature of video coding algorithms, i.e., data transfer and storage is the main cost factor in an efficient realization.

Before selecting a suited architecture, memory optimizations and algorithmic tuning are performed at the highest design level (typically source C code) yielding the largest gains [11]. They transform the video application into a specification consisting of different functional blocks with localized data processing and lead to a cost efficient realization. Memory profiling statistics gained from its simulations provide relevant input for the architecture selection as this optimized system reflects the true implementation cost more closely. Additionally, the "cleaned" and optimized code is a perfect starting point to develop a functional model of the system supporting the hardware-specific design stages.

The proposed pipelined architecture includes the hardware development of the motion estimation (ME) engine to demonstrate that the high-level concepts are applicable and realizable at the lowest level. A field programmable gate array (FPGA) is used as target platform as it offers the possibility to implement an optimized architecture including a customized memory hierarchy. The resulting ME hardware kernel supports up to 30 CIF f/s with only three parallel processing elements (PEs) and minimized data input rates.

The paper is organized as follows. Section II briefly describes the MPEG-4 video encoding algorithm. Section III motivates and defines the different steps of the proposed memory centric design methodology. Sections IV to VIII detail each step of the design. Section IV performs an initial complexity analysis and presents the testbench used throughout the paper to validate the design decisions. Section V tackles the identified bottlenecks with algorithmic optimizations. Section VI continues the complexity reduction by applying memory optimizations. Section VII evaluates the effect of both steps. A suited architecture is selected in Section VIII and a dedicated kernel for the ME is developed. The conclusions in Section XIcomplete the paper.

Manuscript received September 30, 2003; revised January 31, 2004.

K. Denolf, G. Lafruit, and J. Bormans are with the Multimedia Research Group, Design Technology for Integrated Information and Communication Systems Division, Interuniversity Micro Electronics Centre (IMEC), 3001 Leuven, Belgium (e-mail: kristof.denolf@imec.be; gauthier.lafruit@imec.be; jan.bormans@imec.be).

C. De Vleeschouwer was with Interuniversity Micro Electronics Centre (IMEC), 3001 Leuven, Belgium. He is now with the Communications and Remote Sensing Laboratory, Université Catholique de Louvain (UCL), Batiment Stevin, Louvain-la-Neuve 1348, Belgium (e-mail: devlees@tele.ucl.ac.be).

R. Turney is with Xilinx Research Laboratories, San Jose, CA 95124 USA.

Digital Object Identifier 10.1109/TCSVT.2005.846430

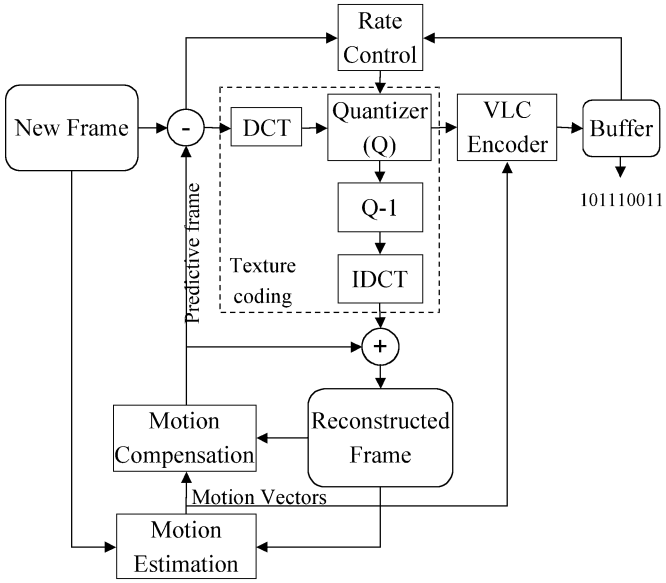


Fig. 1. Functional view of a hybrid video encoder.

II. MPEG-4 Video Coding Scheme

The MPEG-4 part 2 video codec [12] belongs to the class of lossy hybrid video compression algorithms [13]. Fig. 1 gives a high-level view of the encoder. A frame is divided in macroblocks (MBs), containing six blocks of 8×8 pixels: four luminance and two chrominance blocks. The ME enables to exploit the temporal redundancy by searching for the best match for each new input block in the previously reconstructed frame. The motion vectors define this relative position. The remaining error information after motion comensation (MC) is decorrelated spatially using a discrete cosine transform (DCT) transform and is then quantized (Q). The inverse operations Q^{-1} and an inverse discrete cosine transform (IDCT) (completing the texture coding chain) and the MC reconstructs the frame as done at the decoder side. Finally, the motion vectors and quantized DCT coefficienst are variable length encoded and completed with video header information, they are structured in packets in the output buffer. A rate control algorithm sets the quantization degree to achieve a specified average bitrate and to avoid over or under flow of this buffer.

III. Memory Centric Design Approach

A. Motivation Memory Focus

The data transfer and storage has a dominant impact on the realization efficiency of multimedia applications [9]. Indeed, the access speed, size, and power consumption of the available storage components typically form a severe bottleneck in the system, especially in an embedded context [14]. An efficient design flow should, hence, exploit the memory hierarchy to efficiently overcome the yearly growing performance gap between datapath and memory [15]. If the target platform supports customization, the memory architecture can be tailored toward the application characteristics [1], [10]. Providing relevant feedback at an early design phase enables global algorithm/implementation tradeoffs. Moreover, in complex multimedia systems, like MPEG-4, this high level analysis is essential input for an efficient architecture selection.

B. State-of-the-Art Memory Centric Design: Methodologies and Tools

The complexity of modern multimedia processing algorithms requires their definition by a reference software description, which is typically a huge executable specification. This makes a complexity analysis without additional help tedious and error-prone. Reference [7] gives an overview of state-of-the art approaches and presents a specialized tool for high-level algorithmic complexity analysis. In this paper, we run the C-in-C-out ATOMIUM tool framework[1] [16] intensively to validate and guide the design decisions. This tool suite consists of a scalable set of kernels and tools providing functionality for advanced data transfer and storage analysis, code pruning and optimization.

Memory organization and optimization has been described extensively in literature. Reference [17] presents a comprehensive treatment of this related work. Since 1996, research gradually focused on the multimedia and communication application domain related memory organization in an embedded processor context [18], [19]. It has been recognized quite early in compiler theory and high-level synthesis that in front of memory organization related tasks, it is necessary to perform transformations to optimize mainly the loop control flow [11], [20], [21]. Otherwise, the memory organization will be suboptimal.

C. Design Steps

The proposed design approach starts from a reference executable specification. The following list summarizes the different steps to reach a cost-efficient implementation and links them to the section in which the step is applied on the video encoder. During their practical application, step 2 and 3 are typically jointly applied.

1) Pruning and Complexity Analysis (Section IV): pruning extracts the required functionality from the reference code according to the chosen application profile. An initial complexity analysis identifies bottlenecks and points out the first candidates for optimization.

2) Algorithmic Tuning (Section V): simplifies the algorithm of non-normative parts of the application to reduce their implementation cost and/or to enable memory optimizations.

3) High-Level Memory Optimization Steps (Section VI): reduce the memory cost of the application. They focus on minimizing the memory access frequencies, introducing locality (avoid accesses to large memories, implement data-reuse, etc) and shrinking the memory footprint. More details are given in [9] and [10]. In some cases, algorithmic tuning is required to allow for the complete application of all principles.

4) Architecture Selection and Final Implementation (Section VIII): fixes the target system architecture (including the memory hierarchy) and maps on this platform. As the code resulting from the previous design steps is optimized and

[1][Online]. Available: http://www.imec.be/atomium.

TABLE I
CHARACTERISTICS OF THE VIDEO SEQUENCES IN THE APPLIED TESTBENCH

Test case	# frames	Frame rate	Bitrate (kbps)	Compr. factor	PSNR Y
1. m&d 15 qcif fps 20	150	15	20.1	228	33.9
2. m&d 30 qcif fps 60	300	30	61.6	151	36.5
3. m&d 30 cif fps 120	300	30	120.9	296	36.9
4. for 12.5 qcif fps 50	150	12.5	50.1	76	31.1
5. for 25 qcif fps 150	300	25	150.3	51	34.1
6. for 12.5 cif fps 150	150	12.5	150.2	101	31.4
7. for 25 cif fps 450	300	25	450.4	68	34.3
8. c&m 10 qcif fps 300	100	10	300.3	10	32.1
9. c&m 10 cif fps 1000	100	10	998.9	12	32.4
10. c&m 15 cif fps 2000	150	15	2000.1	9	34.5

The last number of the test case name indicates the bitrate in kbps.

TABLE II
MEMORY ACCESSES AND TIME DISTRIBUTION OVER THE MAIN FUNCTIONAL BLOCKS OF THE INITIAL VIDEO ENCODER

Function	Access frequency (Maccesses/s)	Relative # accesses (%)	Relative # cycles (%)
Motion estimation	4149.7	83.9	84.2
Texture coding	286.6	5.8	5.8
Motion compensation	99.6	2.0	1.8
Capture	19.0	0.4	0.7
Reconstruct	26.5	0.5	0.3
Calculate error	11.4	0.2	0.2
Manipulate frame	113.0	2.3	1.5
Padding	53.6	1.1	1.5
Others	186.3	3.8	4.1

"cleaned," it is well suited to produce the profiling/instrumentation data required for the architecture definition. Additionally, it is an ideal starting point to create the functional model for the hardware development.

IV. INITIAL COMPLEXITY ANALYSIS AND TEST BENCH DEFINITION

This section first explains the pruning of the executable specification to extract the parts of the code relevant for the given class of applications. Then the testbench used in the rest of the paper to validate the performance of our design optimizations is defined. An initial cycle and memory access distribution is presented for the typical test case of this set.

The software used as input for this design is the verification model accompanying the Final Draft International Standard Natural Visual Part 2 [22]. The availability of an executable specification at the start of the design process overrides the tedious task to implement a system from scratch. However, in the case of MPEG-4 the software specification contains all functionality, resulting in oversized C code distributed over many files. A necessary first step in the design is thus the extraction of the part of the reference code corresponding to the desired MPEG-4 profile and level. ATOMIUM pruning is used to automate this error-prone and tedious task. It removes the unused functions and their calls based on the instrumentation data of a testbench representative for the desired functionality [16]. This implies careful selection of the set of input stimuli, which has to exercise all the required functionality. Applying automatic pruning with a simple profile dedicated set of coding parameters (not detailed in this paper) shrinks the code to 30% of its original size (first two rows of Table V in Section VII). Thanks to this code reduction and simplification, manual, and interactive reorganization/rewriting becomes feasible.

The testbench of Table I is used to make an initial complexity assessment and to evaluate the effect of the optimization steps. The selected coding samples have different sizes (QCIF and CIF), an increasing degree of movement and are compressed at different rates. All sequences are encoded without data partitioning and with rate control enabled (assuming an infinite buffer to avoid frame skipping). *Mother and Daughter (m&d)* is a low complexity, head, and shoulders type sequence. *Foreman (for)* has a medium complexity and *Calendar and Mobile (c&m)* is a highly complex sequence, with heterogeneous movement (including rotation). The *Foreman* CIF 450 kb/s stream (test case 7 of Table I) serves as typical representative when more in depth analysis is made in the rest of the paper. It is, with varying motion properties through time, a real life example.

Profiling information gained from simulations provide a sufficient basis for an initial estimation of the complexity [8], [7]. Combining the memory requirements measured with ATOMIUM/Analysis with a cycle distribution obtained with Quantify on a HP9000/K460, 180-MHz RISC platform allows for the identification of the most demanding tasks of the coding algorithm [23] and to verify their data-dominance. Table II lists the memory access and time distribution over the main functional blocks of the initial video encoder specification. Next to the processing steps detailed in Fig. 1, frame padding and manipulation are added to the reference software. The first function extends a frame with a border to handle motion vector pointing outside of the image, the second one makes internal copies of a frame.

Fig. 12 (in Section VII-A) groups the accesses to 4 memory size categories: frame memories with as minimal size the image, large buffers containing more than 64 elements, buffers with 9 to 64 elements and registers with maximally 8 elements. In this analysis stage, the wordlength of the elements is not considered. The first bar Fig. 12 represents the non-optimized status: 40% of the total number of accesses are to frame memories, 55% to a large buffer, 2.3% to a buffer and 2.8% to registers. The data flow of the reference encoder is frame based: most of its accesses are to large or frame sized memories.

This is in contrast to the ultimate goal of the memory optimizations: a scheme with the absolute minimal number of (large and, hence, costly to access) frame memories and associated accesses to them. The typical subdivision of a frame in MBs in MPEG-4, makes a MB-based processing to increase data locality a logical solution. However, an important hurdle is the rate control traditionally operating on statistics of a full frame (Fig. 2). Section V-A explains the removal of this rate-distprtion (R-D) modeling barrier.

No new major cycle consuming tasks are identified during the RISC platform analysis: the high correlation between the memory access ratios and the cycle distribution confirms the data-dominance of these functional blocks. The known main bottleneck of a video encoder is the ME [24], taking more than

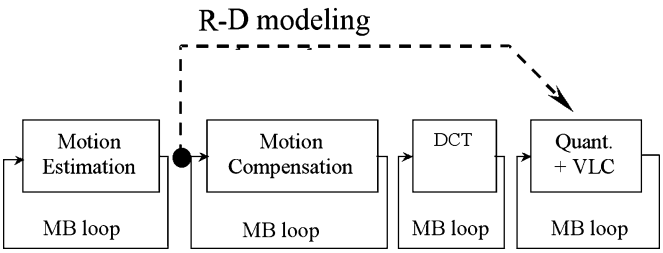


Fig. 2. Simplified block description of a video encoder emphasizing the dependency caused by the rate control. Each step of the algorithm processes the data locally, within a MB loop. A complete MB-based flow is hindered by the rate control dependency.

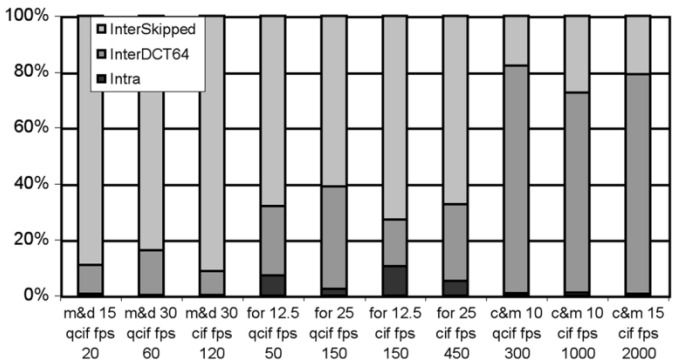


Fig. 3. Relative block type proportions over different video sequences. The amount of skipped blocks is inversely proportional to available bitrate and the complexity of the sequence. The prediction of these blocks eliminates their texture coding.

80% of the resources (Table II). Section V-B uses the implementation freedom available at the encoder side to present a low complexity search algorithm.

V. Algorithmic Tuning

Two kinds of algorithmic optimization are applied to the video encoder: modifications to enable a one-pass sequential processing of the frame MBs (i.e., increasing memory access locality) (Section V-A) and tuning to reduce the required processing for each MB (Sections V-B and V-C). Both aim at trading a minimal compression performance loss for a substantial complexity reduction (Section V-D).

A. Predictive Rate Control

The limited bandwidth resources of existing networks restrict the available output bitrate of a video encoder. Typically, standardized hybrid video coders (like MPEG-4) produce a highly variable output bitrate, depending on the characteristics of the video content. In practice, a first-in, first-out (FIFO) buffer is placed between the coder and the network. As its size is limited, (proportional to the transmission delay) the encoder has to carefully regulate its output rate to avoid buffer overflow or underflow while producing an acceptable visual quality.

The default *RC* adopted by MPEG-4 is a technique combining R-D modeling with sequence pre-analysis to achieve the best performance [25]. This approach makes a one-pass sequential encoder implementation impossible as a measure of the average compensation error energy, e.g., the mean absolute difference (MAD), is needed to fix the quantization parameter (QP). Because this pre-analysis step is performed on the whole compensation error frame, quantization, and bitstream generation can not start before ME has been performed on all the MBs of the frame. This forms a barrier for the memory optimizations (see Section VI-A). The development of a predictive rate control (a detailed description is given in [26]), calculating the MAD by only using past information, breaks this restriction and enables a true macro block-based processing. Additionally, the amount of frame memories is reduced as the storage of the compensation error frame becomes obsolete.

B. Directional Squared-Search ME

The high computational complexity of the full-search ME requires algorithmic tuning to decrease the number of sum of absolute difference (SAD) calculations during the search. Reference [27] describes a directional squared search ME, based on a set of common design rules for efficient ME algorithms derived from literature. This method, like other fast block matching algorithms, trades a minimal loss of coding efficiency to achieve a significant reduction of the number of searched positions and consequently it has an improved cost efficiency. Moreover, the directional squared search ME is developed with hardware implementation constraints in mind, leading to improved data reuse possibilities [28].

C. Intelligent Error Block Processing

The texture coding task is the second largest access and cycle consumer of Table II. This chain of DCT, quantization, inverse quantization and IDCT uses indeed many multiplications and additions. This subsection omits all or part of the computations when the quantized coefficients can reasonably be estimated to be equal to zero.

In motion compensated blocks, the reconstructed prediction error is likely to be reduced to zero due to the quantization of nonrelevant coefficients. Such all-zeros blocks are referred to as skipped in the following, as their texture decoding stage can be omitted on the decoder side. Their amount is inversely proportional to the available bitrate and to the complexity of the video sequence (Fig. 3).

The coding status of a prediction error block is only known after the DCT and quantization process. For a skipped block, the computations in theses steps become overhead as the resulting reconstructed texture block contains only zeros. Algorithmic tuning realizes the prediction of the coding status and avoids useless processing.

This coding status is determined based on (1), by comparing the SAD a block $(\mathrm{SAD}_{\mathrm{block}})$, reflecting the prediction error, to a combination of the QP and the absolute deviation $(\mathrm{AD}_{\mathrm{macroblock}})$ of the MB, quantifying its activity

$$\mathrm{SAD}_{\mathrm{block}} < T \cdot \mathrm{QP} + \frac{\mathrm{AD}_{\mathrm{macroblock}}}{200}. \tag{1}$$

The first term in (1) is proportional to QP as a higher quantization level sets larger error coefficients to zero. References [29] and [30] show all quantized coefficients are zero when $\mathrm{SAD}_{\mathrm{block}} < 20 \times \mathrm{QP}$. The second term in (1) is deducted empirically (more specifically, the factor 200 is derived using extensive testbenches) and reflects the dependency on the AD: a larger

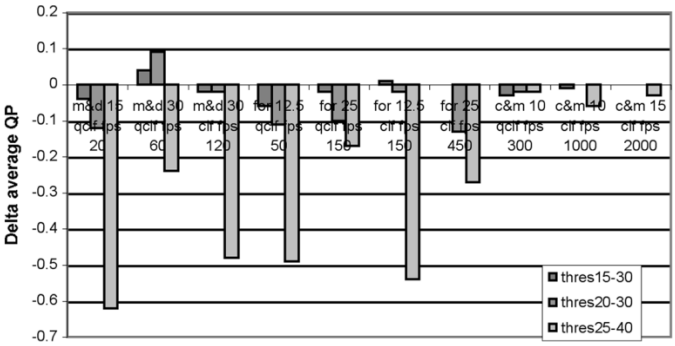


Fig. 4. Average QP delta for different threshold settings ($T_{\text{skip}} = \{15, 20, 30\}$, and $T_{\text{dct8}} = \{30, 30, 40\}$) on the videos in the testbench compared to not using the intelligent block processing (no prediction of the block type). Making the thresholds too flexible results in a loss of detail information (lower average QP).

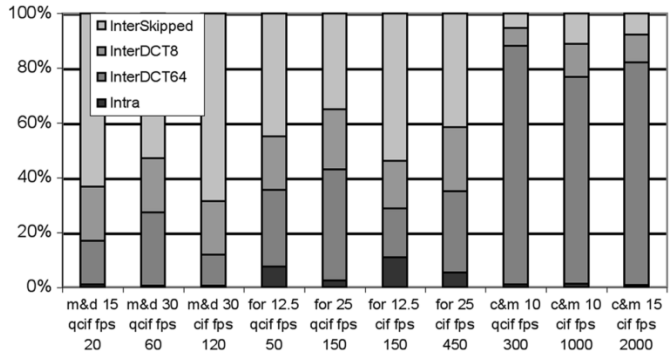


Fig. 5. The block type after the texture coding is accurately predicted to reduce the texture coding complexity. This results in 40% skipped blocks and 20% with simplified processing (row or column only) for the typical *Foreman* CIF 450 kb/s test case.

$\text{SAD}_{\text{block}}$ is acceptable for MBs with high activity. The intuition behind it is that a large activity corresponds to a spreading of the energy among the DCT coefficients. If the energy has been spread among several coefficients, the magnitude of the largest ones is smaller than if the energy was concentrated on a few coefficients. Moreover, high activity also means high frequency coefficients, and these ones are sometimes (depending on the quantization matrix) quantized more coarsely. Hence, if the block prediction error is small enough (evaluation of (1) with $T = T_{\text{skip}} = 20$), the block is expected to result in all-zeros quantized coefficients and consequently, the algorithm decides to encode it as a skipped block.

Next to skipped blocks, observations show that most DCT blocks with small SAD values only contain nonzero quantized coefficients on either their first row or their first column. Specifically, experiments show that using (1) with $T = T_{\text{dct8}} = 30$ identifies the blocks only containing relevant coefficients in one direction. The prediction of these blocks allows for complexity reduction because only one row or one column DCT needs to be computed. In practice, the one-dimensional (1-D) DCT is calculated after initial summations along the rows or columns of the block.

Equation (1) and its values of T, i.e., $T_{\text{skip}} = 20$ and $T_{\text{dct8}} = 30$ appear to provide the best trade off between computational complexity reduction and preserving a homogeneous quality on the video frames (Fig. 4). Actually, when too many blocks are skipped, additional bits become available that are re-allocated to other blocks. To keep the total bit budget constant, the rate control decreases QP and in this way, increases the quality of these encoded blocks. Hence, too large thresholds result in a loss of detail information on the skipped or partially coded prediction error blocks and consequently in a drop of the average QP for the other blocks. Fig. 4 indicates that QP preservation requires a maximum value for T_{skip} and T_{dct8} of 20 and 30, respectively.

Fig. 5 shows the proportion of each kind of texture blocks before processing (DCT/Q/Q^{-1}/IDCT). The amount of blocks predicted for simplified processing corresponds closely to the number of finally not coded blocks when no intelligent processing is used (Fig. 3). The prediction avoids the computation overhead by identifying in advance blocks for which the DCT is expected to result in zero quantized coefficients.

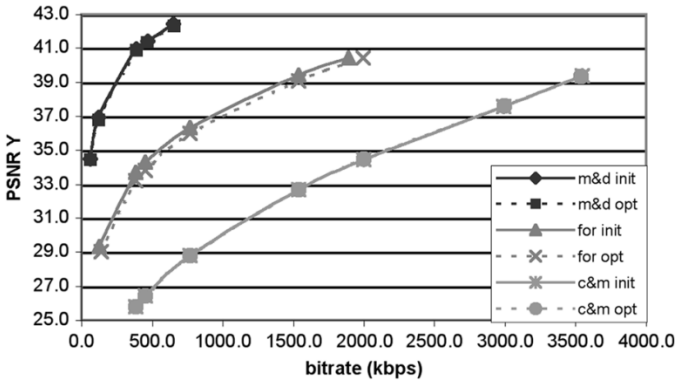


Fig. 6. R-D graphs of the CIF sized video sequences using the initial and the algorithmically tuned video encoder. In worst case, the complexity reduction results in a 0.5-dB compression performance loss.

D. Impact on the Compression Performance

Each of the three algorithmic optimization steps sacrifices a minimal amount of compression performance for a substantial complexity reduction as quantified in Section VII-A. For the typical test case (*for* CIF 450 kb/s) 0.5 dB quality loss is traded for a lower implementation cost: the ME requires 25 times less memory access, the texture coding skips 40% of its processing and the bit rate controller is simplified. Fig. 6 measures the impact on the quality by means of R-D graphs for the CIF sized video sequences of the testbench.

VI. High-Level Memory Optimizations

A. Data Reuse and MB-Based Data Flow

The initial analysis of Section IV points to the ME as the main memory bottleneck and to the frame-based character of the reference software: a worst-case combination from the memory point of view. However, the ME is intrinsically a localized process: the matching criterion computations repeatedly access the same set of neighboring pixels. To reduce these accesses to frame size memories, a memory hierarchy is introduced in the ME, enabling data reuse. Heavily used data is copied from large (frame size) to minimal intermediate memories, used for the actual ME processing. The solution is more efficient as soon as the cost of extra memory transfers (due to copies) is balanced by the advantage of using smaller memories. The

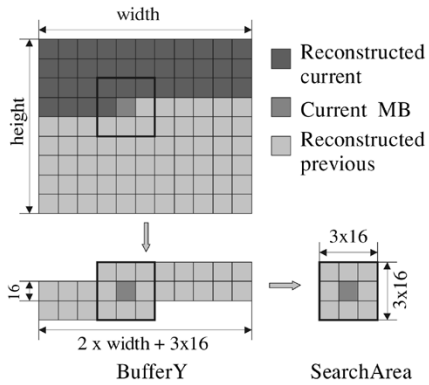


Fig. 7. Memory hierarchy supporting three levels of data reuse for the ME.

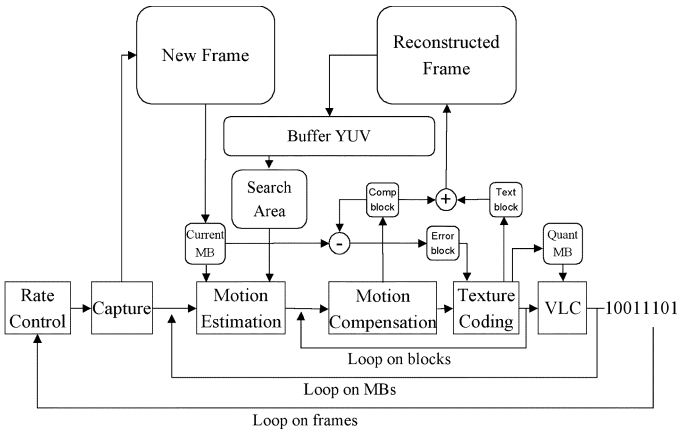


Fig. 8. Memory optimized video encoder works in a MB-based way to localize the processing steps.

choice of an optimal hierarchical memory partition for ME has been extensively studied in [31] and [32], focusing on the full search ME algorithm.

With regard to the adaptive data flow of the algorithmically tuned ME (see Section V-B and [28]), only three levels of hierarchy are relevant. The two additional lowest data reuse levels introduced in [31] and [32] can inherently and adequately be implemented by the ME processing kernel (see Section VIII). The three relevant levels are presented in Fig. 7. For the luminance information, a $(2 \times \text{width} + 3 \times 16) \times 16$ pixels Y-buffer stores the data of the previous reconstructed frame required by the ME/compensation. The search area buffer is a local copy of the values repetitively accessed during the SAD calculation. Both chrominance components have a similar $(\text{width} + 3 \times 8) \times 8$ pixels U/V buffer to copy the data of the previously reconstructed frame, needed by the MC. In this way, the newly coded (macro)blocks can immediately be stored in the frame memory. The MC is performed on the Y/U/V buffer.

To increase locality, the encoding algorithm is reorganized according to Fig. 8, supporting MB-based processing, hence, ending up with block sized (texture block, comp. block, error block) buffers, MB-sized (current MB, quant. MB) memories and layered extensions thereof (search area, buffer YUV, see Fig. 7). The algorithmically tuned rate control (see Section V-A), combined with the introduced redundancy and the MB-based data flow of the coding process make two frame memories sufficient to complete the encoding (Fig. 8): one for the input (new frame) and one for the reconstructed data (reconstructed frame).

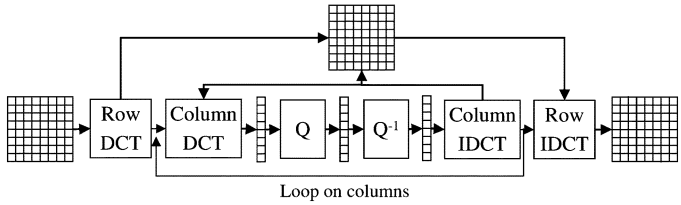


Fig. 9. Texture coding process works column based.

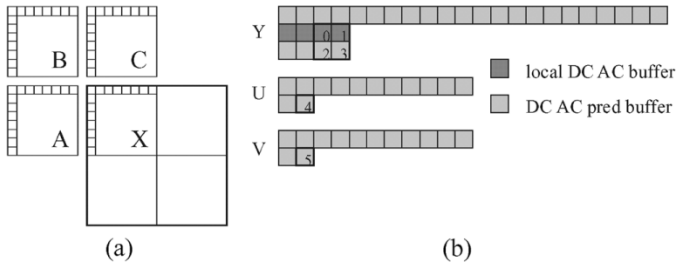


Fig. 10. (a) Neighboring blocks required to perform dc ac prediction of block X. (b) Use of a local buffer allows minimization of the dc ac prediction buffer.

B. Submacroblock Localized Processing and Buffer Size Reduction

A block-based loop is implemented for the DCT/Q/Q^{-1}/IDCT chain of the texture coding. The typical row-column decomposition in a (I)DCT implementations allows a further refinement to column-based processing (Fig. 9).

After the row DCT processing of the 8×8 input block, the column processing sequentially applies the DCT, the quantization, inverse quantization and the IDCT on each column. The row IDCT completes the texture coding chain. This optimization thus interchanges the row and column processing of typical IDCT modules to a column-row decomposition version.

The MC is merged with the block-based texture coding loop to improve locality. The buffer YUV (Fig. 7) allows the immediate storage of newly reconstructed blocks in the frame memory. This updating is only performed if the content of the block has changed (i.e., blocks of the "skipped" type with both motion vectors zero are not copied in the frame memory).

By introducing a local buffer of 4×15 elements, the size of the dc ac prediction buffer can be minimized to $((\text{width}/16) + 2) \times 4 \times 15$ [Fig. 10(b)]. All data required to predict the coefficient of the blocks (marked with 0–5) of the currently decoded MB is available in the local buffer or dc ac prediction buffer. When the neighboring blocks belong to an inter-MB, defaults settings are used for the values of the dc and ac coefficients. By using a third memory containing an interflag per MB in the dc ac prediction buffer, the storing and reading back of these default coefficients can be excluded. This drastically reduces the accesses from/to the dc ac prediction buffer as most MBs in a video sequence are intercoded (Fig. 3).

The size of the buffers required for the dc and ac prediction is minimized. The dc and ac prediction tries to code the dc coefficient and some ac coefficients in the frequency domain (in the first row or the first column) of an intra-MB more efficiently by exploiting the values of the neighboring blocks [Fig. 10(a)].

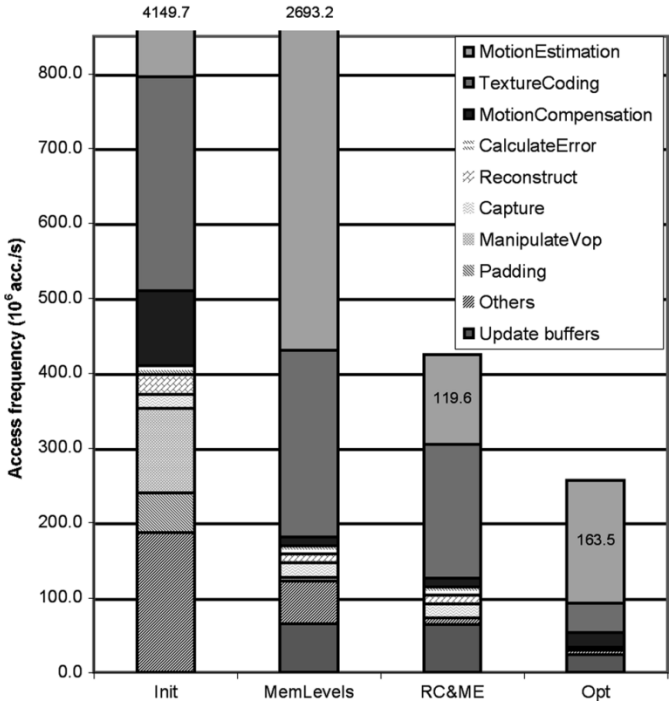


Fig. 11. Evolution of the access distribution over the main functional blocks during the optimization process using the *for* CIF 450 kb/s test case. The ME part has been truncated for the first two steps. The number indicates the complete share.

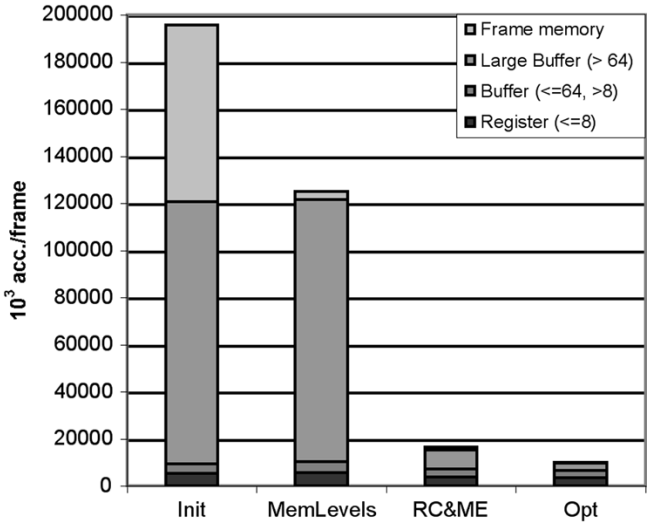


Fig. 12. Evolution of the access distribution over different memory sizes during the optimization process using the *for* CIF 450 kb/s test case. The total amount of accesses is reduced with more than a factor 15 and small memories are favored to restrict the associated cost.

For every block, these fifteen coefficients need to be stored in memory to be able to complete this prediction process.

VII. Optimization Results

The main cost figures of merit aimed for during the complete memory centric optimization process are: 1) the total amount of memory accesses; 2) the memory footprint (i.e., the accumulated memory size); and 3) the locality of the accessed data. Additionally, the benefit of the memory optimizations measured on a generic computer platform is used as sanity check to validate the improved efficiency of the (optimized) functional model. This cross check allows using the profiling, completed with ATOMIUM/Analysis data, as first step in the system architecture development. Finally, an important side effect is the reorganization and clean-up of the C source code, resulting in a reduced number of code lines and, hence, a better starting point for the functional model required during the hardware development process.

A. Memory Access Frequency and Peak Memory Usage

The algorithmic tuning and memory optimizations are jointly implemented on the video decoder. The evolution of the access distribution during the different optimization steps is shown over the main functional blocks in Fig. 11 and over the different memory size categories in Fig. 12. The labels used in these figures correspond to different software versions and are defined as follows.

1) Init: The initial encoder has a frame-based data flow.
2) MemLevels: The introduction of a memory hierarchy, redirects the ME frame accesses to large buffers. This extra storage of the previously reconstructed data makes one frame memory sufficient to complete the encoding process and consequently the internal copying (manipulate frame and padding of Table II) becomes superfluous.
3) RC&ME: The algorithmic tuning of the ME minimizes the number of evaluation steps in the motion vector search resulting in a compacted ME share in Fig. 11 and less large buffer accesses in Fig. 12. The predictive *RC* enables a true MB-based data flow (less frame accesses in Fig. 12).
4) Opt: The intelligent block processing together with the rest of the optimizations further reduce the texture coding contribution and continue favoring buffers and registers.[2]

Most of the optimized encoder accesses are to registers (35%) and buffers (31%). Twenty-eight percent of the transfers are to large buffers and the frame memory take 6%. The ME/compensation and texture coding consume almost all transfers.

Table III lists the total effect of the memory optimizations on all sequences of the testbench. Typically, the access frequency is reduced with almost a factor 20. In the worst case (*Calendar and Mobile*), 200 mega accesses per second are required to encode a CIF sequence at 15 f/s. The peak memory usage denotes the maximum amount of memory allocated by the instrumented arrays during the execution of the program on a test case. It is an estimated lower bound for the actual memory usage [16]. The MB-based encoder reduces the amount of memory with a factor 18.3–35.2, to 85 kB for QCIF and 309 kB for CIF.

B. Performance

Testing the performance on a HP9000/K460 RISC processor running at 180 MHz with UNIX as operating system (Table IV) cross checks the positive effect of the memory optimizations on

[2]The increase of the ME accesses in the opt bar of Fig. 12 is due to the implementation of data reuse at the buffer and register level. The error calculation is merged inside the MC in the Opt version (Fig. 12).

TABLE III
PERFORMANCE IMPROVEMENT OF THE MEMORY OPTIMIZED ENCODER ON A RISC PROCESSOR AT 180 MHz

Test Case	Version	# accesses per frame (10^6 acc. frame)	Access frequency (10^6 acc./s)	Reduction factor	Peak memory usage (kB)	Reduction factor	Bitrate (kbps)	PSNR Y
M&d 15 qcif fps 20	Initial	46.1	692.0		2816.7		20.1	33.9
	Optimized	1.8	27.7	25.0	84.7	33.3	20.0	33.8
M&d 30 qcif fps 60	Initial	44.0	1321.1		2816.7		61.6	36.5
	Optimized	2.0	58.7	22.5	84.7	33.3	65.1	36.6
M&d 30 cif fps 120	Initial	190.3	5710.1		5635.7		120.9	36.9
	Optimized	6.4	192.0	29.7	308.8	18.3	122.2	36.8
For 12.5 qcif fps 50	Initial	48.6	607.6		2969.6		50.1	31.1
	Optimized	2.7	33.4	18.2	84.7	35.1	50.0	30.6
For 25 qcif fps 150	Initial	45.8	1145.2		2969.6		150.3	34.1
	Optimized	2.8	68.8	16.6	84.7	35.1	150.6	33.7
For 12.5 cif fps 150	Initial	208.5	2606.6		5656.7		150.2	31.4
	Optimized	9.9	123.8	21.1	308.8	18.3	150.2	31.1
For 25 cif fps 450	Initial	197.8	4945.8		5656.7		450.4	34.3
	Optimized	10.3	256.3	19.3	308.8	18.3	449.9	33.8
c&m 10 qcif fps 300	Initial	52.0	520.3		2982.6		300.3	32.1
	Optimized	3.4	33.7	15.4	84.7	35.2	299.6	32.1
c&m 10 cif fps 1000	Initial	209.2	2092.2		5666.8		998.9	32.4
	Optimized	13.5	134.5	15.6	308.8	18.4	999.5	32.4
c&m 15 cif fps 2000	Initial	208.1	3122.0		9913.3		2000.1	34.5
	Optimized	13.5	201.9	15.5	308.7	32.1	2000.2	34.5

TABLE IV
CODE SIZE REDUCTION AFTER AUTOMATIC PRUNING AND MANUAL REORGANIZATION AND OPTIMIZATION

Test Case	Initial	Optimized	Speed up factor
m&d 15 qcif fps 20	0.4	9.4	22.8
m&d 30 qcif fps 60	0.4	8.8	20.5
m&d 30 cif fps 120	0.1	2.7	27.7
for 12.5 qcif fps 50	0.4	6.2	15.8
for 25 qcif fps 150	0.4	6.1	14.9
for 12.5 cif fps 150	0.1	1.6	17.3
for 25 cif fps 450	0.1	1.6	16.6
c&m 10 qcif fps 300	0.3	4.4	13.5
c&m 10 cif fps 1000	0.1	1.0	11.2
c&m 15 cif fps 2000	0.1	1.0	11.6

Initial and Optimized denoted the achieved throughput expressed in frames per second of the respective version.

TABLE V
EFFECT OF ALL MEMORY OPTIMIZATIONS ON THE ACCESS FREQUENCY, PEAK MEMORY USAGE AND COMPRESSION EFFICIENCY

Code version	Number of lines	Reduction
Initial	67 k	
Pruned	22 k	3.1
Optimized	9 k	7.6

The reduction is expressed as a factor compared to the initial code size

the efficiency of the functional model. The measured speed up factor ranging from 11 to almost 23 validates this improvement on a generic processor used a test platform for this cross check.

C. Code Size Reduction

ATOMIUM pruning extracts the functionality corresponding to the selected profile (see Section IV). Further manual code reorganization and rewriting shrinks the number of lines to 13.1% of the original (Table V). This last reduction is obtained by flattening the hierarchical function structure and by further simplification of the functionality thanks to memory optimizations.

VIII. HARDWARE DEVELOPMENT

A. Architecture Selection

A typical video processing architecture is heterogeneous and consists of a low-end microprocessor connected with hardware accelerators and external memory through a fast bus [1]–[4], [8]. The hardware engines implement the critical parts of the video algorithm to boost the design to the required real-time performance. In contrast, this paper proposes a pipelined video encoder architecture using a micro controller as system orchestrator.

The MB-based flow of the memory optimized video encoder (Fig. 8) has a tailored memory architecture using buffers of MB or block size for the link between tasks with the highest data traffic. Frame memories are only read once and written once per pixel. Buffers containing multiple MBs exploit data-reuse to feed the reference data to the ME. This localized processing facilitates the integration in a video pipeline. The (macro)block sized buffers are implemented as queues between the different processing blocks, larger buffers and frames become shared memories. Due to the memory optimizations, the size of the queues and the communication passing through them is minimized. Additionally, the amount of local accesses in the processing blocks is also reduced. This makes their HW implementation more feasible. The pipelined video architecture avoids the need for a fast bus typical for hybrid platforms by using specialized data communication mechanisms.

FPGAs offer a flexible means to realize this customized architecture including the dedicated communication channels. They also allow the integration of low-end microprocessors that can perform the system orchestration. The next subsections describe the hardware development of the ME that is then integrated with a Xilinx MicroBlaze 32 bit RISC processor on the Virtex2 FPGA as integrated multimedia test platform.

B. ME Hardware Block

The hardware development of the ME is detailed in this section as implication of both algorithmic tuning and memory op-

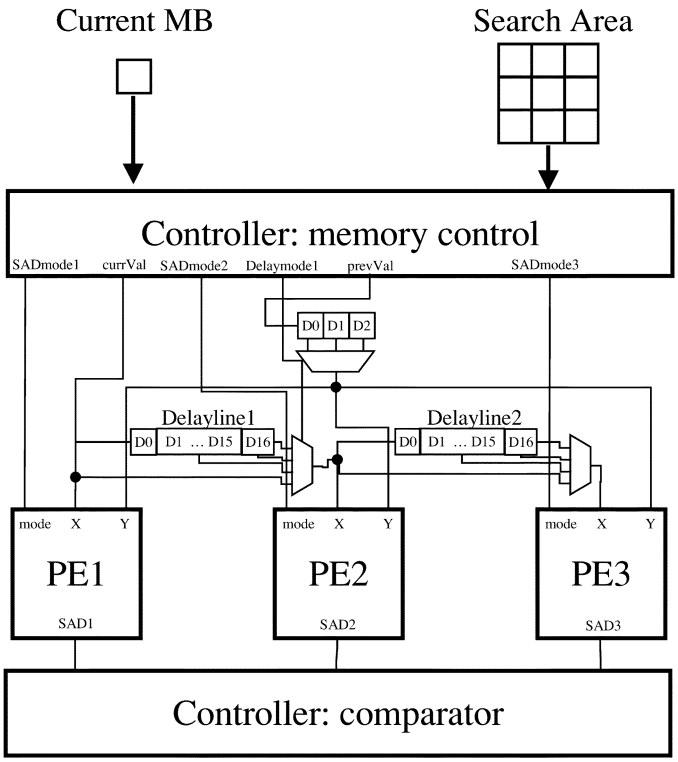


Fig. 13. Three PE proposed ME kernel implementing pixel-level data reuse.

timization can be demonstrated down to the lowest design level. First an architecture corresponding to the algorithmic properties of the directional squared-search ME is developed. Then, the implementation results on a Xilinx FPGA are presented.

1) ME Kernel Architecture: The main targets in developing the ME hardware kernel are: 1) reducing the communication to interface memories by exploiting pixel level data reuse on both current and reference data; 2) minimize the number of parallel PEs providing the required throughput; and 3) maximizing the utilization of these PEs. The architecture (Fig. 13) exploits similar principles as the systolic array concepts used in development of full search ME engines [33], [34] but the adaptive aspect of algorithm is translated in configurable delaylines and multimode PEs. For an in depth description of the VLSI architecture selection, the reader is referred to [28].

The PE is an absolute difference accumulator able to store one or three SADs (two modes). The new data input of the 3 PEs is connected through two variable delaylines (that can be configured as wire or to give a delay of 1, 16, or 17). A third delayline is added on the path of the reference data coming from the search area. The controller on top of these elements fetches the correct data from the interface memories and configures the PEs and delaylines during the different stages of the directional squared-search ME algorithm. An interpolation unit to support the halfpel stage and a comparator to select the best match is also contained in the controller hardware [28] (not detailed in Fig. 13). A detailed controller configuration description is out of the scope of this paper.

The combination of the delaylines and the two modes of the PEs enable to keep the data locally inside the ME hardware kernel during calculation of the SAD at horizontally or vertically adjacent positions and, hence, implement data-reuse at the lowest level of the memory hierarchy as recommended in [31] and [32]. In worst case, PE2 is active 88% of the time and PE3 90%. PE1 is fully occupied during all phases of the algorithm.

TABLE VI
MEASURED AVERAGE CYCLE CONSUMPTION PER MACROBLOCK AND REQUIRED OPERATION FREQUENCY FOR REAL-TIME OPERATION OF THE MOTION ESTIMATION KERNEL

Test Case	Average # cycles per MB	Operation frequency (MHz)
m&d 15 qcif fps 20	1177	1.7
m&d 30 qcif fps 60	1720	5.1
m&d 30 cif fps 120	1545	18.4
for 12.5 qcif fps 50	2054	2.5
for 25 qcif fps 150	2077	5.1
for 12.5 cif fps 150	1895	9.4
for 25 cif fps 450	1972	19.5
c&m 10 qcif fps 300	2312	2.3
c&m 10 cif fps 1000	2306	9.1
c&m 15 cif fps 2000	2299	13.7

2) Implementation Results: The ME hardware description is synthesized using Synplify Pro and mapped, place, and routed on a Xilinx Virtex2 (xc2v2000) using the Xilinx ISE 5.2 tools. The FPGA resource utilization for the ME is 2099 slices (20% of xc2v2000) and 3 Block RAMs (5% of xc2v2000, 2kByte).

A rough comparison with the full search implementations shows that 169 PEs are required to complete the ME of one MB with in cycle budget consumed during the full pell stages of the proposed ME kernel [34]. Seven thousand seven hundred slices would be required to only realize the PEs (i.e., without control or halfpel support).

C. System Integration and Testing

The ME estimation kernel is integrated with a Xilinx MicroBlaze 32-b RISC processor on the Virtex2 FPGA for functional verification and performance measurements. The average number of cycles required to find the motion vectors of a MB is measured on this test platform (Table VI). Real-time constraints are met for all test cases with the ME kernel.

Extended tests with a fixed QP at different framerates confirm the adaptive character of the ME: more cycles are required at lower frame rates to find the larger motion vectors due to the increased time intervals. In this respect, the directional squared-search ME makes a more efficient use of its HW resources than a full search implementation. The latter needs to be designed for the worst case, the first exploits the relaxed timing constraints at lower framerates to perform a more extensive search. A similar observation holds for lower resolutions. The higher image activity in a MB extends the amount of steps performed by the directional squared-search. Again, the relaxed timing constraints are efficiently used.

IX. CONCLUSION

Cost-efficient video compression systems are typically implemented on hybrid architectures containing a low-cost microprocessor connected with specialized units and external memory through a fast bus. After a profiling analysis of the application on a generic processor to determine the system setup, platform specific design steps are applied to achieve the required throughput.

We precede this architecture selection with a combination of algorithmic and memory centric optimizations to tackle first the dominant data transfer and storage cost in multimedia applications. Only then, based on the memory specs of the optimized version, a suited architecture is defined.

The high-level memory optimizations are combined with algorithmic tuning of the rate control scheme to be able to achieve the minimal memory cost (i.e., the smallest amount of frame sized memories and associated accesses to them). On other parts of the encoder scheme (e.g., the ME and the texture coding), the algorithmic tuning decreases the processing requirements. Both optimization steps are complementary and enforce each other, yielding an important reduction of the computational complexity and of the memory resources.

The output of these optimization stages is an optimized specification containing different functional blocks with localized, MB-based data processing and a memory hierarchy tailored towards the application, hence, leading to the definition of a cost-efficient architecture. As side effect of the optimization effort, the video encoder is now defined by an efficient and "cleaned" code: a necessary starting point for a functional model during the HW development.

The selected architecture is a video pipeline implementing (macro)block buffers as queues between the different localized processing blocks. Higher layers of the hierarchy are used as interface memories. The ME engine is realized in hardware and demonstrates the applicability of the memory centric optimization concepts at the lowest design level. The system is mapped on an FPGA as this device offers the possibility to implement an optimized architecture including a customized memory hierarchy. The ME kernel supports up to 30 CIF f/s with only three parallel PEs and minimized data input rates.

REFERENCES

[1] A. Chimienti *et al.*, "VLSI architecture for a low-power video codec system," *Microelectron. J.*, vol. 33, no. 5, pp. 417–427, May 2002.

[2] M. Takahashi *et al.*, "A 60 MHz 240 mW MPEG-4 video-phone LSI with 16 Mb embedded DRAM," *IEEE J. Solid-State Circuits*, vol. 35, no. 11, Nov. 2000.

[3] H. Arakida *et al.*, "A 160 mW, 80 nA standby, MPEG-4 audiovisual LSI with 16 Mb embedded DRAM and a 5GOPS adaptive post filter," in *Proc. IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers*, vol. 1, 2003, pp. 42–476.

[4] J. H. Park *et al.*, "MPEG-4 video codec on an ARM core and AMBA," in *Proc. Workshop Exhibition on MPEG-4*, 2001, pp. 95–98.

[5] P. Pirsch and H.-L. Stolberg, "VLSI implementations of image and video multimedia processing systems," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8, no. 11, pp. 878–891, Nov. 1998.

[6] A. Dasu and S. Panchanathan, "A survey of media processing approaches," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, no. 8, pp. 633–645, Aug. 2002.

[7] M. Ravasi and M. Mattavelli, "High-level algorithmic complexity evaluation for system design," *J. Syst. Architec.*, vol. 48, pp. 403–427, May 2003.

[8] S. Bauer *et al.*, "The MPEG-4 multimedia coding standard: Algorithms, architectures, and applications," *J. VLSI Signal Process.*, vol. 23, no. 1, pp. 7–26, Oct. 1999.

[9] F. Catthoor *et al.*, *Custom Memory Management Methodology*. Norwell, MA: Kluwer, 1998.

[10] L. Nachtergaele *et al.*, "System-level power optimization of video codecs on embedded cores: A systematic approach," *J. VLSI Signal Process.*, vol. 18, no. 2, pp. 89–111, Feb. 1998.

[11] M. J. Irwin, M. Kandemir, N. Vijaykrishnan, and A. Sivasubramaniam, "A holistic approach to system-level energy optimization," in *Proc. Int. Workshop Power and Timing Modeling, Optimization, and Simulation*, 2000, pp. 88–107.

[12] *Information Technology—Generic Coding of Audio-Visual Objects—Part 2: Visual Amendment 1: Visual Extensions*, ISO/IEC JTC1/SC29/WG11 14 496-2N3056, Dec. 1999.

[13] V. Bhaskaran and K. Konstantinides, *Image and Video Compression Standards, Algorithms, and Architectures*. Norwell, MA: Kluwer, 1997.

[14] T. H. Meng *et al.*, "Portable video-on-demand in wireless communication," *Proc. IEEE*, vol. 83, no. 4, pp. 659–680, Apr. 1995.

[15] J. L. Hennessy and D. A. Patterson, *Computer Architecture, a Quantitative Approach*. San Francisco, CA: Morgan Kaufmann, 2003.

[16] J. Bormans *et al.*, "Integrating system-level low power methodologies into a real-life design flow," in *Proc. IEEE Workshop Power and Timing Modeling, Optimization, and Simulation*, 1999, pp. 19–28.

[17] P. R. Panda *et al.*, "Data and memory optimization techniques for embedded systems," *ACM Trans. Design Automat. Electron. Syst.*, vol. 6, no. 2, pp. 149–206, Apr. 2001.

[18] P. R. Panda *et al.*, "Data memory organization and optimizations in application-specific systems," *IEEE Design Test Comput.*, vol. 18, no. 2, pp. 56–68, Jun. 2001.

[19] A. Halambi *et al.*, "EXPRESSION: A language for architecture exploration through compiler/simulator retargetability," in *Proc. 2nd ACM/IEEE Design and Test in Europe Conf. (DATE)*, Munich, Germany, Mar. 1999, pp. 485–490.

[20] F. Catthoor, K. Danckaert, S. Wuytack, and N. Dutt, "Code transformations for data transfer and storage exploration preprocessing in multimedia processors," *IEEE Design Test Comput.*, vol. 18, no. 2, pp. 70–82, Jun. 2001.

[21] L. Benini and G. De Micheli, "System-level power optimization techniques and tools," *ACM Trans. Design Automation Embedded Syst.*, vol. 5, no. 2, pp. 115–192, Apr. 2000.

[22] *Information Technology—Generic Coding of Audio-Visual Objects—Part 5 Amd1: Simulation Software*, ISO/IEC JTC1/SC29WG11 14 496-5, N3508, Jul. 2000.

[23] M. Horowitz, A. Joch, F. Kossentini, and A. Hallapuro, "H.264/AVC baseline profile decoder complexity analysis," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 704–716, Jul. 2003.

[24] P. Kuhn and W. Stechele, "Complexity analysis of the emerging MPEG-4 standard as a basis for VLSI implementation," in *Proc. Int. Soc. Optical Engineering—SPIE*, vol. 3309, San Jose, CA, Jan. 1998, pp. 498–509.

[25] T. Chiang and Y.-Q. Zhang, "A new rate control scheme using quadratic rate distortion model," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 7, no. 1, pp. 246–250, Feb. 1997.

[26] C. De Vleeschouwer, "Model-based rate control implementation for low-power video communication devices," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 12, pp. 1187–1194, Dec. 2003.

[27] C. De Vleeschouwer and T. Nilsson, "Motion estimation for low power video devices," in *Proc. IEEE Int. Conf. Image Processing*, Thessaloniki, Greece, Oct. 2001, pp. 953–956.

[28] C. De Vleeschouwer, T. Nilsson, K. Denolf, and J. Bormans, "Algorithmic and arcitectural co-design of a motion-estimation engine for low-power video devices," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, no. 12, pp. 1093–1105, Dec. 2002.

[29] *MPEG-4 Video Verification Model Version 17.0*, Jul. 2000.

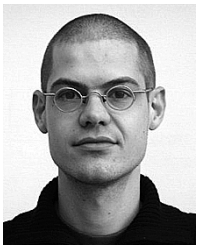
[30] I.-M. Pao and M.-T. Sun, "Modeling DCT coefficients for fast video encoding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 9, no. 6, pp. 608–616, Jun. 1999.

[31] E. Brockmeyer *et al.*, "Low power memory storage and transfer organization for the MPEG-4 full pel motion estimation on a multimedia processor," *IEEE Trans. Multimedia*, vol. 1, no. 2, pp. 202–216, Jun. 1999.

[32] J.-C. Tuan, T.-S. Chang, and C.-W. Jen, "On the data reuse and memory bandwidth analysis for full-search block matching VLSI architecture," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, no. 1, pp. 61–72, Jan. 2002.

[33] L. De Vos and M. Stegherr, "Parameterizable VLIS architectures for the full-search block matching algorithm," *IEEE Trans. Circuits Syst.*, vol. 36, no. 10, pp. 1309–1316, Oct. 1989.

[34] N. Roma and L. Sousa, "Efficient and configurable full-search block-matching processors," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, no. 12, pp. 1160–1167, Dec. 2002.



Kristof Denolf received the Industrial Engineer degree in electronics from the Katholieke Hogeschool, Brugge-Oostende, Belgium, in 1998, and the M.Sc. degree in electronic system design from Leeds Metropolitan University, Leeds, U.K., in 2000.

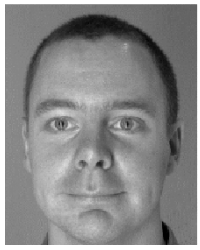
He joined the Multimedia Image Compression Systems (MICS) Group at the Interuniversity Micro Electronics Centre (IMEC), Leuven, Belgium, in 1998. His main research interests are the cost efficient design of advanced video processing systems and the end-to-end quality of experience.

Christophe De Vleeschouwer was born in Namur, Belgium, in 1972. He received the degree in electrical engineering and the Ph.D. degree from the Université Catholique de Louvain (UCL), Louvain-la-Neuve, Belgium, in 1995 and 1999, respectively.

From September 1999 to November 2000, he was a Research Engineer with the IMEC Multimedia Information Compression Systems Group. He is now with the Communications and Remote Sensing Laboratory, UCL, and is funded by the Belgian National Fund for Scientific Research. From August 2001 to June 2002, he was a Visiting Research Fellow at University of California at Berkeley. His main interests concern video and image processing for communication and networking applications, and nonlinear signal expansion techniques and their use for signal analysis and signal interpretation.

Robert Turney was s born in Chilton, Wisconsin, in 1962. He received the B.S., M.S., and Ph.D. degrees in electrical engineering from the University of Wisconsin, Milwaukee, in 1989, 1992, and 2005, respectively.

He has taught courses in DSP and VLSI at the Milwaukee School of Engineering and University of Wisconsin from 1989 to 1999 holding Lecturer and Research Associate positions. From 1994 to 1997, he worked at Camtronics Medical Systems as a DSP Systems Engineer designing medical imaging products. In 1997, he joined Xilinx, San Jose, CA, where he currently is a Senior Staff Researcher in Xilinx Research Labs specializing in multimedia, video, and image processing algorithms. He has been involved in video processing since 1992 in projects such as real-time lossless compression, image enhancement, and noise reduction for real-time image acquisition. Recent activities are related to implementation of new compression standards in the ISO/IEC's JPEG and MPEG standardization committees, namely JPEG2000 and MPEG-4.



Gauthier Lafruit was a Research Scientist with the Belgian National Foundation for Scientific Research from 1989 to 1994, being mainly active in the area of wavelet image compression implementations.

Subsequently, he was a Research Assistant with the Vrije Universiteit Brussel (VUB), Brussels, Belgium. In 1996, he became the recipient of the Scientific Barco award and joined Interuniversity Micro-Electronics Centre (IMEC), Leuven, Belgium, where he was involved as a Senior Scientist with the design of low-power VLSI for combined JPEG/wavelet compression engines. He is currently the Principal Scientist in the Multimedia Image Compression Systems Group with IMEC. His main interests include progressive transmission in still image, video, and 3-D object coding, as well as scalability and resource monitoring for advanced, scalable video, and 3-D coding applications. He is the author or coauthor of around 60 scientific publications, 50 MPEG standardization contributions, five patents (applications) and has participated (and been appointed as evaluator) in several national and international projects (proposals).

Jan Bormans was a researcher at the Information Retrieval and Interpretation Sciences laboratory of the Vrije Universiteit Brussel (VUB), Brussels, Belgium, in 1992 and 1993.

In 1994, he joined the VLSI Systems and Design Methodologies (VSDM), Interuniversity Micro Electronics Centre (IMEC), Leuven, Belgium. Since 1996, he has been heading IMEC's Multimedia Image Compression Systems Group. This group focuses on the efficient design and implementation of embedded systems for advanced multimedia applications. He is the Belgian Head of Delegation for ISO/IEC's MPEG and SC29 standardization committees. He is also MPEG-21 Requirements Editor and Chairman of the MPEG Liaison Group.