Journal of VLSI Signal Processing Systems, 15, 5-20 (1997)

MPEG2 Video Encoding in Consumer Electronics

R.P. KLEIHORST, A. VAN DER WERF, W.H.A. BRÜLS,
W.F.J. VERHAEGH AND E. WATERLANDER
Philips Research Laboratories, Prof. Holstlaan 4, 5656 AA Eindhoven, The Netherlands

Received April 16, 1997;

Abstract. Only very recently, single-chip MPEG2 video encoders are being reported. They are a result of additional interest in encoding in consumer products, apart from broadcast encoding, where a video encoder contains several expensive chips. Only single-chip solutions are cost-effective enough to enable digital recording for the consumer. The professional broadcast encoders are expensive because they use the full MPEG toolkit to guarantee good image quality, at the lowest possible bit-rate. Some MPEG tools are costly in hardware and these are therefore not feasible in single-chip solutions. This results in higher bit-rates, that can be accepted because of the available channel and storage capacity of the latest consumer storage media, harddisk, digital tape (D-VHS) and Digital Versatile Disk (DVD). A consumer product is I.McIC, a single-chip MPEG2 video encoder. It operates in ML@SP mode which can be decoded by all MPEG2 decoders. The IC is highly-integrated, as it contains motion-estimation and compensation, adaptive temporal noise filtering and buffer/bit-rate control. The high-throughput functions of the MPEG algorithm are mapped onto pipelined dedicated hardware, whereas the remaining functions are processed by an application-specific instruction-set processor. Software for this processor can be downloaded, in order to suit the IC for different applications and operating conditions. The IC consists of several communicating processors which were designed using high-level synthesis tools, PHIDEO and DSP Station™.

Keywords: MPEG2, encoding, consumer recording, high-level synthesis, hardware-software codesign

1. Introduction

Today, MPEG2 is an important standard for video compression [1, 2, 3]. Encoding is mainly done by distributors and publishers using professional equipment that is yet too expensive for the consumer market [4]. Broadcast encoders typically use bit-rates ranging from 1.5 to 8 Mbit/s. A minimal bit-rate for an appreciable image quality is desired because of the high-cost of channel rate rented from a channel provider. Therefore, the performance of the broadcast encoders is more important than their high prices. Use of the full coding toolkit, offered by the MPEG2 video coding standard, enables to achieve those goals. Some paths of the coding algorithm are performed simultaneously with different parameter settings. Afterwards, a founded choice can be made on the bit-rate/picture-quality performance, and the better is forwarded. Current hardware technology cannot efficiently implement this type of encoding on a single die. Therefore, professional equipment uses several dedicated chips and high-processing power for the encoding function. In addition, large amounts of high-speed RAM are present in these systems.

It is a challenge to provide the functionality of MPEG encoding at a reasonable price: a

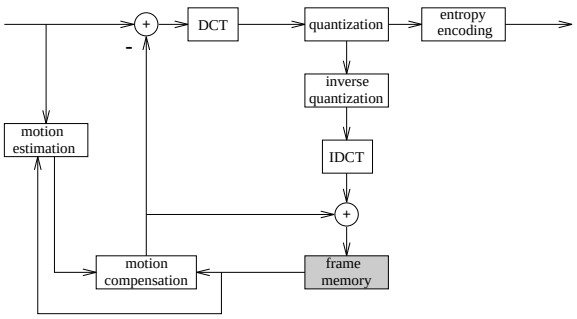

Fig. 1. An overview of MPEG's signal path

single-chip solution. An application for affordable MPEG2 video coding is consumer storage. Here, bandwidth is not as expensive as the broadcaster's channel rates, storage is performed on bulk media such as magnetic disc and tape (DVHS), and optical disc (recordable DVD) and bit-rates from 5 to 15 Mbit/s for good picture quality are tolerable. To achieve good results at these bit-rates, only some significant tools from the MPEG box are necessary and a careful selection can be made as to the benefit of hardware costs.

A hardware versus software implementation of the MPEG algorithm is a key question. MPEG consists on one hand of fixed signal-processing items that demand a high-throughput and are best implemented in hardware; on the other hand there are also flexible settings for different coding environments and applications. A balancing act for hardware versus software solutions has to be performed. The hardware side enables the high-throughput factor, the software side enables a wide range of applications.

The different "unit-blocks of data" within the MPEG algorithm have a noticeable impact on the architecture of encoders. At the entry level, video enters the IC in a pixel format on field basis. The MPEG operations are performed on data sub-blocks (macro-blocks) selected from the frames. Finally, MPEG's result are bit-streams. This means that several types of data-unit are present in the algorithm at different rates. This is often a reason to subdivide the algorithm in several dedicated chips as was done in [5], where separate ICs are used for motion-estimation, pixel processing and the core MPEG coding facility. A solution on a single chip is possible by analyzing inter-processor communication with different data-rates. Possible solutions are the use of FIFOs and communication by shared (external) RAMs.

Typical broadcasters will encode clean material, free from noise and other artefacts. On the other hand, noisy image sequences can be expected in a consumer environment. For instance, terrestrial broadcasts or material from analogue tapes are often noisy. In that case, a large portion of the bitstream will be spent to the coding of noise. To lower the demands for bit-rate and to enhance the image quality, noise reduction should be part of an encoder for the consumer market.

A result of these contemplations is I.McIC, a single-chip MPEG2 video encoder meant for the consumer market that will be dealt with here. Recently, other single-chip encoders have been reported, such as Sony's single-chip MPEG2 codec incorporating 6 high-power DSPs processing macroblocks [6]. Also reported was NEC's low-power MPEG2 encoder employing an adaptive search on-chip motion-estimation algorithm [7]. Additional single-chip solutions are reported by IBM (intra-only) [8], C-cube and ADI-Apogee [9], but not easily found in scientific literature.

The remainder of this paper is as follows: Section 2 discusses MPEG in a nutshell; Section 3 discusses MPEG-compliant derivations that make cost-effective integration possible for consumer coding; Section 4 discusses the architecture arrived on; Section 5 describes the functionality and design of the Compressor; Section 6 describes the use of the PHIDEO tool; Section 7 describes the ASIP; Section 8 describes the verification and simulation of the design; Section 9 discusses the characteristics of the resultant IC and reaches the conclusions. Acknowledgments follow in Section 10.

2. MPEG coding principles

MPEG2 is an image-sequence compression standard. Part of the compression is lossy, part is lossless. An important data-unit within MPEG is the macroblock, which is a composition of a 16×16 pixels luminance (Y) block and two 8×8 chrominance blocks. This is conform the 4:2:0 data format which means that the colour information is subsampled in both directions by a factor of two in relation to the luminance. In most parts of the MPEG process, this composition is handled as six

blocks of 8×8 data values, which are referred to as "dct-blocks".

First, the dct-blocks are transformed to the discrete cosine domain by a 2-dimensional DCT. The resulting coefficients are practically uncorrelated and can be quantized separately and independently with the human visual system in mind. The quantization strength effectively balances the image-quality versus bit-rate and is also used as such by the bit-rate/buffer controller. After this lossy coding part the blocks pass an entropy-coding (lossless) part comprising a Zig-Zag scan (ZZ), a DC predictor, a Run-Length Encoder (RLE) and a Variable Length Encoder (VLE) that generates a set of codewords for the underlying macroblock. In contrast with JPEG, MPEG's VLE codeword tables are fixed. The macroblock is decoded by following the described path in reverse order. This decoding path consists of inverse quantization, inverse DCT and inverse compensation. Figure 1, which is redrawn from [3], shows MPEG's signal path.

Subsequent images of a typical sequence are often quite equal, therefore the encoding switches to a differential mode where only the difference image is coded following the same path as described above. To decrease the difference image even further, motion estimation and compensation are used. This includes finding for each macroblock the most matching equivalent from the previous image and using this as a prediction for the current block. The remaining difference signal is then coded, accompanied by a displacement vector describing the relative position of the prediction in the previous image. The regularly coded image is referred to as an "Intra" coded (I) image, whereas the differentially coded image is referred to as an inter-coded or Predicted (P) image. To further reduce the bit-rate, also Bi-directionally (B) predicted images are possible in MPEG. Then, the contents of an image are predicted from previous and future images. The order of the images is reshuffled in the encoder memory to enable causal prediction at the decoder site. It may be clear that this re-shuffling property demands the availability of large RAMs in the encoder.

Encoding with I, P, and B frames at max 720×576 resolution with 30 frames/s, is referred to as "Main Profile At Main Level" (MP@ML) coding. Encoding with only I and P frames at the same resolution is called "Simple Profile At Main Level" (SP@ML) coding.

3. MPEG compliant implementation for I.McIC

The MPEG2 video standard describes how the bit-stream is decoded to video signals. It does not dictate the way that the bit-stream was created. This has the effect that the standard more or less fixes the decoder but not the encoder. An MPEG decoder has to be able to decode any valid bit-stream, whereas an MPEG encoder only has to produce a valid bit-stream. It doesn't need the full toolkit of MPEG to arrive at this valid bit-stream. In effect, a minimal system could consist of a DCT, quantizer and entropy coding, see Figure 2. One would arrive at a valid MPEG video-elementary stream by inserting header information and performing some (limited) buffer/bit-rate control. From this example, it can be seen that, contrary to popular belief, an MPEG encoder can be less complex than an MPEG decoder. In effect, there are single-chip encoders available on the market that work in this intra-only mode [8].

In order to arrive at an encoder that provides better image quality than the intra-only encoder described above at a given bit-rate, a vast set of tools is available from MPEG's toolbox. Among the well-known are the use of predictive coding (P-frames), bidirectional coding (B-frames) and the bit-rate control mechanism. The use of these tools has an impact on the architecture and cost of the encoder. The application area of the encoder plays a crucial role. I.McIC is primarily meant for the consumer storage market, where the bandwidth is not as expensive as for broadcast encoders that have to rent (part of) a transmission channel. Typical bandwidths for consumer storage applications are 5-15 MBit/s, the higher bandwidth corresponding with the highest image quality.

One of the costly tools within MPEG is the use of motion compensation for P or B frames. It is also a mechanism that is highly desirable for good performance. For an encoder to use motion compensation, a frame memory and a motion estimation function must be present. Motion estimation can be very expensive, and is often performed

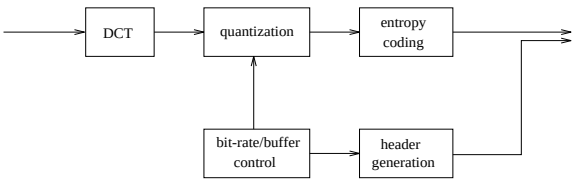

Fig. 2. A minimal MPEG encoder

on separate chips [6, 8]. A breakthrough however is the Recursive-Search (RS) block matching algorithm [10]. Unlike the more expensive full-search blockmatchers that are matching all possible displacements within a search region, the RS blockmatcher only investigates a very limited number of possible displacements. By carefully choosing the candidate displacements, a high performance can be achieved, approaching true motion. The algorithm providing the candidates is based on the 3D recursive search (3DRS) algorithm described in [10]. As only a few matches have to be made, the associated hardware for motion estimation remains relatively small. This motion estimator was proven in an IC for consumer applications in [11, 12]. It is shown in [4] that, despite the fact that the 3DRS ME does not check all possible displacements and therefore will not guarantee to find the best possible fit, the bit-rate for a given quality is not significantly different from full-search block-matching.

I.McIC can share a PCB with an MPEG decoder which demands 16Mbit of DRAM. This RAM can be multiplexed to the encoder when the decoder is inactive. The combination of the 3DRS and available RAM for frame storage gives the opportunity for predictive coding. The amount of RAM is not sufficient for bidirectional coding. More DRAM and an expanded 3DRS motion estimator could enable B frames. However, from experiments described in [4] it shows that the use of P frames indeed gives a large coding benefit, but additional B frames will only marginally decrease the bit-rate. For good quality images in the range of 32 to 37 PSNR, the difference in bit-rates is in the order of 5–20%. Figure 3 shows the bitcosts per individual frame of two coding profiles: IPPPPPPPPPPP and IBBPBBPBBPBB order on the "flower_iso" sequence. After adding these frame bitcosts for a fixed quality the total costs for each profile are found. It appears that at 34 dB PSNR, the loss in bit-rate is 12%.

It is also shown in [4] that this loss in coding efficiency by not using B frames is minimized by the use of a noise-filter as a preprocessing step. Noise reduction is done by a recursive motion-compensated spatio-temporal noise-filter [13]. The noise reducer is smartly injected in the coding loop itself, utilizing the on-chip motion estimator.

4. Architecture

The signal path in I.McIC can be roughly partitioned into an input, compression, and output part (see Figure 4). Each of these parts is a process which communicates with other processes via buffers in a dynamic data-flow fashion [14]. These buffers have a fixed size and consequently they may neither overflow nor underflow. The input process receives a video signal from an independently operating video source and, therefore, dictates the throughput of the complete encoder. Consequently, the input buffer must be emptied faster than it is filled such that no overflow can occur. To prevent underflow for this buffer, the compression processing must be halted now and

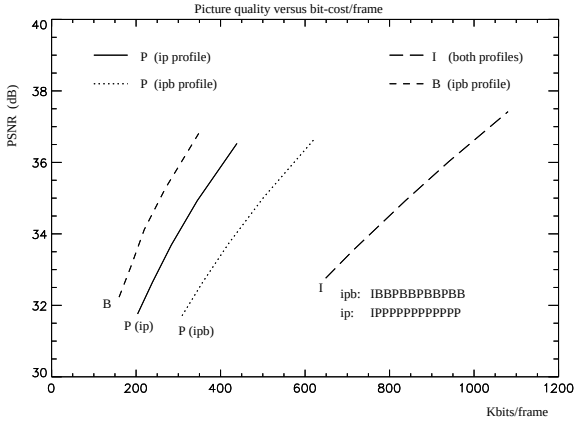

Fig. 3. Comparison IPB versus IP profile

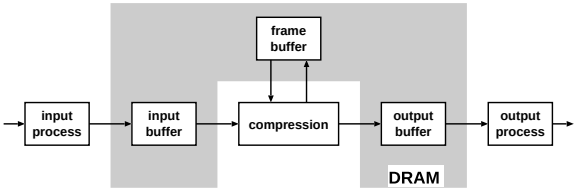

Fig. 4. Signal path of I.McIC and use of external DRAM

then to have the buffer filled. For the output buffer under- and overflow is circumvented by controlling the number of code words produced. To this end, for each macroblock a different quantizer step size can be used which influences the compression ratio. Furthermore, a frame buffer is required to store the previous frame for inter-coding of pictures using motion estimation.

The sizes of the input, output, and frame buffer are quite large. The input buffer must be larger than one field since the processing is frame based and input signals arrive field by field. The storage of one field of chroma and luminance samples in a 4:2:0 format requires 2.4 Mbit. Consequently, the storage of the reference frame requires 4.8 Mbit. The size of the output buffer is determined by the MPEG2 specification and equals about 2 Mbit. All these buffers are stored in 16 Mbit of external DRAM, configured as 4 devices of 16 bit words and organized per device in 512 pages of 512 words. The data width is 64 bits and the address width is 18 bits (9 for the column and 9 for the row). I.McIC can be part of a system in which it time-shares this memory with an MPEG2 decoder.

I.McIC's architecture consist of a control and video processing part. The processing is partitioned into three parts, i.e., line-based processing (Frontend), macroblock-based processing, and bitstream-based processing (Backend). These parts exchange data via a memory interface module with the external memory where the large buffers are located. The accesses from the various parts of I.McIC to the external memory are handled by the interface by devoting time slots in a round-robin fashion. The control part consists of three parts, i.e., a global controller, an I^2C interface, and an Application Specific Instruction-Set Processor (ASIP). Figure 5 shows I.McIC's architecture.

The Frontend receives a digitized video signal with Y, U, and V components and separate synchronization signals, and processes them in ordinary line-based fashion. It performs 4:2:2 to 4:2:0 format conversion by vertical filtering (6 taps with fixed coefficients) and subsampling of the U and V components. Furthermore, it performs horizontal filtering (7 taps zero-phase filter with programmable coefficients) for the Y, U, and V components. The horizontal filter can be used to preprocess difficult-to-code video scenes by reducing unimportant image features that would otherwise dominate the coding effort. Reducing their impact will result in an overall better image quality. Also, the prefilters are used as anti-aliasing filters when the input image is subsampled to SIF format for low bit-rate coding.

The Backend performs byte- and bit-wise processing for packing and bit stuffing, and merges the variable-length coded DCT coefficients generated by the Compressor with the MPEG headers generated by the ASIP. In addition, the Backend has features to fill small remaining coding gaps, by stuffing, when a fixed-rate coding scheme is used.

The macroblock-wise processing is performed in a pipelined fashion by the Compressor and the ASIP. The ASIP is a microcode-programmable processor that runs a program for every macroblock. The ASIP's tasks are application specific such as the compression profiles and buffer-regulation algorithm. In contrast, the Compressor serves as a co-processor performing tasks that are best implemented in hardware, mostly because they are fixed MPEG functions such as cosine transforms.

5. Compressor

The Compressor entails the fixed signal-processing activities on the macroblock-level. Among those are motion estimation (ME) and compensation (MC), frame-field conversion (FF), noise filtering (NF), discrete cosine transform (DCT/IDCT) and quantization (Q/IQ). Most of the actions here are

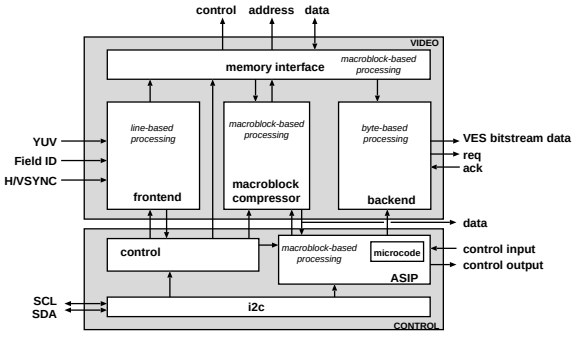

Fig. 5. Top-level architecture of I.McIC

mandatory MPEG operations. An encoder capable of inter-frame coding actually contains a decoding part next to the obvious coding part (see Figure 6). It has to mimic the functionality of the reference MPEG decoder to arrive at the same reference image. This part of the codec has to be fully in-spec with the MPEG standard, whereas the encoding part only has to provide a de-codable signal at an appreciable quality. The functionality of the lossless part of the encoder, the zigzag scan (ZZ) and run-length and variable length encoding (RLEVLE) part is also exactly prescribed in the MPEG2 standard. Four blocks within the Compressor with interesting approaches are the motion estimator, the DCT, IDCT and the noise-filter. These blocks will be described in greater detail below.

One of the most significant parts of the Compressor is the motion estimator. It matches macroblocks from the previous frame, residing in the main memory to the current macroblock. In order to minimize the number of accesses to the external memory, the motion estimator has a cache which contains pixels of a window in the decoded previous frame. The window allows for motion vectors with a vertical range of -6 to +6 pixels and a horizontal range of -8 to +7 pixels per frame. This falls within MPEG's `f_mode=1` category. For each macroblock a small number of candidate motion vectors at half-pel accuracy is considered, returning the vector with the minimum absolute difference. The candidate vectors are generated by the ASIP according to the 3DRS block-matching algorithm. The basics of this algorithm are that, due to hardware/bandwidth constraints only a limited number of candidates can be matched; in our case 5. As a result of the recursion in the algorithm, only candidates with a high-possibility of correctness are generated.

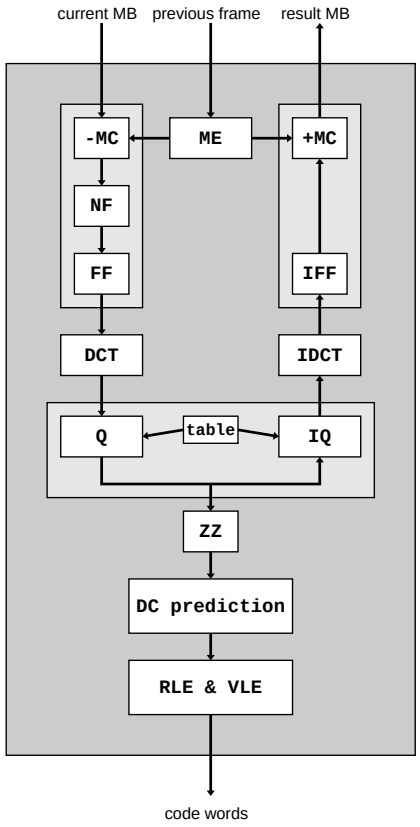

Fig. 6. Architecture of the Compressor

As discussed earlier, in the typical applications of I.McIC, noisy sequences can be expected as source material. To this order, an adaptive noise filter is employed [15]. The structure of the noise filter is shown in Figure 7. For simplicity reasons, the signals are represented as 1-dimensional data. Here $g(k)$ is the observed, noisy, signal; $\hat{f}_{MC}(k-1)$ is the previously filtered (and coded-decoded) signal which is retrieved from the main loop memory. Note that the subscript MC denotes that this signal is motion compensated to lie along the motion trajectory of $g(k)$. The result of the filtering action is $\hat{f}(k)$, this is inserted into the coding chain instead of the noisy $g(k)$. Within inter-coded macroblocks, the signal $\hat{f}(k)-\hat{f}_{MC}(k-1)$ is forwarded.

The Kalman-gain multiplier $0 \leq C \leq 1$ is controlled to adapt the filter to the situation at hand. Setting $C = 1$, forwards the (noisy) observation and no filtering action takes place. If $C = 0$, then only the prediction is forwarded, while for intermediate values of C noise-filtering is achieved [16, 17]. Except from globally controlling this value, C is also directly controlled by the result of the motion compensation. This is done to avoid blurring in situations where MPEG's (translational-) motion model is insufficient, such as in occlusion areas [18] and with extreme displacements out of the scope of the motion caches. Reversely, the adaptivity also increases filtering effort if a good motion-compensation is detected.

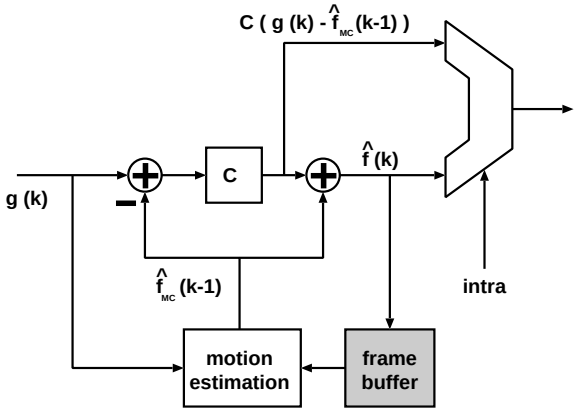

Fig. 7. Filter structure

The implemented IDCT and DCT are time-recursive implementations [19, 20] instead of the often used rotation (or butterfly) algorithms [21]. The recursive algorithm, although more of a challenge for hardware timing, is easier to control for accuracy and the accompanying word widths [22]. These structures represent 2^{nd} order recursive digital filters with fixed coefficients. The transformed coefficients are found from the last output after feeding the input data through the structure from Figure 8.

The structure contains 2 constants-multipliers that are effectively mapped on the same structure in VLSI. The value of the constants dictates the exact coefficient which is calculated. I.McIC utilizes a row-column approach for establishing the 2D (I)DCT of the 8×8 dct_block. With the row-column approach, data-series of 8 values are fed through the structure. Only 8 of these structures with fixed coefficients are then needed to accomplish a 2D (I)DCT by time-folding between (I)DCT across the rows and (I)DCT along the columns.

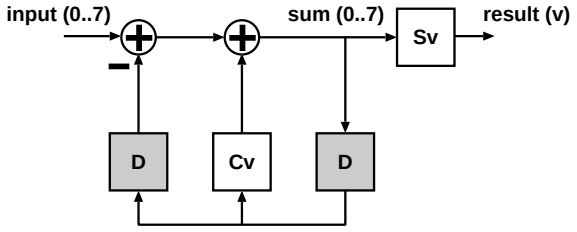

Fig. 8. Recursive (I)DCT stage

6. The use of PHIDEO

The macroblock compressor has been designed using PHIDEO [23, 24], which is a design method and tool set for high-throughput applications. PHIDEO translates a high-level specification of an algorithm into an architecture consisting of processing units communicating via distributed memories, associated address generators, and a controller. PHIDEO generates an RTL-VHDL description that can be synthesized using logic synthesis tools.

The input for PHIDEO is specified in PIF (PHIDEO Input Format), and typically contains many (nested) loops and multidimensional arrays. This entry level is an almost identical match with typical simulation languages such as FORTRAN or C. Therefore, after an (automatic) translation of the PIF description into a sequential simulation language, the design can be simulated at high speed on real image sequences. A sample of the PIF description of the IDCT from Figure 8 is shown in Figure 9, showing the initialization and the recursion of the eight 2^{nd} order recursive structures and the extraction of the final result of the IDCT across eight rows.

There are a number of major steps that have to be taken in order to get from an algorithmic description, which only specifies *what* functions have to be executed, towards an architecture, which specifies *how* they are executed. These steps are: scheduling, memory synthesis, address synthesis, and controller synthesis.

The main task during scheduling is to determine for each of the operations in the PIF description the time at which it has to be executed, and to assign it to a processing unit on which it has to be executed. For each of the operations we assume that it is specified on which type of processing unit it has to be executed, and we assume that these processing units have already been designed. During scheduling, we can therefore take into account the time shapes of the processing units, which describe when they expect input data and when they produce output data.

The goal during scheduling is to minimize the total area that is required by the design. Because of the application domain, not only processing units but also memories have a significant contribution to the total area. So both contributions

```
1.  // RECURSIVE IDCT ALONG THE ROWS OF "input"
2.  // each IDCT block perceives the same input data.
3.  (v : 0 .. 7) 8 :: // 8 times with period of 8
4.  begin
5.  // initialization
6.  sum[v][-1] = clear();
7.  sum[v][0] = input[0];
8.  // regular flow
9.  (w : 1 .. 7) 1 ::
10. begin
11. sum[v][w],result[v][w] = idct_pu(input[w],sum[v][w-1],sum[v][w-2]);
12. end;
13. // the last output can now be extracted.
14. sum[v][8],result[v][8] = idct_pu( ,sum[v][7],sum[v][6]);
15. output[v] = result[v][8];
16. end;
```

Fig. 9. The PIF description of the IDCT

have to be taken into account, and a trade-off has to be made. A special issue to take care of during scheduling is that operations are multidimensional periodically executed [25], which places extra demands on constraint checking and cost evaluation.

A schedule determines the clock cycles in which the processing units produce output data and when they require new input data, i.e., a schedule determines the required delays of the intermediate data. The data, also called variables, are stored in and retrieved from memories. The problem now is to design a configuration of memories and an interconnection network, and to assign the variables to the memories, such that there are no conflicting situations and that the required area is minimal. Conflicts occur when, for instance, two variables have to be stored simultaneously in a memory that has only one input port. In that case the conflict has to be resolved, e.g., by assigning the variables to different memories, or by adding hardware for delaying one of the write actions. During memory synthesis, various types of memories can be taken into consideration, e.g., dual-port and single-port memories.

Memory synthesis results in data schedules, which define for each memory the time when data is written into it, and when it is read out again. Next, we have to determine the addresses at which variables are stored, and address generators have to be synthesized to provide these addresses at the correct times. The main problem here is to determine which variables are to be stored at the same memory address, in order to reduce the memory size.

Finally, a controller has to be synthesized that provides the correct control signals. For instance, the processing units have to be started at the times determined by scheduling, read-enable and write-enable signals have to be generated for the memories, the control signals for the address generators have to be generated, and multiplexers have to be controlled in order to route the data to the correct places.

Figure 10 gives an overview of PHIDEO. As shown, the four subsequent steps in PHIDEO are performed by the tools Jason, Medea, Matchbox, and Paris, respectively. Figure 11 shows PHIDEO's target architecture.

7. ASIP

The ASIP (Application-Specific Instruction-set Processor) has various tasks, among which are header generation, coding control and selection of candidate motion vectors. All tasks that are

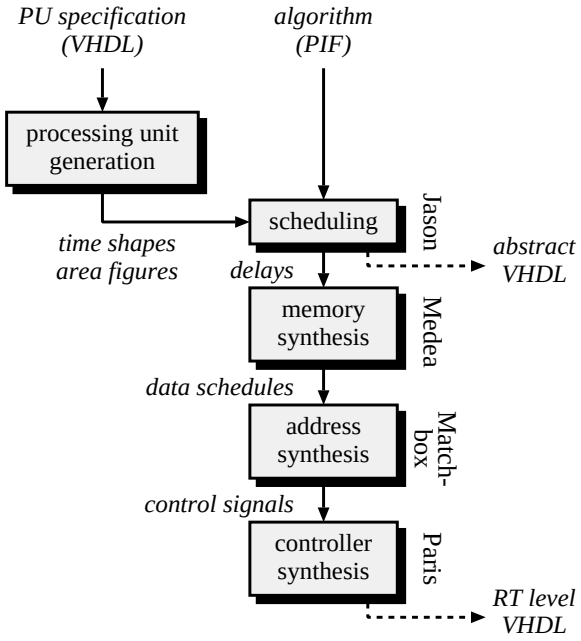

Fig. 10. An overview of the steps in PHIDEO

subject to change in future products for different applications are implemented by the ASIP.

Not only signal data, but also header data is present in an MPEG bitstream [1]. The ASIP generates all headers. For instance, the frame type (I/P) and image dimensions are sent to the decoder through the headers. Also de-quantization information, motion vectors etc. have to be supplied by headers.

The global and local decisions in the coding strategy, such as the coding profile, and the coding accuracy (quantizer level), are performed by the algorithm running on the ASIP. The buffer/bit-rate control part of this algorithm calculates and/or predicts the number of bits allowed for the remaining data and controls the coding accuracy by instructing the Compressor. Both variable bit-rate and fixed bit-rate are possible. Also a task of the ASIP is the generation of motion vector-candidates on basis of the 3DRS block-matcher.

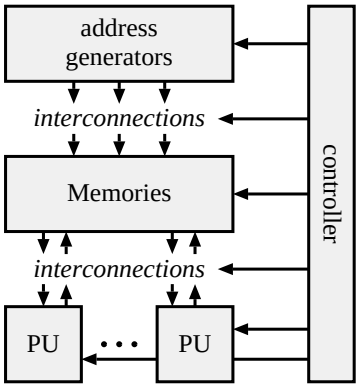

Fig. 11. The target architecture of PHIDEO

The ASIP is generated by the high-level synthesis Mistral2Pro part of the DSP Station^{TM} tool. The design of the ASIP started with the definition of a datapath by its instruction set. This is performed by crafting a representable model algorithm in DFL (Data-Flow Language), scheduling the operations, and generating instruction set and microcode. This sequence of tasks was repeated for different instruction sets until the model algorithm could be executed in a satisfiable number of clock cycles. Then the datapath and instruction set was fixed and the layout of the ASIP was designed. The result of the model algorithm are an instruction word of 162 bits and a datapath of 24 bits wide. Also, it contains the following execution units: an ALU, an address-computation unit, a multiplier/divider, 3 input channels, 6 output channels and 3 RAMs. The datapath contains 20 register files, which are located at inputs of execution units and addressed by the microcode. The microcode RAM can contain 2048 words and is downloadable via I^2C.

Afterwards, other algorithms can be developed using the now fixed datapath as a constraint. The resulting microcode can then be downloaded (by I^2C) in the on-chip microcode RAMs. It is of great importance that the model algorithm described earlier is a proper predictor of the future algorithms that will be downloaded in I.McIC. This model effectively sizes and freezes the datapath and the microcode memory. A positive property of this "reprogrammable" ASIP is that once the datapath generated by the model algorithm is future-proof and fixed, the development of further algorithms can be delayed until the IC itself is present. Now, real-time environments can be used for fine-tuning [26, 27].

8. Design Flow

High-level synthesis tools were used in the design, particularly PHIDEO for the Compressor and DSP

Station^{TM} for the ASIP (see Figure 12). The result of the high-level synthesis is an RTL description that, together with manually written RTL descriptions of other blocks, is mapped to standard cells by logic synthesis. The standard cell network is optimized for timing and area by retiming [28, 29].

For large designs simulations are often bothersome and restrictive because of simulator memory limits and/or simulator slowness. Of great importance for a well-defined behaviour is therefore attention to simulation and verification during the design. As part of the design-input is available in a high-level (synthesis) language the functionality can be simulated from this input. This input is translated in sequential VHDL, which can be simulated, if desired, in combination with the RTL-VHDL input part. Input to the simulation are real image sequences, output are MPEG2 bitstreams that can be fed into a software or hardware decoder. Note that these simulations are sequentially-run programs, and it is therefore easy to detect and correct functional errors in the input format. The simulation speed is also on par with implementations from C or Pascal.

After high-level and netlist synthesis, the netlists are again verified to the previously used sequential implementations of the considered functional block. Now, because of the implementation, a full sequence simulation will almost be out of scope due to slowness. However, verification of some strategically chosen test stimuli will cover most errors, as the functionality was proven during the sequential simulations. Note that only inconsistencies in the synthesis tools have to be detected. Most other tools were formally verified by analysing and pairing combinatorial expressions.

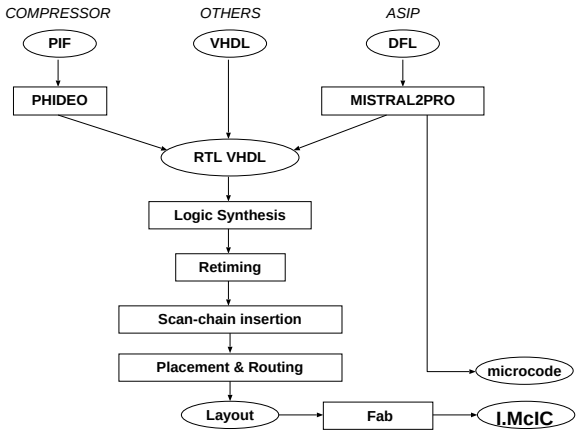

Fig. 12. Design flow of I.McIC

9. Results and Conclusions

In this paper we have described the design of I.McIC, a single-chip MPEG2 video encoder. I.McIC is applicable in particular for storage applications where higher bit-rates can be tolerated (5–15 Mb/s) as opposed to the bit-rates used for transmission (1.5–8 Mb/s). I.McIC operates in MPEG's `ML@SP` mode, which can be decoded by all MPEG2 decoders.

A printed circuit set-up which provides a full codec functionality incorporating I.McIC is shown in Figure 13. It can be seen that I.McIC time-shares 16 Mbit of DRAM with an MPEG2 video decoder, which is organized as 4 times 4 Mbit devices with 60 ns. access time. I.McIC can handle both 50 (PAL) and 60 (NTSC) Hz data formats, in CCIR 601 and SIF resolution. In order to interface smoothly to a CVBS video source, I.McIC uses a line-locked clock generated by an Analogue to Digital Converter (ADC) and running at 27 MHz. A micro-controller provides the I^{2}C input. I.McIC outputs the video Elementary Stream (ES) which is augmented with an audio bitstream to arrive at a Transport Stream (TS). For some circuit characteristics, see Table I.

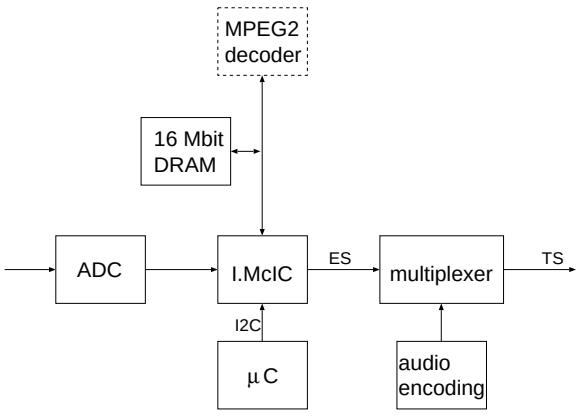

Fig. 13. PCB set-up for an MPEG2 codec functionality using I.McIC

Table 1. I.McIC Characteristics (see also Figure 14)

number of transistors	4.5×10^6
area	$198mm^2$ in 0.5μ
power consumption	2.1 Watts
clock frequency	27 MHz
package	QFP240
supply voltage	3.3 V (5 V for io)
number of embedded memories	42 (for 509Kbit)
total memory bandwidth	42 Gbit/s

Note that I.McIC is highly integrated, since it also contains on one chip motion estimation and compensation, MPEG2 header generation and buffer/bit-rate control (both fixed and variable). The functionality is implemented as dedicated hardware blocks and as embedded software running on an integrated application-specific instruction-set processor. This is a prime example of Hardware-Software Co-Design. Because of the high integration of functionality, I.McIC enables the introduction of MPEG2 encoding in the consumer market, particularly for storage applications.

The design of this IC with a limited number of people was made possible by the extensive use of high-level synthesis tools. The synthesis from a high-level description also enabled sequential verification of the design input on real video signals.

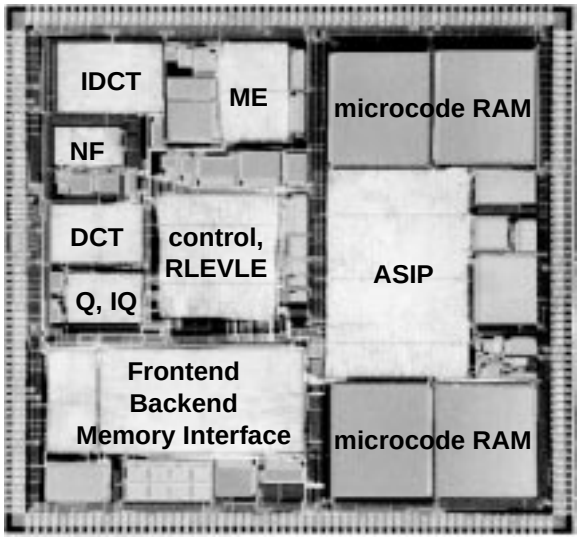

Fig. 14. Micrograph of I.McIC

Due to the flexibility offered by the ASIP, I.McIC itself is being used on a PCB as a real-time test environment for coding strategies in different applications.

Acknowledgements

I.McIC was designed at Philips Research Laboratories, Eindhoven, The Netherlands. This was made possible by the support of many of our colleagues that we wish to acknowledge here. Special acknowledgments are in place for Paul Lippens, Kees Goossens, Marino Strik, Math Verstraelen and Leo Sevat from Philips Research Laboratories and Thomas Friedrich of Philips Semiconductors Systems Laboratory, Hamburg, Germany.

References

1. B. Haskell, A. Puri, and A. Netravali, *Digital video: an introduction to MPEG-2*. Digital Multimedia Standards Series, Chapman and Hall, 1997.
2. J. Mitchell, W. Pennebaker, C. Fogg, and D. LeGall, eds., *MPEG video compression standard*. Digital Multimedia Standards Series, Chapman and Hall, 1997.
3. V. Bhaskaran and K. Konstantinidis, *Image and video compression standards; algorithms and architectures*. Kluwer academic publishers, 1995.
4. W. Bruls, A. van der Werf, R. Kleihorst, T. Friedrich, E. Salomons, and F. Jorritsma, "A single-chip MPEG2 encoder for consumer video storage applications," to appear in *Proc. ICCE '97*, 1997.
5. R. Pacalet *et al.*, "A real-time MPEG2 main profile, main level motion estimator chipset," in *ISSCC97 Digest of technical papers*, pp. 260–261, 1997.
6. E. Iwata *et al.*, "A 2.2GOPS video DSP with 2-RISC MIMD, 6 PE SIMD architecture for real-time MPEG2 video coding/decoding," in *ISSCC97 Digest of technical papers*, pp. 258–259, 1997.
7. M. Mizuno *et al.*, "A 1.5w single-chip MPEG2 MP@ML encoder with low-power motion estimation and clocking," in *ISSCC97 Digest of technical papers*, pp. 256–257, 1997.
8. IBM microelectronics, digital video products, "MPEG-2 digital video I-frame only encoder." http://www.chips.ibm.com/products/mpeg/mpegen.html, Mar. 1997.
9. Audio Digital-Imaging Apogee, "Apogee M-series chips." http://www.adi.net/apogee.html, Aug. 1996.
10. G. de Haan, P. Biezen, H. Huijgen, and O. Ojo, "True-motion estimation with 3-d recursive-search block matching," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 3, pp. 368–379, Oct. 1993.
11. B. de Loore, P. Lippens, P. Eeckhout, H. Huijgen, A. Löening, M. McSweeney, M. Verstraelen, B. Pham, G. de Haan, and J. Kettenis, "A video signal pro-

cessor for motion-compensated field-rate upconversion in consumer television," in *ISSCC digest of Technical papers*, Feb. 1996.
12. G. de Haan *et al.*, "IC for motion-compensated 100 Hz TV with natural-motion movie mode," *IEEE Transactions on consumer electronics*, vol. 42, pp. 165–173, May. 1996.
13. J. Brailean, R. Kleihorst, S. Efstratiadis, A. Katsaggelos, and R. Lagendijk, "Noise-reduction filters for dynamic image sequences: A review," *Proc. IEEE*, vol. 83, pp. 1270–1292, Sep. 1996.
14. E. Lee and T. Parks, "Data-flow networks," *Proc. of the IEEE*, vol. 83, pp. 773–801, May. 1995.
15. E. Dubois and S. Sabri, "Noise reduction in image sequences using motion compensation," *IEEE Transactions on Communication*, vol. 32, pp. 826–831, Jul. 1984.
16. R. McMann, "Digital noise reducer for encoded NTSC signals," *SMPTE Journal*, vol. 87, pp. 129–133, Mar. 1978.
17. T. Dennis, "A spatio/temporal filter for television picture noise reduction," in *Proc. Int. IEE Conf. on electronic image processing*, pp. 243–249, IEE, Jul. 1982.
18. R. Kleihorst, *Noise reduction of image sequences*. PhD thesis, Delft University of Technology, Delft, The Netherlands, Oct. 1994.
19. Z. Wang, A. Jullien, and W. Miller, "recursive algorithms for the forward and inverse discrete cosine transform with arbitrary length," *IEEE Signal Processing Letters*, vol. 1, pp. 101–102, Jul. 1994.
20. M. Aburdene, J. Zheng, and R. Kozick, "Computation of discrete cosine transform using Clenshaw's recurrence formula," *IEEE Signal Processing Letters*, vol. 2, pp. 155–156, Aug. 1995.
21. D. Slawecki and W. Li, "DCT/IDCT processor design for high-data rate image coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 2, pp. 135–146, Jun. 1992.
22. V. Srinivasan and K. Liu, "VLSI design of high-speed time-recursive 2-D DCT/IDCT processor for video applications," *IEEE Transactions on Circuits and Systems for video technology*, vol. 6, pp. 87–96, Feb. 1996.
23. P. Lippens, J. van Meerbergen, A. van der Werf, W. Verhaegh, B. McSweeney, J. Huisken, and O. McArdle, "PHIDEO: A silicon compiler for high speed algorithms," in *Proceedings of the European Conference on Design Automation*, pp. 436–441, 1991.
24. J. van Meerbergen, P. Lippens, W. Verhaegh, and A. van der Werf, "PHIDEO: High-level synthesis for high throughput applications," *Journal of VLSI signal processing*, vol. 9, pp. 89–104, 1995.
25. W. Verhaegh, *Multidimensional Periodic Scheduling*. PhD thesis, Eindhoven University of Technology, Eindhoven, the Netherlands, 1995.
26. M. Strik, "Efficient code generation for application domain specific processors," Chartered Designer's Thesis, Design automation Section, Eindhoven University of Technology, Eindhoven, The Netherlands, Nov. 1994.
27. B. Mesman, M. Strik, A. Timmer, J. van Meerbergen, and J. Jess, "Constraint analysis for code generation," submitted to *10th Int. Symp. on System Synthesis*, (Antwerpen, Belgium), Sep. 1997.
28. C. Leiserson and T. Parks, "Retiming synchronous circuitry," *Algorithmica*, pp. 5–35, 1991.
29. A. van der Werf, J. van Meerbergen, E. Aarts, W. Verhaegh, and P. Lippens, "Efficient timing constraint derivation for optimal retiming high-speed processing units," in *Proc. 7th Int. Symp. on High-Level Synthesis*, 1994.

Richard P. Kleihorst received the M.Sc. and PhD. degrees in Electrical Engineering from Delft University of Technology, the Netherlands, in 1989 and 1994, respectively. In 1989, he worked at the Philips Research Laboratories in Eindhoven, the Netherlands on fuzzy classification techniques for optical character recognition. From 1990 until 1994 he has worked as a Research Assistant, investigating Order Statistics for image processing in the Laboratory for Information Theory of Delft University of Technology, The Netherlands. In 1994 he joined the VLSI design group of Philips Research Laboratories, Eindhoven, The Netherlands. At present his interests include digital signal processing, with emphasis on compression and enhancement of images and image sequences, and architectures for VLSI.

Albert van der Werf received the M.Sc. degree (Honors) in electrical engineering for research in the area of telecommunications in 1987 from the University of Twente, The Netherlands. From 1987 to 1989 he followed a postgraduate course on the design

of VLSI circuits at the Graduate School Twente, The Netherlands. Since 1989 he has been employed by Philips Electronics at its research laboratories in Eindhoven, The Netherlands. His present areas of interest include the development of computer-aided design methodologies for the design of integrated circuits and the design of complex, heterogeneous systems on silicon. Currently, he is pursuing a PhD degree at the University of Technology Eindhoven, The Netherlands.

Fons Brüls was born in Geleen, the Netherlands, on August 15, 1956. He received the degree in electrical engineering at the Hogere Technische School Heerlen, Heerlen, The Netherlands, in 1978. In October 1978 he started working at the Philips Research Laboratories in Eindhoven, The Netherlands. Untill 1987 he was active in the field of integrated analog signal processing in bipolar processes. In 1987 he joined the Storage and Retrieval Group, working in the field of integrated digital video compression. He contributed to the realization of HDTV video storage based on D1 recorders and video compression. At present he focuses on video compression issues in storage systems.

Wim F.J. Verhaegh received the mathematical engineering degree with honors in 1990 from the Eindhoven University of Technology, The Netherlands. Since then, he is a member of the group Digital VLSI at Philips Research Laboratories in Eindhoven, where he is working on high-level synthesis of DSP systems for video applications, with the emphasis on scheduling problems and techniques. Based on this work, he received a Ph.D. degree in 1995 from the Eindhoven University of Technology, The Netherlands.

Erwin Waterlander received the B.Sc. degree, with honors, in electrical engineering from the Enschede Polytechnic (Hogeschool Enschede), the Netherlands, in 1993. His final exam was done at the Pattern Recognition Group of the faculty Applied Physics of the Delft University of Technology concerning the development of a digital signal processing program for educational purposes. He works at the Philips Research Laboratories Eindhoven since 1995 as research assistant in the group Digital VLSI were he joined the I.McIC project. The current interests are mainly layout design and the electrical issues of it.