Implementing an MPEG2 Video Decoder Based on the TMS320C80 MVP

Authors: F. Bonomini, F. De Marco-Zompit, G.A. Mian, A. Odorico, D. Palumbo

ESIEE, Paris

September 1996
SPRA332

TEXAS INSTRUMENTS

IMPORTANT NOTICE

Texas Instruments (TI) reserves the right to make changes to its products or to discontinue any semiconductor product or service without notice, and advises its customers to obtain the latest version of relevant information to verify, before placing orders, that the information being relied on is current.

TI warrants performance of its semiconductor products and related software to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

Certain application using semiconductor products may involve potential risks of death, personal injury, or severe property or environmental damage ("Critical Applications").

TI SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, INTENDED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT APPLICATIONS, DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS.

Inclusion of TI products in such applications is understood to be fully at the risk of the customer. Use of TI products in such applications requires the written approval of an appropriate TI officer. Questions concerning potential risk applications should be directed to TI through a local SC sales office.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards should be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance, customer product design, software performance, or infringement of patents or services described herein. Nor does TI warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used.

TRADEMARKS

TI is a trademark of Texas Instruments Incorporated.

Other brands and names are the property of their respective owners.

CONTACT INFORMATION

US TMS320 HOTLINE	(281) 274-2320
US TMS320 FAX	(281) 274-2324
US TMS320 BBS	(281) 274-2323
US TMS320 email	dsph@ti.com

Contents

Figures

Tables

Implementing an MPEG2 Video Decoder Based on the TMS320C80 MVP

Abstract

This application report presents the preliminary results obtained in the realization of the MPEG2 video decoder on the Texas Instruments (TI™) TM532OC80 MVP. We have addressed and solved the problems of bitstream scanning, variable-length code decoding, and inverse discrete cosine transform (DCT). The results obtained, integrated with preliminary work and available information about time requirements of image reconstruction for predicted and interpolated pictures, allow us to choose the architecture of the MPEG2 decoder. At present, it is possible to predict that a complete decoder will operate in real-time on one 50 MHz MVP for CCIR 601 4:2:0 sequences coded at bit-rates up to about 6 Mbit/s.

This document was part of the first European DSP Education and Research Conference that took place September 26 and 27, 1996 in Paris. For information on how TI encourages students from around the world to find innovative ways to use DSPs, see TI's World Wide Web site at www.ti.com.

Product Support on the World Wide Web

Our World Wide Web site at www.ti.com contains the most up to date product information, revisions, and additions. Users registering with TI&ME can build custom information pages and receive new product updates automatically via email.

Introduction

The goal of this project, carried out in the framework of the TI University Elite Program, is to implement an MPEG2 video decoder [1, 2] on the TMS32OC80 MVP via the TMS320C80 PC-based Software Development Board (SDB).

The real time requirement is the critical point of the project and needs a careful analysis of the decoding process in order to assign the different activities to the available resources (e.g., master and parallel processors, on-chip memory, etc.).

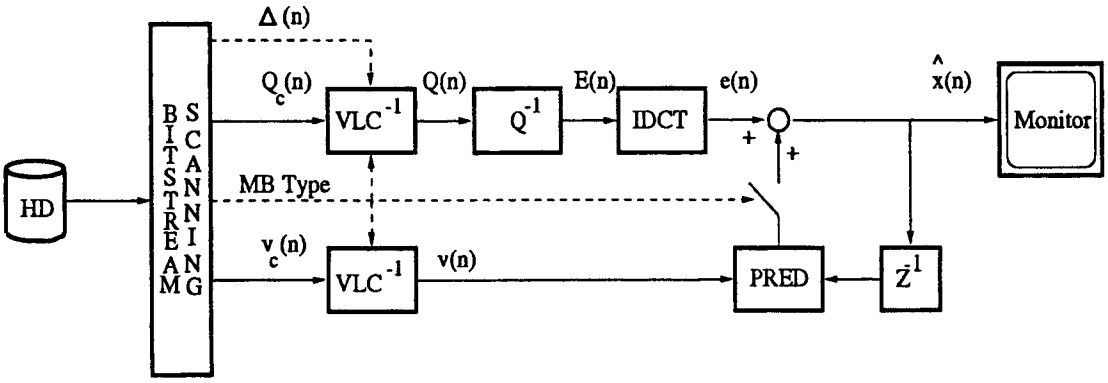
In the sequel, after giving a short description of the decoder, we present the solution chosen for scanning the bitstream, decoding the variable length codes and for the inverse discrete cosine transform. The timings associated with such activities and with the reconstruction of the predicted and interpolated pictures[3] will be presented.

Such data allows us to present an architecture that, in the case of CCIR 601 4:2:0 sequences, matches decoder activities with available resources for bit-rates up to about 6 Mbit/s on a 50 MHz MVP.

The Video Decoder

A schematic block diagram of the decoding process is given in Figure 1. For the sake of simplicity, only the reconstruction process associated with intracoded (I) and predicted (P) pictures is represented (z^{-1} denotes the delay between the reference and the predicted picture).

Figure 1. Decoding Activities



The process can be divided into the following main activities:

- Loading of the coded video bitstream from the external device (in our case the PC hard disk) into the C80 external memory. The coded video average bit-rate ranges from 2 to 16 Mbit/s for CCIR 601 sequences (video coded at 6 Mbit/s is subjectively perceived as fully equivalent to PAL).
- Scanning of the bitstream. The bitstream combines both fixed length, which are to be searched and interpreted first, and variable length codes (VLC).
- Demultiplexing into overhead information (such as motion information: $v_c(n)$, quantizer stepsize: $\Delta(n)$, macroblock type) and quantized DCT coefficients: $Q_C(n)$.
- Decoding of the variable length codes (VLC^{-1}). In particular the VLCs associated to DCT coefficients are to be decoded into pairs {*zero run length, level}* according to the macroblock type.
- Inverse quantization of the DCT coefficients (Q^{-1}), saturation and reorganization from the zigzag scanning order to the natural (row-column) order.
- Inverse discrete cosine transform (*IDCT*) of the DCT reconstructed coefficients and saturation.

- Picture reconstruction. In the case of P pictures, the error prediction, $e(n)$, is added on a macroblock basis to the prediction, which is built up from the reference picture via the motion vectors $v(n)$. In the case of interpolated pictures (B pictures) and because of the particular nature of the bi-directional prediction, two reference pictures, past and future, are used to form the prediction.

- Displaying of the decoded picture $\hat{x}(n)$. The corresponding rate is 15 Mbyte/s for the CCIR 601 4:2:0 sequences considered in this work.

Notice that, except for the construction of the prediction, all the activities are to be executed on the same macroblock in the sequential order as they appear in the previous list. Consequently, it is possible to pipeline the corresponding activities. Contrarily, once the motion vectors are decoded, the macroblock predictions can be constructed in parallel with the activities VLC^{-1}, Q^{-1}, IDCT, associated with the corresponding macroblock.

The syntax of an MPEG2 video bitstream contains six layers, each of which support a specific function:

- sequence
- group of pictures
- picture
- slice
- macroblock
- block

A header is associated with all the layers except the block layer. The parameters contained in the header, excluding those of the macroblock layer, have fixed length codes, and each header begins with a byte aligned start code, from the sequence header to the macroblock header. The presence of the start codes makes the bitstream scanning easier; e.g., it is possible to skip all the variable length coded data associated with the macroblocks of a slice by looking at the start code of the next slice.

A cursory analysis of the expected computational complexities of points 1÷8 suggests giving the master processor (MP) only the tasks of supervising the parallel processors (PPs) activities, of dealing with I/O, and of reading and interpreting the bitstream from the sequence layer to the start codes of the slice level. Their knowledge indeed allows it to start the decoding activities on the PPs.

As for the PP computation organization, it is possible to resort to a horizontal parallelism (“true" parallelism), to a vertical parallelism (i.e., to a pipeline), and to a mixed solution.[4] In any case, the different activities are to be distributed so that the corresponding computation times are almost equal.

Another choice concerns the dimension of the buffer used to exchange data between the PPs. If the buffer is too small, the corresponding computation times may greatly vary according to the local statistical properties of the bitstream data. Contrarily, the greater the buffer, the more constant the computation times. In our case we chose a priori (the choice has been confirmed by subsequent work) a macroblock as the interprocessor exchange unit since its dimension (6x64x2 = 768 byte for the IDCT input in the case of 4:2:0 format) is well within the 2 Kbytes size of one of the three DRAMs of each PP.

Bitstream Scanning and VLC Decoding.

The reading of the bitstream from the external device via a circular buffer and the scanning of the bitstream to extract the coded parameters from the sequence header to the slice header is executed by the MP. The information about the position of slice start codes is sent to a PP, which carries out the decoding at the slice and MB level.

The corresponding program is written in C language and runs in real-time on the MP for bit-rates up to about 60 Mbit/s. To this purpose, it is interesting to note that such a figure has been obtained by carefully tailoring the C code to the MP characteristics. Actually, its first version, derived from the C language version of the MPEG Software Simulation Group decoder, ran in real-time up to only 1 Mbit/s.

In the macroblock layer many data are coded via VLCs. Some VLCs represent single data, while others, such as DCT coefficients codes, represent pairs {run, level}. Since their decoding is most critical, we will specifically refer to them but the conclusions also apply to the other VLCs.

In MPEG2 two 113 elements Huffman-like tables are used for the DCT coefficients. Only the codes with high probability of occurrence are coded with a VLC. The less probable events are coded with an escape symbol followed by fixed length codes. There is the possibility to use either or both of the tables: one for I and one for P and B macroblocks, respectively, or one of the tables for all macroblocks.

Moreover, the first coefficient (DC) of P and B macroblocks has to be decoded via a third 113 elements Huffman-like table, and the DC coefficients of I macroblocks has to be decoded via a different method; i.e., with a predictor and another table.

An example, which shows some symbols of one table, is given in Table 1.

Table 1. VLCs for Some Pairs {run, level)

VLC	Run	Level
10	End of block	
11s	0	1
011s	1	1
0l00s	0	2
0l0ls	2	1
…	…	…
000l00s	7	1
000l0ls	6	1
000ll0s	1	2
000llls	5	1
…	…	…

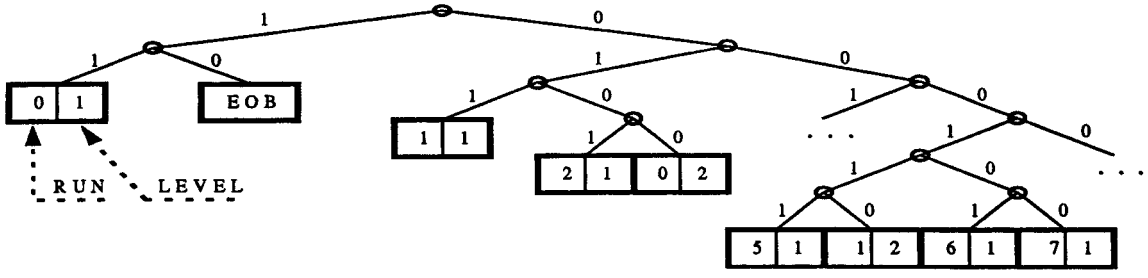
Note: *s* Denotes the Sign of the DCT Coefficient.

A Huffman code is a binary variable length prefix code which is optimal in the sense that, for a given source, no binary variable length uniquely decodable code can give a strictly smaller average length. It gives short codes to the most probable symbols and long codes to the less probable ones.

The corresponding code can be depicted as a *highly unbalanced binary tree,* and there is a one-to-one correspondence between paths from the root node to the leaves and the codewords.

A codeword of length *l* bits corresponds to a path of *l* branches in the tree beginning at the root node (depth *0)* and finishing at a terminal node of depth *l* in the tree. The codeword is the sequence of binary labels of the branches read from the first branch to the branch of depth *l.* A binary tree representation of the codes of Table 1 is given in Figure 2 (the sign bit *s* is disregarded).

Figure 2. Binary Tree Associated with the VLCs of Table 1



Note: Sign bit *s* disregarded.

Assuming a known starting point, decoding a prefix code simply involves, in principle, scanning bits until a valid codeword is found. In practice, binary tree searching is not so simple and can be quite time-consuming. It is important to organize the tree in such a way that fast decoding is possible and a reduced amount of memory is used.

In the sections *"Table-Look-At" Algorithm* and *"Distributing Tree-Balancing" Algorithm* sections, we present the two techniques that appear more suitable for the problem at hand and the reasons that led us to adopt the second one.

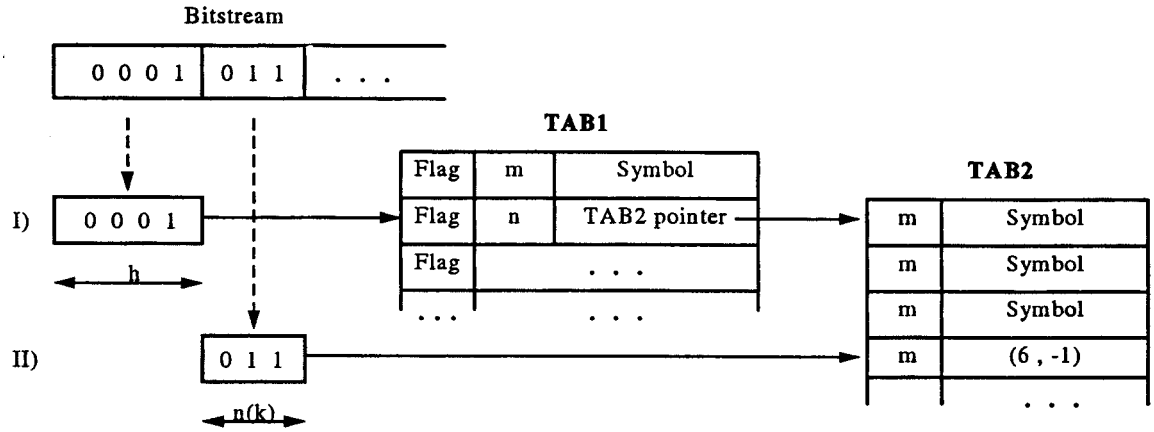
Common to both techniques is the principle of partially balancing the tree by introducing duplicated nodes. Searching in a fully balanced binary tree is, indeed, trivial.

"Table-Look-At" Algorithm

The "table-look-at" algorithm solution derives from the "table-look-at" algorithm presented in *The Art of Computer Programming*.[5] Let us choose a fixed length: *h* bits, which corresponds to a path of depth *h* in the associated tree. We then augmented the tree by associating to each node at depth less than *h* two identical sons and repeating this procedure until the tree is balanced up to the level *h.* In the augmented tree each nonterminal node at depth *h* becomes the root of an unbalanced subtree, and to all such subtrees the balancing procedure is applied again until all the terminal nodes of the original tree are considered.

The balancing operation leads to the algorithm exemplified in Figure 3 (sign bit S included) for *h=4* with reference to the tree of Figure 2. A field of *h* bits is read from the bitstream and is used as the key, *k,* to read the content at position *k* of the first table (TAB1). If the VLC length is not greater than *h,* the table content gives its actual length *m(k)* and the corresponding (run, level) symbol; otherwise it gives the pointer to TAB2 and the number *n(k)* of additional bits to be read from the bitstream. The corresponding field is used to address TAB2, which contains the symbol and the actual VLC length. Figure 3 shows the decoding of the VLC 0001011 associated with the symbol {6,-1}.

Figure 3. Data Structure of the "Table-Look-At" Algorithm



The performance of the method is strongly dependent from *h;* the greater *h* the greater the probability the search finishes at the first step. (If *h* were equal to the maximum VLC length, the search would be completed in one step). At the same time, however, the greater *h,* the greater the size of TAB 1 and the greater the waste of memory associated with short and more probable VLCs.

These considerations lead to the choice *h=8* (the percentage of VLCs with lengths not greater than 8 bit, ranges from 70% to 90% as the bit-rate goes from 2 to 16 Mbit/s). The corresponding PP assembly program requires 16 cycles per-VLC for the activities 4) and 5) of the section, *The Video Decoder*, but the size of all the tables needed to decode the DCT coefficients VLCs is 4368 bytes.

"Distributed Tree-Balancing" Algorithm

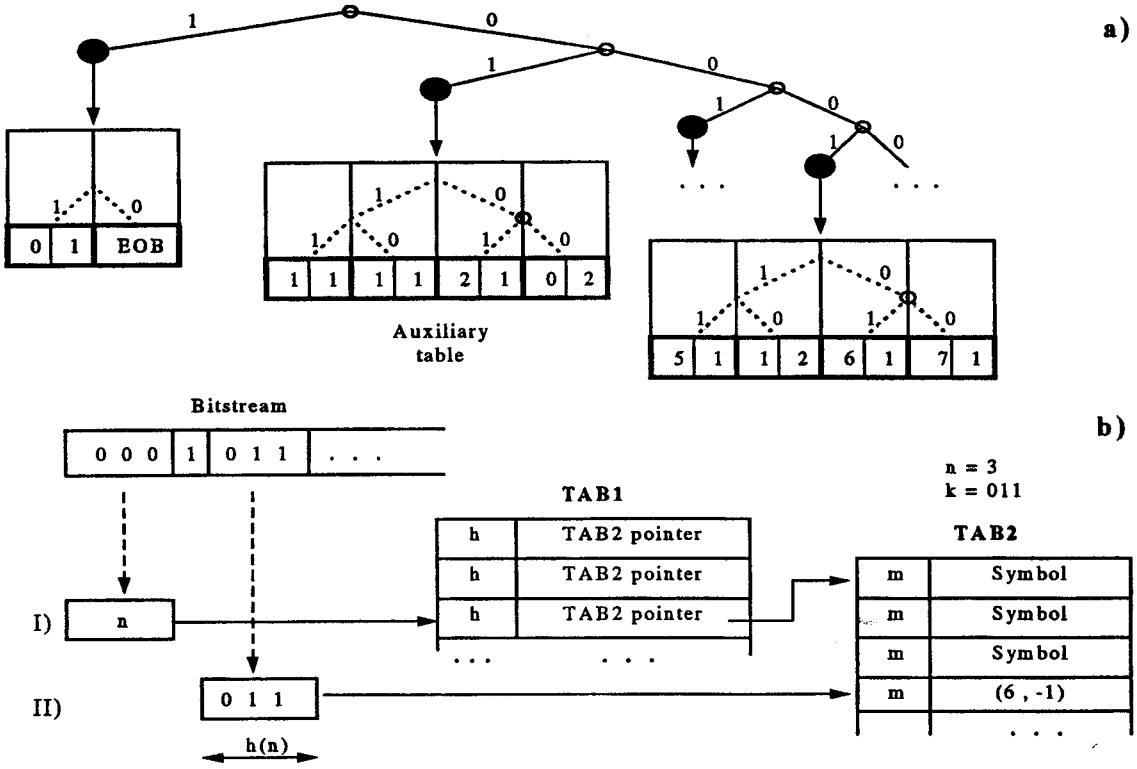
The “distributed tree-balancing” algorithm method arises from the simple observation that most codewords begin with a sequence of zeros followed by a 1 and that, in the PP assembly, there is an instruction which, in one cycle, gives the number of zeros, *n,* to the left of the leading 1 of the content of a register. This makes it easy to identify all leading patterns of type 1..., 01...,001... etc.; i.e., all codewords that start with the same number of zeros or, stated otherwise, to go in one step from the root of the tree to the corresponding internal node (black nodes of Figure 4a).

Each such node becomes the root of the subtree composed by all its descendants. The subtree is balanced, as in the previous technique, by making the depth of all its paths equal to the subtree maximum depth, *h(n)* (duplicating when necessary all the terminal nodes which have depth less than *h(n)).*

In this way, the search of the marked internal nodes is carried out by counting the number of leading zeros and using this quantity to point to a table containing the terminal parts of the associated codewords.

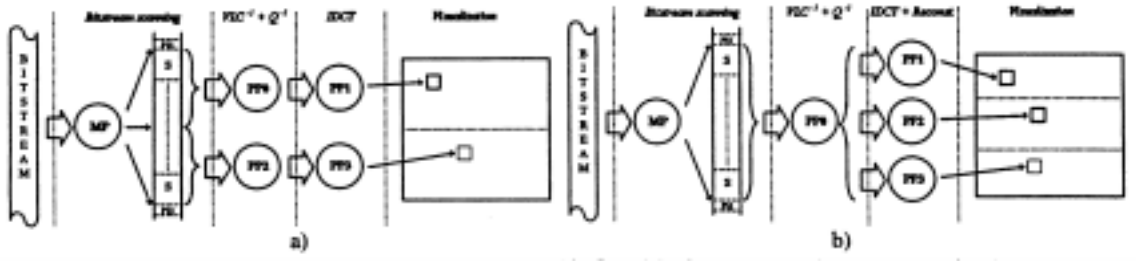
Figure 4b shows the data structures of the algorithm.

Figure 4. Binary Tree Associated with the "Distributed Tree-Balancing" Algorithm



The first table (TAB1) contains *h(n)* and the pointer to a second table (TAB2), which contains the pair {run, level} and the VLC length *m(k).* The decoding of the codeword 0001011 is sketched in Figure 5.

Figure 5. Decoder Organization : a) I Pictures and b) P and B Pictures (PH: Picture Header; S: Slice).



For the activities 4) and 5) of the section, *The Video Decoder*,this method requires only 1664 bytes of DRAM and a constant decoding time of 20 cycles per VLC; i.e., when compared with the first method, memory is sacrificed for processing time.

It has to be taken into account, however, that in order to minimize packet transfers it is preferable to load (in the DRAM) only once all the tables needed to decode all variable length coded data and other data structures, while leaving room for exchange buffers with the PPs and the MP. This consideration led us to prefer the second algorithm (it allows us to reserve 2 Kbytes of DRAM for the bitstream section to be decoded coupled).

Table 2 presents the corresponding decoding times for different VLCs. In the case of DCT coefficients, the per-VLC count is comprehensive of inverse quantization, saturation, mismatch control, and data reorganization from the zigzag order to the row-column order.

Table 2. Processing Times for VLC Decoding (cycles/VLC)

	DC coeff	**Non DC coeff**	**Esc**	**EOB**	**Others**
DCT (I)	18	20	19	22	-
DCT (P,B)	26	21	20	22	-
Other VLCs	-	-	-	-	14

Decoder Architecture

The PP organization requires the knowledge of both the processing times associated with the different activities depicted in Figure 1 and their dependence from the coded video average bit-rate.

The VLC decoding is the task more influenced by the bit-rate. As a matter of fact, the coder controls the output bit-rate through the quantizer stepsize, $\Delta(n)$: the greater $\Delta(n)$, the higher the number of zero DCT coefficients and, correspondingly, the more efficient the run-length coding and the lower the resulting bit-rate. As a result, the per-VLC average number of DCT coefficients decreases with the bit-rate while the VLC decoding time, $t_{vlc,}$ increases. Its actual value depends on the particular video sequence.

In the following we will base our predictions on experimental results obtained by running a (conveniently modified) C language version of the MPEG2 decoder on the sequences "Calendar" and "Flower" coded at various bit-rates. These results, exemplified by the histograms of Figure 6a and b, show that:

- At any bit-rate, t_{vlc} increases, as expected, in going from B to P and 1 macroblocks (e.g., at 4 Mbit/s the average value of t_{vlc} is 26.2, 13.9, and 7.7 μs/macroblock for I, P, and B macroblocks respectively, if the decoding is carried out on one PP).
- t_{vlc} increases (almost proportionally for P macroblocks) with bit-rate.

The actual values show that, to decode the VLCs in real-time frame by frame (i.e., within 40 ms for the European 25 frame/s rate), it is necessary to use two PPs for I pictures and one PP for P and B pictures for rates up to about 6 Mbit/s.

The two other main tasks—IDCT and reconstruction—are less rate dependent. A preliminary estimate of the corresponding processing times expressed in MVP-cycles/pixel is given in Table 3.

Table 3. IDCT and Reconstruction Processes: MVP-Cycles/Pixel for I, P, and B Macroblocks

	I	**P**	**B**
I IDCT	5.8	5.2	5.2
Reconstruction	0	4.1	6.6

As for IDCT, the estimates are based on an assembly version of the Chen's algorithm[6] and include auxiliary operations; e.g., saturation, half-word conversion, etc. Presently, we are optimizing an assembly version of the potentially faster lee's algorithm.[7] The reconstruction estimates are based on available information[3] about an assembly procedure with "minimum" code length and constant reconstruction time.

The data of Table 3 can be translated into time and, associated with VLC decoding times, suggest the PP organization sketched in Figure 5:

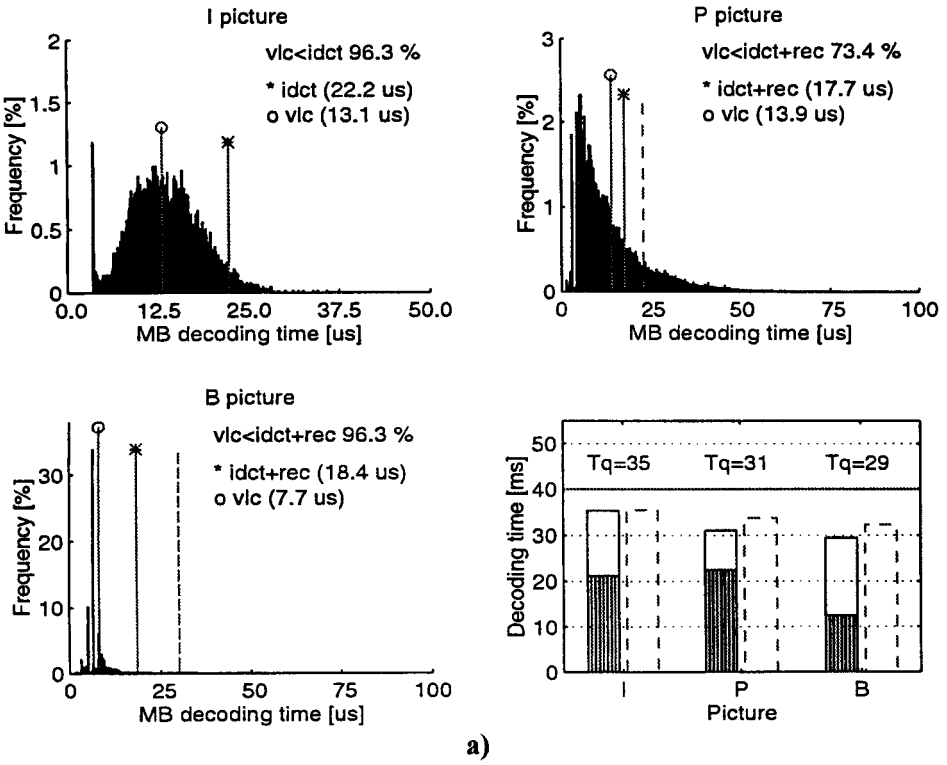
- *I-pictures:* VLC decoding is carried out in parallel by two PPs, each one connected in pipeline with another PP, which does IDCT.
- *P and B-pictures:* VLC decoding is carried out by one PP connected in pipeline with three PPs operating in parallel, which do IDCT and reconstruction.

The corresponding performance was estimated via simulation. The statistics obtained at 4 and 8 Mbit/s are shown in Figure 6a and b, respectively. Each figure gives the *per-macroblock* processing times for I, P, and B pictures. In particular, each subplot shows the histogram of the per-macroblock VLC decoding times: the bar marked with "o" denotes its mean value.

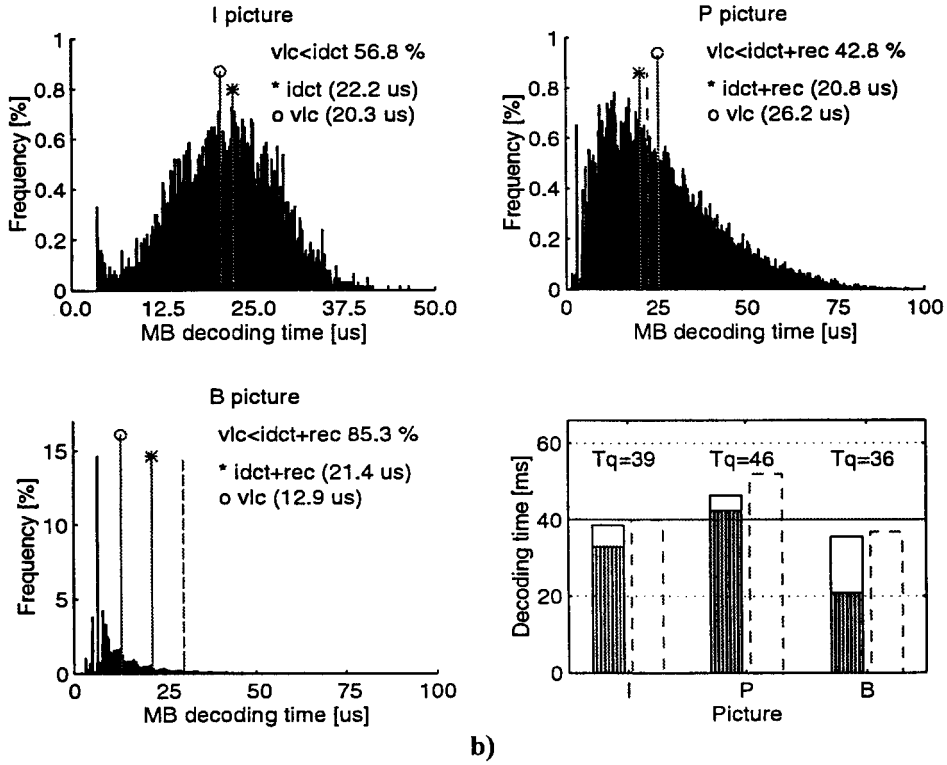
The bar marked with "*" denotes the actual mean value of the other activity; i.e., IDCT or IDCT+reconstruction times, while the dashed bar gives the value one would obtain under the hypothesis of a constant IDCT time. The two values differ, the former being less than the latter, because in P and B pictures some blocks (or MBs) can be zero and the corresponding IDCT can be skipped. In addition, in each subplot, the percentage of time t_{vlc}, is less than the time required by the other activity, is shown.

The bar diagram at the bottom right of the figures gives the *per-frame* processing time. It was computed taking into account the proposed PP organization. The processing time was computed for all MBs as the maximum of the times required by the two pipelined activities (VLC decoding and IDCT or IDCT+reconstruction). The continuous and dashed line bars give the average and worst-case decoding times, while the black bar shows the average processing time associated with VLC decoding only.

Figure 6. VLC Decoding and IDCT+Reconstruction Statistics a) at 4 Mbitls and b) at 8 Mbit/s



a)

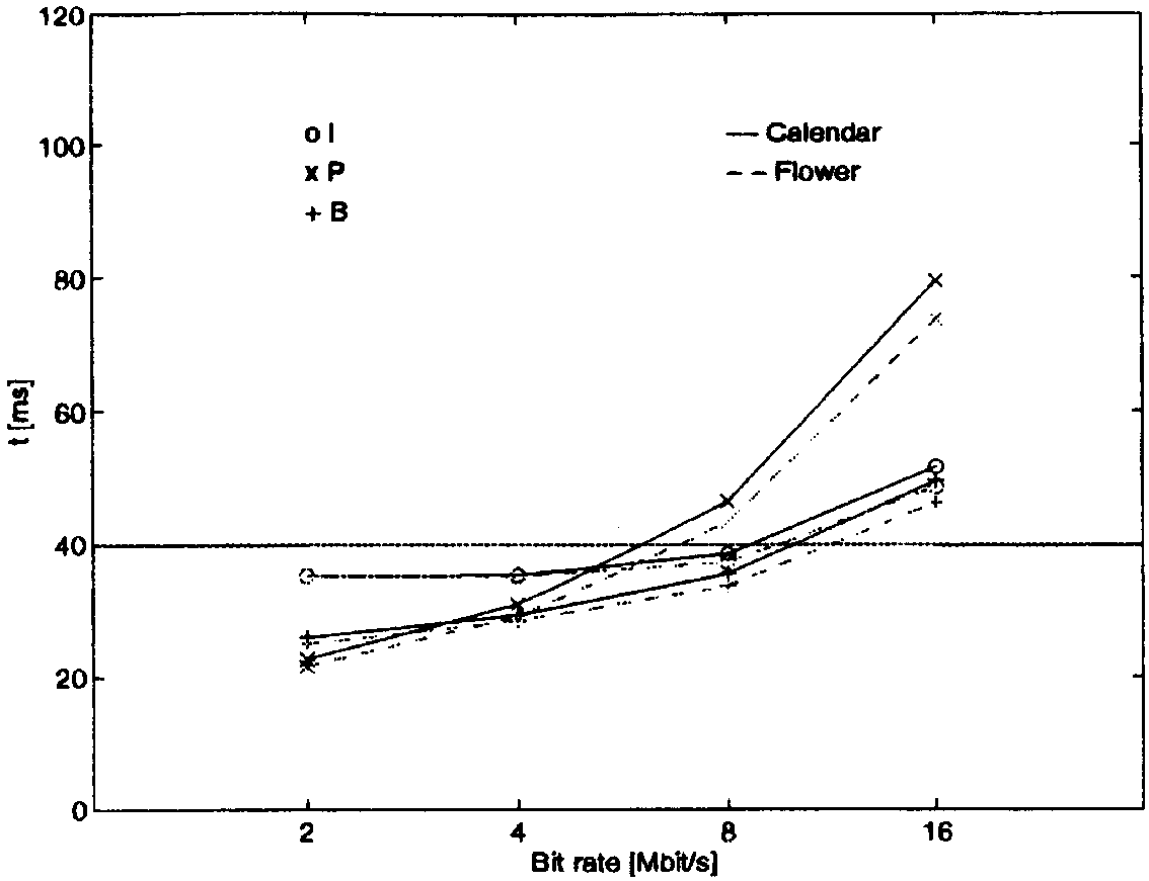


b)

The figures clearly show that VLC decoding becomes more important in determining the processing time as the bit-rate increases. This is particularly true for P pictures, that at 8 Mbit/s (Figure 6a and b) become the only pictures whose decoding time is greater than 40 ms, the value required for real-time operation on a frame by frame basis.

Figure 7 summarizes the expected per-frame decoding times *vs.* bit-rate for I, P, and B pictures. It allows one to predict the decoder will properly operate at bit-rates up to about 6 Mbit/s. Actually, the preliminary version of the MVP decoder has been tested on I pictures and its performance was found to agree with expectations.

Figure 7. Decoding Time for I, P, B Pictures vs. Bit-Rate



From Figure 6b it results that there is a (small) margin for I and B pictures. This fact suggests there is room for some improvement by exploiting the difference between the transmission and visualization orders of the different pictures; a sequence to be visualized as I_1, B_2, B_3, P_4, B_5, B_6, P_7... is transmitted as I_1, P_4, B_2, B_3, P_7, B_5, B_6 This implies that, at the start, the constraints for real time decoding become:

$T_{I1}<2T_q, T_{I1}+T_{P4}+T_{B2}<3T_q, T_{I1}+T_{p4}+T_{B2}+T_{B3}<4T_q,$

where T_q=40 ms; i.e., the frame period, while in the steady state, it must be:

$T_*+T_B<2T_q, T_*+2T_B<3T_q,$

where "*" denotes an I or P picture.

The corresponding simulations show an increase of the maximum bit-rate, which, however, is not sufficient to go to 8 Mbit/s. At this bit-rate, all the constraints are satisfied with some margin with the exception of the PB couples, which do not satisfy the constraint $T_p+T_B<2T_q$ with "probability" about 1/4.

In any case, this analysis makes us more confident that the complete decoder will properly work up to 6 Mbit/s.

Acknowledgments

The authors would like to thank the "old" students of the Image Processing Laboratory: Nicola Griggio and Fabio Valente (Alcatel-Telettra, Milan-IT) and Massimo Martelli (Texas Instruments, Milan-IT) for their valuable support and cooperation.

References

[1] D.J. Le Gall. "The MPEG Video Compression Algorithm," *Comm. ACM,* 34,4,47-58, April 1991.

[2] Draft, Recommendation H.262 – ISO/IEC 13818-2, "Generic Coding of Moving Pictures and Associated Audio: Video," May 1994.

[3] N. Griggio, F. Valente. Alcatel-Telettra (Milan-IT), Private Communication.

[4] M.J. Shute. "Fifth Generation Wafer Architecture", Prentice Hall, 1988.

[5] D.E. Knuth, *The Art of Computer Programming*, vol.3, Addison-Wesley, 1973.

[6] W. Chen, C. Smith, S. Fralick. "A Fast Computational Algorithm for the Discrete Cosine Transform," *IEEE Trans. Communications, 25, 9,* 1004-1009, Sep. 77.

[7] B. Lee. "A New Algorithm to Compute the Discrete Cosine Transform," *IEEE Trans. Acoustics, Speech, Signal Proc.,* 32, 6, 1243-1245, Dec.84.