

# An Implementation of Data Reusable MPEG Video Coding Scheme

Vasily G. Moshnyaga

**Abstract**—This paper presents an optimized MPEG2 video codec implementation, which drastically reduces the number of computations and memory accesses required for video compression. Unlike traditional scheme, we reuse data stored in frame memory to omit unnecessary coding operations and memory read/writes for unchanged macroblocks. Due to dynamic memory sharing among reference frames, data-driven macroblock characterization and selective macroblock processing, we perform less than 15% of the total operations required by a conventional coder while maintaining high picture quality.

**Keywords**—Data reuse, adaptive processing, video coding, MPEG

## I. INTRODUCTION

Video compression scheme, such as MPEG (fig.1) uses block-based motion compensation and a discrete transform (DCT) for coding of prediction residues. The input video frame is split into macroblocks (of  $N^2$  pixels each) which can be treated as intra- or as inter-macroblocks. Inter-macroblocks in P- and B- frames are coded based on motion estimation and compensation (ME) to remove temporal redundancy by subtracting the reference frame data from the current frame data. The predicted macroblock is further partitioned into  $8 \times 8$  blocks and supplied to the 2-D Discrete Cosine Transform (DCT) to eliminate spatial redundancy of the differential error. The DCT produces 64 coefficients, which are then quantized based on the rate control value to remove high frequency components invisible for human eyes. The result then feeds both the variable length coder (VLC) and the local decoder. The former further compresses information and sends it to channel buffer. The latter implies inverse quantization (IQ) and inverse DCT (IDCT) to reconstruct the current image and store it into frame.

Due to large picture sizes, frame memory has the highest capacitance in the coding system, dominating its energy dissipation. Even if DRAM is embedded on a chip, it dissipates almost 1/3 of the total power [1]. The ME and DCT (IDCT) perform most of the coding computations with 66% being ME [2] (assuming Full Search) and about 25% being DCT/IDCT. Depending on implementation, their power consumption may

Manuscript received October 29, 2004. This work was supported by The Ministry of Education, Culture, Sports, Science and Technology of Japan, Grant-in-Aid for Scientific Research C (2) No.14580399 and Grant-in-Aid for Creative Basic Research (A) No.14GS0218. The author is with the Department of Electronics Engineering and Computer Science, Fukuoka University, 8-19-1 Nanakuma, Jonan-ku, Fukuoka 814-0180, Japan (e-mail: vasily@fukuoka-u.ac.jp).

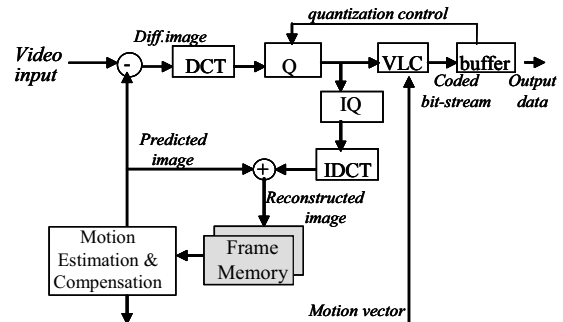


Fig.1: Block diagram of the MPEG video coding

vary from 6% to 30% (for ME) and -9%-20% (for DCT) [1]. Consequently to reduce coding power we must optimize these three units as much as possible.

This paper presents an optimized MPEG-2 video encoding system capable of exploiting picture correlation for reducing computations and memory accesses in video coding. Unlike other designs, the system reuses data stored in frame memory to omit unnecessary coding operations and memory accesses for unchanged macroblocks. Due to dynamic memory sharing among reference frames, data-driven macroblock characterization and selective processing, it may take as less as 15% of total processing operations of traditional coder while maintaining high picture quality.

## II. THE PROPOSED SYSTEM

### A. Main Features

The system employs two techniques to reduce the amount of computations and memory writes in video coding. The first one is memory sharing between the reconstructed frames. The basic idea is to use the same memory for storing adjacent reconstructed picture frames. Because the frame memory data is consumed and renewed gradually, i.e. by macroblocks, the macroblocks of the reference frame  $F_{l-1}$  gradually become useless and can be replaced by data of a newly reconstructed frame under condition that a macroblock  $X \in F_l$  overwrites macroblock  $Y \in F_{l-1}$  with identical coordinates. Thus, while one region of the frame memory stores the data of frame  $F_{l-1}$ , another region of the memory is replaced with pixels of a new frame  $F_l$ . In our architecture, the reconstructed P-picture gradually replaces the previously saved I-picture or P-picture within the same memory without affecting the coding algorithm. Because the current block pixels can not be written

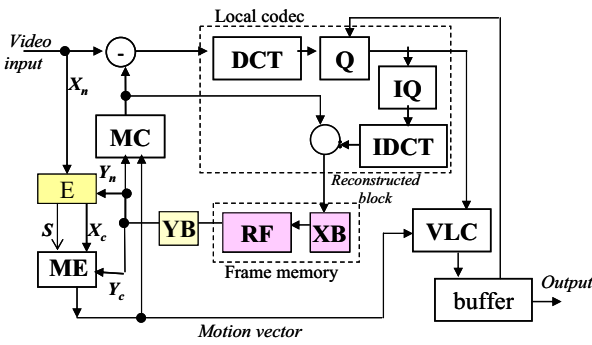


Fig.2: The proposed coding scheme

into the memory until all pixels of the previous block are processed, a FIFO-type buffer (XB) is placed on the memory input.

The second technique employed by the proposed architecture is frame memory data reuse. Additionally to static reuse of the reference pixel data both within and between search regions [3- 7] we reuse macroblock data among the frames [8]. Since a reconstructed frame overwrites its predecessor in block-by-block fashion, we evaluate the picture content of each incoming macroblock and if it is marked as *unchanged*, we reuse the macroblock data thus eliminating all the coding and storing operations for the macroblock. To classify a macroblock  $X \in F_i$ , we count pixels  $x \in X$  which match the corresponding pixels  $y \in Y$  in the reference block ( $Y \in F_{i-1}$ ). (Note that positions of  $X \in F_i$  and  $Y \in F_{i-1}$  are same). Two pixels  $x \in X, y \in Y$  have a match if their four most significant bits are same, i.e.  $AND_{i=0}^{i=3} \{XNOR(x_i, y_i)\} = 1$ . A macroblock  $X \in F_i$  is considered *unchanged* (or *stationary*) if its number of matching pixels  $P(X)$  is larger than a given threshold ( $T$ ). Otherwise, the macroblock  $X \in F_i$  is marked as *changed* (or *non-stationary*) and processed as usual.

### B. Implementation Scheme

Figure 2 outlines our implementation scheme. Similarly to existing realizations, we use a small search area buffer (YB) to reuse data on read. However, the frame memory in our system is composed of the reconstructed frame storage (RF) and a memory buffer (XB). The memory RF is one frame in size and dynamically shared by adjacent reference pictures, as discussed above; E is the macroblock evaluation unit; ME is the motion estimation unit and MC is the motion compensation unit.

The encoding is implemented as follows. The reconstructed image of the first (I-picture) of GOP is stored into the RF memory. When the first macroblock ( $X_n$ ) of the next picture enters the system, the E unit evaluates its content and if  $X_n$  is *stationary*, it stops all operations in the encoder for the macroblock except the VLC, thus producing only zero-vector at the output. The gray pattern in Fig.3 illustrates the disabled modules. If the content of the block is *non-stationary*, we predict the macroblock motion in regards to the data stored in YB, compensate the motion on MC, compute the macroblock image error, encode the error image (on DCT and Q), reconstruct it (on IQ and the IDCT) and finally write the

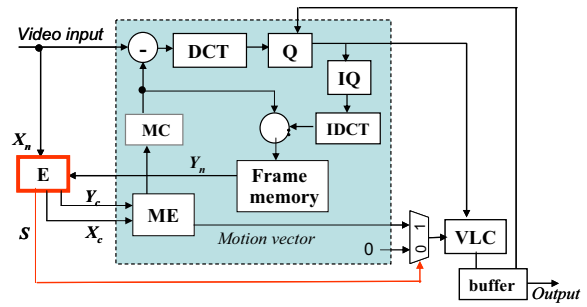


Fig. 3: The disabled macroblock evaluation

reconstructed image of  $X_n$  into the XB and eventually into RF (if the macroblock belongs to the P-picture), to replace the corresponding macroblock of the I-picture. Thus only non-stationary macroblock of the P-frame will be written into the frame memory. The predicted error image and the motion vectors of the encoded macroblock are sent to the channel.

### C. Macroblock Evaluation Unit

Figure 3 shows the macroblock evaluation unit. For each new macroblock  $X_n$  of the current frame, it receives the corresponding search window data  $Y_n$  and generates the macroblock label (MB label). The Pixel Matching Unit takes a new pair of pixels,  $x \in X_n, y \in Y_n$ , and increments the counter if four most significant bits of these pixels are same. After  $N^2$  steps, the MB marking unit compares the value (C) accumulated by the counter to a threshold,  $T$ , and if the  $C > T$ , it labels the macroblock  $X_n$  as *stationary*. Otherwise, the MB is labeled “non-stationary”. With a new block, the counter is reset.

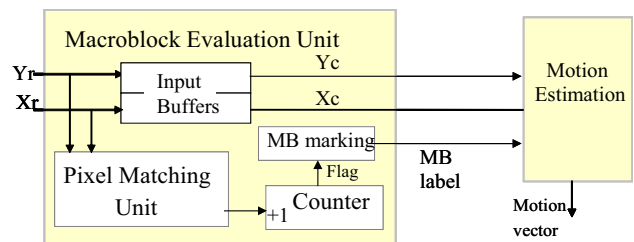


Fig. 4: The Macroblock evaluation unit

The computational overhead of the scheme involves one binary comparison per pixel in Matching Pixel Unit, one addition per pixel for counter, one comparison and two FIFOs to store pixels of the next reference block  $X_n$  and the next candidate block  $Y_n$ . In the encoder, pixels enter the ME unit in the raster-scan mode. Most of the next block pixels are accumulated in FIFO when the current block is processed. Thus we need  $N$  extra locations in FIFO. Due to overlap of the search area of the current block and the search area of the next block, the number of search area pixels to be stored in FIFO is  $(N+p)*p$ .

D. Optimized Processing Units

Another distinguished feature of our system is that main processing units (ME, DCT, Q, IQ, IDCT) operate if and only if the incoming macroblock is “non-stationary”. Whenever the label is ‘stationary’, the units produces zero output (motion vector for ME, coefficients for DCT and Q, etc.) without any computation. With a new block, the counter is reset.

E. Frame Memory Organization

Fig.5 outlines the frame memory organization. The memory controller examines the motion vector of the current macroblock *X* and if it is zero, disables the memory write operation. Thus, macroblocks which have zero-motion vector are not written into the memory. On read, all pixels are placed in the buffer *YB* for the future reuse. The size of *RF* is  $W \times H$ . For  $p=N/2$ , the size of *XB* is  $(2 \times W/N + 2) \times (N/2)^2$  bytes, and that of *YB* is  $(N+1)^2$  bytes.

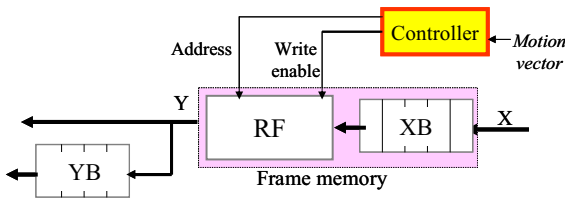


Fig.5. Frame memory organization

III. EXPERIMENTAL EVALUATION

We developed a prototype software implementation of the MPEG2@MPLL video coding scheme and compared to traditional coding scheme (no data reuse) on four standard video streams: *Miss America* (176 144 pixels, 150 frames), *Carphone* (176 144, 382), *Foreman* (176 144, 298) and *Salesman* (252 288, 300). The details of motion estimation algorithm employed in the scheme are given in [9].

Table 1 lists the results observed for different block sizes  $N=8$  and  $N=16$ , ( $p=\pm 8$ ) when the threshold ( $T$ ) accounted 70% of the total number of pixels in the blocks. Here, *RR* is the percentage Fig.4: Frame memory organization the average error picture.

TABLE I THE NUMBER OF UNCHANGED MACROBLOCKS PER FRAME AT T=70%

Video	Total frames	Frame size	N=8		N=16	
			RR	Error	RR	Error
Salesman	288	252x288	85	1.89	75	2.0
Miss-A	150	176x144	86	2.58	82	2.93
Carphone	382	176x144	82	2.46	76	2.81
Foreman	298	176x144	46	2.29	39	2.48

As we see, the smaller the macroblocks, the better the results. For  $N=8$ , the maximum reduction factor can be as much as an 86%. If we increase the threshold the number of stationary blocks decreases, as shown in Figure 6. Also decreases the number of ME computations (*Comp.*) and the picture error, as shown in Figure 7.

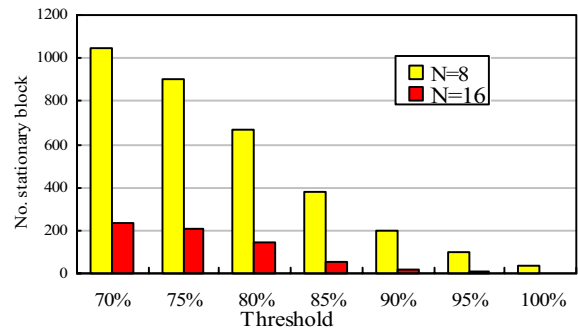


Fig. 6: The number of stationary blocks vs. threshold

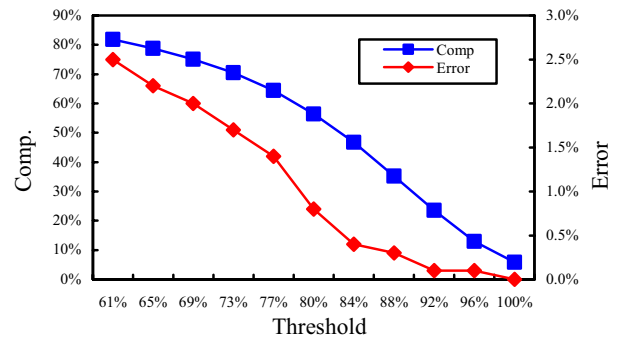


Fig. 7: Error and ME computations vs. threshold (*Miss-America*)

At  $T=84\%$  almost all the sequences had less than 1% picture error (only *Miss-A* had 1.6% error). At the same time, the 70% threshold caused only 2%-3% picture degradation (less than 0.9dB drop in PSNR) while almost doubling the reduction ratio in comparison to  $T=84\%$ . Figure 8 illustrates the picture (*Miss-America*), frame number 76) obtained at the 70% threshold in comparison to the original picture (obtained at the 100% threshold). Since the picture error is almost insensible for human eye, we suggest using  $T=70\%$  to maximize data reuse while maintaining the acceptable picture quality. The actual reuse factor strongly depends on frame to frame picture correlation. As Figure 9 shows, it heavily varies with frames showing the best results on frame 76 (for *Miss-America*), frame 85 (*Carphone*), and frame 175 (*Foreman*). The picture movement in these frames is very low; so the number of static macroblocks is large.

Finally we evaluated the performance of the proposed software coding scheme by measuring the amount of clock cycles consumed by the C version of codec. Table 2 shows the average clock counts required for coding of QCIF image frames required by the proposed scheme (70% matching threshold) in comparison to original coding scheme. As we see, the clock count of the ME and MC tasks is reduced to one-third, while that of quantization (Q) is reduced by a factor of four. Overall, the amount of clock cycles taken by encoding is suppressed by a factor of 3.24x.

TABLE 2  
THE NUMBER OF CLOCK CYCLES REQUIRED FOR CODING

Function	Clock count		Speed-up ratio
	Original	Proposed	
ME+MC	72.14	23.28	3.09
DCT	20.3	5.84	3.47
IDCT	3.6	0.94	3.83
Q	11.5	2.72	4.22
IQ	0.96	0.56	1.74
Others	19.5	6.08	3.21
Overall	128	39.42	3.24



Fig.8 Original picture (above) and the picture obtained for the matching threshold=70% (bottom)

#### IV. CONCLUSION

We have presented a new implementation scheme to reduce the amount of computations and memory accesses in video coder. The data-driven macroblock characterization and adaptive macroblock reuse greatly contributes to fast and efficient video coding with software. As experimental results showed, the scheme was able to maintain high quality coding while saving as much as 85% of the macroblock coding operations and memory writes. To evaluate actual power reduction we are now working on hardware encoder design. An adaptive (run-time) threshold modification is also investigated.

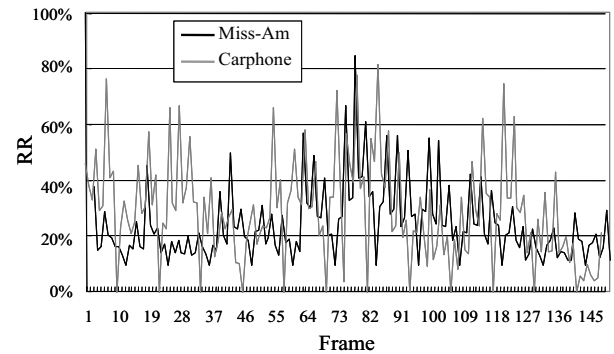
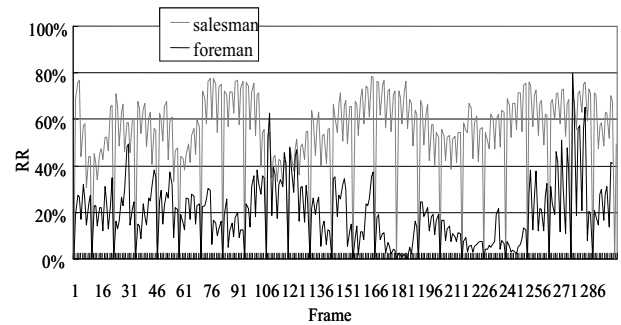


Fig. 9 Variation of the Reduction Ratio per frame

#### REFERENCES

- [1] M. Takahashi, et al, A 60-MHz240mW MPEG-4 Videophone LSI with 16Mb Embedded DRAM, IEEE JSSC, 35 (11), pp.1713-1721, Nov.2000.
- [2] T. Hashimoto, et al, A 90mW MPEG4 Video Codec LSI with the Capability for Core Profile, IEEE ISSCC 2001, pp.140-141.
- [3] J. Tuan, et al, On the data reuse and memory bandwidth analysis for full-search block-matching VLSI architecture, IEEE Trans. CASVT, 12 (1) pp.61-72, 2002
- [4] J. Davidson, et al, Memory access coalescing: A technique for eliminating redundant memory accesses, PLDI, 29 (6) 1994.
- [5] D.Kolson, et al, Integrating program transformations in the memory based synthesis of image and video applications, IEEE ICCAD'94, 27-30.
- [6] S. Wuytack, et al, "Formalized methodology for data reuse exploration for low-power hierarchical memory mappings", IEEE TVLSI Systems, 6 (4) pp.529-537, Dec. 1998..
- [7] T. Onoye, et al., A VLSI architecture for MPEG2 real time motion estimator, IEEE ISCAS 1996
- [8] V. G. Moshnyaga, Reduction of memory accesses in motion estimation by block-data data reuse, Proc. IEEE ICASSP'02
- [9] V.G. Moshnyaga, A new computationally adaptive formulation of block-matching motion estimation, IEEE Trans. CASVT, 11(1), Jan. 2001.

**Vasily G. Moshnyaga** (M'92-SM'03) received his computer engineering degree with Honors from the Sevastopol Device-Making Institute, Sevastopol, USSR, in 1980 and Ph.D. degree in computer engineering from the Moscow Aviation Institute in 1987. From 1986 to 1992 he was with faculty of Technical University of Moldova (Former Kishinev Polytechnic Institute), Kishinev, Republic of Moldova. From 1990 to 1992 he was a visiting scholar in Kyoto University, Japan. From 1992 to 1998 he was a Lecturer of the Department of Electronics and Communication, Kyoto University, Japan. Since 1998 he has been with the Department of Electronics and Computer Science, Fukuoka University, Japan, where he is currently a Professor. His research interests are in the areas of VLSI and computer architectures, low power design methodologies and video processing. He is a senior member of IEEE and a member of Information Processing Society of Japan