Combined Frame Memory Motion Compensation for Video Coding

Nelson Yen-Chung Chang and Tian-Sheuan Chang

***Abstract*—The frame memory has long been the dominant component in a video decoder in terms of energy, area, and latency. We proposed a non-combined frame memory motion compensation (CFMMC) for video decoding which facilitates the characteristic of the perfect-matched macroblock (MB) to avoid unnecessary memory access and to save energy. The statistic result confirms that some sequences have more than 70% of MBs being perfect-matched MB. The CFMMC hardware architecture is further evaluated for latency, area, and energy. The hardware architecture shows that with SRAM-base frame memory, the equivalent gate count can be reduced by 37.7%, and the energy consumption and the latency may also be improved for sequences with enough percentage of perfect-matched MBs. Since the benefit of the CFMMC is highly dependent on the percentage of perfect-matched MBs, it is best suited for applications with large portion of static background, such as video surveillance, video telephony, and video conferencing.**

***Index Terms*—Frame memory, MPEG-4, motion compensation, video decoding, zero-skipping.**

I. Introduction

DESPITE that the latest video coding standard provides much better compression performance as well as extra functionalities, all video coding standards are still consist of motion compensation, transform, and entropy coding. Among these common video coding tasks, motion compensation involves extensive frame memory access. Consequently, the frame memory access becomes the dominating part in the energy consumption of a video decoder. In addition, the requirement of storing the great amount of the frame data would result in a frame memory which occupies most of the silicon area in motion compensation. Therefore, the optimization of the frame memory architecture is of great significance in reducing the area and energy consumption of motion compensation.

The most common frame memory architecture for video coding without bidirectional prediction is the *ping-pong frame memory* (PPFM), which stores the reconstructed current frame and the reference frame in two memories. The PPFM swaps the role of the reconstructed current frame memory and the reference frame memory upon the completion of each frame's motion compensation. As a result, the PPFM requires a memory size of two frames and often accounts for approximately half of the energy consumption in a video decoder [1].

Motivated by the fact that the PPFM is very area and energy consuming, we proposes the *combined frame memory* (CFM) architecture that combines the reconstructed current frame memory and the reference frame memory into one single memory. This architecture uses the presence of the macroblock (MB) with zero motion vector and no residue to reduce the area and energy consumption. This type of MB is identical in the reference frame and the reconstructed current frame. Therefore, copying the MB from the reference frame to the reconstructed current frame can be eliminated in the CFM.

There are two contributions in this work. First, the statistical analysis and the concept of the non-combined frame memory motion compensation (CFMMC) are presented. The percentage of MBs without motion and residue is investigated. The other contribution is the evaluation result of the CFMMC architecture. The implementations of the CFMMC and the *ping-pong frame memory motion compensation* (PPFMMC) are compared. The evaluation result suggests that the CFMMC is suitable for applications with much still background, such as video surveillance, video telephony, and video conference.

The rest of the paper is organized as follow: the next section presents the related work on motion compensation. Section III presents the statistics analysis and the concept of the proposed CFMMC. Section IV presents and discusses the hardware architecture of the prototype CFMMC. Finally, Section V concludes the CFMMC.

II. Related Work

To reduce the size of the frame memory, [2]–[4] adopted a merged-frame approach that stored the reference frame and the reconstructed frame together using one frame memory with the size slightly larger than one frame. Along with the reduced size frame memory and local buffers, these works claimed that the merged-frame approach is capable of reducing the power consumption. Among these works, [2] and [3] proposed an in-place storage optimization for video decoders. The in-place storage used a buffer to store the reference frame data that are overlapped with the reconstructed current frame data in a snake-like manner. To handle the complex address generation and the control, they implemented a prototype using software. [4] also proposed a similar merged-frame memory architecture for motion estimation and compensation in an encoder. Although these works successfully reduced the frame memory size, none of them mentioned further improving the performance of motion compensation by exploiting the characteristic of MBs without motion and residue.

Moshnyaga's works [1], [5] on motion estimation reported the presence of block-data whose content remain unchanged between the adjacent frames. These unchanged block-data are facilitated to eliminate frame memory writes and computations during the motion estimation. In order to reduce memory writes for the unchanged block-data, Moshnyga's work also adopted the merged-frame approach when the coding pattern has no B-frame. Although the result shown was quite well for the test

Manuscript received October 10, 2005; revised June 1, 2006. This work was supported in part by the National Science Committee, Taiwan, R.O.C., under Grant NSC-93-2200-E-009-028. This paper was recommended by Associate Editor F. Pereira.

The authors are with the Institute of Electronics, National Chiao-Tung University, Hsinchu 106, Taiwan, R.O.C. (e-mail: ycchang@twins.ee.nctu.edu.tw).

Digital Object Identifier 10.1109/TCSVT.2006.881867

TABLE I
PERCENTAGE OF PERFECT-MATCHED MBS WHEN QP = 16

Test sequences	QCIF (%)	PSNR(db)	Bitrate(kb/s)	CIF (%)	PSNR(db)	Bitrate(kb/s)
container (A)	91.74	30.51	149.3	88.91	31.28	464.6
mother_daughter (A)	81.42	32.29	94.9	77.65	34.34	275.6
hall (A)	86.21	30.75	160.3	83.86	32.60	482.8
akiyo (A)	91.32	32.27	114.9	89.09	34.53	321.4
coastguard (B)	10.35	29.15	179.2	2.69	29.67	698.0
foreman (B)	24.49	30.32	176.7	23.38	31.37	560.0
news (B)	82.53	30.44	172.4	83.01	32.50	491.8
stefan (C)	15.71	27.48	361.1	20.90	29.09	1274.7
mobile (C)	10.93	26.18	442.3	3.39	27.02	1724.2

sequences listed in their works, the experiment result on video sequences with great amount of motion was absent.

In summary, to the authors' best knowledge, the motion compensation for video decoding that adopts the merged-frame approach and exploits the inter-frame correlation at the same time has not been thoroughly investigated.

III. METHOD

A. Statistics of Perfect-Matched MB

A *perfect-matched MB* is one that has zero-valued MV and no residual. The reconstruction of such MB does not require the summation of the motion compensated (predicted) MB and the residual MB. For instance, a NOT-CODED MB in MPEG-4 [6] is a MB with zero-valued MV and no residual; hence a NOT-CODED MB is a perfect-matched MB. If a MB is a perfect-matched MB, the MB data read from the reference frame memory is the same as the MB data written to the reconstructed current frame memory in the PPFM. Since the perfect-matched MB would be read and written with the same content at the same location, there is an opportunity to eliminate the redundant memory access for a perfect-matched MB. To eliminate the repeat accesses for a perfect-matched MB, the content of the perfect-matched MB must be in the reconstructed frame memory before motion compensation. The only way to achieve this without extra memory access is to merge the reconstructed frame memory with the reference frame. Therefore, it is necessary to use the merged-frame approach to eliminate the memory accesses of a perfect-matched MB.

The percentage of perfect-matched MBs within a frame, which is denoted as P_0 here on, determines the amount of the memory access that can be eliminated. Table I lists the average P_0 of QCIF and CIF sized sequences. The corresponding PSNR and bitrate for each sequence is also provided in Table I. The statistics were gathered from running MPEG-4 VM18 [7] with the quantization parameter (QP) set to 16. The parenthesis next to each sequence represents the class it belongs as classified in [7]. Class "A" to "C" represents different levels of spatial detail and amount of movement, where class "A" is the lowest class and class "C" is the highest class. The statistics shows that lower class test sequences, such as *akiyo, container, mother_daughter, news*, and *hall*, have more than 70% of perfect-matched MBs in average. Other test sequences with more motion, such as *foreman, stefan, coastguard*, and *mobile*, have less than 30% of perfect-matched MBs.

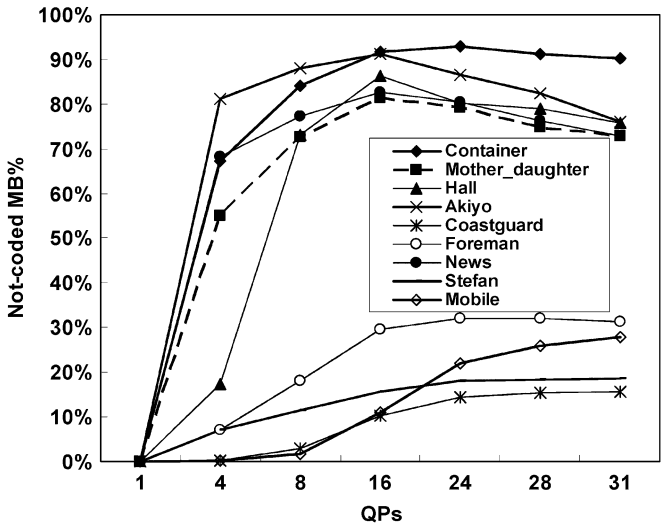

Fig. 1. Impact of different QP values on the percentage of NOT-CODED MB in MPEG-4 for QCIF sequences.

Fig. 1 illustrates the impact of different QP values on P_0 for the QCIF sequences. For most of the sequences with high P_0, the highest P_0 appeared when $\mathrm{QP} = 16$. However, for most of the sequences with low P_0, their P_0 were significantly increased until $\mathrm{QP} = 24$. After $\mathrm{QP} > 24$, the increase became insignificant. It could be that the reconstructed frame's quality would be extremely bad for $\mathrm{QP} > 24$ so that the residue became increasingly larger, and resulted the P_0 to decrease. Nevertheless, for the sequences with low P_0, mobile video applications should result in higher P_0 than those listed in Table I when $\mathrm{QP} > 16$ (Akiyo QCIF with $\mathrm{QP} = 16$, bit rate $= 115$ kb/s, $\mathrm{PSNR} = 32.28$ dB).

B. Combined Frame Memory

The CFM architecture adopts the merged-frame approach with an additional look-up table. Unlike the merged-frame approach in [2], [3], we introduced an additional look-up table to indicate whether the predicted pixel data are in the frame memory or in the local buffers. The CFM architecture consists of three major parts described as follows.

- Main frame memory (MFM): The MFM stores the reference frame data and the reconstructed frame data together. In which the upper part of the MFM stores the reconstructed current frame data, whereas the lower part of the MFM stores the reference frame data. The size of MFM is as large as one single frame, i.e., $176 \times 144 \times 1.5$ bytes for QCIF.
- Vector range strip buffer (VRSB): The VRSB is a rectangular strip of memory that works as an exchange buffer for the reference data. If one MB of reference data in the MFM is to be updated by a reconstructed current MB, this reference MB would be copied into the VRSB as a backup in case the subsequent MB needs it. This avoids the reference frame data from being ruined by the reconstructed current frame data. The size of the VRSB is determined by the height of the vector range and the width of a frame, i.e., $16 \times (176 + 16) \times 1.5$ bytes for QCIF with the vector range of $[-16 : +15]$ for both horizontal and vertical directions.
- Dirty table (DT): The DT is the look-up table that keeps record of which pixels in the MFM are updated. If a MB in the MFM is updated by the reconstructed current frame data, the corresponding dirty bits of this MB will be set. This indicates that the reference pixels in that MB are stored in the VRSB for backup as mentioned earlier. If the subsequent MB requires the reference pixels of this MB, these reference pixels will be read from the VRSB instead of the MFM. The size of the DT varies according to the size of the VRSB, i.e., $(176 + 16)/16$ bits for QCIF with the vector range of $[-16 : +15]$ for both horizontal and vertical directions.

Fig. 2 illustrates the flow chart of the CFMMC. When processing a perfect-matched MB, the CFMMC does not need any memory access since the reference MB and the reconstructed MB are the same and both reside within the MFM at the same location. The only operation carried out is the updating of the index in the DT. For the non-perfect-matched MB case, the CFMMC first checks the DT to determine where the predicted MB pixels are stored. Each pixel in the predicted MB is either read out from the MFM or the VRSB according to the corresponding dirty bit. After the predicted MB is read out, it is summed with the residual to reconstruct the reconstructed MB. Then the current reference MB in the MFM must be copied into the VRSB before the reconstructed MB is written to the same location. Finally, the reconstructed MB is written back to the MFM, and the DT and its index are updated at the end. Fig. 3 illustrates the motion compensation process for two consecutive non-perfect-matched MBs. The reconstruction computation [Fig. 3, Step1 (2)] is performed while the backup of the reference pixels [Fig. 3, Step2 (4)] is taking place.

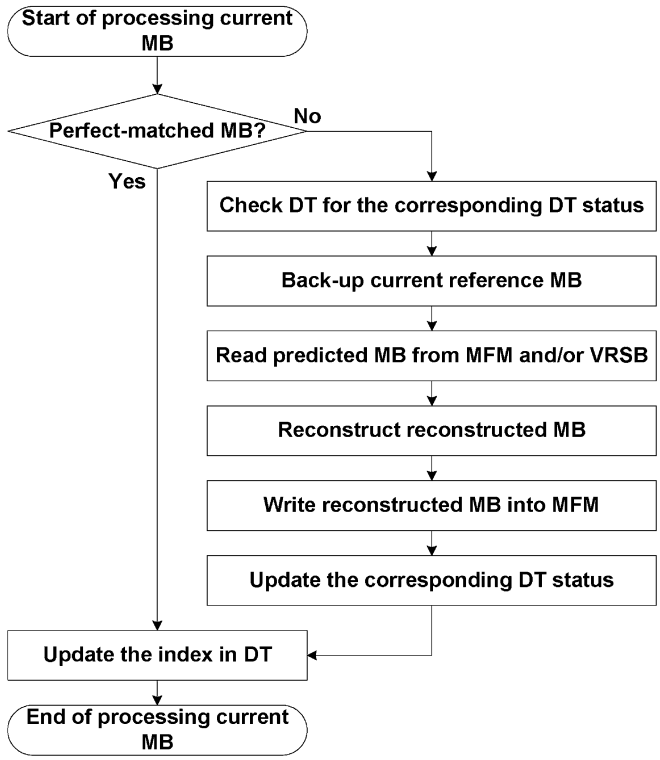

Fig. 2. Flowchart of motion compensation process in the CFMMC.

IV. Hardware Implementation

A. Architecture of the Combined Frame Memory Motion Compensation

The architecture of the CFMMC is illustrated in Fig. 4, which consists of five major parts. The first part includes the *mvprocessor*, the *pblk* and *inblk offsets generators*, and the *dirty table*. This part pre-processes the key information that is needed by the memory accessor, such as the motion vector of the chroma component, the *pblk offsets* (the offsets between the current block and the predicted blocks), the *inblk offsets* (the offsets of predicted pixels within the predicted blocks), and the dirty status. The second part is the *memory accessor*. This part plays the major role of generating the addresses to the MFM and the VRSB and multiplexing data among the memories and buffers. The third part is the *motion compensation controller* which coordinates the tasks among the modules and also interfaces the control signals. The fourth part is the *filter and reconstructor*, which generates the subpel samples and adds the predicted pixels with the residual pixels. Both the CFMMC and PPFMMC use the same filter and reconstructor design.

The memories of the MFM and VRSB are implemented using single-port SRAM [8] with 8-bit data width. The ratio between the energy consumption of the MFM and VRSB is defined as the k value. If k is larger, the energy reduction should be larger. This is likely to be the case when using DRAM for the MFM. In contrast, using SRAM for the MFM would yield smaller k value, thus reducing the energy reduction. The reduction amount of the SRAM MFM case can be considered as the lower bound of the energy reduction.

B. Comparison of Latency and Area

The latency of the CFMMC and PPFMMC hardware architectures are both dominated by the memory access time. The latencies of processing different MB modes are different. Take the MB modes in MPEG-4 Simple Profile for example. Table II

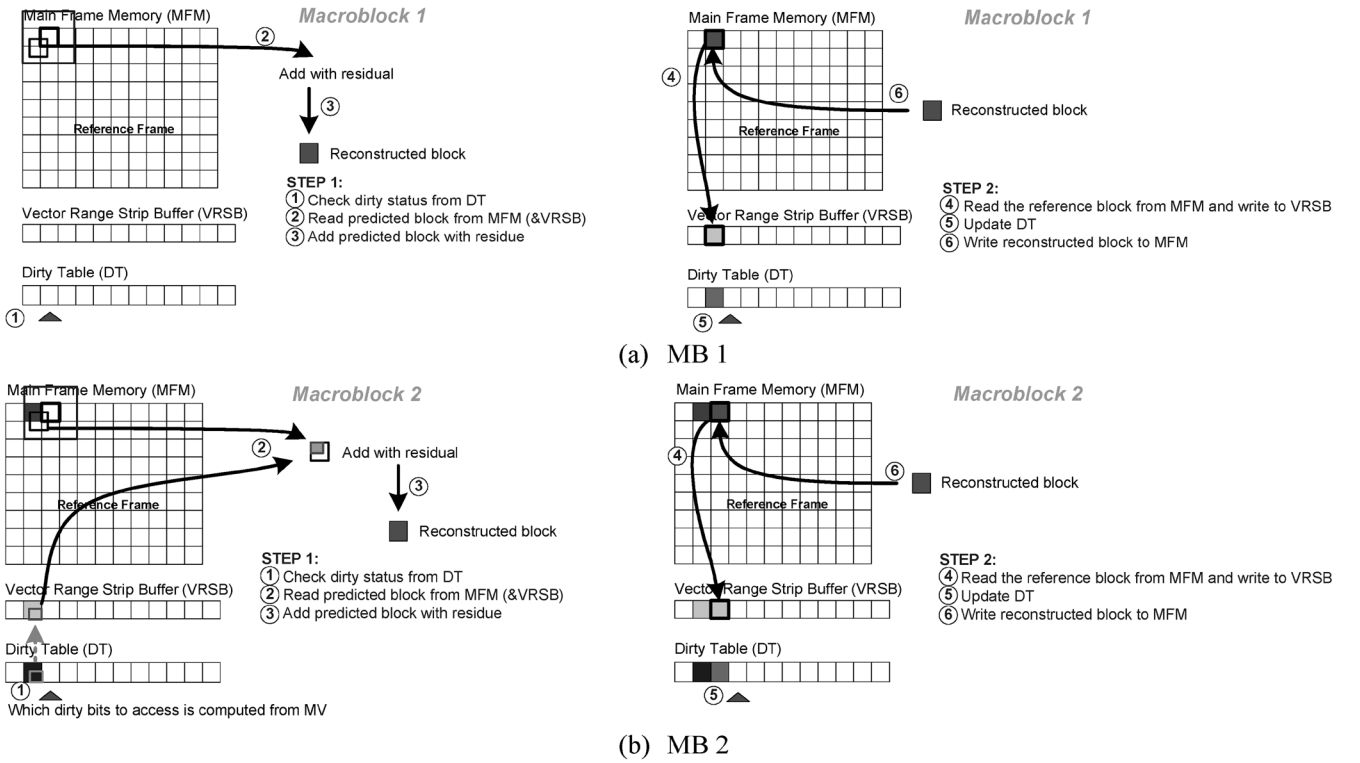

Fig. 3. Processing of nonperfect-matched MBs.

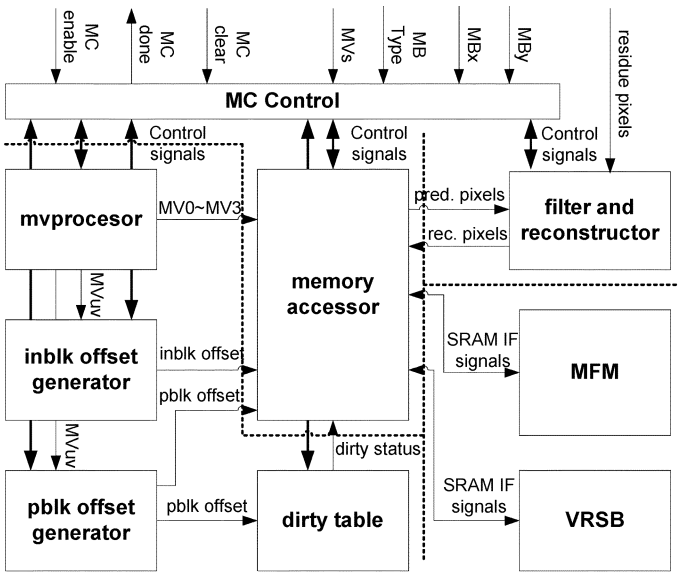

Fig. 4. Block diagram of the CFMMC hardware.

lists the definition of different MB modes and their processing latencies in the two prototypes.

The processing operations of INTRA MBs and INTER-INTRA MBs are similar. Both the CFMMC and PPFMMC check the MB mode ($C_{\mathrm{MB}} = 1$ cycle) first, which requires one cycle. Then they directly write the residue ($C_{\mathrm{WR\_RES\_MB}} = 384$ cycle) into the frame memory according to the MB's position. As a result, the total latencies of processing an INTRA MB in both architectures and processing an INTER-INTRA MB in the PPFMMC are the same, which is 385 cycles. To process an INTER-INTRA MB in the CFMMC, CFMMC must backup the collocated reference MB ($C_{\mathrm{BKUP\_REF\_MB}} = 384$ cycle) into the VRSB and update the dirty table ($C_{\mathrm{UPDATE\_DT}} = 1$ cycle). Consequently, the total latency of processing an INTER-INTRA MB is 770 cycles.

The latency to process an INTER or INTER4V MB in the CFMMC includes the latencies of identifying the MB mode, computing the chroma's motion vectors ($C_{\mathrm{CHROMA\_MV}} = 3$ cycle), reading the predicted pixels ($C_{\mathrm{RD\_REF\_MB}} = 384$ cycle), backup the reference pixels, and writing back the reconstructed pixels ($C_{\mathrm{WR\_REC\_MB}} = 384$ cycle). The reconstruction computation is performed while the backup operation is taking place. Therefore, the latency of processing a MB is 1,157 cycles. In contrast, the latency of PPFMMC does not include the latency of the backup operation and dirty table update. As a result, the total latency of processing an MB in the PPFMMC is 772 cycles.

For NOT-CODED MBs, the CFMMC does not perform any memory access; the total latency of processing a NOT-CODED MB only takes one cycle. The one cycle latency is spent to update the dirty table. On the contrary, the PPFMMC has to perform the same operations of reading and writing the ping-pong frame memory. Consequently, the latency of processing a NOT-CODED MB is exactly the same as that of processing an INTER or INTER4V MB, which is 772 cycles.

If P_0 is high enough, the latency of processing one frame may be reduced because of the latency reduction for NOT-CODED

TABLE II
LATENCIES OF DIFFERENT MB MODES

MB Modes	Description	PPFMMC Latency (cycles)	CFMMC Latency (cycles)
INTRA	Intra MB in I-frames	385	385
		$C_{MB\_MODE}+C_{WR\_RES\_MB}$	$C_{MB\_MODE}+C_{WR\_RES\_MB}$
INTER_INTRA	Intra-coded MB in P-frames	385	770
		$C_{MB\_MODE}+C_{WR\_RES\_MB}$	$C_{MB\_MODE}+C_{BKUP\_REF\_MB}+C_{WR\_RES\_MB}+C_{UPDATE\_DT}$
INTER	Inter MB with only 1 MV	772	1157
		$C_{MB\_MODE}+C_{CHROMA\_MV}+C_{RD\_REF\_MB}+C_{WR\_REC\_MB}$	$C_{MB\_MODE}+C_{CHROMA\_MV}+C_{RD\_REF\_MB}+C_{BKUP\_REF\_MB}+C_{WR\_REC\_MB}+C_{UPDATE\_DT}$
INTER4V	Inter MB with 4 MVs	772	1157
		$C_{MB\_MODE}+C_{CHROMA\_MV}+C_{RD\_REF\_MB}+C_{WR\_REC\_MB}$	$C_{MB\_MODE}+C_{CHROMA\_MV}+C_{RD\_REF\_MB}+C_{BKUP\_REF\_MB}+C_{WR\_REC\_MB}+C_{UPDATE\_DT}$
NOT-CODED	Inter MB with perfect-match	772	1
		$C_{MB\_MODE}+C_{CHROMA\_MV}+C_{RD\_REF\_MB}+C_{WR\_REC\_MB}$	C_{UPDATE_DT}

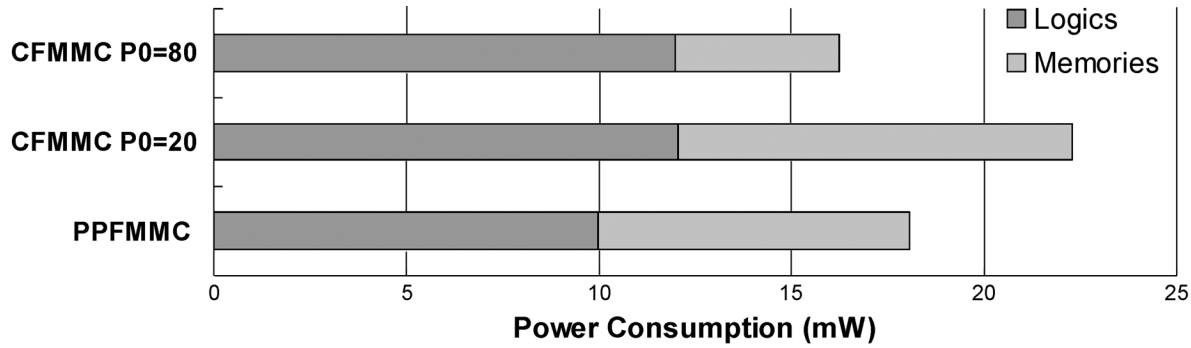

Fig. 5. Power consumption comparison of the PPFMMC and the CFMMC for P_0 =20% and 80%.

MBs; otherwise, the extra latency overhead for INTER_INTRA, INTER, and INTER4 V MBs would increase the overall latency.

The prototypes of the CFMMC and PPFMMC architectures are both synthesized from Verilog RTL design using UMC 0.18-μm 1P6M CMOS technology.[1] Both designs are synthesized with the clock constrained at 50 MHz, which is more than enough to perform real-time decoding even for CIF frame size. The memory size of the CFMMC is 42.3% smaller then that of the PPFMMC. On the other hand, the logics part's gate count of the CFMMC is 40 064, which is 42.9% larger than that of the PPFMMC. However, the total equivalent gate count of the CFMMC (including memories) is actually 37.7% smaller compared with that of the PPFMMC.

C. Simulation Setup

The prototypes are verified by running two types of test patterns. The first type is manually created with P_0 ranging from none to 90% with 10% step size. In this test pattern, the motion vectors are all set to zero and the residues are generated randomly. All the residues within a MB have the same value. The length of these test patterns is three QCIF frames with IPP video coding pattern. These test patterns can help evaluate the power consumption of video sequences with different P_0. The other type of the test pattern is the actual data from the video sequences used in the previous sections. Therefore, the MB types, the motion vectors, and the residues are all real data gathered from the video decoder. However, the length of these real test patterns is reduced to only 15 QCIF frames due to the file size limitation of the switching activity record. These 15 frames are excerpted from the first 15 frames of each video sequences.

[1]UMC Free-of-Charge Libraries. [Online]. Available: http://www.umc.com/english/design/b_3.asp

D. Architecture Energy Consumption Comparison

The gate-level power is reported by using Power Compiler [9]. The signal switching activities are gathered by running at 50 MHz for both the CFMMC and PPFMMC. The reason to use such a high clock rate is to increase the numerical order of the reported power, which corresponds to the energy of processing 109 CIF frames in one second. This can make the comparison of the energy consumption between the CFMMC and the PPFMMC easier.

Fig. 5 compares the energy consumption between the CFMMC and PPFMC running the manually created test patterns. The energy consumed by the logics and memory are shown for P_0 =20% and 80%. For the CFMMC, the memory's energy consumption is sensitive to P_0 while the logics part's energy consumption remains almost unchanged despite the different P_0 values. In contrast to the CFMMC's energy consumption being sensitive to P_0, the PPFMMC's energy consumption is independent of P_0.

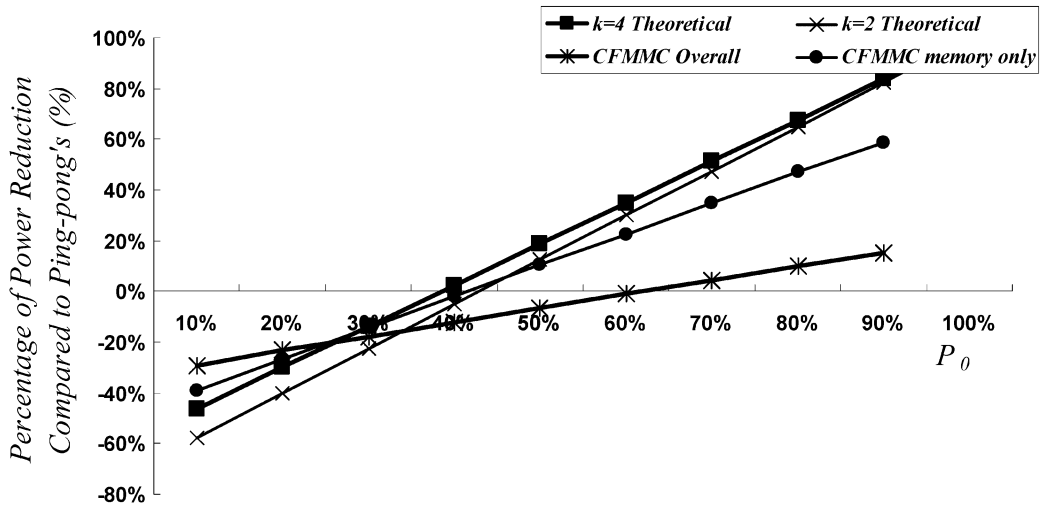

Fig. 6. Plot of the energy reduction percentages of the CFMMC at different P_0.

TABLE III
ENERGY REDUCTION PERCENTAGE OF THE REAL TEST PATTERNS

Test sequences	P_0 of the first 15 frames (%)	Energy reduction percentage (%) compared with the PPFMMC
container (A)	92.93	15.45
mother_daughter (A)	92.83	15.80
hall (A)	95.96	17.19
akiyo (A)	96.87	18.44
container (A)	92.93	15.45
coastguard (B)	34.95	-18.06
foreman (B)	29.39	-21.89
stefan (C)	15.66	-30.66
Mobile (C)	7.47	-31.67

Fig. 6 plots the CFMMC's percentage of energy reduction over different P_0 values. The percentage is defined as the ratio between the reduced energy and PPFMMC's energy. There are two lines related to the CFMMC architectures. One is the line for the memories themselves, the other one is for the overall CFMMC architecture. The line of the CFMMC architecture has smaller slope than the slope of the CFMMC memories line. This is because the energy consumed by the logics reduces the energy reduction. On the other hand, the line of the CFMMC memories has a slightly smaller slope compared with the slopes of the theoretical lines with k =4 and 2. The theoretical lines are derived based on memory access energy only. However, the logics and memories still consume energy even when there's no memory access, thus compromising the theoretical energy model.

The energy reduction percentages of the real test patterns are listed in Table III. The P_0 found in the first 15 frames of each test sequences are also listed. For those test sequences with P_0 >70%, the energy consumptions are reduced by 11%–18%. For sequences with much lower P_0, the energy consumptions are increased by 18%–32%.

V. CONCLUSION

We proposed a CFMMC architecture which is potential in reducing the energy consumption. The statistics on perfect-matched MB under different QPs and resolutions are investigated for the well known video sequences. We found that when the percentage of perfect-matched MBs (P_0) is higher than 50%. For these cases, the CFMMC is likely to reduce both the latency and the energy consumptions due to memory accesses.

The hardware implementation of the CFMMC only requires 62.3% of the silicon area used to implement the PPFMMC. The CFMMC architecture is also capable of reducing the energy consumption by up to 16% when P_0 >70%. However, the CFMMC suffers from energy consumption and latency increases when P_0 is not high enough. Consequently, these limitations limit the application of CFMMC into video surveillance, video telephony, and video conferencing. For these applications, the CFMMC shall guarantee its latency and energy reduction capability.

REFERENCES

[1] V. G. Moshnyaga, "Reducing energy dissipation of frame memory by adaptive bit-width compression," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, no. 8, pp. 713–718, Aug. 2002.
[2] F. Catthoor, "Low power storage exploration for H.263 video decoder," in *Proc. 4th Workshop VLSI Signal Process.*, 1996, pp. 115–124.
[3] L. Nachtergaele, "Low-power data transfer and storage exploration for H.263 video decoder system," *In IEEE Journal on Selected Areas in Communications*, vol. 16, no. 1, pp. 120–129, Jan. 1998.
[4] K. Denolf, "Memory centric design of an MPEG-4 video encoder," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 15, no. 5, pp. 609–619, May 2005.
[5] V. G. Moshnyaga, K. Masunaga, and N. Kajiwara, "A data reusing architecture for MPEG video coding," in *Proc. Int. Conf. Circuits Syst.*, May 2004, vol. 3, pp. 797–800.
[6] *Information technology—Coding of audio-visual objects*, ISO/IEC 14496-2, Dec. 2001.
[7] *MPEG-4 Video Verification Model version 18.0*, ISO/IEC JTC1/SC29/WG11 N3908, Jan. 2001.
[8] UMC 0.18 μm process high-speed single port SRAM generator user manual, Release 4.0. Artisan Components, Inc., Aug. 2000.
[9] Power Compiler User GuideRelease, W-2004.12. Synopsy Inc., Jan. 2005.