Frame Buffer Access Reduction for MPEG Video Decoder

Wei-Cheng Lin and Chung-Ho Chen

***Abstract*—Frame buffer power consumption and bandwidth requirement are two critical design issues in MPEG video decoders due to the overwhelming amount of frame data accesses. This paper proposes a reusable macroblock detector (RMD) which exploits the characteristic of a stationary macroblock (MB) to identify the reusable MBs in the frame buffer and avoid unnecessary data transfers. The RMD on average reduces about one quarter of the frame buffer accesses for 18 video sequences.**

***Index Terms*—Data reuse, frame buffer, low power, MPEG decoder, stationary macroblock.**

I. Introduction

THE MPEG video decoding places a great demand on the memory subsystem which can be a system performance bottleneck (i.e., memory wall problem) and the primary consumer of overall system energy. The memory is typically used as a bitstream buffer storing coded data and frame buffers holding decoded frames. The frame data dominate memory usage because of high video compression rate.

In frame buffer accesses, the current macroblock (MB) always overwrites an MB (i.e., victim MB) belonging to a previous frame. If these two MBs are exactly the same, the victim MB is completely reusable. This means that it is redundant to read the reference MB from the frame buffers and write the current MB into the frame buffers. To avoid these unnecessary memory accesses, this paper presents a reusable macroblock detector (RMD) that uses reference paths (from the current MB to the victim MB) passing through *reference-related MBs* to identify the reusable MB. In this paper, we focus on the MPEG-4 decoder and discuss how to extend the proposed method to H.264.

On average, the proposed design has obtained about 25% savings in frame buffer accesses. We also evaluate the performance improvement of the off-chip SDRAM memory due to the reduction in the frame buffer accesses. The proposed approach on average reduces about 24% of bandwidth utilization and saves about 23% in power consumption.

The remainder of this paper is organized as follows. Section II discusses related research in this field and specifies the problem. Section III presents the RMD algorithm and hardware architecture. Simulation results are discussed in Section IV. Finally, Section V concludes this paper.

Manuscript received June 05, 2007; revised November 26, 2007. First published September 29, 2008; current version published October 24, 2008. This work was supported in part by the National Science Council, Taiwan, under Contract NSC 94-2220-E-006-004 and by the Program for Promoting Academic Excellence of Universities in Taiwan. This paper was recommended by Associate Editor O. C. Au.

The authors are with the Department of Electrical Engineering and the Institute of Computer and Communication Engineering, National Cheng-Kung University, Tainan, Taiwan 70101 (e-mail: kevin@casmail.ee.ncku.edu.tw; chchen@mail.ncku.edu.tw).

Digital Object Identifier 10.1109/TCSVT.2008.2002830

II. Related Work and Problem Statement

A. Related Work

A number of recent researches employ embedded (on-chip) memory as the frame buffer [1], [2] to obtain power savings. Several management policies for embedded memory activation are proposed for energy efficiency; the frame buffer, however, is still responsible for approximately 40% of the total energy [2]. Moshnyaga [3] proposes a scheme to decrease the bit line transitions of the embedded frame buffer. This scheme saves energy by compressing the four most significant bits of image data, which have low switching activity between adjacent pixels due to spatial correlation.

Shih *et al.* [4] try to reduce frame buffer accesses by compressing the image data prior to storing them. These compression methods also reduce frame buffer size, but degrade the resulting image quality. Several recent researches for processor-based implementation improve cache performance [5] or use specialized on-chip software-controlled streaming memories [6] to optimize memory hierarchy usage and reduce external memory accesses.

B. Problem Statement

The MPEG-4 advance simple profile (ASP) [7] decoder including bidirectional prediction generally decodes a video stream with a fixed frame pattern (e.g., IPBBPBB...). However, if the coded bitstream is transmitted through real-time networks, some frames may be discarded at the server side, network nodes, or client side due to insufficient network quality or limited computing resource. B-frames, not being reference frames, have the highest priority to be skipped. Thus, the decoder may decode the video stream with an irregular frame pattern (e.g., IPBPPPB...). In this circumstance, the RMD should address three major issues: 1) specifying all possible reference paths; 2) recognizing the exact locations of the reference-related MBs; and 3) identifying the reusable MB before reading the reference MB.

On the other hand, when frame buffers serve both the decoding and display tasks, the minimum required number of the frame buffers is four [8]. If the decoding rate cannot always remain higher than the display rate, the number of frame buffers (NFB), however, must be increased to reduce the occurrence of frame buffer underflow. Thus, the RMD design must take into account that the NFB is not limited to four.

III. Reusable MB Detector (RMD)

A. Main Idea

The key idea of the RMD is to make use of the reference relationship between the current MB and the victim MB to compare their contents. The detector first discovers all possible reference paths constructed through the reference-related MBs

which exist in the *reference-related frames* with the same frame offset as the current MB. The reference-related frames include (a) all the anchor frames (P- and I-frames) between the current and victim frames based on the display order, and (b) the future reference frame of the current frame (if any). The RMD then tracks the stationary reference-related MBs according to their prediction direction (e.g., forward or backward) for each path to determine if the current MB is equal to the victim MB.

B. MB State Table and Frame State Table

1) MB State Table (MST): For each frame buffer, an MST, which contains as many entries as the number of MBs in a frame, is employed to keep track of whether the MBs stored in the frame buffer are stationary or not.

A stationary MB is one that is entirely identical to the colocated MB in the reference frame. For a B-frame, the stationary MB only exists in the forward or backward prediction mode. If the luma-residual, chroma-residual, and the forward (backward) 16×16 motion vector of an MB in the B-frame are all zero, this MB is referred to as the *forward (backward) stationary MB*. Hence, each decoded MB is classified as a forward, backward, or nonstationary MB.

To minimize the table size, only a direction bit D-bit (with two states: either forward or backward) is used to indicate the prediction direction, and two stationary bits L-bit and C-bit (each with two states: either stationary or nonstationary) are employed to record whether the luma and chroma data blocks are stationary or not for each entry of the table. Note that zero-residual information can be easily obtained by looking up the fields CBPC, CBPY, and CBPB in the MB header [7] without calculation.

2) Frame State Table (FST): We use an FST for each frame buffer to facilitate buffer management and assist the RMD in identifying the reference-related frames. The FST records the states of the corresponding frame by using the following three fields:

- Display_Available (DA): Indicates whether the frame has been displayed or not. The state "A" (available) stands for that the frame is not yet displayed, and the state "U" (unavailable) represents that the frame either has been displayed or the frame buffer is empty.
- Frame_Type (FT): Records the frame type (I, B, P), or holds its initial state "E" if the frame buffer is empty.
- Decoding_Order (DO): Shows the relative decoding order of the stored frames. A larger DO value corresponds to an earlier decoded frame. The DO value ranges from zero (initial value) to $\mathrm{NFB}-1$.

C. RMD Algorithm

This section presents the RMD algorithm which contains six functions (four in the decoding task and two in the display task). The display task starts to execute after the first two anchor frames are completely decoded; the communication between these two tasks is done through the FSTs.

When a new frame arrives for decoding, the *victim frame selection function* chooses an appropriate frame buffer to store this new frame. Then, the *reference-related frames identification function* points out the reference-related frames of which the corresponding MSTs will be looked up by the *reusable data detection function* to detect reusable MBs. After the frame is completely decoded, the *FST update function* updates the FSTs.

When the display task receives a request command for displaying a decoded frame, the *display frame selection function* selects a frame for output. After this frame is completely displayed and if there are no frames available for the next display (i.e., frame buffer underflow), the last displayed frame will be displayed again in the next display period. In this case, the *underflow detection function* does not update the FSTs.

In the RMD algorithm, the frames stored in the frame buffers are represented by a set $\mathbf{F} = \{\mathrm{F_i} | 0 \le \mathrm{i} \le \mathrm{NFB}-1\}$ where $\mathrm{NFB} \ge 4$. The six functions are presented as follows.

1) Victim Frame Selection Function: This function chooses a frame as the victim frame and detects frame buffer overflow. This function should ensure that the victim frame and its MST will not be referred by the future decoded frame. In other words, the reference-related MSTs (corresponding to the reference-related frames) must be kept in memory.

First, except the reference frame(s) of the current frame, all displayed and empty frames $(\mathrm{DA} == \mathrm{U})$ are classified into three sets $\mathbf{F^{UIP}}$, $\mathbf{F^{UB}}$, and $\mathbf{F^{UE}}$ according to frame types. Each frame in the three sets is qualified to be the victim frame. If no frame exists in the three sets, then a frame buffer overflow occurs. The decoder without the RMD can arbitrarily choose a frame from these three sets as the victim frame. However, if the RMD is enabled, victim frame selection must obey the following rules.

- Rule 1: $\mathbf{F^{UB}}$ has the highest victim priority because the B-frame is not the reference-related frame.
- Rule 2: $\mathbf{F^{UIP}}$ has a higher victim priority than $\mathbf{F^{UE}}$. This rule restricts the number of frame buffers used for the minimum requirement when the decoding rate is always higher than the display rate.
- Rule 3: If the candidate set is $\mathbf{F^{UB}}$ or $\mathbf{F^{UIP}}$, the oldest frame in the set is selected as the victim frame, denoted as F-v.

Under rules 1 and 3, the anchor frames between the current and victim frames based on the display order can be held in the frame buffers so all reference-related MSTs are held in the decoder.

2) Reference-Related Frames Identification Function: If the current frame is an I-frame or the victim frame buffer is empty, this function can be skipped since no reusable data exist in the victim frame buffer. Fig. 1 shows the pseudocode of this function. Two basic procedures, used in this function, are given as follows where $\mathbf{F'} \subseteq \mathbf{F}$:

- $\mathrm{MIN}(\mathbf{F'})$: Returns a frame which has the smallest DO value in the $\mathbf{F'}$.
- MID(J, K, $\mathbf{F'}$): Returns a set of frames whose DO values are between J and K, i.e., $\{F_\mathrm{i} \in \mathbf{F'} | \mathrm{J} \le \mathrm{F_i.DO} \le \mathrm{K}, 0 \le \mathrm{i} \le \mathrm{NFB}-1\}$.

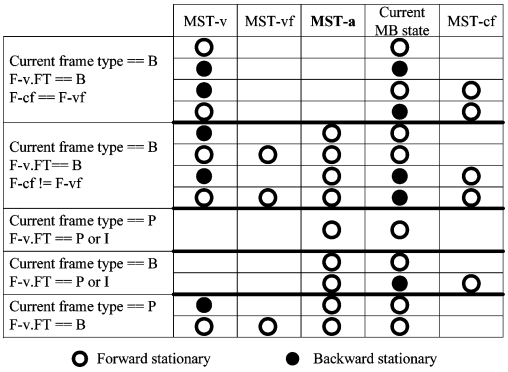
First, all anchor frames except the victim frame are collected into the set denoted as $\mathbf{F^{IP}}$. Then, the reference-related frames are extracted from $\mathbf{F^{IP}}$ and classified into the following three categories for different current frame types and victim frame types.

- F-cf is the future reference frame of the current frame and the last decoded anchor frame, that is, $\text{F-cf} = \mathrm{MIN}(\mathbf{F^{IP}})$.

```
01. Reference-related frames identification function {
02. switch(Current frame type, F-v.FT)
03.   case 'B, B' { F-cf = MIN(F^IP);
04.                 F-vf = MIN(MID(F-v.DO+1, NFB−1, F^IP));
05.                 if (F-cf == F-vf) F-a = Ø ;
06.                 else F-a = MID(F-cf.DO+1, F-v.DO−1, F^IP); }
07.   case 'P, P (or I)' { F-a = MID(0, F-v.DO−1, F^IP); }
08.   case 'B, P (or I)' { F-cf = MIN(F^IP);
09.                 F-a = MID(F-cf.DO+1, F-v.DO−1, F^IP); }
10.   case 'P, B' { F-vf = MIN(MID(F-v.DO+1, NFB−1, F^IP));
11.                 F-a = MID(0, F-v.DO−1, F^IP); }}
```

Fig. 1. Identification and classification of reference-related frames.

TABLE I
CONDITIONS TO REUSE THE VICTIM MB



- F-vf is the future reference frame of the victim frame and the last decoded anchor frame prior to decoding the victim frame, i.e., $\text{F-vf} = \text{MIN}(\text{MID}(\text{F-v.DO} + 1, \text{NFB} - 1, \mathbf{F^{IP}}))$.
- $\mathbf{F\text{-}a}$ contains the anchor frames between the current frame and victim frame based on the display order, but does not include F-vf. In the case "B, B," if the victim frame and current frame have the same future reference frame $(\text{F} - \text{cf} == \text{F–vf})$, this means no anchor frames exist between these two frames $(\mathbf{F\text{-}a} = \emptyset)$.

3) Reusable Data Detection Function: This function looks up MST-v, MST-cf, MST-vf, and $\mathbf{MST\text{-}a}$ (corresponding to F-v, F-cf, F-vf, and $\mathbf{F\text{-}a}$) to check if the victim MB is reusable. The number of the tables in $\mathbf{MST\text{-}a}$ ranges from zero to NFB–2. Table I shows the conditions to reuse the luma (chroma) block of the victim MB for different victim frame types and current frame types. As an example, in the first row, if the victim MB and the current MB are both forward stationary, then these two MBs and the colocated MB in the previous reference frame are the same. In this case, the victim MB is reusable. After the detection is done, the colocated entry in the victim MST is updated to the current MB state.

4) FST Update Function: This function updates FSTs after the current frame is completely decoded, as illustrated in Fig. 2. The DO values of the frames which are decoded later than the victim frame should be increased by one since the victim frame has been overwritten by a new frame, or all the DO values of the used frames $(\text{F}_\text{i}.\text{FT} ! = \text{E})$ should be increased by one if the victim frame buffer is empty before storing the new frame.

```
01. FST update function {
02. for (i=0 ; i<NFB ; i=i+1) {
03.   if (F_i.FT != E && (F-v.FT == E || F_i.DO < F-v.DO)) F_i.DO = F_i.DO+1; }
04. F-v.(DA, FT, DO) = (A , current frame type, 0); }
```

Fig. 2. FST update.

5) Display Frame Selection Function: This function chooses a frame as the display frame denoted as F-d, obeying the display order. First, all frames available for display are collected into the set $\mathbf{F^A}$. $\mathbf{F^A}$-1(F^A-2) represents the first (second) decoded frame in F^A.F^A-1 and is generally used as the display frame. However, if F^A-2 is a B-frame and F^A-1 is an anchor frame (i.e., the future reference frame of $\text{F}^\text{A} - 2$), then $\text{F}^\text{A} - 2$ is used as the display frame because the B-frame is decoded later than its future reference frame but should be displayed earlier.

6) Underflow Detection Function: This function detects frame buffer underflow and updates FSTs after F-d is completely displayed. The underflow is detected by the following methods.

- If the number of the frames in $\mathbf{F^A}$ is two and the current frame is a B-frame (one of the frames in $\mathbf{F^A}$ is the future reference frame and the other is the last displayed frame), then frame buffer underflow occurs because the next display frame is the current frame.
- If the number of the frames in $\mathbf{F^A}$ is one, frame buffer underflow occurs.

Once frame buffer underflow occurs, the FSTs hold its current state without updating. The last displayed frame thus will be chosen again. Otherwise, F-d.DA is set to 'U.'

D. Extension of the RMD to H.264

The proposed approach can be applied to other standard video decoders, such as MPEG-2, H.263, and H.264. Since H.264 adopts multiple reference frame motion estimation and an in-loop deblocking filter to improve coding performance, an MB deemed to be stationary in this standard should satisfy the following three conditions: 1) this MB has a zero-16 $\times$ 16 motion vector and zero-residual; 2) the reference frame chosen is the nearest reference frame(s); and 3) the in-loop deblocking filter does not alter this MB.

E. Hardware Implementation

Fig. 3 depicts a block diagram of the RMD. Four frame buffers are assumed to be used in the decoder. MSTs are stored in the external memory to reduce on-chip memory used. Accessing the MSTs consumes additional bus bandwidth; this overhead, however, is negligible because only 12 bits of MST data at most are transferred for identifying a reusable MB when the NFB equals four.

Two on-chip register files (i.e., RFA and RFB) serving as a ping-pong buffer cache a portion of the MST data in order to tolerate external MST access delay. A register file consists of four columns corresponding to four MSTs and each column contains eight entries. An entry is composed of three bits (L, C, and D). The L (or C or D) bits in the same column work as a shift register while an MB is decoded. Registers "v," "cf," "vf," and "$\mathbf{a}$" are utilized to indicate F-v, F-cf, F-vf, and $\mathbf{F\text{-}a}$, respectively; the

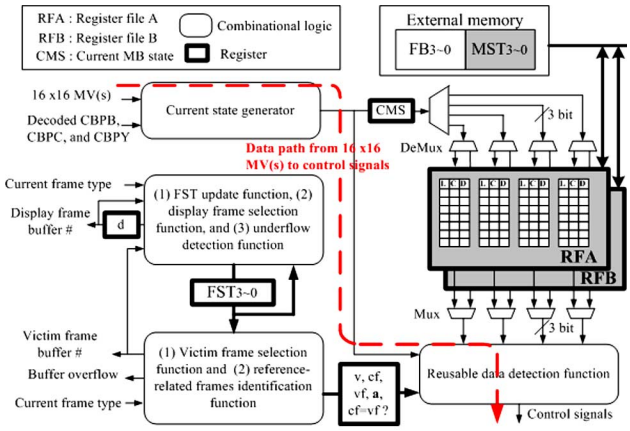


Fig. 3. RMD architecture.

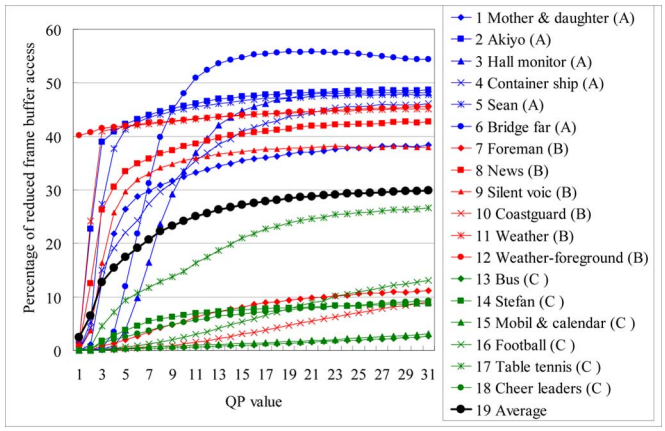


Fig. 4. Percentage of reduced frame buffer accesses versus QP.

corresponding MSTs are loaded into the register files for use in the reusable data detection function.

When a 16×16 motion vector is decoded, the motion vector is translated to a memory address, and then a bus request is issued with this specified address for reading a reference MB in the next clock cycle. The RMD should generate the control signals before the reference MB is read out from the frame buffer. This can be achieved because: 1) CBPC, CBPY, and CBPB fields are decoded earlier than the motion vector according to the MB header structure; 2) the required MST data are already ready in the buffer; and 3) the time delay of the path from the motion vector input to the control signals output (1.63 ns under 0.25-μm CMOS process, $\mathrm{NFB} = 6$) is very short. Thus, the decoder does not take extra clock cycles to wait for the control signals and the bus request for redundant memory accesses can be avoided.

The MPEG-4 ASP video decoder and the RMD are implemented with 0.25-μm CMOS standard cell technology. The decoder hardware complexity is 125 K equivalent gates. The RMD hardware cost is 2.9 K, 4.4 K, and 5.0 K equivalent gates when the NFB equals four, five, and six, respectively. The power consumption of the decoder and RMD is estimated using the PrimePOWER. The decoder consumes 94.92 mW whereas the RMD consumes 1.87 mW $(\mathrm{NFB} = 6)$.

IV. Simulation Results

In this study, we used an MPEG-4 advance simple profile at level 3 (ASP @ L3) video codec to evaluate the RMD performance. The motion estimation search range in the encoder is ± 16 and the Intra-period has 30 frames. The frame pattern is IBBPBBP and the NFB is four. Eighteen video sequences used for experiments are classified into three classes A, B, and C from low to high according to the spatial frequency and the amount of movement [7], as shown in Fig. 4. The frame size of *Football*, *Sean*, *Weather*, *Weather-foreground*, and *Table tennis* is 352×240 pixels, while the rest is 352×288 pixels. Performance comparisons are done on the first 150 frames of *Bus* and *Cheer leaders*, 120 frames of *Football*, and 300 frames of the others.

The decoder works at 27 MHz clock with a 16 Mb SDRAM [9]. External data bus width is 16 bits. In this experiment, the SDRAM row-activation delay, CAS latency, precharge delay, and minimum period of burst refresh commands are one, two, one, and two cycles, respectively. Table II shows the current parameters of the SDRAM. The power consumption of the SDRAM can be calculated by the following equation:

TABLE II
Current Parameters of SDRAM

Operating current (burst length =1) (I_1)	95 mA
Precharge standby current (power down) (I_2P)	2 mA
Precharge standby current (non power down) (I_2N)	15 mA
Active stanby current (non power down) (I_3N)	25 mA
Operating current (burst mode, read or write) (I_4)	130 mA
Refresh current (I_5)	90 mA

$$\mathrm{Power} = \frac{V}{T} \times (I_1 \times 5 \times \mathrm{RA} + I_4 \times (\mathrm{BT} - \mathrm{RA}) + I_2P \times \mathrm{PD} + I_5 \times 2 \times \mathrm{RC}) \times t_{\mathrm{CLK}}$$

where T, V, and t_{CLK} are the total working time of the SDRAM, voltage applied to the device (3.3 v), and clock period (37 ns for 27 MHz), respectively, whereas RA, BT, PD, and RC denote the number of row-activations, burst transfers, clock cycles spent in power down mode, and refresh commands, respectively. The four terms express the induced current of: 1) one burst length data access with a row-activation (activate-precharge pair); 2) burst accesses without the row-activation; 3) power down state; and 4) refresh operation, respectively. For low-power consideration, the SDRAM enters the power down-mode immediately after an MB is completely accessed and the bank is precharged. The active row is precharged immediately after the access is completed (closed page policy). Thus, the power consumption caused by the current I_2N and I_3N is ignored.

Fig. 4 shows the percentage of reduced frame buffer accesses and the impact of different QP values for the 18 video sequences. The RMD is especially effective for low-motion videos, saving about 46% frame buffer accesses for class A on average $(\mathrm{QP} = 16)$. When the QP value changes from 1 to 15, the increase of savings is significant. However, when the QP value is larger than 15, the increase is diminished. This is due to the fact that a higher QP value makes the image blurrier so the difference between the reference frame and original (unencoded) frame is increased,

TABLE III
PERCENTAGE OF REDUCED FRAME BUFFER ACCESSES AT DIFFERENT BIT RATES

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
LB	128	128	128	128	128	128	256	128	128	256	128	128	512	512	384	512	384	512	256
P	31.1	42.8	42.1	35.8	43.7	31.7	10.3	38.5	35.9	6.5	45.1	40.3	2.6	8.7	2.7	8	22.3	9.3	25.4
HB	256	256	256	256	256	256	512	256	256	512	256	256	1024	1024	768	1024	768	1024	512
P	24.0	37.9	30.9	26.8	40.1	23.4	5.4	33.6	31	2	39.1	36.5	0.7	3.5	1.8	3.3	14.4	7.5	20.1

LB: Low bitrate (kbps) HB: High bitrate (kbps) P: Percentage of reduced frame buffer accesses

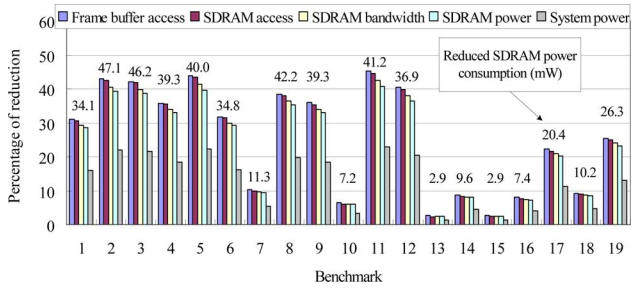


Fig. 5. Percentage of reductions in frame buffer accesses, SDRAM accesses, SDRAM bandwidth usage, SDRAM power, and system power.

and therefore the tendency to increase the amount of zero-residuals is weakened.

The sequences of *News, Silent voice, Weather*, and *Weather-foreground* have more reusable MBs than those of *Foreman* and *Coastguard* despite that they belong to the same class. We examine these sequences and find that the camera taking the former four videos is still while the camera taking the latter two videos is in motion. Thus, it is clear that decoding a video, taken by a still camera with a large portion of static background, can bring many more reusable MBs.

Table III shows the percentage of reduced frame buffer accesses for the 18 video sequences that are encoded with a target bit rate at 30 fps by the encoder enforcing rate control in the MB layer. The high bit rate is set at twice that of the low bit rate, which is set based on video content complexity. The average percentage of the reduced frame buffer accesses is 25.4% for low-bit-rate cases. The higher the bit rate is, the fewer frame buffer accesses are saved. This is because smaller QPs, decided by rate control, are used in the encoder.

Fig. 5 shows the reductions in frame buffer accesses, SDRAM accesses, SDRAM bandwidth usage, SDRAM power, and system power (including both the SDRAM power and decoder power) for a low-bit-rate setting. The percentages of reduced frame buffer accesses and reduced SDRAM accesses are quite similar since the data transfers of the MST data and coded bit-stream are very small compared with that of the decoded frames. The percentage of reduced SDRAM bandwidth usage is higher than that of reduced SDRAM power. This is because the SDRAM must consume background power even without memory access.

The benchmarks of *Akiyo* (2), *Hall monitor* (3), *Sean* (5), and *Weather* (11) have achieved around 40% savings in SDRAM power. For the video sequence *Bus* (13), the RMD only saves 2.9 mW of the SDRAM power; this savings, however, is still larger than the RMD power consumption (1.87 mW). The results show that the system power, SDRAM power, and SDRAM bandwidth usage can be reduced, on average, by 13.0%, 23.3%, and 24.0%, respectively.

V. CONCLUSION

This paper proposes a resuable MB detector to reduce redundant frame buffer accesses for the MPEG-4 ASP video decoder. The key challenge for the proposed RMD is to find out all of the reference-related frames in irregular frame patterns and to ensure that they are all stored in the frame buffer. On average, the detector can eliminate 25.4% of the frame buffer accesses in a low-bit-rate setting.

REFERENCES

[1] T. Hashimoto, M. Ohashi, M. Matsui, S. Kuromaru, T. Mori-iwa, M. Hamada, Y. Sugisawa, H. Tomita, M. Hoshino, T. Nakamura, K. Ishida, K. Watada, T. Fukunaga, and J. Michiyama, "A 27-MHz/54-MHz 11-mW MPEG-4 video decoder LSI for mobile applications," *IEEE J. Solid-State Circuits*, vol. 37, no. 11, pp. 1574–1581, Nov. 2002.

[2] C.-W. Yoon, R. Woo, J. Kook, S.-J. Lee, K. Lee, and H.-J. Yeo, "An 80-/20-MHz, 160-mW multimedia processor integrated with embedded DRAM, MPEG-4 accelerator, and 3D-rendering engine for mobile applications," *IEEE J. Solid-State Circuits*, vol. 36, no. 11, pp. 1758–1767, Nov. 2001.

[3] 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.

[4] C.-W. J. Shih, N. Ling, and T. Ogunfunmi, "Memory reduction by Haar wavelet transform for MPEG decoder," *IEEE Trans. on Consumer Electron.*, vol. 45, no. 3, pp. 867–873, Aug. 1999.

[5] Z. Xu, S. Sohoni, R. Min, and Y. Hu, "An analysis of cache performance of multimedia applications," *IEEE Trans. Comput.*, vol. 53, no. 1, pp. 20–38, Jan. 2004.

[6] A. Ramachandran and M. F. Jacome, "Energy-delay efficient data memory subsystems," *IEEE Signal Process. Mag.*, vol. 22, no. 5, pp. 23–37, May 2005.

[7] *MPEG4 Video Verification Model Version 18.0*, ISO/IEC JTC1/SC29/WG11, Jan. 2001.

[8] D. Isovic, G. Fohler, and L. Steffens, "Real-time issues of MPEG-2 playout in resource constrained systems," *J. Embedded Computing*, vol. 1, no. 2, pp. 239–256, Dec. 2005.

[9] *"Samsung SDRAM 16 Mb H-Die (x16) Data Sheet,"* Aug. 2004, Part NO. K4S161622H.