Reducing energy dissipation of frame memory by adaptive bit-width compression

Moshnyaga, Vasily G.
Department of Electronics Engineering and Computer Science, Fukuoka University

https://doi.org/10.15017/7656

出版情報：IEEE Transactions on Circuits and Systems for Video Technology. 12 (8), pp.713-718, 2002-08. IEEE
バージョン：

Transactions Letters

Reducing Energy Dissipation of Frame Memory by Adaptive Bit-Width Compression

Vasily G. Moshnyaga

***Abstract*—In this paper, we propose a new architectural technique to reduce energy dissipation of frame memory through adaptive bitwith compression. Unlike related approaches, the technique utilizes the fixed order of memory accesses and data correlation of video sequences by dynamically adjusting the memory bitwidth to the number of bits changed per pixel. Instead of treating data bits independently, we group the most significant bits together, activating the corresponding group of bit lines adaptively to data variation. The approach is not restricted to specific bit patterns, nor does it depend on the storage phase. It works equally well on read and write accesses, as well as during precharging. Simulations show that in using this method, we can reduce the total energy consumption of frame memory by 20% without affecting the picture quality. The implementation scheme is simple yet compact.**

***Index Terms*—Bit compression, frame memory, low-power design, video encoders.**

I. Introduction

A. Motivation

With the growing popularity of battery-operated video applications, such as portable video phones, hand-held DVD devices, mobile video conferencing units, etc., reducing energy consumption of video encoders becomes a prime design requirement. In a modern video encoding system such as MPEG2 (Fig. 1), frame memory stores a reconstructed image of the reference frame to be used in motion estimation to eliminate the temporal redundancy of current image frame. Due to large frame sizes, the frame memory has the highest capacitance in the system, dominating its energy dissipation. Although there has been a significant progress in low-power memory design, the frame memory alone still accounts over 35% of the total energy [1]. Therefore, it must be optimized for low energy consumption as much as possible.

To first order, the energy dissipation of a frame memory may be expressed as the sum over all memory accesses (acc) of the energy per access (E_i), or

$$\text{energy} = \sum_{i=1}^{\text{acc}} E_i.$$

Manuscript received December 2001; revised April 10, 2002. This work was supported in part by The Ministry of Education, Culture, Sports, Science and Technology of Japan and by Mitsubishi Electric Corporation.

The author is with the Department of Electronics Engineering and Computer Science, Fukuoka University, Fukuoka, Japan (e-mail: vasily@fukuoka-u.ac.jp).

Publisher Item Identifier 10.1109/TCSVT.2002.800858.

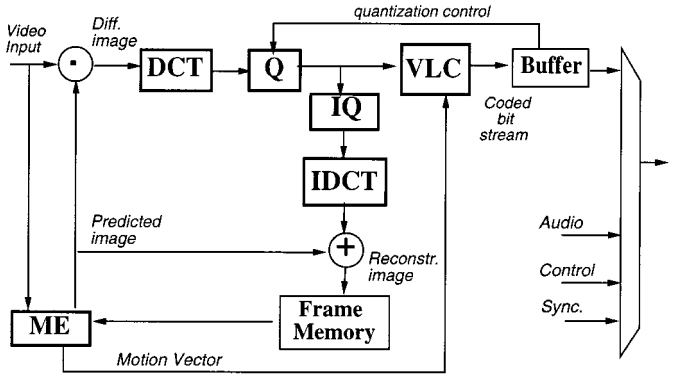

Fig. 1. MPEG2 video coding scheme.

Typically, frame memory consists of two or four synchronous 16-Mbit DRAM units [2]. In these DRAMs, most of the energy per access (E_i) is burned when driving the high capacitance of the bit lines, which are heavily loaded with multiple storage cells [3] and so require a large energy for charging. A single bit-line of a 16-Mbit DRAM, for example, takes as much as three orders of magnitude more power than 1-bit adder [4]. The number of bit lines simultaneously activated per access traditionally equals the access bitwidth m_i. Thus, given the average physical capacitance of a bit line (C) and the supply voltage (V), the frame memory energy dissipation can be expressed by

$$\text{energy} = \sum_{i=1}^{\text{acc}} m_i \times C \times V^2. \tag{1}$$

Although reducing the energy amounts to all factors (acc, m_i, C, V), the saving obtained by lowering the number of memory accesses (acc) and the bit width per access (m_i) is fairly independent of integration technology and is less expensive.

In this letter, we focus on the problem of adaptive reduction of the frame memory bitwidth and present a novel architectural technique, which exploits data correlation in image sequences to save energy.

B. Related Research

Recently, a number of approaches to low-energy memory design have been reported. Many of them have concentrated on technology and circuit optimizations which can reduce bit-line energy (e.g., cell structure modification, charge recycling, limited bit-line swing, data retention circuitry, etc. [5]). At the

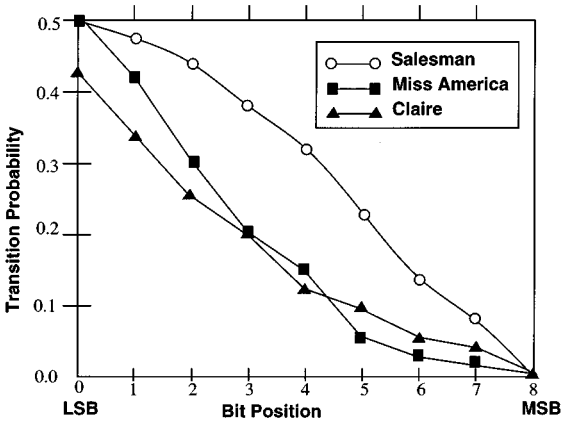

Fig. 2. Activity profile for memory input.

architectural level, research has focused on transition activity reduction through multiple memory splitting into sub-banks, bit-line segmentation, selective line activation [6], DRAM/logic integration [7], [8], and application-specific data-flow transformations [9].

Despite differences, most of these methods do not consider data to be stored, applying the same control/addressing pattern regardless of data variation on the memory input. In multimedia applications, however, data are highly correlated and distributed unevenly. Fig. 2 shows the transition activity profile on the frame memory input for three standard video sequences. Due to high image correlation, the most significant bits (MSBs) clearly have much lower switching activity than the least significant bits (LSBs). Leveraging this bit uneveness by storing the minimal number of changed bits can significantly reduce energy consumption.

Although there have been many attempts to exploit this data asymmetry for processing elements [10]–[12], data, and address busses [13]–[16], only a few works have targeted memory units. Park *et al.* [17] proposed a scheme that uses data variation to diminish bit-line discharging during precharging. The method adds an extra line to each memory byte to indicate if the datum being stored is the true or complement of the intended one. Depending on the precharging bit value ("high"or "low"), a decision is made to store either the true or complemented value on the memory, so as to minimize the number of bits different from the precharging value. Villa *et al.* [18] proposed to dynamically disable the local word line for each zero byte. In this method, an extra bit is attached to each byte to indicate whether the byte contains all zero bits. The method not only allows us to prevent bit-line discharging for each byte when the control bit is set, but also to shrink the memory access to only one bit if the byte is zero. The dependency on the zero bytes, however, severely limits its application to cache memories of general-purpose processors.

C. Contribution

This paper proposes a new approach for reducing the bit-line transition activity in frame memory. Unlike previous techniques, the method exploits the pixel correlation in video sequences dynamically adjusting the memory bit-width to the number of bits changed per pixel. Neither it is restricted to specific bit-patterns (e.g., precharging value or zero bytes) nor to the operation mode. The method works efficiently on read, write, and precharging accesses. Simulation results show that using this method, we can reduce the total memory energy by more than 20% without affecting the picture quality.

	Raw data	#Bits		Stored data	#Bits
5	00101010	8	0	1010	5
4	00101110	8	1	00101110	9
3	11111010	8	0	1010	5
2	11110110	8	0	0110	5
1	11110011	8	1	11110011	9
	Total:	40	EB	Total:	33

Fig. 3. Representation of data compression.

The paper is organized as follows. The next section presents the approach and outlines its implementation. Section III analyzes the performance. Conclusions are drawn in Section IV.

II. Adaptive Bit Width Compression

A. Main Idea

The solution we propose to minimize the power dissipated by the frame memory of a video encoding system belongs to the category of a bit-compression techniques. In particular, we target the reduction of the number of transitions occurring on the bit lines. The proposed approach is based on observation that in video encoders, frame memory is accessed in a fixed order, namely by blocks. Although the actual memory access pattern may be different for different systems (e.g., row-first, or column-first) fashion, the memory-read and the memory-write operations are performed in the same order. Due to the block-based feature of standard MPEG encoding, the order of reading the picture blocks from the memory equals the order of the block writing. Futher, a block consists of $N \times N$ picture elements or pixels. Because of large pixel correlation inside the block, adjacent pixels differ by a few bits. So, activating the whole bit-width with each memory access seems to be unnecessary. Moreover, it is highly expensive from the energy perspective, since precharging of all the lines in a bit-width and then eventually discharging of those, which are connected to zero bits, dissipates large energy.

The key idea of our approach is to compare the pixel data Y written to memory during the last access with that of the current pixel data X. If their k MSBs are equal in pairs, we omit writing the MSBs, storing into the memory only the $n-k$ LSBs of X and the equality bit (EB) that indicates whether the pixel bit-width is compressed. Formally, the EB is defined as

$$EB = \begin{cases} 0, & \text{if } \mathrm{AND}_{i=0}^{i=k} x_i \odot y_i \& c_{i-1} \\ 1, & \text{otherwise} \end{cases}$$

where $c[0] = 1$ and $\odot$ is the equivalence operator defined by the algebraic function $F = ab + a'b'$.

When the MSBs are not equal, all n bits of X and the EB are written. On a read access, we prevent bit-line discharge by disabling the sense-amplifiers of the k MSB-lines when the EB is set to zero. If EB is one, the n data bits are read normally. Thus, instead of accessing the fixed n-bit width, we use an adaptive one, whose length depends on input data variation.

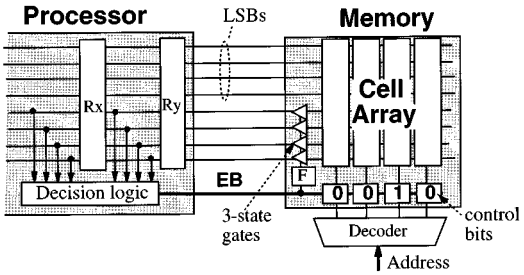

Fig. 4. Illustration of the proposed approach.

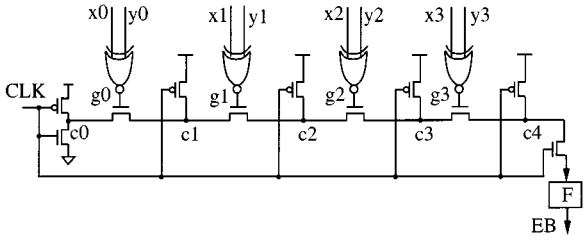

Fig. 5. Decision logic circuitry.

Fig. 3 illustrates how the scheme works, assuming that $k = 4$ and the data are written to the memory sequentially according to the numbers depicted on the left. As figure shows, only two words (first and fourth) of the stream requires full $n + 1$-bit representation, while all the others are stored in the compressed form. Numbers at the bottom indicate the total number of bits which have to be activated to store the data stream in the memory.

B. Implementation Scheme

1) Overview: Fig. 4 outlines the implementation scheme on example of the 8-bit-wide memory ($k = 4$). (Note that multibyte memory access can be easy implemented by applying the scheme at the byte (i.e., pixel) granularity). On the processor side, the scheme includes two data registers Rx, Ry and the decision logic; on the memory side, it involves an extra column of memory cells and a flip-flop F. On the write access, the decision logic compares the incoming data and the outcoming data of register Rx, setting the EB signal to zero if they have four MSBs equal in pairs. The zero EB sets the flip-flop F to disable the bit-lines of the four most-significant bits, thus reducing the memory access to the four LSBs of Ry and the EB. Otherwise, all bits of Ry and the EB are written. Note that signal EB, generated in regards to Rx, is written to the memory jointly with the value of Ry; that is one step ahead. Such a "write-ahead" procedure allows us to disable the MSB bit-lines before storing the Rx value into the memory and save energy on writing, as well as on precharging.

On a read access, the zero EB-bit, which is read at step t, sets the flip-flop F to deactivate the MSB lines at the next step, $t + 1$. Thus, when the EB is zero, only the LSBs are read from the memory; the MSBs are taken from the register Ry.

2) Decision Logic: Fig. 5 depicts the internal circuitry of the decision logic. In this figure, $y0 - y3$ are the flip-flops to store the MSB of the last word written to memory.

The operation proceeds as follows. When CLK is low, the nodes $c1, c2, c3,$ and $c4$ are precharged to V_{dd}. During computation, when CLK goes high, the n pull-down transistor on the left and the n pull-up transistor on the right turn on. Upon comparison of input bit signals in pairs, each XNOR generates $g_i, (i = 0, 1, 2, ..)$, making the low potential from the left to propagate through the transistor chain to the right. If x_i and y_i are equal, then g_i is high; in this case, the node c_{i-1} is connected to the node c_i, conditionally discharging it. Note that the output signal (EB) will be set to zero if and only if all the nodes $c_i, i = 0, .., 4$ are discharged; that is, all four bits of X and Y are the same. The F in the figure denotes the flip-flop.

As Fig. 5 shows, the implementation is regular and simple. The delay introduced by the decision logic is quite small. According to HSPICE simulations (NTT NEL library, 0.5-μm CMOS, $V_{dd} = 3.3$ V), it is equal to a four-input AND gate delay.

3) Memory Circuit Modification: The primary modification to the memory circuitry is to add the EB-indicator bit for each memory byte and a flip-flop (F) to each 8-bit memory bank, as shown in Fig. 6. During the memory write, the decision logic checks whether the MSBs of the current pixel equal the MSBs of the last pixel written to the memory and if so, it nulls the EB_j, setting the flip-flop (F) to zero and thus deactivating both the sense-amplifiers and the precharging circuitry for the lines $b_0 - b_3$). Otherwise, it sets the EB_j to one and writes the eight data bits normally. On a memory read access, both the sense amplifiers and the precharging circuits of the MSB lines will be turned on only if the EB_j is not zero, which is indicated by the flip-flop's storage output.

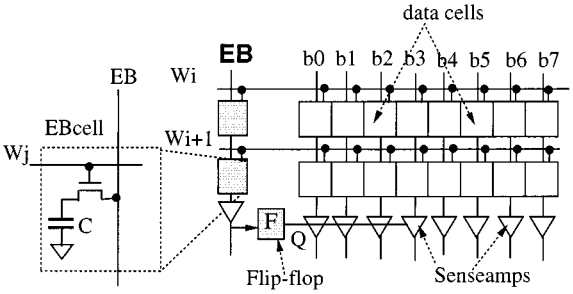

Fig. 6. Modified memory byte structure.

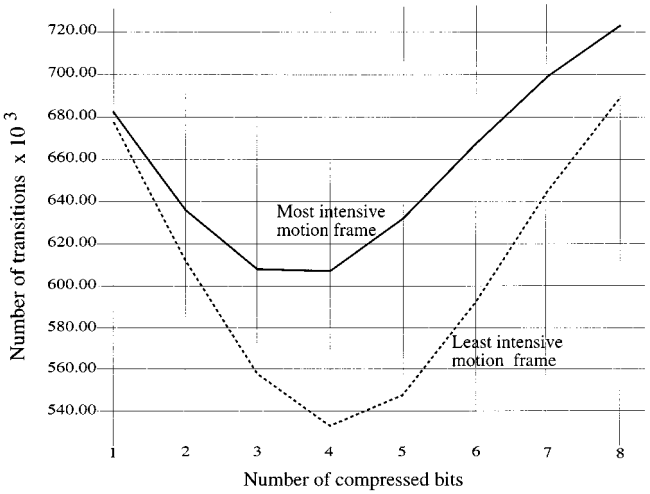

Fig. 7. Bit-line transitions per frame against the number of compressed bits (k).

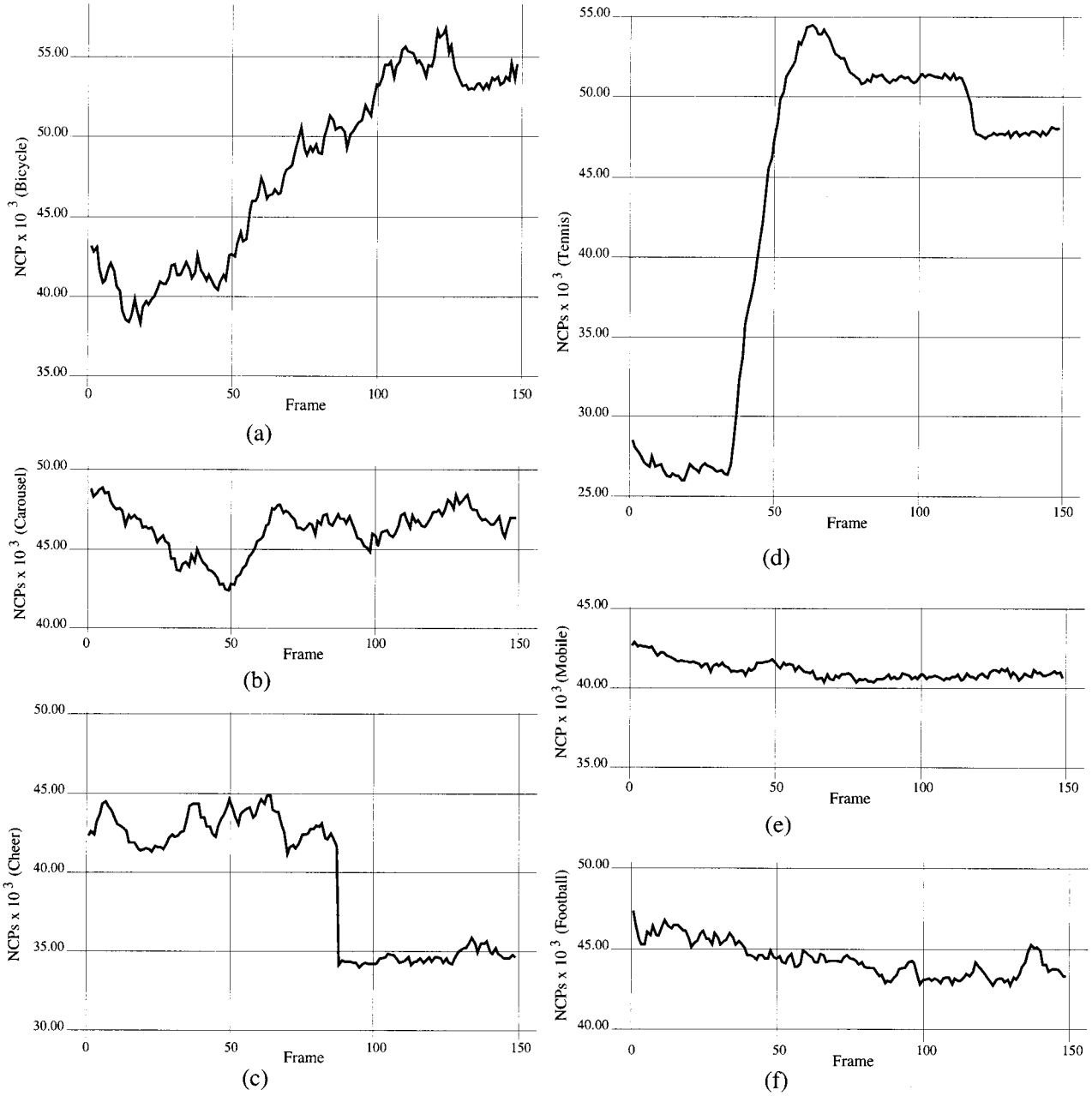

Fig. 8. Variation of the number of compressed pixels per frame for the tested sequences. (a) Bicycle. (b) Carousel. (c) Cheer Leader. (d) Table Tennis. (e) Mobile. (f) Football.

As we see, the most area overhead is introduced by the EBcells. Since an EBcell is attached to every byte of data, the amount of cells in the memory array increases by 1/8. There is an overhead due to decision logic and an extra register (the other one is a conventional memory data register), but it is insignificant in comparison to the EBcells. The flip-flops also do not largely affect the memory size, because only one flip-flop is needed for each 8-bit memory bank. Since the number of banks is small (eight in most DRAMs), we estimate that for the entire DRAM, the extra circuitry imposes around 13% of area overhead. In comparison to conventional memory access, the performance impact of this extra circuitry deals with an extra pipeline stage during which the EB signal is produced. For the current multistage memory instruction execution, this stage can be combined with the address calculation, for example, so as not to incur a time penalty.

III. Evaluation

We tested the scheme in a program which simulated the memory accesses for the full-search block-matching motion estimation in MPEG-2 MP@LL standard (block size: 16×16 pixels, search range: ± 8 pixels). In all the experiments, we ran the program on the first 150 frames of six video sequences: *Bicycle, Carousel, Football, Cheer Leader, Mac_NTSC,* and *Table Tennis* at a frame size of 352×240 pixels and a frame rate of 30 fps.

TABLE I
RESULTS

Video sequence	PSNR(dB)		NCP		
	8-bit	adapt.	maximum	minimum	average
Bicycle	39.822	39.822	56777	35372	47875
Carousel	35.227	35.227	48855	34029	46212
Cheer	51.747	51.747	49876	38340	39456
Football	40.338	40.338	47343	42715	44359
Mac_NTSC	51.388	51.388	42880	40317	41100
T.tennis	37.971	37.971	54461	27088	43723

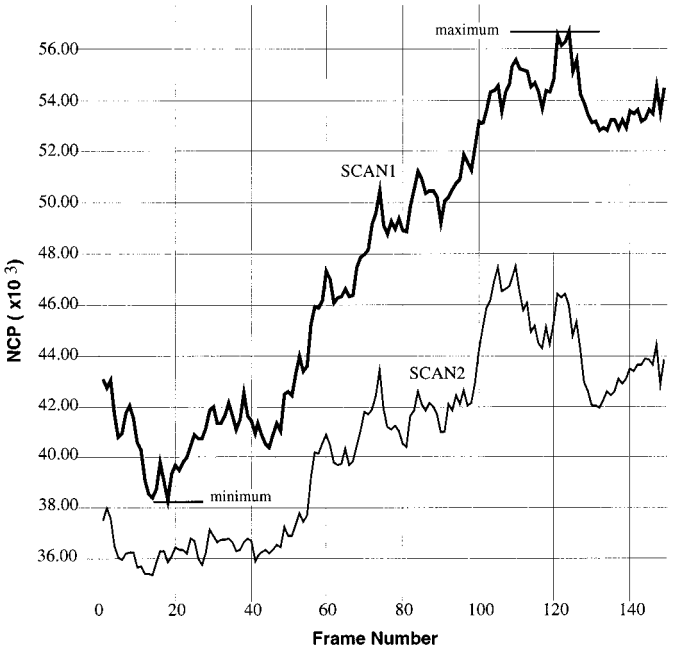

Fig. 9. NCP per frame for two different scan patterns (Bicycle).

We first investigated how various granularities of adaptive bitwidth compression would affect the transition activity of memory bit-lines. For each video sequence, we evaluated the number of signal transitions which occurred in memory bit-lines when the number of compressed bits (k) varied from one (only the first MSB) to eight (all bits). Fig. 7 shows the amount of bit-line transitions observed for the *Bicycle* sequence per frame on write access (read accesses have a similar pattern). Since the results depend on the picture content, we present them for two frames, which demonstrate the most and the least intensive motion in the sequence, respectively. Both figures include the bit-line transitions of the extra EBs. We see that the 4-bit granularity of the compression gives the greatest savings overall. This observation holds for the other sequences as well.

The goal of our second experiment was to evaluate the number of pixels, compressed in the tested sequences for ($k = 4$). Fig. 8 plots the results. As we see, the number of compressed pixels (NCP) varies per frame and per sequence. Sequences and frames with a little picture content variation display a larger NCP. On average, the maximum, minimum, and average number of pixels compressed per frame were 59%, 43%, and 52%, respectively, with the peak of 67% (*Bicycle*).

TABLE II
NCP FOR TESTED SCAN PATTERNS

scan pattern		NCP	
between m/b	within m/b	maximum	average
row-1st	row-1st	56777	47875
row-1st	col-1st	47623	40658
row-1st	zig-zag	56963	48049
col-1st	row-1st	47810	40847
col-1st	col-1st	51243	43514
col-1st	zig-zag	54415	45987
zig-zag	row-1st	48727	41672
zig-zag	col-1st	55563	47093
zig-zag	zig-zag	56294	47581

TABLE III
NCP FOR THE OPTIMAL SCAN

Video	maximum	minimum	average
Bicycle	56963	38963	48049
Carousel	48969	42486	46322
Cheer	45051	34178	39621
Football	47522	42885	44534
Mac_NTSC	42951	40401	41178
T.tennis	54590	26097	43833

To estimate the quality of the results, we compared the peak signal-to-noise ratio (PSNR) computed for the conventional (8-bit access) frame memory and the proposed (adaptive) scheme. Table I lists the results in terms of PSNR, minimum, maximum, and average number of pixels compressed per frame. Because our method efficiently decompresses the pixel representation on the processor side, it maintains the same PSNR as for the conventional (8-bit access) frame memory architecture, even though more than half of the pixels are compressed.

Furthermore, we experimentally found that the number of compressed pixels (NCP) depends on the scan order within the macroblock, as well as between the macroblocks. Fig. 9 depicts variation in the results for two different scan patterns observed for the *Bicycle* sequence. As one can see, changing the scan pattern may affect the NCP as much as 10 000per frame. By trying different scan patterns (see Table II), such as row-first raster scan (*row—1st*), column-first raster scan (*column—1st*), and zig-zag scan, each of which was applied within macroblocks, as well as between the macroblocks, we obtained the best results with the row-first raster scan between the macroblocks and the zig-zag scan within the macroblock (the third row in Table II). Table III outlines the minimum, maximum and average values derived for this (optimal) scan.

Finally, we estimated the energy savings of the proposed solution, assuming that one bit-line activation takes 3 nJ [5]. In comparison to this number, the energy overhead of the 4-bit decision logic (Fig. 5) was ignored, because it was only 46 pJ, according to HSPICE simulation.

Since the proposed scheme disables 4 out of 9 bits, it consumes 5×3 nJ per each compressed access and 9×3 nJ per each full (9-bit) size access. Having the number of pixels compressed per frame (N_c), and the total number of pixels per frame

TABLE IV
ESTIMATED ENERGY DISSIPATION PER FRAME AND SAVINGS RATIO

Video	Energy(min) (mJ)	saving (%)	Energy(av.) (mJ)	saving (%)
Bicycle	1.5933	21	1.7043	16
Carousel	1.6933	16	1.7251	15
Cheer	1.7403	14	1.8055	11
Football	1.7107	16	1.7465	14
Mac_NTSC	1.7433	14	1.7868	12
T.tennis	1.6258	20	1.7549	13
Average		16.8		13.5

(N_{total}), we estimate the energy dissipation per frame of the proposed memory architecture as follows:

$$E_{\text{new}} = [N_c \times 5 + (N_{\text{total}} - N_c) \times 9] \times 3(\text{nJ}). \quad (2)$$

The existing frame memory architectures always use 8-bit wide accesses and, therefore, consume the following energy per frame:

$$E_{\text{ex}} = 8 \times N_{\text{total}} \times 3(\text{nJ}). \quad (3)$$

Thus, the energy savings due to our scheme can be expressed as

$$\text{savings} = \frac{E_{\text{ex}} - E_{\text{new}}}{E_{\text{ex}}} \times 100. \quad (4)$$

Table IV shows the results in terms of minimum and average energy dissipated in our memory architecture per 352×240 pixels frame and the energy saving ratio obtained for the sequences based on (4), $N_{\text{total}} = 84\,480$, and N_c, taken from Table III.

As we see, the proposed approach can save up to 21% of energy for some frames, while on average, it achieves 16% energy reduction.

IV. CONCLUSIONS

A new scheme has been presented for reducing energy consumption of frame memories. In particular, we targeted the reduction of the number of transitions occurring on the bit lines. The scheme eliminates unnecessary signal transitions which take place in the most significant bits of pixel due to sign extension by adjusting dynamically the bit width to the pixel variation. Simulations of the proposed technique using a variety of video signals have shown that our approach can save as much as 21% of energy consumed by the frame memory cell array in comparison to the full resolution memory access, without affecting the picture quality.

The proposed memory energy reduction scheme is strongly application oriented. In fact, it is based on sequential feature of the input video stream which may not be a case for a general-purpose systems, where the memory accesses are of a substantially different nature. On the other hand, it has proven to be particularly suitable to special-purpose multimedia systems, where the memory reading and writing is executed in a sequence or to specific memory structures, such as FIFO, with the fixed accessing order.

ACKNOWLEDGMENT

The author thanks K. Nakasima and M. Fukagawa for their help with the simulation.

REFERENCES

[1] T. Matsumura, H. Segawa, S. Kumaki, Y. Matsuura, A. Hanami, K. Ishihara, S. Nakagawa, T. Kasezawa, Y. Ajioka, A. Maeda, M. Yoshimoto, and T. Sumi, "A chip set for programmable real-time MPEG2 MP@ML video encoder," *IEICE Trans. Electron.*, vol. E81-C, no. 5, pp. 680–694, May 1998.

[2] "MPEG2 encoding LSI's about to change audiovisual equipment for household use" (in Japanese), *Nikkei Electron.*, vol. 3–9, no. 711, pp. 133–1551, Mar. 1998.

[3] B. S. Amrutur and M. Horowitz, "Techniques to reduce power in fast wide memories," in *Proc. Int. Symp. Low Power Electronics*, 1994, pp. 92–93.

[4] B. Gordon, E. Tsern, and T. Meng, "Design of low-power video decompression chip set for portable applications," *J. VLSI Signal Process. Syst.*, vol. 13, pp. 125–142, 1996.

[5] K. Itoh, "Low power memory design," in *Low Power Design Methodologies*, J. Rabaey and M. Pedram, Eds. Norwell, MA: Kluwer, 1996, pp. 201–251.

[6] T. Sugibayashi *et al.*, "A 30 ns 256 Mb DRAM with multi-devided array structure," in *Proc. ISSCC*, 1993, pp. 50–51.

[7] R. Fromm, S. Perissakis, N. Cardwell, and D. Patterson *et al.*, "The energy efficiency of IRAM architectures," in *Proc. 24th Annu. Int. Symp. Computer Architecture*, 1997, pp. 327–337.

[8] T. Watanabe, R. Fujita, and K. Yanagisawa, "Low-power and high-speed advantages of DRAM-logic integration for multimedia systems," *IEICE Trans. Electron.*, vol. E80-C, no. 12, pp. 1523–1531, 1997.

[9] F. Catthoor, "Power efficient data storage and transfer methodologies: Current solutions and remaining problems," in *Proc. IEEE Int. Workshop VLSI*, 1999, pp. 29–34.

[10] Z.-L. He, K.-K. Chan, C.-Y. Tsui, and M. L. Liou, "Low-power motion estimation design using adaptive pixel truncation," in *Proc. Int. Symp. on Low Power Electronics and Design*, Monterey, CA, Aug. 1998, pp. 167–172.

[11] V. G. Moshnyaga, "An MAB truncation scheme for low-power video processors," in *Proc. IEEE Int. Symp. Circuits and Systems*, vol. 4, 1999, pp. 291–294.

[12] J. Y. Tong, D. Nagle, and R. Rutenbar, "Reducing powery optimizing the necessary precision range of floating point arithmetic," *IEEE Trans. VLSI Syst.*, vol. 8, pp. 273–286, June 2000.

[13] M. R. Stan and W. P. Burleson, "Bus-invert coding for low-power I/O," *IEEE Trans. VLSI Syst.*, vol. 3, pp. 49–58, Mar. 1995.

[14] ——, "Low-power encodings for global communication in CMOS VLSI," *IEEE Trans. VLSI Syst.*, vol. 5, pp. 444–455, Dec. 1997.

[15] E. Musoll, T. Lang, and J. Cortadella, "Working zone encoding for reducing the energy in microprocessor address busses," *IEEE Trans. VLSI Syst.*, vol. 6, pp. 568–572, Dec. 1998.

[16] L. Benini, G. D. Micheli, E. Macii, M. Poncino, and S. Quer, "Reducing power consumption of core based systems by address bus encoding," *IEEE Trans. VLSI Syst.*, vol. 6, pp. 554–562, Dec. 1998.

[17] B.-I. Park, Y.-S. Chang, and C.-M. Kyung, "Conforming in-verted data store for low power memory," in *Proc. Int. Symp. Low Power Electronics and Design*, 1999, pp. 91–93.

[18] L. Villa, M. Zhang, and K. Asanovic, "Dynamic zero compression for cache energy reduction," in *Proc. 33rd Int. Symp. Microarchitecture*, Monterey, CA, Dec. 2000.