Reducing Energy Consumption of Video Memory by Bit-Width Compression

Vasily G. Moshnyaga, Koji Inoue, and Mizuka Fukagawa
Department of Electronics Engineering and Computer Science, Fukuoka University
8-19-1 Nanakuma, Jonan-ku, Fukuoka 814-0180, Japan
{vasily, inoue, fukagawa}@v.tl.fukuoka-u.ac.jp

ABSTRACT

A new architectural technique to reduce energy dissipation of video memory is proposed. Unlike existing approaches, the technique exploits the pixel correlation in video sequences, dynamically adjusting the memory bit-width to the number of bits changed per pixel. Instead of treating the data bits independently, we group the most significant bits together, activating the corresponding group of bit-lines adaptively to data variation. The method is not restricted to the specific bit-patterns nor depends on the storage phase. It works equally well on read and write accesses, as well as during precharging. Simulation results show that using this method we can reduce the total energy consumption of video memory by 20% without affecting the picture quality.

Categories and Subject Descriptors

B.3 [**Hardware**]: Memory Structures; B.5.1 [**Hardware**]: Design—*memory design*

General Terms

Design

Keywords

bitwidth-compression, frame memory, low-power design

1. INTRODUCTION

1.1 Motivation

Emerging video application, such as portable video telephone, has elevated needs for low-energy video encoding hardware. In a modern video encoding system such as MPEG, video (or *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 video memory has the highest capacitance in the system, dominating its energy dissipation.

ISLPED'02, August 12-14, 2002, Monterrey, California, USA.

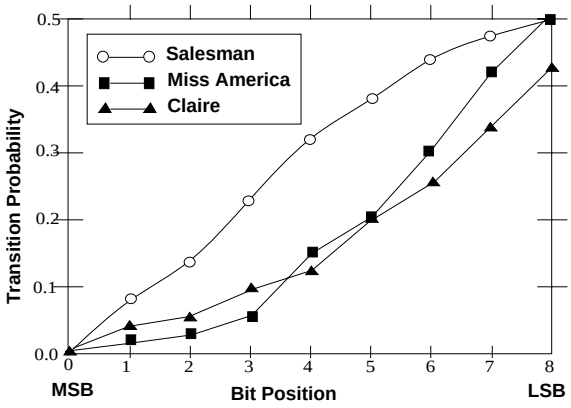

Figure 1: Activity profile for memory input

Although there has been a significant progress in low-power memory design, the frame memory alone still accounts over 40% of the total energy[1]. Therefore it must be optimized for low energy consumption as much as possible.

In MPEG2 encoders, frame memory typically consists of two or four synchronous 16Mbit DRAM units[2]. Bit-lines of these DRAMs are heavily loaded with multiple storage cells[3], and so require a large energy for charging. Since one transition in a single bit-line of the 16Mbit DRAM takes as much as three orders of magnitude more power then the 1-bit adder[4], design solutions capable of minimizing the amount of signal transitions in the frame memory bit-lines are important.

In this paper we focus on reducing the transition activity of the frame memory bit-lines and present a novel architectural technique which exploits the data correlation in image sequences.

1.2 Related Research

There have been reported a number of approaches to low-energy memory design. Many of them have concentrated on technology and circuit optimization which can reduce bit-line energy (e.g. cell structure modification, charge recycling, limited bit-line swing, data retention circuitry, etc.[5]). At the architectural level, research has focused on transition activity reduction through multiple memory splitting into sub-banks, bit-line segmentation, selective line activation, DRAM/logic integration [6], [5],[7].

Despite differences, most of these methods do not consider data to be stored, applying the same control/addressing pat-

tern regardless of data variation on the memory input. In video applications, however, data are highly correlated and distributed unevenly. Figure 1 shows the transition activity profile on the frame memory input for three standard video sequences. Due to high image correlation, the Most Significant Bits (MSB) clearly have much lower switching activity than the Least Significant Bits (LSB). 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[8], [9], data and address busses [10],[11], Up to our knowledge there have been only a few attempts to exploit the input statistics for memory units. Park et al.[13] presented 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.[14] 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 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 limits its application to cache memories of general-purpose processors.

1.3 Contribution

This paper presents a new approach for reducing the transition activity of the frame memory bit-lines. In contrast to previous techniques, we exploit the pixel correlation in video sequences dynamically adjusting the memory bit-width to the number of bits changed per pixel. The proposed method neither depends on the specific bit-patterns (e.g. pre-charging value or zero-bytes) nor to the operation mode. It works well on read, write and precharging accesses, saving up to 20% of the total memory energy without affecting the picture quality.

The paper is organized as follows. The next section describes the approach and outlines its implementation. Section 3 analyzes the performance. Conclusions are drawn in Section 4.

2. ADAPTIVE BITWIDTH COMPRESSION

2.1 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 macroblocks. 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 done in the same order. Due to the macroblock based feature of standard MPEG encoding, the order of reading the picture blocks from the memory equals the order of the block writing. Further, a macroblock consists of $N \times N$ picture elements or pixels. Because of large pixel correlation inside the macroblock, 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.

	Raw data	#Bits		Stored data	#Bits
1	**01101010**	8	**1**	**01101010**	9
2	**01101110**	8	**0**	**1110**	5
3	**01101010**	8	**0**	**1010**	5
4	**11001110**	8	**1**	**11001110**	9
5	**11000011**	8	**0**	**0011**	5
	Total:	**40**	EB	Total:	**33**

Figure 2: An illustration of data compression

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 Most-Significant Bits are equal in pairs, we omit writing the MSBs, storing into the memory only the $n-k$ Least Significant Bits of X and the equality bit (EB), that indicates whether the pixel bit-width is compressed. Formally, the EB is defined as:

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

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 Most-Significant bit-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 either $n/2+1$ bits or $n+1$ bits, adaptively adjusting the bit-width to input data variation.

Figure 2 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 (1st and 4th) of the stream requires full 9-bit representation, while all the others are stored in the compressed (5-bit) form. Numbers at the bottom indicate the total number of bit-line activations necessary to store the data stream in the memory.

2.2 Implementation scheme

2.2.1 Overview

Figure 3 outlines the implementation scheme on example of the 8-bit wide memory (k=4). (Note that multi-byte 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

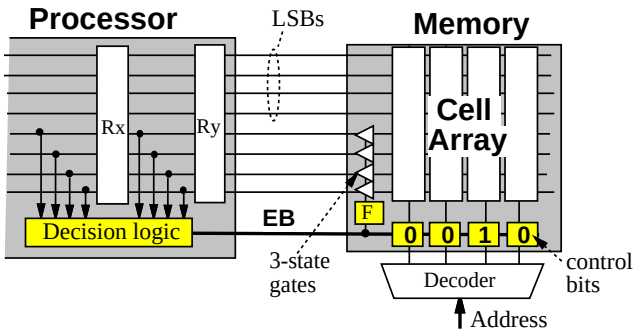

Figure 3: An illustration of the proposed approach

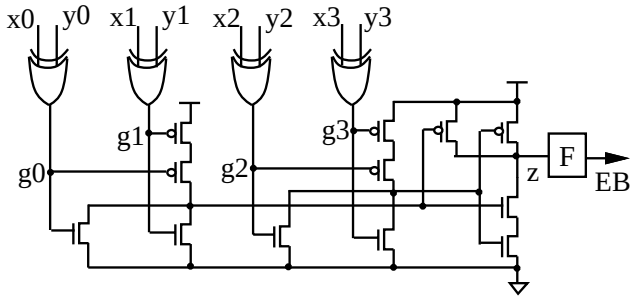

Figure 4: The decision logic circuitry

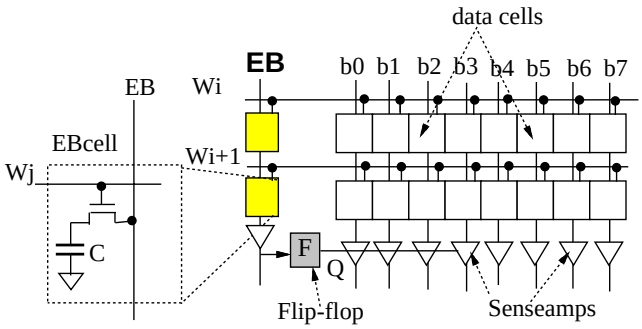

Figure 5: Modified memory byte structure

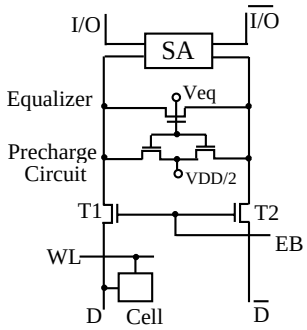

Figure 6: Modified column signal path circuitry

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 as 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.

2.2.2 Decision Logic

Figure 4 depicts the internal circuitry of the Decision Logic. In this figure,$x0-x3$, $y0-y3$ are the MSB bits of the incoming and outcoming data of register Rx. If x_i and y_i are equal, then g_i becomes low. Note that the output signal (z) will be set to zero if and only if all the nodes $g_i, i=0,..,3$ are zeros; that is all four bits of X and Y are same. The F in the figure denotes the flip-flop.

As figure 4 shows, the implementation is regular and simple. The delay introduced by the decision logic is quite small. According to HSPICE simulations (NTT NEL library, 0.5μm CMOS, $V_{dd}=3.3V$), it is less than a 4-input AND gate delay.

2.2.3 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 Figure 5.

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 disabling both the sense-amplifiers and the precharge 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 precharge 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. This selective EB control can be implemented by adding transistors (T1,T2) to the DRAM column signal path circuitry[15], as shown in Figure 6. The sense amplifier (SA), the equalizer and the precharge circuit are activated only when EB is set to 1.

As we see, the most area overhead is introduced by the EBcells. Since a EBcell is attached to every byte of data, the amount of cells in the memory array increases by 1/8. The flip-flops and extra transistors do not largerly affect the memory size, because one flip-flop and 8 extra transistors only are needed for each 8-bit memory bank. Since the number of banks is small (8 in most of DRAMs), we estimate that for the entire DRAM, the extra circuitry imposes around 13% of area overhead. There is an overhead on the processor side 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 memory overhead.

In comparison to conventional memory access, the performance impact of extra circuitry deals with one pipeline stage during which the EB signal is produced. For the current multi-stage memory instruction execution, this stage can be combined with the address calculation, for example, so as not to incur time penalty.

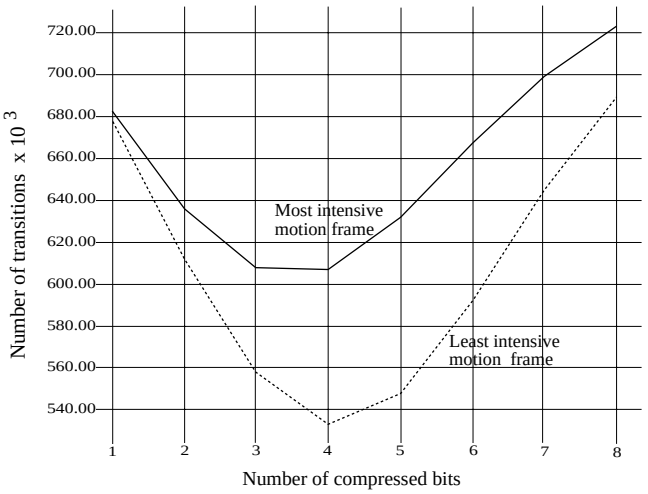

Figure 7: Bit-line transitions per frame against the number of compressed bits (k)

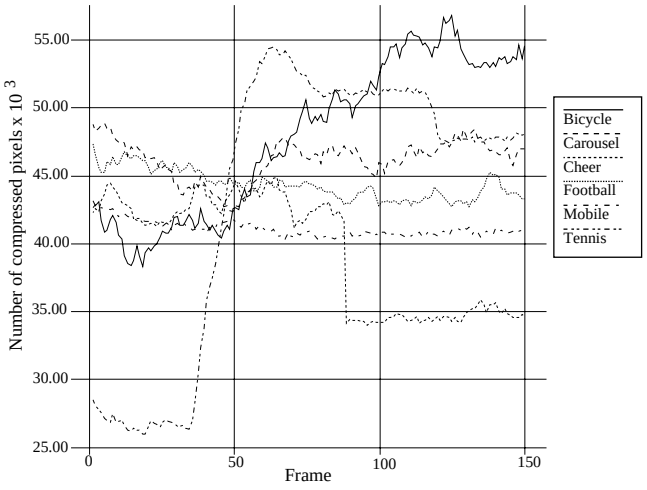

Figure 8: Variation of the number of compressed pixels per frame

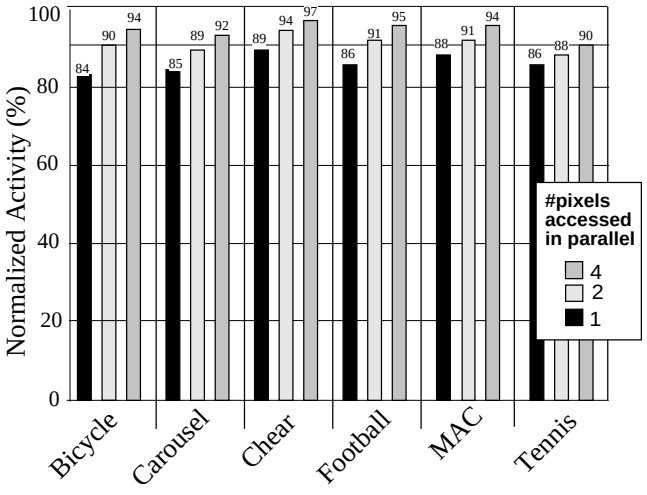

Figure 9: Normalized bit-line activity vs. the number of pixels accessed in parallel

3. 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 (macroblock 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, Table Tennis*; frame size: 352×240 pixels; frame rate: 30 f/s.

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 1 (only the first MSB) to 8 (all bits). Figure 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 bitline 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, assuming that the memory bit-width is limited to 8 (one pixel in a time). Figure 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*).

Figure 9 outlines the average number of bit-line activations when the number of pixels, which are accessed in parallel, increases. (We assumed the four MSB bits of each pixel could be compressed). We observe that the numbers increase with the level of parallelism, i.e. the bit-width. The reason can be explained as follows. When p bytes are accessed simultaneously, the decision logic compares in byte i with byte $i+p$, byte $i+1$ with $i+p+1$, etc. With a larger p, the compared bytes have less bits in common, so the compression occurs less frequently.

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 1 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 pixels is compressed.

Furthermore, we found that the number of compressed pixels (NCP) depends on the scan order within the macroblock as well as between the macroblocks. Figure 10 depicts variation in the results for two different scan-patterns observed for the *Bicycle* sequence. As one can see, chang-

Table 1: The 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

Table 2: 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 3: Amount of compressed pixels 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

ing the scan-pattern may affect the NCP as much as ten thousands per frame. By trying different scan-patterns (see Table 2), such as row-first raster scan ($row-1st$), column-first raster scan ($column-1st$), and zig-zag scan, each of which applied as 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 2). Table 3 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 3nJ[5]. In comparison to this number, the energy overhead of the 4-bit decision logic (Fig.5) was ignored, because it was only 46pJ according to HSPICE simulation.

Since the proposed scheme disables 4 bits out of 9, 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), the total number of pixels per frame (N_{total}), we estimate the energy dissipation per frame of the proposed memory architecture as: $E_{new} = [N_c \times 5 + (N_{total} - N_c) \times 9] \times 3(nJ)$.

The existing frame memory architectures always use 8-bit wide accesses and therefore consume the following energy per frame: $E_{ex} = 8 \times N_{total} \times 3(nJ)$.

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

$$Saving = (E_{ex} - E_{new})/E_{ex} \times 100. \quad (1)$$

Table 4 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 Eq.(1), $N_{total} = 84480$, and N_c taken from Table 3.

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

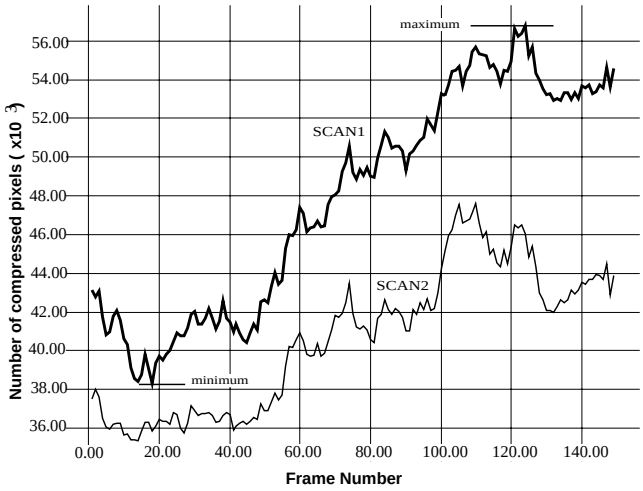

Figure 10: NCP per frame for two different scan patterns (Bicycle)

Table 4: Estimated energy dissipation per frame and saving ratio

Video	Minimum		Average	
	Energy (nJ)	saving (%)	Energy (nJ)	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

4. CONCLUSIONS

A new scheme has been presented for reducing energy consumption of video (or frame) memory. 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 and throughput. Our future work is dedicated to prototype memory chip design.

5. ACKNOWLEDGMENTS

This work was sponsored in part by The Ministry of Education, Culture Sports, Science and Technology of Japan, Grant-In-Aid for Scientific Research No.14580399 (C)(2) and Mitsubishi Electric Corporation, Grant No.T000666TK.

6. REFERENCES

[1] M.Takahashi, T.Nishikawa, M.Hamada, T.Takayanagi, H.Arakida, N. Machida, H. Yamamoto, T. Fujiyoshi, Y. Ohashi, O. Yamagishi, T. Samita, A.Asano, T. Terazawa, K. Ohmori, Y. Watanabe, H.Nakamura, S. Minami, T.Kuroda, and T. Furuyama, "A 60-MHz 240-mW MPEG-4 Videophone LSI with 16-Mb Embedded DRAM", *IEEE J. Solid-State Circuits*, vol.35, no.11, pp.1713-1721, November 2000.

[2] "MPEG2 Encoding LSIs About to Change Audio-visual Equipment for Household Use", *Nikkei Electronics*, vol. 3-9, no. 711, pp. 133–155, March 1998.(in japanese)

[3] B. Amrutur, "Techniques to Reduce Power in Fast Wide Memories", *Proc. 1994 Int. Symp.on Low Power Electronics*, 1994, pp. 92–93.

[4] B. Gordon, E. Tsern, T. Meng, "Design of Low-Power Video Decompression Chip Set for Portable Applications",*J. VLSI Signal Processing Systems*, vol. 13, no. 2-3, pp. 125–142, Aug/Sep. 1996.

[5] K. Itoh, "Low Power Memory Design", in *Low Power Design Methodologies*, J. Rabaey, M. Pedram Eds., pp. 201–251, 1996

[6] R. Fromm, S. Perissakis, N. Cardwell, D. Patterson, et al, "The Energy Efficiency of IRAM Architectures", *Proc. 24-th Int. Symp. on Comp. Architecture*, 1997, pp. 327–337.

[7] T. Watanabe, R. Fujita, K. Yanagisawa, "Low-Power and High-Speed Advantages of DRAM-Logic Integration for Multimedia Systems," *IEICE Trans. Electronics*, vol. E80-C, no.12, pp. 1523-1531, Dec. 1997.

[8] Z-L. He, K-K. Chan, C-Y. Tsui, and M. L. Liou, "Low-Power Motion Estimation Design Using Adaptive Pixel Truncation", *Proc. Int. Symp. on Low Power Electronics and Design*, pp. 171–174, 1997.

[9] V. G. Moshnyaga, "An MSB Truncation Scheme for Low-Power Video Processors", *Proc. IEEE Int. Symp. Circuits and Systems*, vol.4, 1999, pp.291–294.

[10] M. R. Stan and W. P. Burleson, "Bus-Invert Coding for Low-Power I/O", *IEEE Trans. VLSI Systems*, vol. 3, no.1, pp. 49–58, Mar. 1995.

[11] E. Musoll, T. Lang, and J. Cortadella, "Working Zone Encoding for Reducing the Energy in Microprocessor Address Busses", *IEEE Trans. VLSI Systems*, vol.6, no.4, pp. 568–572, Dec. 1998.

[12] L. Benini, G. D. Micheli, E. Macii, M. Poncino, S. Quer, "Reducing Power Consumption of Core Based Systems by Address Bus Encoding", *IEEE Trans. VLSI Systems*, vol. 6, no. 4, pp. 554–562, Dec. 1998.

[13] B.-I. Park, Y.-S. Chang and C.-M. Kyung. "Conforming Inverted Data Store for Low Power Memory",*Proc. Int. Symp. on Low Power Electronics and Design*, 1999, pp. 91–93.

[14] L. Villa, M. Zhang and K. Asanovic "Dynamic Zero Compression for Cache Energy Reduction",*Proc. 33rd Int. Symp. on Microarchitecture*, 2000.

[15] K.Satoh, et al., " A 20ns static column 1 Mb DRAM in CMOS Technology", *ISSCC Dig. Tech. Papers*, Feb. 1985, pp.254-255.