2008 International Conference on Multimedia and Ubiquitous Engineering

Reconfigurable Depth Buffer Compression Design for 3D Graphics System

Tzung-Rung Jung, Lan-Da Van, Wai-Chi Fang*, Teng-Yao Sheu
Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan
* TSMC Distinguished Chair Professor, Dept. of Electronics Eng., National Chiao Tung University, Hsinchu, Taiwan
E-mail: ldvan@cs.nctu.edu.tw , wfang@mail.nctu.edu.tw

Abstract—**Depth buffer bandwidth reduction is one of the most important issues in bandwidth-limited 3D computer graphics system. In this paper, we propose a reconfigurable algorithm for depth buffer compression. For 8x8 tile size and 16-bit depth values, the proposed algorithm can achieve 1.91:1 compression ratio on average and improve 76.9% and 39.4% compared with HA [12] and DDPCM [7], respectively. Furthermore, the devised algorithm supports one-plane and two-plane compression modes and manipulates break points more efficiently.**

I. Introduction

The increasing demand of the depth buffer comparison has led to widespread interests in 3D graphics system. Nowadays, the 3D computer graphics are widely used in many applications such as mobile phones [1], GPS (Global Positioning System), digital TV [2-4], games, biomedical applications [5], where these applications usually use complex GUI (graphical user interface) for generating better 3D images. It is known that 3D computer graphics system requires extreme high memory bandwidth. On the other hand, with the growth of complexity of 3D scenes, the amount of data computation increases substantially. Therefore, in the bandwidth-limited system, how to efficiently compress the depth buffer data to save bandwidth becomes a significant research issue. Fast z-clears compression algorithm [6] uses a dedicated flag to indicate whether a tile is cleared. DDPCM [7], anchor encoding [8], efficient depth buffer compression [12] (i.e., HA method) are effective compression algorithms exploiting the coherence of interpolated depth values. The plane encoding scheme presented in [10] applies the concept of indexing to compress tiles. Depth offset compression [11] saves the differentials based on the z-max value (maximum depth value) and z-min value (minimum depth value) in a tile. Other compression methods are released in [13-15]. Since the compression performance of the above state-of-the-art algorithms are limited and cannot be adaptively compressed according to different scenes, we are motivated to proposed adaptive depth buffer compression algorithm for 3D graphics systems.

This paper is organized as follows: an overview of depth buffer compression algorithms is described in Section II. In Section III, the proposed reconfigurable algorithm has been presented. The simulation and comparison results are presented in Section IV. At last, the brief statements conclude the presentation.

This work was supported in part by the National Science Council (NSC) Grant NSC-96-2220-E-009-038, NSC-96-2221-E-009-220, and MOEA-96-EC-17-A-01-S1-048.

II. An Overview of Depth Buffer Compression Algorithms

In this section, we give an overview of state-of-the-art and existing compression algorithms [6-15].

A. *Fast z-clears*

Fast z-clears [6] is a simple compression algorithm and easy to be implemented. A dedicated bit used to indicate whether the tile is cleared. If the tile is cleared, we can only write back the latest depth values to depth buffer without reading the depth buffer before updating depth values.

B. *DDPCM*

DDPCM (Differential Differential Pulse Code Modulation) [7] is an off-the-shelf data compression algorithm. Since the depth values are linearly interpolated in screen space, DDPCM algorithm is very suitable for this kind of condition. DeRoo *et al.* [7] proposed a depth buffer compression algorithm as illustrated in Fig. 1. DDPCM can achieve 8:1 compression ratio on 8x8 tile size, using a 32-bit depth values and reading 256 bits from memory. DeRoo *et al.* also proposed an extended depth buffer compression algorithm, called two-plane mode, in order to handle specific cases that tiles can be separated into two planes. In 24-bit depth value case and 8x8 tile size, we have to save two 24-bit reference points, the upper left and lower right pixels in the tile, two 24-bit x differentials, two 24-bit y differentials, 57 2-bit predicted terms and four 8-bit break points used to combine two planes based on different reference points. One disadvantage of the DDPCM is that two-plane mode can be only supported when the reference points are at upper-left and lower-right position.

C. *Anchor encoding*

Van Dyke and Margeson [8] proposed a compression scheme similar to the DDPCM algorithm. Instead of setting upper left pixel as a reference point, this compression algorithm select a fixed anchor point, z, from other positions in a tile as shown in the Fig. 2 All we have to save are 16-bit anchor point, 7-bit x differential, 7-bit y differential and 5-bit predicted terms.

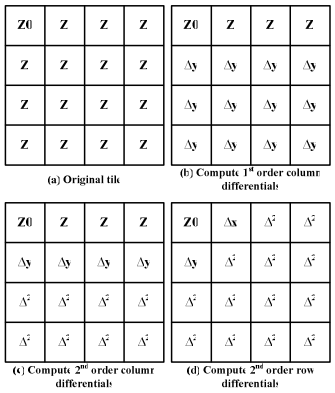
(a) Original tile
(b) Compute 1st order column differentials
(c) Compute 2nd order column differentials
(d) Compute 2nd order row differentials

Figure 1. Illustration of DDPCM [7, 12].

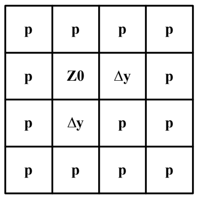

Figure 2. Illustration of anchor encoding [8, 12].

D. *Efficient depth buffer compression (Referred to as HA Method)*

Hasselgren and Akenine-Möller [12] proposed a new depth buffer compression algorithm, which can achieve high compression ratio by exploiting the coherence of interpolated depth values in screen space [12]. The operations of the one-plane mode and two-plane mode are illustrated in Fig. 3 and Fig. 4, respectively. For an example of the one-plane mode, the predicted terms are saved in only 1 bit such that this algorithm can achieve better compression ratio than other compression algorithms. Additionally, this algorithm provides the flexibility to handle two-plane mode cases rather a fixed-position-reference-point of the extended DDPCM algorithm. This algorithm works well, when depth values are interpolated in higher resolution than used for storage [12].

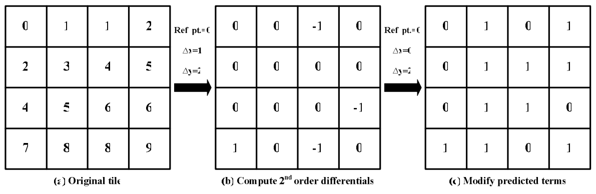
Ref pt.=0
Δx=1
Δy=2
Ref pt.=0
Δx=0
Δy=2
(a) Original tile
(b) Compute 2nd order differentials
(c) Modify predicted terms

Figure 3. Illustration of one-plane mode of HA compression [12].

E. *Plane encoding*

Different from the compression algorithms exploiting the coherence of interpolated depth values in screen space, plane encoding labels triangles in a range of tiles and saves these index numbers eventually. When a pixel is rendered, the depth value corresponding to the coordinate has to be computed as soon as possible for depth-buffer-based rendering algorithms. Van Hook [9] and Liang *et al.* [10] both presented compression algorithms similar to plane encoding. Fig. 5 shows the abstract concept of plane encoding.

The plane encoding can handle several overlapping triangles in a single tile, which is suitable for large tile size. The drawback is that it must store indices and the corresponding counter value in depth tile cache [12].

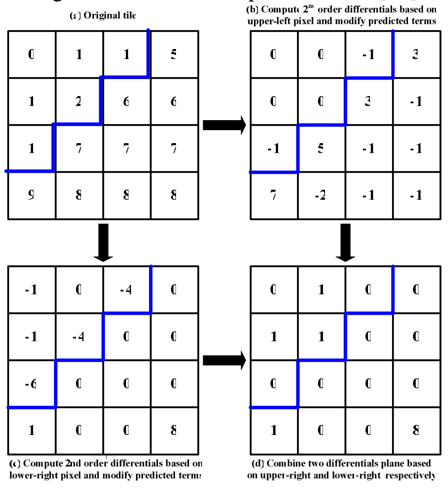
(a) Original tile
(b) Compute 2nd order differentials based on upper-left pixel and modify predicted terms
(c) Compute 2nd order differentials based on lower-right pixel and modify predicted terms
(d) Combine two differentials plane based on upper-right and lower-right respectively

Figure 4. Illustration of two-plane mode of HA compression [12].

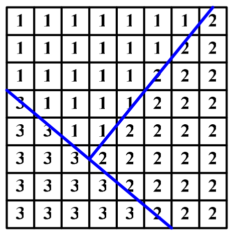

Figure 5. Illustration of plane encoding[9, 12].

F. *Depth offset compression*

Morein and Natale [11] presented depth offset compression illustrated in Fig. 6. For tile-based rendering, assume that we save the z-max (maximum depth value) value and z-min value (minimum depth value) of a tile. The depth values of a tile will be categorized into to representable ranges or unrepresentable range. Representable ranges consist of two regions based on z-max value and z-min value.

If we have stored the z-max and z-min values of the compressed tile, this algorithm can be applied without extra cost. It cannot work well at high compression ratio, but obtains excellent compression probabilities at low compression ratio.

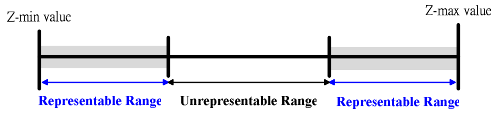

Figure 6. Illustration of depth offset compression [11, 12].

III. PROPOSED RECONFIGURABLE ALGORITHM

In this section, we propose a reconfigurable algorithm for depth buffer compression. According to different scene changes, the proposed algorithm is capable of adaptively employing three compression algorithms including modified DDPCM, modified HA, and general DDPCM, where these compression algorithms exploit the coherence of interpolated depth values in screen space. General DDPCM makes use of 7 bits to save the predicted terms. Our proposed algorithm as shown in Fig. 7 also supports one-plane mode and two-plane mode compression scheme. Using the op-code and break-points, the proposed reconfigurable data flow expressed in Fig. 7 is demonstrated as follows.

In the first step, we compute the 2nd order differentials. In the second step, we calculate and determine the range of these differentials. If any differential is larger than the maximum number or less than minimum number that general DDPCM can serve, we will label this tile as an uncompressed tile. If the tile passes the check point, we will determine whether the tile is one-plane or two-plane. In case the tile belongs to the two-plane mode, we have to compute another set of differentials according to another reference point. After computing another set of differentials, we combine these two sets of differentials separated by break points. Besides, if the tile is in a falling case as shown in Fig. 8(b), we have to compute another two sets of differentials based on lower-left pixel and upper-right pixel, denoted as R, as shown in Fig. 8. We will check the two-plane-mode tile if it is a rising, falling, vertical, horizontal case by a specific combinational circuit without computing sets of differentials according to the four corner pixels. In the third step, we will select which compression algorithm can handle these combined differentials by a simple combinational circuit. At the last step, we pack the compressed data with the op-code together and transfer them to memory bus.

A one-plane mode, for example as shown in Fig. 9, expresses how to compute these differentials. Four cases as shown in Fig. 8 including rising case, falling case, vertical case, and horizontal are handled by the proposed algorithm in two-plane mode. Fig. 10 shows a two-plane mode example and discusses how to compute two sets of differentials according to different reference points and how to combine the two planes. Notice that for hardware-oriented design, we do not completely follow the steps presented by HA [12].

The format of the op-code and break points are depicted in Fig. 11(a) and (b), respectively. The first bit of the op-code represents whether the tile is compressed. The second bit of the op-code indicates whether the tile is one-plane mode or two-plane mode. The third and the fourth bits represent what kind of compression algorithm is applied in horizontal direction. The fifth and the sixth bits represent what kind of compression algorithms is applied in vertical direction. The horizontal direction means the positions, Z2, Z3, Z5, Z6, Z7, Z9, Z10, Z11, Z13, Z14, and Z15, in Fig. 10. The vertical direction means the positions, Z8 and Z12, in Fig. 10. In terms of break point, the first and the second bits indicate what kind of combination modes, such as rising case, is applied to the break points. The third, the fourth and the fifth bits mean the minimum row number of the break points. The sixth, the seventh and the eighth bits mean the column number of the break point with the minimum row number. Notice that the break points will be saved only when the tile is in two-plane mode. Additionally, if the tile is uncompressed, only the first bit of the op-code will be packed with the tile.

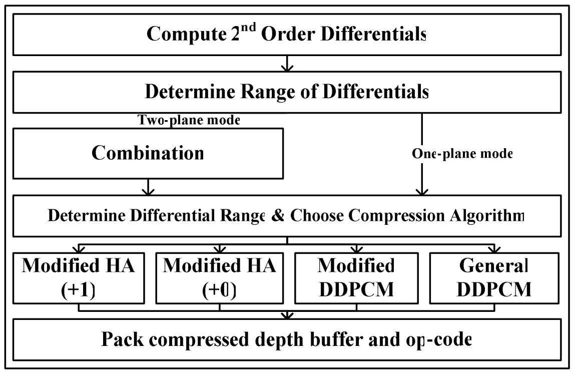
Compute 2nd Order Differentials
Determine Range of Differentials
Two-plane mode
Combination
One-plane mode
Determine Differential Range & Choose Compression Algorithm
Modified HA (+1)
Modified HA (+0)
Modified DDPCM
General DDPCM
Pack compressed depth buffer and op-code

Figure 7. Data flow of the proposed reconfigurable depth buffer compression.

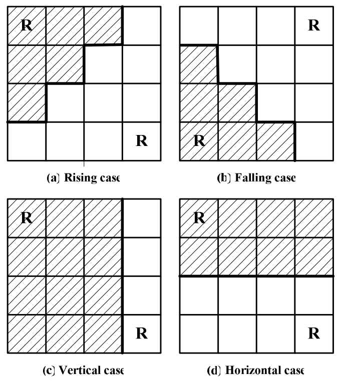

Figure 8. Four cases supported by the proposed algorithm

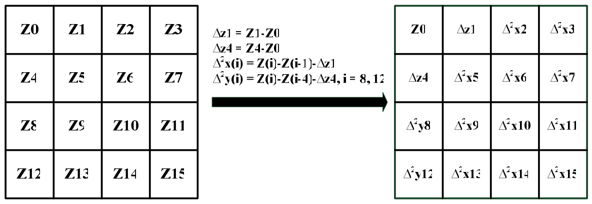

Figure 9. Example of one-plane mode.

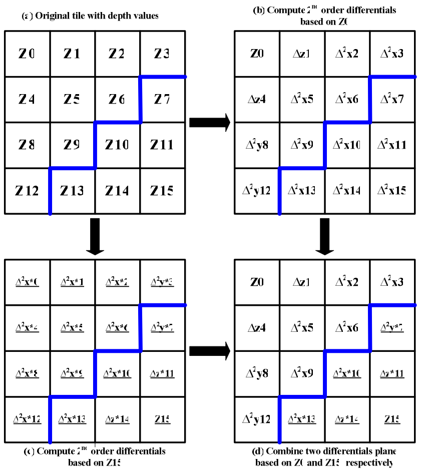
(a) Original tile with depth values
(b) Compute 2nd order differentials based on Z0
(c) Compute 2nd order differentials based on Z15
(d) Combine two differentials plane based on Z0 and Z15 respectively

Figure 10. Two-plane mode of the proposed reconfigurable algorihtm.

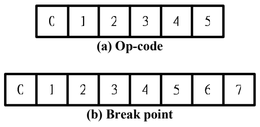

Figure 11. Op-code and break point.

IV. Comparision and Simulation Results

In this section, the comprehensive comparisons as listed in Table 1, 2, and 3 with HA, DDPCM, and anchor encoding in terms of average compression ratio, compression ratio in one-plane mode and two-plane mode, and number of bits of the proposed algorithm, respectively. Except our proposed algorithm, we assume that every compression algorithm requires 1-bit op-code for indicating whether the tile is compressed. The teapot benchmark is used as a reference simulation as shown in Fig. 12. The average compression ratio as listed in Table 1 shows that the proposed scheme outperforms others by 39.4% and 76.9% compared with DDPCM and HA methods. In this simulation, we can find that the average compression ratio of HA method is not better than either DDPCM or our proposed scheme. The reason is that HA compression scheme is suitable for the high precision interpolation. In other words, if the simulation benchmark belongs to the high precision interpolation, it turns out a much higher average compression ratio obtained by HA compression method. Table 2 shows the compression ratio distribution for one-plane mode and two-plane mode. Instead of 26 bits or 32 bits for saving break points in 8x8 tile size applied by efficient depth buffer compression, our proposed scheme only needs 8 bits. Therefore, the compression ratio of the proposed scheme is higher than other existing algorithms. Furthermore, it can be expected that the anchor encoding cannot achieve high compression ratio. Because the anchor encoding use 5 bits to save predicted terms for each compressed tile and does not support two-plane mode. Table 3 shows the total bits when a tile is compressed. Concerning the general DDPCM method, we expect that the size of the compressed tile can be smaller than that of half size of original tile. The distribution of the average compression ratio as shown in Figs. 13 and 14 illustrates the usefulness of our proposed scheme compared with DDCPM and HA compression schemes, respectively. A point in Fig. 13 and Fig. 14 indicates an average compression ratio of five tiles. It is manifest that our proposed scheme can achieve more stable average compression ratio than HA and DDPCM compression methods.

Figure 12. Benchmark scene.

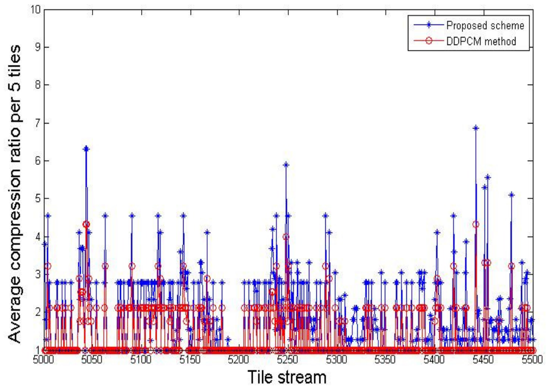

Figure 13. Distribution of the average compression ratio using proposed scheme vs. DDPCM compression method.

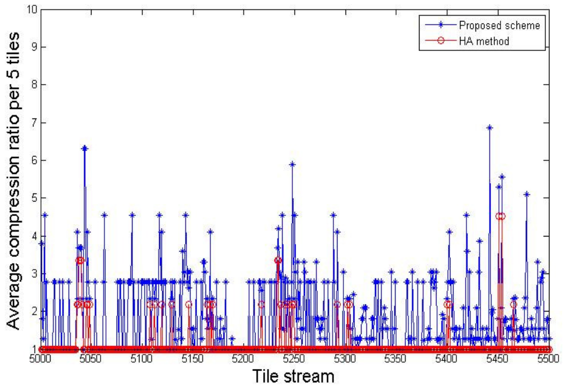

Figure 14. Distribution of the average compression ratio using proposed scheme vs. HA compression method.

Table 1. Average Compression Ratio with 8x8 Tile Size

	Average compression ratio
HA Method [12]	1.08
DDPCM [7]	1.37
Proposed scheme	1.91

TABLE 2. COMPRESION RATIO WITH 8X8 TILE SIZE

	One-plane mode	**Two-plane mode**
Anchor encoding [8]	3.05 : 1	N/A
HA Method [12]	10.89 : 1	6.87 : 1
DDPCM [7]	6.52 : 1	4.85 : 1
Proposed scheme – modified HA	10.56 : 1	7.76 : 1
Proposed scheme – modified DDPCM	6.48 : 1	5.39 : 1
Proposed scheme – general DDPCM	2.21 : 1	2.13 : 1

TABLE 3. SIZE OF COMPRESSED TILE USING PORPOASED SCHEME

Vertical	**Horizontal**	**One-plane mode (bits)**	**Two-plane mode (bits)**
Modified HA	Modified HA	97	132
Modified HA	Modified DDPCM	152	184
Modified HA	General DDPCM	427	444
Modified DDPCM	Modified HA	103	138
Modified DDPCM	Modified DDPCM	158	190
Modified DDPCM	General DDPCM	433	450
General DDPCM	Modified HA	133	168
General DDPCM	Modified DDPCM	188	220
General DDPCM	General DDPCM	463	480

V. COMCLUSION

In this work, an adaptive algorithm for depth buffer compression is presented. This proposed algorithm not only supports modified HA, modified DDPCM as well as general DDPCM algorithms, but also handles one-plane mode and two-plane mode compression. In addition, the algorithm saves the break points more efficiently for 8x8 tile size. From the experimental results, the proposed reconfigurable scheme can provide more stable average compression ratio to achieve the quality guarantee performance.

REFERENCES

[1] Tomas Akenine-Möller and Jacob Ström, “Graphics for the masses: a hardware rasterization architecture for mobile phones,” ACM Transactions on Graphics, vol. 22 , pp.801-808, July 2003.

[2] TS 102 812, “DVB Multimedia Home Platform (MHP) Specification 1.1”, Nov. 2001.

[3] B. Javidi and F. Okano, “Three-Dimensional Television, Video and Display Technology,” Springer-Verloag, 2002.

[4] A. Redert, M. Op de Beeck, C. Fehn,W. IJsselsteijn, M. Pollefeys, Van Gool, E. Ofek, I. Sexton, and P. Surman, “Attest vadvanced three-dimensional television system,” Proc. Of 3DPVT, pp. 313–319, 2002.

[5] T. Heinonen, A. Lahtinen and V. Hakkinen, “Implementation of three-dimensional EEG brain mapping,” Computers and Biomedical Research 32, pp. 123–131, 1999.

[6] S. Morein., “Method and apparatus for efficient clearing of memory,” in US Patent 6,421,764, 2002.

[7] J. DeRoo, S. Morein, B. Favela, M. Wright, “Method and apparatus for compressing parameter values for pixels in a display frame,” in US Patent 6,476,811, 2002.

[8] J. Van Dyke, J. Margeson, “Method and apparatus for managing and accessing depth data in a computer graphics system,” in US Patent 6,961,057, 2005.

[9] T. Van Hook, “Method and apparatus for compression and decompression of Z Data,” in US Patent 6,630,933, 2003.

[10] B.-S. Liang, Y.-C. Lee, W.-C. Yeh, and C.-W. Jen, “Index rendering: hardware-efficient architecture for 3-D graphics in multimedia system,” in *IEEE Transactions on Multimedia*, vol. 4, no. 2, pp. 343-360, June 2002

[11] S. Morein, M. Natale, ”System, method, and apparatus for compression of video data using offset values,” in US Patent 6,762,758, 2004.

[12] J. Hasselgren, T. Akenine-Möller, “Efficient depth buffer compression,” *Graphics Hardware*, pp. 102-110, 2006.

[13] S. Morein, “ATI Radeon HyperZ technology,” *in Hot3D Proc. ACM SIGGRAPH/Eurographics Workshop on Graphics Hardware*, Aug. 2000.

[14] C.-H. Chen and C.-Y. Lee, “Two-level hierarchical Z-buffer with compression technique for 3D graphics hardware,” The Visual Computer, Springer, vol. 19, no. 7-8, pp. 467-479, Dec. 2003.

[15] C.-H. Yu and L.-S. Kim, “A hierarchical depth buffer for minimizing memory bandwidth in 3D rendering engine: depth filter,” ISCAS '03, vol.2, pp.II-724- II-727, May 2003.