A LOW-COST RASTER ENGINE FOR VIDEO GAME, MULTIMEDIA PC AND INTERACTIVE TV*

Chein-Liang Chen, Bor-Sung Liang, and Chein-Wei Jen
Department of Electronics Engineering and Institute of Electronics
National Chiao Tung University
Hsinchu, Taiwan, R.O.C.

ABSTRACT

A low-cost Raster Engine (RE) is designed and implemented to improve the performance of 3D computer graphics and image composition application for video games, multimedia PCs and interactive TVs. Three operation modes: Gouraud and Phong shading algorithms, and image composition are incorporated in this chip. Modified digital differential analyzer (DDA), 2-level pipeline, and constant execution time for calculating $\cos^n \theta$ are proposed as the features of this design. The accelerator is implemented by 0.8um SPDM CMOS VLSI technology and able to release more then 50% CPU loads.

1. INTRODUCTION

The multimedia system plays a more and more important role in nowaday consumer electronics, especially in video games, multimedia PCs, and interactive TVs. It integrates text, graphic, video, image, and audio signals. In order to handle many various signals, the multimedia systems must deal with a huge amount of operations in a short period of time, especially 3D computer graphics in interactive multimedia systems and virtual reality system. To improve the performance of 3D computer graphics, there are many previous works to achieve this goal. However, because the total-hardware-solutions approaches are usually very expensive [1-8], the cost is too high to afford for end-users. Since the rasterization of 3D computer graphics is massive and regular , it is suitable to be accelerated by specific hardware. But the low-cost approaches suffer from the restriction of one specific algorithm only which utilize the dedicated hardware working with a host CPU [9,10]. hence, a low-cost, PC-based graphics subsystem is proposed in this paper, called raster engine (RE) , to offer more operation modes and algorithm which will be suitable for different applications.

The objects in 3D computer graphics are modeled by geometric primitives, like triangles or polygons. After the geometric engine transforms the coordinates of primitives into the device coordinates, each primitive can be decomposed into several scan-lines; a scan-line is made of many pixels. Rendering is the process of creating images from object models, while rasterization is the process of calculating pixel values from geometric primitives.

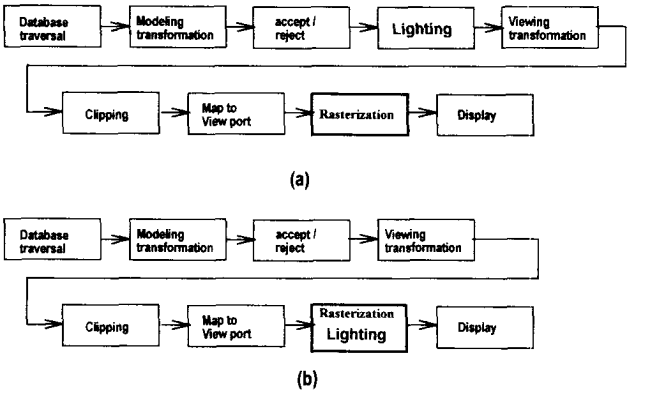


Fig 1. The rendering pipeline

(a) Z-buffered Gouraud shading (b) Z-buffered Phong shading

Gouraud shading and Phong shading algorithms have been developed for rasterization. Fig 1(a) and Fig 1(b) show the graphics rendering pipelines for Gouraud shading and Phong shading[11]. The blocks before rasterization in the pipeline obtain the primitives by travering database, and perform the coordinate transformation, translation, rotation, and clipping, which are usually done by the geometric engine or CPU. The other blocks of the pipeline deal with rasterization and display. The main difference between two algorithms is when to calculate the pixel intensities by the local reflection model. The much-loved reflection model is Phong reflection model [12]:

$$I = I_a \times K_a + [I_i \times K_d (\boldsymbol{L} \cdot \boldsymbol{N}) + I_i \times K_s (\boldsymbol{H} \cdot \boldsymbol{N})^n] \quad (1)$$

Let

$$\boldsymbol{L} \cdot \boldsymbol{N} = \cos \alpha$$
$$\boldsymbol{H} \cdot \boldsymbol{N} = \cos \beta$$

it will change to

$$I = I_a \times K_a + [I_i \times K_d (\cos \alpha) + I_i \times K_s (\cos \beta)^n] \quad (2)$$

*This work was supported by the National Science Council, Taiwan, R.O.C under Grant NSC83-0404-E009-041

Manuscript received June 12, 1995

where

I_a : intensity of the ambient light

I_i : intensity of the light source

L : the unit vector from pixel to the light source

N : the normal vector of the pixel

$H = (L+V)/2$,

where V is the vector from the pixel to the viewer

n : gloss to the model high light

K_a, K_d, K_s : coefficients to model the characteristic of the material

The Phong reflection model imitates the lighting behavior to a degree that produces a first-order approximation to photorealism. We must assume that all light source is point source, all geometry except the surface normal vector is ignored, and ambient is modeled as a constant, because the Phong reflection model is a empirical model . It is simplicity and imparts a acceptable realism sufficient for many applications. Although it carries a recognizable computer signature, it is still a good model for low-cost 3D computer graphics solutions.

In Fig 1(a), the lighting is executed before rasterization, therefore Gouraud shading algorithm applies the local reflection model only at each vertex of the primitives for computing intensities; then, linear interpolation is utilized to produce the intensities in primitives interior on the scan-line basis. However, the scheme can not cope well with highlight. if there is a highlight area inside the primitives, the intensity of pixels will be truncated and replaced by the interpolated value of vertices. So, even Gouraud shading algorithm is very fast and generally used for 3D computer graphics system, that is the main disadvantage.

On the other hand, Phong shading algorithm is known as a normal-vector interpolation algorithm. As shown in Fig 1(b), the lighting is executed with rasterization and it is for every pixel inside a primitive. The local reflection model is employed after normal vector interpolations, therefore the highlight is well solved. But a huge amount of operations are necessary to deal with the complex calculations of local reflection model on each pixel.

Another important function for multimedia system is image or video signal composition. It is a good way to combine multiple video sources into a single image or to produce the antialiasing result. Porter and Duff [13,14] have proposed RGB α model to combine separate elements. We take the operation of f **over** b described in their papers as an example:

f **over** b:

$$rgb_c = rgb_f + (1-\alpha_f)\, rgb_b \qquad (3)$$

$$\alpha_c = \alpha_f + (1-\alpha_f)\, \alpha_b$$

where

rgb : means the R,G,B value

α : means the proportion of the covering area to the whole pixel value

c : means combined value

f : means front-end value

b : means back-end value

Since the **over** operation needs only an interpolative operation, it can be integrated into the RE chip.

In order to dominate RE chip, the host CPU provides necessary parameters. The RE runs in its own sequence to perform Z-buffered Phong shading, Z-buffered Gouraud shading or RGB α composition according to the selected rendering mode, and then stores its result into frame buffer.

2. ALGORITHMS IN RE

2.1 Modified DDA

Digital differential analyzer (DDA) [12] is nowadays in widespread use for calculating the intensities of pixels on scan-lines in Gouraud shading, as illustrated in Fig 2. It finds how many the values increases along the scan-line, and then repeatedly adding the increment.

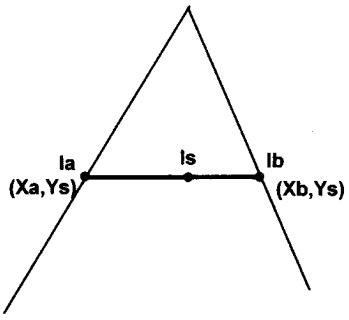


Fig 2. Scheme of DDA

$$\Delta I_s = \frac{\Delta x}{x_b - x_a}(I_b - I_a) \qquad (4)$$

$$I_{s,n} = I_{s,n-1} + \Delta I_s$$

where

I_a, I_b : intensities of pixel (Xa,Ys),(Xb,Ys)

Δx : the steps from the initial pixel (Xa,Ys) along the scan-line

ΔI : the increment intensity

$I_{s,n}$: current intensity

$I_{s,n-1}$: the previous intensity

The simple interpolation equation for composition is described as following:

$$I_s = I_a(1-\alpha) + I_b\alpha \qquad (5)$$

where α is the weighting factor to combine I_a and I_b

In order to integrate composition mode into RE, we have to modify DDA equation to combine interpolation. After modifying the Eq(4) and Eq(5) , we proposed is

$$I_s = (I_b - I_a)\,\alpha' + I_a \qquad (6)$$

$$\alpha' = \frac{\Delta x}{(x_b - x_a)} \quad \textit{for DDA}$$

$$= \alpha \quad \textit{for interpolation}$$

where

I_a, I_b : intensities of (Xa,Ys),(Xb,Ys)

Δx : the steps from the initial pixel (Xa,Ys) along the scan-line

I_s : current intensity

By assigning different value to α', and select the operation mode, then we can integrate RGB α composition into RE successfully.

2.2 Approximated Phong shading method

The local reflection model calculation is necessary for each pixel in Phong shading algorithm. Theoretically, square-root operations are needed for each pixel in Phong reflection model to calculate the distance $|L|$ to the light source and distance $|V|$ to the viewer from a pixel to normalize the L and V vectors in Eq(1). Fig 3 shows the relationship between the light source and a pixel.

If the light source is far enough, the distance from the light source to the pixels in a local area will be almost equal. With this assumption, the pixel 1 in Fig 3 is chosen as the reference pixel, and use the distance *S1* to approximate *S2*. In the sense, instead of calculating each S for each pixel, a single S, said *approximated S* and it is *S1* in this example, is applied to approximate all S in a local area so that a lot of operations are eliminated, and the performance is greatly improved. The introduced error can be easily controlled by updating the *approximated S* according to an error threshold that is set by users. Therefore, approximated Phong shading is realizable.

Table 1 shows the operations needed to render a 10 × 10 polygon with gloss n=4.

Shade	Phong				
	+	−	×	÷	√
assumed cost	1	1	2	4	8
total Ops	2160	770	4780	260	200
Ops by CPU	500	170	1380	260	200
Ops by RE	1660	600	3100		
Shade	**Approximated Phong method**				
	+	−	×	÷	√
assumed cost	1	1	2	4	8
total Ops	1467	770	2998	62	8
Ops by CPU	104	170	192	62	2
Ops by RE	1363	600	2806		

Table 1. Operations for rendering a 10×10 polygon with gloss n=4:

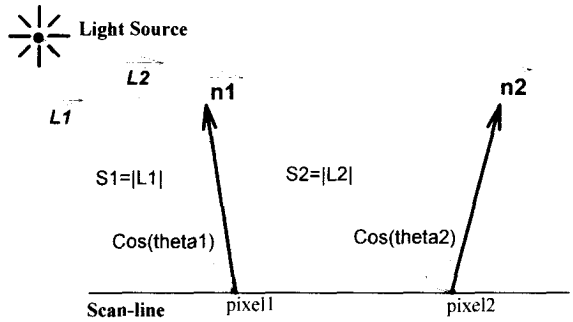


Fig 3. Schematic showing the light source and a pixel

According to Table 1, the $TotalOps(+,-,\times,\div,\surd) \times Cost(+,-,\times,\div,\surd)$ is 14530 for Phong algorithm, and 8497 for the approximated Phong method. By applying the approximated Phong method, about 40% total operations have been reduced. Thus, the performance can be greatly improved. The other two rows on the table will be discussed later.

3. HARDWARE DESIGN

3.1 Proposed architecture

There are three major functions to be executed sequentially in the rendering or composition process. They are interpolation for interpolating normal vector, the calculation of $\cos\alpha$ and $\cos^n\beta$ as shown in Eq(2), and the accumulation of R, G, B pixel values. The Interpolator implements the interpolation function, and the Phong shader implements the cosine functions and the accumulation of R, G, B pixel values, as shown in Fig 4. The block diagram of RE is shown in Fig 4. The $\frac{1}{Sh}$ and $\frac{1}{Sl}$ are parameters to normalize the vectors H and L, and the other parameters are the same as those in the reflection model.

The detail architecture of Interpolator is shown in Fig 5. There are two kinds of multiplexer **MUX** and **MUX*** shown in Fig 5. **MUX** is a normal multiplexer, while **MUX*** takes

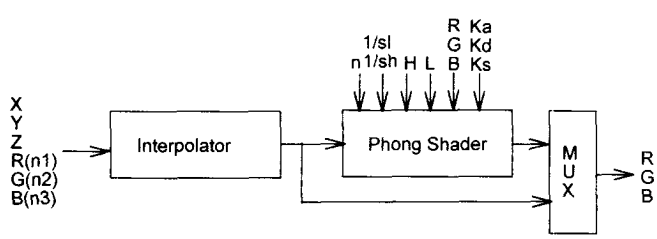


Fig 4. Block Diagram of RE

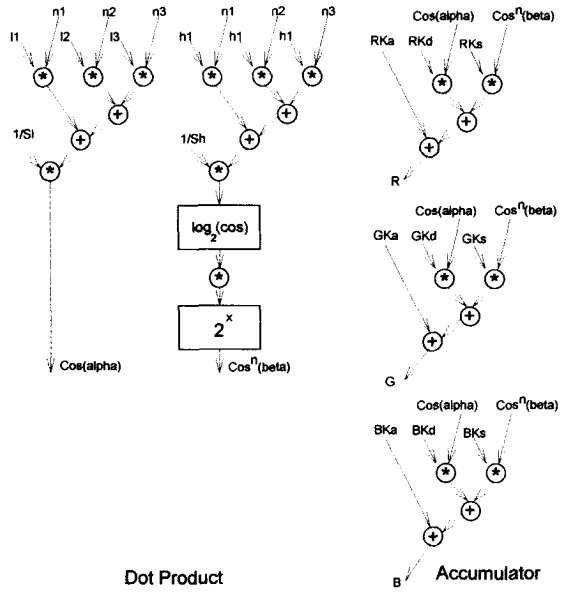


Fig 6. SFG of Phong Shader

1's complement of the input value to feed the following adder, so the adder can execute the subtract function directly.

To explore high performance and concurrence, 2-level pipeline and constant execution time for calculating $\cos^n \theta$ techniques are used in Phong shader.

3.1.1. 2-level pipeline

There are two major functions in the Phong shader, calculation of cosine function (dot-product of two normal vectors) and accumulation of pixel values. The Phong shader is pipelined into two stages according to its functional nature. In addition, there is a 2-substage pipelined multiplier to execute a dot-product or accumulate pixel values so that RE can speed up further.

3.1.2. Constant execution time for calculating $\cos^n \theta$

To achieve a constant throughput rate, the execution time for calculating $\cos^n \theta$ must be constant. Table Look-up technique can solve this problem [15]. The disadvantage is the number of tables depends on the gloss n. Another approach [16] is to approximate $\cos^n \theta$ piecewisely by second order polynomials. This approach suffers from too many operations and conditions to get the approximated value.

Our approach is to use two tables ($\log_2$ and 2^X, each has 512 words) with a multiplier and a barrel shifter to avoid the dependency on the gloss n. The operations consist of one multiplication, two Table Look-up and one shift. So, constant execution time for calculating $\cos^n \theta$ is achieved.

From the above discussion and the signal flow graph (SFG) of the reflection model, as shown in Fig 6, we design the architecture of the Phong shader as shown in Fig 7. In Fig 7, the (X, Y, Z) and (n1, n2, n3) denote the position and normal vector of a pixel. The values of **n1**, **n2**, **n3**, and **Z** are the results of the Interpolator, **X** is implemented by a up-counter to increase the X value step by step, and **Y** remains as a constant because RE renders a scan-line segment each time. The *COS(alpha)* and $COS^n(beta)$ correspond to $\cos \alpha$ and $\cos^n \beta$ separately. The values of n1, n2, n3, $\cos \alpha$ and $\cos^n \beta$ are between 0 and 1. Since RE can not cope with fraction, we scale n1, n2, n3 up to 16 bits, and *COS(alpha)*, $COS^n(beta)$ up to 9 bits. Then, during the operation they are shifted right by 16 bits or 9 bits to scale down the values to get the right answers.

3.2. Control circuit

Each functional stage has its own timing sequence, but the execution time for each stage must be the same because RE is a functional pipelining system.

According to the sequential characteristic, the control signals are generated by a counter-based finite state machine(FSM). This control circuit is a three level FSM. The first level is to determine the state of execution, for example, initial-setting state or execution state. The second level is for the sub-states in each functional stage. The third level is embedded in the multiplexer to reduce the control signal lines. For example, a 3-to-1 multiplexer usually needs two signals, S0 and S1, to select the correct input to output. Here, we use only one control signal to trigger the counter inside the multiplexer to select the correct input. Because of

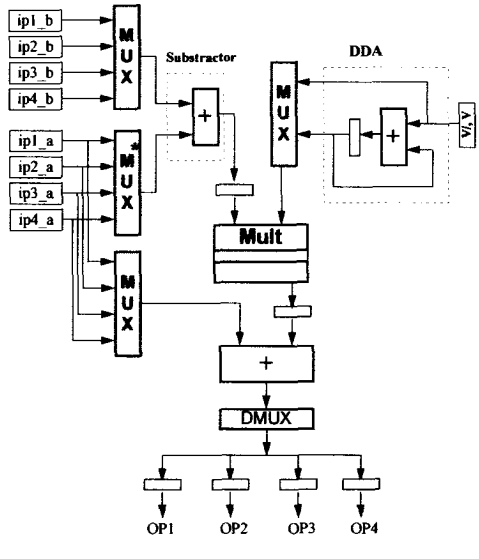


Fig 5 Detail design of Interpolation

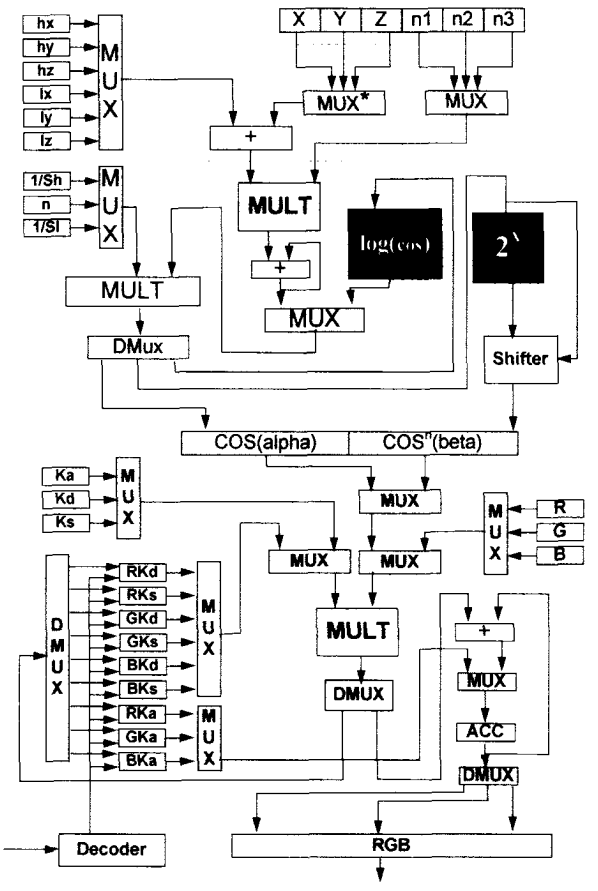


Fig 7. Architecture of Phong Shader

the 3-level FSM designs, the number of control signal lines are reduced almost by half.

3.3. Functions of RE

According to different operation modes, RE can perform composition, Gouraud shading, and Phong shading algorithm. In rendering mode, RE operates on scan-line basis. So, RE renders pixel values for the corresponding scan-line segment after receiving necessary parameters, for example, the coordinates and normal vectors (or intensities) of two ends of the scan-line segment, from the host CPU to RE. In composition mode, the host CPU transfers R, G, B, and α values to RE pixel by pixel, and the compositing operation is performed for each pixel.

Since each functional stage in RE has its own sequence, 16 system clocks are needed if Phong shading algorithm is selected; 8 system clocks are needed if Gouraud shading or composition mode is selected. The expected maximum system clock is 50MHz. Consequently, the pixel update rate is 3M pixels per second for Phong shading and 6M pixels per second for Gouraud shading and composition.

3.4 Operation analysis

According to Table 1, the proportions of operations performed by RE is derived as follows:

In Phong algorithm:

$$(Ops\ by\ CPU)\times Cost = 6070$$

$$(Ops\ by\ RE)\times Cost = 8460$$

$$\frac{Ops\ by\ RE}{Total\ Ops} = \frac{8640}{8640+6070}\times 100\% = 58.2\ \%$$

In approximated Phong method:

$$(Ops\ by\ CPU)\times Cost = 922$$

$$(Ops\ by\ RE)\times Cost = 7575$$

$$\frac{Ops\ by\ RE}{Total\ Ops} = \frac{7575}{7575+922}\times 100\% = 84.8\%$$

$$\frac{Reduced\ Ops\ by\ CPU}{Ops\ by\ CPU\ in\ Phong} = \frac{6070-922}{6070}\times 100\% = 89.1\%$$

The above shows that RE can release more than 50% CPU loads, and 89% CPU operations are reduced if the approximated Phong method is applied.

4. LAYOUT AND SIMULATION RESULT

4.1 The Layout View of RE System

The layout view of this design, Rendering Engine RE, are illustrated in Fig 8 respectively. The gate count of this chip is about 22K, and the die size is about $7684.5um \times 7444.8um$.

4.2 Simulation result

The RE is designed and implemented by using ITRI/CCL 0.8um SPDM CMOS cell library in the OPUS environment. Since Phong shading algorithm needs the Interpolator to interpolate normal vector and the Phong shader to calculate the pixel values, it is taken as a simulation example. The Verilog simulation result of Phong shading algorithm is shown in Fig 9. The (R, G, B)(see Fig 4) of the light source is (25, 200, 116), and (Ka, Kd, Ks) of the polygon is (0.2, 0.1, 0.1). The $\frac{1}{Sh}$=0.00116, $\frac{1}{Sl}$=0.00061, and gloss n=4. The $\boldsymbol{H}$ vector is (1669, 497, 275), and the $\boldsymbol{L}$ vector is (2140, -1110, 2100). The normal vector of the initial pixel is (0.229, 0.688, 0.688), and the incremental vector for the normal vector is (-0.0015, -0.0645, 0.0313). The coordinates of the two end pixels are (1, 1, 50) and (10, 1, 50). The simulation result shows that RE can operate up to 50MHz so that the pixel update rate is 3M pixels per second for Phong shading, 6M pixels per second for Gouraud shading and composition. Thus, 30K 10×10, Z-buffered, Phong shading polygons per second or 60K 10×10, Z-buffered, Gouraud shading polygons per second is possible. This chip has been fabricated in TSMC, Taiwan. The testing results show it can perform correct function at 50M Hz.

5. CONCLUSION

In this paper, we have proposed a PC-based low-cost raster engine to enhance the performance of computer graphics and image composition. For Phong shading, an approximated formulation has been derived which can reduce great computation amount without sacrificing the image quality. Some design techniques have also been revealed and included so that the hardware can be easily pipelined and a high throughput design can be resulted. This chip has been designed, verified, and fabricated in TSMC, Taiwan.

REFERENCE

[1] C.B.Harrell and F.Fouladi, "Graphics rendering architecture for high performance desktop workstation," *Computer Graphics Proceedings, Annual Conference Series 93*, pp.93-99, 1993.

[2] D.Kirk and D.Voorhies, " The rendering architecture of the DN10000VS," *Computer Graphics*, vol. 24, pp. 229 -307, Aug 1990.

[3] H.Fuchs, J.Poulton, J.Eyles, T.Greer, J.Goldfeather, D.Ellsworth, S.Molnar, G.Turk, B.Tebbs, and L.Israel, " Pixel-planes 5 : A heterogeneous multiprocessor graphics system using processor-enhanced memories," *Computer Graphics*, vol. 23, pp. 79-88, July 1989.

[4] K.Akeley and T.Jermoluk, "High-performance polygon rendering," *Computer Graphics*, vol. 22, pp. 239-246, Aug. 1988.

[5] K.Akeley, " Superworkstation the silicon graphics 4D/240GTX superworkstation," *IEEE Computer Graphics and Applications*, pp. 71-83, July 1989.

[6] K.Akeley, " Reality Engine graphics, " *Computer Graphics Proceedings, Annual Conference Series 93*, pp. 109-116, 1993.

[7] S.Molnar, J. Eyles, J.Poulton, "PixelFlow: High-Speed Rendering Using Image Composition," Computer Graphics, vol. 26, pp. 231-240, July 1992.

[8] M.F/Deering and S.R.Nelson, "LEO: A System for Cost Effective 3D Shaded graphics," *Computer Graphics Proceedings, Annual Conference Series 93*, pp.101-108,1993

[9] C.Dowdell and L.Thayer, "Scalable graphics enhancements for PA-RISC workstations," *IEEE Compcon*, spring 1992.

[10] K.Harney, M.Keith, C.Lavelle, L.D.Ryan, and D.J.Stark, "The i750 Video Processor: A Total Multimedia Solution ," *Communication of ACM*, vol.34, pp.65-78, April 1991.

[11] Foley, van Dam, Feiner, Hughes, *Computer Graphics: principles and practice, 2nd Edition*, Addison Weley,1990

[12] A.Watt, *3D Computer Graphics, 2nd Edition*, Addison Weley, 1993.

[13] T.Duff, "Composition 3-D rendered images," *Siggraph*, vol.19, pp.41-44, Nov. 1985.

[14] T.Porter and T.Duff, "Compostiing digital images, " *Computer graphics*, vol.18, pp253-259, 1984

[15] G.Knittel, "Verve: Voxel engine for real-time visualization and examination," *EUROGRAPHICS'93*, vol.12, no.3, pp.c-37 - c-48,1993

[16] J.Jayasinghe, A.Kuijk, and L.Spaanenburg, "A Display Controller for an Object-level Frame Store System," *Advances in Computer Graphics Hardware III* ,Springer-Verlag, 1990.

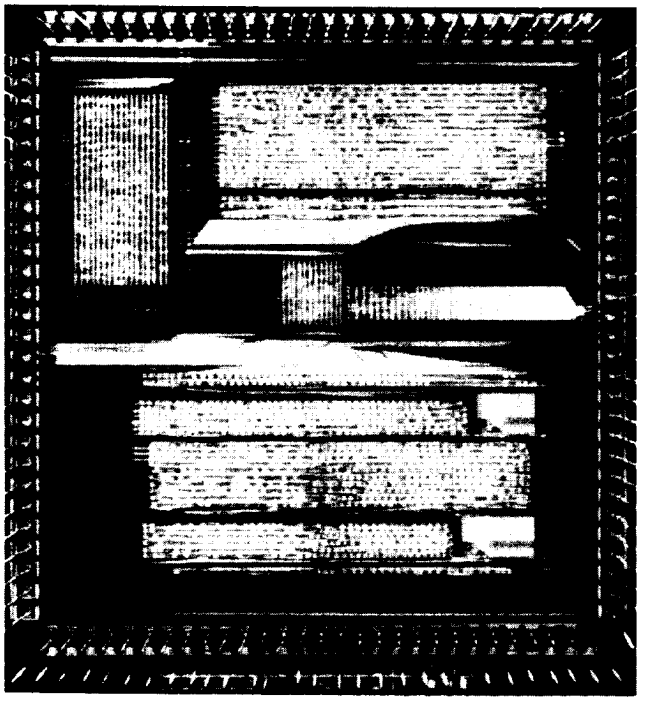


Fig 8. The photograph of RE chip

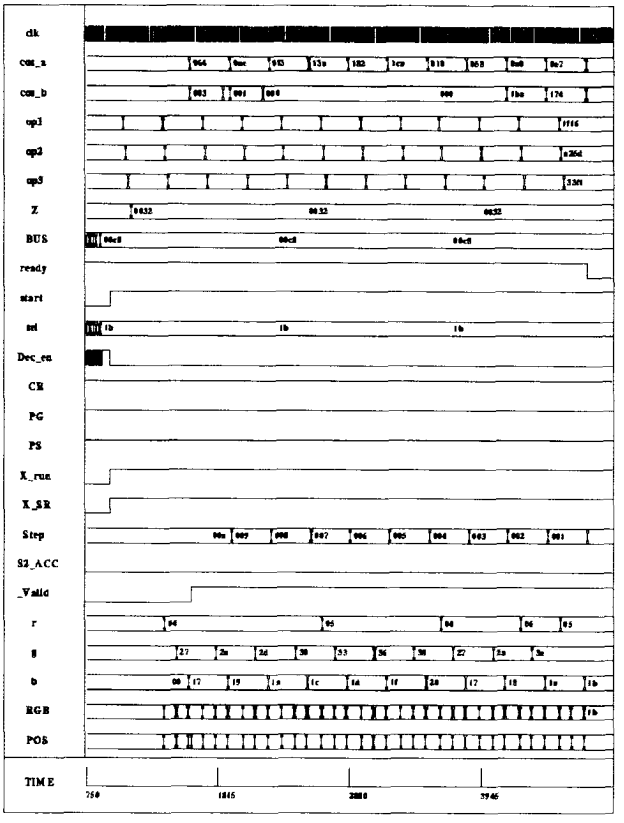
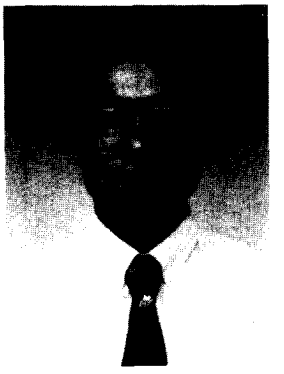


Fig 9. Result of Phong shading algorithm



Chein-Liang Chen received the B.S. degree in control engineering from National Chiao Tung University, Hsinchu, Taiwan in 1990 and the M.S. degree in electronic eng-ineering from National Chaio Tung Uni-versity in 1994.

His major research interests include 3D computer graphics, computer architecture and VLSI architecture for signal processing.

Chein-Wei Jen was born in Shanghai, China, in 1948. He received the B.S. degree from National Chiao Tung University in 1970, the M.S. degree from Stanford University in 1977, and the Ph.D. degree from National Chiao Tung University in 1983.

He is a professor in the Department of Electronics Engineering and the Institute of Electronics, National Chiao Tung University, Hsinchu, Taiwan. From 1985 to 1986 he was a Visiting Researcher at the University of Southern California, USA. His current research interests include VLSI signal processing, VLSI architecture design, design automation, and fault tolerant computing. He is a member of IEEE and Phi Tau Phi.

Bor-Sung Liang was born in Kaohsiung, Taiwan, R.O.C. in 1972. He received the B.S. degree in electronics eng-ineering from National Chiao Tung Uni-versity, Hsinchu, Taiwan in 1994.

His major research interests include 3D computer graphics, computer architecture and VLSI architecture for signal processing.