A Low Power 3D Rendering Engine with Two Texture Units and 29Mb Embedded DRAM for 3G Multimedia Terminals

Ramchan Woo, Sungdae Choi, Ju-Ho Sohn, Seong-Jun Song, and Hoi-Jun Yoo

Semiconductor System Laboratory, Department of EECS
Korea Advanced Institute of Science and Technology, Daejeon, Korea
{ural@eeinfo.kaist.ac.kr, hjyoo@ee.kaist.ac.kr}

Abstract

A low-power 3D rendering engine with 2 texture units and 29Mb embedded DRAM is designed and integrated into an LSI for portable 3G multimedia terminals. Texture-mapped 3D graphics with perspective-correct address calculation and bilinear MIPMAP filtering can be realized while consuming the low power with the help of clock gating, precision-controlled Look-Up Table dividers, texture address alignment and embedded DRAM. The performance is scalable and it reaches up to 100Mpixels/s and 400Mtexles/s at 50MHz. The chip is implemented with 0.16μm pure DRAM process to reduce the fabrication cost. The logic and DRAM consume 46mm^2 and 140mW at 33MHz operation. The 3D graphics images are successfully demonstrated by the fabricated chip on the PDA system board.

1. Introduction

As the mobile electronics market matures, 3G multimedia terminals such as PDAs or smart cell-phones are getting popular. And their applications are already migrating to the realtime multimedia, even to the 3D gaming applications [4]. Therefore, much research about hardware-accelerators [1-3] and software-only solutions [4-5] has tried to put 3D graphics rendering into the handheld devices. However, they are still below the market requirements showing only limited shading operations, without the texture mapping that is mandatory for the 3D gaming applications.

In order to draw texture-mapped 3D graphics on the mobile terminals, huge memory bandwidth and capacity must be provided to store the frame, depth and texture images. Therefore, Embedded Memory Logic (EML) process is one of the promising solutions since it integrates both DRAM and logic on a single die. However, this EML technology costs too much because the logic must be designed with the different transistors from the DRAM [1-3]. Therefore, it has not been widely used on the low-cost mobile platforms yet.

In this work, we designed and implemented a 3D rendering engine using the pure DRAM technology to reduce the fabrication cost while keeping the huge memory bandwidth [6]. Using the DRAM process enables us to further reduce the power consumption because off-chip loading to the rendering memory is completely eliminated. We optimize the circuits and architectures so that the rendering engine with two texture units and 29Mb embedded DRAM are realized while satisfying the requirements of the battery lifetime and the physical dimensions of mobile terminals.

2. System Architecture

The system architecture of the proposed rendering engine is shown in Fig. 1. It consists of a main pixel pipeline, a post processing unit and a dozen of rendering DRAMs. The main pixel pipeline performs shading and texturing with two pixel processors, each of which contains a high-performance texture unit inside.

After the pixel is being processed in the main pipeline, the post processing unit recalculates the pixel data for the realtime special effects such as antialiasing, motion blur, and fog [7]. The 29Mb rendering DRAMs contain frame buffers, depth buffers and texture memories. 12 independently-controlled DRAMs reduce the power consumption since the only necessary memories can be selectively activated.

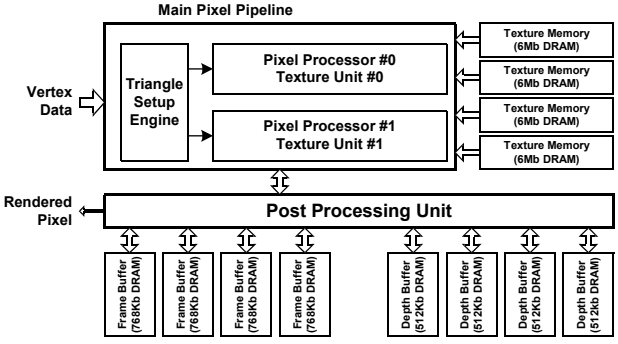
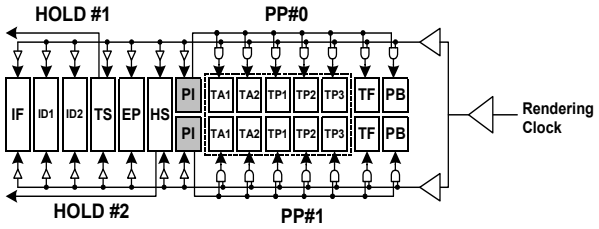


Figure 1 : Rendering Engine Architecture

3. Low Power Rendering Pipeline Structure

Fig. 2 shows the main rendering pipeline and describes its operation. It is composed of 14 multi-pipelined stages to maximally save the power consumption. After fetching the vertices and shaping the triangle, the rendering engine varies the operation cycles in the next stages according to the size (HOLD #1) and shape (HOLD #2) of the triangle by stopping the previous pipeline stages.



Pipe	Description
IF	Instruction Fetch, Main Power Control
ID1	Instruction Decode #1
ID2	Instruction Decode #2, Triangle Shaping
TS	Triangle Setup
EP	Edge Processor
HS	Horizontal Setup, Span Generation
PI	Pixel Interpolation, Depth-Comparison, Depth-Buffer Interface, Clock Gating Control
TA1	Texture Address #1, LOD calculation, 1/w dividision
TA2	Texture Address #2, Address Merging
TP1	Texture Prefetch #1, Bank Address Aggregation Texture Memory Command Generation
TP2	Texture Prefetch #2, Texture Memory Read
TP3	Texture Prefetch #3, Texture Data Alignment, Reverse Prodecure of Address Alignment
TF	Texture Filter
PB	Pixel Blending

Figure 2 : Main Rendering Pipeline

Also, the rendering engine suspends the following pipeline by gating off the clocks in each pixel processor according to the results of the depth-comparison in the PI stage. Therefore, we place the depth-compare-unit in the earlier pixel stage unlike the case in the high-performance PC graphics chipsets. Since the rendering engine contains two pixel processor (PP) and each PP contains its own texture unit fetching 4 texels/cycle, the pixel fill rate and texel rate are 100Mpixels/s and 400Mtexels/s at 50MHz, respectively.

Even though setting up the triangle took more than 7,000 cycles when it was calculated in the general purpose RISC processor, the previous work [1-3] didn't contain the hard-wired setup engine because of its logic complexity. In this work, however, we simplify the setup algorithm and implement it inside the TS stage to enhance the overall 3D performance.

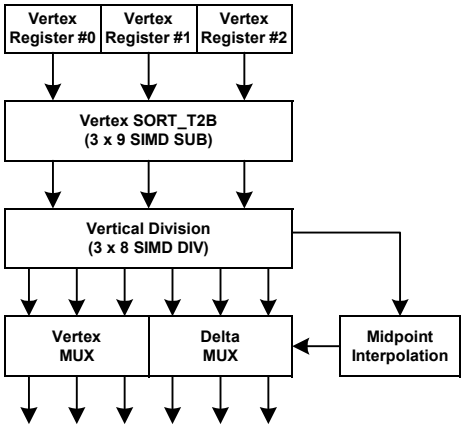


Figure 3 : Triangle Setup Engine

As shown in Fig. 3, it contains three 9-way SIMD SUBs, three 8-way SIMD DIV units and a midpoint-interpolation unit. The total calculation time from the vertex register to the final MUX is less than 20ns and it decides the maximum operation frequency of the rendering engine – 50MHz. Because the insufficient precision in the fixed point datapath results in the severe artifacts in the drawing of large polygon, we implemented the 8-way SIMD divider by using 8 integer multipliers, 8 shifters and one precision-controlled Look-Up Table (LUT) as shown in Fig. 4.

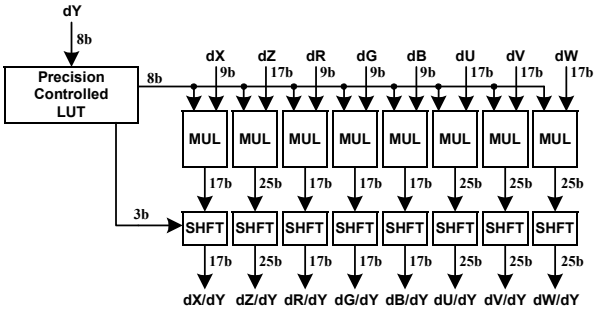


Figure 4 : 8-way SIMD Divider

Because the reciprocal value of the divisor is always equal to or smaller than one, fixed point representation of the reciprocal in the fixed 8-bit LUT can result in too much loss of the precision. The larger the divisor is, the more leading zeros occur in the LUT. Therefore, all leading zeros can be eliminated so that only meaningful 8-bit mantissa following after zeros and 3-bit corresponding fractional point locations are stored in the

LUT. To restore the fractional point, the results are right-shifted at the last stage after being multiplied with the mantissa. This precision-controlled fixed-point LUT divider consumes lower power and smaller area compared with the standard IEEE-754 floating-point divider while delivering the required precision for the setup operation.

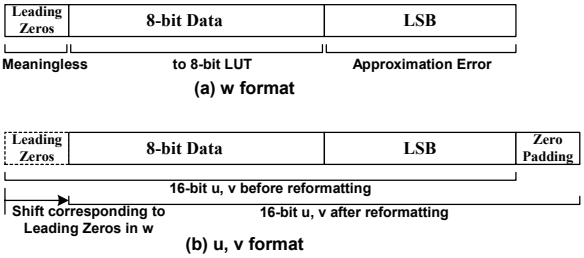


Figure 5 : (a) w format, (b) u, v format

4. Texture Unit

Even if the screen resolution of targeted PDA is limited, the rendering quality itself cannot be sacrificed. The rendering engine must calculate the pixels as correct as possible within the boundary of the required power consumption. Therefore, this rendering engine contains two texture units, each of which supports perspective-correct address calculation and bilinear MIPMAP texture filtering. To perspective-correctly calculate the texture address, per-pixel division is required. This operation can be described as the following equation:

$$U = u/w \text{ and } V = v/w \quad \text{..... [Eq. 1]}$$

$$(\text{where}, 0 \le (U, V) \le 1) \quad \text{..... [Eq. 2]}$$

$$\therefore w \ge u, \; w \ge v \quad \text{..... [Eq. 3]}$$

Each operand (u, v, and w) has 16-bit precision in the datapath so that 16-bit / 16-bit divider is required to calculate the perspective-correct texture address (U and V). However, by the definition of the texture addresses as shown in Eq. 2, the range of w can be limited as in Eq. 3. These facts can be used to reduce the power consumption and the area in the address calculation stage. The w can be represented in a binary form as in Fig. 5(a). The bits are composed of leading zeros, 8-bit data, and LSBs. We use only this 8-bit data component to search in the LUT since the leading zeros are meaningless. Even if trashing LSBs can cause 0.78% calculation error, it reduces the divisor bit-width from 16 to 8, resulting in more than 95% area reduction in the divider. Before being fed in the LUT divider, u and v are also reformatted to match w as in Fig. 5(b).

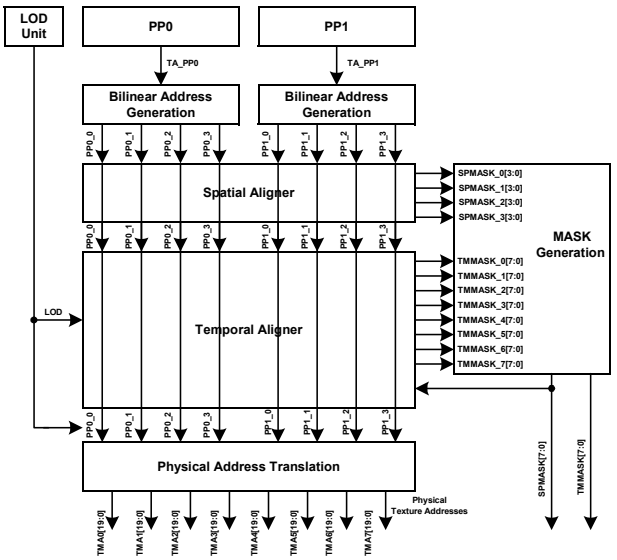


Figure 6 : Address Alignment Logic

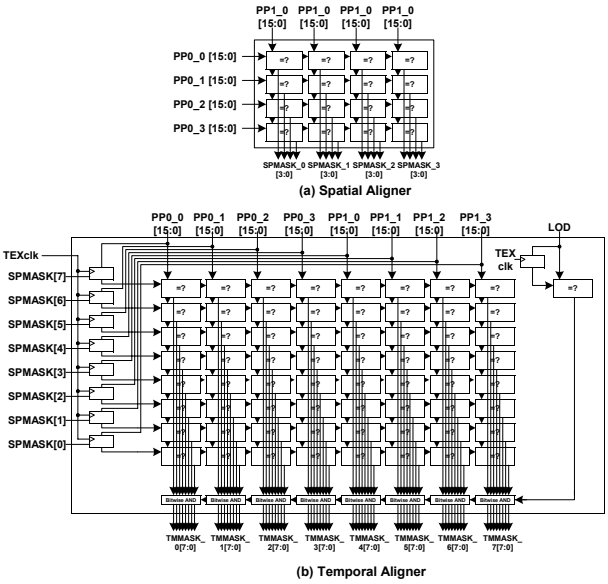


Figure 7 : AAL Circuit

The bilinear MIPMAP texture filtering is also implemented to improve the pixel quality further. This filtering generates as many as 8 texture memory requests at every cycle. But, Address alignment Logic (AAL) [Fig. 6] reduces these requests to 2.5 on an average working as a simple texture cache with Spatial Aligner and Temporal Aligner. By the definition of the MIPMAP selection, the texture footprints from two adjacent pixel processors are separated by approximately 1-texel

distance. Therefore, some of the texels can be merged together in the Spatial Aligner [Fig. 7(a)]. The Temporal Aligner [Fig. 7(b)] functions the similar operations of the spatial aligner in the time domain. Unlike the conventional texture cache architecture, additional data register is not necessary for exploiting the temporal locality because previous texel data is already stored in the pipeline latch.

5. Embedded DRAM Architecture

To save the power consumption of the Embedded DRAMs as well as to optimally utilize their bandwidth, we designed three different DRAM types. As described in Table 1, the characteristics of each memory are optimized according to their operation requirements. To cover the 256 x 256 screen resolution which covers the screen resolution of most of current cell phones, 4 frame macros and 4 depth macros are used in the chip. Also, 4 texture memory macros amount to 24Mb and store MIPMAP texture images for the 3D gaming applications. The Embedded DRAMs can provide 2.4GByte/s bandwidth which is sufficient for the 3D rendering at 50MHz.

Table 1 : Characteristics of Embedded DRAM Macro

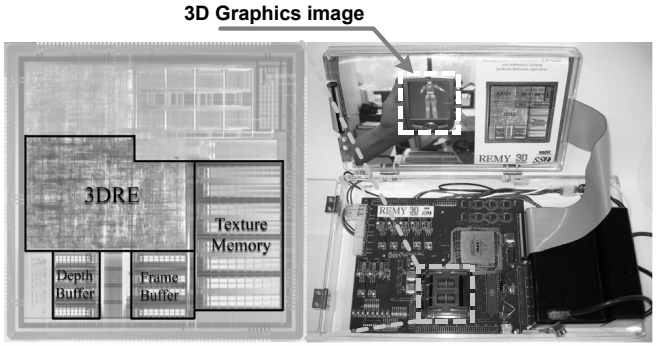
	Frame Buffer	Depth Buffer	Texture Memory
T_{RC}	20ns		
Macro Size	768Kbit	512Kbit	6Mbit
I/O Interface	24bit read 24bit write	16bit read 16bit write	24bit I/O
Commands	Read-Modify-Write Read Write Auto Refresh		Read Write Auto Refresh
Latency	0	0	1

6. Implementation

The 3D rendering engine with embedded DRAM is integrated into the PDA-LSI which contains a 32bit RISC processor and power management unit as well [6]. It is fabricated using 0.16um 1-W 3-Al DRAM process to implement both the logic and memory into the single chip with low fabrication cost. Fig. 8(a) shows the die photograph and table 2 summarizes its features. It can draw 24bit texture-mapped pixels with the drawing speed of 100Mpixels/s and 400Mtexels/s at 50MHz, which is 50 times faster than the minimum performance requirement of the PDA and cell-phones with 320x240-resolution LCD. The first silicon is successfully working and realtime 3D graphics images are demonstrated on the system evaluation board as shown in Fig. 8 (b).

Table 2 : Rendering Engine Features

Process Technology	0.16um DRAM 1-W 3-Al
Operation Frequency	5M ~ 50MHz (Scalable)
Components	3D Rendering Engine 29Mb DRAM
Power Supply	Rendering Logic : 2.5V Embedded DRAM :2.0V
Power Consumption with Embedded DRAM	140mW @ 33MHz (Texture Mapped) 80mW @ 33MHz (Shading Only)
Rendering Performance	10M ~ 100Mpixels/s 40M ~ 400Mtexels/s
Total Area	46mm^2



(A) (B)

Figure 8 : Die Photograph and System Evaluation Board

References

[1] Ramchan Woo, et al., "A 120mW 3D Rendering Engine with 6Mb Embedded DRAM and 3.2Gbyte/s Runtime Reconfigurable Bus for PDA-Chip," JSSC, pp. 1352-1355, Oct. 2002

[2] Chi-Weon Yoon, et al., "An 80/20MHz 160mW Multimedia Processor Integrated With Embedded DRAM, MPEG-4 Accelerator, and 3D Rendering Engine for Mobile Applications," ISSCC, pp. 142-143, 2001

[3] Yong-Ha Park, et al., "A 7.1GB/s Low-Power 3D Rendering Engine in 2D Array-Embedded Memory Logic CMOS," ISSCC, pp. 242-243, 2000

[4] Khronos Group, "Brining 3D Gaming to Cell Phones," Game Developers Conference 2003.

[5] Gopi K Kolli, "3D Graphics Optimizations for ARM Architecture," Game Developers Conference 2002.

[6] Ramchan Woo, et al, "A 210mW Graphics LSI Implementing Ful 3D Pipeline with 264Mtexels/s Texturing for Mobile Multimedia Applications", ISSCC pp 44-45, 2003

[7] Tomas Akenine-Moller, et al, "Real-Time Rendering", 2nd Ed., A K Peters, 2002.