Development of a 3-D Graphics Rendering Engine with Lighting Acceleration for Handheld Multimedia Systems

Byeong-Gyu Nam, Min-wuk Lee, and Hoi-Jun Yoo

Abstract — *A low-power three-dimensional (3-D) graphics rendering engine with lighting acceleration is designed and implemented for handheld multimedia terminals. The lighting unit is hardware implemented and integrated into the chip for the low-power acceleration of the 3D graphics applications. We adopt the following three steps to handle the memory bandwidth problem for rendering operations. I) We find bilinear MIPMAP is the best texture filtering algorithm for handheld systems based on our developed energy-efficiency metric. With this observation, we adopt bilinear MIPMAP for our texture filtering unit, which requires only 50% of texture memory bandwidth compared with trilinear MIPMAP filtering, II) We put the depth test operation into the earlier stage of the graphics pipeline, which eliminates texture memory accesses for invisible pixels, III) We develop a power-efficient small cache system as the interface to rendering memory. The accelerator takes 181K gates and the performance reaches 20Mpixels/s. A test chip is implemented with 1-poly 6-metal 0,18um CMOS technology. It operates at the frequency of 20MHz with 14.7mW power consumption[1].*

Index Terms — 3D Graphics, Lighting, Rendering Engine, Low Power, Handheld System.

I. INTRODUCTION

As the mobile electronics market increases rapidly, third-generation (3G) multimedia terminals such as cellular phones and personal digital assistants (PDAs) are becoming more popular. They already require real-time multimedia applications such as MP3, MPEG-4 audiovisual codec and even three-dimensional (3-D) computer graphics.

For mobile wireless applications, power consumption is one of the main problems due to their limited battery lifetime. Moreover, since real-time 3-D computer graphics inherently requires extremely large computing power and memory bandwidth, the realization of a 3-D graphics system in a mobile computing environment is a challenging problem.

Recently, there have been several researches on the low power 3-D graphics for hand-held devices, including hardware accelerators [1]-[4] as well as software libraries [5]-[6]. In hardware acceleration approaches, even if the rendering stage is accelerated by hardware, the geometry stage is still mainly implemented with software [6] or with limited hardware function of arithmetic units [1], which leads to extra power overhead. In ordinary cases for wireless applications, the RISC based application processors (AP) [7] or digital signal processors (DSP) [8] are used to process geometry stage computing. Although their simple datapath such as multiply-and-accumulate (MAC) unit is suitable for the transform operation of geometry stage computing, they can not provide the required performance of lighting or vertex shading operations, which require complex special functions like reciprocal square root or powering operation in geometry processing.

In this work, we designed and implemented a 3-D graphics rendering engine augmented with lighting capability in order to increase its performance and power efficiency. The architectures of the complex reciprocal square root (RSQ) and the powering (POW) units, required by the OpenGL lighting equation are optimized for the lighting engine of the real-time 3-D graphics applications on handheld devices.

The requirement of huge memory bandwidth required for texture mapping, depth test and alpha blending is critical for a high rendering performance. Previous works used embedded memory as a solution to this problem [3][4]. However, integrating large memory on a single die with logics brings increases of chip size and fabrication cost. As an alternative solution, in this architecture, three kinds of mechanisms are adopted to reduce external memory bandwidth requirement effectively for the rendering operations. The texture filtering algorithms are analyzed and the bilinear MIPMAP filtering [9] is adopted for the reduced memory bandwidth and the best energy efficiency. Moreover, the depth test is placed into the earlier stage of the graphics pipeline to remove unnecessary texture memory access for invisible pixels from the viewer. Also, a cache system for the rendering memory transactions is introduced: texture cache, depth cache and pixel cache as the interface to texture memory, depth buffer and frame buffer, respectively. Although large cache memory can reduce the miss rate significantly, it takes large area and its parallel search of large number of tags leads to huge power consumption. Therefore, a small size cache system is implemented to reduce power overhead.

With all of the features mentioned above, the accelerator reduces computing overhead from the AP by offloading lighting and rendering computations and consequently realizes low-power consumption.

This paper is organized as follows. In the next section, we present target specifications of a 3-D accelerator for the current target 3G systems. In section 3, we explain the detailed architecture of the rendering engine with the proposed lighting engine and memory bandwidth reduction schemes to attain high performance and low power consumption.

[1] The authors are with the Semiconductor System Laboratory, Department of Electrical Engineering and Computer Science, Korea Advanced Institute of Science and Technology (KAIST), Daejeon 305-701, Korea (e-mail: bgnam@eeinfo.kaist.ac.kr; hjyoo@ee.kaist.ac.kr).

Contributed Paper
Manuscript received July 15, 2005

Implementation results are presented in section 4. Finally we conclude in section 5.

II. Target System

Fig. 1-(a) shows a 3G system that contains a RF frontend, a baseband modem for communication, an application processor for multimedia processing, memories, and peripherals. There are three kinds of requirements to be met for the 3-D graphics to be realized in these systems, i.e. the performance of 3-D graphics, memory bandwidth, and power consumption. In this section, we will cover the specific requirements for the above issues.

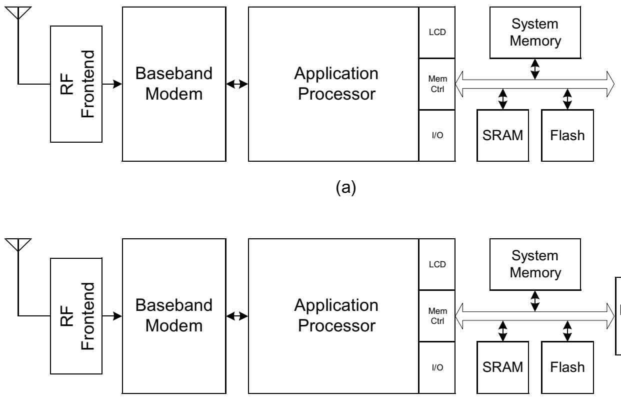


Fig. 1. Examples of target 3G system. (a) Conventional 3G system, (b) 3G system with RamP-Lite.

A. Performance

We analyzed the performance of 3-D graphics pipeline to identify the system level performance need. In this analysis, we used our developed 3-D graphics library, Mobile-GL, which is compatible with OpenGL-ES [10] and optimized with the fixed-point arithmetic for the 3-D graphics pipeline [11].

Fig. 2-(a) shows the rendering performance when the 3-D graphics pipeline implemented with Mobile-GL runs on the 400MHz PXA-255 application processor [7]. For QVGA (240×320) screen resolution, pixel fill rate of 11.5Mpixels/s is required for 30 frames/s and average depth complexity of five, which is the usual case for the handheld 3-D graphics applications. However, the graph shows the performance is far below the requirement. To solve this problem, the rendering stages should be accelerated since the rendering stage takes 77% of processing power in the 3-D graphics pipeline.

Although the rendering stages are accelerated by hardware, the geometry stage is still the next performance bottleneck, which is described in Fig. 2-(b). Under the same drawing complexity with rendering operations, the geometry stage should process 1.15Mpolyons/s since the average pixel count in a polygon is about ten [11]. However, the polygon processing rate is still under the required performance.

This problem can be solved if the lighting part, which takes about 83% of the geometry computation, is also hardware accelerated. This is shown in Fig. 2-(c).

B. Memory Bandwidth

In general, 3-D graphics rendering requires a huge amount of memory bandwidth for rendering operations like texture mapping, depth test and alpha blending. Recently, mobile DDR SDRAM can provide maximum bandwidth of 400MB/s, but less than 50% is available due to the bus contention by multiple IPs in current handheld systems [12]. Also, the vertex data transfer from the host processor to rendering engine takes about 40MB/s. Therefore, less than 150MB/s of the memory bandwidth can be assigned to pure rendering operations.

However, the rendering engine with trilinear MIPMAP filtered texture mapping operation [9], which is an ordinary case for PC graphics systems requires 440MB/s for its rendering performance of 20Mpixels/s. Therefore, we have to find a proper texture filtering algorithm for our target systems.

C. Power Consumption

The most important design factor for the handheld devices is the low power consumption because of their limited battery lifetime. Since current Li-ion battery can supply 2000mWh, system power budget for LCD, CPU, memory and others is only about 800 mW for a 2~3-hours [13]. Under this situation, only 200~300 mW can be allocated to the 3-D rendering engine including the rendering memories, thus, just tens of milliwatts can be assigned to the 3-D rendering engine core.

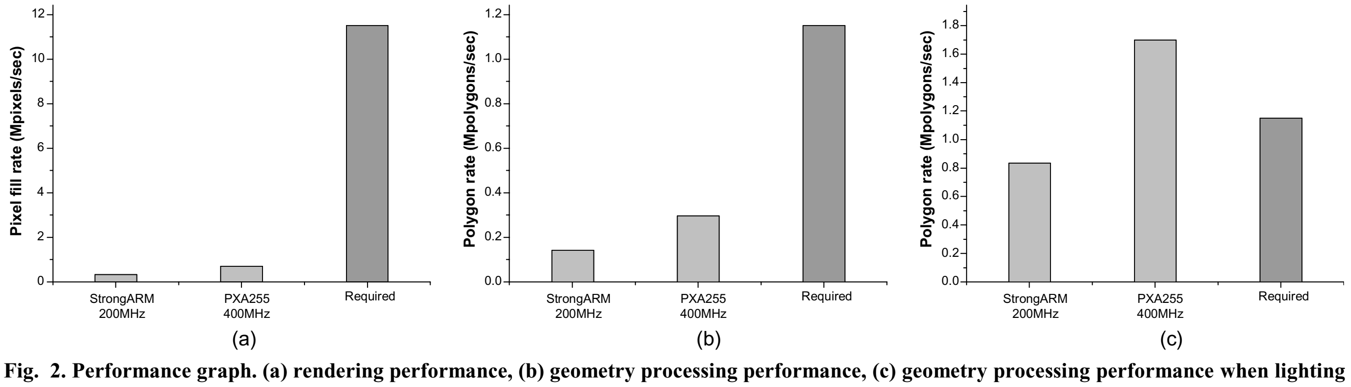


Fig. 2. Performance graph. (a) rendering performance, (b) geometry processing performance, (c) geometry processing performance when lighting is hardware accelerated..

To realize the 3-D graphics on handheld systems under these system constraints mentioned above, we have to design a hardware accelerator for rendering and lighting operations to meet the performance and power constraints. In this accelerator, we also need some kind of schemes to handle memory bandwidth requirements effectively.

We propose a small power-efficient rendering engine, named as RamP-Lite, augmented with the lighting engine, the energy efficient texture filtering unit and the small cache system to realize the 3-D graphics on handheld devices under the constraints mentioned above.

In the system configuration of Fig. 1-(a), our graphics accelerator can be attached as a companion chip to the application processor. Our goal is to offload the computation overhead from the application processor in processing real-time 3-D graphics applications. Consequently, we can attain high performance and low power consumption by utilizing dedicated hardware for computation intensive operations.

As a result, the remaining parts of the graphics pipeline such as transformation, clipping and projection can be easily implemented as software by utilizing the MAC unit of the conventional AP that is already installed in current handheld systems. The RamP-Lite resolves performance bottlenecks of the conventional 3-D graphics pipeline on handheld devices with minimal modifications to existing systems. Fig. 1-(b) shows the modification.

III. 3D Graphics Accelerator

In order to design a power-efficient rendering engine, we adopt several kinds of approximation techniques for the pipeline design at the cost of negligible image quality loss. Details of the techniques for lighting and texture filtering will be presented in the following sections.

The pipeline of RamP-Lite is mainly composed of two parts: geometry and rendering stages. We mainly adopted the fixed-point arithmetic units for the entire pipeline since they consume less power than floating point units [1][5][6]. Fig. 3 shows the organization of the main pipeline.

A. Lighting Engine

In order to support lighting in a separated chip from the AP, where other parts of geometry computations are processed, conventional 3-D graphics pipeline dataflow [14] should be modified. Fig. 4 describes this modification.

Although this modification is similar to the deferred lighting proposed in [15], the deferred lighting that processes lighting in the middle of the rendering process requires completion of lighting operation in one rendering cycle in order to preserve the throughput of the rendering engine. Moreover, they assume a software solution to the lighting engine, while our focus is on the hardware acceleration in the pipeline of current handheld system environment.

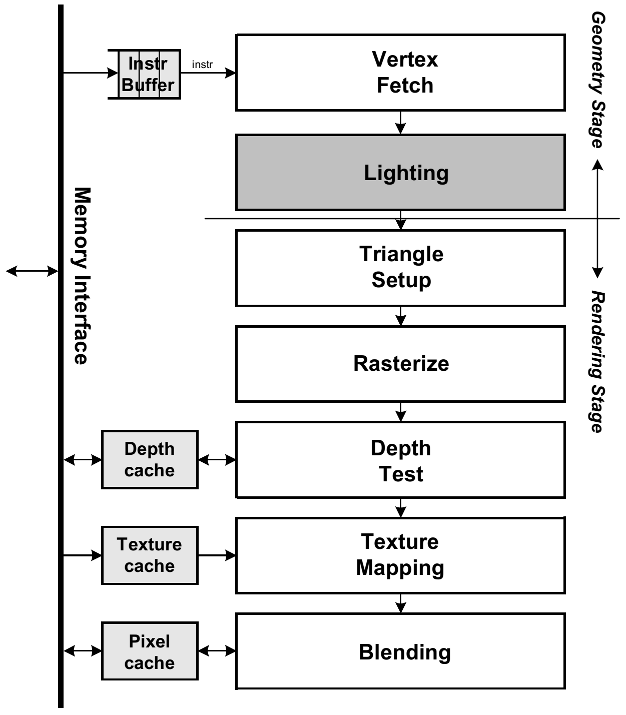


Fig. 3. The organization of RamP-Lite.

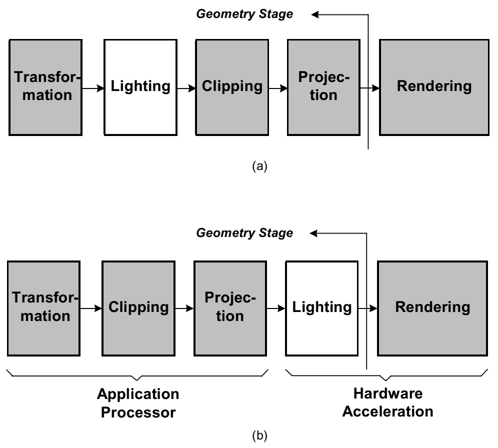


Fig. 4. Pipeline reordering. (a) Original 3-D graphics pipeline, (b) Proposed pipeline.

Since the lighting stage is moved to a hardware accelerator (i.e. the next stage of projection), the parameters needed for lighting e.g. normal (N), view (V), light (L) vectors and material properties should be transferred to the accelerator. These parameters should bypass the projection stage on the application processor to insure that they are not distorted by the projection stage for correct lighting computation.

We used two approaches to reduce the overhead from the hardware implementation since direct implementation of the lighting engine into the hardware requires too much area overhead. Firstly, we implemented the lighting pipeline stages with multiple cycle operations by folding the internal arithmetic units three times. The rendering throughput degradation from this approach is negligible since less than 3% of triangles in a QVGA screen size are composed of less than three pixels from our statistics. On the other hand, every rendering pipeline stages proceed in one cycle to lower operating clock frequency without sacrificing the rendering performance for low power design.

Secondly, we simplified the format of the lighting parameters into Q8.8[2] instead of direct implementation of Q16.16. We found that about 96% of the lighting parameters can be represented by Q8.8. The maximum errors are 1% for normal and view vectors, respectively, and 3% for light vectors with this representation.

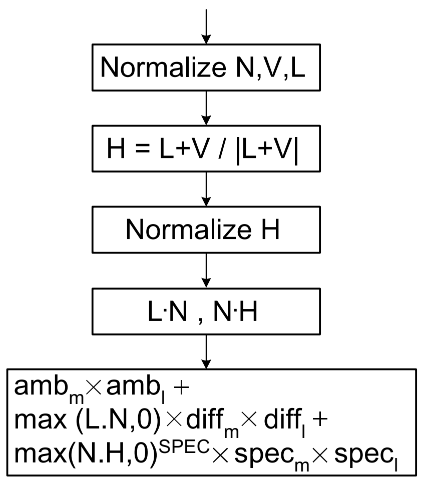

Normalize N,V,L
H = L+V / |L+V|
Normalize H
L·N , N·H
amb_m × amb_l + max (L.N,0) × diff_m × diff_l + max(N.H,0)^SPEC × spec_m × spec_l

Fig. 5. The pipeline of lighting engine.

Once the lighting parameters are fed into the lighting engine, they pass through the pipeline shown in Fig. 5, which implements the OpenGL lighting equation [14]. Each stage proceeds in three cycles. The N,V,L vectors are normalized first and the half vector H is evaluated. The H vector is also normalized to evaluate the lighting equation. There are two special operations needed for the evaluation of this lighting equation, i.e. RSQ for the normalization of the vectors and POW for computation of the specular term. The functional units for these operations are shown in Fig. 6.

Basically, the lighting parameters fed into the engine is managed and stored as fixed-point numbers. However, when special functions like RSQ or POW are necessary, the data is temporarily converted to floating point numbers since insufficient precision in the fixed-point datapath may result in severe artifacts when drawing large polygons. Then, the results of the operation return to fixed-point numbers.

[2] Qm.n represents the format of a fixed point number, where 'm' represents the number of bits used for integer part and 'n' represents for that of fraction part.

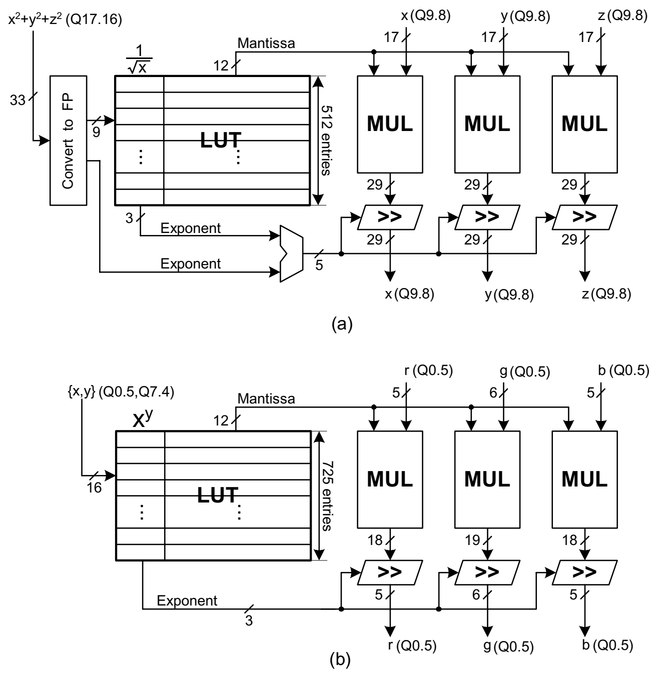


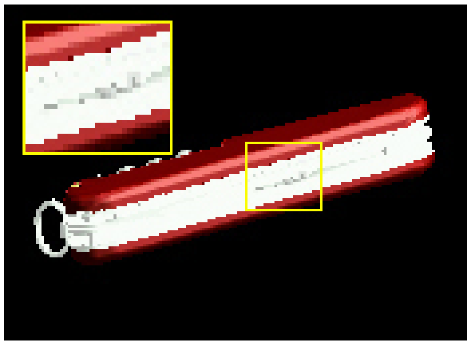
Fig. 6. Special function units. (a) Reciprocal square root (RSQ) unit, (b) Powering (POW) unit.

For the normalization operation, we used one precision-controlled look-up table (LUT) for the RSQ and 3 multipliers and shifters for each vector component respectively. In the precision-controlled LUT, all leading zeros are eliminated and only meaningful 12-bit mantissa and a 3-bit exponent for corresponding fractional point locations are stored. The mantissa is used for the multiplier operand in evaluating the characteristic value of the RSQ result. The exponent is used for shifting the characteristic value to make the actual RSQ result.

To address the LUT, the sum of square (SSQ) of each vector component is used. However, we do not directly supply the SSQ value as the address since 33-bit (Q17.16) SSQ value for the address requires overly large LUT. Therefore, we convert the SSQ to the floating point format, 6-bit exponent and 9-bit mantissa, by dropping all leading zeros and taking only the nine most significant bits (MSBs) for the address simplification. By using this 9-bit mantissa for the LUT address, 512-entry LUT with 12-bit mantissa and 3-bit exponent can be used. The 3-bit exponent from the LUT should be compensated with the 6-bit exponent from the floating point SSQ.

For the powering operation, the architecture of the functional unit is basically similar to the RSQ unit. In this architecture, the simplification for the input value is also taken to reduce LUT size. Since the input range of the powering operation x^y for OpenGL specular lighting is $x \in [0,1]$ and $y \in [0,128]$, we restricted the format of x and y as Q0.5 and Q7.4, respectively. Thus, 725-entry LUT with 12-bit mantissa and 3-bit exponent can be used for POW operation.

These precision-controlled LUT based units save the power and the area of the lighting engine by 85% and 75%, respectively, compared with the IEEE-754 single-precision floating-point units while delivering the required precision. In Fig. 7, a lit, smooth-shaded scene with different material properties for each part is rendered to show reliability of the proposed scheme.


(a)

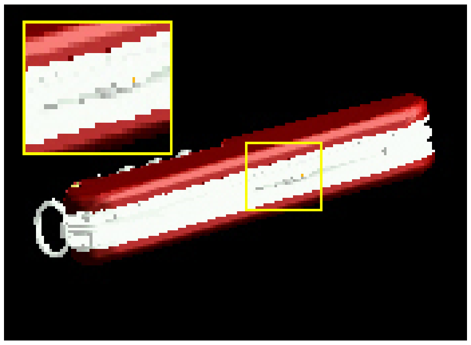

(b)

Fig. 7. Rendering effects for reduced precision. (a) Lighting with Q16.16, (b) Lighting with Q8.8.

B. *Rendering Engine*

As shown in Fig. 3, the rendering engine of RamP-Lite mainly consists of five stages. Each stage performs the rendering operations which include triangle setup, shading, texture mapping, depth test and blending.

In RamP-Lite, we adopt three steps to reduce the external memory bandwidth for rendering operations. Since the texture mapping is the main source of memory bandwidth requirement, we mainly focus on the reduction of texture memory transaction.

Firstly, we use bilinear MIPMAP filtering algorithm for texture filtering. Usually, high-end 3D graphics rendering engines adopt trilinear MIPMAP filtering to enhance their image quality extremely. However, it is too expensive for the handheld systems with small screen size since trilinear MIPMAP filtering requires large energy consumption as well as a huge amount of memory bandwidth, i.e. requiring eight texels for each pixel. Therefore, we should select a cost-effective filtering algorithm for handheld systems.

In order to find the proper one, we analyzed the energy efficiency of each filtering algorithm based on the energy consumption and image quality of scenes generated by them. The energy consumption is extracted from our simulator, which models the energy consumption for the capacitance of the external system bus and memory. The energy for each filtering algorithm is shown in Fig. 9-(a). We analyze the image quality based on the aliasing factor of the scene since texture filtering is for the antialiasing of a texture-mapped image. Therefore, we perform discrete cosine transform (DCT) for the test scenes to get the information of high-frequency factor in the scene that contributes to aliasing. The extracted frequency spectrum and its coefficient matrix are used for evaluating the *image quality* metric defined by us as in equation (1). We define the image quality of a scene as the sum of DCT coefficients weighted inversely proportional to their corresponding frequency.

$$image\_quality = \sum_{u=0}^{\pi}\sum_{v=0}^{\pi}\frac{F(u,v)}{u+v} \qquad (1)$$

$$where\ F(u,v)\ is\ the\ 2D\ DCT\ coefficients$$

The DCT results are shown in Fig. 8 and evaluated image quality indices are shown in Fig. 9-(b). To proceed the analysis, we define another term, the *energy efficiency* of a texture filtering algorithm as in equation (2).

$$energy\_efficiency = \frac{\Delta image\_quality}{\Delta energy} \qquad (2)$$

In Fig. 9, we should notice that the energy consumption significantly increases, while the image quality does not increase so much when trilinear MIPMAP filtering is used. This means that the energy efficiency drops abruptly at the trilinear MIPMAP point, which is shown in Fig. 9-(c). Therefore, we adopt bilinear MIPMAP filtering which requires four texels for rendering one pixel. With this algorithm, we can reduce the external memory bandwidth requirement to 280MB/s.

In order to reduce the memory bandwidth requirement further, we moved the depth test stage to the earlier stage of pipeline before the texture mapping stage. In conventional 3D graphics, the depth test stage is located in the final stage of the pipeline, where it culls pixels invisible from the viewer and prevents it from being drawn into the frame buffer. We can utilize this property to avoid texture memory access for the invisible pixels, which can lead to memory bandwidth reduction. Since about 40% of pixels fail this depth test, this scheme can reduce the average external memory bandwidth requirement to 200MB/s. Fig. 10 shows the pipeline ordering for the early depth test.

Also, we adopt a power-efficient small cache system to effectively reduce the external memory bandwidth requirement to the level of 150MB/s under. For this cache system, we use 16 entries for texture cache and 8 entries for depth and pixel cache, respectively. The miss rate for these small caches are

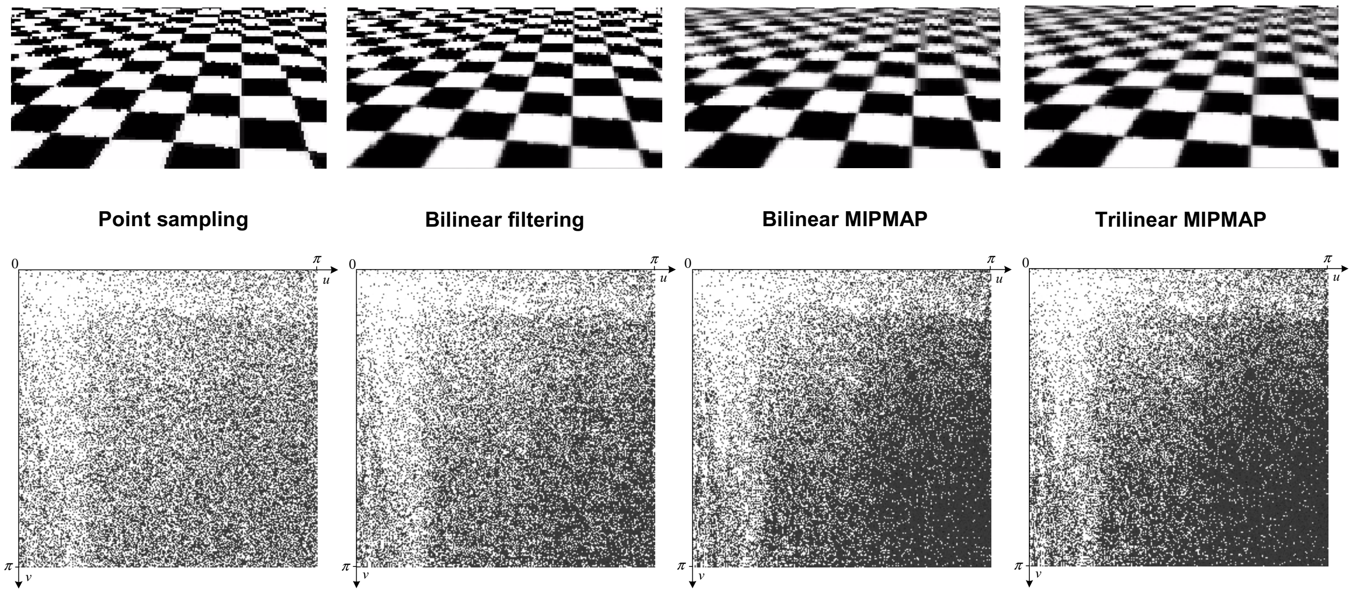


Fig. 8. The DCT results (below) for rendered images (above) by different texture filtering algorithms. The 2D DCT coefficients are represented as points with gray level intensity. (a) Point sampling, (b) Bilinear filtering, (c) Bilinear MIPMAP, (d) Trilinear MIPMAP.

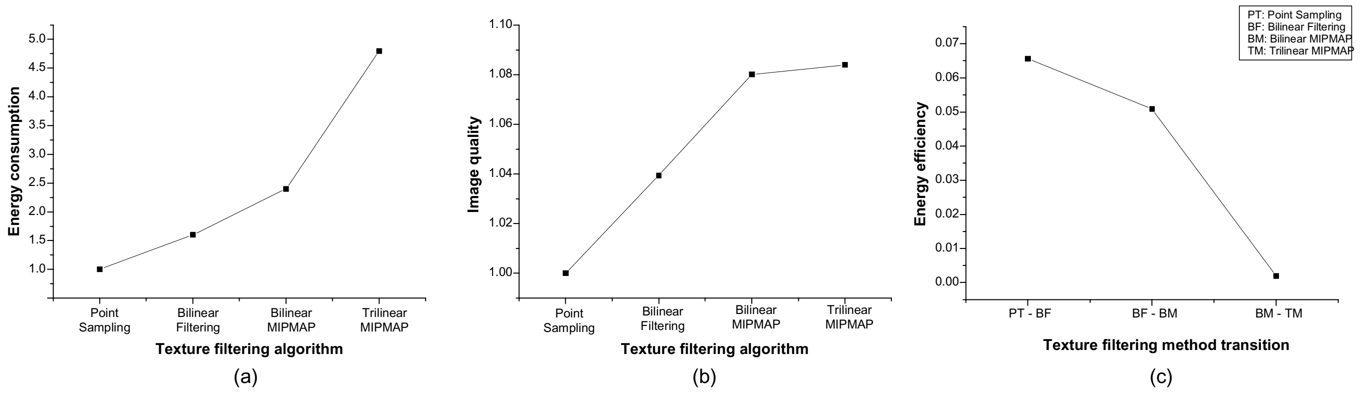


Fig. 9. The comparison of energy and image quality indices for the texture filtering algorithms. These values are normalized to the point sampling case. (a) Energy consumption, (b) Image quality, (c) Energy efficiency.

64%, 29% and 35%, respectively. Although the miss rates are fairly high, this can reduce the external memory bandwidth requirement to 93MB/s, which is the amount the current handheld system can support for rendering engine. Fig. 11 shows the transition of the external memory bandwidth requirement according to the schemes we adopt.

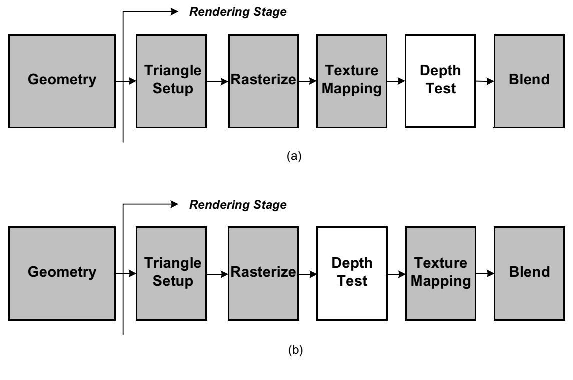


Fig. 10. Depth test reordering. (a) Original 3-D graphics pipeline, (b) Early depth test.

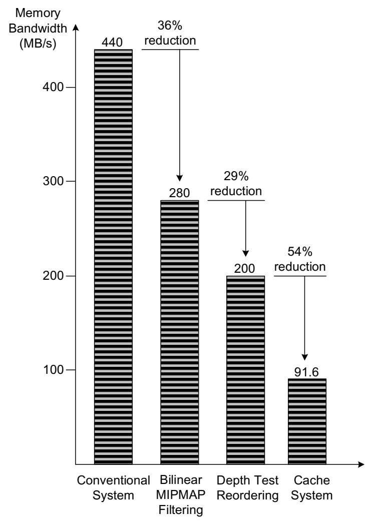


Fig. 11. Memory bandwidth reduction of RamP-Lite.

IV. Implementation Results

The target system is modeled and prototype FPGA system is developed. The system incorporates QVGA size LCD screen. The host is the PXA-255 processor [7] and it manages the entire system including the system memory. In RamP-Lite, the cache system runs at a four times higher clock frequency than that of the rendering core to process tag-comparison and Read-Modify-Write (RMW) operations in one cycle time of the core clock. This is needed for depth-test and alpha-blending operations to sustain the maximum throughput of the engine. The interrupt service routine is implemented for the host to service memory transaction requests from the RamP-Lite under its cache misses. The software library is implemented and runs on the PXA-255. It is aimed to support a 3D graphics library for the geometry operations like transformation, clipping and projection.

Based on the FPGA implementation result, we have developed a fully synthesizable Verilog-HDL model for the RamP-Lite and implemented it into a test chip with Samsung 1-poly 6-metal 0.18um CMOS technology. The synthesized logic consists of 181K gates. The die size is 5mm × 5mm while the graphics core takes only 1.59mm × 1.59mm. And it operates at the frequency of 20MHz with 14.7mW power consumption. A chip microphotograph is shown in Fig. 12 and prototype system in Fig. 13.

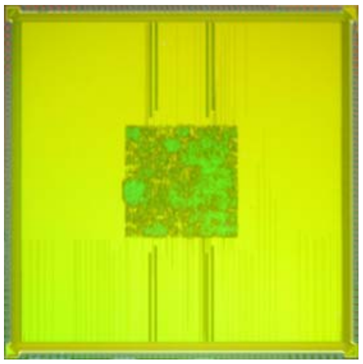


Fig. 12. Chip microphotograph.

Fig. 13. Prototype system.

TABLE I
Chip Specification

Unit	Unit Symbol
Process technology	Samsung 0.18 1-poly 6-metal CMOS
Supply voltage	Core: 1.8V, I/O: 3.3V
Operating frequency	Main pipeline: 20MHz, Cache: 80MHz
Power consumption	14.7mW
Gate counts	181K gates
Chip size	Die: 5mm × 5mm (pad limited)
	Core: 1.59mm × 1.59mm
Package	208pin QFP

V. Conclusion

A low-power three-dimensional (3-D) graphics rendering engine augmented with lighting acceleration capability is designed and implemented for handheld multimedia terminals. The lighting unit is hardware implemented and integrated into the chip for the low power acceleration of the 3D graphics applications. The lighting engine uses approximated parameter values for the lighting computation, which leads to 85% power saving with a maximum 3% error. In order to cope with the memory bandwidth problem, three steps are used. We adopt the bilinear MIPMAP filtering algorithm based on our proposed energy-efficiency metric for the texture filtering algorithms. Based on this metric, the bilinear MIPMAP texture filtering is the best for the low-power rendering engine. The depth test stage is put into the earlier stage of the pipeline to eliminate the texture memory access for invisible pixel rendering. The power efficient, small rendering cache system was also proposed. The test chip runs at 20MHz with pixel fill rate of 20Mpixels/s and power consumption of 14.7mW. The implementation results show that the proposed schemes are suitable for the low-power handheld systems with small size screen.

References

[1] J.-H. Sohn, J.-H. Woo, M. Lee, H.-J. Kim, R. Woo, and H.-J. Yoo, "A 50Mvertices/s Graphics Processor with Fixed-Point Programmable Vertex Shader for Mobile Applications," *ISSCC Digest of Technical Papers*, 2005.

[2] M. Imai, T. Nagasaki, J. Sakamoto, H. Takeuchi, H. Nagano, S. Iwasaki, and et al., "A 109.5mW 1.2V 600Mtexels/s 3-D Graphics Engine," *ISSCC Digest of Technical Papers*, 2004.

[3] R. Woo, S. Choi, J.-H. Sohn, S.-J. Song, Y.-D. Bae, C.-W. Yoon, and et al., "A 210-mW Graphics LSI Implementing Full 3-D Pipeline With 264 Mtexels/s Texturing for Mobile Multimedia Appliocations," *IEEE Journal of Solid State Circuits*, Vol.39, Feb. 2003.

[4] M. Kameyama, Y. Kato, H. Fujimoto, H. Negishi, Y. Kodama, Y. Inoue, and H. Kawai. "3D Graphics LSI Core for Mobile Phone - Z3D," *SIGGRAPH/Eurographics Workshop on Graphics Hardware* 2003.

[5] G. K. Kolli, "3D Graphics Optimization for ARM architecture," *Game Developer Conference*, 2002.

[6] K. Yosida, T. Sakamoto, and T. Hase, "A 3D Graphics Library for 32-bit Microprocessor for Embedded Systems," *IEEE Transactions on Consumer Electronics*, Vol.44, Aug. 1998.

[7] Intel PXA-255 Processor Developer's Manual, *Intel Corporation*, March, 2003.

[8] OMAP5910 Dual-Core Processor Technical Reference Manual, *Texas Instrument Incorporated*, July. 2002.

[9] J. P. Ewins, M. D. Waller, M. White, and P. F. Lister, "MIP-Map Level Selection for Texture Mapping," *IEEE Transactions on Visualization and Computer Graphics*, Vol. 4, No. 4, Oct.-Dec. 1998.
[10] OpenGL-ES Common/Common-Lite Profile Specification Version 1.0, *Khronos Group*, 2003
[11] M.-W. Lee, B.-G. Nam, J.-H. Sohn, N. Cho, H. Kim, K. Kim, and H.-J. Yoo, "A Fixed-point 3D Graphics Library with Energy-efficient Cache Architecture for Mobile Multimedia Systems," *ISCAS*, 2005.
[12] PowerVR-MBX, available: http://www.arm.com/products/solutions/3Dgraphics.html
[13] W. R. Hamburgen, D. A. Wallach, M. A. Viredaz, L. S. Brakmo, C. A. Waldspurger, J. F. Bartlett, T. Mann, and K. I. Farkas, "Itsy: Stretching the Bounds of Mobile Computing," *IEEE Computers*, April, 2001.
[14] M. Segal, and K. Akeley, "The OpenGL Graphics System : A Specification Version 1.2.1," *Silicon Graphics Inc*. April 1, 1999.
[15] B.-S. Liang, and C.-W. Jen, "Computation-Effective 3-D Graphics Rendering Architecture for Embedded Multimedia System," *IEEE Transactions on Consumer Electronics*, Vol. 46, No. 3, August 2000.

Byeong-Gyu Nam received the B.S. degree in computer engineering from Kyungpook National University, Daegu, Korea, in 1999 and M.S. degree in computer science from Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea, in 2001. He is currently working toward the Ph.D. degree in electrical engineering at KAIST. From 2001 to 2002, he had been a research engineer in parallel system research laboratory, Electronics and Telecommunication Research Institute (ETRI), Daejeon, Korea. His research interests include low-power design and implementation of 3D graphics processors and embedded microprocessors for handheld systems.

Min-wuk Lee received the B.S. degree in electronics engineering from Kyungpook National University, Daegu, Korea, in 2003 and M.S. degree in electrical engineering from Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea, in 2005. Since 2005, he has been an engineering staff in nVidia, Seoul, Korea. His research interests include real-time 3D graphics for handheld systems. He is now working for the graphics library for handheld systems.

Hoi-Jun Yoo (M'95) graduated from the Electronic Department of Seoul National University, Seoul, Korea, in 1983 and received the M.S. and Ph.D degrees in electrical engineering from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, in 1985 and 1988, respectively. His Ph.D. work concerned the fabrication process for GaAs vertical optoelectronic integrated circuits.

From 1988 to 1990, he was with Bell Communications Research, Red Bank, NJ, where he invented the two-dimensional phase-locked VCSEL array, the front-surface-emitting laser, and the high-speed lateral HBT. In 1991, he became Manager of a DRAM design group at Hyundai Electronics and designed a family of fast-1 MDRAMs and synchronous DRAMs, including 256M SDRAM. From 1995 to 1997, he was a faculty member with Kangwon National University. In 1998, he joined the faculty of the Department of Electrical Engineering at KAIST, and currently leads a project team on RAM Processors (RAMP). In 2001, he founded a national research center, System Integration and IP Authoring Research Center (SIPAC), funded by Korean government to promote worldwide IP authoring and its SOC application. Currently he is the Project Manager for SoC in Korea Ministry of Information and Communication. His current interests are SOC design, IP authoring, high-speed and low-power memory circuits and architectures, design of embedded memory logic, optoelectronic integrated circuits, and novel devices and circuits. He is the author of the books *DRAM Design* (Seoul, Korea: Hongleung, 1996; in Korean) and *High Performance DRAM* (Seoul, Korea: Sigma, 1999; in Korean).

Dr. Yoo received the Electronic Industrial Association of Korea Award for his contribution to DRAM technology in 1994 and the Korea Semiconductor Industry Association Award in 2002.