

An 8M Polygons/s 3-D Graphics SoC With Full Hardware Geometric and Rendering Engine for Mobile Applications

Jeonghun Kim, Hanjun Choi, Sungyeal Yoon, Taesik Bang,
Jongchan Park, Chaehyun Jung, and Jason Cong

Abstract—A 3-D graphics SoC, with a multi-layer advanced micro-controller bus architecture (AMBA) system, full pipelined hardware 3-D graphics accelerator, and clock/power management is proposed as a 49-mm² die for 180-nm CMOS technology. This system-on-chip (SoC) minimizes power consumption with a clock/power management unit according to its operation mode and applications. The 3-D graphics accelerator has a full pipelined architecture which improves full 3-D graphics performance to 8M polygons/s and consumes 108 mW at 100 MHz and 1.8 V. The LCD bypass mode and power-down mode consume 4.32 mW and 180 uW, respectively. The 3-D graphics accelerator also supports stereoscopic 3-D function with an alternative left-right drawing method and achieves a 59% improvement in 3DG performance compared to previous work.

Index Terms—3-D graphics (3DG) accelerator, 3-D display, 3DG system-on-chip (SoC), geometric engine, rendering engine.

I. INTRODUCTION

Recently, real-time 3-D graphics (3DG) have been widely employed on mobile devices such as cellular phones, car navigation systems and portable media players. At the heart of these devices are 3DG chips that provide the computing power for the mobile applications [1]–[7]. These chips are classified, according to their architectures, into programmable 3DG accelerators and full hardware ones. Programmable 3DG accelerators are used to increase flexibility by allowing different programs to be used on the chip. However, this results in very complex processor architectures [2]–[5], and these chips generally have high power consumption. The full hardware 3DG accelerators generally consume less power than programmable 3DG accelerators, but the power consumption is still high in spite of a hardware optimization [1], [6].

We propose a 3DG SoC with a fully pipelined hardware geometric and rendering engine and a clock and power management unit (CPMU) which supports several operation modes for various applications. These operation modes can be reconfigurable to reduce power consumption and to meet various needs of mobile applications with a clock gated scheme and a clock divider. To achieve low power consumption in a normal display mode without the 3DG operation, a liquid crystal display (LCD) bypass mode is designed that consumes only 4.32 mW. The bypass mode allows the 3DG SoC to be inserted between a baseband processor and LCD module. The low power consumption of this SoC allows it to be employed into existing mobile devices with minimum changes for 3DG acceleration.

Manuscript received November 20, 2009; revised March 12, 2010; accepted May 03, 2010. Date of publication June 28, 2010; date of current version July 27, 2011. This work was supported in part by the National Science Foundation CC0903541.

J. Kim and J. Cong are with the Computer Science Department, University of California, Los Angeles, CA 90095 USA (e-mail: kimjhun@cs.ucla.edu; cong@cs.ucla.edu).

H. Choi, S. Yoon, T. Bang, J. Park, and C. Jung are with the R&D Center, Nexuschip Co., Seoul 138-160, Korea.

Digital Object Identifier 10.1109/TVLSI.2010.2051568

II. 3DG SoC ARCHITECTURE AND APPLICATIONS

There is a tradeoff between flexibility and power consumption in the design of the SoC. A high flexibility of 3DG requires a high performance processor with VLIW architecture or a SIMD instruction set [2]–[5], but a full hardware engine allows low power consumption because it has optimized function units for 3DG. We choose the fully hardware pipelined 3DG accelerator for low power consumption. To support various applications and utilize the advantages of SoC [3], we integrated the ARM9 reduced instruction set computer (RISC) core, dual SDRAM controller and CPMU. The CPMU supports several operation modes: “normal mode,” “3DG application mode,” “3DG acceleration mode,” and “bypass mode.”

The 3DG SoC consists of three masters and four slaves as shown in Fig. 1. The three masters are ARM9 core, direct memory access (DMA) controller, and 3DG accelerator; the four slaves are a memory controller, internal memory controller, APB peripherals, and SDRAM controller. Intellectual properties (IPs) are integrated into a three-layer AMBA bus system which allows simultaneous access to three buses from masters to slaves. The memory controller supports flash, ROM, SRAM, and SDRAM. There is another SDRAM controller for the 3DG accelerator and the display controller. To avoid collisions among LCD, ARM9, and 3DG and to be able to produce a stable display on LCD, the output arbiters (shown in Fig. 1) support a round-robin and a programmable arbitration.

Mobile devices have various system applications and features according to performance, power consumption, and customer needs. To satisfy these features, the 3DG SoC allows users to select a combination of baseband, application processor, 3DG accelerator, and 3DG SoC at system level. So, the proposed 3DG SoC is configured to the normal operation mode, 3DG application, and 3DG acceleration mode according to the application to minimize power consumption and performance in the SoC point of view. Fig. 2(a) is the normal mode for a low-end application, in which all blocks operate with a maximum performance. In this application, the ARM9 RISC is used as a multimedia application which is a low-end 2-D video/image processing and a 3DG application program interface (API) at the maximum frequency in the normal mode, and the 3DG accelerator is used for 3DG acceleration.

In high-end application, the SoC can be configured to the 3DG application or 3DG acceleration mode according to where the 3DG API is located. The high-end application supports a high bandwidth video and image processing, a digital multimedia broadcasting (DMB) as well as a 3DG acceleration. Fig. 2(b) shows the system architecture which operates the 3DG application mode in the case of a high-end application. The 3DG application mode operates a 3DG hardware engine and a 3DG API in ARM9 at a minimum frequency to reduce the load of the application processor. This mode reduces 36% of power consumption compared to the normal mode. Alternatively, the 3DG acceleration mode operates only the 3DG hardware engine, moving the 3DG API into an external application processor and using clock-gating to reduce the power consumption on the ARM9 and the unused hardware blocks. This mode reduces 21% of power consumption compared to the 3DG application mode. In both applications, the 3DG SoC consumes less than 4.3 mW in a LCD bypass mode, when the entire block goes to the power-down mode.

Even though some mobile systems have a 3DG hardware engine, we need a 3DG API and a library to optimize the 3DG accelerator in order to use real 3DG acceleration. For that we support OpenGL ES 1.1+Ext, JSR 239, 3 D3M API, and a 3DG library. OpenGL ES is an API for using 2-D and 3-D graphics on a mobile application and an embedded system. The API provides a low-level interface between

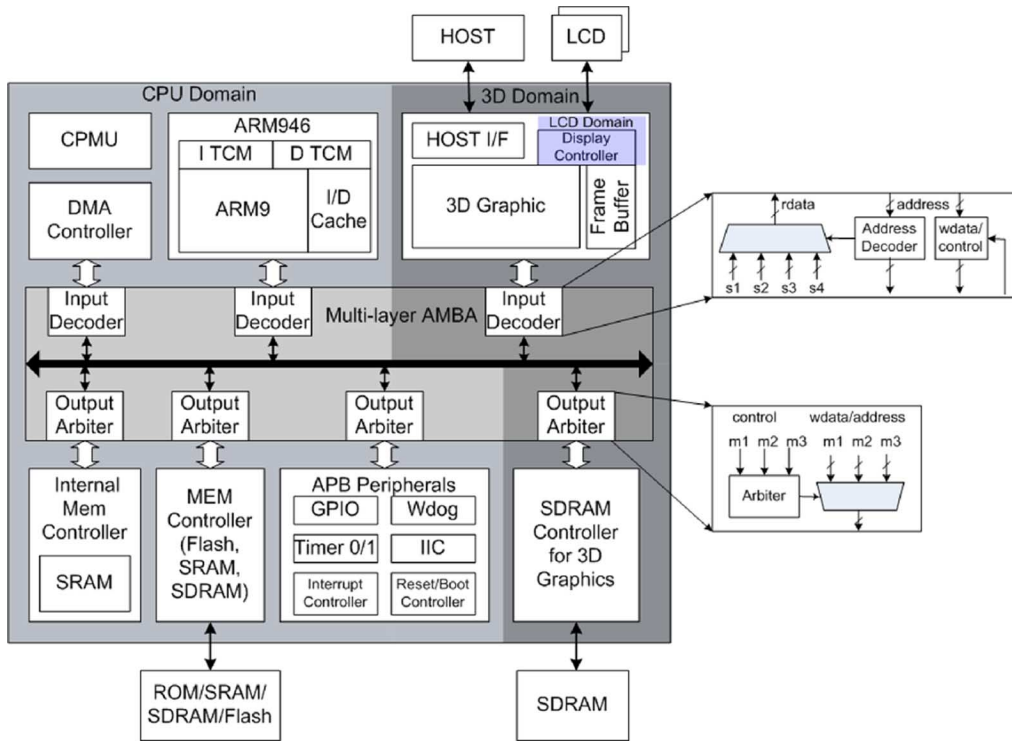


Fig. 1. Block diagram of the proposed 3DG SoC.

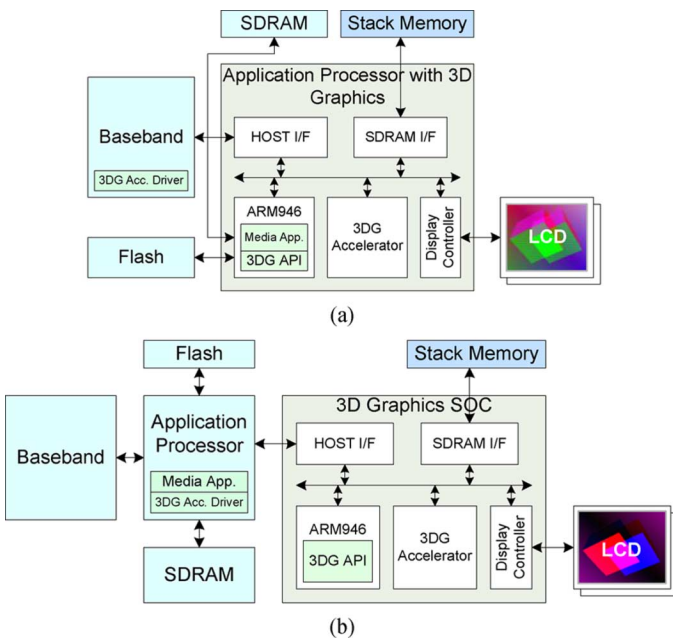


Fig. 2. System architecture for mobile applications: (a) normal mode for low-end application and (b) 3DG application mode for high-end applications.

software/hardware 3DG accelerator and software applications. These standard 3DG APIs allow mobile and embedded applications to support 3DG games and various 3-D graphics functions. Even though OpenGL ES is based on the 3-D graphics pipeline, each function can be used in the full 3DG hardware engine or the combination of hardware and software routines. This flexibility gives users the advantage of easily programming 3DG applications using the hardware engine functions of OpenGL ES. OpenGL ES includes an embedded graphics library

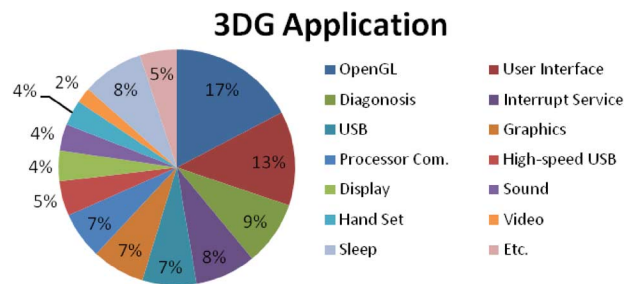


Fig. 3. Task percentages of CPU on a 3DG application.

(EGL), it allows users to program 3DG applications regardless of platform and OS. Users can also use an open 3DG library or a third-party library. Fig. 3 shows the task percentages of a CPU on a 3DG application with 3-D contents when the 3DG SoC is used as the application processor. OpenGL ES API occupies 17.3% of runtime in the normal mode for low-end applications. In this case, we can reduce power consumption by adjusting the clock frequency of CPMU on a normal operation. An example of another application is the following: once an application processor without a 3DG accelerator is used, we can easily add a 3DG function using the 3DG SoC which operates in the 3DG application mode with OpenGL ES API and 3DG accelerator for high-end application. OpenGL ES API can be moved to 3DG SoC, so we can reduce power consumption with the bypass mode of 3DG SoC once the application processor performs a normal operation—such as phone, camera, or some other normal operation.

III. PROPOSED 3-D GRAPHICS ACCELERATOR

The three functional blocks of the 3DG accelerator—the geometric engine (GE), the rendering engine (RE), and the display controller—are shown in Fig. 4. The polygon-based operation block performs culling, clipping check and viewport transformation. The GE adopts a five-

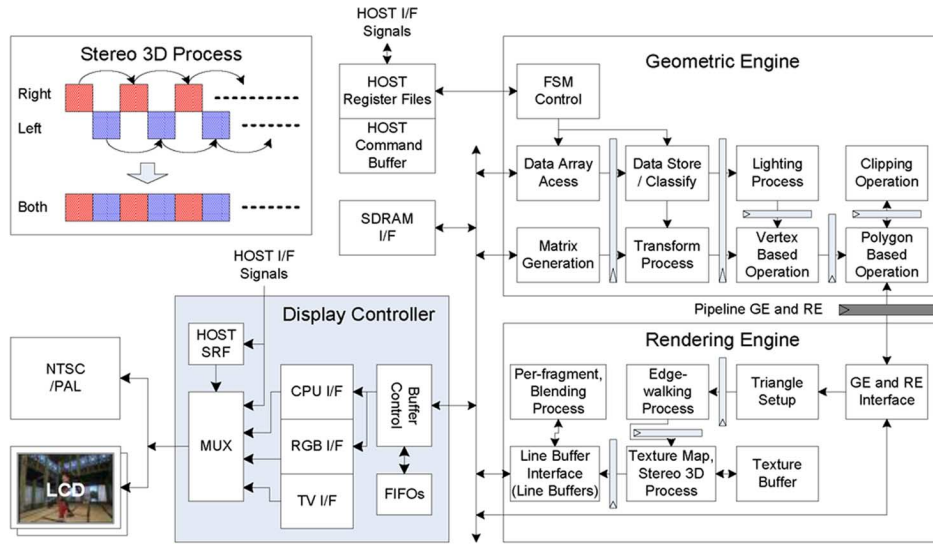


Fig. 4. Diagram of the 3-D graphics architecture with geometric and rendering engine.

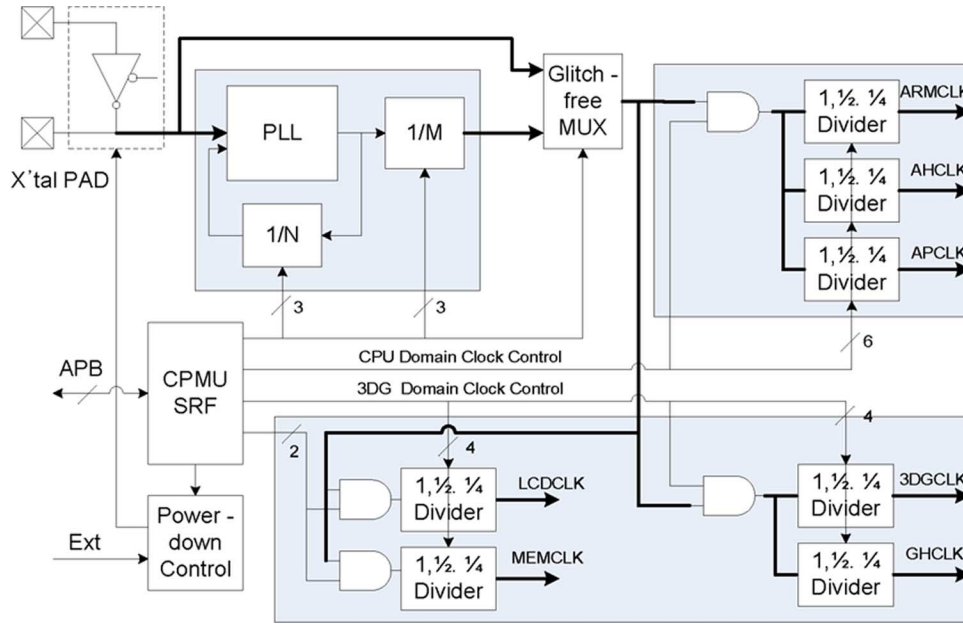


Fig. 5. Clock and power management unit.

stage multi-cycled pipeline with an eight-cycle stage latency, which has model-view transform, lighting calculation, projection transform, clip test and perspective division transform stages. The finite-state machine (FSM) block controls the overall operation of the pipeline and an index cache for the GE.

The 3DG pipeline consists of a GE and an RE processing. The GE receives three vertices from a host processor, and then makes a triangle. After that, the GE continuously transforms another triangle, while considering the direction of a visual point, and performs the lighting process to make a color and the clipping process. Finally, the polygon-based process is performed. The maximum performance of geometry transformation is 37.5 Mvertices/s, if no other operations are performed. We achieved the best performance in a 3DG game using the full hardware clipping process without a decrease in speed. The most important process for improving performance in the 3DG accelerator is the clipping transaction. Even though the 3DG accelerator possesses

a high 3-D transaction speed and pixel fill rate, it is unrealistic to expect high performance in real contents without accelerating the clipping process. For that, the clipping operation block checks the boundary line, and then performs the rendering process of only internal value even if the object exists on an external area in terms of viewing contents—such as a 3-D background passing by in a 3-D racing game. The GE engine achieves a high transaction speed using fully hardwired clipping check and operation block while enjoying a game. Even real contents require a high clipping transaction.

In the RE, the engine writes triangles to a frame buffer. The RE consists of the GE/RE interface, triangle setup, edge-walking process, texture map/stereoscopic 3-D process and per-fragment/blending process blocks. The RE has triangle setup, texturing, fog, and per-fragment operation pipeline stages with the blocks working in coordination. Generally, it takes two clocks to write a triangle in the frame buffer because the 3DG calculation has to perform a depth

TABLE I
TECHNOLOGY AND SPECIFICATION COMPARISON

Section	This work	ISSCC04[1]	JSSCC06[2]	JSSCC06[3]	JSSCC07[4]	JSSCC08[5]	VLSI-DAT[6]
Technology	0.18um 6M	0.18um 5M	0.18um 6M	0.13um 7M	0.18um 4M	0.13um 7M	0.18um 6M
Frequency	100MHz	75MHz	200MHz	166MHz	100MHz	100MHz	139MHz
Power Consumption	108mW	109.5mW	155mW	407mW	157mV	195mW	472.6mW
Power Supply	1.8V	1.2V	1.8V	1.2V	1.875V	1.2V	1.8V
Pixel Rate	1.5	2	1	2	na	1	2
3D Performance	8	4.7	3.6	7.55	4.4	9.1	8.69

comparison and read/write operation between GE and the frame buffer. It means that the pixel rate is 0.5 pixels/clock. The proposed 3DG accelerator achieves 1.5 pixels/clock using the pipeline structure. The GE/RE interface block controls the GE and RE pipeline stages and data exchange between GE and RE. The per-fragment block and the texture map block have a line buffer and texture buffer whose size is 131 Kbytes to improve a data bandwidth. The maximum LCD size is 1024×1024 pixels and a single texture is 256×128 pixels. We proposed an alternative left-right drawing method (ADM) for the stereoscopic 3-D processing. The conventional method has a performance inefficiency because it continuously fills every image in each left and right image line, whether or not this needs to be done. The proposed ADM improves 50% of the fill rate in comparison with the conventional one. The stereoscopic 3-D block supports red/blue, barrier, and glassed types. The 3DG SoC supports OpenGL ES 1.1, JSR-239 and D3DM APIs including seven primitives, eight light sources and a stereoscopic 3-D display. The 3DG performance is 8 Mpolygons/s at 100 MHz. The display controller supports the bypass mode which passes the video image through the 3DG SoC and directly to the LCD; it supports other modes which send the video image from 3DG accelerator to the LCD through a first-in-first-out (FIFO) and RGB/CPU/TV interfaces.

Fig. 5 shows the block diagram of CPMU. The CPMU's role is to manage power consumption depending on the needs of the application. The 3DG SoC has three power domains—CPU, 3DG, and LCD, and six operation modes—the full, 3DG application, 3DG engine, SDRAM refresh, LCD, and power-down mode. The PLL covers frequencies ranging from 25 to 200 MHz with 25 MHz steps at 25 MHz of input clock frequency. The phase-locked loop (PLL) locking time is less than 600 μ s and it consumes 1.8 mW. The CPU and 3DG domain controller supports various clock combinations for each block through software according to its purpose and target performance. The normal mode in the 3DG SoC operates as a low-end application processor. The ARM9 operates with 3DG API at 25 MHz, and the 3DG accelerator operates at 100 MHz in the 3DG application mode. To reduce power consumption, the ARM9 can be turned off and 3DG turned on only for the hardware 3DG in the 3DG acceleration mode. The power consumption of the power-down mode is under 180 μ W.

IV. EXPERIMENTAL RESULTS

Fig. 6 shows the experimental and evaluation board for 3DG SoC. The evaluation board consists of the 3DG SoC, DDR memory, flash memory, LCD display, etc. We used Futuremark's benchmarks to evaluate our 3DG SoC and to compare the previous works [1]–[6]. These contents are organized in the multi-texture which uses mipmaps (mip is a Latin abbreviation for *much in a small space*) and is responsible for using a 3-D graphics transaction. The first image at the bottom of Fig. 6 is the result of depth testing. The second

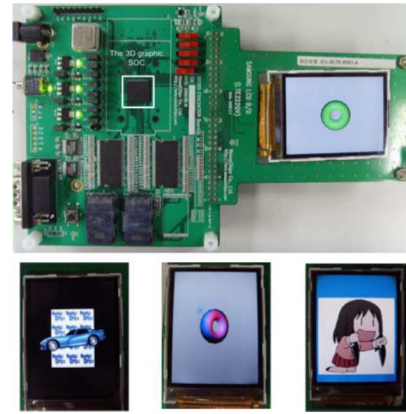


Fig. 6. Board and images of depth testing, lighting, and texture mapping (bottom images, left to right).

image is the result of the lighting test. The third image shows the result of texture mapping.

Table I is the comparison of technology, operation frequency, and performance with the previous works [1]–[6]. The JSSCC06 and JSSCC08 were designed for 0.13- μ m CMOS technology. Even though these works have an advantage of a more advanced technology, the proposed SoC achieved a better performance and a lower power than these works [3], [5].

The power consumption of each operation mode is shown in Fig. 7(a). We use Samurai 3-D movie to compare with the previous works [1]–[6]. We measure the power consumption and performance with the Samurai movie on full, 3DG application and 3DG acceleration mode as shown in Fig. 7(a). In the normal mode, the 3DG accelerator occupied 47% of the total power consumption, ARM946 and peripherals are 41% and 8%, respectively. In the 3DG application mode, RM946 consumes 17% of the total power, so the power consumption is 36% less than the normal mode when the SoC operates as an application processor. In the 3DG acceleration mode, we also reduce to 21% less than the total power of the 3DG application mode. These results allow users to choose the best mode according to system architecture and its application. The 3DG application and acceleration mode of SoC can be specifically selected by its application to enable power reduction.

Fig. 7(b) shows a comparison of the performance of the 3DG with the previous works [1]–[6]. The 3DG performance is measured, including transformation, clip check/processing, lighting, and texture blending for rendering. The performance index of the 3DG drawing speed/power consumption is used to compare the tradeoff between power consumption and performance. The 3DG accelerator improves the performance by 59% compared to the best work [5]. The work in [2] achieved 50

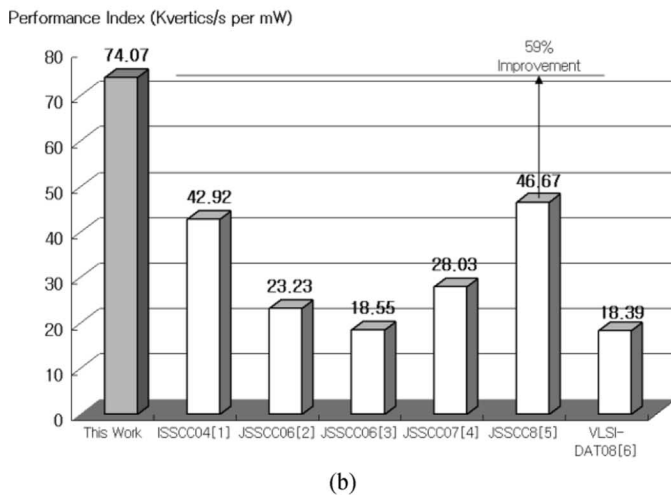
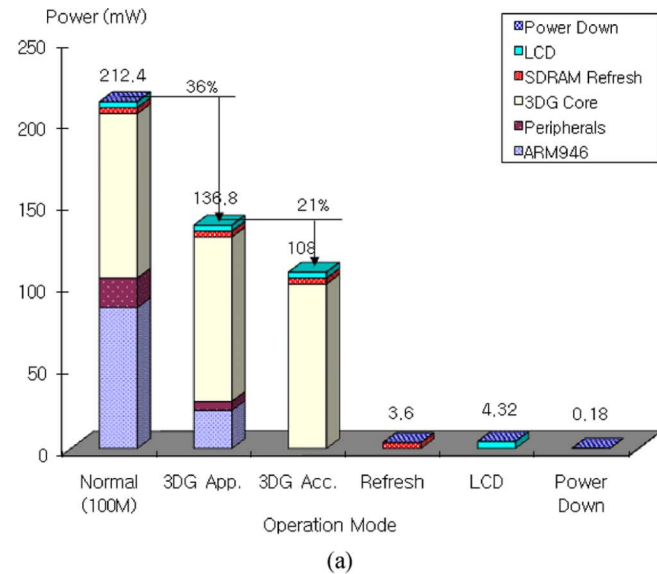


Fig. 7. Power consumption and performance comparison: (a) power consumption for each mode and (b) performance comparison of 3DG with previous works.

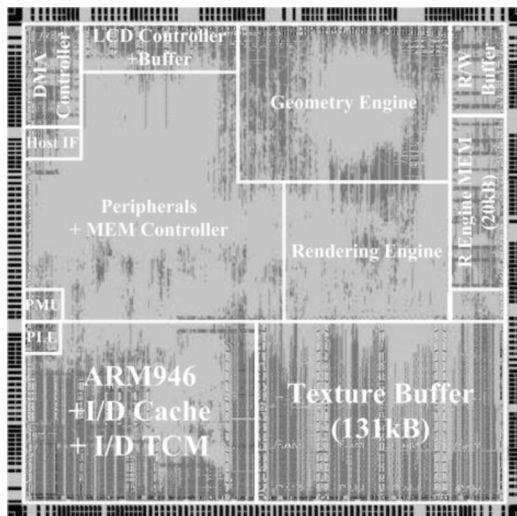


Fig. 8. Micrograph of 3-D graphics SoC in 0.18- μ m CMOS.

Mvertices/s, but it was only for the geometry performance. Full 3DG performance is 3.6 Mvertices/s and 155 mW [2]. We achieved 8 Mvertices/s and 108 mW in full 3DG performance.

TABLE II
CHIP SPECIFICATION AND CHARACTERISTICS

Section	Description	
Technology	Fujitsu 0.180um 1P6M CMOS	
Die Size	Chip: 49 mm ² (7mm x 7mm), 3DG Accelerator: 14.4 mm ²	
Power Supply	1.8V- Core, 2.8V-I/O	
Gate Counts	3.5M gates (including 3DG Accelerator (1M gates) and SRAM)	
Operation Frequency	Input:25MHz, output:25~200 MHz	
Power Consumption	Normal mode: 212.4mW at 100Mhz 3DG Application mode:136.8mW 3DG Acceleration mode:108mW Power-down mode : 180uW	
Performance	MCU	32 bit ARM946 with Cache/TCM
	Geometry	24 Mvertices/s
	Rendering	150 Mpixels/s, 150 Mtexels/s
	Full 3DG	8 Mpolygons/s
	Pixel fill rate	1.5 Pixels/clock
Features	Standard	Open GL-ES 1.1+Ext, JSR 239 API, 3D3M
	3DG Stereo Display	Red/Blue, Barrier and Glasses Types
	3DG Accelerator	Full-hardware Geometric/Rendering Engine Stereoscopic 3DG with 3 Types Support 7 Primitives, 8 Light Sources Flat/Gouraud Shading, Culling, Clipping Support up to 1024x1024 LCD

V. CONCLUSION

A 3DG SoC is designed for mobile applications. It integrates a fully hardwired 3DG accelerator, low power management block, multimedia processor and LCD controller. The SoC is fabricated in a 0.18 um 1P6M CMOS process. The chip size of the 3DG SoC is 49 mm², which includes the 3DG accelerator, 131 kB texture buffer, and the ARM9 which has 16 K I/D-cache and 8 kB I/D TCM. The dual SDRAM controllers allow the ARM9 and 3DG to access each external SDRAM while avoiding data collisions. The SoC has reconfigurable operation modes to support various mobile applications with minimum power consumption. The SoC achieves 8 Mpolygons/s full 3DG, 24 Mvertices/s geometry performance and 1.5 pixels/clk. Table II summarizes the chip specification and features. Fig. 8 shows the chip micrograph of the 3DG SoC.

REFERENCES

- [1] M. Imai, T. Nagasaki, J. Sakamoto, H. Takeuchi, H. Nagano, S. Iwasaki, M. Hatakenaka, J. Fujita, K. Keino, T. Motomura, and T. Ueda, "A 109.5 mW 1.2 V 600 M texels/s 3DG engine," in *ISSCC Dig. Tech. Papers*, Feb. 2004, pp. 332–333.
- [2] J. Sohn, J.-H. Woo, M.-W. Lee, H.-J. Kim, R. Woo, and H.-J. Yoo, "A 155-mW 50-Mvertices/s graphics processor with fixed-point programmable vertex shader for mobile applications," *IEEE J. Solid-State Circuits*, vol. 41, no. 5, pp. 1081–1091, May 2006.
- [3] D. Kim, K. Chung, C.-H. Yu, C.-H. Kim, I. Lee, J. Bae, Y.-J. Kim, J.-H. Park, S. Kim, Y.-H. Park, N.-H. Seong, J.-A. Lee, J. Park, S. Oh, S.-W. Jeong, and L.-S. Kim, "An SoC with 1.3 Gtexels/s 3-D graphics full pipeline for consumer applications," *IEEE J. Solid-State Circuits*, vol. 41, no. 1, pp. 71–84, Jan. 2006.
- [4] C. H. Yu, K. Chung, D. Kim, and L.-S. Kim, "An energy-efficient mobile vertex processor with multithread expanded VLIW architecture and vertex caches," *IEEE J. Solid-State Circuits*, vol. 42, no. 10, pp. 2257–2268, Oct. 2007.

- [5] J. H. Woo, J.-H. Sohn, H. Kim, and H.-J. Yoo, "A 195 mW-152 mW mobile multimedia SoC with fully programmable 3-D graphics and MPEG4-H.264-JPEG," *IEEE J. Solid-State Circuits*, vol. 43, no. 9, pp. 2047–2056, Sep. 2008.
- [6] R. T. Gu, W.-S. Huang, C.-C. Wang, W.-C. Shiu, T.-Y. Ho, C.-H. Tsai, T.-C. Tien, D.-J. Zhang-Jian, S.-Y. Chiu, I.-J. Huang, Y.-N. Chang, S.-F. Hsiao, J.-H. Hong, C.-N. Lee, and M.-C. Chiang, "An 8.69 Mvertices/s 278 Mpixels/s tile-based 3D graphics full pipeline with embedded performance counting module, real-time bus tracer and protocol checker for consumer electronics," in *Proc. IEEE Symp. VLSI-DAT*, Hsinchu, Taiwan, Apr. 2008, pp. 59–62.
- [7] J. H. Woo, J.-H. Sohn, H. Kim, J. Jeong, E. Jeong, S. Lee, and H.-J. Yoo, "A 152 mW mobile multimedia SoC with fully programmable 3D graphics and MPEG4-H.264-JPEG," in *IEEE Symp. VLSI Circuit Dig. Tech. Papers*, Jun. 2007, pp. 220–221.

Hardware Implementation of Rayleigh and Ricean Variate Generators

Amirhossein Alimohammad, Saeed Fouladi Fard, and
Bruce F. Cockburn

Abstract—Compact and fast implementations of digital Rayleigh and Ricean variate generators are presented. Polynomial curve fitting is utilized along with a combination of logarithmic and uniform domain segmentation to provide accuracy, compactness and fast variate generation. A typical instantiation of the proposed Rayleigh generator occupies 124 (<1%) of the configurable slices, two dedicated multipliers (<1%), and one on-chip block memory (<1%) of a Xilinx Virtex-5 field-programmable gate array (FPGA) and operates at 317 MHz, generating 317 million Rayleigh variates per second. The Ricean variate generator implementation on the same device utilizes 366 (<1%) of the logical slices, three on-chip block memories (<1%), and 11 (2.8%) of the dedicated multipliers. The application of the Rayleigh and Ricean variate generators is demonstrated in a FPGA-based bit error rate simulator that measures at hardware speeds the symbol error rate performance of a typical wireless communication system over Rayleigh and Ricean fading channels.

Index Terms—Fading simulator, fading variate generator, Rayleigh fading, Ricean fading.

I. INTRODUCTION

The Gaussian, Rayleigh, and Ricean distributions have been applied to model and simulate a variety of different scientific and engineering systems. An especially important application of these distributions is to model wireless fading channels. The classical model of a communication channel is the additive white Gaussian noise (AWGN) channel,

Manuscript received October 18, 2009; revised February 27, 2010; accepted May 09, 2010. Date of publication June 28, 2010; date of current version July 27, 2011. The work of B. F. Cockburn was supported in part by the Natural Sciences and Research Council of Canada (NSERC) and by the Alberta Informatics Circle of Research Excellence (iCORE).

A. Alimohammad and S. Fouladi Fard are with Ukalta Engineering, 4344 Enterprise Square, 10230 Jasper Avenue, Edmonton, AB T5J 4P6, Canada (e-mail: amir@ukalta.com; saeed@ukalta.com).

B. F. Cockburn is with the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB T6G 2V4, Canada (e-mail: cockburn@ualberta.ca).

Digital Object Identifier 10.1109/TVLSI.2010.2051465

where the transmitted signal $s(t)$ is corrupted by the addition of white Gaussian noise $n(t)$ thereby producing a received signal $y(t) = s(t) + n(t)$ [1]. In a more accurate model of wireless channels, the received complex envelope is expressed as $y(t) = g(t)s(t) + n(t)$, where the fading gain $g(t)$ is a complex Gaussian random variable with independent quadrature components. If this fading process has a zero (nonzero) mean then the envelope $|g(t)|$ of the gain has the Rayleigh (Ricean) distribution [2].

The radio channel is usually the key factor that limits the performance of a wireless communication system. System performance is commonly characterized through the symbol error rate (SER) versus signal-to-noise ratio (SNR) relationship and this is typically measured experimentally using Monte Carlo (MC) simulations on workstations. Wireless communication systems are increasingly complex and the number of possible operating modes that must be verified has increased dramatically. As the number of possible operating modes increases (e.g., more than 300 modulation and coding schemes are present in the IEEE 802.11n standard), the bit-true fixed-point MC simulation times become a bottleneck to timely product design and verification.

Hardware-based simulation of digital communication systems offers significant speedups compared to software simulations, with no significant loss in accuracy [3], [4]. Recently, several hardware implementations of Gaussian variate generators have been proposed (see [5], [6], and their references). However, hardware implementations of other important distributions, such as the Rayleigh and Ricean distributions, have received far less attention [7], [8]. This brief extends our earlier work on designing Gaussian variate generators (GVGs) [6]. We now present high-throughput and compact Rayleigh and Ricean variate generators that are suitable for implementation on a field-programmable gate array (FPGA). We utilize the Box–Muller (BM) algorithm [9] to efficiently implement a Rayleigh variate generator. This generator is then enhanced to generate variates with the Ricean distribution. The Ricean variate generator can in turn be used to generate variates for two other important distributions: the Gamma distribution and the Chi-squared distribution with two degrees-of-freedom. The Gamma and Chi-squared distributions have been used to model interference in wireless communication systems [10].

The sequel is organized as follows. Section II presents our FPGA implementation of the Rayleigh variate generator. Section III presents the new Ricean variate generator. The implementation costs and simulation results are presented in Section IV. Concluding remarks appear in Section V.

II. RAYLEIGH VARIATE GENERATOR

Let n_i and n_q be two independent normally-distributed variates with zero means and equal variance σ^2 . The variable defining the magnitude $r = \sqrt{n_i^2 + n_q^2}$ has a Rayleigh distribution with mean $\sigma\sqrt{\pi/2}$ and variance $(4 - \pi)\sigma^2/2$ [11]. To implement a Rayleigh variate generator, instead of generating two independent Gaussian variables, n_i and n_q , and then computing the magnitude of the complex Gaussian-distributed variate $n = n_i + j n_q$, where $j^2 = -1$, we use the well-known BM algorithm. According to this algorithm, if u_1 and u_2 are two independent uniformly-distributed pseudorandom numbers (PNs) in the interval (0, 1) and $f(u_1) = \sqrt{-2 \ln(u_1)}$, then $n_1 = f(u_1) \times \sin(2\pi u_2)$ and $n_2 = f(u_1) \times \cos(2\pi u_2)$ are two independent variates from a zero-mean, unit-variance Gaussian distribution $\mathcal{N}(0, 1)$. Therefore the variate $r = \sqrt{n_1^2 + n_2^2} = f(u_1)$ follows the Rayleigh distribution.