Optimization of Portable System Architecture for Real-Time 3D Graphics

Ju-ho Sohn, Ramchan Woo and Hoi-Jun Yoo

Semiconductor System Laboratory, Department of Electrical Engineering and Computer Science
Korea Advanced Institute of Science and Technology, 373-1, Yusong-gu, Daejon 305-701, Korea
(E-mail) sohnjuho@eeinfo.kaist.ac.kr

ABSTRACT

The optimal architecture of personal digital assistants (PDA) system for real-time 3D graphics was analyzed by simulating the 3D applications on the various Advanced RISC Machines (ARM) processor platforms. Simulation results show that for 256x256 screen resolution, even the performance of 200MHz StrongARM with 160MHz floating point unit (FPU) shows only 1.78 % of the requirement of full 3D pipeline. To realize the real-time 3D graphics on PDA, the optimal architecture must contain hardware acceleration engine with embedded DRAM as the rendering stage. In this architecture, MAC-enhanced ARM9 without FPU that is used as a host processor can provide the necessary geometry operations and we verified this architecture by the implementation of a PDA chip.

1. INTRODUCTION

The multimedia applications become increasingly important for portable digital devices such as PDA's. And one of them is the real-time 3D computer graphics such as 3D fax, 3D advertisement, 3D game and 3D navigation [1]. The real-time 3D graphics are still challenging for today's PC platform [2] because their operations require high computing complexity and memory bandwidth. Therefore, the 3D graphics on PDA is much more difficult because the processing power is also limited by battery lifetime. However, the performance requirements of PDA can be lower than those of PC because of its limited screen resolution.

Recently, several researches have tried to provide more multimedia functionalities with PDA by increasing the performance of its host processor [3]. However, the real-time 3D graphics cannot be easily realized in the conventional PDA architecture, in which the host processor performs all calculations with limited memory bandwidth. To break the performance barrier while achieving the low power consumption, hardware acceleration is necessary [4].

In this paper, we optimize the PDA architecture for the real-time 3D graphics by simulating the 3D applications on various host processors for the case with and without FPU. Then, we propose an optimal architecture that consists of an ARM9 host processor without FPU and hardware rendering engine with embedded DRAM. The proposed architecture was verified by its implementation as a PDA chip [1][5].

2. SIMULATION ENVIRONMENT

The simulation environment consists of three components; 1) graphics library, 2) target platforms and 3) applications as shown in Fig. 1. The first component, 3D graphics library based on the OpenGL specification, is implemented into 3D pipeline as shown in Fig. 2. The library consists of geometry and rendering stages. The geometry stage processes polygon data from input models by performing operations such as transformation, lighting, and perspective projection. Especially, the light effect is calculated by blending ambient, specular, diffuse, and emission component originated by each light source. Using the data calculated by the geometry stage, the rendering stage draws pixels to the screen buffer. It first sets up triangles in 2D screen from 3D geometry data and performs interpolation to calculate edge coordinates of each triangle. Then it renders each pixel by shading and texture mapping. The rendering stage also performs alpha-blending for translucent objects and z-comparison for hidden surface removal. The operations of rendering stage are memory-intensive because it needs frequent accesses

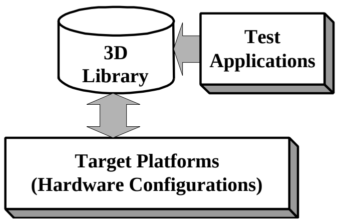

Fig. 1: Simulation Environment

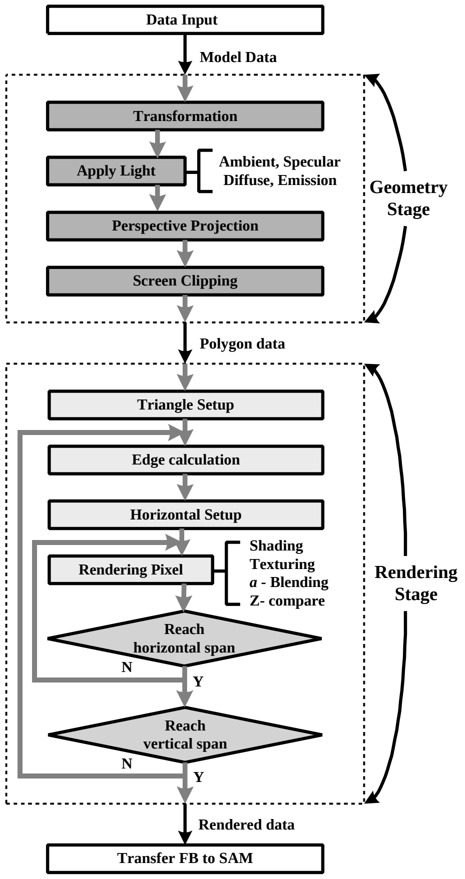

Fig. 2: Pipeline of 3D Computer Graphics

to frame buffer, z buffer and texture buffer.

The second component is the target platforms for simulation. The ARM processor family that has the reduced instruction set computer (RISC) architecture is widely used as the host processor of mobile systems recently because of its high MIPS/W [6][7]. 3 different ARM processor platforms were used for simulation. The first one is ARM7. It has 3-stage pipeline with unified memory port, which limits the memory access. The other one, ARM9, is an improved core with the same instruction set architecture (ISA) as ARM7. ARM9 has 5-stage pipeline with separate memory ports for data and instruction. Also, the balanced pipeline of ARM9 gives higher performance than ARM7. And the other platform is a StrongARM that has 5-stage pipeline with the modified-Harvard architecture with separate instruction and data caches. StrongARM is different from ARM9 in that it has a dedicated branch adder that operates in parallel with the register read stage so that its penalty of taken-branch is reduced by one-cycle. It also has 12-bit iterative multiplier for 32-bit multiplications, while other ARM cores have 8-bit multiplier. And FPU is attached through the coprocessor interface, by which the consecutive execution stalls the integer datapath of ARM processors.

The third component of the simulation environment is the benchmark applications. The specifications of target applications have 6,800 average polygons, 16 average pixels per polygon, and 24-bit true color on 256 x 256 screen resolution. Fig. 3 shows the test scenes captured from animated image sequences. The left scene was used mainly for analyzing texturing and the right for lighting. The applications use Gouraud shading and point sampling texturing with MIP-map. With these applications, using cycle-accurate simulator in ARM software development kit [8], we simulated their performance for each configuration. With ARM profiler and tracer, the portion of the time spent on each sequence in 3D pipeline was calculated and memory transactions were traced. The ARM7500 FE [9] was referenced to calculate the floating-point instruction calls. SRAM interface was used for memory modules.

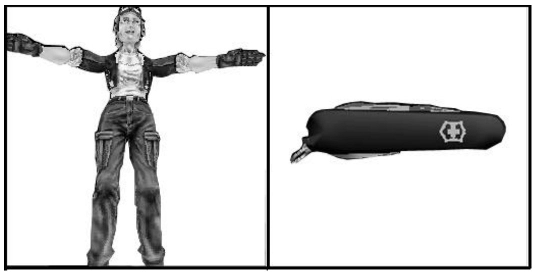

Fig. 3: Test Scenes

3. SIMULATION RESULTS

While analyzing the performance of 3D pipeline, we gathered the polygon calculation rate and pixel fill rate of geometry stage and rendering stage, respectively. The performance of geometry stage was measured with or without lighting, and rendering stage was performed with or without texturing, respectively. This is because the lighting and the texturing requires more processing complexities than the rest of operations

Fig. 4 and Fig. 5 show the polygon calculation rate of geometry stage. When lighting is omitted, its polygon calculation rate was increased by 2.74, 2.95 and 3.08 times for ARM7, ARM9 and StrongARM, respectively. When using FPU, the performance improvement is not as good as expected because the integer datapath is stalled by consecutive execution of floating point operations in the graphic library. In the architecture of 200MHz ARM9 with 80MHz FPU, the improvement was just 42.8% compared with that of ARM9 without FPU.

Fig. 6 and Fig. 7 show the pixel fill rate of rendering stage. Rendering stage also shows the higher performance without texturing than with texturing and it doesn’t show the dramatic improvement when using FPU. While geometry stage shows twice higher performance in the 5-stage pipeline processor such as ARM9 than that in the 3-

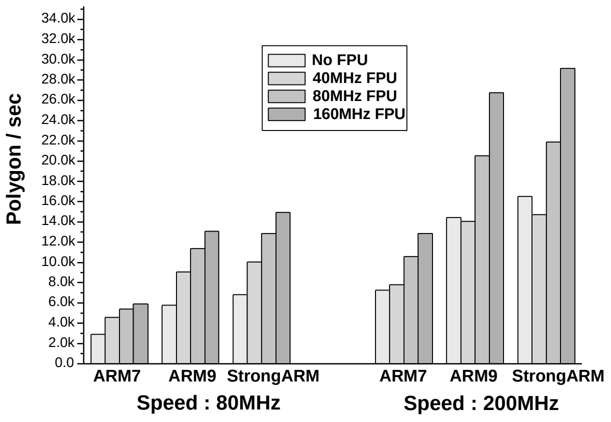

Fig. 4: Geometry performance without lighting

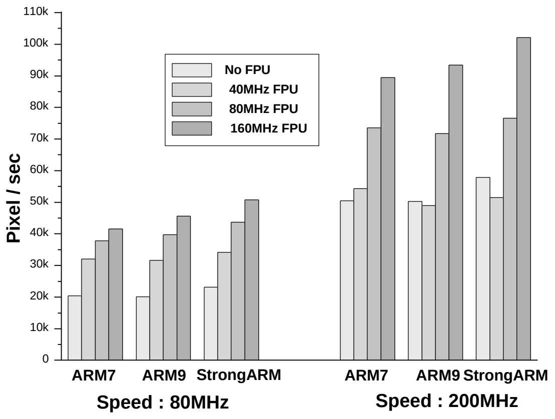

Fig. 6: Rendering performance without texturing

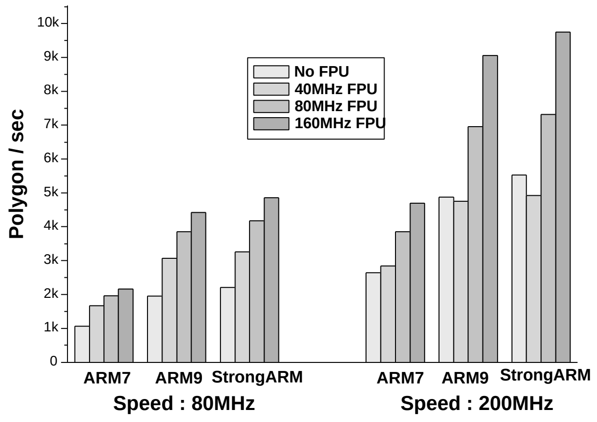

Fig. 5: Geometry performance with lighting

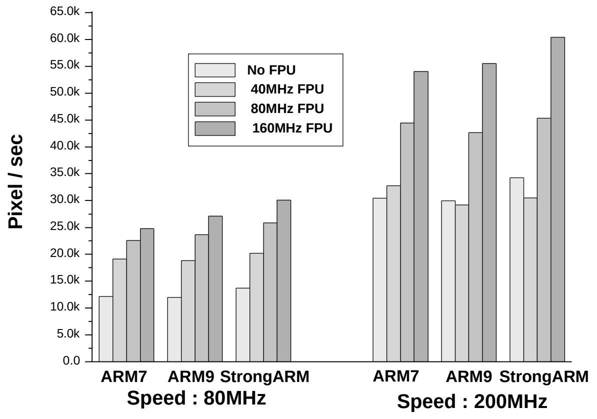

Fig. 7: Rendering performance with texturing

stage pipeline processor at the same clock, rendering stage shows almost the same performance in both of ARM7 and ARM9. It is because the operations of geometry stage are computation-oriented while those of the rendering stage are memory-oriented so that the pipeline efficiency has more influences on the geometry stage.

Fig. 8 shows clock cycle usage of each sequence of 3D pipeline normalized to ARM7 cycle time without FPU. The most time consuming part of the geometry stage was the calculation of specular lighting due to the distance calculation between light source and object as well as normal vector of the object. To calculate specular lighting, floating-point divisions and square root operations are required. For the rendering stage, texturing consumed most of time, because it uses logarithmic and exponential operations to find level of detail (LOD) value, and it frequently accesses texture memory.

Fig. 9 shows instruction pattern of the geometry and rendering stage with SRAM interface as memory system. The rendering stage has more memory access cycles than geometry stage in all of the processor types. It means that the memory bandwidth is more critical than computing complexity in the rendering stage. The portion of load/store cycles is cut in half for ARM9 and StrongARM that use Harvard architecture. It is because in Harvard

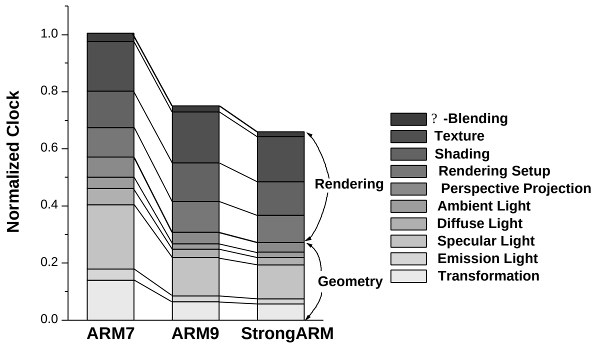

Fig. 8: Comparison of 3D Pipeline Usage

architecture, the instruction and data can be fetched simultaneously.

Fig. 10 compares the required performances of 3D graphics and performances of different PDA architectures. For 256x256 screen resolution, pixel fill rate of 2M pixel/sec is required for 15-frame/sec and average depth complexity of two. Only rendering operations on existing ARM-based host processor is still far below the requirements by 1/30. Even for cell-phone, the requirements are also far higher. The performance of 200MHz StrongARM with 160MHz FPU, which shows the best performance in our simulation, shows only 1.78 % of the requirements of full 3D pipeline. It shows only

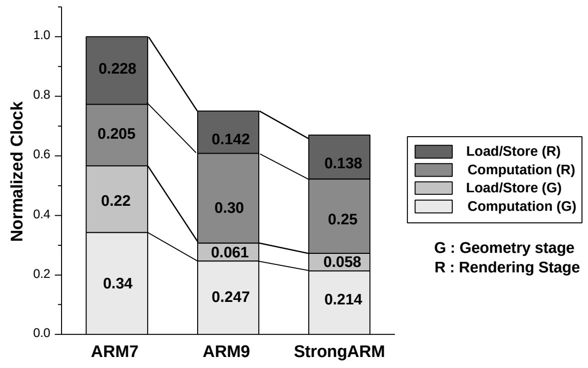

Fig. 9: instruction Pattern of 3D Pipeline

35.5k pixels/s. However, the dedicated rendering accelerator, which was devised for PDA [5], shows much higher pixel fill rate than required. And the host processor performs only geometry operations in this architecture. Based on the simulation results of average 16 pixels per one polygon, the required performance of the geometry stage is 125k polygon/sec for multimedia PDA. Even if the performance of 200MHz ARM9 without FPU shows 15k polygons/sec, it can be ten times improved by the optimization of floating point 3D graphics library into the integer one [10]. An additional multiply and accumulate (MAC) unit can further reduce cycle times of geometry operations, which are mostly matrix operations.

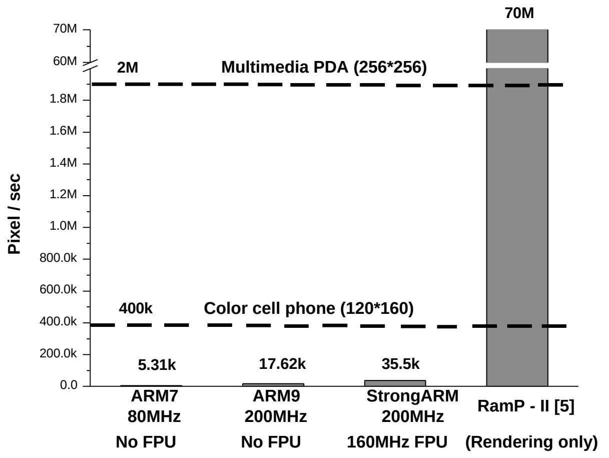

Fig. 10: Performance and its requirements on different mobile systems

4. CONCLUSION

The 3D applications were simulated and analyzed on various host processor platforms to optimize the PDA architecture for real-time 3D graphics. In the 200MHz host CPU with 80MHz FPU, only 40% performance improvement in the full 3D pipeline is obtained sacrificing the area and design complexity for the implementation of FPU. For 256x256 screen resolution targeted for PDA, even the performance of 200MHz StrongARM with 160MHz FPU shows only 1.78 % of the requirement of full 3D pipeline because of the insufficient processing power and limited memory bandwidth.

To realize the real-time 3D graphics on PDA, the optimal architecture must contain hardware acceleration engine with embedded DRAM as the rendering stage. In this architecture, MAC-enhanced ARM9 without FPU that is used as a host processor can provide the necessary geometry operations. In order to verify the architecture, a PDA chip was implemented [1], and Fig. 11 shows the microphotograph of the PDA chip.

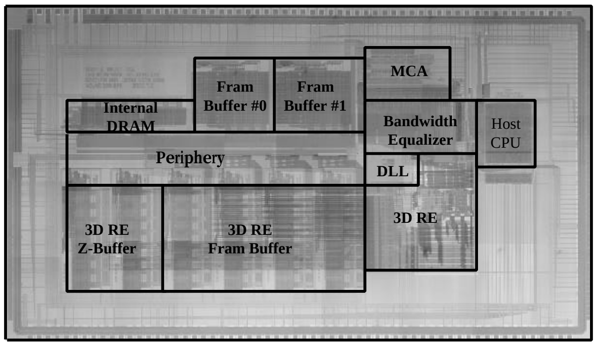

Fig. 11: Microphotograph of PDA chip

5. REFERENCES

[1] Chi-Weon Yoon, et al, “A 80/20MHz 160mW Multimedia Processor integrated with Embedded DRAM MPEG-4 Accelerator 3D Rendering Engine for Mobile Applications,” ISSCC, Dig of Tech. Papers, pp.142-143, 2001-10-23
[2] JayTorborg, et al, “Talisman: Commodity Realtime 3D Graphics for the PC,” in Proc. SIGGRAPH, pp.353-363, 1996
[3] William R. Hamburgen, et al,”Itsy: Stretching the Bounds of Mobile Computing,” IEEE Computer, pp. 28-36, Apr, 2001
[4] Yong-Ha Park, et al, “A 7.1-GB/s low-power rendering engine in 2-D array-embedded memory logic CMOS for portable multimedia system,” Journal of Solid-State Circuits, pp. 944-955, June, 2001
[5] Ramchan Woo, et al, “A 120mW embedded 3D graphics rendering engine with 6Mb logically local frame-buffer and 3.2Gbyte/s run-time reconfigurable bus for PDA-chip,” Symp. on VLSI Circuits, Dig of Tech. Papers, pp. 95-98, 2001
[6] http://www.arm.com
[7] Steve Furber, “ARM: System-on-chip Architecture”, 2nd edition, Addison-Wesley press, 2000
[8] ARM Software Development Tookit version 2.50 User Guide, Advanced RISC Machines, Nov, 1998
[9] ARM7500 FE Datasheet, Advanced RISC Machines, Sep, 1996
[10] Kanako Yoshida, et al, “A 3D graphics library for 32-bit microprocessors for embedded systems,” IEEE Trans. on Consumer Electronics, pp. 1107-1114, Aug, 1998