10.6 A 50Mvertices/s Graphics Processor with Fixed-Point Programmable Vertex Shader for Mobile Applications

Ju-Ho Sohn, Jeong-Ho Woo, Min-Wuk Lee, Hye-Jung Kim, Ramchan Woo, Hoi-Jun Yoo

KAIST, Daejeon, Republic of Korea

The real-time 3D graphics is one of the most important features in 3G mobile applications such as cellular phones and PDAs. Their graphics processors integrate many rendering hardware to provide more realistic images within limited battery lifetime and small memory bandwidth [1-4]. Moreover, the average eye-to-pixel angle is larger than that of a PC, because users often hold the small screens closer to their eyes. Therefore, every pixel in mobile applications should be drawn with higher quality than that in a PC. Until recently, only the dedicated hardware graphics engines, which are datapath-centric, have been studied to provide high drawing speed with low power consumption for mobile applications [2-3]. However, the flexibility for OpenGL or DirectX shading language extensions should be supported for high quality of graphics images. In this work, the implementation of a user-programmable graphics processor in mobile applications is reported. The hardware is designed with programmable shading architecture using fixed-point arithmetic, and optimized to an ARM10 co-processor for high performance as well as flexibility in low power consumption.

The graphics processor contains a user-programmable 128b, 4×32b fixed-point SIMD vertex shader [1] as shown in Fig. 10.6.1. To implement the programmability for high quality graphics, the vertex shader is configured as instruction set extensions of ARM10 using co-processor architecture. Moreover, enhanced from the conventional architecture of ARM co-processor, the proposed vertex shader has an internal code memory with dedicated vertex program control unit in order to process various vertex programs independently of ARM10. Although the vertex shader has multiple register files for processing of the streaming vertex data, input/output vertex register files have only 1-read/1-write port and global register-forwarding logic is used only in the general-purpose SIMD register file to reduce the chip area. The intermediate results of SIMD multiply are bypassed locally in arithmetic units.

Figure 10.6.2 shows the block diagram of the graphics processor consisting of a 32b RISC processor, 128b user-programmable vertex shader, a rendering engine (RE) and a programmable frequency synthesizer (PFS). The co-processor interface connects the vertex shader to ARM10-compatible 32b RISC processor, transferring the vertex shader instructions. The vertex shader performs all per-vertex operations such as geometry transformations and lighting, and accelerates primitive assembly such as clipping and culling. RE employs a low power 128b SlimShader rendering engine [2] with 26kB SRAM graphics cache system. The graphics cache consists of 2D-screen-array direct-mapped frame and depth caches, and two 2-way set associative texture caches. RE is responsible for the rasterization and the per-pixel operations such as alpha blending and texture mapping. The internal vertex buffer is used for RE commands transfer from the vertex shader. Since all intermediate data and instructions are transacted through the co-processor interface and the vertex buffer, the system bus interfaces are used only to transfer input vertices and output pixel data. Both of the RISC with 16kB I/D caches and the vertex shader operate at 200MHz. RE operates at the quarter frequency, 50MHz, for low power consumption.

The chip controls its power consumption at both instruction and block levels. The vertex shader reduces power consumption by instruction-level power-control scheme as shown in Fig. 10.6.3. ARM10 drives co-processor instruction valid (CPINSTV) signal only when vertex shader instructions are called. Using CPINSTV, the clock signal of each SIMD register file can be gated off when it is not required. CPINSTV also reduces the power dissipated in datapath of SIMD arithmetic units by eliminating the unnecessary transactions. Therefore, the co-processor architecture of the vertex shader shows fine-grained power management on an instruction-by-instruction basis, and achieves up to 43% power reduction in various graphics applications.

To manage the dynamic power consumption at the block level, PFS of Fig. 10.6.4 is used. Although the previous implementation supports only abrupt change between three frequencies (2×, 1×, 0.5×) [2], this PFS can continuously tune the target frequencies with PLL-type frequency synthesizer covering wide frequencies ranging from 8 to 271MHz. Once the operation mode is selected by OP_MODE (fast/normal/slow), FREQ_CTRL sets the target frequency. The frequency in fast mode can vary from 32 to 271MHz with 1MHz steps, in normal mode from 16 to 135.5MHz with 500kHz steps, and in slow mode from 8 to 67.75MHz with 250kHz steps. Since the 3D graphics applications are executed at a given frame rate, only a finite amount of pixels should be drawn within the time slot of a single frame. After the vertex shader and RE finish drawing pixels, their datapaths are unnecessarily clocked for the rest of the time until the next frame restarts. The software graphics library running on the RISC measures the average workload of the current frame, and sets the target frequency of PFS adaptively for processing of the next frame. Even though the frequency of the clock output (CKout) changes continuously before it locks to the target frequency, the chip can operate reliably since all logics are designed with fully static circuits and the chip communicates with off-chip devices asynchronously. The PLL locking time is less than 50μs and it consumes less than 2mW. PFS provides four different clocks, and each clock can be selectively gated on or off by the software.

The chip is fabricated in a 0.18μm 6M CMOS standard logic process. The die size of processor core is 23mm^2 including 2M logic transistors and 96kB SRAM. The standard bi-directional asynchronous SRAM interface allows it to operate with any existing microprocessor or mobile-system chipset. It consumes 155mW at 200MHz RISCclk / 50MHz REclk and 1.8V. The chip achieves 50Mvertices/s geometry performance and 200Mtexels/s drawing speed with bilinear MIPMAP texturing. Figure 10.6.5 summarizes the chip features, and Fig. 10.6.6 shows the chip micrograph.

Figure 10.6.7 compares the performance of the proposed architecture and that of the previous works [2-4]. The performance index of graphics processing speed / power consumption is used for the comparison of power consumption and performance. The parallel operations of ARM10 and vertex shader improve the performance by 1.8 times.

References:

[1] Ju-Ho Sohn et al., "A Programmable Vertex Shader with Fixed-point SIMD Datapath for Low Power Wireless Applications," *Graphics Hardware*, pp. 107-114, 2004.

[2] Ramchan Woo et al., "A 210mW Graphics LSI Implementing Full 3D Pipeline with 264Mtexels/s Texturing for Mobile Multimedia Applications," *ISSCC Dig. Tech. Papers*, pp. 44-45, 2003.

[3] Masatoshi Imai et al., "A 109.5mW 1.2V 600Mtexel/s 3-D Graphics Engine," *ISSCC Dig. Tech. Papers*, pp. 332-333, 2004.

[4] Fumio Arakawa et al., "An Embedded Processor Core for Consumer Appliances with 2.8GFLOPS and 36M Polygons/s FPU," *ISSCC Dig. Tech. Papers*, pp. 334-335, 2004.

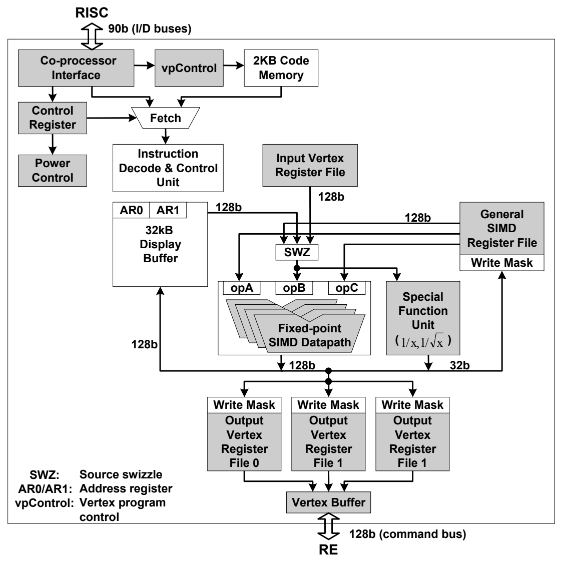
RISC
90b (I/D buses)
Co-processor Interface
vpControl
2KB Code Memory
Control Register
Fetch
Power Control
Instruction Decode & Control Unit
Input Vertex Register File
128b
AR0 AR1
32kB Display Buffer
General SIMD Register File
Write Mask
SWZ
opA opB opC
Fixed-point SIMD Datapath
Special Function Unit
(1/x, 1/√x)
32b
Write Mask Output Vertex Register File 0
Write Mask Output Vertex Register File 1
Write Mask Output Vertex Register File 2
SWZ: Source swizzle
AR0/AR1: Address register
vpControl: Vertex program control
Vertex Buffer
128b (command bus)
RE

Figure 10.6.1: User-programmable fixed-point SIMD vertex shader.

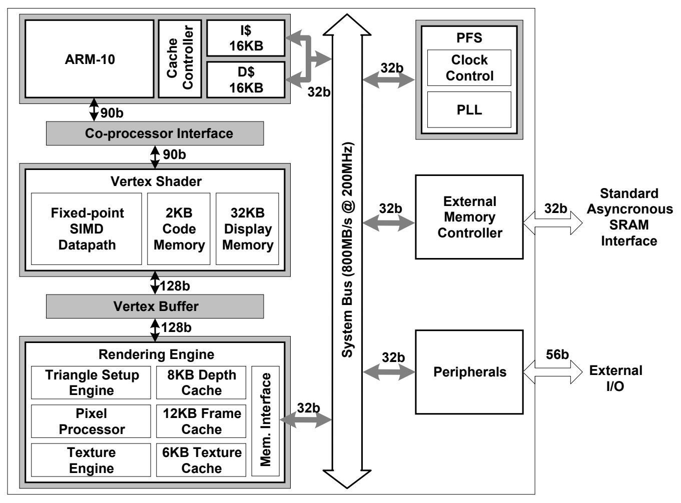

Figure 10.6.2: Block diagram of graphics processor.

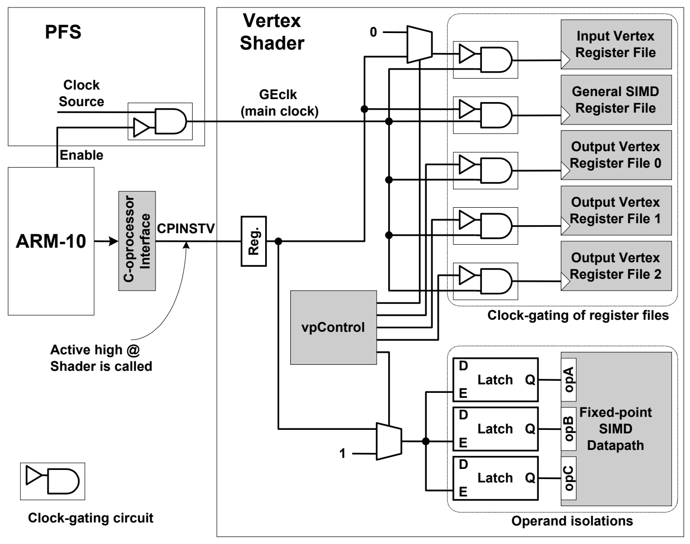
PFS
Clock Source
Enable
ARM-10
C-oprocessor Interface
CPINSTV
Active high @ Shader is called
Vertex Shader
GEclk (main clock)
Reg.
0
1
vpControl
Input Vertex Register File
General SIMD Register File
Output Vertex Register File 0
Output Vertex Register File 1
Output Vertex Register File 2
Clock-gating of register files
D Latch Q E
opA opB opC
Fixed-point SIMD Datapath
Operand isolations
Clock-gating circuit

Figure 10.6.3: Instruction-level power control.

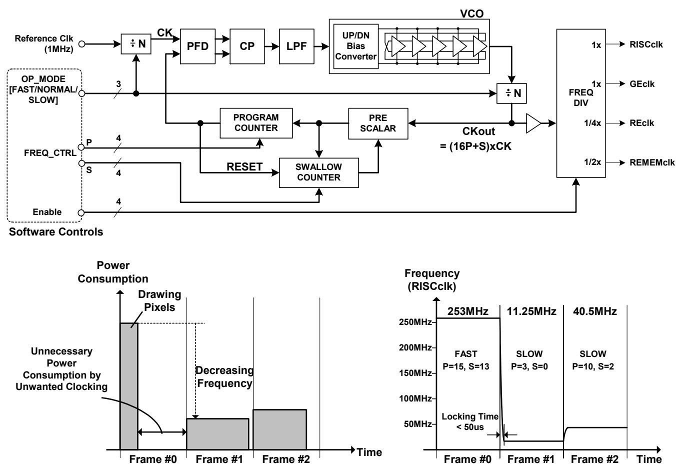

Figure 10.6.4: Programmable frequency synthesizer.

Process technology		0.18µm 6M CMOS
Power supply		1.8V(core), 3.3V(I/O)
Transistor counts		2M Logic 768kb SRAM (96kb)
Die size		4.8mm by 4.8mm (core) 6.0mm by 6.0mm (chip)
Operating frequency (RISC,Vertex shader / 3DRE)		Fast : ~200MHz/50MHz Normal : ~100MHz/25MHz Slow : ~50MHz/12.5MHz
Power consumption		<155mW
Package		256 pin PBGA
Performance	General	1000MIPS (including ARM and vertex shader) 80MFLOPS(software emulation)
	Geometry	50Mvertices/s (Geometry transformation)
	Rendering	50Mpixels/s, 200Mtexels/s (Bilinear MIPMAP filtered pixel)
	Full 3D Pipeline	3.6Mvertices/s (sustaining) (Including full OpenGL lighting, clip check and texturing)
Graphics Functions	Programmability	Vertex program version 1.1 compatible
	Screen resolution	up to 512 x 512 pixels
	Triangle setup	Hardware-accelerated triangle setup engine
	Shading	Gouraud / Flat
	Texture mapping	Point/Bilinear MIPMAP filtering (perspective-correct)
	Antialiasing	x2, x4

Figure 10.6.5: Features summary.

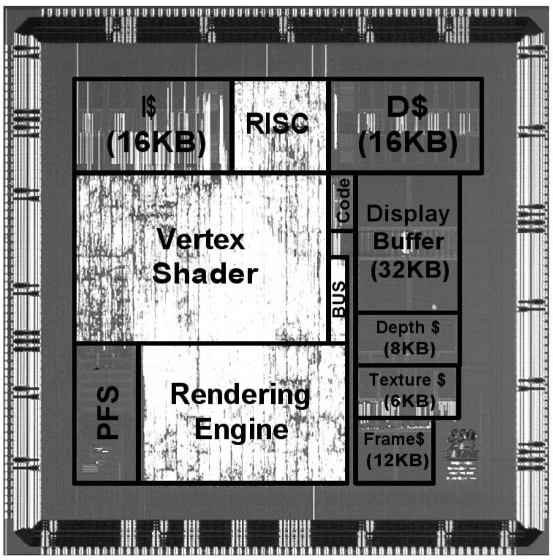

Figure 10.6.6: Chip photograph.

Continued on Page 592

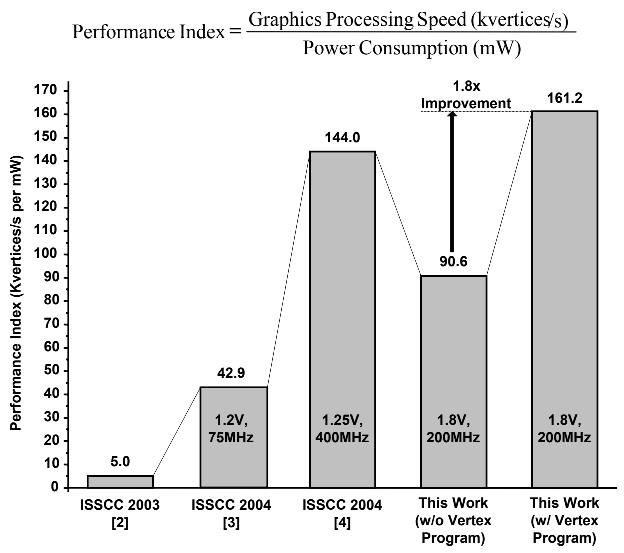

Figure 10.6.7: Performance comparison.