A Power-Aware Reconfigurable Rendering Engine Design with 453MPixels/s, 16.4MTriangles/s Performance

Chih-Hao Chao, Yen-Lin Kuo, An-Yeu Wu, and *Weber Chien
Graduate Institute of Electronics Engineering, National Taiwan University
*Sitronix Technology Cooperation
Taipei, Taiwan ROC

Abstract—This paper presents a power-aware dynamically reconfigurable rendering engine design, which changes power as rendering throughput and image quality change. At algorithm level, a precision-aware shading scheme is proposed to improve the power efficiency of the conventional shading algorithm through the combination of precision detection and fraction masking techniques. At architecture level, a processing element (PE) based scalable architecture is combined with dynamic task scheduling and dispatching techniques to raise hardware utilization rate and reduce computation latency. Finally a prototyping design which delivers 453MPixels/s, 16.4MTriangles/s, 2.24MPixel/mJ is presented.

I. INTRODUCTION

Real-time 3-D computer graphics play an important role in current multimedia application. The need for better image quality and real-time response drives the system designers to increase the rendering performance. Thus it causes enormous power consumption. Many researches show that low power design is the near-term grand design challenge and power will be the only limitation in the future. To evaluate a design, power efficiency, throughput, and area are the design metrics. The power efficiency of a rendering system is defined as pixel rendering throughput over power [1] i.e. how many pixels can be output with the same energy consumed during computation (Pixels/mJ).

Traditional low-power VLSI design skills are constrained by iteration bound and further power reduction relies on other techniques. Static power reduction mainly relies on using multi-threshold voltage cells and most efforts are done by EDA tools. For dynamic power reduction, dynamic voltage scaling (DVS), dynamic frequency scaling (DFS), and gated clock are popular approaches. [2] indicates that the increasing importance of power awareness for VLSI systems. A power aware system is defined as a system which is able to adjust its power consumption in response to varying operating conditions. The changes may be brought by the time-varying inputs, desired output quality, or just environmental conditions. Regardless of whether they were engineered for being power aware, systems display variations in power consumption as operating conditions change.

According to the above definition, for a rendering system, high power awareness means the capability to adjust its power consumption regarding to rendering throughput and image quality. The relation between throughput and dynamic power consumption must be positively related and as linear as possible. To achieve this goal, processing element (PE) based scalable architecture and dynamic reconfiguration schemes are adopted for linear scaling. Because the limitation of parallel execution of the rendering algorithm restricts the scalability and the response of power consumption, a proper division of rendering steps helps to improve both scalability and hardware utilization. Inside PE, to keep power consumption being linearly related to the output pixel rate and image quality, we must take algorithm power efficiency into consideration. In this paper, we improve the power performance with two proposed methods:

- Precision-aware linear interpolation.
- PE-based scalable architecture with dynamically task scheduling, dispatching, and processing.

The architecture verification is through FPGA and a prototyping chip design gives the physical specification for performance measurement and power simulation.

II. ALGORITHM LEVEL DESIGN

Reference [3] shows that triangles are decomposed into two stages for Gouraud shading: Span generation and span interpolation. The function of span generation and span interpolation are demonstrated in Fig.1. Span generation creates the pixels on the three edges based on the three vertices of a triangle. For each span, the pixels are linearly interpolated based on the pixels on the leading edge and the trailing edge. The two stages are similar because they are both a subset of generalized multi-dimensional linear interpolation.

The nature of a rendering system is time-varying volume data processing [4]. For an arbitrary sized triangle, it consists of three vertices, which contain $\{X, Y, Z, R, G, B\}$ information, and has $O(1)$ space complexity. At span generation stage, the space complexity increases to $O(N)$. At span interpolation stage, the space complexity increase to $O(N^2)$. It should be taken into consideration that the total energy consumed to compute the linear interpolation is not only by iterations of generating pixels. It also consumes energy to setup the algorithm. There are many candidates of algorithm to implement the generalized multi-dimension linear

interpolation. The chosen algorithm should cost less energy for both algorithm setup and generation of each pixel.

Traditional division-free scan converting algorithms can be extended to be multi-dimensional and operate at integer number systems. However, these algorithms suffer two problems. Because division-free algorithms adopt residue arithmetic to compute, the slope limitation is not the only constraint to solve the pixel density problem. Any edge and span computation must follow the slope restriction. They cannot guarantee constant pixel generating rate while any of the absolute $\{dR, dG, dB, dZ\}$ is larger than the absolute distance on spatial domain. Also, they all need more mapping from first octant to the others.

Division based interpolation provides constant rendering rates in all cases. However, they need to compute the slope of the edge as the incremental approach, an extension of Digital Differential Analyzer (DDA) algorithm [5]. It is important that the slope computation shouldn't be a heavy overhead for interpolation. In many systems [3, 6], the slope computation takes a series of subtraction and shift to relieve the need of division hardware or takes ROM and multipliers as the SIMD division unit [7]. [6] restricts the divisor between 2 and 8 to reduce slope computation overhead, including the hardware cost and the number of iteration cycle. Here we adopt another approach, which can be generalized to compute any integer division and scales computation power elegantly with controllable error quantity.

Fig.2 shows an example to explain the precision-aware linear interpolation. The precision requirement of slopes in Fig.2 (a) and Fig.2(b) are different. To prevent fixed-point error accumulation from affecting the integer part value of color domain $\{R\}$, the precision required for Fig.2(a) is 2^{-4} , and the precision required for Fig.2(b) is 2^{-8} .

Here we modify the division cycle of DDA algorithm to be variable. For simplification, D denotes the dividend and S denotes the divisor. Equation (1) and (2) describe their values are in such ranges. They are represented by unsigned integer numbers. Q_F denotes fixed-point quotient and Q_R denotes real quotient. The relationship between Q_F and Q_R follows (3) and are represented by unsigned fixed-point numbers.

Figure 1. Decomposition of Triangle Shading

Figure 2. Different Precision Requirement Example

$$2^k \leq D < 2^{k+1} \quad (1)$$

$$2^m \leq S < 2^{m+1} \quad (2)$$

$$Q_R = Q_F + E \quad (3)$$

These variables satisfy (4). The accumulation error $E \cdot S$ is in (5) and should be smaller than 2^{-1} (6). The inequality is solved in (7) and the minimum E is obtained by (8). For radix-2 pre-aligned iterative division, the number of iteration cycle is shown by (9). It takes $(k-m)$ cycles for the integer part quotient and $(m+2)$ cycles for the fraction part.

$$D = Q_R \cdot S = Q_F \cdot S + E \cdot S \quad (4)$$

$$E \cdot S = D - Q_F \cdot S \quad (5)$$

$$E \cdot S < 2^{-1} \quad (6)$$

$$E < S^{-1} \cdot 2^{-1} = 2^{-(m+1)} \quad (7)$$

$$E = 2^{-(m+2)} = 2^{-(m+2)} \quad (8)$$

$$I = (k-m) + (m+2) = k+2 \quad (9)$$

The flow chart of linear interpolation approaches are shown in Fig.3(a). The proposed precision-aware shading scheme is shown in Fig.3(b). A small precision detection unit is used to setup the iteration cycles of division. The number of iteration cycles is reduced according to (9) and the redundant precision in fraction bits are masked to zero.

Figure 3. Interpolation Scheme Comparison

III. ARCHITECTURE LEVEL DESIGN

Due to the time-varying volume data characteristic of rendering, we adopt the block-pipelining approach between key stages. Fig.4(b) shows the three main processing stages: span generation, span interpolation, visibility test and texturing. [7] and [8] show different architecture designs for the rendering system. In [7], the architecture takes constant pipelining to control the clock gating at each pipeline stage

precisely. In [8], the processor-based reconfigurable architecture balances the computation loading in rendering system and reduces the total computation time for given tasks. To raise the hardware utilization of our design, the fixed bonding of processing units and memory partitions should be released and a scheduling unit handles the data exchange. The proposed system architecture is shown in Fig.4 (a). Command dispatcher handles the bus protocol of command interface and dispatches different types of rendering commands to different processing stages. Triangle/span dispatchers, buffers, and span/pixel schedulers form the data path for power aware computing. The data path changes itself adaptively as workload change. Flow control units monitor the buffer status and determine the number of Span Generation PEs (SGPEs) and Span Interpolation PEs (SIPEs). Gated clock is applied to non-operating SGPEs and SIPEs to adjust power at triangle processing level (span generation) and span processing level (span interpolation). Visibility test and texturing unit eliminate redundant texture memory access as [7] did.

The functional verification of the proposed architecture is through FPGA design flow. The texturing of Lena is shown in Fig. 5 by our prototyping design. It also shows the rendering result of given 256 randomly generated triangles.

Figure 4. (a) Block Pipelining and (b) System Architecture

Figure 5. Texturing and Triangle Shading by FPGA Prototyping Design

IV. IMPLEMENTATION RESULTS

A. Layout and Specification

The layout of the power aware rendering engine design is shown in Fig. 6. Table I summarizes the specification of the macro design.

Figure 6. Layout of The Design (Macro)

TABLE I. SPECIFICATION OF THE PROTOTYPING DESIGN

Process Technology	TSMC 0.18um CMOS 1P6M
Power Supply	1.8V
Operating Frequency	250 MHz (Worst case condition)
Transistor Count	1,510K
Core Size	2mm x 2mm
Target Application	Realtime Rendering for 2D/3D Graphics System
Rendering Performance	453M Pixels/s, 14K Triangles/s (Large Triangle) 338M Pixels/s, 249K Triangles/s (Middle Triangle) 94M Pixels/s, 16M Triangles/s (Small Triangle)
Power Performance	3.333 MPixels/mJ (Large Triangle) 2.314 MPixels/mJ (Middle Triangle) 0.664 MPixels/mJ (Small Triangle)

TABLE II. PERFORMANCE COMPARISON WITH AND WITHOUT PRECISION AWARE SHADING SCHEME

Triangle Size	Without Precision Aware (250MHz)			With Precision Aware (250MHz)			Pixel Rate Improvement	Triangle Rate Improvement
	Bus Bandwidth Utilization (%)	Pixel Rate (Pixels/s)	Triangle Rate (Triangles/s)	Bus Bandwidth Utilization (%)	Pixel Rate (Pixels/s)	Triangle Rate (Triangles/s)		
(Pixels)								
255 (Large)	94	451,309,504	13,788	95	453,162,512	13,845	1.004×	1.004×
171	91	436,385,122	29,874	92	439,688,840	30,106	1.008×	1.008×
128	88	420,539,853	50,563	89	426,163,549	51,258	1.013×	1.014×
102	84	400,642,288	75,493	86	411,179,437	77,525	1.026×	1.027×
85	81	388,710,198	104,969	84	401,517,329	108,484	1.033×	1.034×
51 (Middle)	64	306,914,399	226,364	70	337,728,263	249,033	1.100×	1.100×
25	49	237,437,646	694,029	56	269,857,890	788,327	1.137×	1.136×
10	31	149,563,268	2,318,853	38	183,925,468	2,873,880	1.230×	1.239×
5	21	102,272,638	4,987,693	27	130,869,240	6,491,562	1.280×	1.302×
2 (Small)	15	72,408,615	12,272,594	19	94,348,577	16,463,422	1.303×	1.342×

TABLE III. POWER COMPARISON WITH AND WITHOUT PRECISION AWARE SHADING SCHEME

Triangle Size (Width in Pixels)	Without Precision Aware (250MHz)			With Precision Aware (250MHz)			Efficiency Improvement
	Avg. Power (mW)	Pixel Energy (pJ)	Efficiency (MPixels/mJ)	Avg. Power (mW)	Pixel Energy (pJ)	Efficiency (MPixels/mJ)	
255 (Large)	252.878	522.886	1.912	141.438	300.036	3.333	1.743×
171	260.713	574.251	1.741	141.605	316.316	3.161	1.815×
128	270.472	618.162	1.618	141.708	326.254	3.065	1.895×
102	278.323	661.095	1.513	142.044	336.344	2.973	1.966×
85	285.752	704.085	1.420	142.452	347.400	2.879	2.027×
51 (Middle)	306.574	868.855	1.151	143.702	432.199	2.314	2.010×
25	321.664	1308.613	0.764	145.377	512.971	1.949	2.551×
10	345.762	2265.598	0.441	148.560	815.510	1.226	2.778×
5	382.853	3627.960	0.276	152.533	1185.358	0.844	3.061×
2 (Small)	478.930	6422.840	0.156	161.054	1505.987	0.664	4.265×
Average	318.392	1757.435	1.099	146.047	607.838	2.241	2.753×

B. Performance and Power

The proposed precision aware scheme reduces the total switching activity to interpolate the given triangles. From the view of algorithm efficiency, the setup time is tightly related to the space complexity of the processing datum. Table II shows the rendering performance speed-up is from 1.004× to around 1.342×, which results from reduced iteration division. The shortened computation time and the eliminated redundant precision reduce the switching activity. Therefore total energy is saved and reduces the average power consumption. Table III shows the power simulation result of PrimePower. The power efficiency is improved from 1.743× to 4.265× depending on the given triangle size. For smaller triangles, less precision is needed and therefore more power is saved by the proposed precision aware interpolation. It is noted that even the triangle is as large as 255 on both width and height, there is still 1.743× improvement in power efficiency. The shorter spans in the triangle use less precision and result in the energy saving. The average power consumption is 146.0 mW when the macro is operating at 250MHz. It takes on average 607.8 pJ to rendering a pixel. The average power efficiency of the power aware rendering engine is 2.241MPixels/mJ. Table IV. shows the comparison among other works. The proposed design has higher power efficiency. It is noted that this work has higher throughput because of the shorter critical path. Frequency scaling and source voltage scaling techniques can be applied while the performance specification is not very high.

TABLE IV. COMPARISON WITH RELATED WORKS

	Yoo[1]	Sony[9]	MBX[10]	This Work
Technology (μm)	0.18	0.18	0.18	0.18
Throughput (MPixels/s)	20	150	120	453
Frequency (MHz)	10	75	80	250
Power (mW)	17	246.4	115.2	146.0
Efficiency(MPixels/mJ)	1.18	0.61	1.04	2.24

V. CONCLUSIONS

In this paper, a power aware rendering engine design is presented. The power awareness is achieved through algorithm level and architecture level design. The precision

aware shading approach has been advocated over conventional incremental linear interpolation. The elimination of redundant precision calculation reduce the setup overhead of division-based shading algorithm, and the fraction bit masking reduce the switching activity during interpolation of an edge or a span. The scalable, dynamically reconfigurable system architecture promptly controls the working status of computation units and scales power as workload changes. Finally a prototyping design proves the proposed shading approach and system architecture do save power and provides high performance. The power efficiency of rendering can be improved through proposed approaches.

REFERENCE

- [1] Jeong-Ho Woo *et al.* "A 1.2MPixels/s/mW 3-D Rendering Processor For Portable Multimedia Application", ASSCC 2005, Vol. 12, pp. 297-300.
- [2] Manish Bhardwa, Rex Min, and Anantha P. Chandrakasan, "Quantifying and Enhancing Power Awareness of VLSI Systems", IEEE Trans. VLSI Syst., vol. 9, no. 6, Dec. 2001, pp. 757-772.
- [3] A. Kugler, "The setup for triangle rasterization," Proceedings of 11th Eurographics Workshop on Computer Graphics Hardware, Aug. 1996, pp. 49-58.
- [4] Tzi-Cier Chiueh and Kwan-Liu Ma, "A parallel pipelined renderer for time-varying volume data", Parallel Architectures, Algorithms, and Networks, 1997. (I-SPAN '97) Proceedings. Third International Symposium on, 18-20 Dec. 1997, pp. 9-15.
- [5] J. D. Foley, A. V. Dam, S. IC. Feiner and J. F. Hughes, "Computer Graphics – Principles and Practice", 2nd Edition, Addison Wesley Publishing, 1997.
- [6] Ramchan Woo and Hoi-Jun Yoo, "Design and Implementation of Low-Power Embedded 3D Graphics Rendering Engine for Mobile Applications using the Embedded Memory Logic Technology", M.S. Thesis, KAIST, 2001, http://ssl.kaist.ac.kr/%7Eramchan/papers/MS_Paper.pdf
- [7] Ramchan Woo *et al.*, "A Low-Power 3-D Rendering Engine With Two Texture Units and 29-Mb Embedded DRAM for 3G Multimedia Terminals", IEEE JSSC, Vol. 39, No. 7, Jul. 2004, pp. 1001-1009.
- [8] J. Chen *et al.*, "A Reconfigurable Architecture for Load-Balanced Rendering", SIGGRAPH, 2005 Jul. 31, pp. 71-80.
- [9] Masatoshi Imai *et al.*, "A 109.5mW 1.2V 600Mtexel/s 3-D Graphics Engine", ISSCC 2004 Dig. Tech. Papers, pp. 332-333.
- [10] PowerVR-MBX, <http://www.arm.com/products/solution/3DGraphics.html>