An effective rasterization architecture for mobile vector graphics processors

Jinhong Park[1a)], **Jinwoo Kim**[1], **Woo-Chan Park**[2], **Youngsik Kim**[3], **Chelho Jeong**[4], **and Tack-Don Han**[1]

[1] *Dept. of Computer Science, Yonsei University*
134 Shinchong-Dong, Seodaemun-Gu, Seoul, 120–749, Korea
[2] *Dept. of Computer Engineering, Sejong University*
98 Kunja-Dong, Kwangjin-Gu, Seoul 143–747, Korea
[3] *Dept. of Game & Multimedia Engineering, Korea Polytechnic University*
2121, Jungwang-Dong, Shihung-City, Kyounggi-Do, 429–793, Korea
[4] *mGine Corporation*
16–2, Soonae-Dong, Seongnam-City, Kyounggi-Do, 463–825, Korea
a) *jhp@yonsei.ac.kr*

Abstract: This paper proposed a novel index board rasterization architecture which reduces mathematical calculations and memory traffic for vector graphics. The proposed architecture uses the cell based method which has advantages in computational complexity, and generates the active span by referring to only valid cells and placing them in scanline order with two internal SRAMs. The proposed architecture reduces the amount of calculation by an average of 59.4% and also the external memory traffic by an average of 30.0% compared to the traditional architecture.

Keywords: openVG, vector graphics, rasterization, hardware accelerator
Classification: Electron devices, circuits, and systems

References

[1] K. Pulli, "New APIs for mobile graphics," *Proc. SPIE-The International Society for Optical Engineering*, vol. 6075, pp. 1–13, 2006.
[2] K. Kallio, "Scanline edge-flag algorithm for antialiasing," *Theory and Practice of Computer Graphics Conference*, pp. 81–88, June 2007.
[3] D. Kim, K. Cha, and S. I. Chae, "A high-performance OpenVG accelerator with dual-scanline filling rendering," *IEEE Trans. Consum. Electron.*, vol. 54, no. 3, pp. 1303–1311, Aug. 2008.
[4] Anti Grain Geometry (AGG), [Online] http://www.antigrain.com/
[5] Khronos Group Inc, "OpenVG Specification 1.0.1," [Online] http://www.khronos.org/openvg

© IEICE 2011
DOI: 10.1587/elex.8.835
Received April 19, 2011
Accepted May 12, 2011
Published June 10, 2011

1 Introduction

As recent mobile devices require high quality graphics applications, vector graphics (VG) service such as texts, maps, games, animations, and graphical user interface (GUI) has become more important. Thus the implementation of VG hardware on mobile devices is becoming a key issue [1].

A geometry described with VG is defined by one or more paths. Each path consists of a series of edges. Through the rasterization process of each edge, numerous pixels on the edge, called cells, can be generated. The cell data is defined as two values: the coverage to present filtered alpha value, and the area to provide the size of pixels affected by edge for anti-aliasing. With these cells, active spans can be generated and then final anti-aliased pixel colors within each active span can be calculated.

Based on the rendering style, previous methods can be divided into two categories: active edge based [2, 3] and cell based [4] scanline processing. The first method uses an active edge table (AET) for tracking active edges on the current scanline. After cell generation with the active edges on current scanline, active spans are generated in the scanline processing. The second method calculates the cells by applying line drawing algorithms to edges within the path and then active spans are generated by sorting the cells within each scanline in order of the x-axis.

From the viewpoint of memory traffic, the first method is superior because the number of edges is generally less than the number of cells. On the other hand, the computational complexiy of the first method is higher than the second method because it requires additional edge table (ET) setup and calculation of the intersecting point between a scanline and an edge.

This paper, based on the second method, proposes an index board rasterization hardware architecture in order to reduce external memory traffic. The index board denotes the array type of internal memory for a given axis and is accessed directly by either the x or y coordinate value. The proposed architecture takes advantage of the lower hardware cost of the second method while alleviating the memory traffic problem resulting in an architecture suitable for mobile devices compared to the previous methods. In order to meet this aim, the proposed architecture includes a Y-index board internal SRAM to store the number of cells generated for each scanline, and an X-index board internal SRAM to store the sorted cells of the scanline size. The traditional sorting method was replaced by our proposed method, which reads only valid cells stored in the external memory through referring the Y-index board SRAM and the X-index board SRAM. The implementation results show that the proposed architecture reduces external memory traffic by up to 53.6% compared to that of [4].

2 Related works

In active edge based scanline processing [2, 3], active edges are tracked by using AET and are used for scanline processing. For the cell generation, the slope of edge, intersection test between each edge and the x-axis, coverage,

© IEICE 2011
DOI: 10.1587/elex.8.835
Received April 19, 2011
Accepted May 12, 2011
Published June 10, 2011

and area should be calculated with all the active edges for each scanline, resulting in an increase of computational complexity.

In cell based scanline processing [4], cells are generated for each edge and then are placed to the corresponding scanline. The slope calculation is performed only once for each edge and the intersection test does not occur, so that the computational complexity is lower than that of the active edge based scanline processing [2, 3]. On the contrary, the generated cells should be stored in the external memory for later scanline processing, so that the external memory traffic increases. Moreover, the cells belonging to different edges should be sorted for scanline processing, which increases computational complexity and memory traffic.

3 Proposed rasterization architecture

This paper proposes a novel rasterization architecture to reduce external memory traffic and the comparison operations caused by sorting. For this purpose, we proposed an index board sorting unit which includes a Y-index board SRAM and an X-index board SRAM.

Fig. 1 shows the proposed rasterization architecture. In the external memory, a path buffer to store a series of edges of the path and a cell array which is a 2D array are included. The cell array can be indexed per scanline corresponding to the y-coordinate value, and every cell with the same y-coordinate value are stored in the same scanline position. Y-index board SRAM stores the number of cells generated for each scanline. X-index board SRAM is a buffer to store the sorted cells for a corresponding scanline.

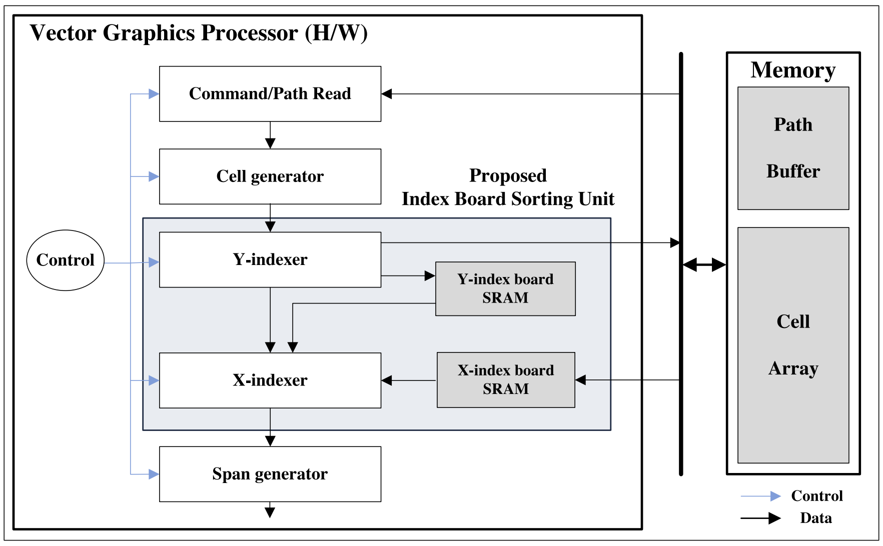


Fig. 1. Proposed Rasterization Architecture.

The processing flow of the proposed architecture is as follows. First, with a given path, the cells are generated with a series of edges and stored in the corresponding scanline of cell array in the generated order. The number of generated cells for each scanline is updated to Y-index board SRAM. After

© IEICE 2011
DOI: 10.1587/elex.8.835
Received April 19, 2011
Accepted May 12, 2011
Published June 10, 2011

every process of generating cells of the path is completed, the cells in the same scanline are retrieved from the cell array and then active spans are generated by the scanning direction in the scanline. The generated active span is transferred to the span and the remaining processes are executed.

In retrieving the cells from the cell array, the Y-index board SRAM is accessed to obtain the valid size of memory access for the cells within a corresponding scanline. The retrieved cells are stored into the X-index board SRAM by using the x-coordinate value of each cell as the index of the X-index board SRAM. For example, if x-coordinate value of the retrieved cell is k, the cell is stored into k-th position of X-index board SRAM. If the location in which cells are indexed overlaps, the current cell is accumulated with the previouly stored cell. Through this process, cells can be sorted without any comparison operation, reducing computational complexity and external memory traffic.

In the point of the compuation requirement, the proposed method to generate cells is the same as [4], whereas there is no calculation for cell sorting in the proposed method. On the other hand, the external memory traffic requirement of the proposed method is to store the generated cells into the cell array and to retrieve the cells from the cell array. Note that there is no additional external memory traffic for cell sorting of [4].

In the point of burst transaction of the external memory, the proposed method is more effective compared to the active edge based method which requires numerous external memory accesses on the edge data in order to maintain the active edge table. Unfortunately, it is difficult to access continuous memory address space in accessing the edge data and the active edge table. On the other hand, the proposed method can retrieve a group of cells by burst transaction from the cell array of the external memory into the X-index board rasterization.

The cell information consists of a 24-bit for representing x-coordinate, which is the same size as [2] and [3], a 9 (1 sign + 8 magnitude)-bit for representing a coverage according to the specification of [5], and a 17 (1 sign + 16 magnitude)-bit for representing an area according to area-sampling anti-aliasing of [4]. Therefore, the number of bits required to store one cell data is 50.

Assuming a resolution of VGA, the Y-index board uses 0.59 KB of SRAM as 10-bit of memory space is required. On the other hand, the X-index board uses 3.9 KB of SRAM as 50-bit of memory space is required to store cells per index.

4 Experimental result & hardware implementation

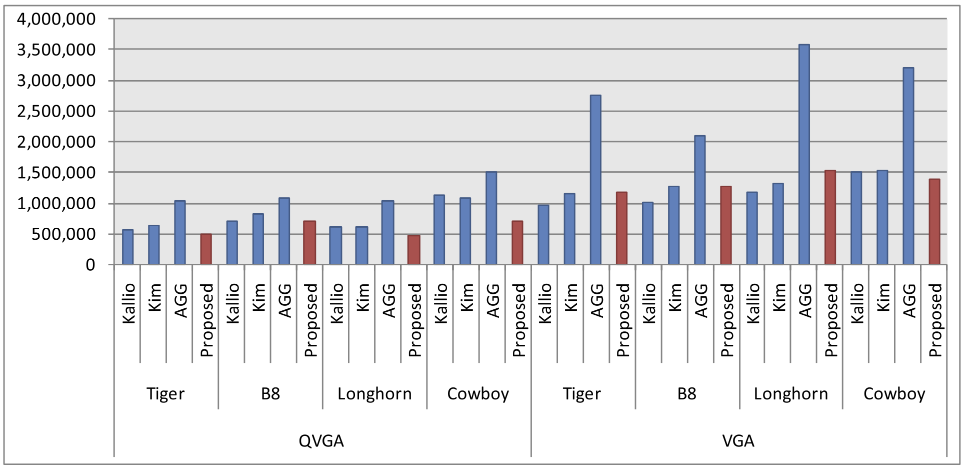
To compare the proposed rasterization architecture with the traditional architecture, *Tiger*, *B8*, *Longhorn*, and *Cowboy* images of [4] are used as test-benches.

The memory access formula of [3] was used in order to calculate extenal memory traffic of [2] and [3]. Equation (1) calculates the external memory

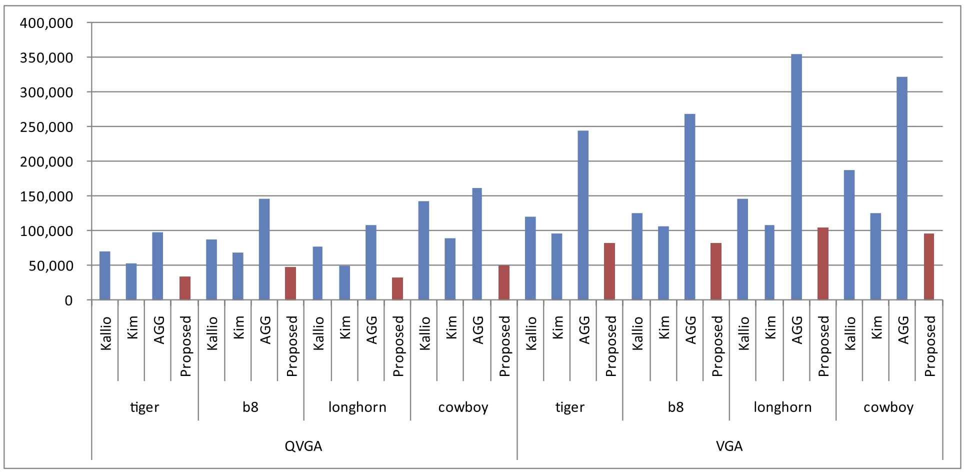
© IEICE 2011
DOI: 10.1587/elex.8.835
Received April 19, 2011
Accepted May 12, 2011
Published June 10, 2011

traffic of [4] and the proposed architecture. Each term represents the number of memory write, sorting, and memory read operations. In equation (1), N stands for the number of edges, $Cells\ of\ Edge_i$ stands for the number of cells created by each edge, and $K1$ stands for the number of bytes to store one cell which includes each x-coordinate value and coverage data for anti-aliasing.

$$\begin{aligned} Memory\ Traffic &= \sum_{i=1}^{N} Cells\ of\ Edge_i \times K1 + \sum_{i=1}^{N} Cells\ of\ Edge_i \\ &\times \log\left(\sum_{i=1}^{N} Cells\ of\ Edge_i\right) \times K1 + \sum_{i=1}^{N} Cells\ of\ Edge_i \times K1 \end{aligned} \tag{1}$$



(a) **Number of External Memory Traffic (Byte)**



(b) **Processing Throughput (Number of External Memory Accesses)**

	divider	multiplier	adder	comparator
Kallio	1	2	3	1
Kim	1	2	5	1
AGG	1	1	3	1
Proposed	1	1	3	0

(c) **Comparison of Hardware Resources for a Pipeline**

Fig. 2. Experimental Result.

© IEICE 2011
DOI: 10.1587/elex.8.835
Received April 19, 2011
Accepted May 12, 2011
Published June 10, 2011

Fig. 2 shows the external memory traffic, processing throughput and required hardware resources of the proposed architecture and previous methods for four benchmarks. Fig. 2 (a) shows that the external memory traffic of the proposed architecture is reduced by 35.4% ∼ 53.6% compared to [4], 0% ∼ 37.1% compared to [2], and 15.4% ∼ 33.4% compared to [3] at QVGA. In VGA resolution, the traffic is reduced by 39.1% ∼ 57.5% compared to [4] and is similar compared to [3]. On the other hand, the traffic is slightly increased compared to [2] in cases of *Tiger*, *B8*, and *Longhorn* benchmarks, whereas in case of *Cowboy* benchmark with high scene complexity it is slightly reduced compared to [2].

For throughput comparison, we assume that hardware resource is not considered so that only the external memory access overhead can degrade the overall performance. For this purpose, we count the number of external memory accesses for each method. We also assume that 8 burst AHB bus transaction is adopted in external memory access. Fig. 2 (b) shows that the throughput of the proposed method is improved by approximately 128.9% compared to [2], 58.5% compared to [3], and 211.8% compared to [4] at QVGA. In VGA resolution, throughput is improved by about 57.0% compared to [2], 19.1% compared to [3], and 219.8% compared to [4].

The hardware resources needed to construct a pipeline of each method are shown in Fig. 2 (c). As a result, the proposed method can reduce a multiplier for a pipeline compared to [2] and [3].

The proposed architecture is built on an FPGA board containing a Xilinx Virtex4 LX200 FPGA operating at 48 MHz. The rendering time of the proposed architecture is 11.4, 12.3, 13.2, and 8.3 frame/second for *Tiger*, *B8*, *Longhorn*, and *Cowboy*, resepectively. The implementation in FPGA board is shown in Fig. 3.

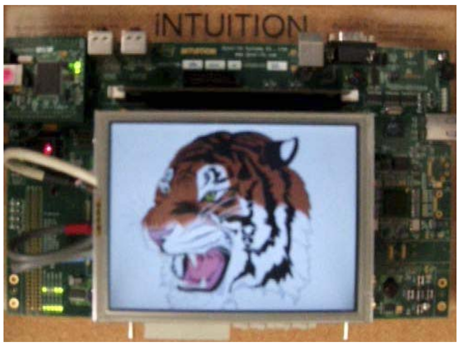


Fig. 3. Implementation in FPGA Board.

5 Conclusion

This paper proposed an effective rasterization architecture with a small amount internal SRAM each for the X-index board and the Y-index board to reduce the memory traffic without sorting for active span generation. Experimental results show that the proposed architecture achieves lower com-

© IEICE 2011
DOI: 10.1587/elex.8.835
Received April 19, 2011
Accepted May 12, 2011
Published June 10, 2011

putational complexity and higher processing throughput compared to the previous approaches.

Acknowledgments

This work was supported by the Small & Medium Business Administration of Korea (No. S0807222-E0821730-10100001).

DOI: 10.1587/elex.8.835
Received April 19, 2011
Accepted May 12, 2011
Published June 10, 2011