Research Inventy: International Journal Of Engineering And Science
Vol.2, Issue 8 (March 2013), Pp 01-05
Issn(e): 2278-4721, Issn(p):2319-6483, Www.Researchinventy.Com

Implementation of a 3D Graphics Rasterizer with Texture and Slim Shader on FPGA

[1,] Ajay Kashyap, [2,] Ashish Sharma
[1,2,] *Department of Electronics and Communication Engineering, HIET, Shahpur, Himachal Pradesh, India*

Abstract *–In this paper, we designed 3D graphics hardware with rasterizer having texture and slim-shader for the efficient 3D graphics accelerator. The rasterizer consists of vertical shader and triangle set-up with AAL slim-shader. We developed all modules (vertex shader, pixel shader, slim shader, clipping engine, triangle set-up engine and raster operator) of 3D pipeline on FPGA using RTL design. It is designed to support the OpenGL ES 2.0 and Shader model 3.0. This paper also gives a brief description about generic graphics pipeline and OpenGL pipeline.*

Keywords- *3D Graphics, OpenGL, 3D pipeline, AAL, FPGA, Texturing, Shading, Rasterization.*

I. INTRODUCTION

3D graphics and imaging is going through rapid advancement with developments in hardware technology. It has applications ranging from medical imaging , scientific visualisation, visual reality stimulation to high end gaming, and many more. 3D Graphic hardware is one of the major applications in computer industry. Many companies are currently struggling to develop 3D graphics accelerators. Emphasis is placed on design issues and decisions for 3D graphics rasterizer design, which includes triangle set-up, texturing and shading process unit. The 3D graphics processing consists of several computational steps, including geometry processing and rasterization. The 3D graphics pipeline is shown in Figure1.

A scene is described in three dimensions which is called the world coordinate system, and task of geometry engine is to apply a set of transformations and additional operations such as projection, clipping, cutting and lightening. Geometry transformations handle the position and orientation of models in the screen [1]. OpenGL is a software API for 3D graphics, a combination of more than 120 functions used to specify 3D models, operations in various parameters that control rendering. Rendering is the process of generating a 2D image of a 3D scene [4]. In most cases OpenGL ES 2.0 is used. In the embedded apparatus, the OpenGL ES 2.0 is the standard application program interface which is defined in order to process the graphics [2][3]. OpenGL ES improves the programmability using shader. Shader describes detail part of 3D image.

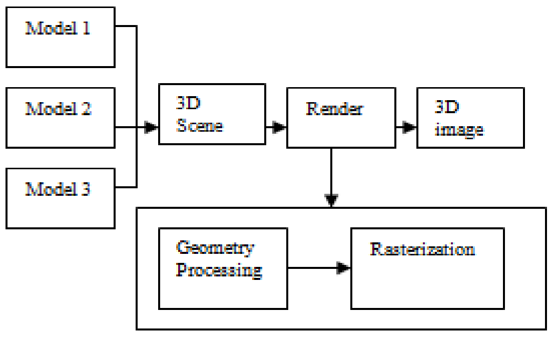


Figure 1 A generic 3D graphics Pipeline

Proposed hardware supports low power and high performance using RTL with openGL ES 2.0 and shader model 3.0 was supported in order to strengthen the programmable capability. An operation was forced in FPGA (field programmable gate array) in order to verify this. The FPGA test board has 128kbyte of SDRAM of network interface and serial communication port. The cache system for 3D graphics accelerator was implemented by the internal SRAM on FPGA chip. The 3D graphics library was implemented with Mesa3D library [7].

II. RASTERIZATION

In rasterization vertices associated with each primitive has 2D screen coordinates, primitives are essentially triangles. In addition to 2D screen coordinated each vertex has color and texture coordinates with are used to color and texture the triangle. From this point on rasterization gets complex with enormous amount of data sets and computations [8].Rasterization is a process of taking a 2D image and converting it into pixels or dots for output on a video display. Task of rasterization is to identify all pixels that are covered by the primitive (triangle in our case), assign color and additional attributes to each such pixel [5]. Scan converting a triangle generates a set of pixels which fall inside triangle which hereafter referred as fragments. Each fragment gets its own x, y screen coordinates, color and texture coordinates computed by interpolating color, texture values across triangle edge and across scan lines.

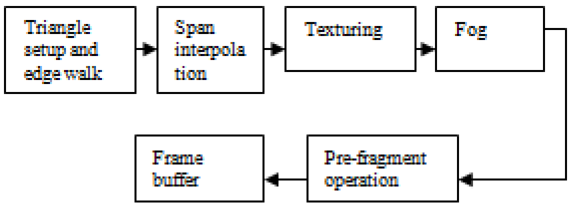


Figure 2 Rasterization pipeline

A. Scan

2D screen can be thought as rectangular grid of pixel having values x and y [9][10]. The x value increases along horizontal and y along vertical respectively. Each scan line is represented by y. Rasterization unit gets 2D screen co-ordinates of vertices associated with each primitive. Scan conversion starts by sorting the vertex in decreasing order of y, which starts with top most scan line transverses down to bottom most scan line that intersects the primitive.

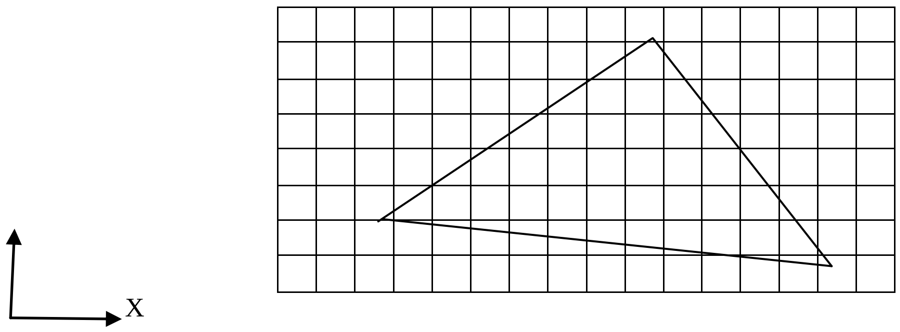


Figure3. Triangle covering pixel centres.

Scanning generates fragments for each pixel that falls inside the primitive (triangle) with respective x, y screen co-ordinates [6]. Scan conversion also assigns texture co-ordinates; color and death values to each fragment generated depending upon the shading and texture modes. Task of triangle setup, edge-walk and span interpolation is to generate fragments that fall inside the given triangle at the same time assign, color, depth and texture co-ordinates to each such fragment.. Task of triangle setup, edge-walk and span interpolation is to generate fragments that fall inside the given triangle at the same time assign, color, depth and texture co-ordinates to each such fragment.

B. Texturing

Texturing is a process of printing an image on to a rendered surface. It is one of the most useful technique used in modern graphics machines to render realistic images. Texture mapping proceeds through three phases to assign color to each pixel that falls inside the polygon (triangle).

- Compute texture co-ordinates at each pixel inside the primitive.
- Read texels from texture images.
- Filter texels.
- Modify the fragment color.

Textures are simple rectangular arrays of data for example textures can be represented as rectangular grid of texels each of which can be addressed with(u, v) texture co-ordinates just as screen pixels can be addressed with x, y screen co-ordinates. Texels contain color information such as RGB and in some case an alphabet value is also present. Each fragment gets its texture co-ordinates and the location of the base texture image in the texture memory. The idea of texture mapping is to read respective texel RGBA values from memory filter according to filtering mode to remove anti-aliasing artifacts and distortion. There are three types of filtering modes point sample, bilinear, and trilinear.

Several buffers are needed for 3D graphics processing, which are the frame buffers for storing color values of each pixel, the alpha buffer for storing transparency value of each pixel, and the depth (z) buffer for storing depth value of each pixel in local memory (SDRAM). All of these buffers have 16-bit size per one pixel on this platform, in addition after a screen is rendered, every buffer could be cleared before the next scene generates. Proposed 3D graphics hardware operated with 35MHz in the increased vertex5 and it is able to render the 70,000 triangles at 30 frames/second.

III. PROPOSED RASTERIZATION

A triangle vertex data structure in addition to screen co-ordinates x and y will also contain RGBA color values which are used to assign color to individual fragments during scan conversion. Presently shadings are of two types flat and gourmand shading. Flat shading assigns single color to a primitive depending on surface normal as a result, all the pixels that fall inside are assigned the same color. Flat shading is really very simple to implement. To fill a triangle with flat shading we set same color to each pixel inside the triangle whereas in goround shading each vertex gets a different color. The proposed slim shader performs the main rendering operation such as texturing, shading, blending and depth comparison. Memory programmer (MP) enables the special effect such as anti-aliasing, motion-blurr and fog to be programmable. Integrated DRAM with partial board line scheme provides sufficient bandwidth and capacity required for 3D rendering operation. In slim shader vertical strip assignment is done as shown in figure4. Triangle is divided into two alternative strips of A and B. We have two pixel processors PP0 and PP1. Here horizontal order rasterization is done.

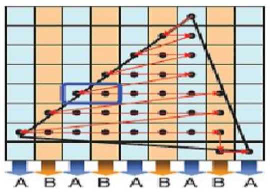


Figure4. Slim shader triangle setup

A triangle setup engine (TSE), which contains single cycle parallel divider distributes triangle to two pixel processors (PP), it enhances the overall performance of 3D pipeline by accelerating setup operation. Slim shader prevents the unnecessary shading and texturing by gating of the clock. For real time special effects, MP post-processes where the rendered pixels of the previous frame transferring them to the display control. Texture memory is integrated with two texture engines (TE). Per pixel dividers are implemented in each TE to support perspective correct addressing which can remove the artifacts in large triangles. Rasterization engine with slim shader is shown in the figure5.

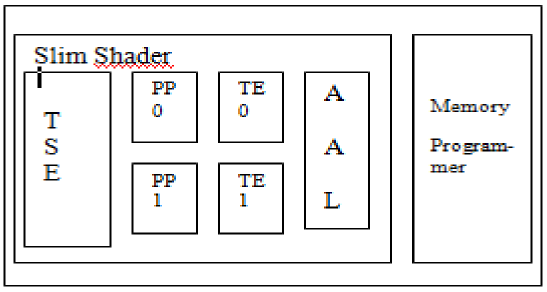
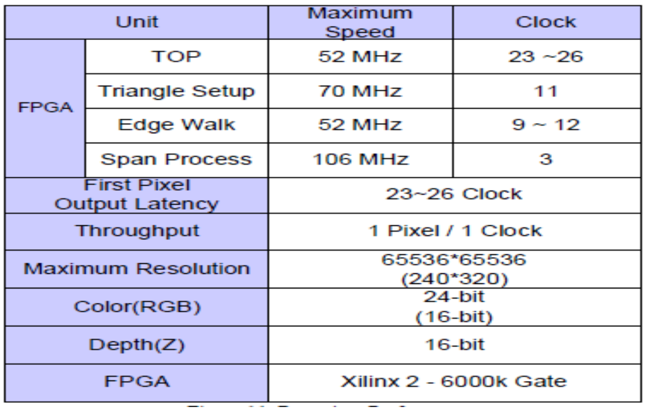


Figure5. Slim shader

Here since TEs perform the bilinear MIPMAP filtering to draw more realistic images, 8 texel requests are generated at every cycle fetching 8 texels directly from 8 TMs may consume large amount of power due to concurrent data transition in many capacitive I/Os and the activation power of TMs themselves. Therefore, the number of memory requests is reduced by adopting address alignment logic. Since the two PPs process adjacent pixels, the nature of MIPMAP filtering allows special Aligner to find and remove the overlapped texels. Then Aligner compares the current texture addresses with previous ones and leaves only the different addresses. Since AAL reduces the average number of requests to less than 2.5, texels can be fetched from only TMs at every cycle. AAL saves power by reducing the number of activated TMs while doubling the performance compared with single TE architecture. AAL reduces 68% of energy required as compared to single TE architecture.

IV. VERIFICATION

A 3D graphics verification system is comprised of the platform including the test bench for simulation and ARM processor subsystem and 3D graphics IP module targeted in FPGA. The platform performing the host controller roll off 3D graphics module is used in a simulation and does a role of a platform. A 3D graphics accelerator usually produces a very large amount of pixel data. It is almost impossible to compare the simulation data and C program data by hand. Automated tools are needed for the simulation data comparison, data presentation and data acquiring in 3D graphics pipeline. The 3D graphics accelerator verification environment is composed of a C program language model of the 3D graphics pipelining.. An FPGA target board is used to verify the functionality of the implementation. The target board is used to verify the entire SoC architecture, which consists of ARM processor, an FPGA chip and peripheral blocks. Figure 6 shows the estimated performance of the embedded 3D graphics accelerator.



Unit		Maximum Speed	Clock
FPGA	TOP	52 MHz	23 ~26
	Triangle Setup	70 MHz	11
	Edge Walk	52 MHz	9 ~ 12
	Span Process	106 MHz	3
First Pixel Output Latency		23~26 Clock	
Throughput		1 Pixel / 1 Clock	
Maximum Resolution		65536*65536 (240*320)	
Color(RGB)		24-bit (16-bit)	
Depth(Z)		16-bit	
FPGA		Xilinx 2 - 6000k Gate	

Figure 6. Rsterizer Performance

V. CONCLUSION

The 3D graphics rasterizer architecture is designed and verified with an FPGA board. The proposed 3D graphics rasterizer has a triangle setup, edge-walk, slim shader, AAL and span processing unit. It supports slim shading with 16-bit color and 16-bit depth values. Scan conversion algorithm are used in the design of the 3D graphics rasterizer. Simulation results were visualized and intermediate date about the graphic pipeline were verified by comparing the results of a C model. This paper suggests effective 3D graphic hardware. It is designed to support the openGL ES 2.0 and shader model 3.0. We design 3D graphic accelerator and verify on FPGA platform.

REFERENCES

[1] Foley, Van Dam; "Computer Graphics Principles and Practice", Second Edition, 2003.
[2] OpenGL ES, http://www.khronos.org/opengles.
[3] Richard S, "OpenGL Super bible", Third Edition, 2004.
[4] Bo Han and Bingfeng Zhou,"Efficint Video Decoding on GPUs by Point Based Rendering", Graphics Hardware, 2006.
[5] Anders Kugler," The Setup for Triangle Rasterization", 11th Euro graphics workshop on Computer Graphics Hardware, August 26- 27, Poitiers Ffrance, 1996.
[6] "An Effective Pixel Rasterization Pipeline Architecture for 3D Rendering Processors", IEEE Transactions on Computers, Vol.52., No-11, PP1501-1508, November 2003.
[7] Mesa 3D Graphics Library, http://www.mesa3d.com.
[8] Z. S. Hakura and A. Guptsa," The Design and Analysis of a Cache Architecture for Texture Mapping", Proceeding of the 24th International Symposium on Computer Architecture, PP 108-120, June 1997.
[9] Hyun Jae Woo," A Design of an Effective Control and Execution Method For Geometrical Engines and Rasterization within Embedded 3D Graphics Accelerator", Yonsei Univ. Phd Thesis, 2003.12
[10] Tomas, Eric," Rel-Time Rendering", Second Edition, 2002

AUTHORS PROFILE

Mr. Ajay Kashyap received his M. Tech degree from GBTU, Lukhnow,Uttar Pradesh, India and his BE from NMU, jalgaon,India.Presently he is working as HOD in Department of ECE in Himachal Institute of Engineering and Technology. He had wide experience In Teaching and carried out many project at Graduate And post graduate level for his students.

Mr. Ashish Sharma is Pursuing his MTech degree from BUEST, Baddi, Himachal Pradesh. and he is BE in E&TC from NMU, Jalgaon (M.S.), India. He is presently Working with Himachal Institute of Engineering and Techmology, Kangra, India. He has enough Experience in his field and he is very young and dynamic member of his institution.