Computer Engineering and Intelligent Systems www.iiste.org
ISSN 2222-1719 (Paper) ISSN 2222-2863 (Online) DOI: 10.7176/CEIS
Vol.10, No.3, 2019

3D Stereo Rendering Using FPGA

Marwa Riyadh Ahmed Dr. Basma MohammedKamal Younis
Department of Computer Technology Engineering, Technical Engineering College,
Northern Technical University, Mosul, Iraq

Abstract
Stereo rendering presents a virtual 3D scene from two slightly different vantage points. It is of great importance in the field of machine vision, robotics and image analysis. This paper proposes a stereo vision system that is realized in a single field programmable gate array (FPGA). Calculations of the stereo pairs are made by using two-center projection (off-axis) method. The first red resultant image is for left eye while the second blue one is for right eye; the 3D illusion is produced when looking to them using anaglyph. This computer graphic hardware system is implemented using Spartan3E XC3S500E FPGA kit. The execution time for the proposal is 1266 faster than OpenGL time with maximum operating frequency of 35.417 MHz, while the max occupation area reaches 84%.

Keywords: Computer Graphic; Stereoscopic; anaglyph; FPGA; two-center projection ;Off-axis Method; stereo pairs.
DOI: 10.7176/CEIS/10-3-04
Publication date: April 30th 2019

I. Introduction

Stereoscopic 3D is a popular form of entertainment, robotic vision system, and it is fast becoming a large industry that attempts to recreate the human vision system by using two or more 2D views of the same scene to derive 3D depth information about the scene[1][2][3].

The visual system is a part of the central nervous system, which gives organisms the ability to process visual detail, as well as enabling the formation of several non-image response functions. It detects and interprets information from visible light to build a representation of the surrounding environment. The visual system carries out a number of complex tasks, including the reception of binocular perception from a pair of two-dimensional projections; the identification and categorization of visual objects; assessing distances to and between objects; and guiding body movements in relation to the objects seen [1]

When formation a 3D graphic scene, two projected images are used to capture separate graphic of the same object from slightly different angles at one fixed viewpoint. The left one is shown only to your left eye while the right projected image is shown only to your right eye, and then fuses these two graphics to give stereoscopic vision.

To see 3D scene correctly anaglyph can be used Anaglyph 3D is the name given to the stereoscopic 3D result achieved by means of encoding each eye's image using filters of different colors, typically red and blue. Anaglyph 3D images contain two differently filtered colored images, one for each eye. When viewed through the "anaglyph glasses", each of the two images reaches the eye in which it is intend for revealing an integrated stereoscopic image. The visual cortex of the brain fuses this into the perception of a three-dimensional scene or composition [4].

Oskar Th., in 2006, investigates how disparity estimation may be used to visualize an object on a 3D-screen. This work has been implemented in MATLAB and C then comparisons between the different implementations have been presented [5].

In 2010, S. Jin et al. have built an FPGA-based stereo vision system using census transform, which can provide dense disparity information with additional sub-pixel accuracy in real time. The proposed system was implement within a single Virtex-4 XC4VLX200-10 FPGA from Xilinx including all the pre and post-processing functions. The hardware implementation is more than 230 times faster when compared to a software program operating on a conventional computer [6].

Akhil Valsaraj et al., in 2015, proposed a stereo vision system that provides dense depth maps in real-time from monochrome cameras. The entire process is realized in a field programmable gate array (FPGA). FPGAs, being inherently parallel in nature, were quite suitable for their problem and were founded to be much faster for such computer vision applications from previous studies [7].

Authors in 2016 design and implement a stereo vision system that gave a real-time depth map of the scene with minimum cost. The system was implement successfully on an FPGA and the output was observe to be in correspondence with the idea of stereo vision. The system can reach frame rates greater than 200fps. [8]

In 2017, Te-Chi Hsiao and Chin-Jung Yang applied a graphics processing method for three-dimensional images on the buffers; first buffer was for storing right-view contents and second buffer was for storing left-view contents, and they used steps: when a current VSync status indicates that a display engine is not operating within a right VSync period of a right-view frame, the drawing engine draws the right-view contents stored in first buffer; when current Vsync status indicates that the display engine is not operating within a left Vsync period of a left-view frame, the drawing engine draws the left-view contents stored in second buffer; during the right Vsync period of the right view frame, the display engine displays right-view contents stored in first buffer; and during the left Vsync period of the left-view frame, the display engine displaying left view contents stored in second buffer[9].

This paper is organized into five sections. In ***section 1,*** it has been tackled an introduction and the previous related researches. ***Section 2*** includes basic rendering operations used for management operations that are used to create stereoscopic 3D scene. While ***section 3*** describes a system hardware design and implement then ***section 4*** gives analysis of results observed after implantation. Finally ***section 5*** ends this paper with conclusions.

II. Theory

There are many methods to calculate stereo pairs used to create a perception of depth, that sets up a virtual camera and rendering two stereo pairs, and the most used methods are ***Toe-in*** and ***off axis*** methods[8]. The later one is used in this work, it introduces no vertical parallax and it therefore creates the less stressful stereo pairs. It requires a non-symmetric camera frustum, this is supported by some rendering packages, in particular, OpenGL.[1,10]. See figure 1.

This method calculates stereo pairs as follows: ***First*** choose the camera aperture, typically between 45 and 60 degrees. ***Next,*** choose a focal length, the distance at which objects in the scene will appear to be at zero parallax. Objects closer than this will appear in front of the screen, objects further than the focal length will appear behind the screen. How close objects can come to the camera depends somewhat on how good the projection system is but closer than half the focal length should be avoided. ***Finally,*** choose the eye separation to be 1/30 of the focal length [1,10].Figure 2 shows these calculations.

Suppose AB = dleft and BD = dright. Also, the same level of projection used for each eye of the left eye and right eye ,making use of similar triangles to find the parameters for glFrustum() ,an OpenGL function ,for each of the two eyes. [1]

The left eye calculations:

$$\begin{aligned}
&L = left = -\overline{AB} = -d_{left}\\
&R = right = BD = d_{right}\\
&B = bottom\\
&T = top\\
&N = neardistance\\
&F = fardistance\\
&f = focal\ length\ of\ camera\\
&e = eye\ separation\\
&\theta = field\ of\ view\\
&\rho = aspect\ ratio = \frac{width\ (W)}{height\ (H)}
\end{aligned} \qquad \ldots(1)$$

Where W and H are the width and height of the near-plane, and $L;R;B;\ T$ denote the left, right, bottom and top boundary coordinates of the near-plane respectively. Assuming that the near-plane lies in the x-y plane with y-axis pointing upward, therefore:

$$\begin{aligned}
&T = N\tan\frac{\theta}{2}\\
&B = -T\\
&H = T - B = 2T
\end{aligned} \qquad \ldots(2)$$

Applying similar triangle properties to calculate the half-width α of the projection plane

$$\frac{2a}{f} = \frac{W}{N} \qquad \ldots(3)$$

Therefore,

$$\frac{a}{f} = \frac{W}{2N} = \frac{\rho * 2T}{2N}\ \rho * \frac{T}{N} = \rho * \tan\frac{\theta}{2} \qquad \ldots(4)$$

from (4):

$$a = f * \rho * \tan\frac{\theta}{2} \qquad \ldots(5)$$

Once a is known, we can determine the distances b and c

$$\begin{aligned}
&b = a - \frac{e}{2}\\
&c = a + \frac{e}{2}
\end{aligned} \qquad \ldots(6)$$

From similar triangles:

$\frac{d_{left}}{b} = \frac{N}{f} = \frac{d_{right}}{c}$

Also, $L = -d_{left} = -b * \frac{N}{f}$ …(7)

$R = d_{right} = c * \frac{N}{f}$ ….(8)

Combining (4), (5), and (6):

$L = -b * \frac{N}{f} = -a * \frac{N}{f} + \frac{e}{2} * \frac{N}{f} = -\rho * \frac{H}{2} + \frac{e}{2} * \frac{N}{f}$

$R = c * \frac{N}{f} = a * \frac{N}{f} + \frac{e}{2} * \frac{N}{f} = \rho * \frac{H}{2} + \frac{e}{2} * \frac{N}{f}$ …(9)

Variables L;R; T, and B are used as the input parameters for the *glFrustum()* function of the left eye.

$$\begin{pmatrix} \frac{2N}{R_l - L_l} & 0 & \frac{R_l + L_l}{R_l - L_l} & 0 \\ 0 & \frac{2N}{T-B} & \frac{T+B}{T-B} & 0 \\ 0 & 0 & -\frac{F+N}{F-N} & -\frac{2F*N}{F-N} \\ 0 & 0 & -1 & 0 \end{pmatrix} \quad …(10)$$

Similarly equations for the right eye can be obtained:

$T_r = T_l = N \tan\frac{\theta}{2}$

$B_r = -T_r$

$L_r = -R_l = -c * \frac{N}{f} = -\rho * \frac{H}{2} - \frac{e}{2} * \frac{N}{f}$

$R_r = L_l = b * \frac{N}{f} = \rho * \frac{H}{2} - \frac{e}{2} * \frac{N}{f}$ …(11)

Variables L;R; T, and B are used as the input parameters for the *glFrustum()* function of the right eye.

$$\begin{pmatrix} \frac{2N}{R_r - L_r} & 0 & \frac{R_r + L_r}{R_r - L_r} & 0 \\ 0 & \frac{2N}{T-B} & \frac{T+B}{T-B} & 0 \\ 0 & 0 & -\frac{F+N}{F-N} & -\frac{2F*N}{F-N} \\ 0 & 0 & -1 & 0 \end{pmatrix} \quad …(12)$$

III. Proposed System

The flowchart of figure 3 illustrates the overall system procedures. Firstly, all buffers must be initialized. Two types of buffers were used as shown in figure 4, frame buffer and accumulator buffer. Then GPU starts to calculate stereo pair using off axis method explained in the previous section. The first **red** projected image for left eye is calculated then stored in accumulated buffers, then after calculating second **blue** projected image for right eye it will be added to the earlier one in accumulated buffer resulting the output 3d image that must be seen with a special anaglyph classes. After that, this image is copied to the frame buffer in order to display it via VGA on the external monitor. A block diagram of the designed hardware unit of the overall system is shown in Figure 4, containing a GPU, the refresh controller unit, and video RAM (frame and accumulator buffers) between them.

The system has been implemented using Spartan-3E XC3S500E FPGA kit. The graphics-processing unit (GPU) is responsible for all arithmetic operations of the system explained in the next section, i.e. calculating the stereo pairs, writing resultant pixels in the frame-buffer, controlling handshaking operations between the frame-buffer and accumulator-buffer. These buffers has been designed using dual-port block RAM with a size of 64KB.

Finally, the refresh controller reads the content of the frame-buffer and sends it to the monitor according to a specific time.

Before implementing designed system on FPGA, a pre operation were needed for equations of stereo pairs, which are calculated using MATLAB, because of difficulty of implementation and inclusion within VHDL language, as well as the low memory problem (Limit Block RAM) in Spartan-3E (XC3S500E). So, equations are calculated using MATLAB then resultant values feed to GPU to complete model manipulation using VHDL.The FSM method used to design the proposed GPU, which contains several states as shown in Figure 5, each one handles a specific work.

The first state *clear buffer* of this FSM is used to clear frame buffer then constants for equations explained in section 2 would be calculated in the second state *initial*. While the third state *Model points* forms our object vertices, these vertices will be drawn using next state, which called *drawing object* by using Bresenham's algorithm. The *test* state ensures that the target object drawing is complete if not the *continue* state will give necessary handshaking between buffers to do this work. Finally, the *finish* state will send the object final pixels to the next parts in order to display it on the monitor.

The dual-port block RAM, internally divided into frame buffer and accumulation buffer. The first red image for the left eye is stored on frame buffer as a step to calculate the stereoscopic scene. These points are then send to the accumulation buffer. After calculating, the second blue image for the right eye results will be accumulated the result and send to the refresh controller to display our scene on the monitor.

IV. Implementation Summary and Results

A 3D Stereo Rendering architecture, presented in this work, is successfully tested and results are prove. These results include the performance of the 3D rendering operations based on off axis technique to create stereo pairs and Bresenham's line drawing algorithm[11][12] to draw objects, which have been implemented on FPGA hardware platforms and with OpenGL, results will be shown in this section side by side with their timing analysis, speedup, and performance metrics ...etc.

All algorithms used in this work have been recently tested first using OpenGL on a personal computer with the following specifications: CPU frequency 2.20 GHz, Intel Core i5. Table 1 below shows the components that were occupied by the designed system on FPGA such as the number of block RAM, Flop Flops and other internal components.

Samples from models used in this work using both OpenGL and FPGA are shown in figures 6 and 7, while the overall system is shown in figure 8.

V. Conclusions

The main objective of this work has been achieved with the following points: a processing unit was created to construct an arithmetic system of equations for the stereoscopic vision system and by focusing on the off-axis method in the projection process for each eye; a frame buffer and accumulator buffer are set up to store the projections image on it and then displayed on the screen; The model is plotted using 2D Bresenham's line generation algorithm is that uses integer calculations to avoid the floating point that occupies a large area of an FPGA and slows down the speed. processing accuracy is directly relative to the number of bits used to represent each vertex, but this will negatively affect on FPGA area and occupation resources.

The hardware platform (Spartan-3E XC3S500E FPGA kit) have been successfully used in this work to build the present design for 3D rendering system, Only two colors, have been needed one per projection image for the frame store (256 x 256) and limited number of models were used due to the limited number of BRAMs in the FPGA used. The execution time is 1266 faster than OpenGL time with maximum operating frequency of 35.417 MHz, while the max occupation area reaches 85%.

References

[1] Fore June, ***"An Introduction to 3D Computer Graphics, Stereoscopic Image, and Animation in OpenGL and C/C++"***, 1st Edition, Create Space, a DBA of On-Demand Publishing, LLC (November 2011).

[2] Kasim Terzić , MilesHansard**, *" Methods for reducing visual discomfort in stereoscopic 3D: A review ",*** by Elsevier; Signal Processing: Image Communication 47(2016)402–416.

[3] Chelsea Sabo, Robert Chisholm, Adam Petterson, and Alex Cope**, *"A lightweight, inexpensive robotic system for insect vision"***, by Elsevier; Arthropod Structure & Development 46 (2017) 689-702.

[4] M. Lancelet, T. Martin, B. Solenthaler and M. Gross , ***"Anaglyph Caustics with Motion Parallax"*** , Pacific Graphics, Volume 35 (2016), Number 7.

[5] Oskar Thulin, ***"Intermediate View Interpolation of Stereoscopic Images for 3D-Display"***, MSc. thesis, Department of Electrical Engineering, Linköpings university, Sweden,2006.

[6] S. Jin, J. Cho, X. D. Pham, K. M. Lee, ***"FPGA Design and Implementation of a Real-Time Stereo***

Vision System", IEEE Transactions on Circuits and Systems for Video Technology, vol. 20, no. 1, january 2010.

[7] Akhil Valsaraj, Abdul Barik, Vishak P. V., and Midhun K. M.,***"Stereo Vision System implemented on FPGA***", by Elsevier; ICETEST2015, Procedia Technology 24 (2016) 1105 – 1112.

[8] Akhil Valsaraj, Abdul Barik, Vishak P. V., ***" Stereo Vision System implemented on FPGA"***, International Conference on Emerging Trends in Engineering, Science and Technology (Elsevier, ICETEST- 2015)

[9] Te-Chi Hsiao, Chin-Jung Yang***, " Graphics Processing Method For Three-Dimensional Images Applied to First Buffer for Storing Right-View Contents and Second Buffer for Storing Left-View Contents and Related Graphics Processing apparatus Thereof"*** , United States Patent, US 9,558,531 B2,Jan. 31, 2017

[10] Paul Bourke, ***"Calculating Stereo Pairs",*** Technical Notes, http://paulbourke.net/stereographics/stereorender/# Jan 2018

[11] Ne'am Salim ***," Modified Z-Buffer Design and Its FPGA Implementation "***, Mtech. Thesis, Technical College / Mosul,2013

[12] Dr. Basma Mohammed Kamal Younis, Ne'am Salim Mohammed Sheet ***," Hardware Implementation of 3D-Bresenham's Algorithm Using FPGA",*** Tikrit Journal of Engineering Sciences/Vol.20/No.2/March 2013, (37-47)

Table 1: Device Utilization Summary

Logic Utilization	***Used***	***Available***	***Utilization***
Number of occupied Slices	***3,941***	***4,656***	***84%***
Number of Slice Flip Flops	***1,937***	***9,312***	***20%***
Number of 4 input LUTs	***6,112***	***9,312***	***65%***
Number of bonded IOBs	***21***	***232***	***9%***
Number of RAMB16s	***15***	***20***	***75%***
Number of BUFGMUXs	***2***	***24***	***8%***
Maximum Operating Frequency	***35.417MHz***		

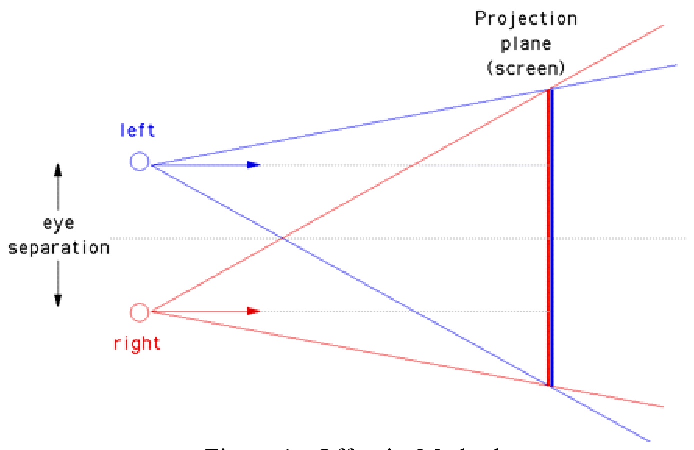
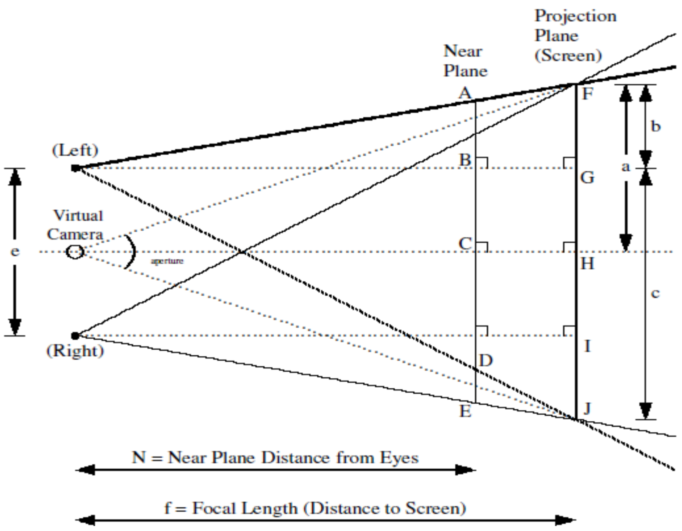
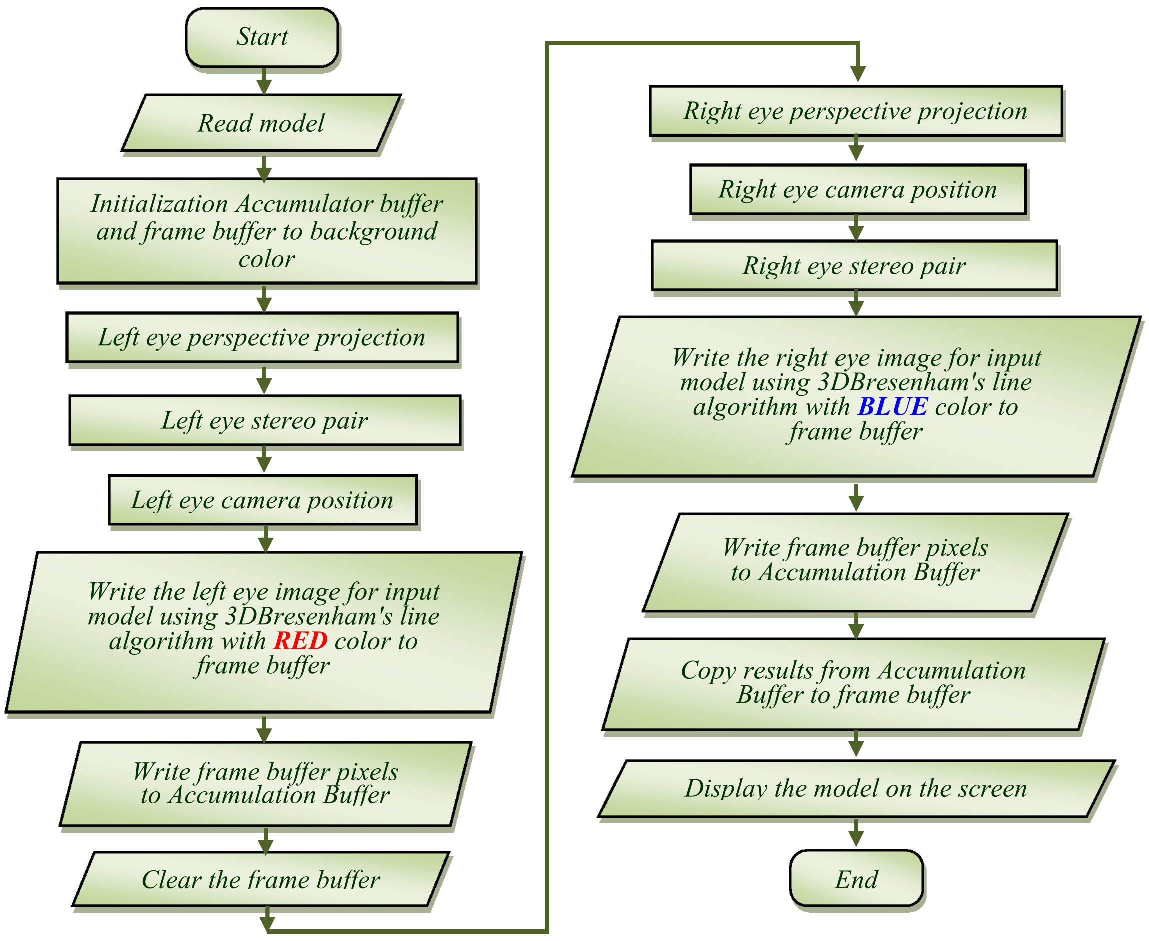


Figure 1: Off-axis Method


Projection Plane (Screen)
Near Plane
A
F
b
(Left)
B
G
a
Virtual Camera
e
C
H
aperture
c
(Right)
I
D
E
J
N = Near Plane Distance from Eyes
f = Focal Length (Distance to Screen)
Figure 2 Off-axis Projection Calculations


Start
Read model
Initialization Accumulator buffer and frame buffer to background color
Left eye perspective projection
Left eye stereo pair
Left eye camera position
Write the left eye image for input model using 3DBresenham's line algorithm with RED color to frame buffer
Write frame buffer pixels to Accumulation Buffer
Clear the frame buffer
Right eye perspective projection
Right eye camera position
Right eye stereo pair
Write the right eye image for input model using 3DBresenham's line algorithm with BLUE color to frame buffer
Write frame buffer pixels to Accumulation Buffer
Copy results from Accumulation Buffer to frame buffer
Display the model on the screen
End
Figure 3: Overall System flowchart

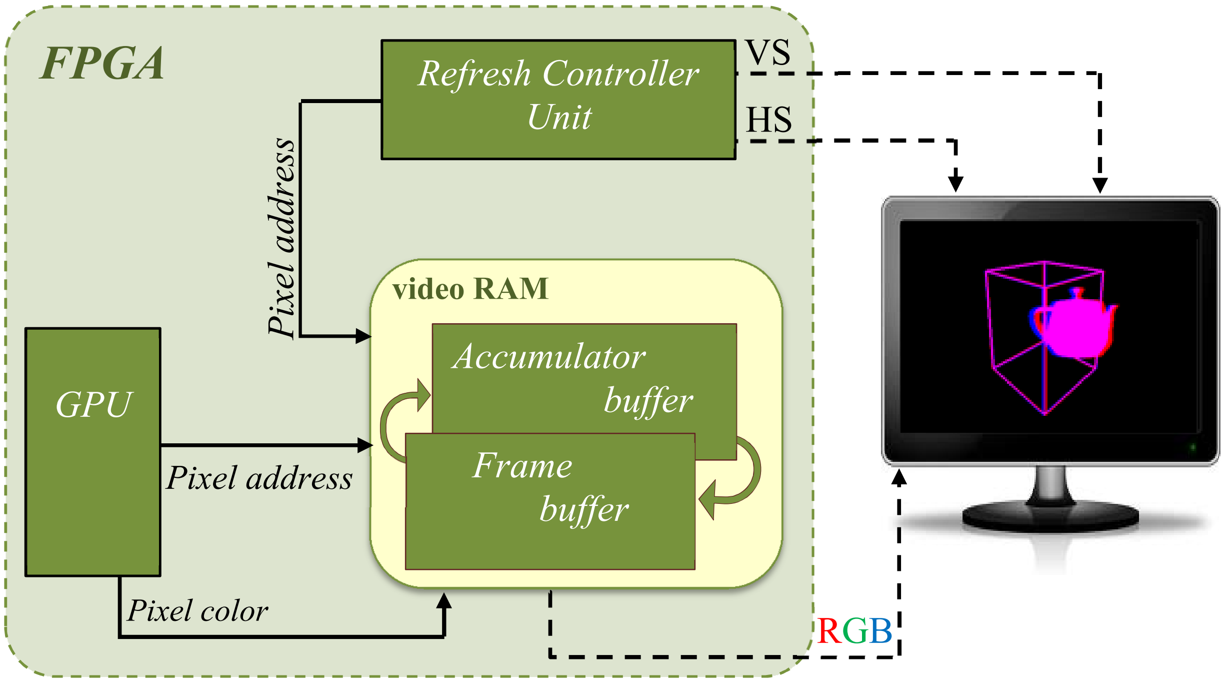


Figure 4: The Designed Hardware Graphics System

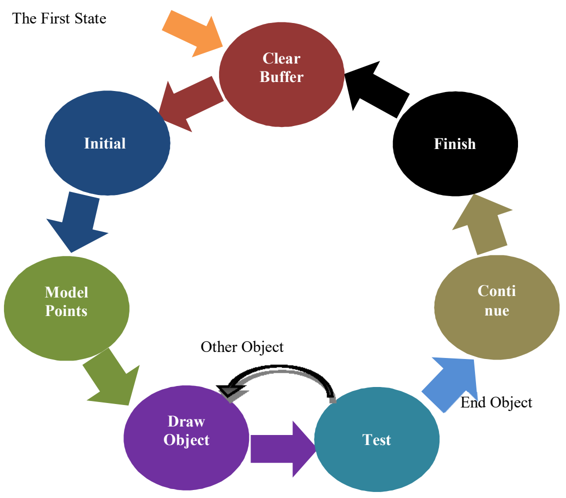


Figure 5: Graphics Processor Unit States

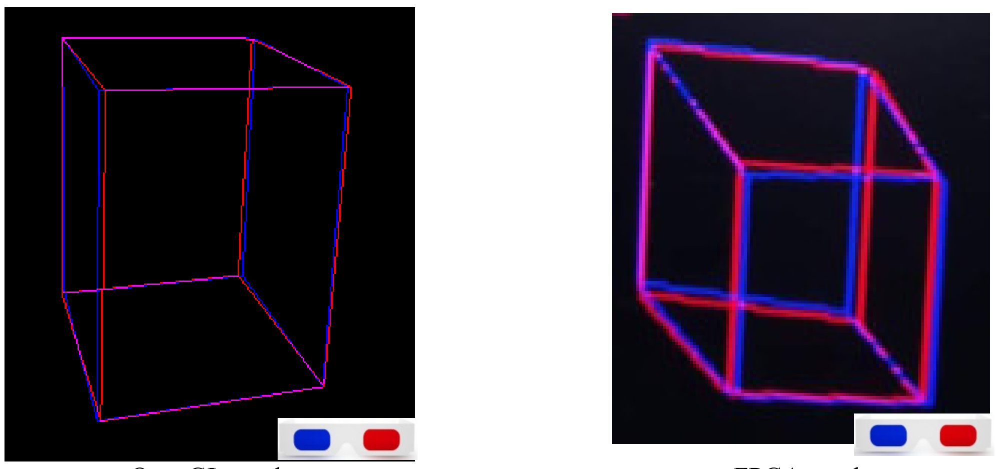


Figure 6: OpenGL vs. FPGA result.

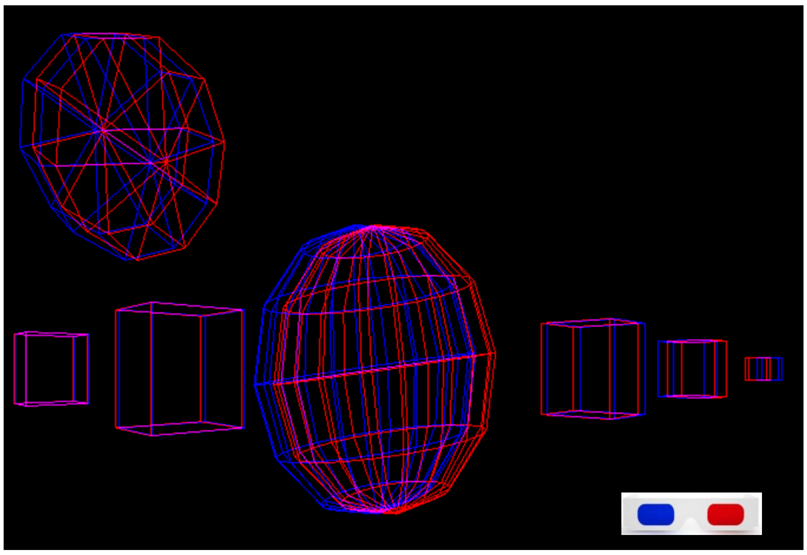


Figure 7: Sample Scene from OpenGL Result.

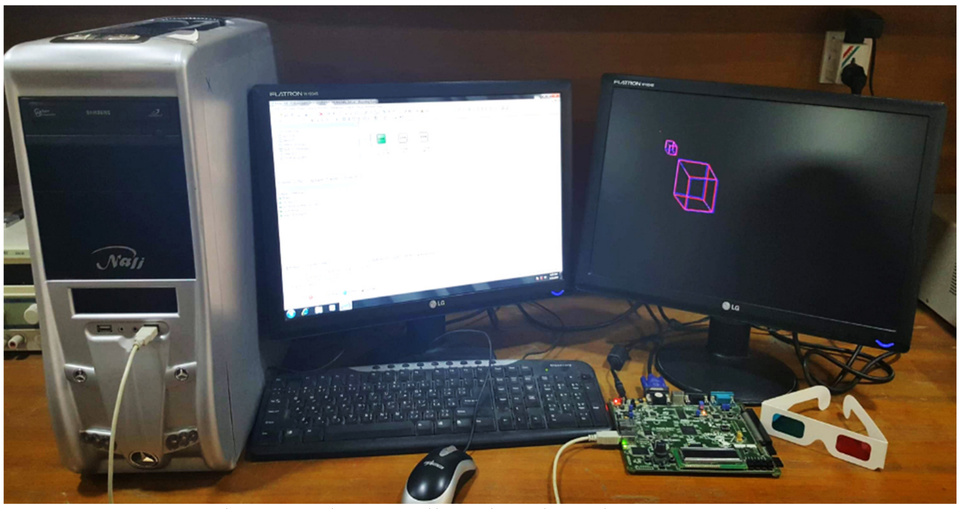


Figure 8: The Overall Desinged Hardware System.