Fpga Design And Implementation

Of A Scan Conversion Graphical Sub-System

Fakhraldeen H. Ali **Amar I. Dawod**

Department Of Computer Engineering

University of Mosul

Abstract

One Major modeling primitive in the field of Computer Graphics is a planar polygon. This polygon can have an arbitrary number of vertices and different shapes. In this paper a graphic sub-system is designed and implemented using Field Programmable Gate Array (FPGA). One of the main tasks of the hardware designed is scan-converting convex planar polygons required to update an image in the image memory or video RAM which is used as a Frame Buffer. A facility to read the pixels (Picture Elements), from the frame buffer, for display on the monitor of the computer is also included in the design.

Keywords: frame buffer, scan-conversion, polygons, pixels, FPGA

تنفيذ وتصميم وحدة تحويل مسح كمنظومة رسم فرعية

باستخدام البوابات المبرمجة حقليا

فخر الدين حامد علي **عمار ادريس داؤد**

قسم هندسة الحاسبات

جامعة الموصل

الخلاصة:

إن احد أهم الأشكال الأساسية في مجال الرسم باستخدام الحاسوب هو المضلع المستوي. هذا المضلّع يُمكن أن يَأخُذ عدداً عشوائياً من الرؤوس وأشكال مختلفة. في هذا البحث تم تصميم و تنفيذ منظومة رسم فرعية باستخدام البوابات القابلة للبرمجة حقلياً. إحدى المهامِ الرئيسية للكيان المادي المُنفذ هو تحويل المسح للمضلعات المستوية و خزنها في ذاكرة الصورة من اجل إجراء عملية التحديث. كما تضمن التصميم إمكانية القراءة للنقاط الصورية (عناصر الصورة) من ذاكرة الصورة وعرضها على شاشة الحاسوب.

Received 19 April 2007 **Accepted 10 Nov. 2007**

1- Introduction:

One of the main effective means for communication is the picture. Computer graphic is an exiting growing field for visual communication. Nowadays computer graphic finds roots in diverse areas of application such as education, research, medicine, training, business, advertisement, and entertainment. Some of the most sophisticated computer graphics systems are those being used for producing video images in real time. This requires downloading repeatable tasks on the hardware [1][2].

In raster graphic, the operation of creating an image in the frame buffer memory is termed scan conversion. After transformation, the projected visible parts of a three dimensional (3D) object are scan converted. The scan conversion produces pixels which are stored in the frame buffer from which they are then taken, using read cycles, for display [3]. Scan converting of an image is simply accomplished by modifying the intensity of all corresponding pixels in the frame buffer memory. This requires access to the frame buffer a huge number of times which makes it essential to adopt a fast scan conversion algorithm for real time graphic systems [1]. The frame buffer is a RAM memory which can be represented as a rectangle matrix of pixels in which two dimensional images are stored. Each pixel consists of a fixed number of bits defining the color resolution of the system. These pixels are normally displayed by a raster technique where the information is fed to the screen as a series of horizontal lines [2]. A basic raster display system contains a frame buffer memory, a graphic controller, a refresh controller, and a monitor (refer to Fig (3)) .

The refresh controller access the frame buffer periodically to obtain the data necessary to refresh the monitor and display the image stored in the frame buffer. The graphic controller accesses the frame buffer to update the image. The basic operation of the graphic controller is scan conversion of the image into a set of pixel intensity values for storage in the frame buffer [4]. The tradeoff between the access of the refresh controller and the access of the graphic controller is a key idea for the architecture of many graphic systems [5]. The current design, as shown in Fig (3), overcomes this problem by using dual port frame buffer memory.

The main function of the scan conversion unit is to resolve each polygon (which is a planar face) into its constituent pixels and store them into the frame buffer memory. This unit receives a high level description list of the polygons, called display list. The scan conversion unit needs to clip these polygons to the required screen while they are being scan converted. This can be achieved by a hardware clipper [6][7].

The hardware clipper is a two dimensional automatic clipper which operates at a fast speed compared to any software clipper. A rectangular clipping screen or window can be defined by four clipping registers where each register is loaded with a boundary value it represents. Each polygon is decomposed into horizontal lines when it is scan converted and each line is tested against the screen borders. These portions of horizontal lines which are inside the screen are drawn and those which are outside the screen are discarded. On the other hand, horizontal lines which are above or below the clipping window are entirely eliminated.

After introducing scan conversion a review of some related published works is thought useful. In 1987 the researcher Roman P.Molla designed three systems with different algorithms to implement a scan conversion unit for a straight line segment using serial processing and parallel processing. The paper discussed the performance , cost and the error ratio for the three designed systems [8]. In 1993 Andreas Schilling and Wolfgan Straber introduced an algorithm that deals with hidden surface elimination problem at pixel level. The hardware implementation was divided into three stages, for pipelining, to improve the performance. The architecture designed used 12000 gates and the performance is claimed to be 20 M pixel/sec [9]. In 1994, Molnar et al has introduced a classification of different architectures that implement the scan conversion operation using parallel processing depending on the basic stages of image generation. These stages are fragmentation stage in which the scene is divided into a group of small parts to implement the scan conversion on them later, the assignment stage where parts of the scene are allocated to the parallel processing units and finally defragmentation stage , where the partial results are collected and then stored in the frame buffer[10]. In 1996 C. Scott Ananian and Greg Humphreys suggested three different architectures to implement ray tracing algorithm. The designed hardware includes two units. The first unit is responsible for the implementation of the scan conversion operation, the second unit is a ray casting unit. The paper discussed the performance and cost for the three designed architectures [11].

In 2004, David Harris discussed the performance of the OpenGL lighting unit which is responsible for light simulation and brightness. The paper introduced a hardware implementation for using integer mathematics and the architecture consisted of multipliers and look up tables[12]. In 2005 a group of researchers (Praveen Bhaniramka, et al) working in Silicon Graphic Company, introduced a real time graphic system which is mainly managed by the OpenGL library. The system is splitted into four parallel units each operates as a distinct part to generate the desired scene. In addition to that, the pipeline technique is implemented in each unit. However, the paper discussed the performance of the library in real time graphic systems[13].

2- Scan Conversion algorithm (for a planar polygon):

The calculations performed in scan conversion take advantage of various coherence properties of a scene that is to be displayed. What we mean by coherence is simply that the properties of one part of a scene are related in some way to other parts of the scene so that the relationship can be used to reduce processing. This involves incremental calculations applied along a single scan line or between successive scan lines. In determining edge intersection, we can set up incremental coordinate calculations along any edge exploiting the fact that the slope of the edge is constant [7][14]. Figure 1 shows two successive scan lines crossing the borders of a polygon. The slope of each polygon boundary straight line segment can be expressed in terms of scan line intersections:

$$M = (Y_{k+1} - Y_k)/(X_{k+1} - X_k) \quad \ldots\ldots. \quad (1)$$

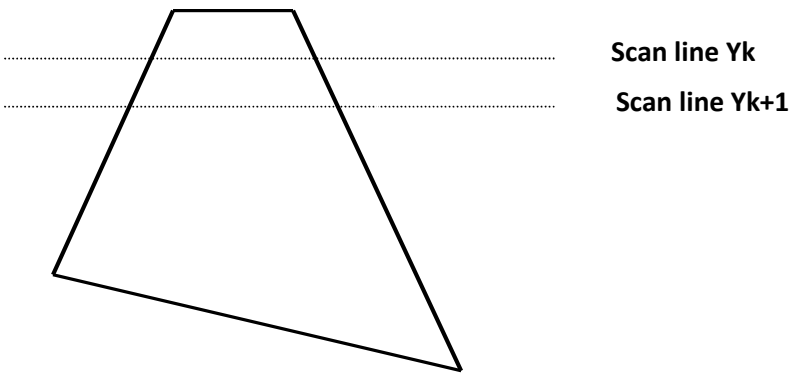


Figure (1) A polygon scan conversion (up to down)

K: is a counter

Since the change of the y coordinate between two successive scan lines is simply :

$$Y_{k+1} - Y_k = 1 \quad \ldots\ldots\ldots (2)$$

The new intersection x value is determined from the X intersection value Xk of the preceding scan line as :

$$X_{k+1} = X_k + 1/M \quad \ldots\ldots (3)$$

Each successive X intercept can thus be calculated by adding the inverse of the slope and rounding the result to the nearest integer value noting that the slope may be negative or positive depending on which is

greater (Xk+1 or Xk). The increment of X by the amount 1/M along an edge can be accomplished with integer operations by recalling that the slope M is the ratio of two integers (S*dy/dx) where dX and dY are the difference between the edge endpoint X and Y coordinate values and S decides if the increment or decrement operation is required to X coordinate value.

$$dx=x2-x1 \text{ if } x2 > x1 \text{ and } S = 1 \quad \ldots\ldots\ldots(4)$$

$$dx=x1-x2 \text{ if } x1 > x2 \text{ and } S = -1 \quad \ldots\ldots\ldots(5)$$

Thus incremental calculation of X along an edge for successive scan lines can be expressed as:

$$Xk+1 = Xk + S*dx/dy \quad \ldots\ldots\ldots..(6)$$

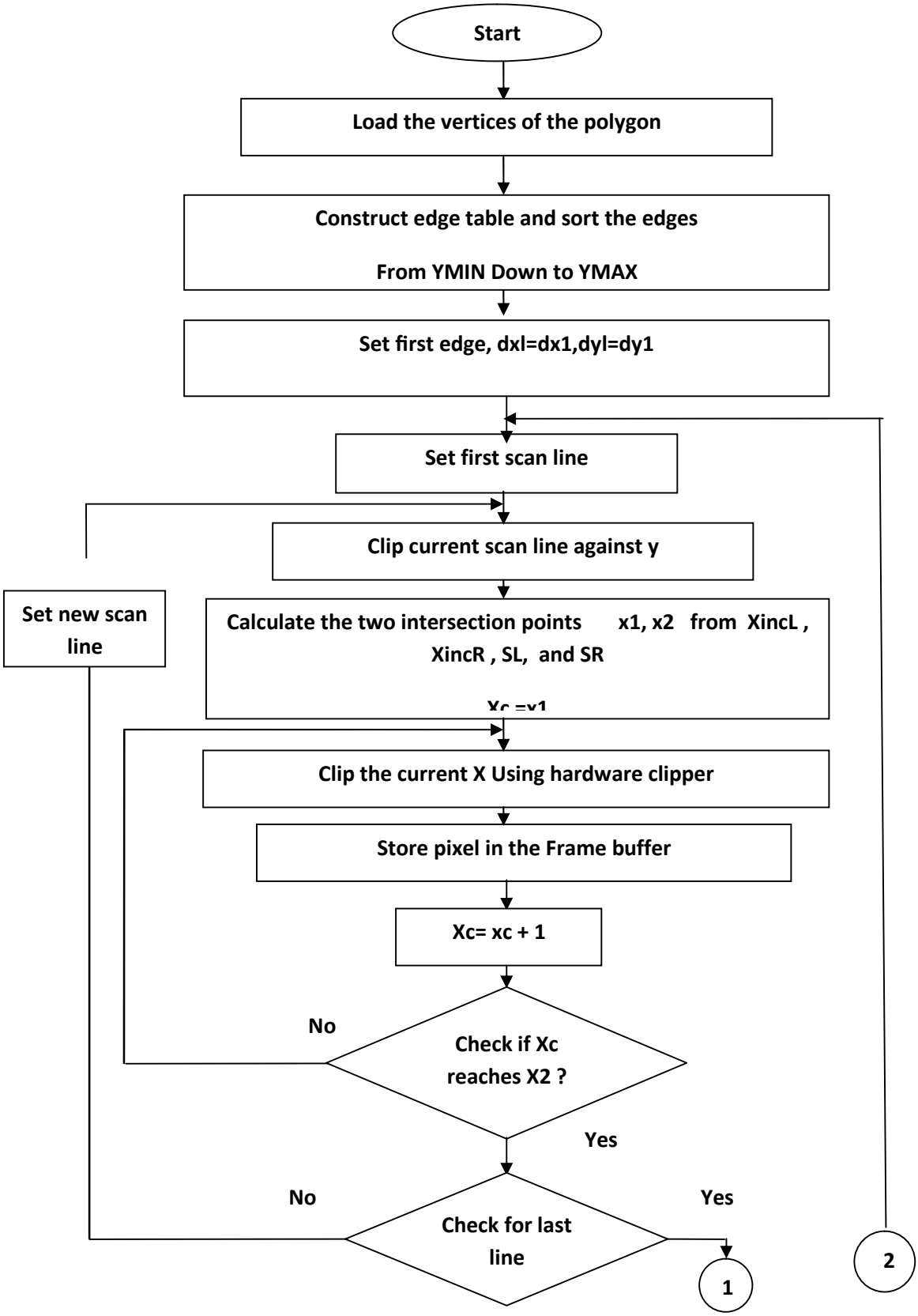
Using this equation, we can perform integer evaluation of the X intercept by initializing a counter to 0, then incrementing the counter by the value of dx each time we move to a new scan line. Whenever the counter value becomes equal to or greater than dy, we increment (or decrement depending on the sign of S) the current X intersection value by (1) and decrease the counter by the value dy. This procedure is equivalent to maintaining integer and fractional parts for X intercept and incrementing the fractional part until we reach the next integer value [6].

The algorithm should begin by ordering the polygon sides on their Y value. It starts with the smallest Y value and scan down the polygon, and should construct edges table for storing the slope of each edge [7].

Figure 2 illustrates a flowchart for the implemented scan- conversion algorithm.

3- Hardware design :

A block diagram of the designed graphical unit is illustrated in figure (3). The graphic controller is interfaced to both the display list memory and to the frame buffer. The graphic controller reads a display list of polygons which are scan converted and the pixels outcome are stored into the frame buffer. The whole image is updated when all its visible polygons are scan converted.


Start
Load the vertices of the polygon
Construct edge table and sort the edges
From YMIN Down to YMAX
Set first edge, dxl=dx1,dyl=dy1
Set first scan line
Clip current scan line against y
Calculate the two intersection points x1, x2 from XincL , XincR , SL, and SR
Xc =x1
Clip the current X Using hardware clipper
Store pixel in the Frame buffer
Xc= xc + 1
Check if Xc reaches X2 ?
No
Yes
Check for last line
No
Yes
Set new scan line
1
2

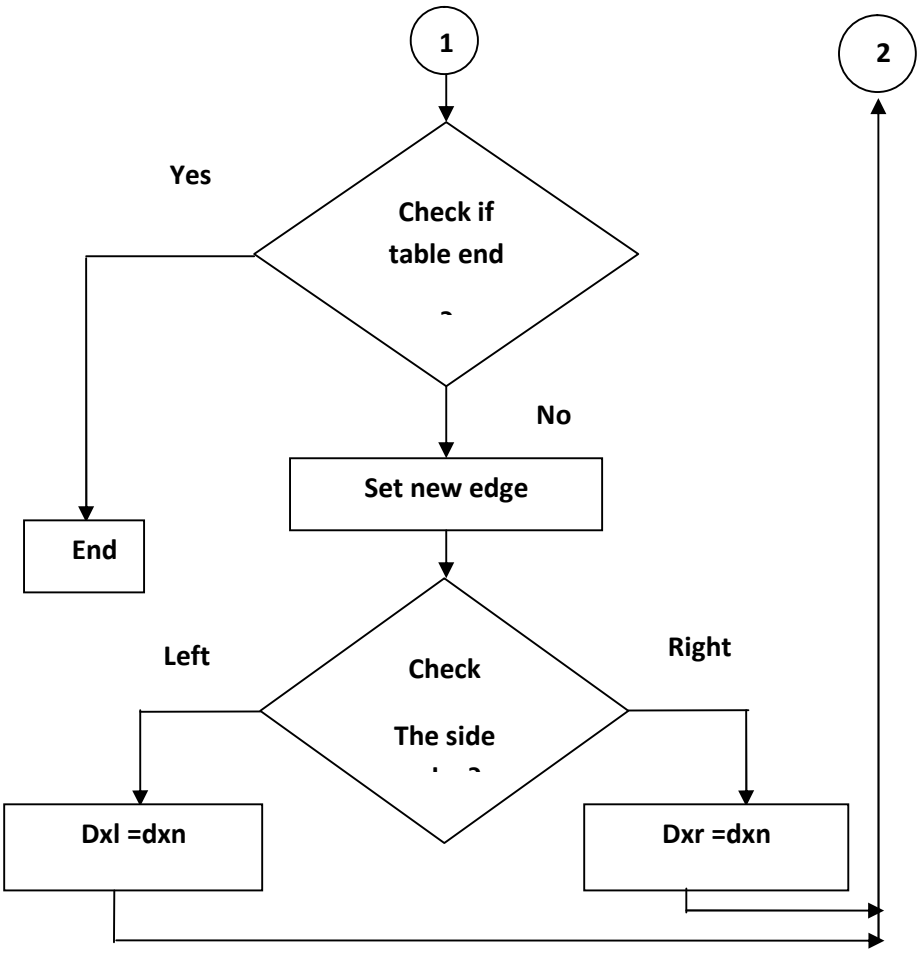


Figure (2) Scan-conversion algorithm

Where : dxl & dyl for the left side, dxr & dyr for the right side , Xc is moving x , from x1 to x2, Yc is current y.

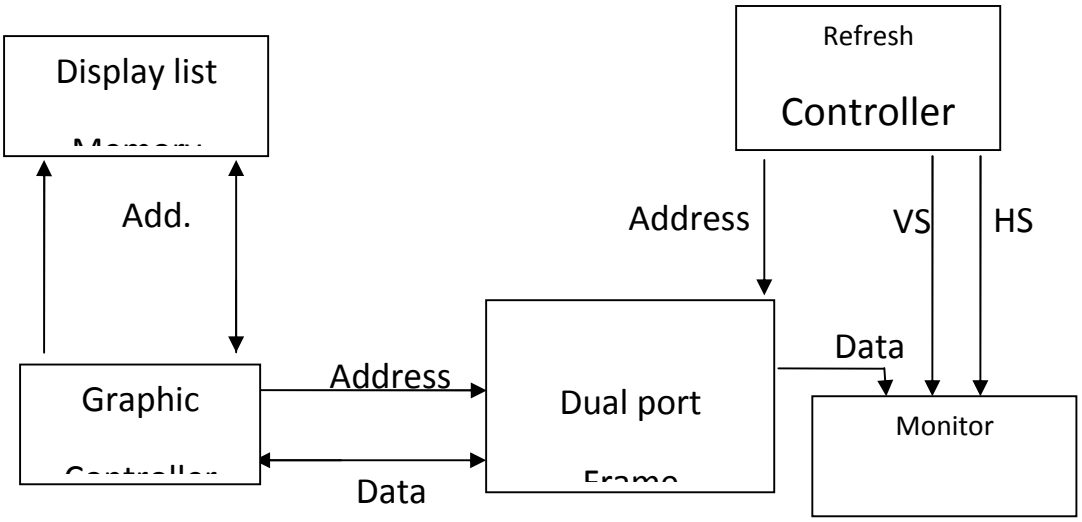


Fig (3) Hardware graphic sub-system

The arithmetic section of the implemented graphic controller searches the polygon vertices to determine the smallest y coordinate and compute the slope of each polygon edge connecting this vertex to the preceding and succeeding ones. From these slopes the graphic controller determines the intersections of each scan line with the polygon edges which in turn define the beginning and end of each span. The polygon border calculator passes the information of each span to the hardware clipper, the hardware clipper clips each span to the screen using clipping registers.

The graphic controller computes the corresponding address value of the frame buffer using x, and y coordinates. Figure (4) shows a block diagram of the designed graphic controller.

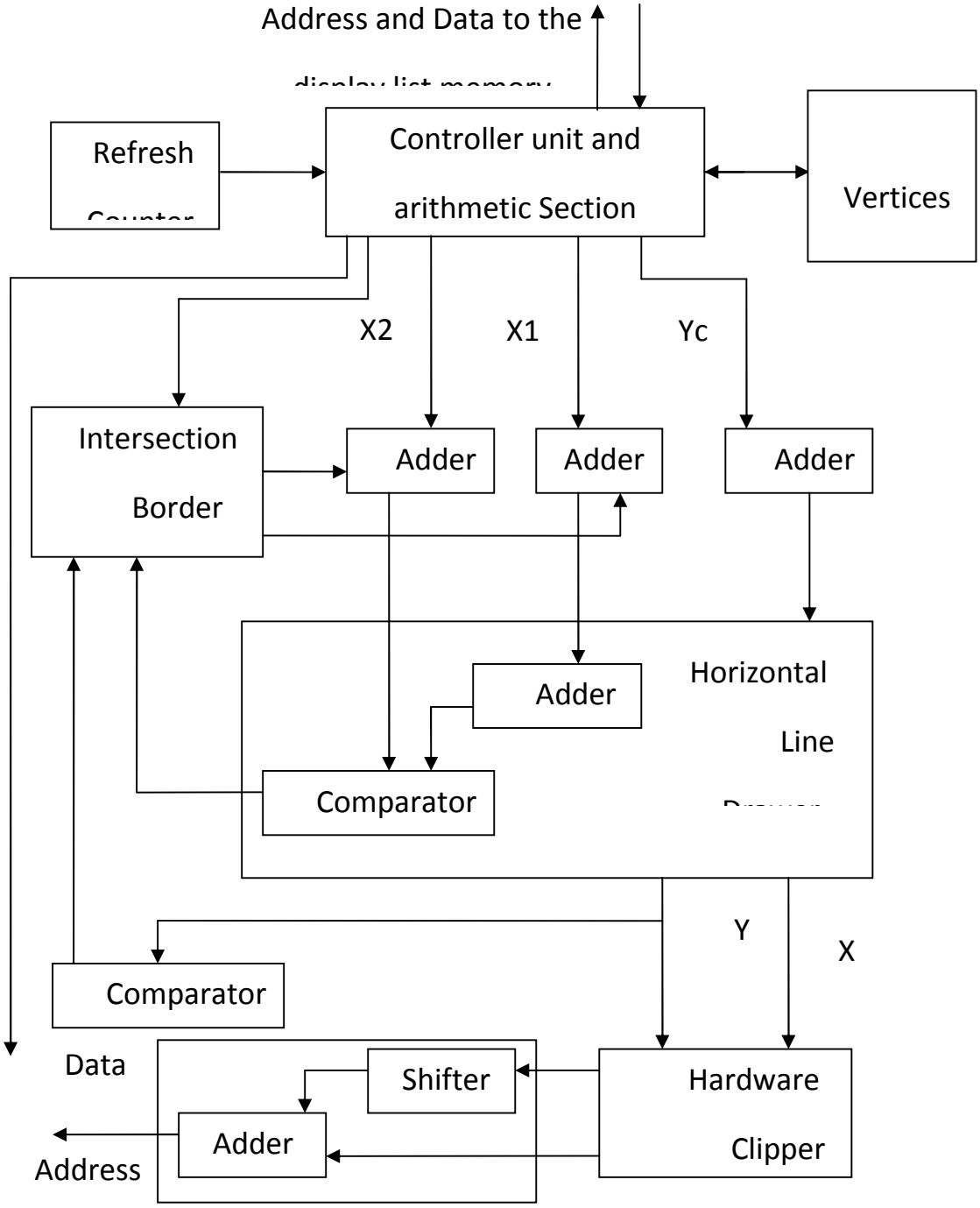


Fig. (4) The designed graphic controller

The refresh controller generates the necessary address and control signals for interfacing the frame buffer to a raster monitor. It generates vertical synch and horizontal synch to scan the monitor, in synchronism with the address, to access the frame buffer. So pixels are read from the frame buffer and applied to the Monitor. The address is synchronized with V synch and H synch, to ensure that the pixel is applied to the

electron gun of the monitor at the correct time [4]. Figure 5 shows a block diagram of the designed refresh controller.

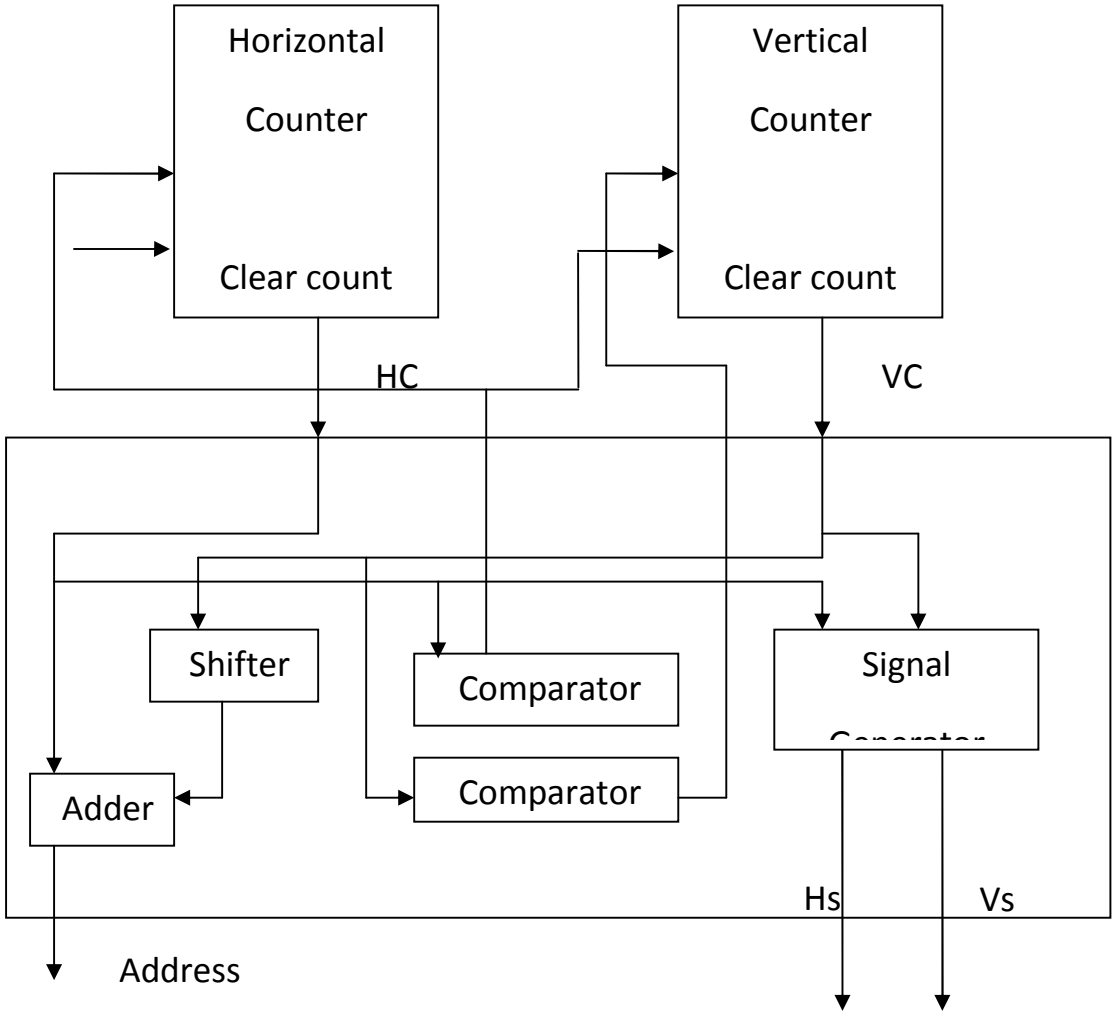
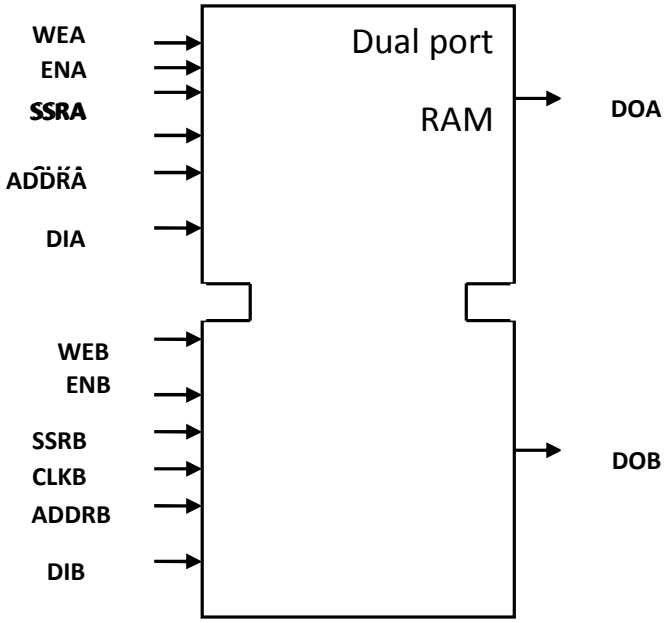


Fig (5) Refresh controller hardware

A dual port RAM (or Frame Buffer) provides two sets of data ports one set is used for reading and the second for writing (refer to Fig 3 and Fig 6). The refresh controller uses one for reading data to refresh the monitor, and the graphic controller uses the other port for writing to update the image in the frame buffer[15].

Each port can be used independent of the other while accessing the data memory cells. Each port is fully synchronized with an independent clock. All input pins of port A have setup time referenced to the CLKA pin and its data output bus DOA is time referenced to the CLKA. All

input pins of the port B have setup time referenced to the CLKB pin and its data output bus DOB is time referenced to the CLKB [15]. Table 1 shows the logic of the control signals and mode of operation.



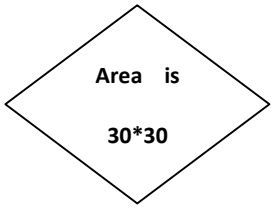
Fig(6) Dual port RAM

CLK (A or B)	EN	SSR	WEA/ WEB	DOA/ DOB	Operation
↑	0	X	X	No Change	No operation
↑	1	1	X	Set	Set
↑	1	0	0	Change	Read
↑	1	0	1	No Change	Write

Table (1) RAM control logic

4- Performance Results :

The performance speed of the scan conversion unit is affected by the number of visible polygons and their areas in a scene. Many ways can be adopted to specify the speed of the implemented scan conversion unit. One of these methods is computing the speed with which this unit can write pixels into the frame buffer. The Max speed of the scan conversion unit is 50 M pixel per second, i.e the unit takes 20 ns to write one pixel in the frame buffer. Another way, the speed is reported is the number of erasing or clearing the frame buffer (clearing the screen) which means setting each pixel color to the background value. The size of the frame buffer is 64 K pixel, in case 256 * 256 resolution, which is required to be written to clear the frame buffer. So computing this speed for the implemented scan conversion unit gives a value of 763 number of clearing the screen per second. In the third method, the number of polygons scan-converted per second, which is a measure of the image complexity, provides an excellent indication of how well this unit operates. The polygon shown in figure 7 is used to carry out such measurement.



The scan conversion unit converts successive polygons, each is shifted one pixel down and one pixel to the right. The designed scan conversion unit is able to scan convert (54,945) polygons per second. Table 2 below, shows the parameters values of the designed and implemented scan conversion unit.

Fig (7)

Measurement polygon

Samples of the hardware performance waveforms are given in figure(8) and figure(9). As shown in figure(8) the input of two line end points are initialized and then the Graphic Controller computes the difference between the edge endpoint coordinate values(dx , dy), and calculates the error to evaluate the change of the y coordinate and x coordinate at each step. In figure (9) the inputs are the vertices of a polygon where the Graphic Controller arranges the data input to determine the first scan line and then calculates the span of current scan line (Xmin , Xmax) to fill it with pixels.

Frame buffer: designed used	 640*480 pixels 256*256 pixels
Pixel frequency	25 MHZ
Horizontal display	0.0256 ms
Horizontal retrace	0.0064 ms
Vertical display	15.36 ms
Vertical retrace	1.312 ms
Scan conversion speed	50 M pixel per second
No. of clearing the creen	763 times/sec
Number of polygons (30 x 30 pixels)	54,945 polygons/sec

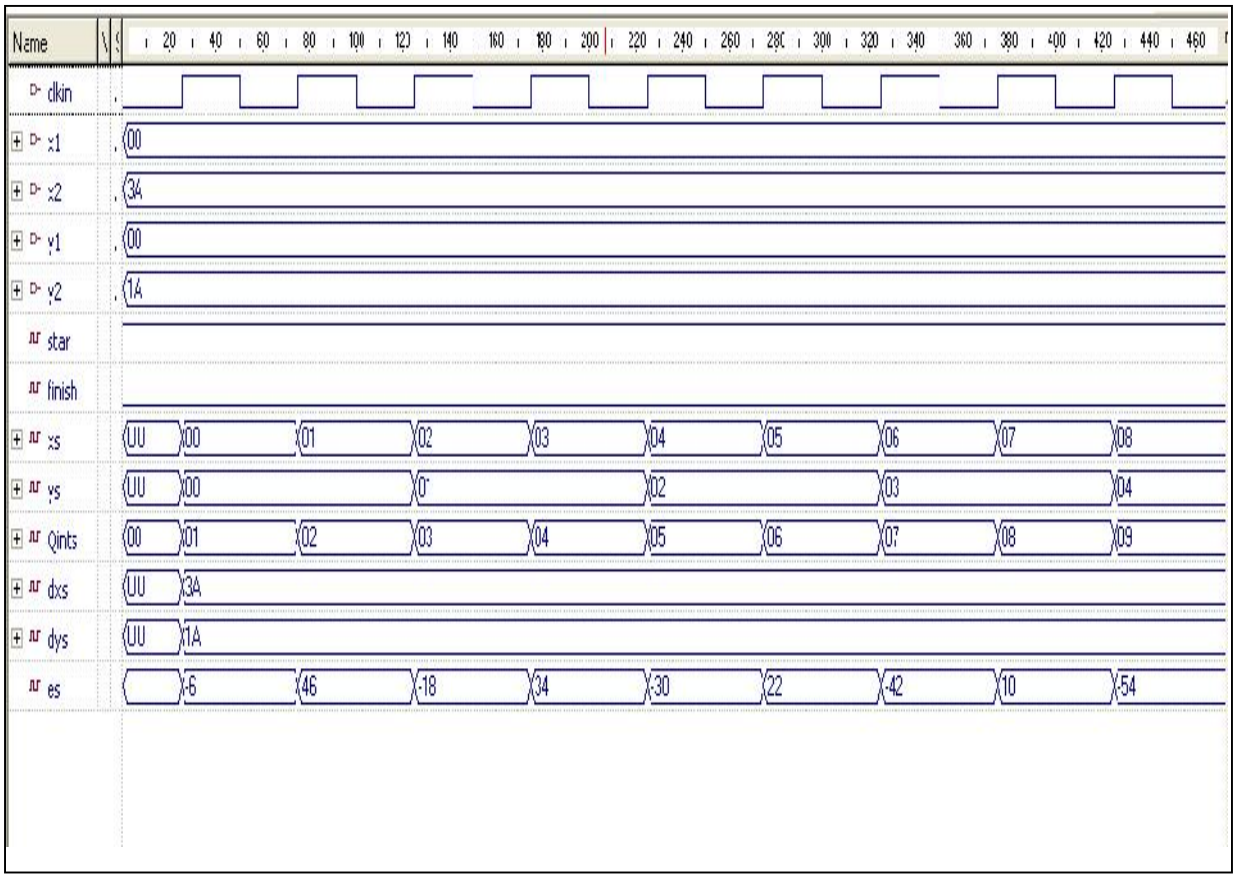
Table (2) Parameters value

5- Conclusions:

1- A scan conversion algorithm is designed in a way suitable to be implemented in hardware.

2- The hardware designed consists of two main sections, one for updating the image, and the other for refreshing the monitor. The transfer of graphical data from the first part to the second is accomplished (asynchronously) through the frame buffer memory which is dual ported to increase the data speed.

3- To correlate the calculated speed of the scan conversion unit which is 50 M pixel/sec with the measured speed (54,945) polygons/sec (each has 30x30 pixels), the total number of pixel is calculated for the second and compared with the first. Such calculations show that the speed 50 M pixel/sec is reduced to 49.3 M pixel/sec because the scan conversion unit losses some available write cycle due to the time required to perform some internal processes before producing data information at its output.



Fig(8) Sample waveforms for a line generation.

4- A simple and fast clipper, being implemented in the software, replaces a complex and slow software clipper. However, the scope of its performance is bounded by the capacity of the clipping registers.

5- The performance speed can be further improved using higher frequency version of FPGA.

6- The resolution is limited by the available RAM and can also be improved using FPGA version with higher capacity of internal RAM.

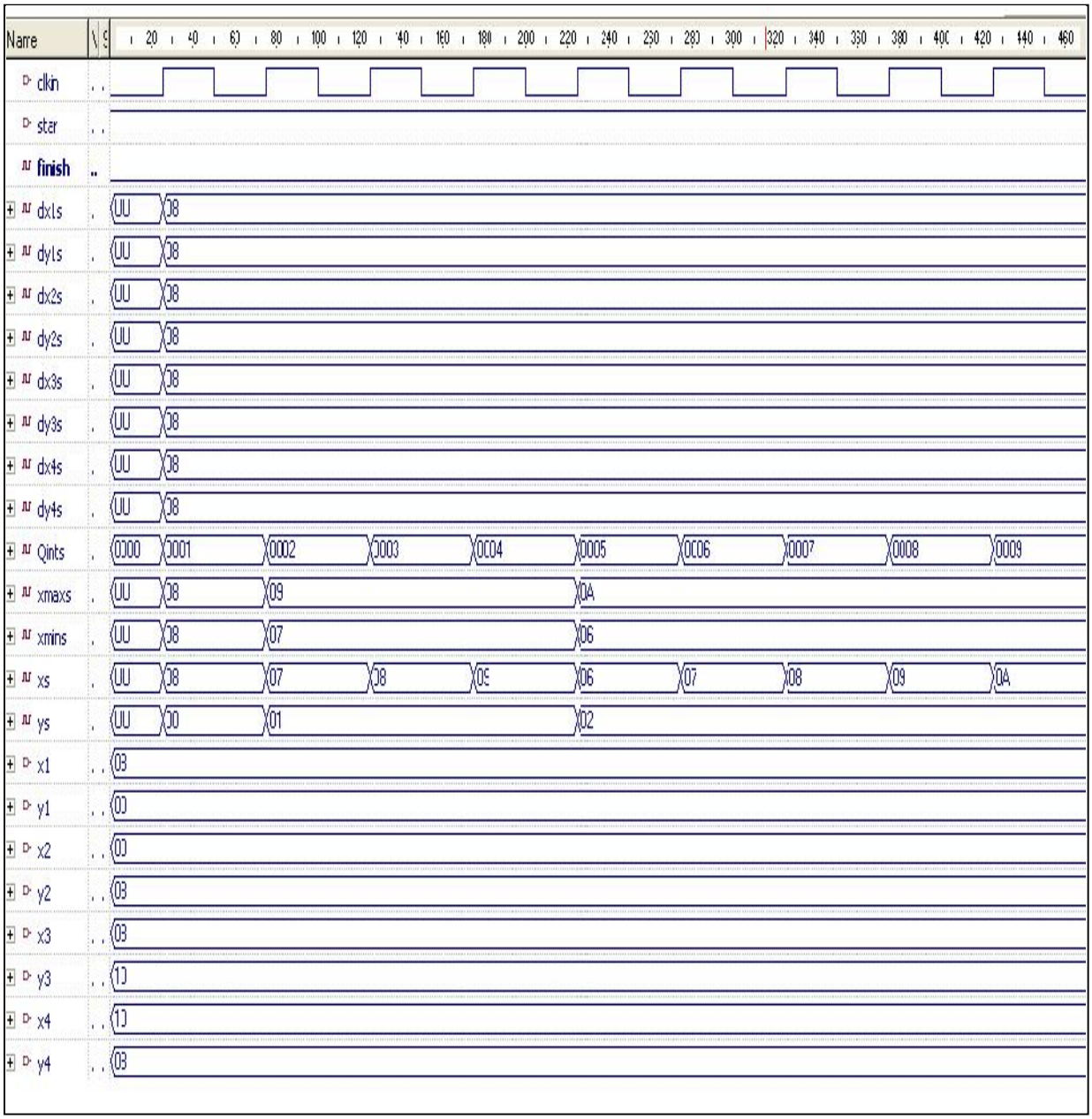


Fig (9) Sample waveforms for a polygon generation

References:

[1] Basma Mohammed-Kamal , " Depth –Buffer for 3D Graphics ", B.Mc. ,Electronic and Communication Dept of Electrical Engineering College ,Mosul University,1999.

[2] Stefan Schlechtweg,),"Non-Photorealistic Computer Graphics Modeling, Rendering and Animation" ,assistant professor at the University of Magdeburg (Germany ,(2005).

[3] Jones Gomes " image processing for computer graphic",

(1999), Prentice Hall International , Inc,ch7.

[4] Edward Angel , " Interactive Computer Graphic , A Top- Down Approach Using

OpenGL " Third Edition 2003,ch1,3.

[5] Jeremy W. Sheaffer1,Kevin Skadron ," Characterizing Architectural Vulnerability ardware", Dept. of Computer Science, University of Virginia (2006).

[6] Donald Hearn and M. Pauline, "Computer graphic C version"

Third edition, (1997), Prentice Hall International , Inc,ch3.

[7] F.S. Hill, Jr. "computer graphic using OpenGL " second

edition , (2001), Prentice Hall International , Inc,ch2.

[8] Roman P.Molla Vaya , "Parallel Fixed Point Digital Differential Analyzer " , Computer Journal , Vol. 23, No. 1, pp:46-52, 1987.

[9] Andreas Schilling , Wolfgang Straber , "EXACT: Algorithm and Hardware Architecture for an Improved A-Buffer", Computer Graphics, vol. 27, no. 4, August 1993 (SIGGRAPH '93 Proceedings), pp: 85–92.

[10] S. Molnar , M. Cox , D. Ellsworth and H. Fuchs , " A Sorting Classification of Parallel Rendering " , IEEE Computer Graphics And Algorithms , pages 23-32, July 1994.

[11] C. Scott Ananian, Greg Humphreys "A Hardware Accelerated Ray-tracing Engine" , Princeton University, Computer Graphics (SIGGRAPH '96 Proceedings), volume 21, pp: 95-102, 1996.

[12] David Harris, "An Exponentiation Unit for an OpenGL Lighting Engine", Member, IEEE , IEEE Transactions on Computers , VOL. 53, NO. 3, March 2004, pp: 251-256.

[13] Praveen Bhaniramka , Philippe C.D. Robert , Stefan Eilemann," OpenGL Multipipe SDK: A Toolkit for Scalable Parallel Rendering", IEEE Symposium on Volume Visualization and Graphics, PP: 119-126, 2005.

[14] David F. Rogers " Mathematical element for computer Graphic" (1997) McGraw-Hill Inc, ch3.

[15] User Manual Spartan-3 FPGA Family: Complete data Sheet, DS099 March 4, 2004.

The work was carried out at the college of Engg. University of Mosul