Raf. J. of Comp. & Math's., Vol. 15, No. 2, 2021
DOI:

Bresenham's Line and Circle Drawing Algorithm using FPGA

Areej H. Ali
areej.csp75@student.uomosul.edu.iq

Riyadh Z. Mahmood
riyadh.zaghlool@uomosul.edu.iq

Department of Computer Science
College of Computer Science and Mathematics,
University of Mosul, Mosul, IRAQ

Received: 15/11/2020 **Accepted: 10/01/2021**

ABSTRACT

In Bresenham's line drawing algorithm, the points of an n-dimensional raster that have to be selected are determined forming a close approximation to a straight line existed between two points. It is widely used for drawing line primitives in a bitmap image (for example: on a computer screen), since only integer addition, subtraction and bit shifting are used. These three operations are cheap concerning standard computer architectures. In addition, it is an incremental error algorithm. It is among the oldest algorithms that have been developed in computer graphics. An extension to the original algorithm may lead to draw circles. This research deals with the Bresenham's line and circle drawing algorithm based on FPGA hardware platform. The shapes on the VGA screen are displayed via internal VGA port that is built in the device.

Keywords: FSM, VHDL, VGA port, LCD.

1. Introduction

Computer graphics deals with manipulating visual and geometric information adopting techniques of computation. It concentrates on basics of mathematic and computation of image generating and processing more than merely aesthetic aspects. It is sometimes differentiated from the visualization field, although both of them have many similarities [1].

Displaying a picture, of any size, on a screen is not an easy task Therefore, computer graphics is used in order to simplify this operation.

In fact, computer graphics is a branch of computer science dealing with images generation using computers. Nowadays, it is a key technology in the field of video games, digital photography, film, computer displays and many other applications. The term computer graphics was used in 1960 by Verne Hudson and William Fetter of Boeing, the researchers of computer graphics. In addition, it is sometimes abbreviated as (CG), or regarding film as (CGI), Computer Generated Imagery [2].

The term computer graphics has many topics to deal with including: design of user interface, sprite graphics, animation, ray tracing, and so on.

Computer graphics has the responsibility to display effective and meaningful art and image data to the consumers. In addition, it is used to process image data coming from the physical world, like photo and video contents [3].

1.1 FPGA

A Field Programmable Gate Array (FPGA) is an integrated circuit designed to be configured by the customer or designer after manufacturing-hence "field-programmable". The FPGA configuration is generally specified using a hardware description language (HDL), similar to that used for an application-specific integrated

circuit (ASIC) (circuit diagrams were previously used to specify the configuration, as they were for ASICs, but this is increasingly rare). FPGAs can be used to implement any logical function that an ASIC could perform. The ability to update the functionality after shipping, partial re-configuration of the portion of the design and the low non-recurring engineering costs relative to an ASIC design (notwithstanding the generally higher unit cost), offer advantages for many applications [4][5].

1.2 literature Review

In 2009, a new quick line drawing algorithm that differs from that of traditional Bresenham's algorithm was presented by Niu Lianqiang and Feng HaiWen. They dealt with a line as an aggregation of many line segments. They also coordinated candidate pixel points differences in each step of traditional algorithm which is replaced by the errors length of every segments in the newly algorithm. Each judgment and operation can create a line segment through keeping the integer arithmetic advantages .Then the numbers of operation and output are reduced. In addition, the character of skew symmetric is pointed out in the algorithm and the property of direct draw without operating of some particular lines is considered as well [6].

In 2011, Chikit Au and Tony Woo, discovered the reasons for little previous works. The mid-point concept within a unit interval was generalized to that of the nearest neighbors that involve a Voronoi diagram. In their work, the extension of the three-dimensional depends on the basic idea of Bresenham's Algorithm of a minimum distance between the grid points and the line. The structure of Voronoi diagram is shown for grid points to which the line might be approximated. They also investigated the deployment of integer arithmetic and three-dimensional extension symmetry of the algorithm to enhance the efficiency of computation [7].

In 2011, an algorithm as a three-dimensional development for the existed two dimensional (DDA),Digital Differential Analyzer, was implemented by using the FPGA that is configurable for the applications of real time, and it was designed by Fakhrulddin Hamid Ali . This algorithm is required to be a solution for the problem of hidden surface in the image space when depth or Z buffer method in 3D field computer graphics is used. This paper came into a conclusion that the unit of hardware could produce pixels at, of 120M pixel speed per second that assumes a very short time is being lost while the increment values is computing[8].

In 2013, a system of 2D graphics using FPGA, which is composed of peripheral IPcores (Bresenham's, DDR Memory Controller, and VGA) a CPU IP core, and PLB bus for which CPU and the entire cores of peripheral IP was designed and implemented by Kahraman Serdar Ay and Atakan Dogan. In addition, some APIs and graphics drivers were developed so as to complete the total graphics creating process. The designed work is divided into two major parts as a design of hardware and a design of software. In the design of hardware, cores of graphics IP, such as Bresenham were designed. Then, interface logic was developed depending on PLB IPIF in a way that the cores of graphics IP could be connected with PLB bus and, eventually, they could be put under control by their particular drivers. Finally, two other peripheral IP cores and a PowerPC CPU core, Bresenham and other graphics IP cores, and are fetched together around a PLB bus in order for completing the design of hardware. Through the connection of a monitor to VGA output, the graphics system was examined. The examples of monitor images rendered totally in Virtex-II FPGA chip on developing board. TheBresenham IP core design needs 2415 slice flip-flops, 2482 slices , and 4383 four-input LUTs with instantiating of PLB IPIF entity.

Furthermore, 1095 slice flip-flops ,1042 slices, , and 1850 four-input LUTs are required. Bresenham IP core could rasterize in 0.1 ms a line of 500 pixels [9].

2. Theoretical Background

2.1 Bresenham's Line Drawing Algorithm

It is used for a line scan conversion. It was also developed by Bresenham and it is considered an effective method since just only integer addition, subtraction, and multiplication are involved. These three operations are performed very quickly in order that lines could be generated rapidly. In this manner, the selected next pixel is that the one having the least distance from the true line. The following shows the work of this method:

Bresenham's algorithm is another algorithm of incremental scan-conversion. Its name comes after Jack Bresenham who worked for 27 years at IBM before his academic years. He developed the algorithm in early years of 1960s at IBM. Its use of only integer calculations is one of the most important advantages of this algorithm. At the time we have a point (xk, yk), then we should determine whether we draw the point (xk+1, yk) or (xk+1,yk+1).It is to be noted that, at any cases, if we move to xk+1, yet we should decide moving to yk oryk+1. On the other hand, the increment of the other variable is decided through examination of the distance, which is known as decision variable or the error, between the location of the actual line and the nearest pixel [10]. See Figure 1.

The algorithm for Bresenham’s Line:

1. Start
2. Read the line end points (x1,y1) and (x2,y2) such that they are not equal.
3. Calculate dx=x2-x1and dy=y2-y1
4. x=x1 and y=y1 [Initialize starting point]
5. e=2*dy-dx [Initialize value of decision variable]
6. i=1
7. plot(x,y)
8. while(e>=0) {
   y=y+1
   e=e -2*dx
   }
   x=x+1
   e=e + 2*dy
9. i=i+1
10. if(i<=dx)then go to step ٧
11. end.

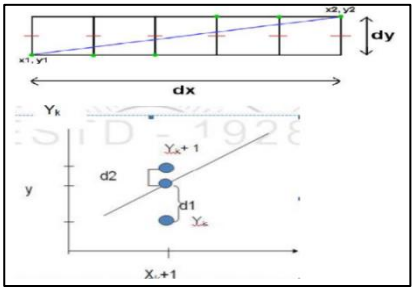


Figure 1: The distance between the location of actual line and nearest pixel

2.2 Bresenham's Circle Drawing Algorithm

It is an algorithm of circle drawing which selects the position of nearest pixel for completing the arc. The special point of the algorithm is that it uses just only integer arithmetic making it, remarkably, quicker than any other algorithms that use floating point arithmetic and classical processors [10].

As circle per Eight way symmetry property, a circle could be divided into eight octants, with each one of 45-degrees. This algorithm calculates the location of pixels of 45 degrees at the first octant and extending it to the rest of other seven octants. The algorithm draws a pixel in every one of the eight octants for each circle of the pixel (x, y), as it is clear below:

Assumption: Centre of circle is origin. Figure 2 shows the 8 octants with the corresponding pixel.

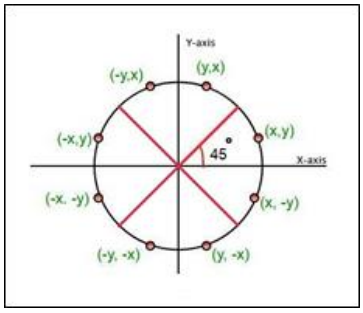


Figure 2: The eight octants with the corresponding pixel

Point (x, y) is on the circle and in the first octant. For calculating the next location of pixel, it could be either

- N (x+1, y) or
- S (x+1, y-1)

It is determined through the use of the decision parameter d as:

- if d <= 0, then N (x+1, y) is to be selected as being next pixel.
- if d > 0,then S (x+1, y-1) is to be selected as being next pixel.

The Pseudo code for Drawing a circle for a specified radius „r" and center (xc, yc) that starts with (0, r) and moves at first quadrant until x=y (For example 45 degree), first octant [11].

Initial conditions:

- x = 0
- y = r
- d = 3 - (2 * r)

Steps:

1. Set initial values of (xc, yc) and (x, y)
2. Calculate decision parameter d to d = 3 – (2 * r).
3. call display Bresenhm Circle(int xc, int yc, int x, int y) method to display initial(0,r) point.
4. Repeat steps 5 to 8 until x < = y
5. Increment value of x.
6. If d < 0, set d = d + (4*x) + 6
7. Else, set d = d + 4 * (x – y) + 10 and decrement y by 1.
8. call display Bresenhm Circle (int xc, int yc, int x, int y) method.

9. Exit.

2.3 Transformations

An object is moved by transformations to another position in the screen where a point,in 2D, could be translated through adding (tx, ty),the translation coordinate, to that of X,Y, the original coordinate, in order to obtain X′,Y′, the new coordinate.

In fact, transformations play a vital role in the field of computer graphics for repositioning the graphics on the screen and, eventually, changing their orientation or size [12].

2.4 Translation

On the screen, an object is moved, by translation, to a different position. A point existed in 2D could be translated through adding translation coordinate (tx, ty) to that of the original coordinate X,Y to obtain the new coordinate X′,Y′.

We could write that:

X’ = X + tx

Y’ = Y + ty

The pair (tx, ty) is known as the shift vector or translation vector. The above-mentioned equations could be represented by the use of the column vectors [12].

P=[X][Y]

p' = [X′][Y′]T = [tx][ty]

It can be written as:

P’ = P + T

2.5 Rotation

Concerning rotation, the object can be rotated in a specific angle θ theta from its own origin. Through the following figure, it is apparent that the point PX, Y.

is situated at angle φ out of the horizontal X coordinate with the distance r out of the origin [13].

Let‟s assume that you wish rotating it at the angle θ. Upon rotating it to a new location, a newly point P‟ X′,Y′ will be obtained. Figure 3 shows the process of rotation.

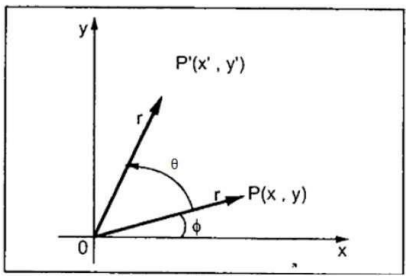


Figure 3: The process of rotation

The use of standard trigonometric the original coordinate of point P(X,Y) could be shown as:

X=rcosϕ.....(1)

Y=rsinϕ.....(2)

In a similar manner, the point P‟ X′,Y′ can be represented as:

$x'=r\cos(\phi+\theta)=r\cos\phi\cos\theta-r\sin\phi\sin\theta$.....(3)
$y'=r\sin(\phi+\theta)=r\cos\phi\sin\theta+r\sin\phi\cos\theta$.....(4)
With substitution of equation 1& 2 in 3 & 4, In a respective way, we will obtain:
$x'=x\cos\theta-y\sin\theta$
$y'=x\sin\theta+y\cos\theta$
By representing the above- mentioned equation in the matrix form:
$[X'Y']=[XY][\cos\theta-\sin\theta\sin\theta\cos\theta]$OR
P' = P . R
In which R represents the rotation matrix:
$R=[\cos\theta-\sin\theta\sin\theta\cos\theta]$

The rotating angle could be either positive or negative. As for positive rotating angle, the above rotation matrix can be used. Concerning negative angle rotating, the matrix will be changed as it is illustrated below:

$R=[\cos(-\theta)-\sin(-\theta)\sin(-\theta)\cos(-\theta)]$
$=[\cos\theta\sin\theta-\sin\theta\cos\theta](\because\cos(-\theta)=\cos\theta\text{and}\sin(-\theta)=-\sin\theta)$

2.6 Scaling

Scaling transformation is used, for changing the object size. In this process, you either compress or expand the object dimensions. The process of scaling can be done through multiplying the original coordinates object with factor of scaling for gaining the wanted result [14].
Let‟s assume that X,Y are the original coordinates
, the factors of scaling are (SX, SY), and the produced coordinates are X′,Y′
Mathematically,this could be represented as written below:
X' = X . SX and Y' = Y . SY

The object in X and Y direction is scaled by the scaling factor SX, SY in a respective way. The above-mentioned equations could be also represented in matrix form as follows:
$(X'Y')=(XY)[Sx00Sy]$
OR
P' = P . S

In which S represents the scaling matrix. The Figure 4 shows the scaling process.

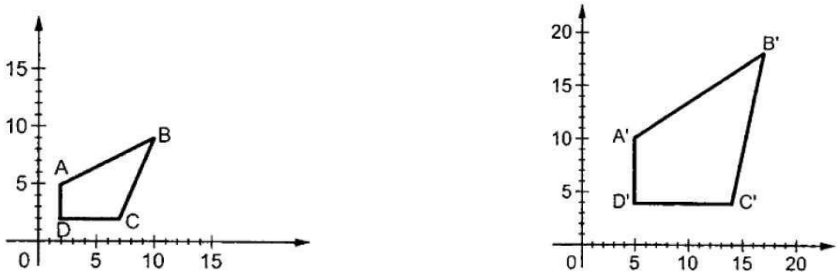


Figure 4: process of scaling

If values are provided less than 1 to the scaling factor S.Then, the object size could be reduced. If values greater than 1 are provided then the object size can be increased.

3. Hardware Implementation

3.1 Hardware System Design

Figure 5 presents a set of steps included in the over-all hardware implementation structure of the suggested system which is the drawing shapes (Line and Circle) are using Bresenham's drawing algorithm with scaling, moving, rotation and coloring depends on the values tuned through the Rotary Encoder. The operations are selected via 4-slide switches as shown in Table 1 and finally displaying all shapes on the SVGA screen through VDA ports.

The resolution of the SVGA display obtained is (800*600) pixels to display the shapes with a high accurate and clear.

Rotary Encoder was used to change Tx and Ty for moving, Sx and Sy for scaling, angle for rotation and 8-different color for coloring the mansion shapes.

All scaling, moving, rotation, drawing, coloring and displaying processes were done sequentially at 50MHz frequency successfully. The operations are displayed on the LCD screen.

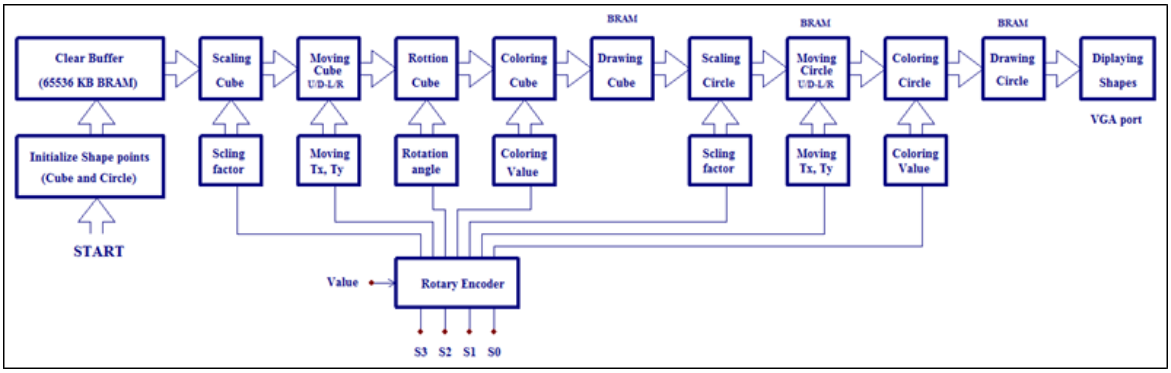


Figure 5: A set of steps included in the over-all hardware implementation

Table 1: Four Slide Switches Operation Selector

Selector in Decimal	Selector in Binary	Operation
0	0000	Circle Scaling
1	0001	Circle Moving L/R
2	0010	Circle Moving U/D
3	0011	Circle Coloring
4	0100	Cube Scaling
5	0101	Cube Moving L/R
6	0110	Cube Moving U/D
7	0111	Cube Coloring
8	1000	Cube Rotation
9	1001	None
10	1010	None
11	1011	None
12	1100	None
13	1101	None
14	1110	None
15	1111	None

As we notice in the above table, there is no circle rotation because it does not affect the circle drawing. Only the operations on the circles are scaling, movement and coloring. The blank cells on the operation field of the above table can be used to the future works like doing shadow and bolding all the shapes.

The value of the four slide switches is implemented in VHDL as multiplexer to select all operations on the system.

3.2 Finite State Machine

The hardware implementation of the system design was carried out by using Finite State Machine (FSM), where the drawing processes goes through a set of states liable to change from one to another based on the responses to programmed conditions. Figures 6a, 6b, 6c present the flowchart of the suggested system for basic drawing operations. It also shows the different states side by side with the possible transitions as shown in the flowchart of Figure 7.

The implementation of the Rotary Encoder, slide switches selector, Horizontal and Vertial timing of the VGA port signals are also used FSM to obtain all system requirements values to draw the shapes.

Some of states of the FSM was divided into several sub-states to obtain a maximum frequency of the kit to correspond with the frequency of the horizontal and verical timing of the VGA port (50 MHz) that is the default frequency of the SPARTAN 3E-XC3S500 starter kit, that is because it was used a (800*600) pixels resolution of the SVGA screen with 50Hz.

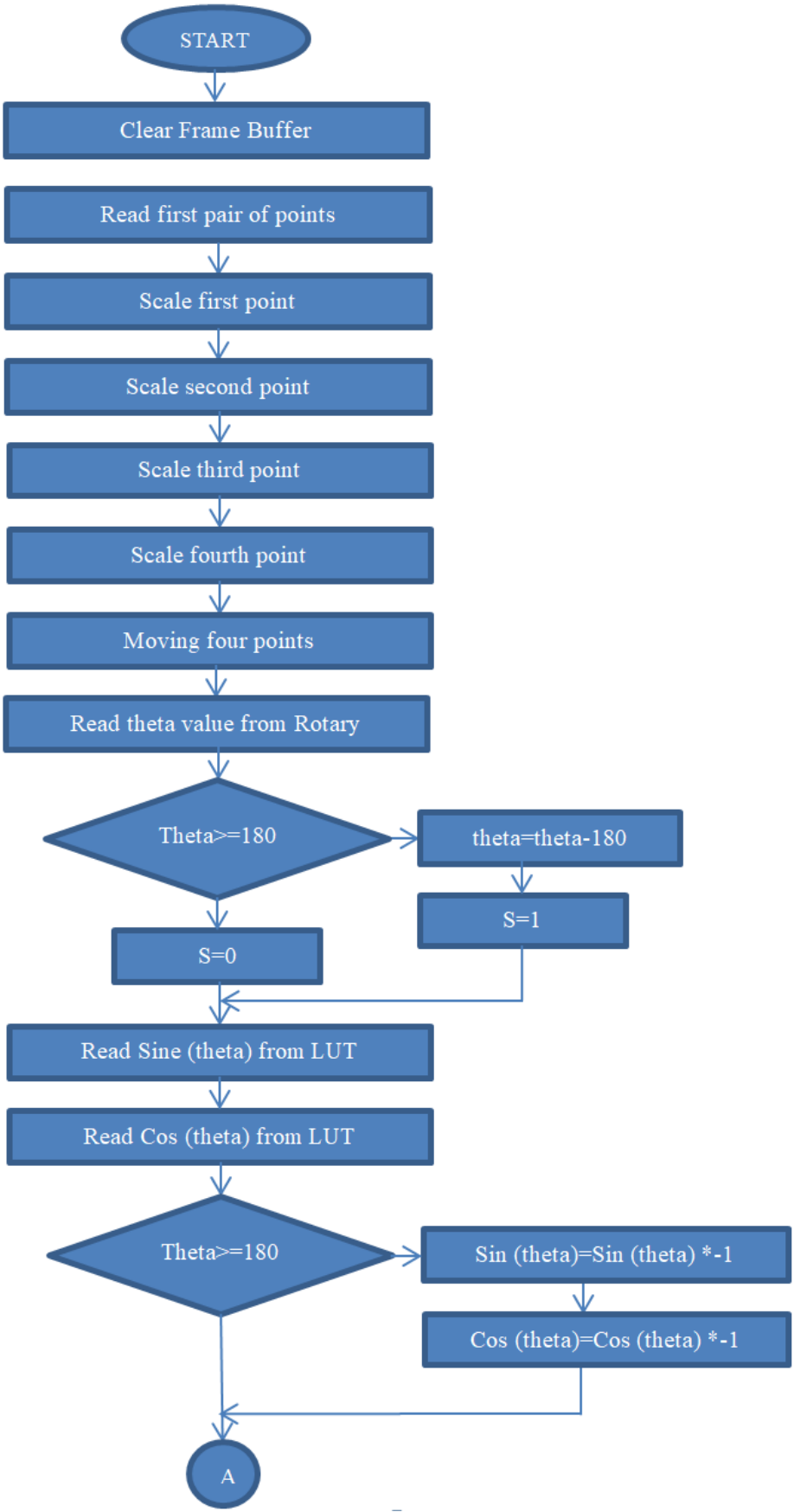


Figure 6a: System Design Flowchart- part 1

Figure 6b: System Design Flowchart- part 2

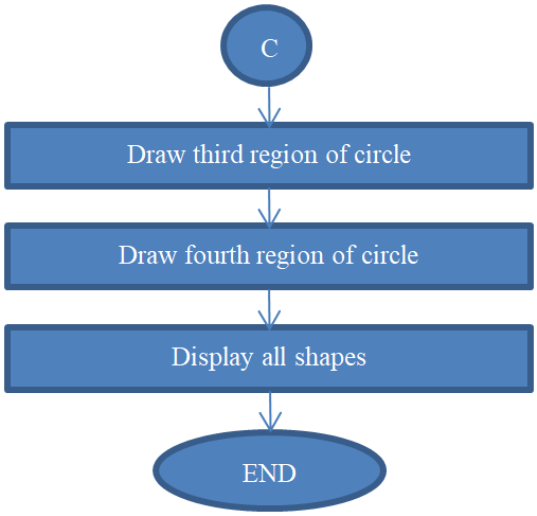


Figure 6c: System Design Flowchart- part 3

3.3 Look Up Table (LUT)

In the rotation process, it needs to calculate SIN and COS. The angle should be in the range (0 – 359). To obtain SIN and COS functions, it should be implemented by using Look Up table. Since we have 360 degree, then it should be used a memory size of 360*17-bit, 1-bit for integer part of and 16-bit for fraction part of the result. The size of the above memory reduced to 180*17-bit instead of 360*17-bit. It means that the contents of memory of the functions of SIN and COS of the angles between (0-179). Then, to calculate SIN or COS of the angle less than 180, the result was directly obtained from Look Up Table, but if the angle was greater than or equal 180, the result should multiplied by -1. That's all.

The total memory size needed to implement the above functions was 360*17-bit only instead of 720*17-bit.

Table 2 illustrates the overall BRAM memory size for the complete system design.

Table 2: Overall BRAMs needed to implement all drawing processes

Frame Buffer BRAM	3-bit	16-bit	[2:0]	[15:0]	65536	196608	12
SIN BRAM	17-bit	8-bit	[16:0]	[7:0]	180	16384	1
COS BRAM	17-bit	8-bit	[16:0]	[7:0]	180	16384	1
Total No. of Block RAM needed =							**14**

4. Results Discussion

Bresenham's algorithm for line and circle drawing, moving, scaling and rotation are implemented using VHDL based on SPARTAN 3E-XC3S500. Furthermore, the additional VHDL processes are used to implement moving, scaling and rotating including the uses of 800*600 SVGA timing to draw the whole shapes, four slide switches for selecting the operation on the chosen shape, Rotary Encoder for controlling the three operations the scaling, moving and rotation, and 16*2 built in LCD display for showing the current one of the seven operations.
At the beginning, the size of the frame buffer is (256*256) pixels, each pixel includes 3-bit to color any shape. Then eight colors are included on each line and circle. Table 3 shows the original points of the cube and circle.

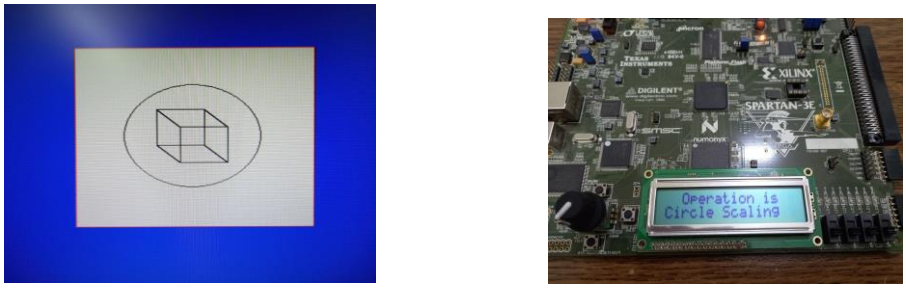
Table 3: Original points of the Cube and Circle

Points	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Cube	(50,50)	(150,50)	(150,150)	(150,50)	(75,75)	(125,75)	(125,125)	(75,125)
Points	Center		Radius					
Circle	(125,122)		63					

Figure 7 (a) illustrates the original position of the whole points related to the cube and circle that are drawn via SPARTAN 3E-XC3S500 FPGA VGA port in the SVGA screen. White with red color of boundary are selected for frame buffer background, while blue is selected for display. In the frame buffer, as it is mentioned in the above table, the Cube and the Circle are drawn in the previous position.

- **Original shapes display**

As it was mentioned earlier, the selected screen resolution is (800*600) SVGA before using 50MHz refresh rate for displaying the shapes in a clear way and with as high as possible resolution.



(a) (b)

Figure 7: (a) Original position of the Cube and Circle, (b) Circle scaling operation type

- **Circle scaling**

As Figure 7 (b) shows, when the slide switches are put at (0000) position, LCD displays the operation type (which is Circle Scaling). Figure 8 illustrates all operations done by the designed system.

Figure 8: All operations

5. Conclusion and Future Works

5.1 Conclusions

The main purpose of this work is to design and implement a graphics processor using FPGA, which is able to handle 2D rendering graphics. Bresenham's algorithm for line and drawing is commonly used in various applications such as graphic design, game design, and so on. The following points summarize the important conclusions that are got out of this research:

- Implementing of Bresenham's algorithm for line and drawing based on FPGA that is used as a GPU processor to increase the speed of drawing any shape and the use of another algorithm that can be applied on this algorithm such as translation, rotation and scaling.
- The resolution of the VGA display is 50MHz 800*600 which requires increasing of the design frequency that is affected by the programming skills.

- The used line generation algorithm is the 3D Bresenham's algorithm using integer calculations so as to avoid the floating point occupying a large area of an FPGA and also slows down the speed.

5.2 Future works

There are wide areas for enhancing the processes of drawing and translation that area applied on any drawn shapes as a future works:

- It could be used as an enhanced FPGA device having a larger Block RAM increasing the frame buffer size. It can also be used for increasing frequency of a system to get a resolution, (800*600 SVGA), that is mentioned above.
- Using DDR-SDRAM enables the designed system processing largedata (64 M-Byte memory size) and with high speed (100 MHz frequency) which led to the improving of the performance efficiency of the system.
- This work could be enhanced so as to draw 3D shapes as a future work in the case when the better FPGA platform is existed.
- It can be used as an additional complex algorithm for enhancing the result such as edge smoothing to get a perfect result.

REFERENCES

[1] vrlab.epfl.ch., 2014, "Graphics/vision publications acceptance rates statistics", 2014, Retrieved 2014-05-01.

[2] Goldwasser, S.M., 1983, “Computer Architecture For Interactive Display Of Segmented Imagery. Computer Architectures for Spatially Distributed Data” Springer Science & Business Media. pp. 75-94 (81). ISBN 9783642821509.

[3] Information Processing Society of Japan, 2015, "LINKS-1 Computer Graphics System-Computer Museum", Retrieved 15 June 2015.

[4] Baumann C., 2010, "Field Programmable Gate Arrays (FPGA)", University of Innsbruck.

[5] Rose J., Kuon I., Tessier R., 2007, "Fpga architecture: Survey and challenges", Foundations and Trends in Electronic Design Automation.

[6] Niu Lianqiang and Feng HaiWen , 2009, "A Line Segments Approximation Algorithm of Grating Lines", Journal of 2009 International Forum on Computer Science-Technology and Applications, IEEE,Computer Society , Vol. 2 , Pages: 34-37, 2009.

[7] Chikit Au and Tony Woo, 2011, "Three Dimensional Extension of Bresenham’s Algorithm with Voronoi Diagram", Journal of Computer-Aided Design, Vol. 43, Issue: 4, Pages: 417-426, 2011.

[8] Fakhrulddin Hamid Ali , “Depth Buffer DDA Based on FPGA“, 2011, Journal of Al-Rafidain Engineering ,Vol.19 , No.5 , October 2011.

[9] Kahraman Serdar Ay, Atakan Dogan, 2013, "Hardware/Software Co- Design of A 2D Graphics System on FPGA", Electrical and Electronics Engineering, Anadolu University, Eskisehir, Turkey, International Journal of Embedded Systems and Applications (IJESA) Vol.3, No.1, March 2013.

[10] Zingl, Alois, 2012,. "A Rasterizing Algorithm for Drawing Curves" (PDF)., The Beauty of Bresenham's Algorithms

[11] vrlab.epfl.ch., 2018, "Murphy's Modified Bresenham Line Algorithm", homepages.enterprise.net. Retrieved 2018-06-09. Blender.org, Open source 3D creation, https://www.blender.org/ (09.2017)

[12] S.Fawad, 2006, "Adapting Bresenham Algorithm ", Journal of Theoretical and Applied Information Technology ,Vol. 2 Issue: 2 ,Pages: 27-30, 2006.

[13] Wayne Carlson, 2003, “A Critical History of Computer Graphics and Animation”, Archived April 5, 2007, at the Wayback Machine. Ohio State University

[14] Jon Peddie, 2013, “The History of Visual Magic in Computers: How Beautiful Images are Made in CAD, 3D, VR and AR” , Springer, 2013, p. 101, ISBN 978-1447149316