# Anti-Aliased DDA

Fakhraldeen Hamid Ali
**Computer Engineering Departement - Unversity Of Mosul**

**Email: fhali310@yahoo.com**

## Abstract

Bit-mapped images are prone to the jaggies (stair-step effect along edges) because the computer uses small dots to build images. This effect is called aliasing and the technique used to reduce it is called antialiasing. This paper investigates aliasing along straight line segments or edges, its origin, and how it is affected by the orientation or slope of the segment. A method for antialiasing or smoothing the straight line segments by modifying the intensity of the pixels is presented. Hardware implementation of this method is finally formulated and tested using Field Programmable Gate Arrays (FPGA).

**Keywords**: pixel, jaggies, antialiasing, raster, FPGA.

**تنعيم المستقيم المرسوم بالمحلل الرقمي التفاضلي**

**فخرالدين حامد علي**
**كلية الهندسة \ جامعة الموصل**

**الخلاصة**

تتعرض الرسوم المرسومة بالحاسوب الى ظهور مايشبه الدرج على طول الحافات المستقيمة وذلك بسب بناء هذه الرسوم من نقاط صورية محددة. في هذا البحث عرض و تحليل لهذه الظاهرة واسباب حدوثها وكيفية تأثر هذه الظاهرة بزاوية او ميل المستقيم . للتقليل من ظاهرة التدرج او لتنعيم المستقيم يتم معاملة لون او الكثافة الضوئية للنقاط التي تستخدم في بناءه اعتماداً على بعد مركز النقطة الصورية عن المستقيم .اضافة الى ذلك يقدم هذا البحث تصميما بسيطاً للخوارزمية المعتمدة على الطريقة المذكورة منفذة بالكيان المادي بالاعتماد على مصفوفة البوابات المبرمجة حقلياً .

**Received 14 Feb. 2008** **Accepted 9 June 2008**

## 1-Introduction

Today, almost all displays are raster displays where raster graphics use a matrix of pixels (picture elements) to represent images [1,2]. The rasterization of a straight line segment can be accomplished using the line drawing algorithm called a Digital Differential Analyzer (DDA). On the other hand, it can be done using Bresnham's algorithm (a modified DDA) which uses integer mathematics only [3]. However, both produces the same pixels with the same aliasing effect. A line segment is defined by an infinite set of points which lie between two end points or vertices but its rasterization represents it with its samples or defined fixed positions only based on the screen resolution. Smooth straight lines appear stair like lines when displayed for variety of reasons, the most common being that the output device ( display monitor ) does not have enough resolution to portray a smooth line as mentioned above. This means that sampling is done below the Nyquist rate ( under sampling ) [ 4,5,6].

In digital signal processing, anti-aliasing is the technique of minimizing the distortion artifacts known as aliasing when representing a high-resolution signal at a lower resolution. Antialiasing is used in digital photography, computer graphics, digital audio, and many other domains [7,8]. In the image domain , aliasing artifacts can appear as jagged appearances of smoothed straight edges as mentioned above.
Fundamentally, there are three methods which can be used for antialiasing. Since aliasing in computer-generated graphics is a spatial aliasing, an obvious solution is to increase the sampling rate or raster resolution which decreases aliasing effect. This method is limited by the display hardware [9,10]. In the second method, super sampling is used which is a technique of collecting data points at greater resolution ( usually by a power of two ) than the final data resolution. These data points are then combined or averaged (down sampling) to the desired resolution. This technique is refer to as post-filtering the image [6,11,12,13,14,15]. The third method of antialiasing treats each pixel as a finite area rather than a point and is known as pre-filtering the image [14,16,17]. The last two methods can only be implemented using systems with a display of more than two intensities per pixel. The third method has a computational advantage over the second one since it does not requires a large memory for storing the image at the sub-pixel stages. This is why pre-filtering techniques are cheaper, but still effective, when implemented. For this reason, this method is adopted in this research work.

## 2- Previous work

In this section a review of latest related work is summarized. In 1987, the researcher Roman P.Molla designed three systems for different algorithms to implement scan conversion for a straight line segment. The Digital Differential Analyzer(DDA) and Bresenham algorithms were designed using serial processing in addition to the implementation of the DDA algorithm using parallel processing. The research discussed the performance , cost and the error ratio for the three designed mentioned systems [18]. Andreas Schilling presented in 1991 a hardware realization of an algorithm for antialiasing. He mainly used PLA's in his design. The algorithm is based on subpixel mask look up table [11]. In 1993, Andreas Schilling and Wolfgan Straber introduced an algorithm that deals with hidden surface elimination problem at pixels level. The algorithm provided the solution of the aliasing problem resulted in the scan conversion operation. The hardware implementation was divided into three stages in order to apply the pipeline technique to improve the performance. The designed architecture costs 12000 gate and the chip has ability to produce 20 M

pixel/sec[19]. Molnar introduced in 1994 a classification of a different architectures that implement the scan conversion operation using parallel processing. The architecture classification depends on the basic stages of the image generation operation. 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. Assignment stage responsible for allocating parts of the scene to the parallel processing units. Defragmentation stage to collect the partial results and store them in the frame buffer [20]. In 1996, C.Scott Ananian and Grog Humphregs suggested three different architecture to implement the ray tracing algorithm. The designed hardware includes two units. The first unit is responsible of the implementation of the scan conversion operation, the second unit is the ray casting unit. The paper discussed the performance and cost for the three designed architecture [21]. In 2004, P. Beaudoin and P. Poulin proposed a mechanisim to compress the antialiasing buffer and limit the bandwidth requirements for hardware edge antialiasing.The presented method supports the usual OpenGL fragement-related functions [5]. In 2004 also, Y. K. Liu et al presented an integer one-pass algorithm for voxel traversing along a line. The proposed approach is based on a modification of the well known Bresenham algorithm [2]. In 2005, M Golipour-Koujali investigated different problems with their solutions when applying antialiasing to conic sections [10]. D. Wang et al presented in 2006 an antialiasing method using a DSP-based display system for removing the undesired jaggies occurred in the line drawing. It is concluded in this paper that the application of antialiasing on color lines takes 60 times the time required without antialiasing [4].

## 3- Aliasing

The level of aliasing in a straight line segment generated by the DDA is a function of its orientation. When the line is horizontal or vertical, no aliasing appears and all the generated pixels are exactly located a long the straight line. This means that the error value for each generated pixel is zero. This is also true when the slope of the line is +1 or -1 . For all other cases, aliasing is produced as a function of two parameters. The first parameter is the DDA accumulating error, which is repeatable in nature. The second parameter is the slope or orientation of the line. In figure (1) a demonstration example of the aliasing along a single straight line segment is shown (up) with the corresponding error function ( f ) is shown down.

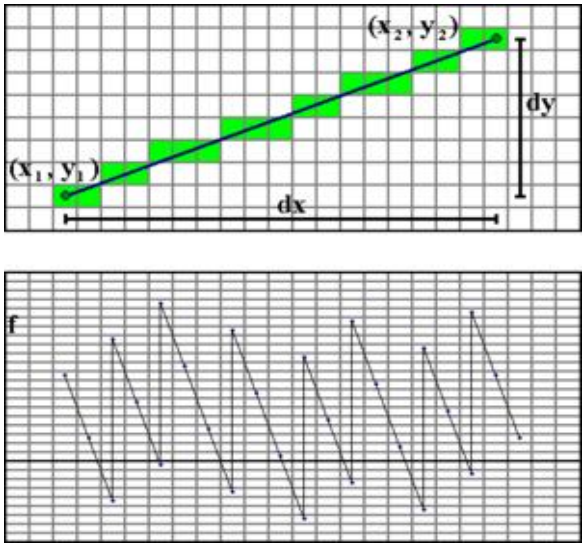



Fig.(1) Aliasing example with accumulating error f
( borrowed from reference [22] )

Figure (2) Illustrates the aliasing with orientation. An examination of the figure shows that the nature of aliasing is repeatable 8 times through the angle value from zero to 360 degrees due to symmetry. In addition to variation with orientation or slope, aliasing nature is repeated also along the straight line segment itself while the slope is constant as mentioned before.

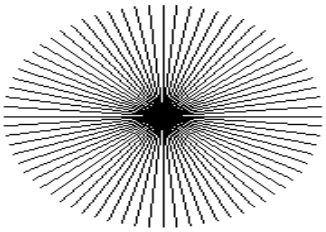


Figure (2) Aliasing versus orientation

To report how the straight line aliasing varies with the slope, a single accumulating RMS error value (LE) is defined and computed for each line segment considering the error of all its pixels. This is repeated for all lines of interest of a slope value between 0 and 1 (or angle between 0 to 45 degrees)**.** The results are presented in figure (3).

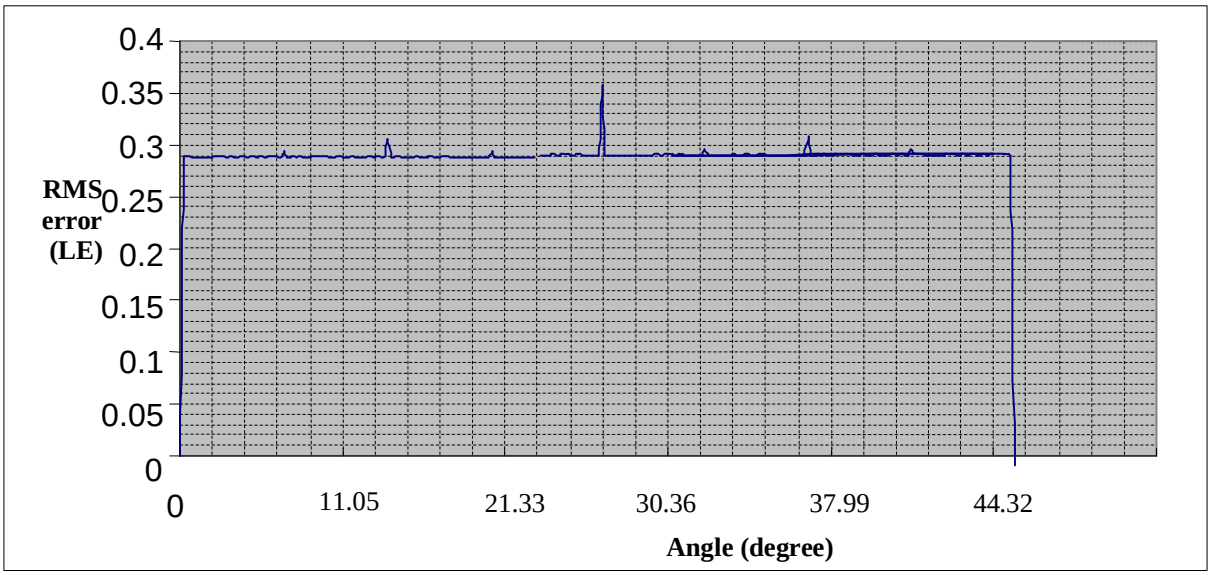



Figure (3) Line RMS error (per unit) as a function of angle

Examination of figure (3) shows that the variations of LE are negligible around a measured nominal value 0.288 (per unit) through out the range except an abrupt increase of LE at certain angle values. At these values, listed in table (1), a relatively stronger level of aliasing is produced in particular.

Table (1) Special high-aliasing angles

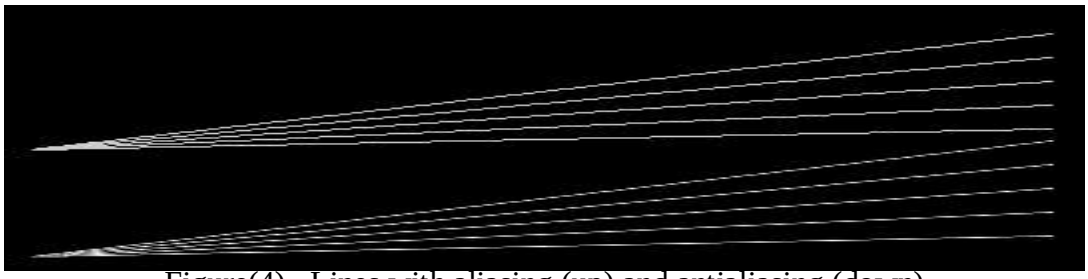
| Angle (degree) | LE (RMS) | LE- increase (over 0.288) |
|---|---|---|
| 0.0 | 0.000 | - |
| 7.12 | 0.293 | 1.5 % |
| 14.03 | 0.306 | 6 % |
| 20.55 | 0.293 | 1.5 % |
| 26.56 | 0.353 | 22 % |
| 32.00 | 0.293 | 1.5 % |
| 36.86 | 0.306 | 6 % |
| 41.18 | 0.293 | 1.5 % |
| 45.00 | 0.000 | - |

## 4- Pre-filtering

To reduce aliasing , the intensity of each affected pixel is computed using a convolution integral [22]. The intensity function is convolved with a proper sampling filter across the pixel area. This, in effect, produces a modified intensity value for each of those pixels partially covered by the image. Unfortunately, the evaluation of such integral is too expensive, in terms of the processing time, and moreover it is not easy to implement using microcomputers. Therefore, a simpler approach is to modify the intensity of each affected pixel according the percentage coverage of its area by the image. This technique is used to modify Bresenham's algorithm when using a DDA to draw a single straight line segment on a raster display. The intensity function used for antialiasing is exponential which is ymmetric around the origin and can be presented by the following equation:

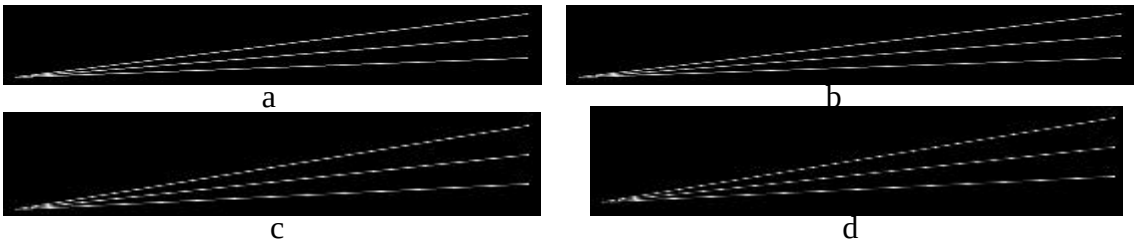
$$I = \exp ( - | kf | ) \qquad k = 2 \qquad (1)$$

Where: "f" represents the error function of a value range (-0.5 to 0.5).
"I" represents the normalized intensity value (0 to 1).
Line segments with different slopes are shown before antialiasing and after with a value of "k" being 2 in figure (4).



Figure(4) Lines with aliasing (up) and antialiasing (down)

On the other hand different levels of antialiasing can be produced using different values of "k" ( k = 2,4,6,8 ) as demonstrated in Figure(5).




Figure(5) Different levels ( a to d) of antialiasing

Two other intensity functions, in addition to the exponential function, are also used for antialiasing but no significant differences are noticed. These functions are line function and cosine function which are given below.

$$I = 1 - |k f | \qquad k = 1.264 \qquad (2)$$
$$I = \cos (k f ) \qquad k = 2.388 \qquad (3)$$

## 5- Hardware solution

In this section a hardware implementation using FPGA of the proposed antialiasing method is presented. A block diagram of the hardware designed unit is illustrated in figure (6). The line pixel calculation part is responsible for the scan conversion operation of a line segment defined by its two input vertices V1(X1,Y1) and V2(X2,Y2). The scan conversion operation begins when the start signal is set "ON". The address of the frame buffer is calculated for each pixel from its computed coordinate values (Xc,Yc) to load the intensity data at the right location. The intensity values of the color register are modified according to the error value to perform anti-aliasing. The lookup table contains discrete intensity function values for

smoothing a straight line segment by modifying the intensity of its pixels. The color register content is multiplied by the output of the lookup table to adjust the intensity before being stored in the frame buffer (image memory). This is performed for each pixel produced by the line pixel calculation part.

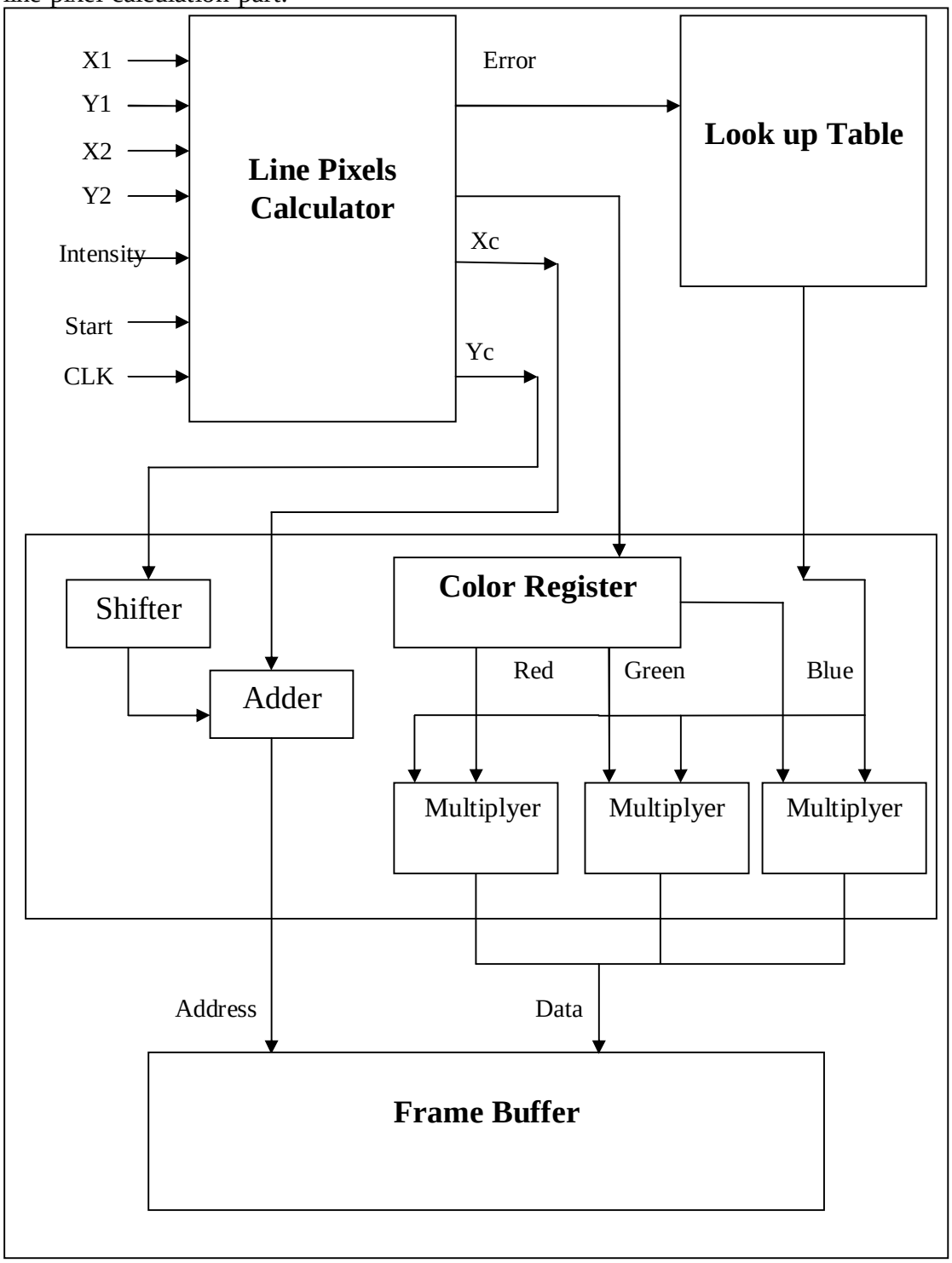



Figure (6) Designed hardware unit

The hardware unit is implemented using VHDL and synthesized using FPGA available on the kit-board Spartan-3E [23,24,25]. Figure (7) shows the simulation wave forms for an example executed by the implemented hardware unit. Table (2) shows the utilization resources of Spartan3 Kit that is used to implement the unit.

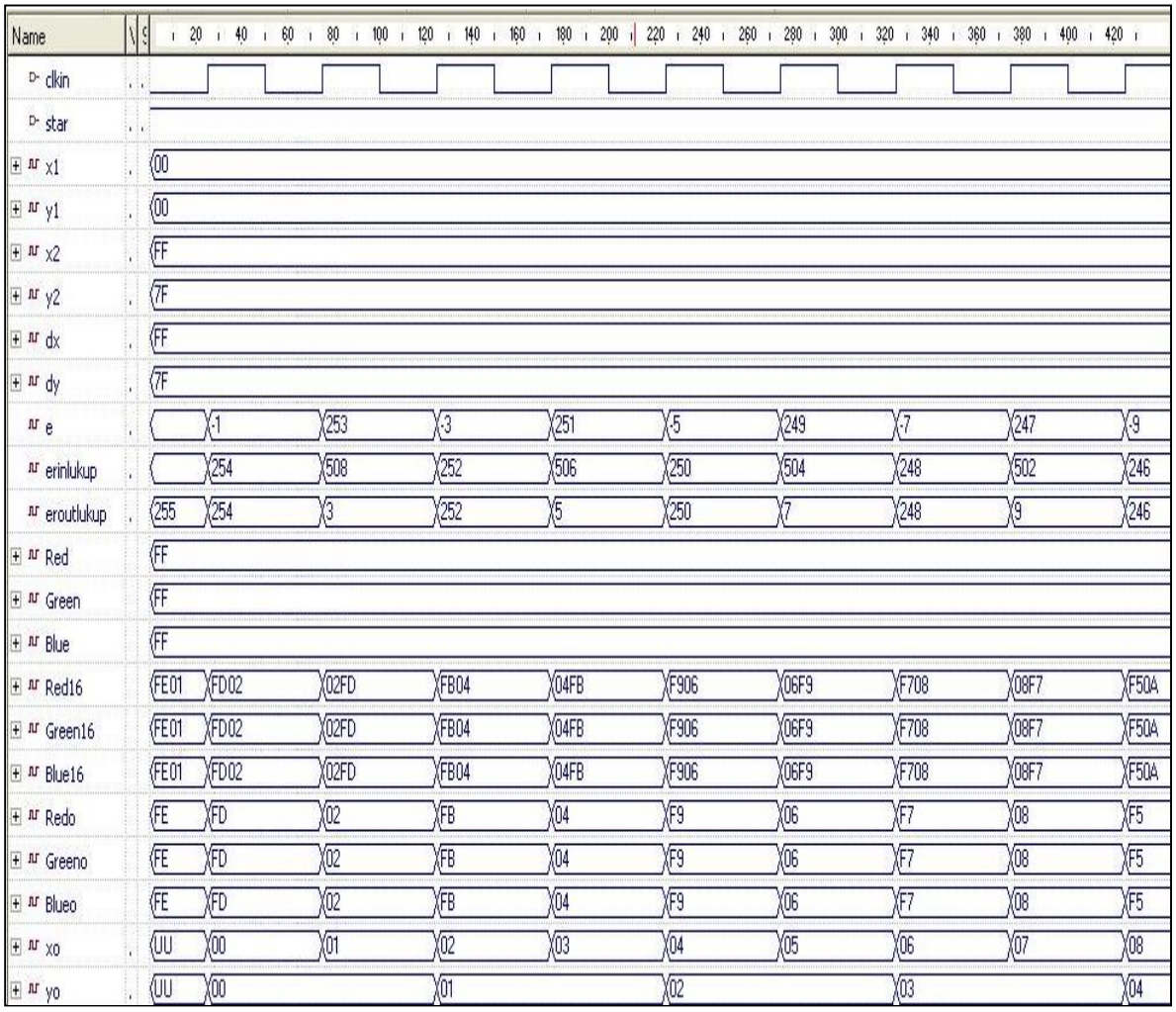



Figure (7) Example sample-waveforms

Where:
Star: start signal.
x1, y1 & x2, y2: the first and second vertices.
dx & dy: the coordinate difference.
e: the error.
erinlukup: the error address input to the lookup table.
eroutlukup: the error intensity function value output from the lookup table.
Red16 & Green16 & Blue16: the modified intensity 18 bit color.
Redo & Greeno & Blueo: the modified intensity 8bit color rounded-values.
xo, yo: the computed pixels x and y coordinates.

In figure(7) the inputs vertices are (0,0) and ( FF hex., 7F hex.). The scan conversion operation begins when the 'star' signal becomes "ON". The line pixels calculator computes the coordinate difference values dx (FF) and dy (7F) and then determines the greater coordinate difference to evaluate the error in a correct way. After that, from the value of the error the increment in x and increment in y coordinates are computed to determine all pixels between the two vertices and at each pixel a new value of the error is computed. In addition to that, the error is used to address the look up table after the required modification (the memory address must be positive integer value) to extract the intensity function value from the look up table

for antialiasing. Each intensity or color value (Red , Green , Blue) is modified by multiplication with the value from the look up table to perform smoothing. However, the simulation first step values in figure (7) are calculated theoretically according to Bresenham's algorithm, for comparison, in the following steps:

dx = x2-x1 = 255 – 00 = 255 =FF hex.
dy = y2-y1 = 127 – 00 = 127 =7F hex.
e=2dy-dx = 2*127 – 255 = -1.
erinlukup= e+255 = 254.
eroutlukup= intensity function(erinlukup) =254.
Red16= Red * eroutlukup = 255 * 254 = 64770 = FD02 hex.
Green16= Blue16=Red16 = FD02 hex.
Redo= MSB of Red16 = FD hex.
Greeno= Blueo= Redo = FD hex.

Table(2) Resources utilization (for 64*64 Pixel, each pixel three byte)

| Type Resources(or Frequency ) | Utilized Resources | Total Resources | Ratio |
|---|---|---|---|
| Number of Slices | 195 | 1920 | 10% |
| Number of Slices Flip flops | 75 | 3840 | 1% |
| Number of 4 input LUTs | 342 | 3840 | 8% |
| Number of Bounded IOBs | 144 | 173 | 83% |
| Number of Block RAMS | 7 | 12 | % 58 |
| Number of GCLKs | 1 | 8 | 12% |
| Number of MULT18X18s | 3 | 12 | 25% |
| Maximum Operating Frequency | 112.007MHZ | | |

## 6- Conclusions

The error produced by scan-converting any straight line segment to its displayable pixels is repeatable in nature and it is a strong function of its slope or orientation. The same pixel error values are produced whether using the DDA or Bresenham's algorithm. A variable level of antialiasing is feasible via only changing the value of one parameter (k) in the intensity function used for antialising. On the other hand, adopting a look up table form of function representation permits using any intensity function by just reprogramming the table with its discrete values. However, two main contributions in the field of antialiasing in computer graphics can be concluded out of this paper:
1-The novel analysis of the pixel error presented in figure (3) and table (1).
2-The simple and flexable design and of the proposed method using FPGA.

## References

1-Joey Lawrance ," The History of Computer Graphics ", CS 551 Term Paper, University of Utah, February 19, 2004.
2-Y.K. Liu, B. Zalik and H.Yang," An Integer One-Pass Algorithm for Voxel Traversal ", Computer Graphics Forum, Volume 23 , Number 2, 2004, pp 167-172.
3. Angel, E., and Morrison, D. Speeding up Bresenham’s Algorithm. IEEE Computer Graphics and Application, 11(6), pp. 16-17, 1991.

4-Dusheng Wang , Hock Lye Toh , Xiang Chen , Fan Yang ," A Simple Anti-Aliased Method For Straight Line Drawing Based On DSP Platform ", Posters proceedings ISBN 80-86943-04-6 , WSCG' 2006 , January 30 – February 3, 2006.
5- Philippe Beaudoin and Pierre Poulin ," Compressed Multisampling for Efficient Hardware Edge Antialiasing", LIGUM, Dept. I.R.O., University of Montreal, March 2004.
6-Peter Longhurst, Kurt Debattista, Richard Gillibrand, Alan Chalmers ," Analytic Antialiasing for Selective High Fidelity Rendering ", Dept. of Computer Science , University of Bristol, BS8 1UB, UK, 2005.
7-John Amanatides and Don P. Mitchell ," Antialiased of Interlaced Video Animation ", Computer Graphics, Volume 24, Number 4, August 1990. pp. 77-82.
8-M. Zwicker, W. Matuisk, F. Durand, and H. Pfister, "Antialiasing for Automultiscopic 3D Displays", Eurographics Symposium on Rendering, 2006.
9- WU, X. *An Efficient Antialiasing Technique*. Computer Graphics, vol. 25, No.4, pp. 143-152, Jul 1991.
10- M. Golipour-Koujali, " General Rendering and Antialiasing Algorithms for Conic Sections ", PhD thesis, London South Bank University, England, pp. 120-123 May 2005.
11-Andreas Schilling ," A New Simple and Efficient Antialiasing with Subpixel Masks ", Computer Graphics , Vol. 25, no. 4 , July 1991 ( SIGGRAPH '91 Proceedings ), pp. 133-141.
12-Peter Young ," Coverage Sampled Antialiasing ", Technical Report , NIVIDIA Corporation , October 30, 2006.
13- Fabris, A.E., Robust Antialiasing of Curves, PhD thesis, University of East Anglia, Norwich, November 1995.
14- Fabris, A.E. and Forrest, A.R. Antialiasing of curves by discrete prefiltering. In Computer Graphics (SIGGRAPH'97 Proceedings), August 1997.
15- Molnar, S. Efficient Super-sampling Antialiasing for High-performance Architecture. Technical report, 91-023, University of North Carolina, 1991.
16- Lathrop, O., Kirk, D. and Voorhies, D. Accurate Rendering by Sub-pixel Addressing. IEEE Computer Graphics & Applications, pp. 45-53, Sep 1990.
17-Qyvind Ryan ," Application Of Antialiasing In An Image Processing Framework Setting ", Dept. of Informatics, University of Oslo, Norway, 2006.
18- Roman P.Molla Vaya , "Parallel Fixed Point Digital Differential Analyzer " , Computer Journal , Vol. 23, No. 1, pp:46-52, 1987.
19- 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.
20- 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.
21- C. Scott Ananian, Greg Humphreys "A Hardware Accelerated Ray-tracing Engine" , Princeton University, Computer Graphics (SIGGRAPH '96 Proceedings), volume 21, pp: 95-102, 1996.
22- "http: // en.wikipedia.org/wiki/ Bresenham's_line_algorithm".
23- Liu, N. and Rockwood, P. *Antialiasing by Gaussian Integration*. IEEE Computer Graphics and Application, pp. 58-63, May 1996.
24-" Spartan-3E, FPGA Family : Functional Description ", DS312-2(V3.5), March 16, 2007, © 2006 Xilinx Inc.
25-" Spartan-3E Starter Kit Board User Guide:, www.xilinx.com UG230 (V1.0) March 9, 2006, © 2006 Xilinx Inc.
26- " JTAG Loader – Quick Guide- II ", ©2004XilinxInc.http://www.Support.Xilinx.com / xlnx/xweb/xil-ix-home.jsp.

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