Tikrit Journal of Eng. Sciences/Vol.18/No.3/September 2011, (89-105)

Hardware Implementation Of Line Clipping A lgorithm By Using FPGA

Amar I. Dawod
Computer Engineering Dept. - University of Mosul

Abstract

The computer graphics system performance is increasing faster than any other computing application. Algorithms for line clipping against convex polygons and lines have been studied for a long time and many research papers have been published so far. In spite of the latest graphical hardware development and significant increase of performance the clipping is still a bottleneck of any graphical system. So its implementation in hardware is essential for real time applications. In this paper clipping operation is discussed and a hardware implementation of the line clipping algorithm is presented and finally formulated and tested using Field Programmable Gate Arrays (FPGA). The designed hardware unit consists of two parts : the first is positional code generator unit and the second is the clipping unit. Finally it is worth mentioning that the designed unit is capable of clipping (232524) line segments per second.

Keywords: Clipping, Graphical Pipeline, Real time, FPGA.

تنفيذ الكيان المادي لخوارزمية القطع لقطعة خط مستقيم باستخدام البوابات القابلة للبرمجة حقليا

الخلاصة

إن متطلبات سرعة الأداء في مجال الرسوم الحاسوبية وتطبيقاتها في حالة تزايد مستمر أكثر من أي تطبيق أخر حيث أن عملية القطع و خوارزمياتها للخطوط المستقيمة و المضلعات تعتبر من إحدى أساسيات هذا المجال فقد تم إجراء الدراسات و البحوث الكثيرة و لوقت طويل جداً في هذه الخوارزميات. و بالرغم من التطور الكبير و الزيادة الهائلة في سرعة الأداء إلا أن عملية القطع لا تزال تمثل عنق الزجاجة في أي منظومة رسم حاسوبية. ولذا يعتبر تنفيذها من خلال المكونات المادية ضروري جداً في تطبيقات الزمن الحقيقي. نوقشت عملية القطع من خلال هذا البحث فضلاً عن ذلك فقد تم تنفيذ و تصميم الكيان المادي لخوارزمية القطع المقترحة إذ تم إعدادها و اختبارها بشكل نهائي باستخدام مصفوفة البوابات القابلة لإعادة البرمجة حقليا. أن الوحدة المصممة تتألف من جزءين ، يكون الجزء الأول مسؤول عن توليد الرمز ألموقعي أما الجزء الثاني فيكون مسؤول عن تنفيذ عملية القطع. وأخيرا فمن الجدير بالذكر أن الوحدة المصممة قادرة على تنفيذ عملية القطع لـ(232524) قطعة خط مستقيم في الثانية الواحدة.

الكلمات الدالة: القطع، منظومة رسم حاسوبية، مصفوفة البوابات القابلة لإعادة البرمجة حقليا

Introduction

With procedures for displaying output primitives and their attributes, a variety of pictures and graphs can be created. One of the most fundamental operation in the implementation of computer graphics is clipping. The clipping is done because of the limitation that no imaging system can see the whole world at once and the limitation of the display window of any display device. The human retina has a limited size corresponding to an approximation 90- degree field of view. Cameras have film of limited size and we can adjust their fields of view by selecting different lenses. We obtain the equivalent property in the synthetic camera by placing a clipping rectangle of limited size in the projection plane[1].

Quite often we wish to display only a portion of the total picture. In this case a window is used to select that portion of the picture which is to be viewed(much like clipping or cutting out a picture from a magazine). This is known as clipping. Generally any procedure that identifies those portions of a picture which are either inside or outside a specified region of a space is referred to as a clipping algorithm or simply clipping. The two dimensional region against which an object is to be clipped is called a clip window. The process of clipping determines which elements of the picture lies inside the window and hence they are visible. The algorithm selected for clipping depends on the geometric shape of the clipping window[2].

Blinn J.F. in 1991 described the standard computer graphics transform-clip-draw pipeline, and an overview of the clipping function is given. A simple algorithm for performing line clipping is presented. Homogeneous clipping, Z clipping, and global clipping using the algorithm are discussed[3].

Patrick G. M. in 1992 introduced a paper presenting a new 2D polygon clipping method, based on the Sutherland-Cohen 2D line clipping method. After discussing three basic polygon clipping algorithms, a different approach is proposed, explaining the principles of a new algorithm and presenting it step by step. A proposed implementation of the algorithm is given along with some results. A comparison between the proposed method, the Liang and Barsky algorithm, and the Sutherland-Hodgman algorithm is also presented, showing performances up to eight times the speed of the Sutherland-Hodgman algorithm, and up to three times the Liang and Barsky algorithm. The algorithm proposed can use floating point or integer operations, which can be useful for fast or simple implementations[4].

Václav Skala in 1996 presented A new algorithm for line clipping by convex polygon with O(1) processing complexity. It is based on dual space representation and space subdivision technique. The suggested algorithm also demonstrates that pre-processing can be used in order to speed up solution of some problems in computer graphics applications significantly. Theoretical considerations and experimental results are also presented[5].

Nishita T. and Johan H. in 1999 Introduced a paper discusses a curved tubular object which is a surface swept by a sphere/circle moving along a curve. For the trajectory curve, a 3D Bezier curve is employed, and its radius can be varied along the curve. In general, its surface cannot be defined by a closed form, while a high degree of polynomial must be solved for ray/surface intersection. This paper proposes an effective rendering method which uses a scan line algorithm for detecting curved tubular objects on the projection plane.

The calculation of the distance from a point to a curve plays an important role in the algorithm. Bezier Clipping Method is employed for this calculation[6].

Mingjun Zhang and Chaman L. Sabharwal in 2002 presented an improved parametric line clipping algorithm. The line clipping algorithm is extended to polygon clipping. The implementations of both algorithms are claimed to be novel and outperform many previous algorithms in the literature. This is supported by theoretical consideration and experimental results on randomly selected lines and polygons. The algorithms are implemented in Java. The Java applet allows the user to visualize the experimental results by comparing the existing algorithms and the new algorithms[7].

Skala V.I in 2004 presents a new robust and fast algorithm for line clipping by a convex polygon. The algorithm uses a preprocessing procedure in order to obtain significant speed up. The proposed algorithm is especially convenient for applications where points or lines are represented in homogeneous coordinates. The algorithm does not use division in floating point representation since the resulting points are in homogeneous coordinates. The algorithms benefit if vector-vector hardware supported operations can be used[8].

O'Toole A.J. , Harms J. and Snow S.L in 2005 described a database of static images and video clips of human faces and people that is useful for testing algorithms for face and person recognition, head/eye tracking, and computer graphics modeling of natural human motions. For each person there are nine static "facial mug shots" and a series of video streams. The videos include a "moving facial mug shot," a facial speech clip, one or more dynamic facial expression clips, two gait videos, and a conversation video taken at a moderate distance from the camera. Complete data sets are available for 284 subjects and duplicate data sets, taken subsequent to the original set, are available for 229 subjects[9].

Yong Kui Liu , Xiao Qiang Wang and Shu Zhe Bao in 2007 introduced a universal algorithm for polygon clipping, which is a frequent operation in GIS. In the proposed solution, the clipping polygons can be concave and may include holes. This algorithm is based on so-called entry/exit intersection point property, which has to be explicitly determined only at the first calculated intersection point. It uses a simple but efficient data structure based on a single-linked list. Boolean union and the difference between input polygons can also be determined after small modifications. This algorithm can easily be adapted to Boolean operations between regions composed of polygon sets[10].

The Clipping Algorithm

Line clipping is a fundamental approach whose efficiency directly affects the performance of a whole graphics system, involves several parts. First a given line segment is tested to determine whether it lies completely inside the clipping window. If it does not the line segment is tested to determine whether it lies completely outside the window, if it is not completely inside or completely outside, intersection calculation must be performed with one or more clipping boundaries to compute the new segment line[11-12].

Generally the method speeds up the processing of line segment by performing initial tests that reduces the number of intersection that must be calculated. Every line end point in a

picture is assigned a four digit binary code , called a position code or region code, that identifies the location of the point relative to the boundaries of the clipping window. Regions are set up in reference to the boundaries as shown in figure (1). Each bit position in the region code is used to indicate one of the four relative coordinate positions of the point with respect to the clip window: to the left , right, top, bottom[12-13]. By numbering the bit positions in the region code as 1 through 4 from right to left , the coordinate regions can be correlated with bit position as

Bit 1: left side.
Bit 2: right side.
Bit 3: down side.
Bit 4: upper side.

A value of 1 in any bit position indicates that the point is in that relative position otherwise the bit position is set to 0. Bit values in the region code are determined by comparing endpoint coordinate values (x , y) to the clip boundaries. For example bit 1 is set to 1 if x < Xmin. The other three bit values can be determined using similar comparisons. Now the region codes for all line endpoints have been established and so we can quickly determine which lines are completely inside the clip window and which are clearly outside. Any lines that are completely lies within the window boundaries have region code of 0000 for both endpoints and these lines are accepted. Any lines that have a 1 in the same bit position in the region codes for each endpoint are completely outside the clipping window and these lines are rejected because the two endpoints are outside from the same region. A method that can be used to test lines for total clipping is to perform the logical AND operation with both region codes. If the result is not 0000 the line is completely outside the clipping region. Lines that can not be identified as completely inside or completely outside a clip window by these tests are checked for intersection with the window boundaries. Such lines may cross into the window interior. The clipping process is started for a line by comparing an outside endpoint to a clipping boundary to determine how much of the line can be discarded. Then the remaining part of the line is checked against the other boundaries and this operation is continued until either the line is totally discarded or a section is found inside the window[1-14]. The algorithm is designed to check line endpoints against clipping boundaries in the order left , right , bottom, top.

Different orientations of different line segments are shown in figure(2). Line ab is completely inside the clipping window and no clipping process is required. On the other hand line cd is rejected since its end vertices are both outside the window and the logical AND of their codes produces a non zero, This shows that both vertices are a way from one side of the clipping window which excludes any possibility of intersection of the line segment with the window, Line ef has one vertex inside the window while the second vertex is outside, This line requires clipping so its intersection point has to be computed and its external parts is rejected when this intersection point replaces the outside vertex, If both vertices are outside and the logical AND of their codes produces zero then the line segment between them may or may not intersect the clipping window, Lines gh and kl in figure 2 have these properties and require more tests[15].

The intersection of line segment with the clipping window are computed using parametrical definition. A line segment which is between two endpoints P1 and P2 can be represented parametrically as follows:

$P(t1) = P1 + t1(P2 - P1)$ or
$t1 = (P(t1) - P1)/(P2 - P1)$ …1
$P(t2) = P2 + t2(P2 - P1)$ or
$t2 = (P(t2) - P2)/(P2 - P1)$ …2

The parameter t has any value in the range 0 to 1 where each value represents a distinct single point P(t) between the end points P1 and P2 inclusive.

In figure (3) two line segments are shown. The summation value of parameters (t1 + t2) for the line ab is less than 1. The same value for line cd is more than 1. This suggests that any line segment is rejected if it produces (/t1/+/t2/) > 1 and its intersection are computed otherwise. This test is performed only for those line segments which have both end vertices outside the clipping window and the logical AND of their end vertices codes is zero[15].

When a vertex is away from more than one side of the clipping window like vertex (a) in figure (3) which is away from both the left and bottom sides at the same time more than one intersection exists. The intersection which produces a higher value for the parameter t is the required one[15].

Finally the calculation of any intersection point will include computing the parameter (t) first using a known coordinate value of this point which is set to the border parameter value. The computed value t is then used to calculate the other unknown coordinate value of the intersection point according to the following equations:

For the intersection with left or right sides respectively:

$t1 = (X - X1) / (X2-X1)$, $Xi=X$, $Yi=Y1+ t1 (Y2-Y1)$ …3
$t2 = (X - X2) / (X2-X1)$, $Xi=X$, $Yi=Y2+ t2 (Y2-Y1)$ ...4

where X is Xmin or Xmax.

For the intersection with top or bottom sides respectively:

$t1 = (Y - Y1) /(Y2-Y1)$, $Yi=Y$, $Xi=X1+ t1 (X2-X1)$ ...5
$t2 = (Y - Y2) / (Y2-Y1)$, $Yi=Y$, $Xi=X2+ t2 (X2-X1)$ ...6

where Y is Ymin or Ymax.

Figure (4) illustrates a flowchart for the implementation of line clipping algorithm.

The Hardware Unit

In this section a hardware implementation using FPGA of the clipping algorithm is presented. A hardware unit is divided in two parts as shown in figure (5) & figure(6). The first unit is responsible for positional code generation. A block diagram of the positional code designed unit is illustrated in figure (5). The inputs of this unit are the clipping window and the vertices of a line segment that will be clipped. Positional code unit compares V1 and V2 to the clipping window by testing it against left and right sides and against up and down sides to create C1 and C2. After that the unit checks the vertices if they are completely inside the clipping window or outside and so it gives enable signal to the second unit that clips the line and computes the new vertices after performing the clipping operation.

When the enable signal is "ON" the clipping unit starts to check V1 and V2 if they are outside from the left and right sides using AND operation to compute the new vertices after computing the normalized parameters(t1 & t2). After that the designed unit checks the vertices from the up and down sides and performs the extra test to determine if the second intersection is necessary and computes the new vertices after computing the normalized parameter.

Test and results

The hardware unit is implemented using VHDL and

synthesized using FPGA available on the kit-board Spartan-3E. Figure (7) shows the simulation wave forms for an example 1 executed by the implemented hardware unit. Table (1) shows the utilization resources of Spartan3 Kit that is used to implement the unit.

As shown in figure(7) the input representation is 24 bit, 12bit for integer and 12 bit for fraction. The inputs vertices for test example are (f9c.000 h , 320.000 h) and (4b0.000 h ,f38.000 h) which are equivalent to (-100, 800) and(1200, -200). The input clipping window are (000.000 h , 000.000h) and (3ff.000h , 31f.000h) which are equivalent to (0,0) and (1023,799). The first step of the implemented unit is computing C1 and C2, as shown in the simulation C1 is 5 that mean V1 is out from the lift and down sides and C2 is A hex that mean V2 is out from the right and up sides. After that accept1 ,reject1 and enable is generated from C1 and C2 to accept the line if the two vertices is inside the window or reject the line if they are outside, The two signals are off as shown in figure (7) and enable signal is on to enable the clipping unit to compute the intersection of the vertices with clipping window to generate the new vertices. In the next clock the vertices are checked if they are outside from the left and right side and t1 and t2 and the new vertices are computed. However, the simulation of this step values in figure (7) are calculated theoretically according to the clipping algorithm for comparison. Because V1 is out from the left side So t1 is calculated according to the equation (3) to be 000.13b hex which equivalent to 0.0769043 and then by using t1, yn1 is calculated to be 2d3.188 hex which is equivalent to 723.096 , and hence the new vertex is (0,723). On the other hand V2 is out from the right side So t2 is calculated according to the equation (4) to be fff.cc2 hex which equivalent to -0.13623 and then by using t2, yn2 is calculated to be fc0.3b0 hex which is equivalent to -63.7695. In the next clock the vertices are checked if they are outside from the down and up side and if the intersection is necessary after extra test is performed t1 and t2 and the new vertices are computed. Because yn1 < Ymax so the second intersection is not necessary although V1 is outside from down side and because V2 is outside from the up and yn2 < Ymin so the second intersection is necessary to compute the correct vertex. The simulation of the this step values in figure (7) are calculated also theoretically for comparison. So t2 is calculated to be fff.ccd hex which is equivalent to - 0.199951 and then by using t2, xn2 is calculated to be 3ac.104 hex which is equivalent to 940.064 and hence the new vertex is (940,0).

The final step of the designed unit is computing the addition of the absolute value of t1 and t2 and comparing it with 1 to decide if the clipped line is accepted or not and set the accept1 signal as shown in the simulation figure.

Figure(8) shows the simulation wave forms for a second example executed by the implemented hardware unit.

In figure(8) the inputs vertices for test example 2 are (064.000 h ,0c8.000 h) and (258.000 h ,028.000 h) which are equivalent to (100, 200) and(600, 40), The input clipping window are (000.000 h , 000.000h) and (1ff.000h , 1ff.000h) which are equivalent to (0,0) and (511,511), As shown in the figure C1 and C2 are computed in the first step and their values are 0 and 2 respectively, That means V1 is inside the clipping window and V2 is outside the clipping window from right side, After that accept1 ,reject1 and enable are generated

form the value of C1 and C2 to accept the line if the two vertices are inside the window or reject the line if they are outside, The two signals are off as shown in figure (8) and enable signal is on to enable the clipping unit to compute the intersection of the vertices with clipping window to generate the new vertices, Then the clipping unit checks vertices if they are outside from the left and right sides, As shown from the value of C1 ,V1 is inside the window so the vertex doesn't require to compute the intersection of its coordinate and the parameter value (t1) so t1 is equal to zero and the values of xn1 & yn1 are loaded by the input vertex V1(064.000 h ,0c8.000 h), On the other hand the second vertex is clipped because C2 is 2 which means that V2 is outside from the right side so t2 , xn2 and yn2 are calculated to determine the correct clipped vertex, After the applying equation (4) to compute the value of t2 which is equal to fff.d27 h which is equivalent to -0.17797 and then compute the value of xn2 which is equal to 1ff.000 h which is equivalent to 511 and finally computes the value of yn2 which is equal to 044.7a0 h which is equivalent to 68.4766.

In order to test the designed unit in complete graphic system the clipping unit is attached to the scan conversion graphical sub-system[16]. So the complete graphic system consist of five parts ,display list memory ,line clipping unit(which is designed in our paper) ,graphic controller, frame buffer and refresh controller.

The main function of the graphic controller is to resolve each line into its constituent pixels and store them into the frame buffer memory. The scan conversion unit needs to clip these lines to the required screen before they are being scan converted. This can be achieved by the designed unit. The vertices values of the displayed lines are downloaded to the graphic system using JTAG bus and after that these values are stored in the display list memory, and then the clipping unit read tow vertices from the display list memory to computes the new vertices of the clipped line and give the start signal to the graphic controller to begins the scan conversion operation to produces pixels which are stored in the frame buffer from which they are then taken by refresh controller, using read cycles, for the display operation.

Figure (9) shows the results scenes that produced by complete graphic system. As shown there are two simple scene each one contain simple object consist of multiple line with red background, in the first scene all lines inside the window and no clipping process is required. On the other hand when the same object is generated with shift in X and Y coordinate as shown in the second scene, the clipping process is necessary to clip all lines that lie outside the window to perform the display operation of the scene correctly.

Performance and conclusions

The execution time is considered one of the most important aspects of any real time graphic system. Usually the execution time of a graphic system is a function of the complexity of the load. The graphical load complexity can be measured by the total number of vertices or number of lines used in data base. To speed up the execution of the clipping algorithm implementation in hardware is required. In order to practically examine the clipping designed unit it is used to execute line clipping with 1000 lines entered to the implemented unit for high load testing. The time elapsed in this test is 43 μ seconds which means that the designed unit is able to clip (232524) lines per second. A line clipping

algorithm is designed in a way suitable to be implemented in hardware and can be adopted to clip the polygon where each polygon can be considered as collection of lines and each line is clipped individually with the designed unit then clipped polygon is enclosed by portions of the clipping boundaries. It is very obvious from designed hardware unit that the processing element of each vertex is implemented in parallel mode as shown in figure (5) and figure (6) so the performance of the designed unit is increased but on the other hand the hardware cost is increased too. When the reduction hardware cost is critical factor and very essential the designed unit can be adopted too because it is implemented in suitable way to execute the clipping operation sequentially on each vertex so the hardware cost is minimized to half but on the other hand the execution time is nearly duplicated.

References

1- Edward Angel, " Interactive Computer Graphic, A Top-Down Approach Using OpenGL " Third Edition 2003.
2- Donald Hearn and M. Pauline, "Computer graphic " ,Third edition, Prentice Hall International, Inc (1997).
3- Blinn J.F. "Jim Blinn's corner-a trip down the graphics pipeline: line clipping" Computer Graphics and Applications, IEEE ,Volume: 11, No: 1 page(s): 98-105 ,ISSN: 0272-1716 , Jan 1991.
4- Patrick G. M. "A new, fast method for 2D polygon clipping: analysis and software implementation" ACM Transactions on Graphics (TOG) Volume 11 , Issue 3, Pages: 276 – 290,ISSN:0730-0301, 1992.
5- Václav Skala , "Line clipping in E2 with O(1) processing complexity", Computers & Graphics Volume 20, No 4, Pages 523-530, July-August 1996.
6- Nishita T. , Johan H., "A scan line algorithm for rendering curved tubular objects" , Computer Graphics and Applications, Proceedings Seventh Pacific Conference, Page(s) :92 - 101 Digital Object Identifier 10.1109/PCCGA.1999.803352, Oct. 1999.
7- Mingjun Zhang, Chaman L. Sabharwal , "An efficient implementation of parametric line and polygon clipping algorithm" Symposium on Applied Computing Proceedings of the 2002 ACM symposium on Applied computing Madrid, Spain, Pages: 796 – 800 ,ISBN:1-58113-445-2, Year of Publication: 2002.
8- Skala V.I ,"A new line clipping algorithm with hardware acceleration" Computer Graphics International, Page(s):270 – 273, Digital Object Identifier 10.1109/CGI.2004.1309220, Proceedings 19-19 June 2004.
9- O'Toole A.J., Harms J., Snow S.L., Hurst D.R., Pappas M.R., Ayyad J.H., Abdi H., "A Video Database of Moving Faces and People" , IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 27, NO. 5, ,Page(s):812-816 ,Digital Object Identifier 10.1109/TPAMI, MAY 2005.
10- Yong Kui Liu , Xiao Qiang Wang , Shu Zhe Bao , Matej Gomboši , Borut alik," An algorithm for polygon clipping and for determining polygon intersections and unions", Computers & Geosciences, Volume 33 , No 5 , Pages 589-598 ISSN:0098-3004, May 2007.
11- Wang Jin , Lu Guo-dong, Peng Qun-sheng and Wu Xuan-hui ," Line clipping against polygonal window

algorithm based on the multiple virtual boxes rejecting", Journal of Zhejiang University SCIENCE , Volume 6 ,No 1, Pages:100-107, ISSN 1009-3095, 2005.

12- Mark S. Sobkow, Paul Pospisil and Yee-Hong Yang. "A Fast Two-Dimensional Line Clipping Algorithm via Line Encoding", Computer & Graphics, Vol. 11, No. 4, Pages. 459–467, 1987.

13- David F. Rogers " Mathematical element for computer Graphic", McGraw-Hill Inc (1997).

14- F.S. Hill, Jr. "computer graphic using OpenGL " second edition , Prentice Hall International , Inc (2001).

15- Fakhraldeen H.ALI , "A Concurrent Processing System For The Generation of Real-Time Three Dimension Graphics " , Ph.D thesis , Bradford University ,U.K , 1989.

16- Fakhiraldeen H. Ali ,Amar I. Dawod ," FPGA Design And Implementation Of A Scan Conversion Graphical Sub-System", Al-Rafidain Engg., Vol. 8, No. 2, 2007.

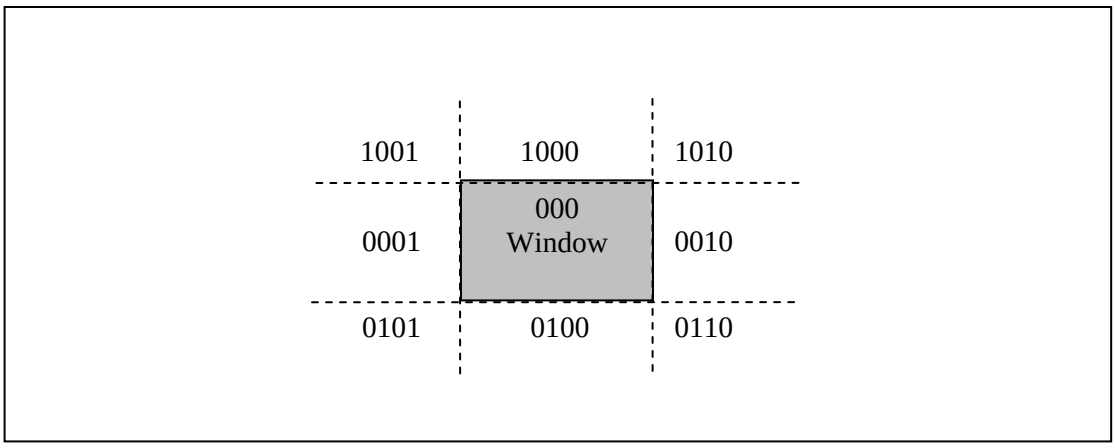
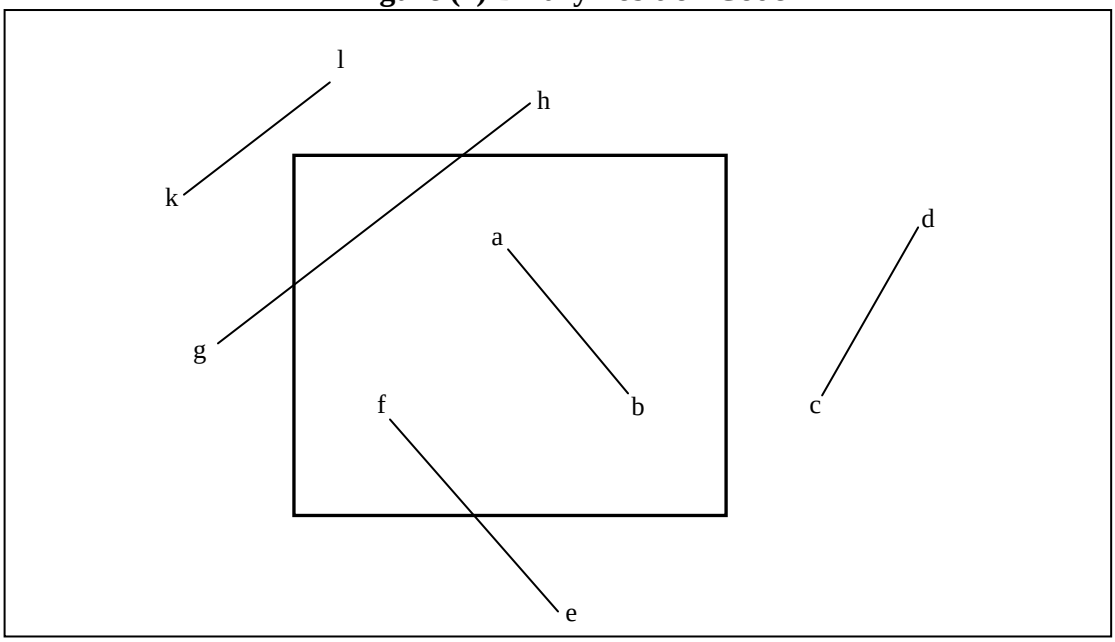


Figure (1) Binary Position Code



Figure(2) Different Orientations Of Line Segments

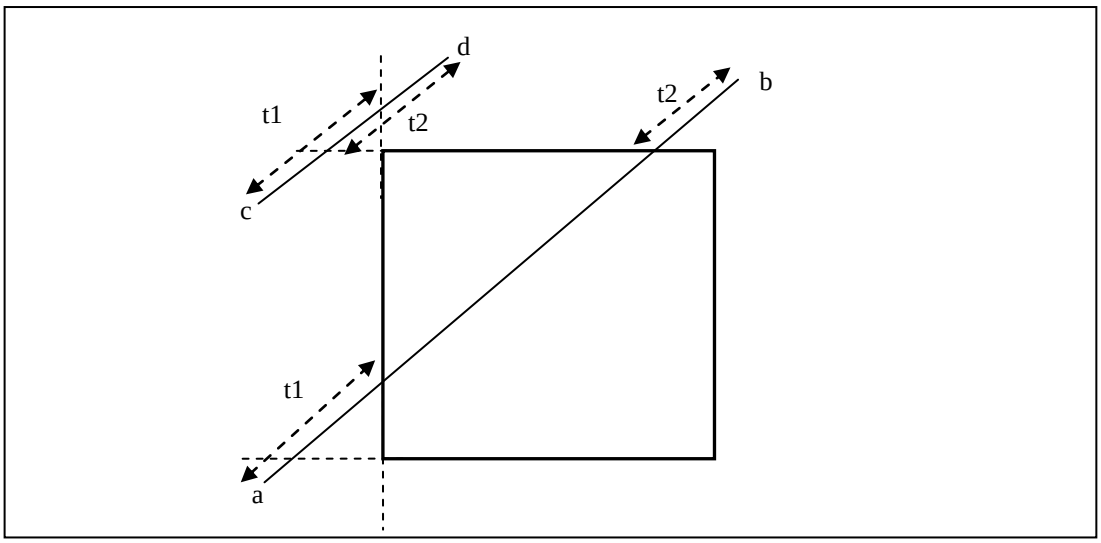
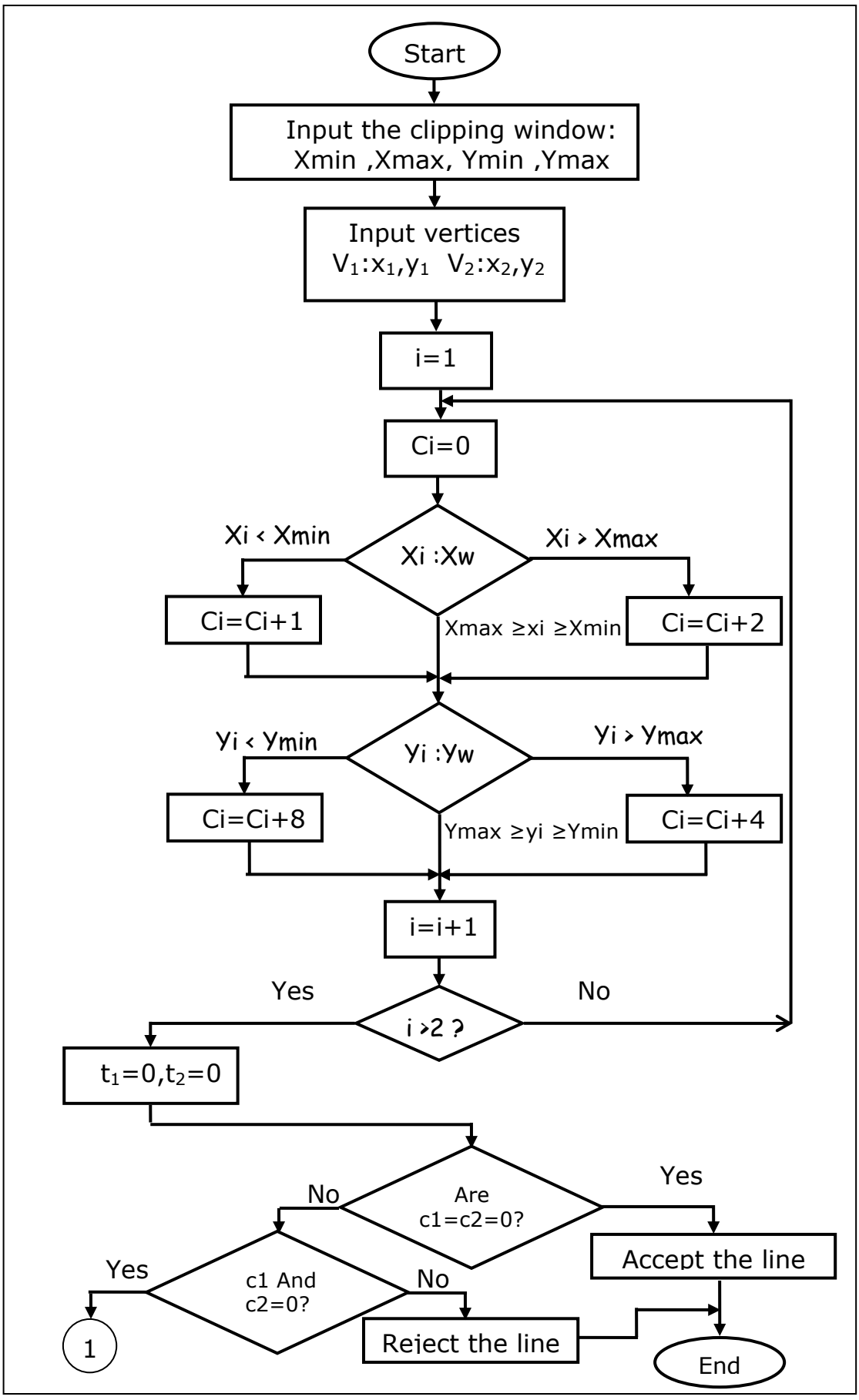


Figure (3) Line Segment And Their Parametrical Portions

Tikrit Journal of Eng. Sciences/Vol.18/No.3/September 2011, (89-105)



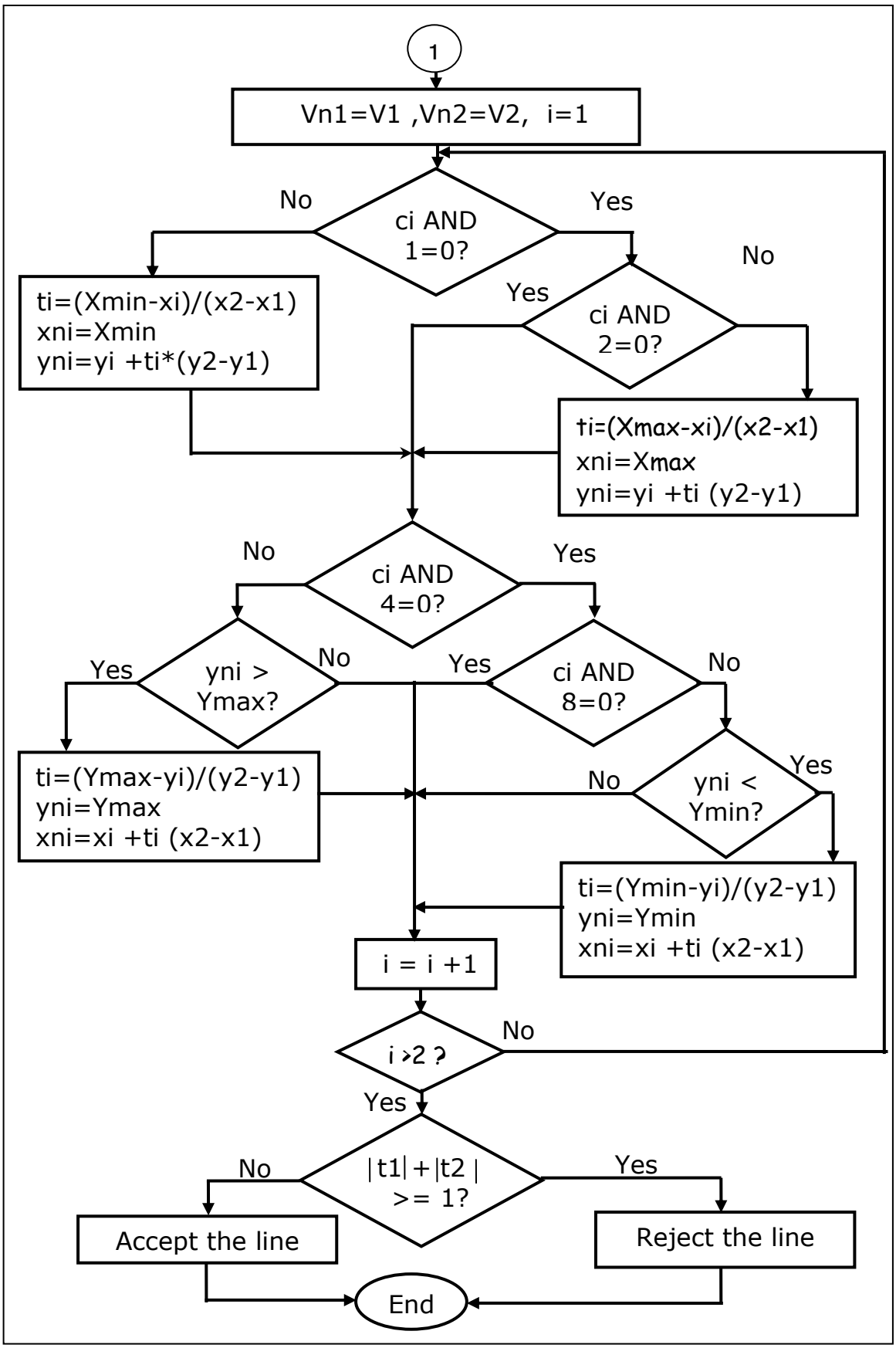


Figure (4)The Clipping Algorithm

Tikrit Journal of Eng. Sciences/Vol.18/No.3/September 2011, (89-105)

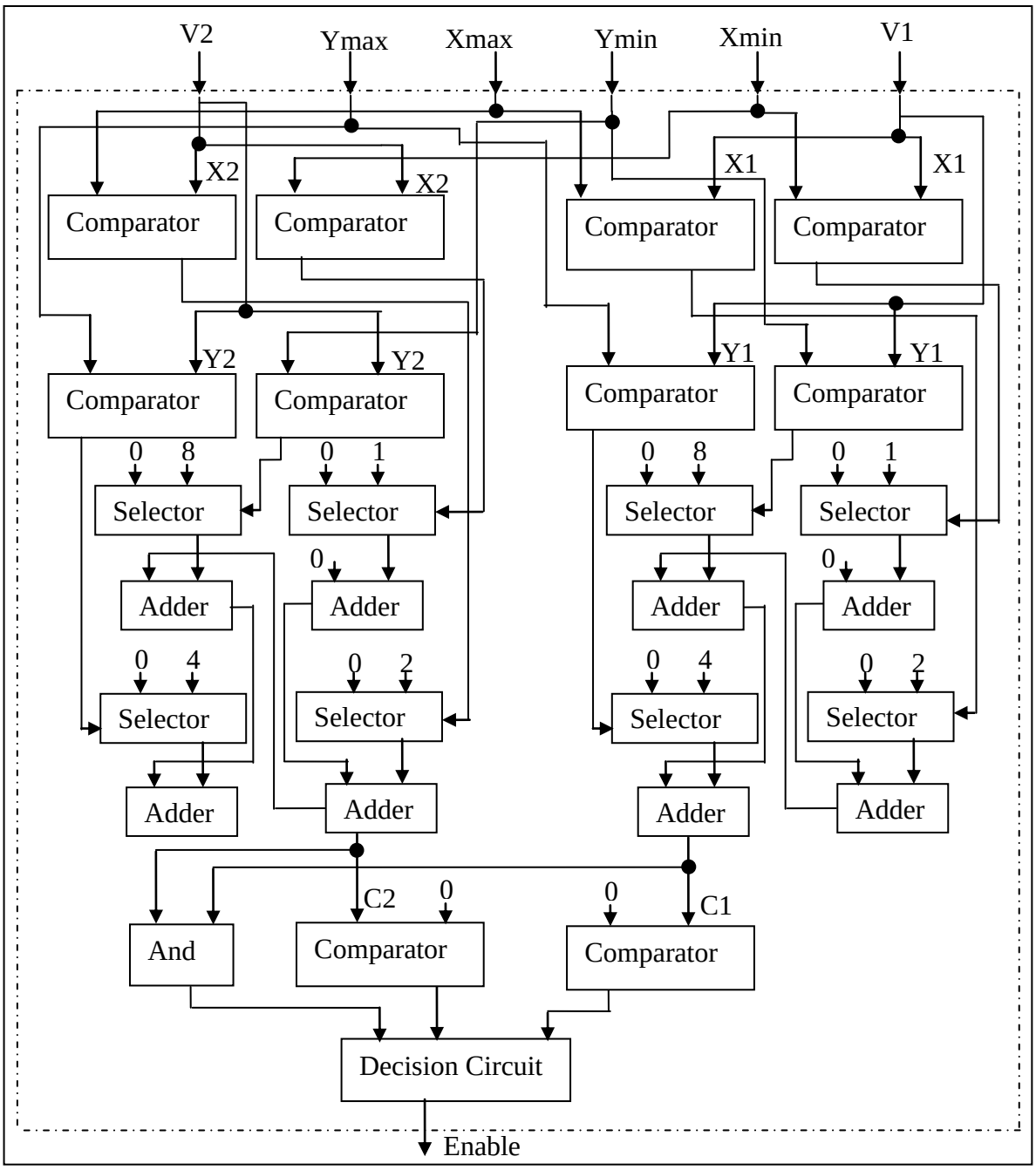


Figure (5)Positional Code Unit

Tikrit Journal of Eng. Sciences/Vol.18/No.3/September 2011, (89-105)

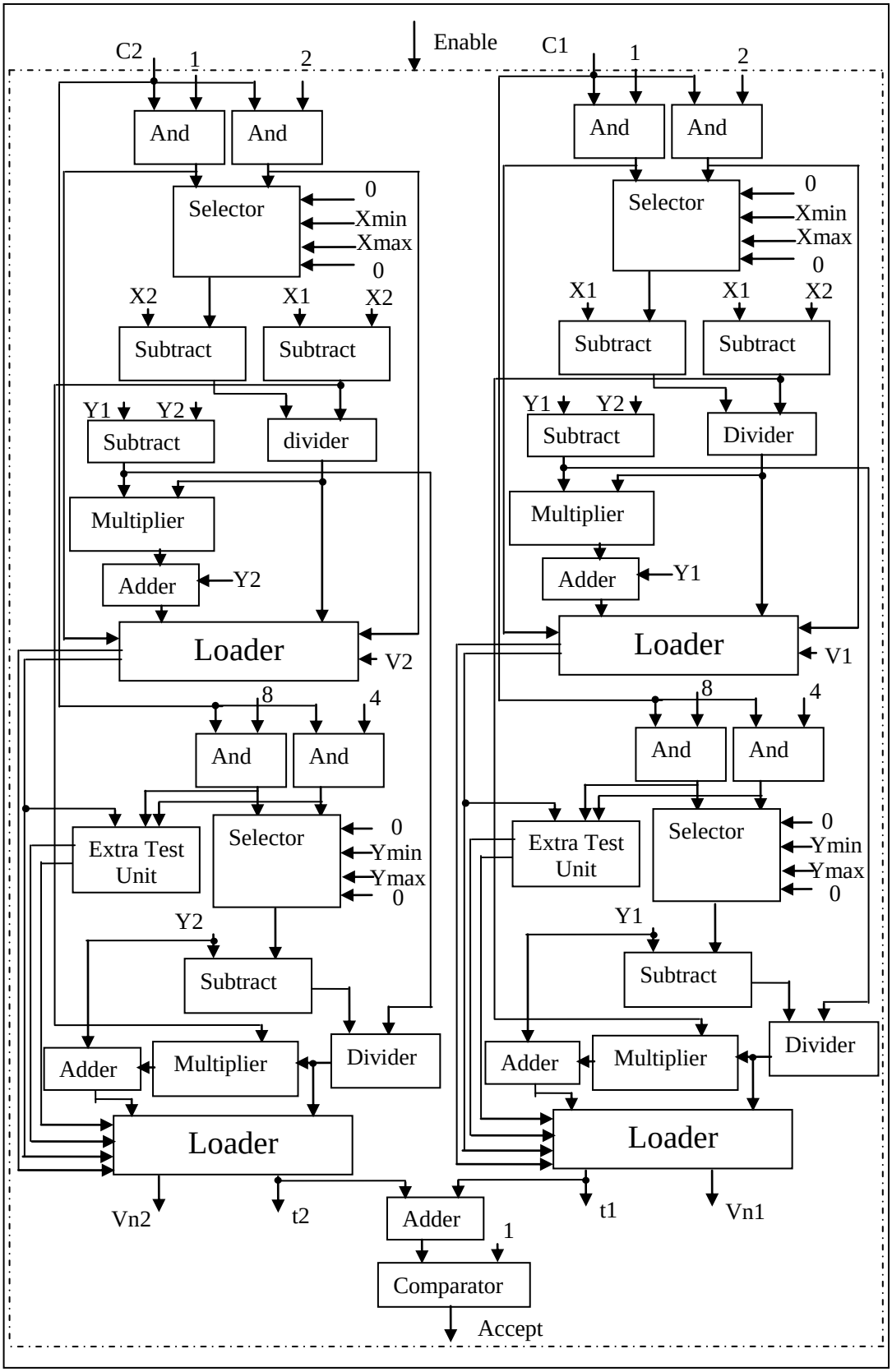
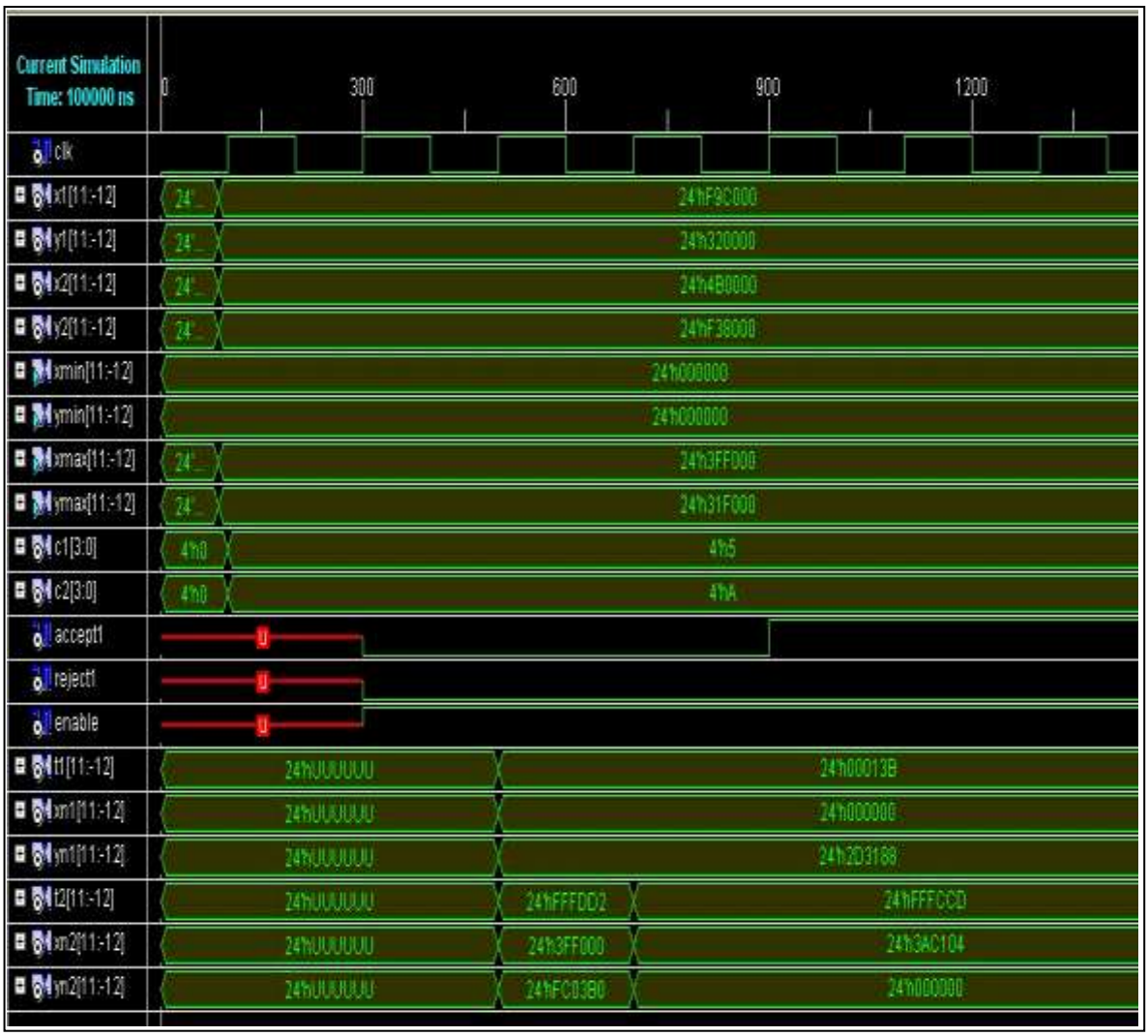


Figure (6) The Clipping Unit

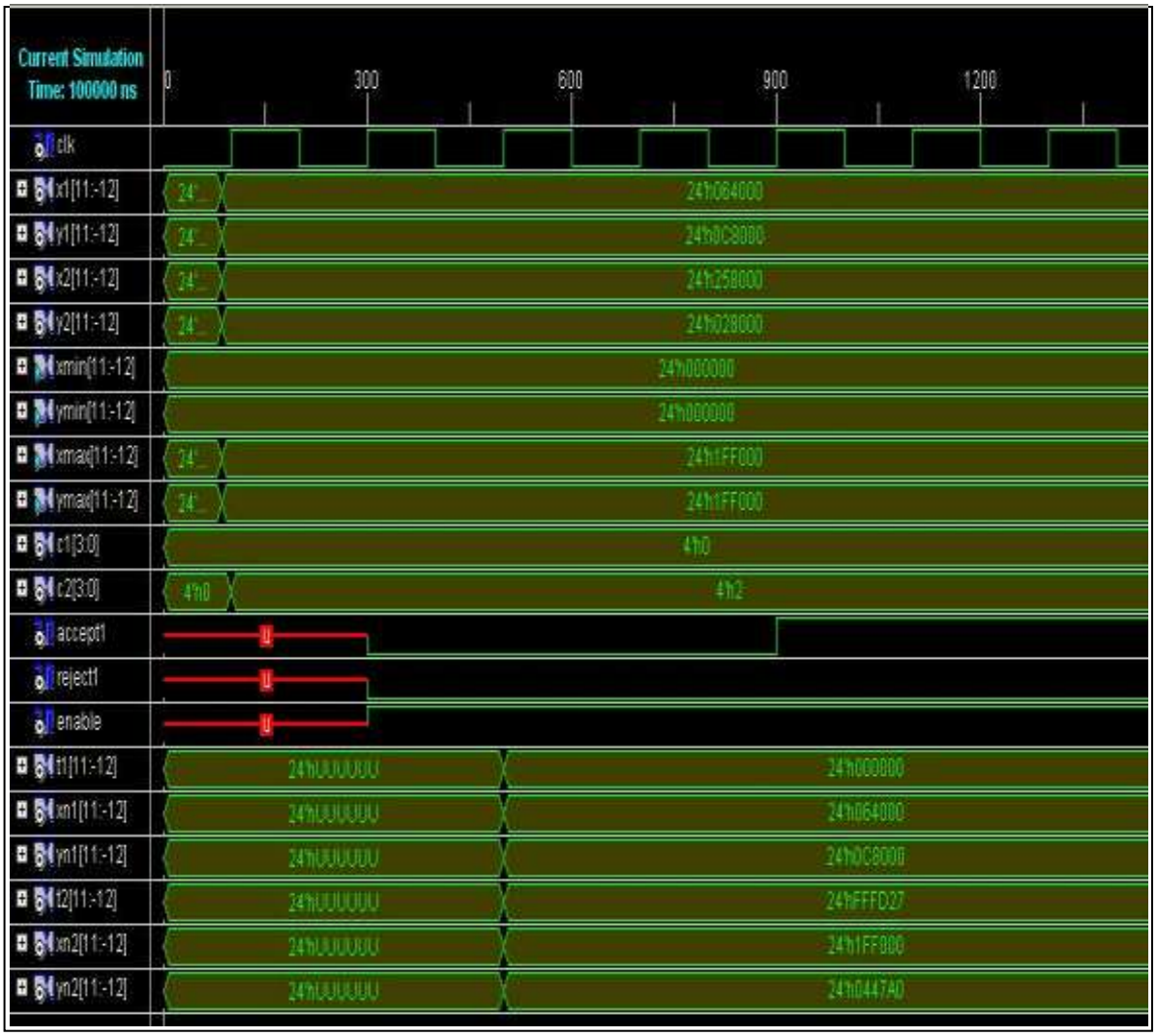
Tikrit Journal of Eng. Sciences/Vol.18/No.3/September 2011, (89-105)



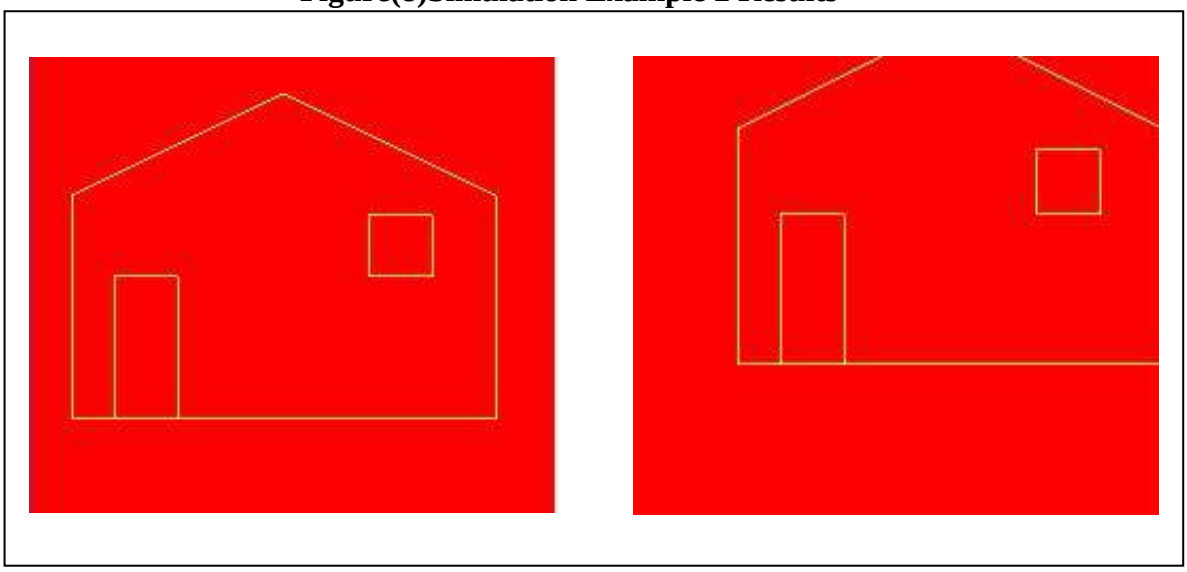
Figure(7)Simulation Results of Example(1)

Where:
x1, y1 & x2, y2: the first and second vertices.
Xmin & Ymin & Xmax & Ymax: Clipping window.
C1 & C2 : positional code for the first and second vertices.
Accept1: accept line signal.
Reject1: reject line signal.
Enable :enable signal.
t1&t2 :normalized parameters.
xn1, yn1 & xn2, yn2: the new vertices.

Tikrit Journal of Eng. Sciences/Vol.18/No.3/September 2011, (89-105)



Figure(8)Simulation Example 2 Results



Figure(9) Generated Scenes

Table(1) Resources Utilization Of the Implemented Unit

Type Resources (or Frequency)	**Utilized Resources**	**Total Resources**	**Ratio**
Number of Slices	2467	4656	53%
Number of Slices Flip Flops	209	9312	2%
Number of 4 input LUTs	3724	9312	40%
Number of Bounded IOBs	144	232	31%
Number of Block RAMS	0	20	0 %
Number of MULT18X18s	8	20	40%
Number of GCLKs	1	24	4%
Maximum Operating Frequency	116.262 MHZ		