Transformation Matrix for 3D computer Graphics Based on FPGA

Dr. Fakhrulddin H. Ali
Dept. Of Computer Engineering /University of Mosul - IRAQ
Email: fhali310@yahoo.com

Abstract

The real time of the computer graphics system performance is one of the fast many computing applications. The 3D (three-dimensional) geometric transformations are one of the most important principles of interactive computer graphics, which are essential for modeling, viewing and animation. This paper tends to construct a general form of a single matrix representation for multiple geometric transformations for three-dimensional objects. This way, a speed up factor of 1 to 5 can be gained. Architecture is designed, and implemented as a hardware unit, and then tested for the single matrix transformation, using Field Programmable Gate Array (FPGA).

Keywords: 3D graphics, lookup table, FPGA, concatenation.

مصفوفة تحويلات الصور الحاسوبية للتطبيقات الثلاثية الأبعاد باستخدام البوابات القابلة للبرمجة حقليا

د فخر الدين حامد علي
قسم هندسة الحاسبات في كلية الهندسة / جامعة الموصل ـ العراق

الخلاصة

إن متطلبات سرعة الأداء في مجال الرسوم الحاسوبية وتطبيقاتها في حالة تزايد مستمر حيث أن التحويلات الهندسية والتحريك في الفضاء ثلاثي الأبعاد تعتبر من إحدى أساسيات النمذجه والرؤيا وتوليد الصور المتحركة بالحاسوب . يقدم هذا البحث طريقة للتعبير عن عدد من التحويلات الهندسية للصور وتحريكها بتحويل واحد معبر عنه بمصفوفة عامة واحدة اختصارا حيث يؤدي ذلك إلى تسريع الأداء بمقدار 1 إلى 5 مرات .تم تصميم معمارية بالكيان المادي وتنفيذها للغرض أعلاه باستخدام مصفوفة البوابات القابلة لإعادة البرمجة حقليا.

Received: 15 – 6 - 2011 **Accepted: 31 – 10 - 2011**

1 Introduction

With procedures for displaying output primitives and their attributes, a variety of pictures and graphs can be created. In many applications, there is also a need for altering or manipulating objects, or parts of them, where design applications and facility layouts are created by arranging the orientations and sizes of the component parts of a scene. On the other hand, animations are produced by moving the camera or the objects in a scene along an animation path. Changes in orientation, size, and shape are accomplished with geometric transformations that alter the coordinates description of objects. The basic geometric transformations are translation, rotation and scaling [1,2]. The challenge of producing realistic images of animation scenes is the speed with which these transformations are applied [3]. This requires fast execution of addition, multiplication, and trigonometric which attracted and still do many research work and different architectures as presented by the following review.

In 1990 the researchers Hai Nang Lin and Henk J. Sips proposed a fast way CORDIC implementation for calculations of a number of arithmetic basic function. The paper discusses the speed that is limited by the carry propagation in the adders and the I/O throughput. It stated that Speed can be improved by introducing redundancy in the calculation circuit and throughput by doing I/O transfers while calculating [4]. In 1993 the researchers Dwight Hill and Nam-Sung Woo discussed the utility of allowing each block's single large table (e.g., one 5-input, 32-bit table) to be reconfigured into smaller tables (e.g., eight 4-bit tables). Results describing the efficiency of packing some standard benchmark circuits into various configurations are presented and the cost/benefits are discussed. The paper shows that a logic block containing four lookup tables, each of which is 8-bit RAM, which is the best choice if only the area efficiency is considered. The researchers also show that if the circuit speed is considered, a logic block containing two lookup tables, each of which contains 16 bits of RAM, is the best choice [5]. In 1999, John N. Lygouras described a new technique for lookup table implementation using linear interpolation to achieve Memory Reduction in Look-Up Tables. This paper describes a new hardware technique, providing high-resolution trigonometric functions, the sine and cosine for an angle θ from a significantly reduced size lookup table (LUT). The method takes advantage of the symmetries of these trigonometric functions around several axis and the fact that the cosine function can be derived by shifting the sine function by $\pi/2$ backward. This shifting is achieved using a very fast hardware technique. A linear interpolation technique can be used if a further reduction in memory is desired. The described system can be used for function generators, robotic arm controllers, position control systems and others [6].

In 2000 Kharrat M.W., Loulou M., Masmoudi N. and Kamoun L. introduced a paper for an optimization of CORDIC algorithm implementation, which mainly offers a silicon area occupation reduction and gives good precision in calculating trigonometric functions such as sine and cosine function. To validate or test the new method, an implementation of angle decomposition equation using FPGA technology is presented. This approach shows a considerable surface reduction and good precision for calculation of a resolution less than 20 bits [7]. In 2001 the researchers Bernard Tiddeman and David Perrett described a new method for creating visually realistic moving facial image sequences that retain an actor's personality (individuality, expression and characteristic movements) while altering the facial appearance along a certain specified facial dimension. The paper combines two existing technologies, facial feature tracking and facial image transformation, to create the sequences.

The paper also create 'virtual cartoons' by transforming image sequences into the style of famous artists [8]. In 2003 the researcher Ali M. A. Abbas introduced a Hardware Implementation of Transformation of Rendering Algorithms. The proposed algorithms is first simulated in software using C++ then transformed to hardware design, specified in VHDL and simulated in Model-Technology environments assuming the delay times of a real FPGA device. The results demonstrate that, these hardware schemes could provide appropriate pixel drawing time, enough for real-time rendering [9]. In 2005 the researchers Bensaali, F., Amira, A., Uzun I.S. and Ahmedsaid A. Introduced a paper investigating the suitability of Field Programmable Gate Array (FPGA) devices as a low cost solution for implementing 3D affine transformations. A proposed solution based on processing large matrix multiplication has been implemented, for large 3D models, on the RC1000-PP Celoxica board based development platform using Handel-C, a C-like language supporting parallelism, flexible data size and compilation of high-level programs directly into FPGA hardware [10]. In 2007 Faycal Bensaali , Abbes Amira and Reza Sotudeh described field-programmable gate arrays in implementing floating-point arithmetic. In this paper, a floating-point adder and multiplier are presented. The proposed cores are used as basic components for the implementation of a parallel floating-point matrix multiplier designed for 3D affine transformations. The cores have been implemented on recent FPGA devices. The performance in terms of area/speed of the proposed architectures has been assessed and has shown that they require less area and can be run with a higher frequency when compared with existing systems [11].

However, this article summaries the subject and introduced the current work. The rest of this paper is organized as follows: article number (2) introduces the geometric transformations, article number (3) deals with matrix concatenation. In article (4) the trigonometric function evaluation is presented. Matrix multiplication is in article (5). The implementation and testing are in article (6). The Conclusions and performance evaluation are stated in article (7). Finally, the references are stated at the end.

2 Geometric transformations

Many graphics applications involve a sequence of geometric transformations. Fundamental to all computer graphic systems is the ability to simulate both the movement and the manipulation of images in a scene. These processes are described in terms of translation, scaling, and rotation. They applied to each individual vertex and repeated to all object vertices to achieve the required image transformation. These operations are described in a mathematical form, which is suitable for computer processing to achieve the image manipulation and motion [3, 12]. Translation is applied to an image by repositioning it along a straight-line path from one coordinates location to another. A single vertex is translated by adding a translation distance; tx , ty and tz to the original coordinate position of the vertex V(x,y,z) to move it to a new position V(x', y', z') where[13] :

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & tx \\ 0 & 1 & 0 & ty \\ 0 & 0 & 1 & tz \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad (1)$$

Where: tx , ty, and tz are the translation constants

Scaling transformation alters the size of an image. The operation can be carried out by multiplying the coordinate value V(x,y,z) of each vertex by scaling factors sx, sy and sz to produce the transformation as shown by equation number (2) below [13].

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} sx & 0 & 0 & 0 \\ 0 & sy & 0 & 0 \\ 0 & 0 & sz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad (2)$$

Where: sx, sy, and zs are the translation constants

The rotation is applied to a vertex by repositioning it along a circular path in the xy plane in a clockwise or anti-clockwise direction. Rotation around the z-axis can be accomplished by the following equation [1,13]:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad (3)$$

Where: $z' = z$ and θ is the angle of rotation (+ve for anticlockwise rotation)

To generalize the rotation to be about any arbitrary axis of rotation (parallel to the z-axis), three matrices can be concatenated to a single one to accomplish that. The first matrix translates the vertex to be rotated by tx = - rx and ty = - ry. The second matrix rotates the translated vertex about the z-axis. And finally the third matrix translates it back by tx = rx and ty = ry. The three matrices are multiplied and the results are presented using a single transformation matrix in equation (4).

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & a \\ \sin\theta & \cos\theta & 0 & b \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad (4)$$

Where:

$a = -rx \cos\theta + ry \sin\theta + rx.$

$b = -rx \sin\theta - ry \cos\theta + ry.$

3 Concatenation (rotation, scaling, and translation)

To produce a sequence of transformations such as, for example, rotation followed by scaling then translation, the coordinates must be calculated one step each time. First, coordinate positions are rotated, then these rotated coordinates are scaled, and finally the outcome coordinates are translated. On the other hand, a more efficient approach would be to combine the transformations so that the final coordinates positions are obtained directly from the initial or original coordinates, thereby eliminating the calculation of intermediate

coordinates values so that all transformations can be expressed as a single transformation matrix as described in equation (5).

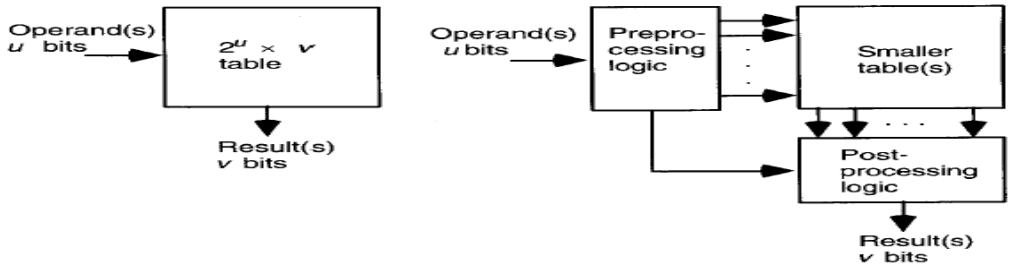
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} sx\cos\theta & -sx\sin\theta & 0 & sx^{*}a+tx \\ sy\sin\theta & sy\cos\theta & 0 & sy^{*}b+ty \\ 0 & 0 & sz & tz \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad (5)$$

4 Trigonometric function evaluation

The great challenge in geometric transformation is the sine function evaluation required for rotation. There are several methods for computing this function (the sine or cosine). One of the techniques is evaluating their series expansion by means of addition, multiplication and division operations. The hardware implementation of the sine function by their series is complicated due to the accuracy requirement of considering a number of terms in the series expansion. There are other techniques that are used currently in wide area of applications to evaluate trigonometric functions, among these methods are the lookup table and the CORDID algorithm [14].

4.1 Look up table method

High-speed approximations of the sine and cosine functions are often used in digital signal or image processing and even in digital control. Computation using a look up table is an attractive method because the required memory is available much denser than random logic in VLSI realization. Multi-megabit look up tables are already practical in some applications, even larger tables should become practical in the near future as memory density continues to improve. The use of tables reduces the cost of hardware development (design, validation, and testing), provides more flexibility for last-minute design, and reduces the number of different building blocks or modules required for arithmetic system design [14]. Tables stored in read only memory (especially if individual entries or blocks of data are encoded in error-detecting or error- correcting codes) are more robust than combinational logic circuit, thus leading to improved reliability. With read/write memory and reconfigurable peripheral logic, the same building block can be used for evaluating many different functions by simply reloading appropriate values in the table. This feature facilitates maintenance and repair. There are more than one technique to implement the look up table, one of them is a direct look up table. The direct look up table evaluation of trigonometric function requires the construction of a $2^{u} * v$ table (u is the address and v is the output depth) that holds for each combination of input values (requiring a total of u bits) the desired v-bit result (the sine value). The u-bit string obtained from concatenating the input values is then used as an address into the table with v-bit value read out from the table directly forwarded to the output as shown in figure(1). Such arrangement is quite flexible but some times it is required to reduces the table size especially in larger representation of a trigonometric function in some application so an advantage of the symmetries of these trigonometric functions around several axis can be considered. One solution is to reduce the table size by storing in the table half wave or quarter wave instead of full wave of a sine or cosine values and applying preprocessing steps to the input of the table and post processing to the output value from the table. This approach is called indirect look up table as in figure (1) [15].



Direct access **Indirect access**

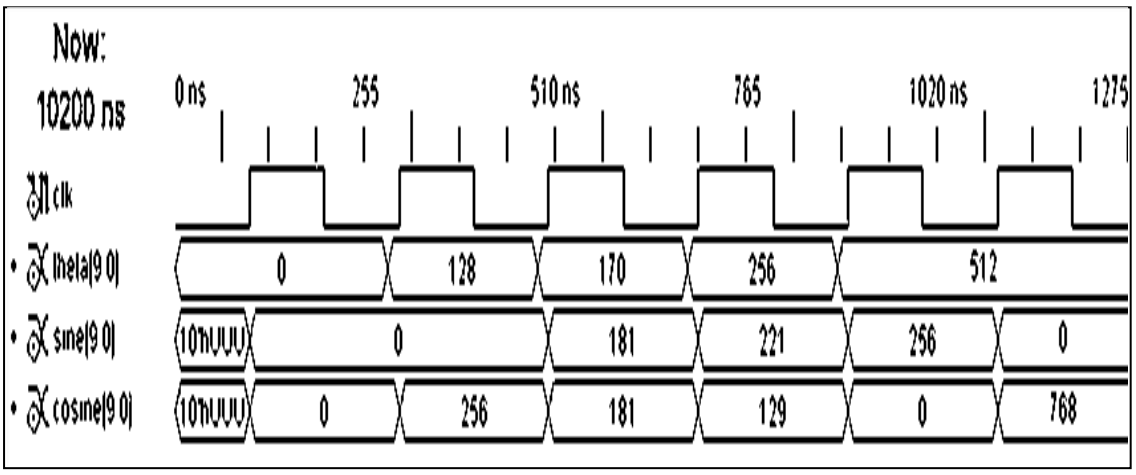
Figure(1): Direct and indirect look up table

Figure (2) shows the simulation waveforms for an example executed by the implemented lookup table in a direct mode (because the current application doesn't require high representation) for 10-bit input (theta), and 10-bit output (sine & cosine), two bits for integer (1 bit for sign), and eight bits for fraction (fixed point representation). The output (sine or cosine) is produced after one clock. Equation (6) defines the relationship between the integer input angle theta and the actual radian angle Θ [16].

Θ = theta (2Π / theta _width) (6)

Where theta _ width is (2 ˆ 10) = 1024

Examining the waveforms**,** the first input of theta is zero so the output of sine value is zero and the output of cosine is 256 (01.00000000) which is equivalent to 1. The second input is 128 (45 degrees) so the output of sine and cosine is 181 (00. 10110101) which is equivalent to 0.70703125.



Figure(2): Simulation example results using lookup table

4.2 CORDIC algorithm

The Co-ordinate Rotation Digital Computer (C0.R.DI.C) algorithm is an iterative procedure to evaluate various elementary functions. It was introduced in 1959 by Volder and it is still interesting to many researchers due to its simple hardware structure. The CORDIC algorithm is capable of evaluating many elementary trigonometric functions such as Sine and Cosine. CORDIC is an iterative procedure for the calculation of the rotation of a two-

dimensional vector, in linear, circular or hyperbolic coordinate systems, using only add and shift operations [17]. Its current applications are in the field of digital signal processing, image processing, filtering, matrix algebra, etc. The simple form of CORDIC is based on the observation that if a unit length vector with an end point position at (x , y)=(1 , 0) is rotated by an angle z, it's new end point position will be at (x' , y') = (cos z, sin z). Thus, cos z and sin z can be computed by finding the coordinates of the new end point of the vector after rotation by z [14]. The CORDIC algorithm consists of two operating modes, the rotation mode (suitable for sine and cosine) and the vectoring mode, respectively. In the rotation mode, a vector (x , y) is rotated by an angle θ to obtain the new vector (x', y'). In every micro rotation i, fixed angles of the value arctan (2^{-i}) which are stored in a ROM are subtracted or added from/to the angle remainder θi, so that the angle remainder approaches zero. In the vectoring mode, the length R and the angle towards the x-axis α of a vector (x , y) are computed (see figure 3). For this purpose, the vector is rotated towards the x-axis so that the y-component approaches zero. The sum of all rotation angles is equal to the value of α , while the value of the x-component corresponds to the length R of the vector (x, y). The mathematical relations for the adopted rotation mode are given by equations (7), (8), and (9) [17].

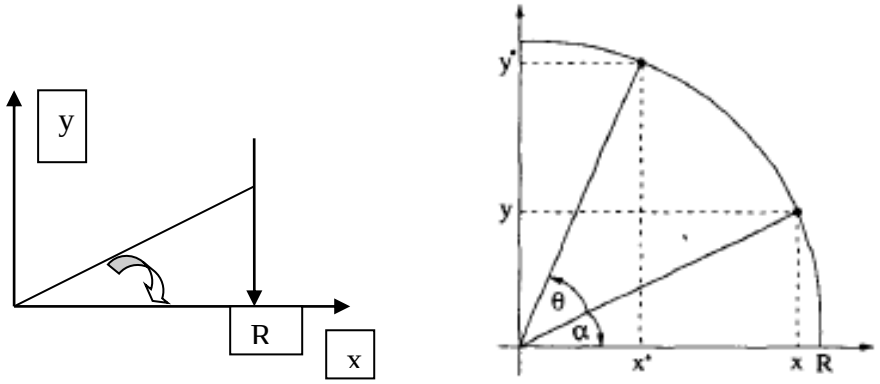


Figure (3): The vectoring (left) and rotation mode (right) of the CORDIC algorithm

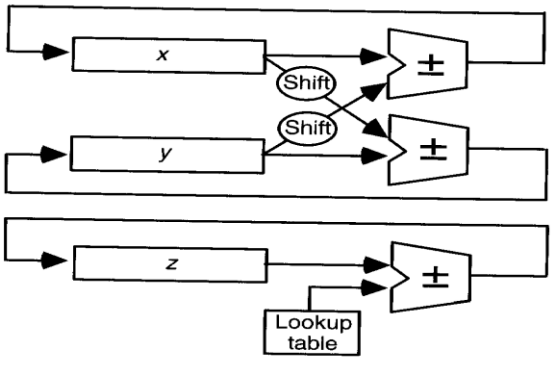
$$x^{(i+1)} = x^{(i)} - d_i\, y^{(i)}\, 2^{-i} \quad (7)$$

$$y^{(i+1)} = y^{(i)} - d_i\, x^{(i)}\, 2^{-i} \quad (8)$$

$$z^{(i+1)} = z^{(i)} - d_i \tan^{-1} 2^{-i} \quad (9)$$

i = step index.

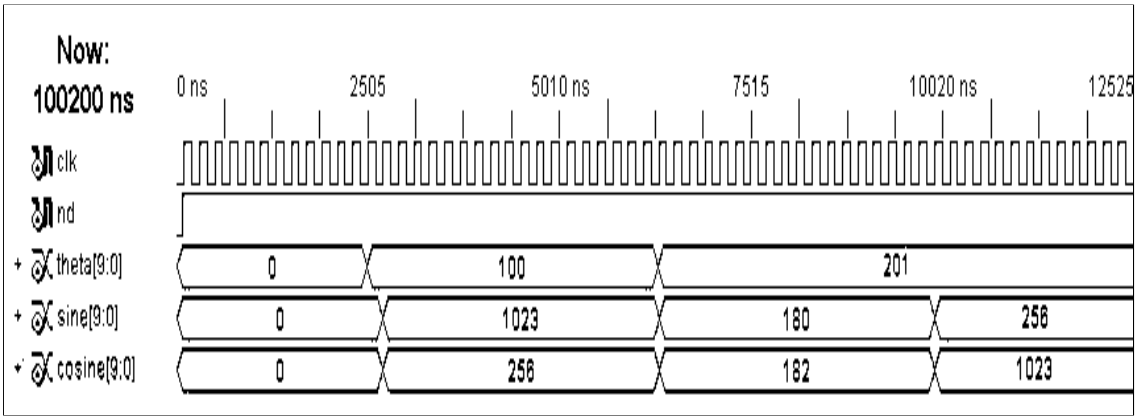
The computation of $x^{(i+1)}$ or $y^{(i+1)}$ requires an i-bit right shift and an add/subtract. If the function ($\tan^{-1} 2^{-i}$) is pre computed and stored in a table for different values of i, a single add/subtract suffices to compute $z^{(i+1)}$. Each CORDIC iteration thus involves two shifts, a table lookup, and three additions[14]. The hardware implementation for CORDIC arithmetic is shown in figure (4). It requires three registers for x, y, and z, a lookup table to store the values of ($\tan^{-1} 2^{-i}$) and two shifters to supply the terms 2^{-i} x and 2^{-i} y to the adder/subtract units. The di factor (-1 or 1) is accommodated by selecting the (shifted) operand or its complement.



Figure(4): Hardware representation for CORDID method

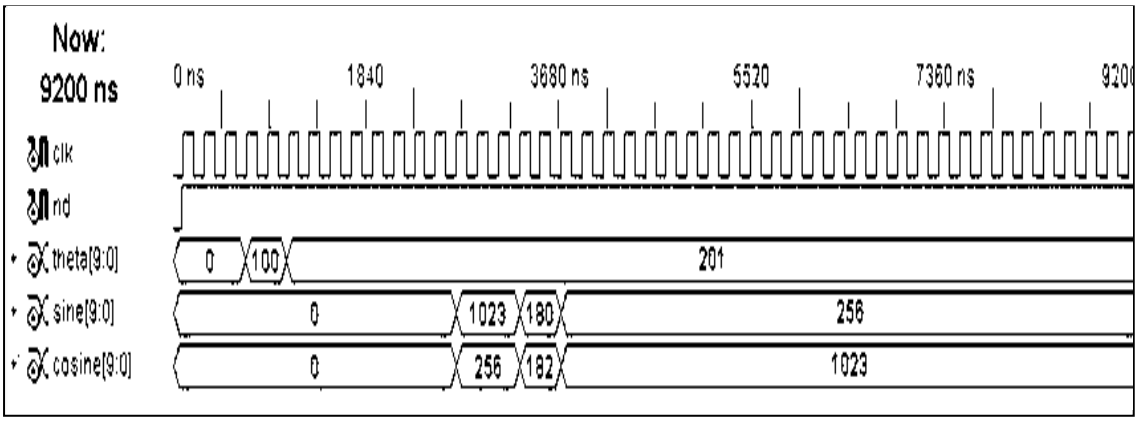
There are several techniques to implement the CORDIC algorithm, it can be implemented using only one stage as shown in figure (4) and so the silicon area is reduced but each output appears after n clock (where n is the input width) and this technique is called serial word. On the other hand it can be implemented using n stages (repeating the hardware) and applying the pipeline so the first output only appears after n clock and the other outputs each appears after a single clock successively. This method is called parallel pipeline word.

Figure (5) shows the simulation waveforms for an example executed by the implemented serial word CORDIC where the input width of theta is 10-bit, 3-bit for integer (1 bit for sign) and 7-bit for fraction to represent the input range from Π to – Π, and 10-bit for output (sine & cosine). Two -bits for integer (1 bit for sign) and 8-bits for fraction (fixed point representation) [16]. In figure (5) the first input is zero and the output of cosine is 256 (01.00000000) which is equivalent to 1, and the output of sine is 1023 (11.11111111) which is equivalent to -0.00390625 (approximately zero). The second input is 100 (000.1100100) which is equivalent to 0.781 (Π /4) and the output of sine is 180 (0010110100= 0.703) and the output of cosine is 182 (00.10110110 = 0.711).



Figure(5): CORDIC simulation results of a serial word example

Figure (6) shows the simulation waveforms for an example executed by the implemented parallel pipeline word CORDIC. The first output appears after 10 clocks and the others each after one clock.



Figure(6): CORDIC simulation results of a parallel pipeline word example

5 Matrix multiplication

Matrix multiplication is required to compute new vertices from old vertices multiplied each by the concatenation matrix The multiplication operation can be implemented using several techniques, one of these techniques is array processors in a parallel mode where each processor computes one element of the result matrix. Highly parallel computing structures become the major application area for multimillion transistor chips. Such computing systems have structural properties that are suitable for VLSI implementation. The matrix vector product can be described by the equations shown below:

$$C = A.B \tag{10}$$

$$C_i = \sum_{K=1}^{n} a_{ik} b_k \qquad \text{for } 1 \le i \le m, \qquad n = m = 4 \tag{11}$$

$$\begin{bmatrix} C1 \\ C2 \\ C3 \\ C4 \end{bmatrix} = \begin{bmatrix} a11 & a12 & a13 & a14 \\ a21 & a22 & a23 & a24 \\ a31 & a32 & a33 & a34 \\ a41 & a42 & a43 & a44 \end{bmatrix} \begin{bmatrix} b1 \\ b2 \\ b3 \\ b4 \end{bmatrix} \tag{12}$$

Figure (7) shows the designed unit for implementing matrix multiplication using array-processing elements.

6. Implementation and testing

The Field Programmable Gate Array (FPGA) is a new approach to ASIC design that can dramatically reduces manufacturing turn around time and cost. An FPGA consists of a regular array of programmable logic blocks that can be interconnected by a programmable routing network [16]. So using this technique the circuit hardware can be implemented. Figure (8) shows a block diagram of the implemented unit. The inputs of the unit are the parameters of the transformation matrix in addition to the input vertices, and the output is the vertices produced after transformations. The matrix coefficients calculating unit is responsible for computing the elements of the transformation matrix and loading the registers. Figure (9) shows the simulation waveforms of example1 and figure (10) shows the simulation waveforms of example2 executed by the unit for a concatenation matrix (implementing rotation, scaling, and translation) defined by equation (13) and using a lookup table to implement the trigonometric functions.

As shown in figure (9), the input representation is 16 bit, 8bit for integer and 8 bit for fraction, where Xin is set to 02.00h,, and Yin=02.00h in the example, Zin=02.00h, sx=0f.00h, sy=0a.00h, sz=0a.00h, tx=05.00h, ty=0f.00h, tz=04.00h, rx=00.00h, ry=00.00h. (rx and ry are zero, for rotation about the z axis), and for lookup table the input is theta=080h (128d = 45 degree). The first task of the matrix coefficients calculation unit is addressing theta to the lookup table to determine the sine and cosine value (0.b5=00.10110101) which is equivalent to 0.70703125. After that the calculation of the coefficients begins, ($sx * \cos\theta$) is calculated as c00 which is equivalent to 10.60546875 and ($- sx * \sin\theta$) is calculated as c01 to be (000a.9b00h) then truncated and converted to negative (2's complement) as Cm01 (f5.65h)

which is equivalent to -10.60546875. In the second row where (sy * sinθ) is calculated as c10 (0007.1200h) then truncated as Cm10 (07.12) which is equivalent to 7.078125. After that the new coordinates are computed by applying the matrix multiplication between the input matrix of vertices and the transformation matrix (refer to equation (13)).

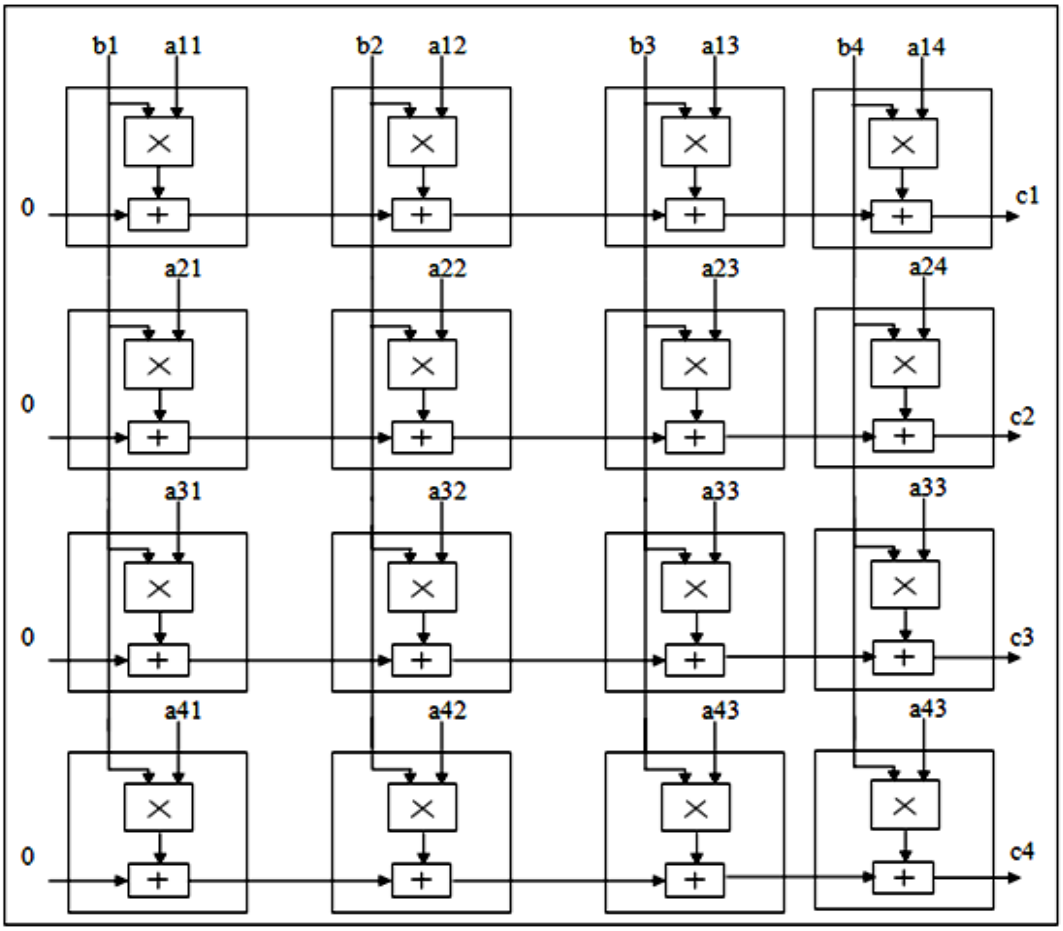


Figure (7): Array processing elements for matrix multiplication

As shown in the simulation waveforms the new x coordinate is computed to be 0005.0000h , the new y coordinate is 002b.4800h which is equivalent to 43.28125 and the new z coordinate is 0018.0000h which is equivalent to 24.

$$\begin{bmatrix} xn \\ yn \\ zn \\ 1 \end{bmatrix} = \begin{bmatrix} 10.60546875 & -10.60546875 & 0 & 5.00 \\ 7.0703125 & 7.0703125 & 0 & 15 \\ 0 & 0 & 10 & 4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 2 \\ 2 \\ 1 \end{bmatrix} \qquad (13)$$

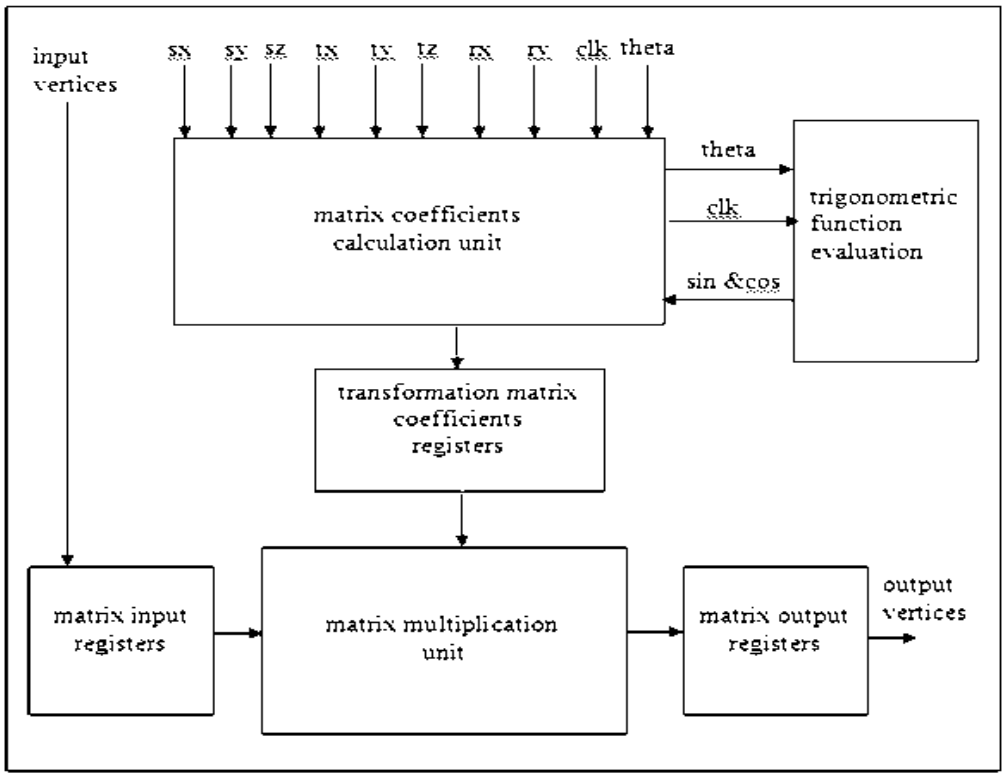


Figure (8): Implemented transformation unit

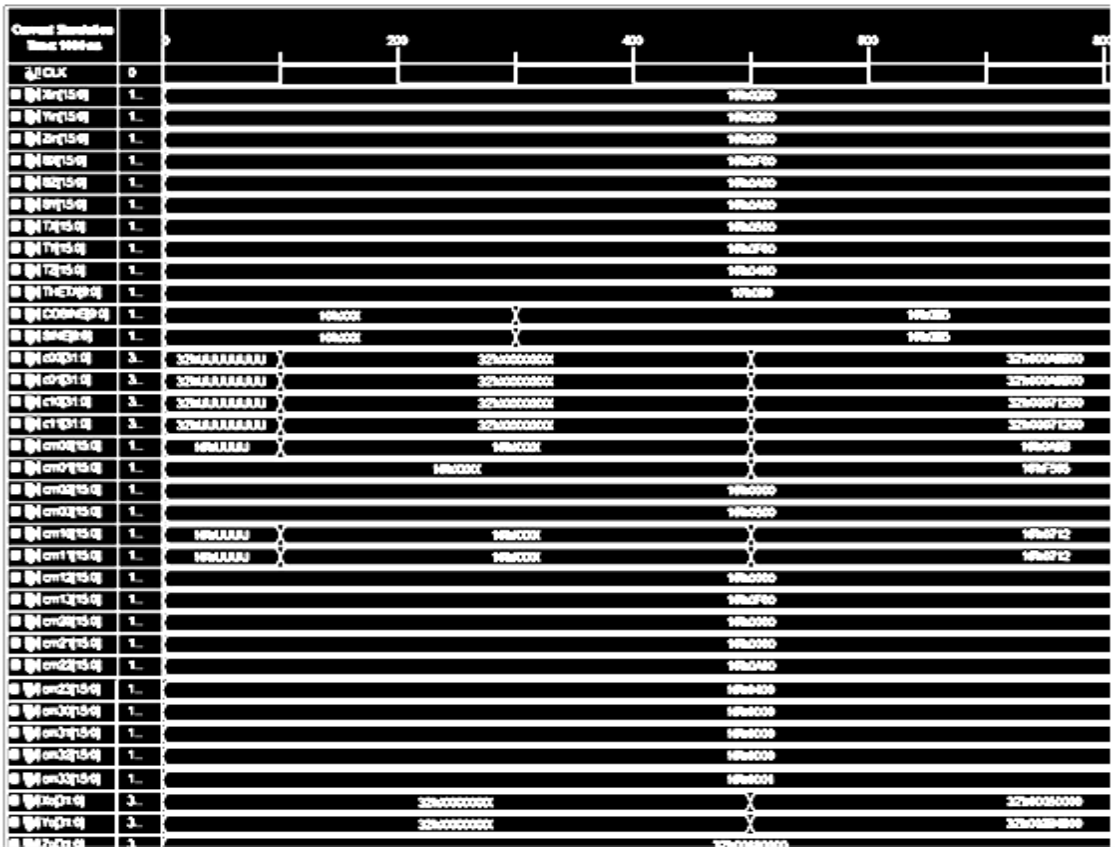


Figure (9): Simulation results of example 1

Figure (10) shows the simulation waveforms for a second example (example2) executed by the implemented unit. In this example the concatenation matrix is executed to carry out rotation about a rotation center (rx, ry) referring to equation (13). As shown in figure (10) the inputs are set to be xin=05.00h, yin=05.00h, zin=05.00h , rx=0a.00h, ry=0a.00h, (sx=01.00h , sy=01.00h, tx=00.00h, ty=00.00h (the last four parameters are set to be not effective in equation (13)), and for the lookup table input theta=0aah (170d=60 degree). The sine and cosine values are computed first using theta input to the lookup table to determine the sine value (0.ddh=00.11011101) which is equivalent to 0.86328125 and cosine value (0.81=00.10000001) which is equivalent to 0.50390625. After that the matrix coefficients are calculated, (cosθ) is the first element and (-sinθ) is the second element which is calculated by converting 00.ddh to 2's complement (ff.23). Next a or (-rx cosθ + ry sinθ + rx) coefficient is calculated to be 000d.9800h which is equivalent to decimal value (13.593751) then b (-rx sinθ - ry cosθ + ry) coefficient is calculated to be fffc.5400h (equivalent to -3.671875). The a and b coefficients are truncated and loaded in Cm02 and Cm12 respectively. Then the new coordinates are computed through multiplying the input matrix of vertices by the transformation matrix as presented by equation (14). As shown by the simulation waveforms, the new x coordinate is 000b.cc00h which is equivalent to 11.796875, the new y coordinate is 0003.2a00h which is equivalent to 3.1640625 and the new z coordinate is 0005.0000h which is equivalent to 5.

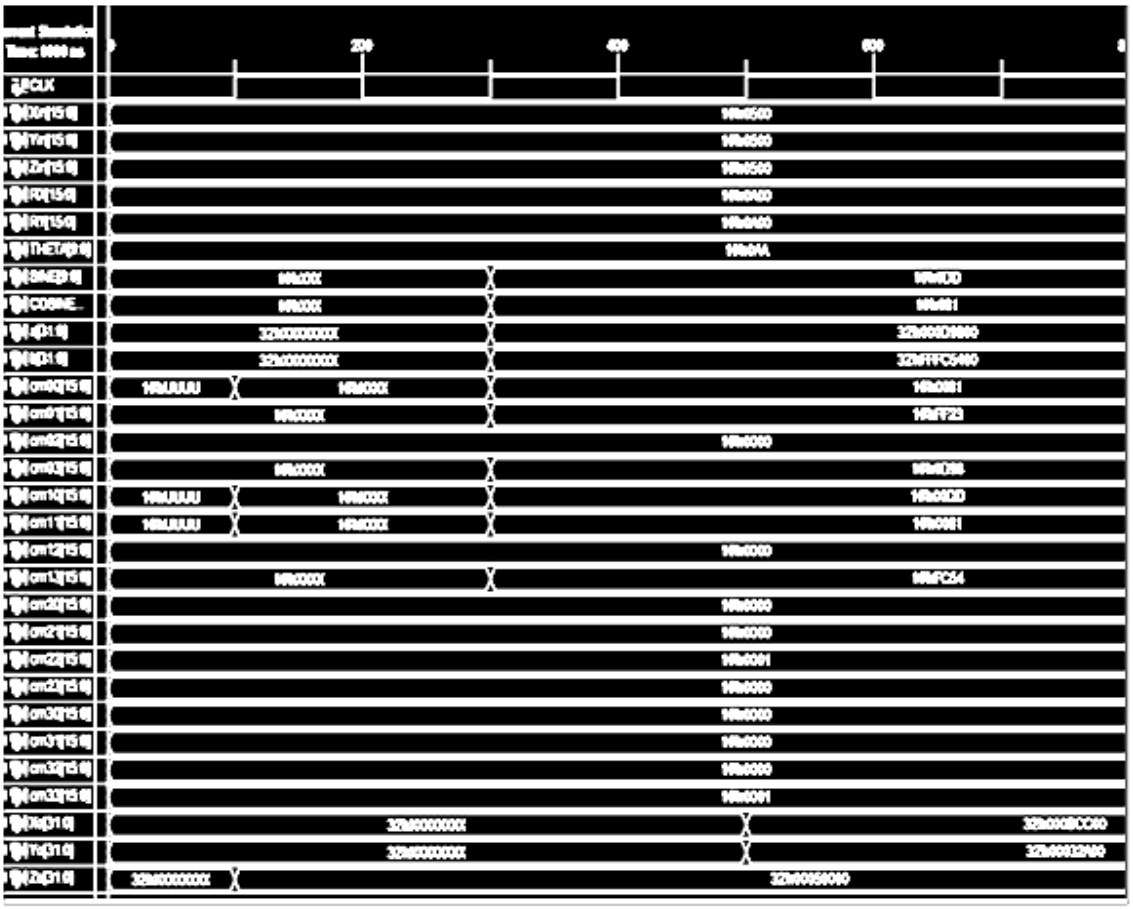


Figure (10) Simulation results of example 2

$$\begin{bmatrix} xn \\ yn \\ zn \\ 1 \end{bmatrix} = \begin{bmatrix} 0.50390625 & -0.86328125 & 0 & 13.593751 \\ 0.86328125 & 0.50390625 & 0 & -3.671875 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 5 \\ 5 \\ 5 \\ 1 \end{bmatrix} \quad \textbf{(14)}$$

Table(1) Resources utilization using lookup table

Type Resources (or Frequency)	Utilized Resources	Total Resources	Ratio
Number of Slices	113	4656	2%
Number of Slices Flip flops	116	9312	1%
Number of 4 input LUTs	207	9312	2%
Number of Bounded IOBs	223	232	96%
Number of Block RAMS	1	20	5%
Number of MULT18X18s	12	20	60%
Number of GCLKs	1	24	4%
Maximum Operating Frequency	101.360MHZ		

Table (1) shows the utilization resources of Spartan3E kit used to implement the unit and the maximum frequency using lookup table. Table (2) shows the utilization resources of spartan3E kit used to implement the unit and the maximum frequency using CORDIC algorithm. With lookup table the synthesized unit utilizes one block RAM and CORDIC implementation does not utilize any block RAM but only silicon slices to implement the shift and add/sub operation. Of course the utilized area for the lookup table is greater than that for CORDIC but the maximum operating frequency is more which justifies adopting lookup table method for real time applications.

Table(2) Resources utilization using CORDIC algorithm

Type Resources (or Frequency)	Utilized Resources	Total Resources	Ratio
Number of Slices	232	4656	5%
Number of Slices Flip flops	279	9312	3%
Number of 4 input LUTs	465	9312	5%
Number of Bounded IOBs	223	232	96%
Number of Block RAMS	0	20	0 %
Number of MULT18X18s	10	20	50%
Number of GCLKs	1	24	4%
Maximum Operating Frequency	70.335 MHZ		

7 Conclusions and performance evaluation

One of the major concerns of real time graphics is the speed of execution. The execution time of a graphic system is function of the complexity of a polygonal graphical object which can be measured by the number of vertices used to represent it in data base and

the time required to transform and process them. In this paper a general method is presented to reduce five matrix operations (one for translation, one for scaling, and three for rotation about an arbitrary parallel axis of rotation) to a single operation each of them is a (4 x 4) matrix multiplication. So the total transformation processing time can be reduced this way to only 20% of its value which is significant for real time systems. This is true for any other sequence of the three mentioned transformations if the concatenation matrix is derived for it following the same procedure presented in this paper.

The performance of the transformation unit is affected by three stages, the first is the sine & cosine evaluation stage. This stage depends on the method used to compute the sine and cosine values (Lookup Table or CORDIC Algorithm). The second stage is the transformation matrix coefficients calculation with which the time of execution is constant (one clock) in the present design. The final stage is the matrix multiplication stage required to determine the output vertices. As confirmed by figure (7), the designed matrix multiplication unit time of execution is constant too and is equal to one clock.

In order to conduct further examination of the performance of the designed system it is used to execute object transformation with 1000 vertices entered to the implemented unit in the two modes, lookup table and CORDIC. With the first technique the time elapsed is 9.89542 μ seconds which means that the designed unit is capable to transform around (101 M) vertices per second. With the second technique the time consumed is 14.40250 μ seconds so the designed unit is capable to transform around (69M) vertices per second. An important issue is the justification of performance in real time. If the vertical frequency of monitoring or the speed of display is assumed 100 frames per second then the available time for the transformation of all vertices is 10 msec. If a graphical object is assumed to have high level of complexity with 100000 vertices then the designed system should transform vertices with minimum speed of 10 M vertex per second. This justifies the capability of the designed system to operate in real time.

However, the cost of gaining the mentioned speed is an absolute overall error which is computed to be between (0) and (0.00281) for the two examples when compared to calculating the new transformed vertices using a pocket calculator. The maximum theoretical absolute error due to quantization of theta occurs for the sine at small values of theta near zero (its derivative cosine is maximum) which is equal to the LSB value (0.006135 for 10 bit). Naturally, the error can further be reduced by increasing the resolution of numbers using more bits for both theta and its sine (or cosine) since the relation between them is linear and the sine of theta equals to theta (in radian) when theta approaches zero. However, reasonable small errors are not significant to most computer graphics applications since round of errors are inevitable and will always be unavoidable when generating pixels at absolute address values from rounded x and rounded y transformed values whether the Digital Differential Analyzer (DDA) or Bresenham algorithm is used for scan conversion.

References

[1] Donald Hearn and M. Pauline, “Computer Graphic C Version” , Third edition, (1997), Prentice Hall International , Inc.

[2] Edward Angel , “ Interactive Computer Graphic , A Top- Down Approach Using OpenGL “,McGraw-Hill Inc, Third Edition 2003.

[3] Bruce A. Wallace , " Merging and Transformation of Raster Images for Cartoon Animation " , Computer Graphics Vol.15, No. 3, August 1981.
[4] Hai Nang Lin, Henk J. Sips , "On-Line CORDIC Algorithms", IEEE TRANSACTIONS ON COMPUTERS, Vol. 39, No. 8, August 1990.
[5] Dwight Hill , Nam-Sung Woo ,"The Benefits of Flexibility in Lookup Table-Based FPGA's", IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS. Vo.12., No 2.,February 1993.
[6] John N. Lygouras, "Memory Reduction in Look-Up Tables for Fast Symmetric Function Generators", IEEE TRANSACTIONS ON INSTRUMENTATION AND MEASUREMENT, Vol. 48, No. 6, December 1999.
[7] Kharrat M.W., Loulou M., Masmoudi N., Kamoun L. , "A New Method to Implement CORDIC Algorithm" , IEEE International Conference on Electronics, Circuits and Systems , Vol. 2, Issue 1 , 2000, Pages:715 – 718.

[8] Bernard Tiddeman, David Perrett, "Moving Facial Image Transformations Based on Static 2D Prototypes", in 9 Int. Conf. in Central Europe on Computer Graphics, Visualization and Computer Vision 2001 (WSCG `2001).
[9] Ali M. A. Abbas, " Transformation of Rendering Algorithms for Hardware Implementation " , Ph.D. Thesis , Department of Control Engineering and Information Technology Faculty of Electrical Engineering and Informatics Budapest University of Technology and Economics Budapest, 2003.
[10] Bensaali, F., Amira, A., Uzun I.S., Ahmedsaid A. ,"An FPGA implementation of 3D affine transformations" , Proceedings of IEEE International Conference on Electronics, Circuits and Systems Vol. 2, Issue 14, Dec. 2005, Pages: 715 – 718.
[11] Faycal Bensaali , Abbes Amira, Reza Sotudeh, "Floating-Point Matrix Product on FPGA " IEEE/ACS International Conference on Computer Systems and Applications, 2007, Pages: 466-473.
[12] Zoran Popovic , AndrewWitkin ,"Physically Based Motion Transformation", Computer Science Department Carnegie Mellon University, SIGGRAPH 99, Los Angeles, August 8–13, 1999.
[13] Robert C. Zeleznik , Kenneth P. Herndon , " SKETCH: an interface for sketching 3D scenes " , International Conference on Computer Graphics and Interactive Techniques ACM SIGGRAPH 2006.
[14] Behrooz Parhami, "Computer Arithmetic: Algorithm and Hardware Designs" , New York Oxford University Press 2000.
[15] Amir H. Farrahi, Majid Sarrafzadeh, "Complexity of the Lookup-Table Minimization Problem for FPGA Technology Mapping", IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, Vol. 13, No. 1 I ,November 1994.
[16] "Spartan-3E, FPGA Family : Functional Description ", DS312-2(V3.5), March 16, 2007, © 2006 Xilinx Inc.
[17] Ghariani, M.; Masmoudi, N.; Kharrat, M.W.; Kamoun, L. "Design and chip implementation of modified CORDIC algorithm for Sine and Cosine functions application: PARK transformation", Proceedings of the Tenth International Conference on Microelectronics , Vol. 10, Issue 11 , 1998, Pages:241 – 244.

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