

# High-speed Parameterisable Hough Transform Using Reconfigurable Hardware

Dixon D. S. Deng, Hossam ElGindy

School of Computer Science and Engineering  
University of New South Wales  
Anzac Parade, Kensington 2033 NSW

{dixond, hossam}@cse.unsw.edu.au

## Abstract

Recent developments in reconfigurable hardware technologies have offered high-density high-speed devices with the ability for custom computing whilst maintaining the flexibility of a software solution. These features are well suited to image processing algorithms that are computationally intensive and repetitive in nature. Very deep pipelining and parallelism, features often required for real time image analysis can be achieved easily using hardware design. The Hough Transform is a powerful and robust global image processing tool for feature recognition and detection. The CORDIC algorithm uses simple shift and addition operations to implement complex trigonometric functions. This paper combines the application of the novel CORDIC algorithm to the Hough Transform and reconfigurable technology to propose a parameterisable Hough Transform with real-time processing throughput.

*Keywords:* Hough Transform, CORDIC, reconfigurable hardware.

## 1 Introduction

Proposed by Volder in 1959, the COordinate Rotational Digital Computer (CORDIC) algorithm can be used to calculate elementary trigonometric functions such as sine, cosine, tangent, and arctangent as well as ln and exp. It can also be used in complex value calculations for matrix triangularization, etc. Based on an iterative algorithm that rotates a vector in two-dimensional linear, circular or hyperbolic coordinate systems, the computational attractiveness of the CORDIC is its use of simple shift and addition to implement complex calculations.

The Hough Transform has traditionally been implemented using software and complex processor architecture. These are either slow or complicated due to the transform's intensive calculations of trigonometric, multiplication and addition operations. Section 2 introduces the CORDIC algorithm, followed by the Hough Transform in section 3. Section 4 explains the adaptation of the CORDIC algorithm to the Hough Transform followed by the description of a simple, high performance parameterisable Hough Transform architecture. The proposed architecture has been

implemented on a fast-prototyping hardware platform and its performance evaluated in section 6.

## 2 Radix-2 CORDIC

Consider the situation where a vector  $(r, \varphi)$  is to be rotated through an angle  $\theta$  in the positive direction where angles  $\varphi$  and  $\theta$  are measured with respect to the X-axis. The problem is depicted below in Figure 1:

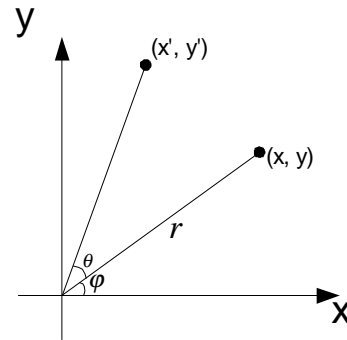


Figure 1 Rotation of vector  $(x,y)$  by  $\theta$

Suppose that the original vector is represented in rectangular components  $(x, y)$  in the Cartesian space, the resultant vector after rotation can be represented by  $(x', y')$ . The relationship between the original and rotated vectors is shown below:

$$x' = x \cos \theta - y \sin \theta = (x - y \tan \theta) \cos \theta \quad (1)$$

$$y' = y \cos \theta + x \sin \theta = (y + x \tan \theta) \cos \theta \quad (2)$$

The CORDIC algorithm performs such a vector rotation by means of a sequence of micro-rotations.

### 2.1 Micro-rotations

Vector rotation is carried out by means of a finite sequence of iterations where the  $i^{\text{th}}$  step rotates the corresponding vector in either the positive or negative direction over a pre-fixed elementary angle  $\alpha_i$ , which is defined by:

$$\alpha_i = \arctan(2^{-i}) \quad (3)$$

Thus, the entire  $i^{\text{th}}$  micro-rotation can be described as follows:

$$x_{i+1} = x_i - \delta_i y_i \tan \alpha_i = x_i - \delta_i y_i \times 2^{-i} \quad (4)$$

$$y_{i+1} = y_i + \delta_i x_i \tan \alpha_i = y_i + \delta_i x_i \times 2^{-i} \quad (5)$$

$$\theta_{i+1} = \theta_i + \delta_i \tan 2^{-i} = \theta_i + \delta_i \alpha_i \quad (6)$$

$$i = 0, 1, 2, 3, \dots$$

where  $\delta_i \in \{-1, 1\}$  and controls the direction of vector rotation for step  $i$ . This procedure continues until  $\theta_i$  is suitably close to the desired rotation angle. Apart from a factor of  $\cos\theta$ , equations (4) and (5) are identical to (1) and (2) respectively.

Neglecting the  $\cos\theta$  term result in a rotated vector that is  $K$  times larger than the desired vector, where  $K$  is the value of the scale factor, and is defined by the equation below for  $n$  micro-rotations:

$$K = \prod_{i=0}^{n-1} \frac{1}{\cos \alpha_i} = \prod_{i=0}^{n-1} \sqrt{1 + 2^{-2i}} \quad (7)$$

Because each CORDIC calculation performs an identical number of micro-rotations over the same set of angles, the value of  $K$  remains constant for a particular CORDIC solution.

## 2.2 Scale factor compensation

A simple method for scale factor compensation involves the continued addition of the value  $1/K$ . Used by Zhou (1995), the result of addition is then passed on as input to the CORDIC where micro-rotations are performed. This technique effectively pre-scales the input value to CORDIC calculations, and is simple to implement due to the static nature of operations and values used.

The disadvantage of using serial scale factor compensation is that pixels in the input binary image must be processed sequentially. Most pre-processed binary images only contain feature points that are not likely to represent a large proportion of the image space. Therefore, many calculations are wasted.

Parallel scale factor compensation was proposed by Villalba (1995) as an alternative to serial scale factor compensation. Consider a new angle  $\beta$ , which is defined as:

$$\beta = \arccos(1/K) \quad (8)$$

where  $K$  is the scale factor.

Instead of rotating the vector by  $\theta$  and obtaining a single resultant vector, the parallel scale factor compensation technique performs two rotations through angles  $(\theta+\beta)$  and  $(\theta-\beta)$ . The two resulting vectors are shown as  $(x^+, y^+)$  and  $(x^-, y^-)$  below:

$$\begin{bmatrix} x^+ \\ y^+ \end{bmatrix} = K \begin{bmatrix} x \cos(\theta + \beta) - y \sin(\theta + \beta) \\ y \cos(\theta + \beta) + x \sin(\theta + \beta) \end{bmatrix} \quad (9)$$

$$\begin{bmatrix} x^- \\ y^- \end{bmatrix} = K \begin{bmatrix} x \cos(\theta - \beta) - y \sin(\theta - \beta) \\ y \cos(\theta - \beta) + x \sin(\theta - \beta) \end{bmatrix} \quad (10)$$

By performing a semi-sum of  $x^+$  with  $x^-$  and  $y^+$  with  $y^-$  and performing simple algebraic manipulation, we obtain:

$$\frac{x^+ + x^-}{2} = K(x \cos\theta \cos\beta - y \sin\theta \cos\beta) \quad (11)$$

$$\frac{y^+ + y^-}{2} = K(y \cos\theta \cos\beta + x \sin\theta \cos\beta) \quad (12)$$

Now by substituting the value of  $\cos\beta$  from equation (8) into the above equations, the right hand side of equations (11) and (12) corresponds to equations (1) and (2) respectively. That is, by performing two rotations in parallel and a semi-sum of the results, a correctly scaled version of the CORDIC algorithm is achieved. The benefit of this scheme over the serial scale factor compensation is that only non-zero (feature points) in the image need to be processed.

## 2.3 Micro-rotation sequence

Any elementary angle can be described by a sequence of micro-rotations using equation (13). This equation allows the pre-calculation of the rotation sequence for a given angle. The ability to calculate the micro-rotation sequence prior to vector rotation greatly simplifies implementation.

Let the rotation sequence be defined by the vector  $\mathbf{M} = (m_0 m_1 \dots m_{n-1})$ . The angle  $\theta$  represented by  $\mathbf{M}$  is then

$$\theta = \sum_{i=0}^{n-1} (-1)^{m_i} \alpha_i \quad (13)$$

Consequently, a change in the value of  $m_{n-1}$  in state vector  $\mathbf{M}$  produces a change in the angle  $\theta$  of  $2\alpha_{n-1} = 2^{2-n}$ . By varying  $\mathbf{M}$  within the range  $(00\dots0)$  to  $(11\dots1)$ ,  $2^n$  angles within the range  $[-100^\circ, 100^\circ]$  can be covered by the CORDIC algorithm.

It can be shown that some states represented by vector  $\mathbf{M}$  are redundant. That is, there are overlapping angle values among the  $2^n$  possible states. To obtain a uniform distribution of angles in the range  $[-100^\circ, 100^\circ]$ , redundant states must be removed. The overlapping angle states appear when the four most significant values in the micro-rotation vector  $\mathbf{M}$  change values from 0 to 1 or from 1 to 0. To remove angle redundancy, substitute angle states are pre-determined and replace an overlapping state once it is reached. Table 1 below gives an example of where overlapping states are located for positive angle states in the range  $[0^\circ, 90^\circ]$  with  $n=12$  micro-rotations.

$m_0 m_1 m_2 m_3$	$m_4 m_5 m_6 \dots m_{11}$
0000	10110000
0001	00010101
0010	00011000
0011	00010011
0100	01001100
0101	00010101
0110	00011000
0111	00010011

Table 1 non-overlapping micro-rotation states

### 3 Hough Transform

Proposed by Hough in 1962, the Hough Transform determines global relationships between pixels in an image. It is commonly used in image processing to establish whether pixels lie on a curve of a specific shape. The advantage of using the Hough Transform for global image feature extraction is its robustness even in cases of noise or partial occlusion.

A point with coordinates  $(x_i, y_i)$  can have an infinite number of straight lines passing through it in unquantised space. These lines can be described in polar coordinates in the  $(\rho, \theta)$  space by the equation:

$$\rho = x_i \sin \theta + y_i \cos \theta \quad (14)$$

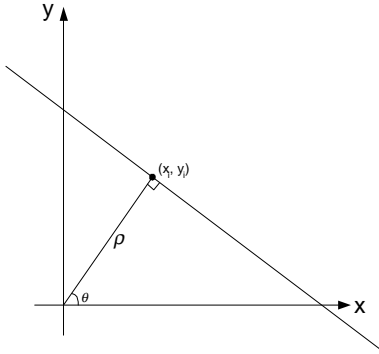


Figure 2  $(x, y)$  and  $(\rho, \theta)$  relationship

Figure 2 above shows the relationship between  $x_i, y_i, \theta$  and  $\rho$ . The range of angle  $\theta$  is  $[0, \pi]$  measured with respect to the x-axis and  $(x_i, y_i)$  are coordinates relative to the centre of the image. Given these conditions, for an N-by-N image with origin at the centre of the image, the range of  $\rho$  is  $[-N/\sqrt{2}, N/\sqrt{2}]$ .

To achieve computational efficiency, the parameter space of the Hough Transform is quantised or subdivided into  $N_\theta$ -by- $N_\rho$  accumulator cells.  $N_\theta$  is the number of  $\theta$  quantisation levels, while  $N_\rho$  is the number of  $\rho$  quantisation levels. The result is  $N^2$  computational complexity for an N-by-N image. Initially, the accumulator cells are set to a value of zero. The transform is then applied to the image with appropriate  $(\rho_i, \theta_i)$  accumulator cells incremented for each calculation. Therefore, for  $N^2$  image points in the x-y plane,  $N^2 N_\theta$  calculations are needed to perform a complete Hough Transform.

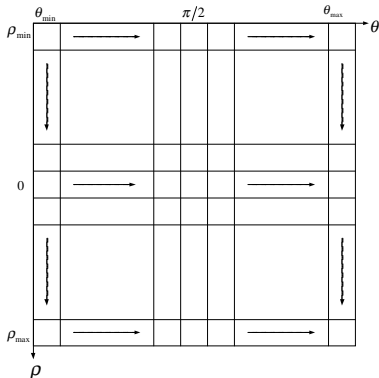


Figure 3 Quantisation of parameter space

High  $N_\rho$  and  $N_\theta$  values are desirable to achieve high precision Hough Transform results.

Under Hough Transform, a point in the x-y plane is translated into a sinusoidal curve in the  $\rho$ - $\theta$  plane (also known as the parameter space). Furthermore, collinear points in the x-y plane will have their associated sinusoidal curves in the parameter space intersecting at  $(\rho_j, \theta_j)$ , where

$$\rho_j = x \sin \theta_j + y \cos \theta_j \quad (15)$$

describes the line.

The Hough Transform thus involves calculating the  $\rho$ -value for all  $\theta$  in the range  $[0, \pi]$  for each feature pixel in the binary image. Mathematically, it is performing regression estimation over the parameters of a binary image. Post-processing using a threshold value of  $\rho_{\text{threshold}}$  determines whether a  $(\rho, \theta)$  pair describes a line in the original image.

### 4 Hough Transform using CORDIC

The similarities between equations (14) and (1) and (2) mean the application of the CORDIC to the Hough Transform is relatively natural. Consider the rotation of a vector  $(x, y)$  by  $\theta \in [0, \pi/2]$ , then

$$\rho_{0-\pi/2} = x \cos \theta + y \sin \theta \quad (16)$$

$$\rho_{\pi/2-\pi} = x \cos(\theta + \pi/2) + y \sin(\theta + \pi/2) = y \cos \theta - x \sin \theta \quad (17)$$

By interchanging the x and y inputs to the CORDIC computation, equation (2) matches the  $\rho_{0-\pi/2}$  value in

equation (16) above. The  $\rho_{\pi/2-\pi}$  value in equation (17) also corresponds to equation (1). Therefore, both outputs of the CORDIC algorithm are utilised in Hough Transform calculations, effectively halving the computational complexity. As a result, the range of  $\theta$  is reduced to  $[0, \pi/2]$ .

The micro-rotation sequence used by the CORDIC algorithm can be pre-calculated and stored in lookup tables or calculated during the CORDIC computation using work by Zhou (1995). Suppose the number of quantisation levels of  $\theta$  is  $N_\theta$ , then

$$\Delta \theta = \frac{\pi}{N_\theta} \quad (18)$$

So for an n-iteration CORDIC, the ratio of  $\Delta \theta$  to micro-rotation stepsize  $2^{-n}$  is

$$a = \frac{\Delta \theta}{2^{-n}} = \frac{\pi}{N_\theta 2^{2-n}} \quad (19)$$

By interpreting the rotation vector  $\mathbf{M}$  as a number, an increment by  $a$  will produce a corresponding change of  $\Delta \theta$  in the angle. In order to minimise error, it is usually desirable to obtain an  $a$  value that is close to an integer.

## 5 High-speed parameterisable Hough Transform architecture

A Hough Transform using CORDIC algorithm can be implemented in hardware devices for high-speed processing. Using reconfigurable hardware, parameters of the Hough Transform do not have to be fixed during development to achieve high computation rates (section 5.4).

This section describes components used in the composition of a parameterisable Hough Transform. Fixed-point number representation is preferred for its higher precision and arithmetic simplicity when compared to floating-point numbers (Zimmerman, 1999).

### 5.1 CORDIC for Hough Transform architecture

The CORDIC algorithm is highly pipelinable due to its regular and repetitive nature. From equations (4) and (5), the  $i^{\text{th}}$  iteration right-shifts the results of the  $(i-1)^{\text{th}}$  stage by  $i$  bits and performs exactly 1 addition and 1 subtraction. Hence by capturing the operations of one iteration of the CORDIC algorithm into one stage of a pipeline, then an  $n$ -iteration CORDIC can be realised by an  $n$ -stage pipeline. This pipelined CORDIC architecture with its  $x$  and  $y$  inputs swapped for Hough Transform is shown in the diagram below:

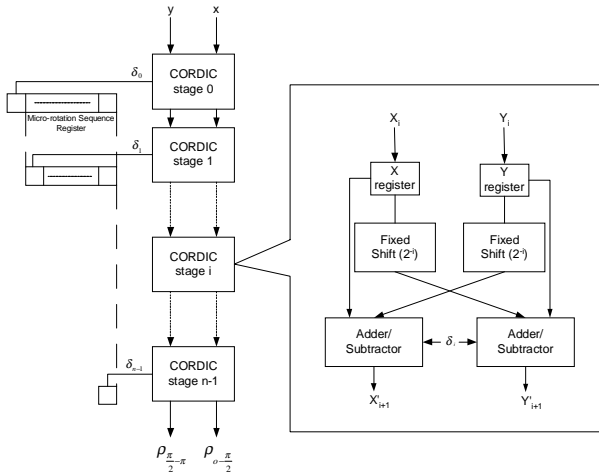


Figure 4 Pipelined CORDIC architecture for Hough Transform

A simple register pipeline propagates micro-rotation sequences alongside the CORDIC pipeline. Fixed-shifters are employed in each stage that shifts its input by  $i$  bits for stage  $i$ .

For  $n = 12$  iterations, the latency of the pipelined CORDIC is 12 clock cycles. However, the data throughput is one set of complete calculations per clock cycle.

### 5.2 Micro-rotation generator architecture

The micro-rotation sequence used in CORDIC calculations can be obtained by pre-calculating the sequences for all angles and storing them in a Look-Up-

Table (LUT) or using equation (13) to calculate sequences at the same rate as the CORDIC.

The latter method allows the Hough Transform angle resolution parameter to be changed without recalculating the entire set of sequences because only the step size  $a$  needs to be altered. Hence, the only values that need to be pre-calculated are the starting sequence and the angle-overlap boundaries. Figure 5 below shows the architecture of a micro-rotation generator.

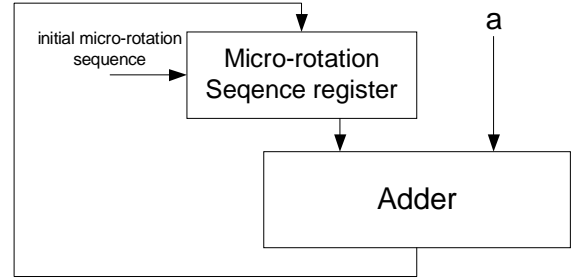


Figure 5 micro-rotation generator design

A small LUT stores the angle-overlap boundaries and their corresponding replacement sequences so the CORDIC does not carry out any redundant calculations. This LUT is omitted from Figure 5.

### 5.3 Scale factor compensation

The serial scale factor compensation scheme can be implemented using the design shown in Figure 6 below. The  $x$  and  $y$  values have an initial value of  $-N/2K$  where  $N$  is the dimension of an  $N$ -by- $N$  square image and  $K$  is the scale factor.  $1/K$  is then iteratively added to the  $x$  and  $y$  register values when the next pixel is to be processed. The  $-N/2K$  value converts screen coordinates to image coordinates while the  $1/K$  factor ensures the input to the CORDIC is pre-scaled by the scale factor.

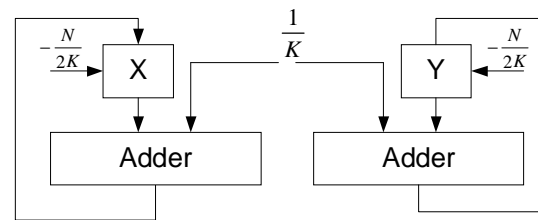


Figure 6 Serial scale factor compensation design

A CORDIC with serial scale factor compensation is simple to implement and processes an entire binary image instead of only those pixels with a value of 1.

When only feature points of a binary image need to be Hough Transformed, the parallel scale factor compensation scheme can be adopted. Consider the value of the scale factor in the chosen 12-iteration CORDIC implementation. Using equation (7), for 12 iterations,  $K \approx 1.64676$ . Therefore, using equation (8), the  $\beta$  angle used in parallel scale factor compensation in equations (10) and (11) is  $52.6088^\circ$ . Since the range of  $\theta$  values

used in performing the Hough Transform is  $[0^\circ, 90^\circ]$ , then with parallel scale factor compensation, the CORDIC must cover an angle range of  $[-52.6088^\circ, 142.6088^\circ]$ . However, the CORDIC algorithm can only perform calculations for angles within the range  $[-100^\circ, 100^\circ]$ .

Calculations using equation (10) is unaffected by the difference in angle range because it performs calculations within the range  $[-52.6088, 37.3912]$ . However, equation (9) involves calculations with angle values of  $\theta + \beta > 100^\circ$ . By letting  $\alpha = 90^\circ - (\theta + \beta)$ , then consider

$$x' = x \cos \alpha - y \sin \alpha = x \sin(\theta + \beta) - y \cos(\theta + \beta) \quad (20)$$

$$y' = y \cos \alpha + x \sin \alpha = y \sin(\theta + \beta) + x \cos(\theta + \beta)$$

By interchanging the above  $x$  and  $y$  values, then we have the following equivalence between  $x'$  and  $y'$  of equations (20) and (9):

$$\begin{aligned} x' &= -x^+ & (21) \\ y' &= y^+ \end{aligned}$$

Therefore,  $x^+$  and  $y^+$  is calculated for angles  $-52.6088^\circ$  to  $37.3912^\circ$  while  $x^-$  and  $y^-$  is calculated for angle  $37.3912^\circ$  to  $-52.6088^\circ$ .

The divide-by-2 operation of the semi-sum operation in equations (11) and (12) can be carried out at the input to the CORDIC to avoid arithmetic overflow for fixed-point arithmetic. The resultant architecture is shown in Figure 7 below:

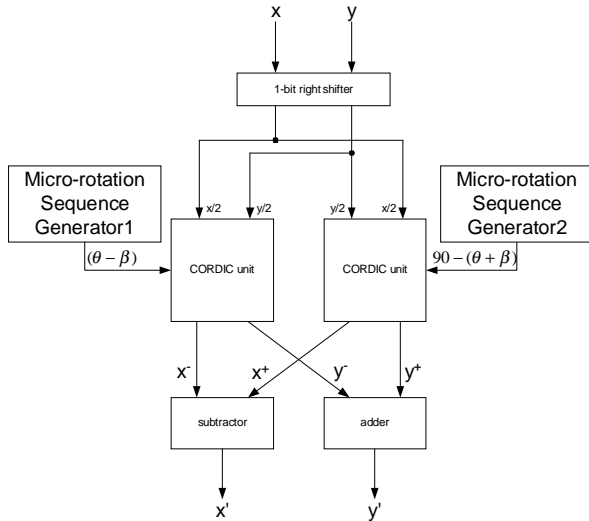


Figure 7 Parallel scale factor compensation design

## 5.4 Parameterisable Hough Transform

The proposed component designs form the basis of a Hough Transform that is parameterisable by image size,  $\rho$  resolution and angle resolution. Larger  $N_\rho$  and  $N_\theta$  values lead to finer-grain parameter space that produce more accurate Hough Transform results. The absolute limit of both angle and  $\rho$  resolution depends on the underlying CORDIC implementation.

To alter angle resolution, only the  $\alpha$  constant in the micro-rotation sequence generator need to be changed according to equation (19). The rest of the architecture remains untouched.

Angle precision is limited by the length of the micro-rotation sequence. If extra rotational precision is required for the pipelined CORDIC architecture, one additional pipelined stage is required for serial scale factor compensation, two if parallel scale factor compensation is employed. Hence, there is a linear increase in hardware resources required to increase rotational precision. However, calculation throughput is unaffected by this change.

The procedure to vary  $\rho$  resolution in the Hough Transform is similarly non-intrusive. The resolution limit is dependant on the data representation used in the underlying CORDIC. Higher arithmetic precision can only be achieved by using larger bit widths to represent data. This generally leads to a linear relationship between size of data width and size of hardware resources as well as path delays. Since CORDIC arithmetic precision is usually much higher than  $\rho$  precision in order to minimise the cumulative error introduced by micro-rotations, the method of changing  $\rho$  resolution is to simply alter the number of most significant bits from the CORDIC result to the Hough Transform output.

To adapt the Hough Transform to new image sizes, one can either change the proportion of integer/fractional bits used in data representation or alter the computation data width. Both of which can be easily achieved using reconfigurable devices.

## 6 Performance and error analysis

A Hough Transform using 16-bit fixed-point arithmetic, 12-iteration CORDIC was implemented using a Xilinx XC4010XL-PC84 FPGA for fast prototyping. Design entry used the VHDL hardware description language.

An FPGA is a reconfigurable logic device consisting of a two-dimensional array of RAM-based programmable cells known as Configurable Logic Blocks (CLBs) (Xilinx, 2001). Each CLB consists of a number of function generators implemented as memory look up tables (LUTs), storage elements to latch generator results, as well as inputs and outputs. In addition, dedicated carry logic circuitry is available in the CLB's function generators for the fast generation of carry and borrow arithmetic logic to increase the efficiency of adders, subtractors, accumulators, comparators and counters. Programmable routing resources (channels) provide interconnections between the inputs and outputs of configurable elements within an FPGA to appropriate networks. The functionality of each circuit block can be customized via configuration bit-streams.

The Xilinx XS4010XL-PC84 FPGA is a medium capacity device capable of running at moderate speeds. It has 400 CLBs arranged into a  $20 \times 20$  array, which is equivalent to approximately 10000 gates.

## 6.1 Hough Transform performance

The Hough Transform using pipelined CORDIC with serial scale factor compensation utilises 83% or 333 out of 400 CLBs of the XC4010XL FPGA. This implementation can be clocked at more than 40 MHz with a computational complexity of  $\theta(N^2)$  for an N-by-N image. At this frequency, a Hough Transform of a 128-by-128 binary image with 128 discrete angles ( $\Delta\theta = 1.40625^\circ$ ) takes  $\frac{128 \times 128 \times 64}{40 \times 10^6} = 0.0262$  seconds to transform one image. At approximately 38 frames a second, this architecture can perform real-time Hough Transforms.

Hough Transform with parallel scale factor compensation scheme was not implemented because of the limitation on CLB resources. However, the path delay for such a design will be similar to a Hough Transform using serial compensation. Hence, assuming 1/p proportion of the image are feature points, then the processing rate is  $\frac{128 \times 128 \times 64}{40 \times 10^6 \times p} = \frac{0.0262}{p}$  seconds for one image. Since  $p \geq 1$ , this design can Hough Transform a 128-by-128 image at a minimum of 38 frames per second.

A typical software program that performs Hough Transform was written as a basis to compare microprocessor performance and that of the FPGA. A program was constructed to perform 128  $\rho$ -value calculations using sine and cosine functions for  $128 \times 128 = 16384$  pixels. The following C++ code shows the calculation segment of the program.

```

register int i, j;
register double k;
start = clock();
register double result = 0;
for (i = 0; i < 128; i++) {
    for (j = 0; j < 128; j++) {
        for (k = 0; k < PI; k += PI/128) {
            i*sin(k) + j*cos(k);
        }
    }
}
finish = clock();

```

Figure 8 Code segment of software Hough Transform

This program was developed in Microsoft Visual C++ 6.0, built, and run in Debug mode on a PIII 667MHz with 133MHz FSB and 512KB cache Windows 2000 PC. Results show that the software Hough Transform can process 16384 pixels in 0.921 seconds. This is over 35 slower than the throughput of a comparable pipelined CORDIC Hough Transform. The performance advantages made available by an FPGA's ability for custom computation is clear from this comparison.

## 6.2 Precision and error analysis

The Hough Transform error depends on the quantisation of the parameter space as well as errors introduced by the CORDIC. For an N-by-N image with  $N_\theta$  and  $N_\rho$

quantisation levels of the parameter space, then

$$\Delta\theta = \frac{\pi}{N_\theta} \text{ and } \Delta\rho = \frac{\sqrt{2}N}{N_\rho}$$

$$\frac{\Delta\theta}{2} \text{ and } \frac{\Delta\rho}{2} \text{ respectively.}$$

Two non-ideal factors adversely affect the CORDIC algorithm precision, namely limited rotational precision and arithmetic precision. The rotational accuracy of the CORDIC depends on the number of micro-rotations it carries out (Zhou, 1995). For an N-by-N image, the choice of n' iterations instead of n results in a maximum absolute error show in equation (22) below.

$$\begin{aligned} |error|_{\max} &= 2 \times \frac{N}{\sqrt{2}} (2^{-n} + 2^{-(n+1)} + 2^{-(n+2)} + \dots) \\ &= N\sqrt{2} \times 2^{-n} (1 + \frac{1}{2} + \frac{1}{4} + \dots) \\ &= N \times 2^{-(n-1/2)} \end{aligned} \quad (22)$$

The arithmetic precision of the CORDIC depends on the number of fractional bits chosen to represent data. When M fractional bits are used, a single CORDIC micro-rotation contributes a maximum absolute error of  $2^{-M}$ . Because this error is cumulative due to the CORDIC's iterative nature, the maximum absolute error for an n-micro-rotation CORDIC is  $2^{-M} n$ .

The total error in performing a Hough Transform using the CORDIC is the sum of rotational and arithmetic errors. Hence, the use of 128-by-128 images with 12-iteration CORDIC and 16-bit signed 2's complement number representation with 8 fractional bits gives an absolute error of less than 0.135. Given that  $\rho$  resolution is usually 1, this error value is acceptable.

To improve CORDIC precision, one can either increase the number of micro-rotations to enhance rotational accuracy or expand the number of bits used in data representation, both schemes involve trade offs in terms of hardware resources needed and the speed at which the designs can process data.

One stage in the 16-bit pipelined CORDIC requires 18 CLBs when implemented in the XC4010XL FPGA. Hence, at least 18 more CLBs are required, 36 if parallel scale factor compensation is employed if extra rotational precision is required for a pipelined CORDIC architecture. There is a linear increase in hardware resources required to increase rotational precision. However, calculation throughput is unaffected by this change.

Higher arithmetic precision can only be achieved by using larger bit widths to represent data. This leads to higher hardware costs and longer path delays. In order to measure the effect of higher precision in a reconfigurable environment, a single arithmetic unit used in the pipelined CORDIC is synthesised and implemented for various data widths. An arithmetic unit in the CORDIC performs two fixed-shift operations, 1 addition and 1 subtraction. This experiment accurately determines the effect of required precision on size and speed of design.

Figure 9 and Figure 10 below show the results of the investigation.

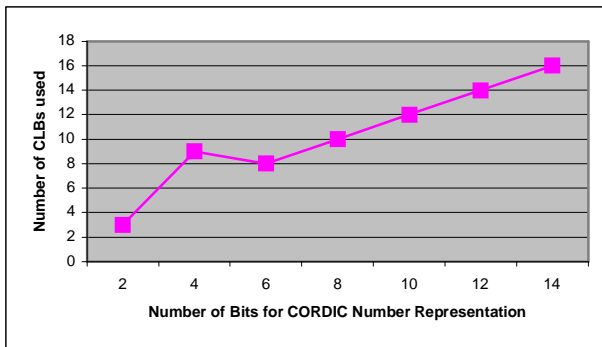


Figure 9 Precision versus CLBs required

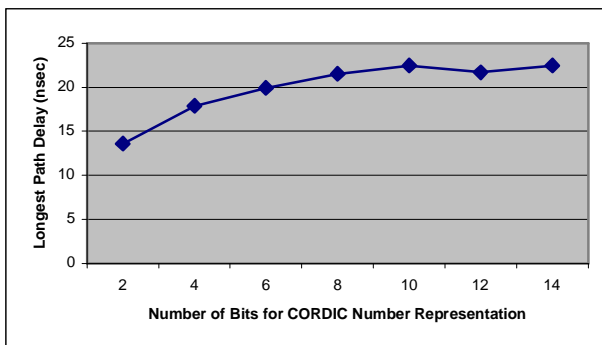


Figure 10 Precision versus Longest Path Delay

The above graphs clearly show that precision is linearly related to hardware resources required. Interestingly, there appears to be a logarithmic relationship between precision and the speed at which designs can be clocked. Generally, for high precision, a linear increase in hardware resources leads to a linear rise in path delays. The presence of dedicated carry logic circuitry in the FPGA and design optimisation by development tools may be the reason for the relationship between precision, area and speed in this case.

## 7 Conclusion

The vector rotation and trigonometric computation provided by the CORDIC algorithm not only applies to Hough Transform, but are also used in other image processing applications such as graphics animation, Discrete Cosine Transform (DCT) and Discrete Fourier Transform (DFT).

Utilising reconfigurable devices' ability for custom computation, work carried out here has shown that a Hough Transform processor using the pipelined CORDIC algorithm is a viable alternative to software and complex processor solutions. Using only shifts and additions, a small custom parameterisable FPGA Hough Transform processor offers ASIC performance with software flexibility.

## 8 References

HOUGH, P.V.C. (1962): Methods and Means for Recognizing Complex Patterns, U.S. Patent 3,069,654

VOLDER J. (1959): The CORDIC Computing Technique, IRE Transactions on Computers, 330-334

VILLALBA J., HIDALGO J.A., ANTELO E., BRUGUERA J.D. AND ZAPATA E.L. (1995): CORDIC Architecture with Parallel Compensation of the Scale Factor', *Proc. International Conference on Application Specific Array Processors (ASAP'95)*, 258-269

XILINX (2001) [online], Available: <http://www.xilinx.com> [2001, October 14]

ZIMMERMANN, R. (1999): *Computer Arithmetic: Principles, architecture, and VLSI Design*. Integrated Systems Laboratory, Swiss Federal Institute of Technology (ETH), Zurich

ZHOU, F., KORNERUP, P. (1995): A High Speed Hough Transform Using CORDIC, *IMADA preprint, Odense University Denmark*, 1995-27