Design of CORDIC-based Digital Protective Relay

A Thesis submitted in partial fulfillment of the requirements for the

degree of

Bachelor of Technology

in

Electronics and Communication Engineering

By

Manisha Swain

Roll No. 109EC0226

Stutee Natak

Roll no. 109EC0226

Under the supervision of

Dr. Kamala Kanta Mahapatra

Professor

Department of Electronics and Communication Engineering,

National Institute of Technology, Rourkela

Session 2012-2013

National Institute of Technology, Rourkela

C E R T I F I C A T E

This is to certify that the Thesis entitled, '**Design of CORDIC-based Digital Protective Relay'** submitted by **Stutee Nayak and Manisha Swain** in partial fulfilment of the requirements for the award of **Bachelor of Technology Degree** in **Electronics and Communication Engineering** at the **National Institute of Technology, Rourkela** is a bona fide work carried out by them under my supervision. To the best of my knowledge and belief, the matter embodied in the Thesis has not been submitted by them to any other University/Institute for the award of any Degree/Diploma.

Date

Prof. Kamala Kanta Mahapatra

Dept. of Electronics and Communication Engineering,

National Institute of Technology, Rourkela

ACKNOWLEDGEMENT

This project in itself is an acknowledgement to the motivation, drive and the technical assistance contributed to it by so many people. It would never have seen the light of day without the timely help and guidance that it received from them.

Firstly, we would like to express our sincere thanks and deepest regards to our guide **Dr. K K Mahapatra, Professor, Department of Electronics and Communication Engineering, NIT Rourkela**, who has always been the driving force behind this work. We thank him for giving us the opportunity to work under him by putting a trust in our credentials and capabilities, and helping us in exploring our potential to the fullest.

We are grateful to **Prof. S. Meher**, Head of the Department of Electronics and Communication Engineering, for permitting us to use the facilities available in the department to carry out the project successfully.

We are thankful to **Mr. Vijay Sharma**, PG student in the Department of Electronics and Communication Engineering, NIT Rourkela, for his generous help and continuous encouragement in various ways towards the completion of this project.

Last but not the least we would like to thank all our friends for their support. We are thankful to our classmates for all the thoughtful and mind stimulating discussions we had, prompting us to think beyond the obvious.

Manisha Swain

Stutee Nayak

ABSTRACT

Protective relays are used to show the current status of a given power system. It is used to determine whether parameters like voltage or current violate any trip or reset specification or condition. In this project, overcurrent time invariant model of protective relay is implemented using CORDIC algorithm.

Here we use CORDIC algorithm to implement the relay flowchart. All of the arithmetic calculations and relaying algorithms can be managed by a CORDIC processor. As CORDIC algorithm is simple to implement, it has a very low FPGA footprint and it negates any need of multipliers as well.

In this project:

1) A basic implementation of sine/cosine calculation of CORDIC using VHDL in Xilinx ISE 10.1 is implemented to check the accuracy
2) Various modules needed to calculate the protective relay status of trip or reset condition were developed
3) A final implementation of overcurrent relay was done and accuracy checked

CONTENTS

LIST OF FIGURES:

LIST OF TABLES:

CHAPTER 1

INTRODUCTION

1.1 MOTIVATION

Protective relays are an integral part to maintain safety in the electrical system. Compared to any static or analogue relay, the digital relay usually lacks computing resources but, provides wider range of settings as well as other digital functions. Here we use CORDIC algorithm to implement the relay flowchart. As CORDIC, acronym for Coordinate Rotation Digital Computer is a simple and hardware-efficient method, all of the arithmetic calculations in meters and relaying algorithms can be managed by a CORDIC processor. Even if the relay used covers multi-channel sources and various relay algorithms, its processing speed must be appropriate for operations in real-time environment. As the given digital model for the time-current characteristic is neither a simple curve adaptation nor LUT (lookup table) based extrapolation, this model easily provides not only various characteristic curves for relay co-ordination, but also highly accurate relaying behavior because of the inherent accuracy of the CORDIC algorithm

1.2 PROBLEM STATEMENT

The primary aim of this project is to develop a CORDIC based protective relay system which follows inverse time characteristics. To implement this in VHDL and verify its functionality and accuracy.

1.3 ORGANIZATION OF THESIS

This thesis has been divided into five chapters. The first one give the basic aim, motivation as well as the work intended. In chapter 2, the various facets and methods in which CORDIC can be implemented is explored. And along with it, basic algorithm to implement functions like multiplication, division, exponentiation etc. Chapter 3 visits the essentials of protective relay and the equations used. It also provides a flowchart which is implements. The fourth chapter gives the simulation results and outputs of the modules created. The conclusion and future work has been proposed in chapter 5.

CHAPTER 2

CORDIC OVERVIEW

2.1BACKGROUND

CORDIC (Coordinate Rotation DIgital Computation) is based on ancient principles of 2-D geometry. It was first implemented in 1959 when an iterative computational formula was given by Jack E. Volder [1] which calculated trigonometric functions like sine and cos values, division and multiplications. The microprocessors used in Digital Signal Processing are not fast and the algorithms designed for these processors do not map properly with hardware [2]. CORDIC uses only shifts and add functions that makes it really fast and hardware mapping becomes easy. For iteration, additional accuracy bit accuracy is produced by CORDIC [2]. Applying vector rotations all the trigonometric functions can be derived or calculated. Vector rotations can be used for converting polar coordinate into rectangular and rectangular coordinates into polar. Iterative method is implemented to get the result in accumulators.

2.2CORDIC ALGORITHM

2 methods of computing are Rotation and Vectoring

2.2.1 ROTATION

In this method, coordinates of a vector and the angle for rotation are known. After rotation through given angle the final coordinates for original vector is calculated [3]. Angle to be rotated is initialized in the angle accumulator. Magnitude of residual angle inside angle accumulator diminishes by rotation after each iterations. Decision after every iteration based on sign of residual angle at each step. Equations for rotation mode are:

$$z_{i+1} = z_i - d_i * \text{atan}(2^{-i})$$

$$x_{i+1} = x_i - y_i * d_i * 2^{-i}$$

$$y_{i+1} = y_i + x_i * d_i * 2^{-i}$$

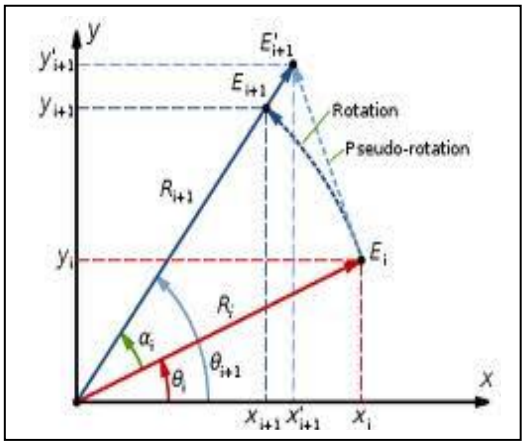


Fig-2.1 Rotation

where $d_i = -1$ if $z_i < 0$, and $+1$ otherwise;

$i = 0, 1, ..., N-1$, and N here, is total number of iterations.

Hence, this gives the following result as N approaches $+\infty$:

$$z_N = 0$$

$$x_N = A_N(x_0 \cos z_0 - y_0 \sin z_0)$$

$$y_N = A_N(y_0 \cos z_0 + x_0 \sin z_0)$$

Where:

$$A_N = \prod_{i=0}^{N-1} \sqrt{1 + 2^{-2i}}$$

.

Rotation angle is between $-\pi/2$ to $\pi/2$

2.2.2 VECTORING

In this method, coordinates of a vector are given and we have to calculate the magnitude and angular argument of original vector [3]. In this mode the rotator (CORDIC) rotates input vector to angle that is required to align resultant vector with x axis. Rotated angle and scaled magnitude of original vector are the result of this vectoring operation. Direction of next rotation is determined by the sign of residual y component. If angle accumulator is initialized to zero, it will hold the traversed angle at the end of iterations.

Equations for vectoring mode are:

$$x_{i+1} = x_i - y_i * d_i * 2^{-i}$$

$$z_{i+1} = z_i + d_i * \text{atan}(2^{-i})$$

$$y_{i+1} = y_i + x_i * d_i * 2^{-i}$$

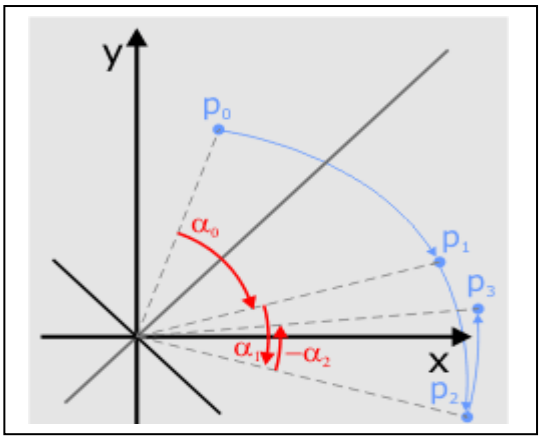


Fig2.2 vectoring

is the angle accumulator

where $d_i = +1$ if $y_i < 0$, and -1 otherwise;

$i = 0, 1, ..., N-1$, and N here, is total number of iterations.

As N approaches $+\infty$:

$$x_N = A_N \sqrt{x_0^2 + y_0^2}$$

$$y_N = 0$$

$$z_N = z_0 + \operatorname{atan}(y_0/x_0)$$

Where:

$$A_N = \prod_{i=0}^{N-1} \sqrt{1 + 2^{-2i}}$$

.

N is chosen so that it is a large-enough constant. A_N is computed before calculation.

2.3 CORDIC IN GENERAL:

Angular implements for rotations should be calculated in a decreasing mode. Angular magnitude for the first rotation can be chosen by various methods. Magnitude of angle chosen for the first rotation is 90°. Let coordinates after rotation of 90° be X1 and Y1.

Y2= ±X1 = R1sin($\theta 1 \pm 90°$)

X2= ±Y1 = R1cos($\theta 1 \pm 90°$)

In the first step a perfect rotation is performed. For this reason first step is unique.For the ith step with vector magnitude of Ri and angle θi with reference to x axis let Xi and Yi are the coordinates:

Yi= Ri(sinθi)

Xi= Ri(cosθi)

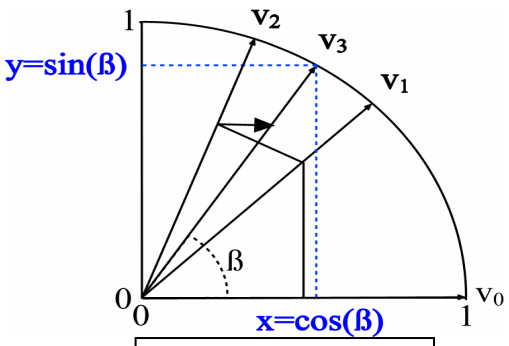


Fig 2.3 iterations

αi be the angle associated with each computing step [3].

$\alpha i = \tan^{-} 2^{-(i-2)}$

It concludes that rotation through ±αi can be accomplished through simple shifting and adding process that describes the simplicity and fast application of CORDIC algorithm.

General equations for rotated components are:

$Y_{i+1} = Yi \pm 2^{-(i-2)}Xi$

$X_{i+1} = Xi \pm 2^{-(i-2)}Y$

Right hand side equations are obtained through simple shift and add operations.

$$x_{i+1} = x_i - \mu d_i y_i 2^{-i}$$
$$y_{i+1} = y_i + d_i x_i 2^{-i}$$
$$z_{i+1} = z_i - d_i e_i$$
$$e_i = \arctan\left(2^{-i}\right) \text{ for } \mu = 1 \text{ (Circular rotation, } i = 0,1,2,...)$$
$$e_i = 2^{-i} \text{ for } \mu = 0 \text{ (Linear rotation, } i = 0,1,2,...)$$
$$e_i = \operatorname{artanh}\left(2^{-i}\right) \text{ for } \mu = -1 \text{ (Hyperbolic rotation, } i = 1,2,3,...)$$

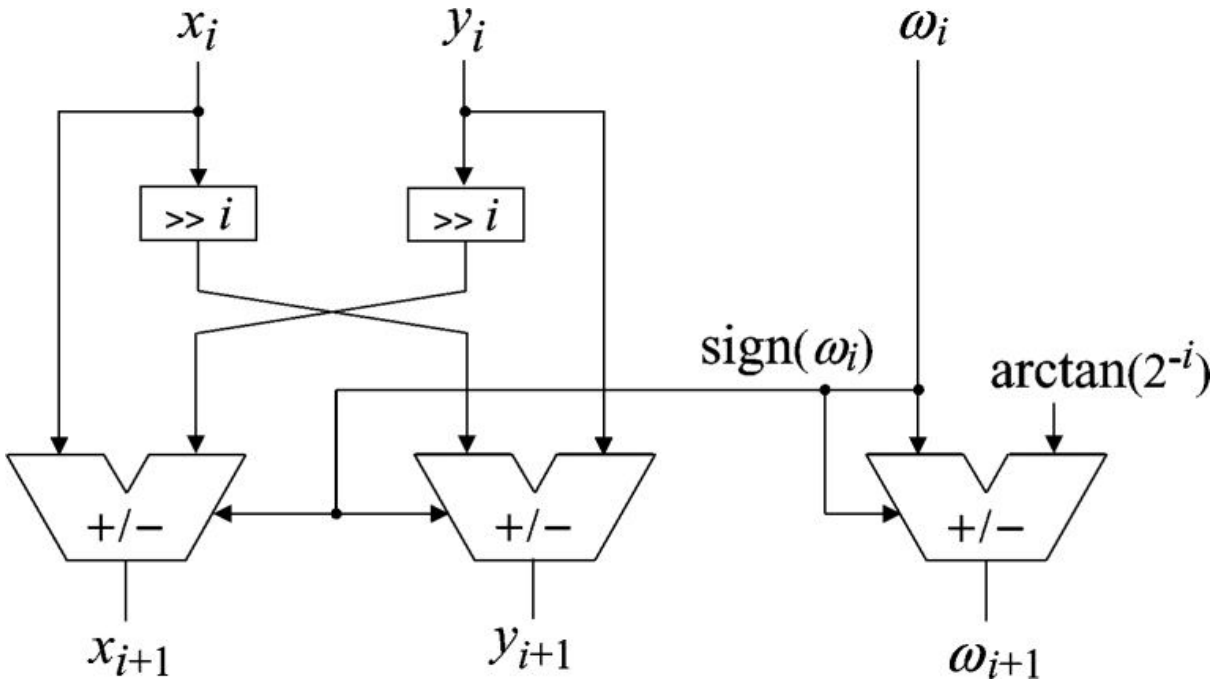
For different modes of rotation different values of the constant is used. Basic CORDIC processor is used for calculating sin and cosine values and in this module circular rotation is used for calculating coordinate values. In case of multiplication and division using CORDIC linear rotation mode is implemented for shifting and adding to get desired results. Exponential function using CORDIC is calculated using hyperbolic rotation mode and logarithm function is implemented using exponential function.

General CORDIC Equations [1]:

Table 2.1 CORDIC modes:

m	rotation mode	vectoring mode
0	$x_n = K(x_o \cos\omega_0 - y_o \sin\omega_0)$	$x_n = K\sqrt{x_o^2 + y_0^2}$
	$y_n = K(y_o \cos\omega_0 + y_o \sin\omega_0)$	$y_n = 0$
	$\omega_n = 0$	$\omega_n = \omega_0 + \tan^{-1}(y_0/x_0)$
1	$x_n = x_o$	$x_n = x_o$
	$y_n = y_o + x_o\omega_0$	$y_n = 0$
	$\omega_n = 0$	$\omega_n = \omega_0 + (y_0/x_0)$
-1	$x_n = K_h(x_o \cosh\omega_0 - y_o \sinh\omega_0)$	$x_n = K_h\sqrt{x_o^2 - y_0^2}$
	$y_n = K_h(y_o \cosh\omega_0 + y_o \sinh\omega_0)$	$y_n = 0$
	$\omega_n = 0$	$\omega_n = \omega_0 + \tanh^{-1}(y_0/x_0)$

Fig 2.4 :Hardware Implementation of CORDIC[1]:



2.4 CORDIC IMPLEMENTATION

2.4.1 MULTIPLICATION

The CORDIC algorithm used for multiplication can be derived from the following series[5]:

$$z = x*y$$

$$= y*\sum_{i=1}^{B} ai*2^{-i}$$

$$= \sum_{i=1}^{B} ai*(y*2^{-i})$$

above equations indicate that z is made of shifted versions of y. The coefficients a_i are calculated by making x equal to zero bitwise. If ith bit of x is not zero then is shifted right by ith bit and final value is added to current value of z. Then by subtracting 2^i from x, the ith bit is removed. When x is negative, adding 2^{-i} removes ith bit in twos complement format. In both cases z is signed product of x and y corrected to B bits. It is similar to shift and add algorithm for multiplication. But here instead of left shifts right shifts are used that allows signed numbers for use. It is similar to rounding of standard multiplication algorithm result to most significant B bits. Algorithm is as follows[5]:

```
multiply(x,y)
 {
  for(i=1;i<=R;i++)
    {
     if(x>0)
       x=x- x*2^(-i)
       z=z+ y*2^(-i)
```

```
    else
      x=x+ x*2^(-i)
      z=z- y*2^(-i)
    }
  return(z)
}
```

Here x and y are assumed to be fractional ranging from -1 to 1. Algorithm can be allowed for higher ranges if decimal point is allowed to float.

CORDIC equations for multiplication:

2.4.2 DIVISION

Division algorithm cab be derived from the equation: z=x/y

Writing z in expanded series form we get x - z*y=0 [5]

$x- y*\sum_{i=1}^{B} ai*2^{-i} = 0$

$x- \sum_{i=1}^{B} ai*(y*2^{-i}) = 0$

Quotient z can be calculated bitwise by making x is equal to zero through right shifted versions of y. ith bit of z is set if current residual is non-negative. ith bit of z is cleared if current residual is negative. Algorithm is as follows[5]:

```
  divide(x,y)
 {
   for(i=1;i<=R;i++)
  {
```

```
    if(x>0)
      x=x- y*2^(-i)
      z=z+ z*2^(-i)
     else
       x=x+ y*2^(-i)
       z=z- z*2^(-i)
     }
   return(z)
}
```

2.4.3 EXPONENTIAL AND LOGARITHMIC

2.4.3.1 THEORY

sinh and cosh (hyperbolic functions) and exponential functions can be calculated in rotation mode. Exponential value is[6]:

$$e^x = \cosh(x) + \sinh(x)$$

Logarithmic functions can be calculated using vectoring mode($Z_0=0$):

$$Z_n=\tanh^{-1}(y/x)$$

If x=w+1 and y=w-1 then

$$Z_n=1/2 \ln w$$

2.4.3.2 EXPONENTIAL FUNCTION IMPLEMENTATION

Outputs Xn and Yn gives the hyperbolic function values cosh(z) and sinh(z). It affects the scale factor value K at input (X) and zero at the input (Y)[6].

$$z=z_1 + p\ln 2$$

Here p is an integer (z/ln2)

To implement exponential, we use[6]:

$e^z = e^{zl+p\ln 2} = e^{zl}\, e^{p\ln 2} = 2^p\, e^{zl}$

2^p corresponds to p number of left shifts.

2.4.3.3 LOGARITHMIC FUNCTION IMPLEMENTATION

w+1 affects the input (X) and w-1 affects input (Y) resulting ½ ln(w) at output (Zn)[6]. For hyperbolic rotation mode:

$x_{i+1} = x_i + y_i * d_i * 2^{-i}$

$y_{i+1} = y_i + x_i * d_i * 2^{-i}$

$z_{i+1} = z_i - d_i * \tanh^{-1}(2^{-i})$

where $d_i = -1$ if $z_i < 0$, $+1$ else

CHAPTER 3

RELAY AND DIGITAL ALGORITHM

3.1 RELAY:

Relay is defined as an switch operated electrically. When it is necessary to control a circuit by low-power signal or where one signal controls several circuits. Protective relays are digital instruments used in modern power systems. They have calibrated operating characteristics and are used for protecting circuits from faults or overloads.

3.2 RELAY TRIPPING THEORY:

Calculating the rms value of the fundamental frequency (current) and deciding if that calculated current exceeded a threshold (already given) is one of the major functions of the cordic algorithm [9]. If the calculated current is more than threshold value, relay algorithm issues a command that trips the circuit breaker after a time delay. As the current changes during any fault, the instant when trip command is to be issued is determined by the following equation [9]

$$\int y(t)dt > K$$

Where y(t)=0 if current is less compared to pick up current value and y(t) = K/tr(t) if current if greater than or equal to pick up current value. Here K is the target number, tr(t) is operating time of relay for the current observed at time t[9].

As digital relays are nothing but quantized value of sample of currents taken at an interval of ΔTs, numerical integration is performed:

$$\sum_{m=1}^{N} (X)m > K$$

Here (X)m=0 if current is less than the pick-up current value. (X)m = KΔT/(tr)m if current is greater than or equal to pick-up value. m represents m^{th} sample after current exceeds the pick-up value for first time. tr(m) represents relay operating time which corresponds to rms current that is estimated while receiving the m^{th} sample. NΔT is the required time for tripping. N is not known but ΔT is. Therefore calculating values of (X)m is required after each sample.

3.3 ALGORITHM:

Voltage proportional to current in the protected circuit is the input to relay logic. Sampling of the voltage is done at a given rate. The quantized data are processed so that rms value of the fundamental frequency component of current can be determined [9].

- Variable "SUM" is set to zero.
- New sample that represents the relay current is obtained.
- RMS value of the fundamental frequency component of the current is determined.
- Calculated rms value is compared with the pick-up current value.
- If calculated value is greater than or equal to pick-up current value, then next step is followed or else last step is followed.
- (X)m value corresponding to the already computed values is obtained.
- This number is available for few values of currents and for rest values, interpolation is restored.

- Numbers starting from 6^{th} step to the variable "SUM" are added.
- "SUM" is larger than or equal to the target K is checked. If it satisfies, then a command to trip is issued or else next step is followed.
- We wait until intersampling time has elapsed and then algorithm reverts back to 2^{nd} step.
- Again the relay resetting process is started.

The overcurrent relay is reset as soon as current inside protected circuit becomes less than pick-up current value. Each time rms current value is less than the pick-up value resetting is done by diving "SUM" with 2[9].

3.4 INVERSE TIME OVERCURRENT RELAYS:

3.4.1 TIME-CURRENT RELATION:

CO and IAC are the most frequently used electrochemical overcurrent relays. With time dials ranging from ½ to 11, they have moderately inverse time current relay. Overcurrent relays are used for protection of lines connected in series and different time dial settings are selected for this. 300-400 ms of time interval is required between operating curves of the two relays adjacent to each other. This setting is required so that correct operation of the devices can be done in a sequence and unnecessary trips are avoided. Most frequently used static analog relay is GEC MOGG type. It uses equations and curves of time-current as follows [7]:

$$t_{trip}(I) = TD(A/M^{p}-1)$$

$$t_{reset}(I) = TD(tr/1-M^{q})$$

where TD= time dial options

t_{trip}= operating time to trip in seconds

t_{reset}= result time in seconds

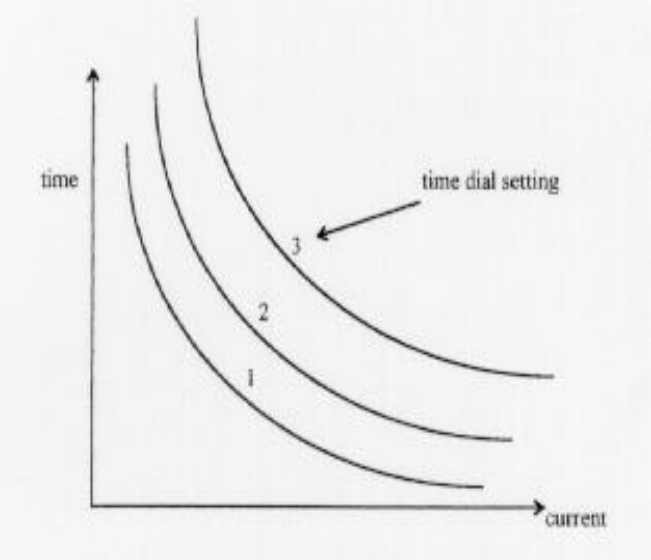
M= multiplier of pick-up current

P= exponent constant

q= exponent constant

A= constant

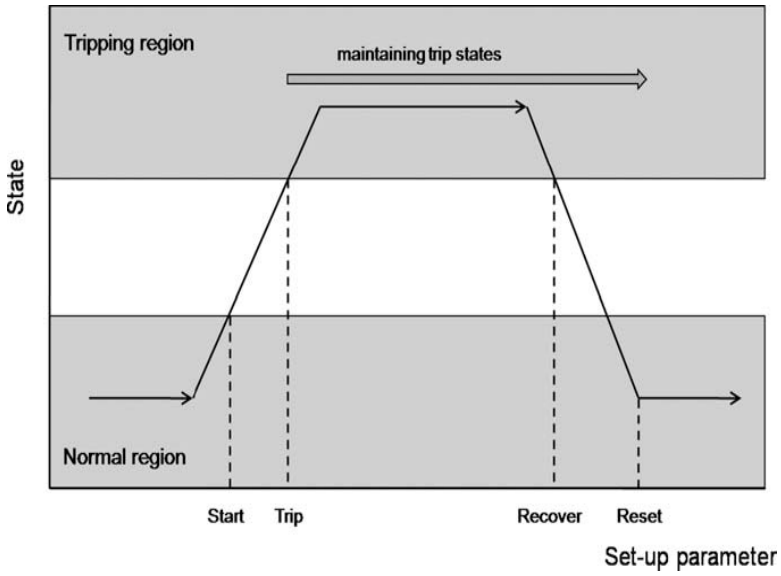
Fig 3.1:Relation between current and time at different Time Dial Setting:



3.4.2 OVERCURRENT RELAY SWITCH AND ALGORITHM:

Protective relays are used to show the current status of a given power system. It is used to determine whether parameters like voltage or current violate any trip or reset specification or condition. Over-current relay is a measuring relay that operates when the value of the current exceeds the setting (operating value) of the relay. Fig(3.2) shows the abnormal behaviour in the over-current system.

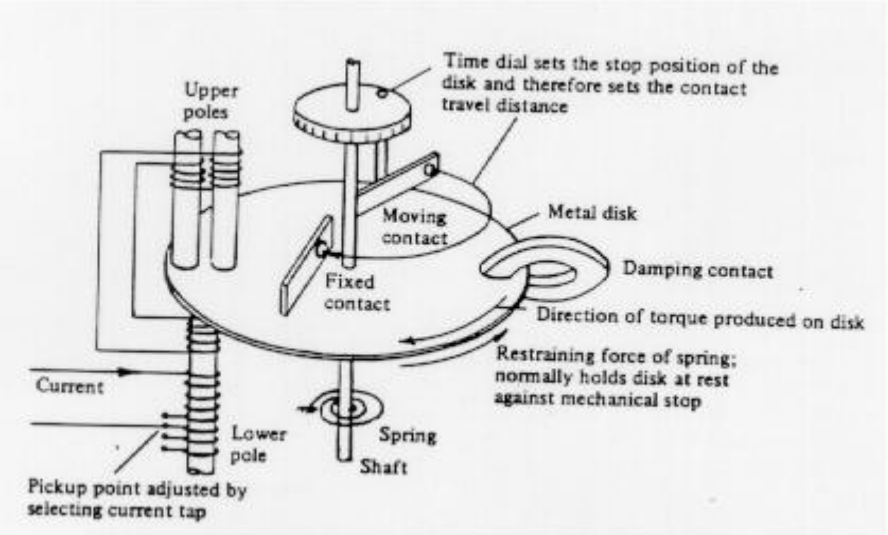
Fig 3.2 Trip State



A time over-current relay is a relay system with an inverse time characteristics (inverse means that relay operates faster as the current value increases). The predetermined value of current is called pick-up current. We refer to the time curves which are families of curves that are scaled in time dials. The higher the value of time dial is, the longer the delay becomes at each current level. In case of digital relays, the data acquisition and computation is to be done discretely.

The most commonly used overcurrent relay usually incorporates both, the instantaneous unit as well as the time overcurrent unit. The instantaneous response is mostly provided by a moving armature unit. Its basic purpose is to operate on large currents. The inverse time response is mainly provided by an induction disk unit of the system and is set to operate for lower fault currents. The induction disk unit operates on similar principle as an induction motor.

Fig 3.3 Time Dial Setting



The metal disk gets mounted on a shaft which can freely rotate. The current coils are kept fixed. These create magnetic field which induces eddy currents in the metal disk. Then, the magnetic field of the eddy currents interact with the magnetic field of the stationary coils and this produces torque on the disk. Here, the disk and its shaft rotate and thus bringing the moving contact towards the fixed contact, getting into a closed position. Now, the motion of the shaft is opposed by a spring which then returns the disk and the moving contact into an open position when the current drops below a particular value. The time to close the contact depends on the contact travel distance which is set with the help of a time dial. The pick-up current is adjusted by selecting the current taps on the current coil. The relays are usually available with three ranges of current taps of value: 0.5 to 2.0 A, 1.5 to 6.0 A, and 4 to 16 A. The time dial has normally positions marked as 0 to 10, where for 0 setting, the contact is permanently closed.

.

Since in this case, we use non-definite time protection groups, the following characteristic equations are utilized[10]:

$I_r=I_{RMS}/I_{pu}$

$$t(Ir)=\begin{cases} \left(\left(\frac{A}{Ipr-1}\right)+B\right)TDS, & Ir>1 \\ \left(\frac{tr}{Iqr}\right)TDS, & 0<Ir<1 \end{cases}$$

Here A, B, p, q, tr and TDS (time dial setting) are relay-specific parameters.

Ipu denotes the pre-determined pickup current. In our work, inverse relaying algorithms related to voltage and frequencies other than current are also based on above equations.

The inverse relay generates a trip signal when,

$To \leq (Ir)$

where To is the tripping time delay. Then we can rewrite above equation when Ir>1 as:

$\sum_{i=1}^{N} \Delta t/t(Ir,i)$

Dt denotes sampling period every relay input. Finally, in digital circuits, we can determine the trip condition by testing the following inequality:

$$\sum_{i=1}^{N} Ip,r,j-1/(A+B(Ip,r,j-1)) \geq TDS/\Delta T$$

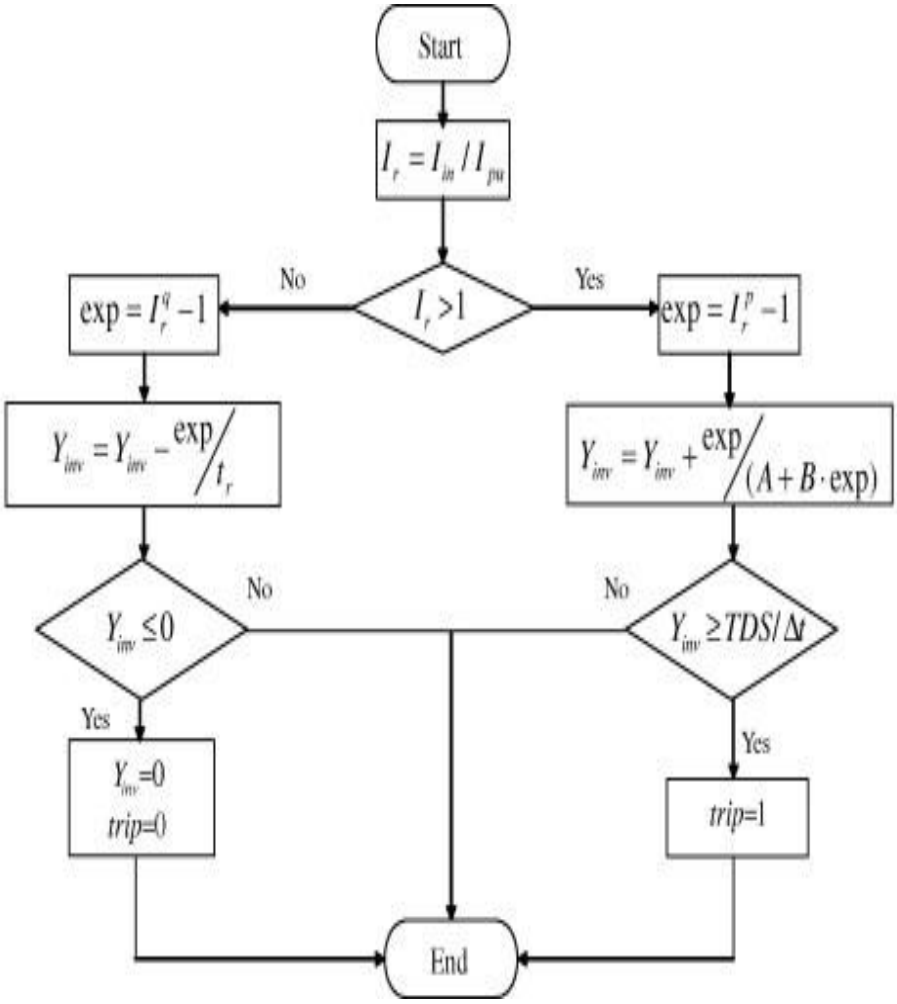
Where $Ip,r,j= I^{p}_{r,j}$

Here, delta t refers to the sampling period per relay input. From [10],

the flowchart of the algorithm is given in below figure. If Ir is greater than unity, the relay behaviour is affected by the given characteristics. Or else, it follows the reset decaying method that follows equation mentioned above. The CORDIC algorithm is used in the following

section to implement the above equations efficiently.

Fig 3.4: Flow chart for inverse-time protection:



[7] Software model for inverse time overcurrent relays incorporating IEC and IEEE standard curves.

In this project, we attempt to simulate a CO type relay, using the above mentioned equations to model the basic time current curves. Table 3.1 shows the basic values for the required parameters for a custom approach depending on which CO type model we choose [7].

Table 3.1: Constants and exponents for CO type relays

	A	B	K	p	q	t_r
CO-2	0.2663	0.03395	0.028	1.2969	2.0	0.5
CO-5	5.6143	2.18592	0.028	1.0	2.0	15.75
CO-6	0.4797	0.21359	0.028	1.5625	2.0	0.875
CO-7	0.3022	0.1284	0.028	0.5	2.0	1.75
CO-8	8.9341	0.17966	0.028	2.0938	2.0	9.00
CO-9	5.4678	0.10814	0.028	2.0469	2.0	5.5
CO-11	7.7634	0.02758	0.028	2.0938	2.0	7.75

If for all cases K is assumed as zero and q as 2.0, then the equations and parameters are modified to take the form given by IEEE Std-C37.122-1996. The constants are:

Table 3.2: IEEE constants and exponents for CO relays

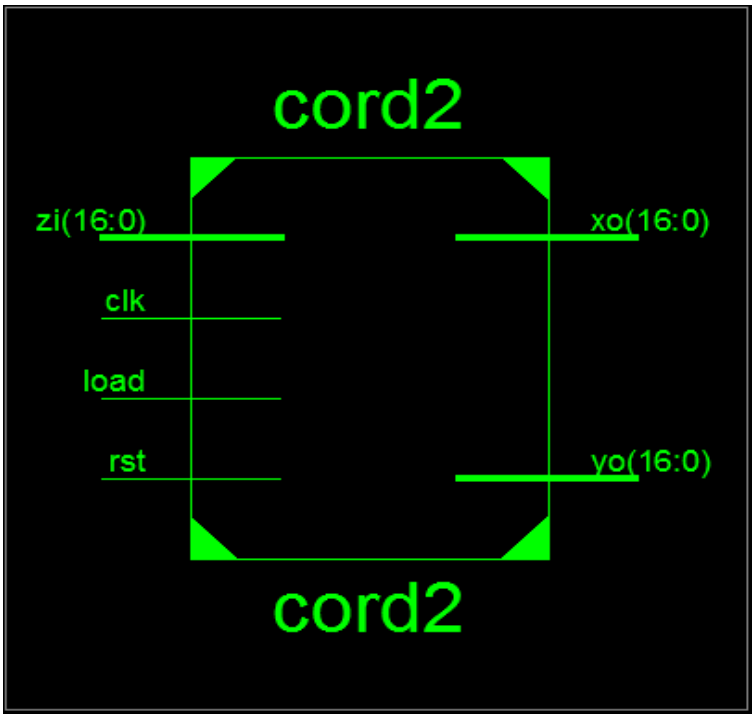
	A	B	P	q	t_r
CO-7	0.0094	0.0366	0.02	2.00	1.08
CO-9	3.784	0.0984	2.00	2.00	4.2
CO-11	5.616	0.026	2.00	2.00	5.3

CHAPTER 4

RESULTS AND SIMULATIONS

4.1 BASIC CORDIC PROCESSOR

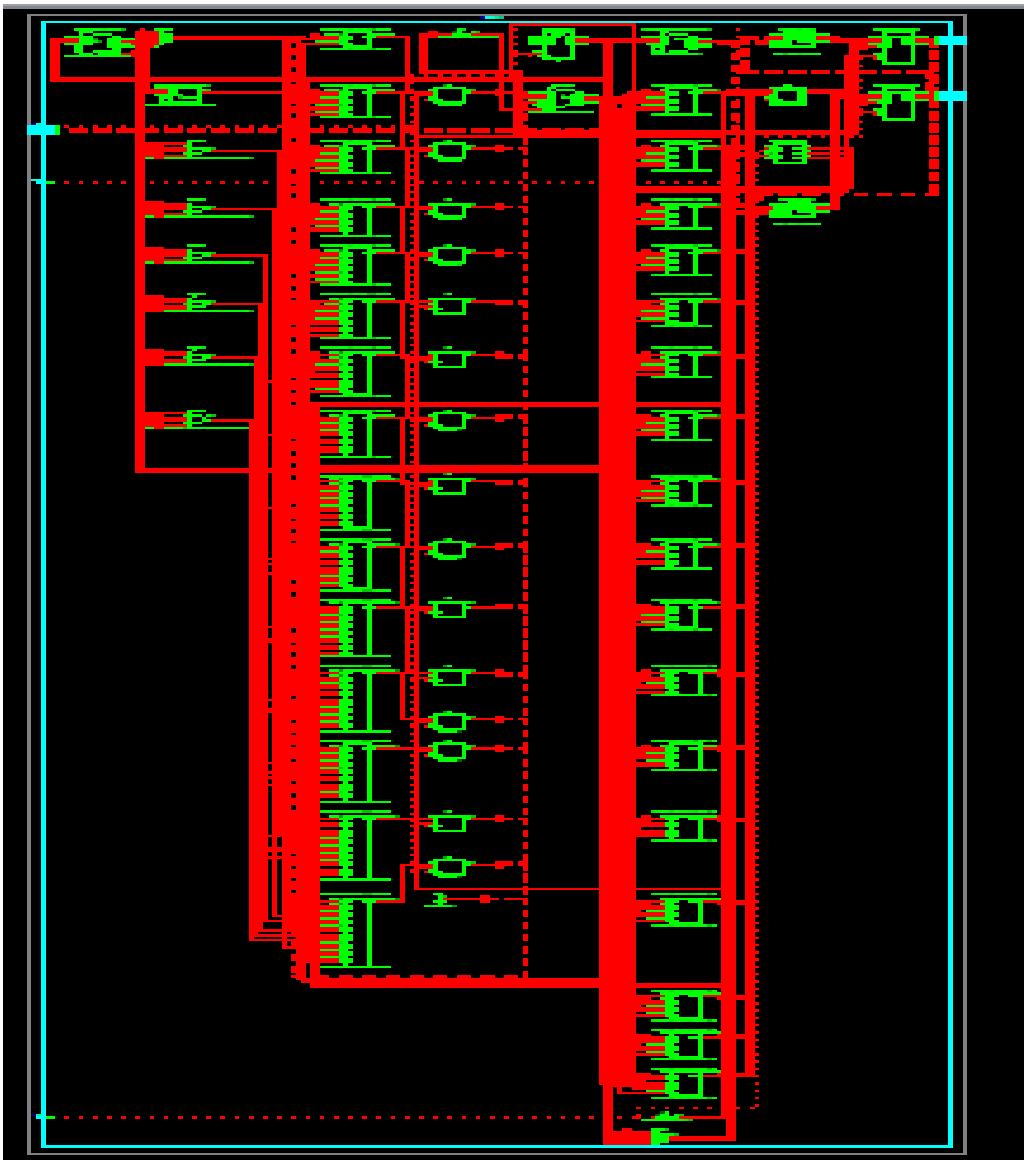
4.1.1 BLOCK DIAGRAM:



4.1.2 SUMMARY:

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	153	5472	2%
Number of Slice Flip Flops	74	10944	0%
Number of 4 input LUTs	289	10944	2%
Number of bonded IOBs	54	240	22%
Number of GCLKs	1	32	3%

4.1.3 RTL SCHEMATICS:



Input:

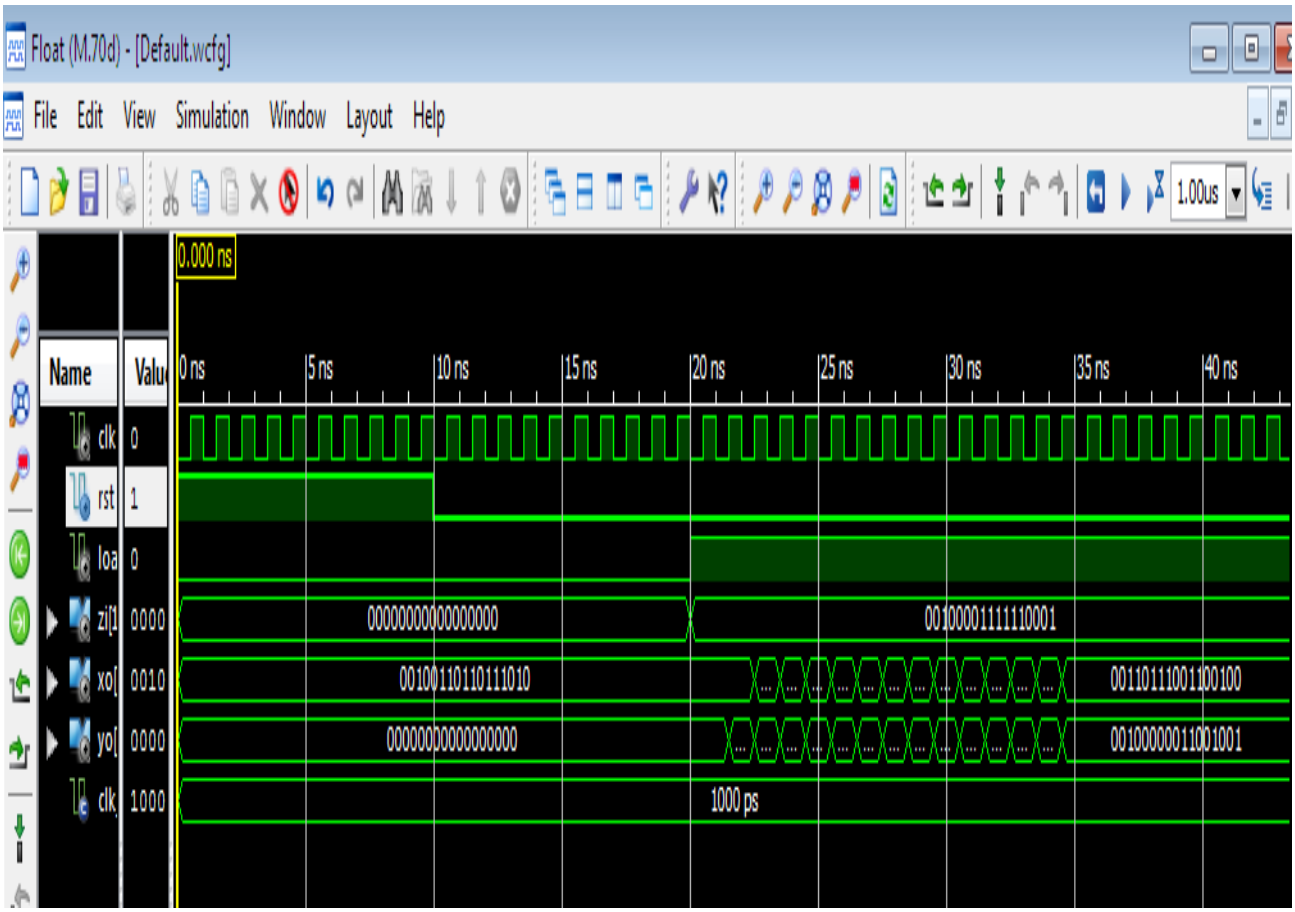
Zo(16:0)

Outut:

Xo(16:0)

Yo(16:0)

4.1.4 TEST BENCH:



4.1.5 RESULTS:

Input:

Zi(angle)= pi/6

=0.5235 =00100001111110001

Output:

Xi = cos(Zi) = 00110111001100100

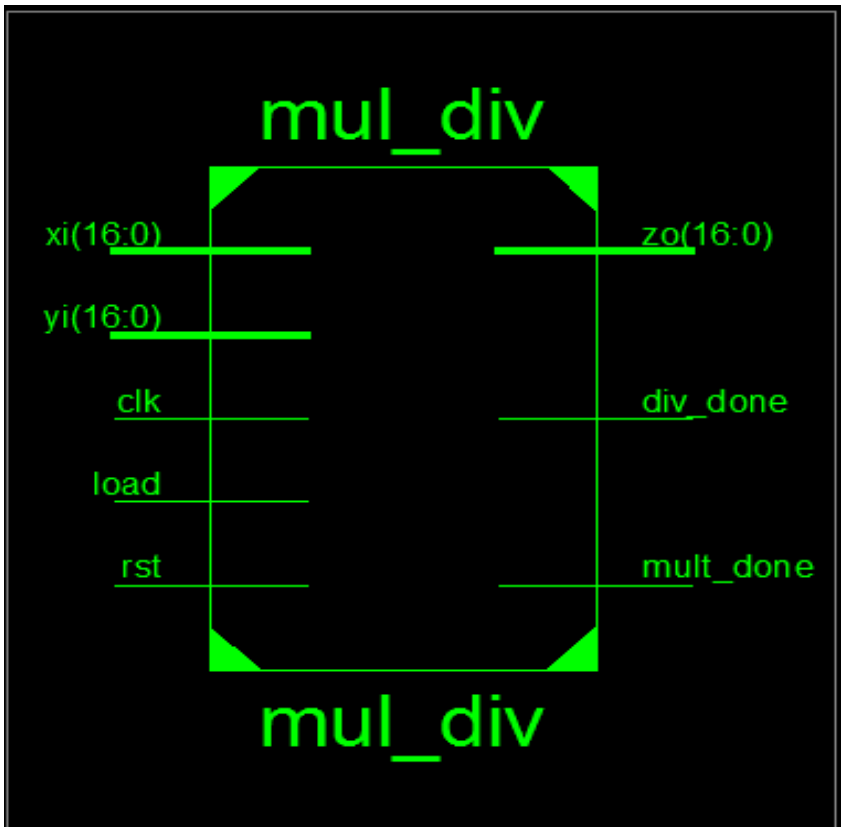
= 0.8624

Yi = sin(Zi) = 00100000011001001

= 0.5061

4.2 MULTIPLICATION AND DIVISION USING CORDIC:

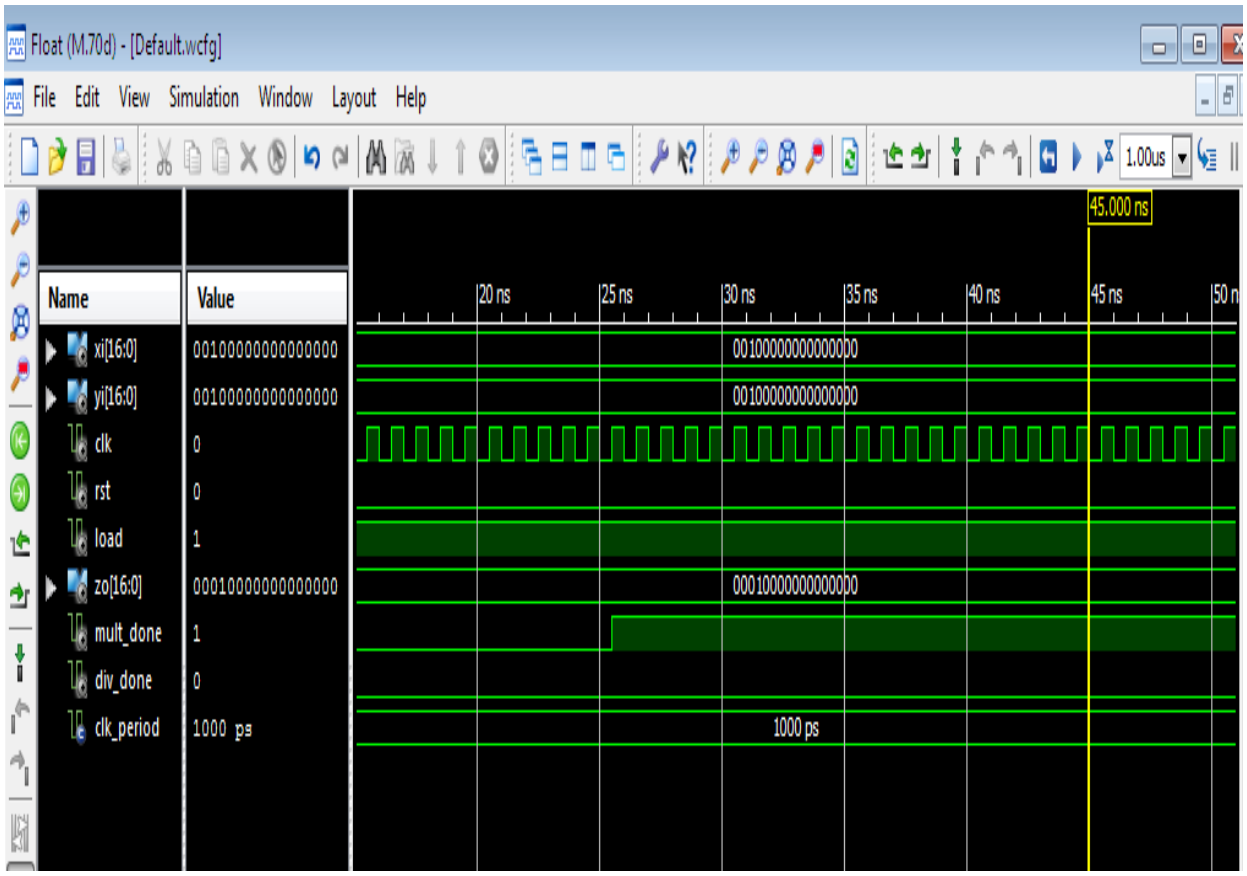
4.2.1 BLOCK DIAGRAM:



4.2.2 SUMMARY:

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	174	5472	3%
Number of Slice Flip Flops	45	10944	0%
Number of 4 input LUTs	336	10944	3%
Number of bonded IOBs	56	240	23%
Number of GCLKs	1	32	3%

4.2.3 TEST BENCH FOR MULTIPLICATION:



4.2.4 RESULTS:

Multiplication (load=1)

Input:

xi=00100000000000000 =0.5

yi=00100000000000000 =0.5

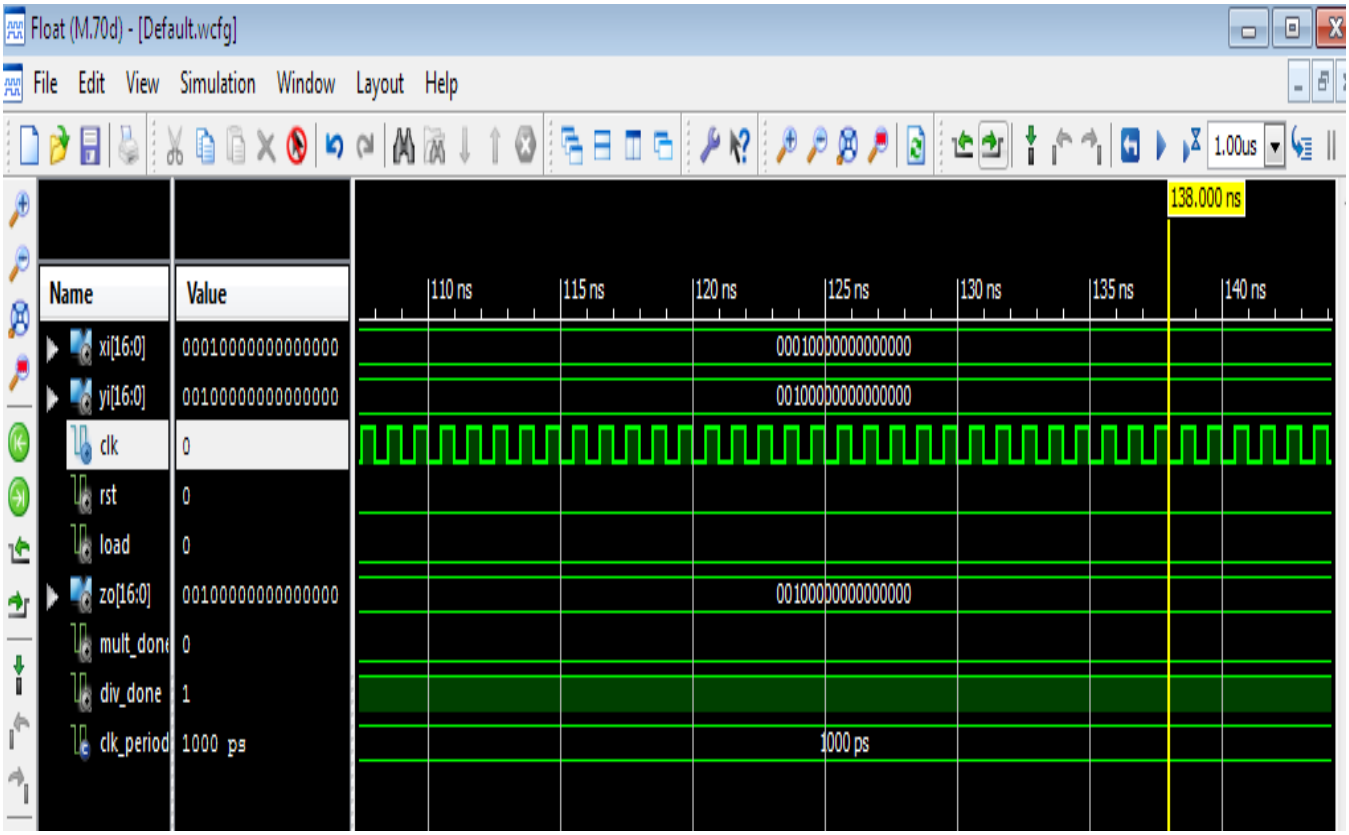
Output:

zo= 0.25 =00010000000000000

mult_done=1

div_done=0

4.2.5 TEST BENCH FOR DIVISION:



4.2.6 RESULTS:

Division (load=0)

Input:

Xi= 00010000000000000 =0.25

Yi= 00100000000000000 =0.5

Output:

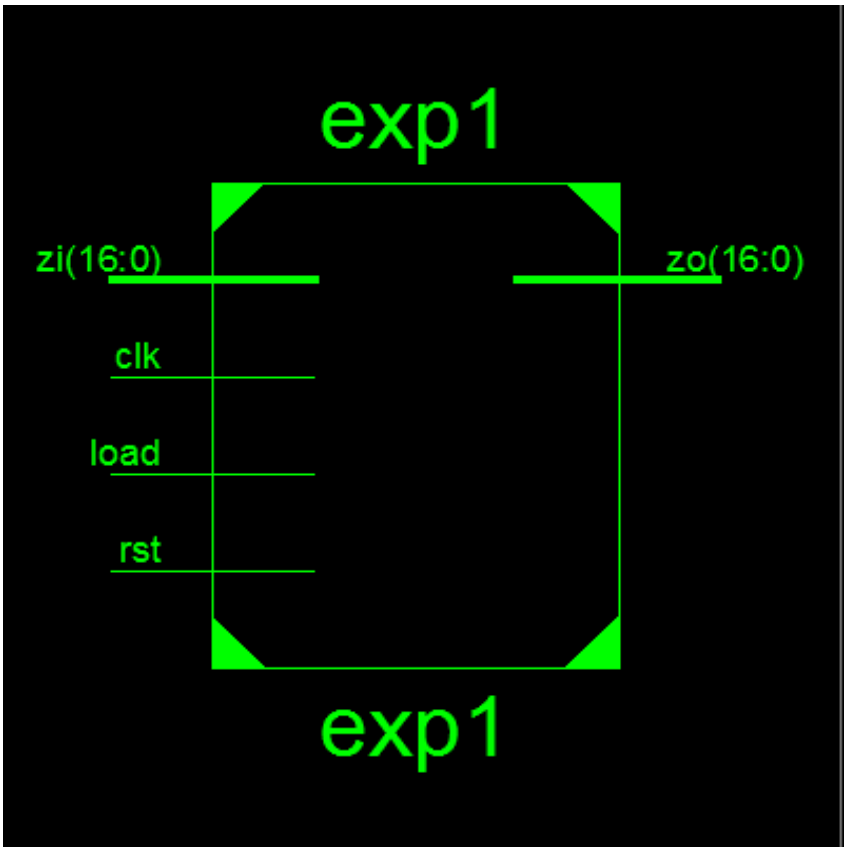
Z0= 00010000000000000 =0.25 (Xi/Yi)

mult_done=0

div_done=1

4.3 EXPONENTIAL USING CORDIC:

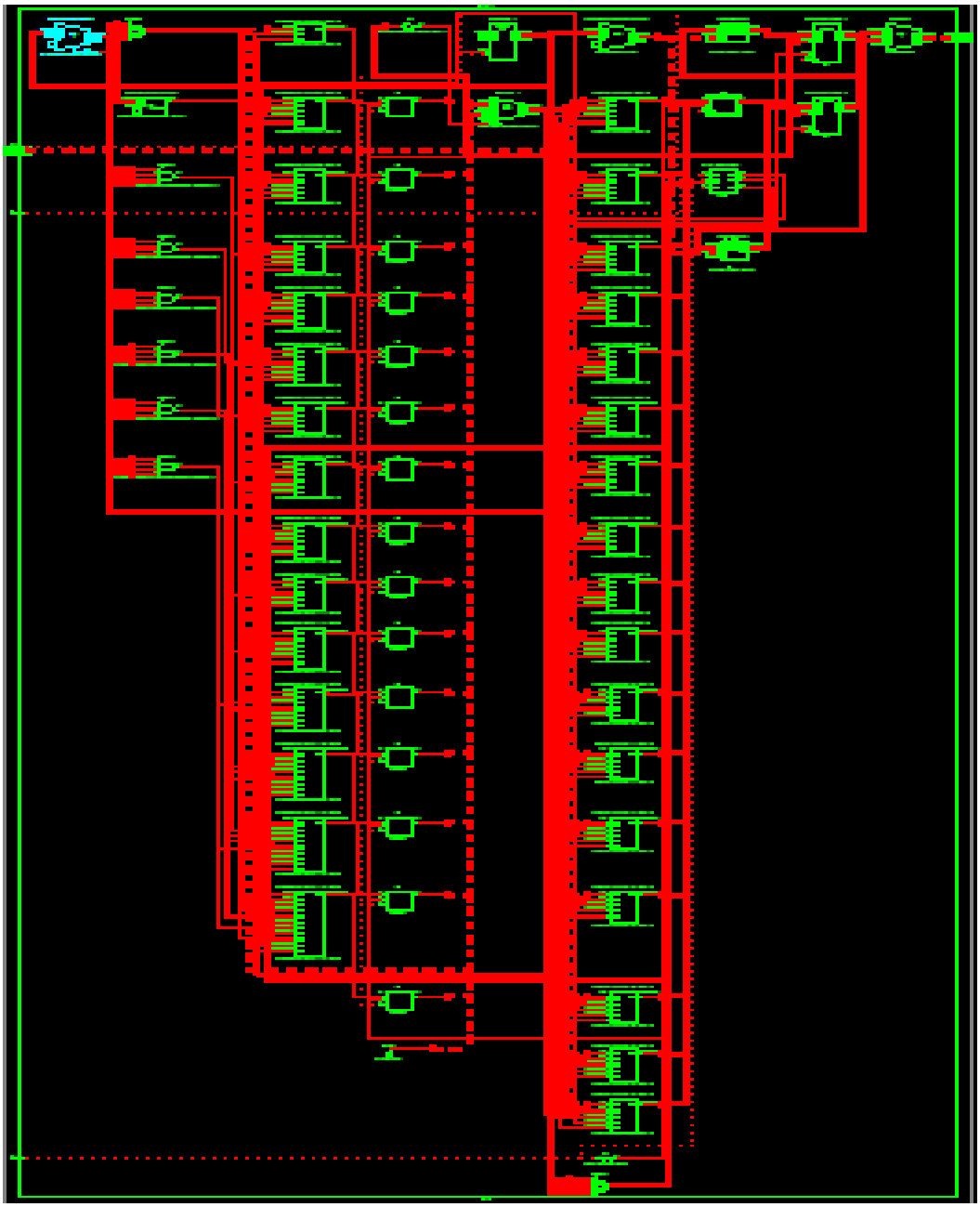
4.3.1 BLOCK DIAGRAM:



4.3.2 SUMMARY:

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	159	5472	2%
Number of Slice Flip Flops	81	10944	0%
Number of 4 input LUTs	304	10944	2%
Number of bonded IOBs	37	240	15%
Number of GCLKs	1	32	3%

4.3.3 RTL SCHEMATICS:



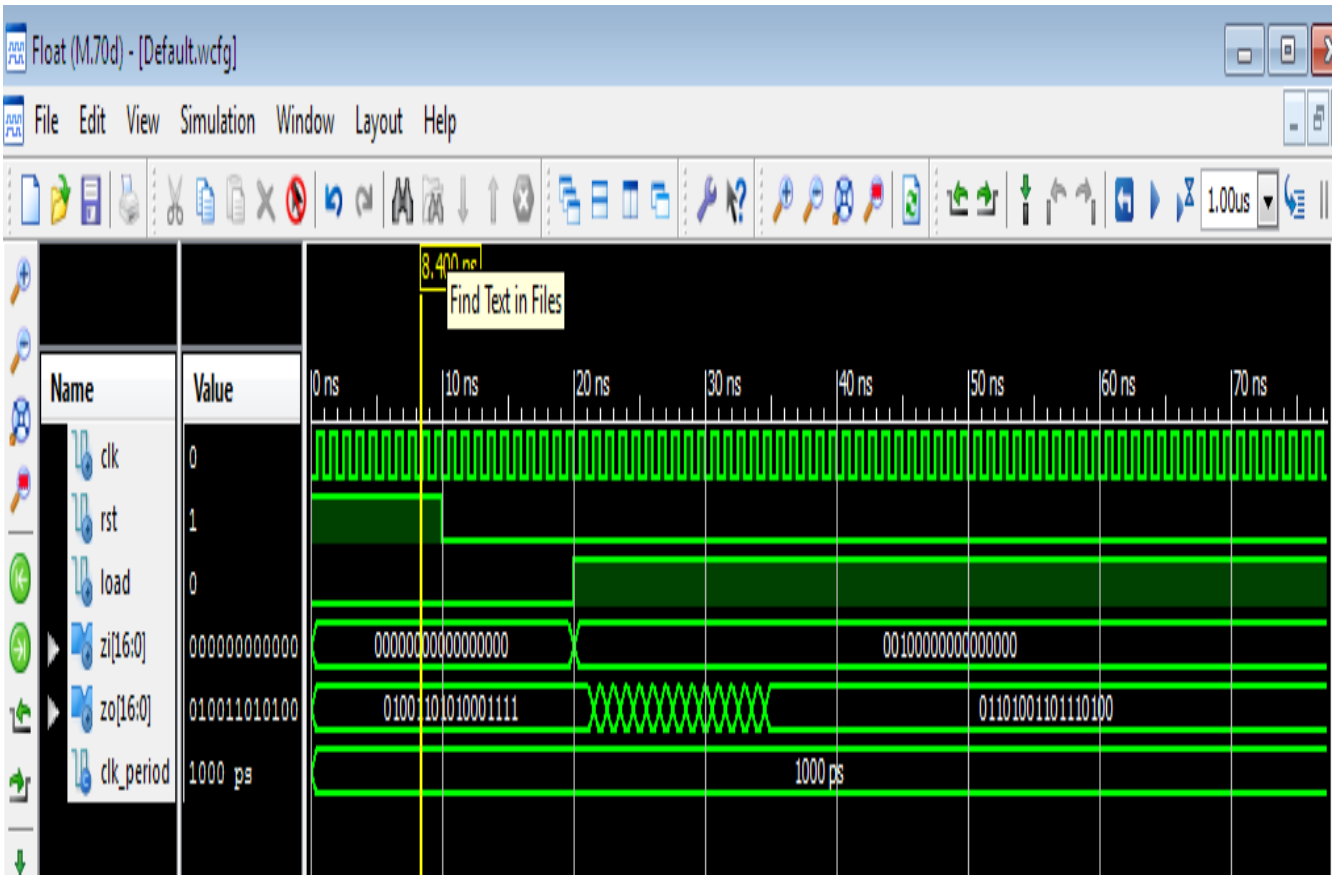
Input:

Zi(16:0)

Output:

Zo(16:0)

4.3.4 TEST BENCH:



4.3.5 RESULTS:

Input:

$e^{0.5}$

Output:

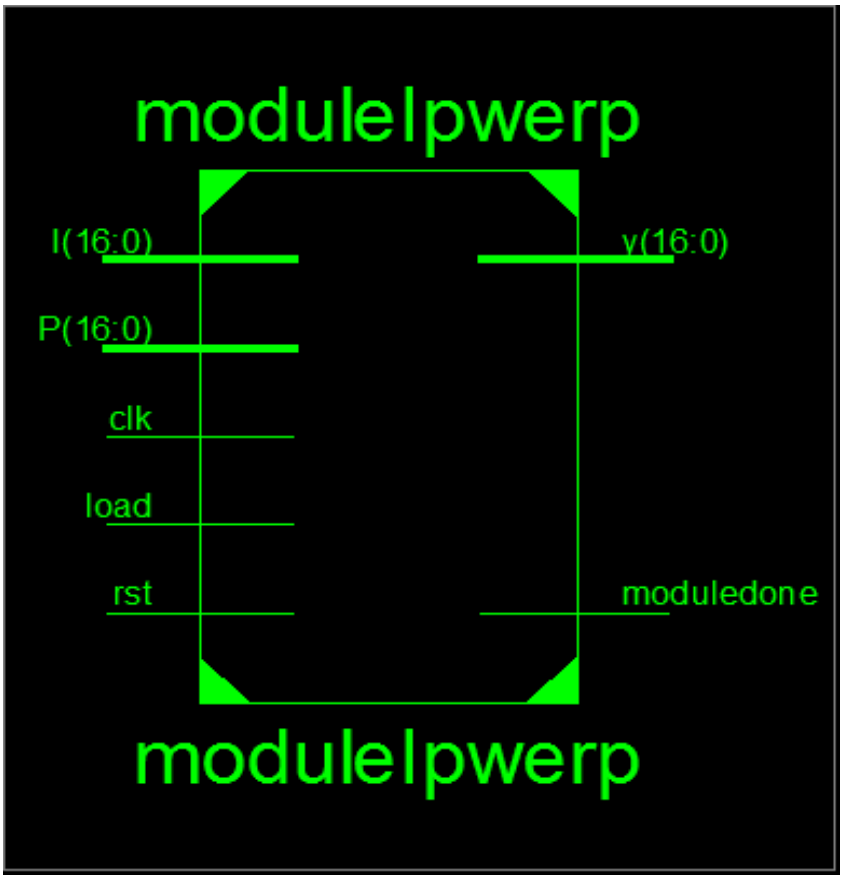
Z0=1.6519

Actual answer:

1.6487

4.4 LOGARITHM USING CORDIC:

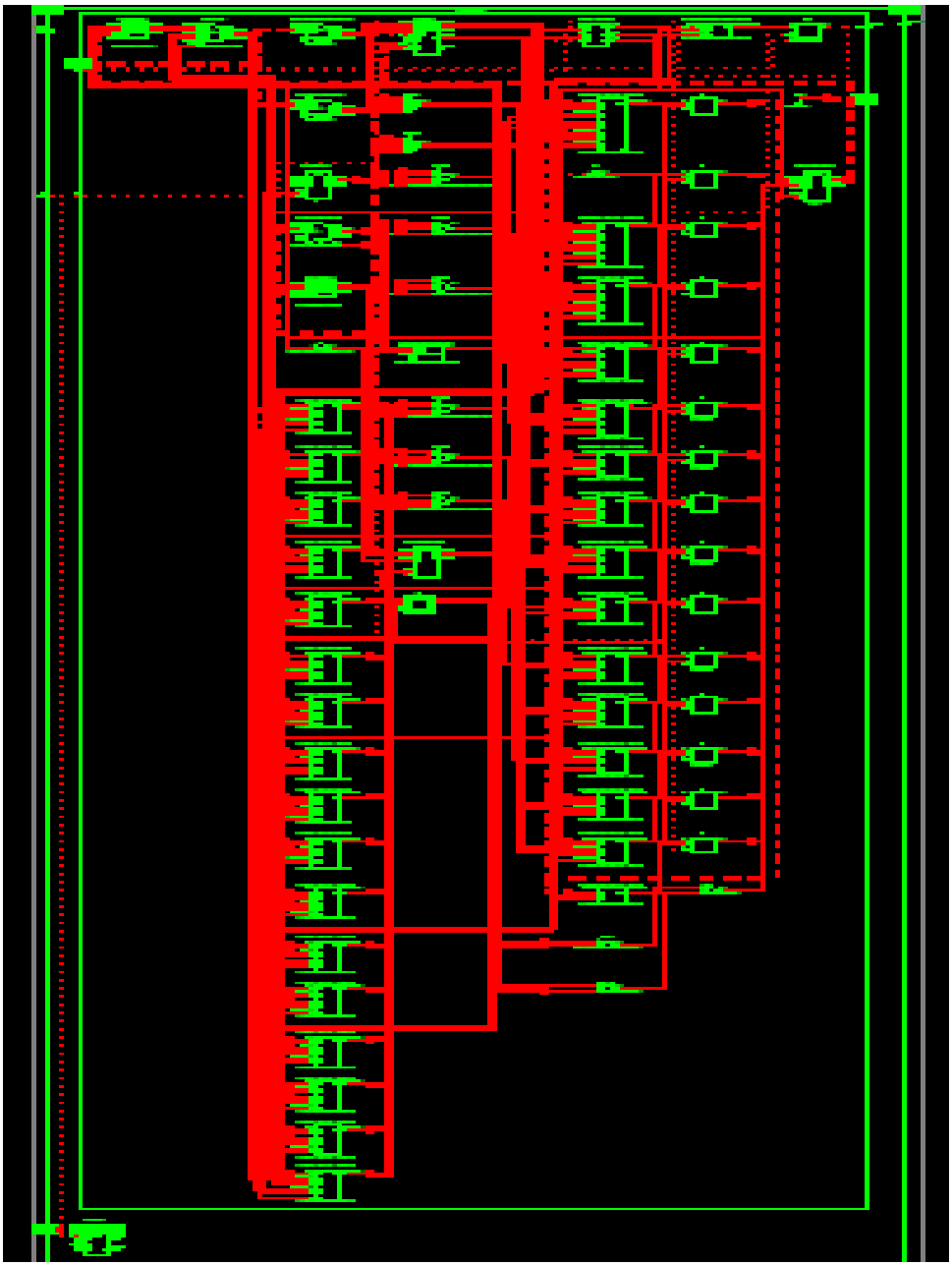
4.4.1 BLOCK DIAGRAM:



4.4.2 SUMMARY:

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	649	5472	11%
Number of Slice Flip Flops	269	10944	2%
Number of 4 input LUTs	1238	10944	11%
Number of bonded IOBs	55	240	22%
Number of GCLKs	1	32	3%

4.4.3 RTL SCHEMATICS:



Input:

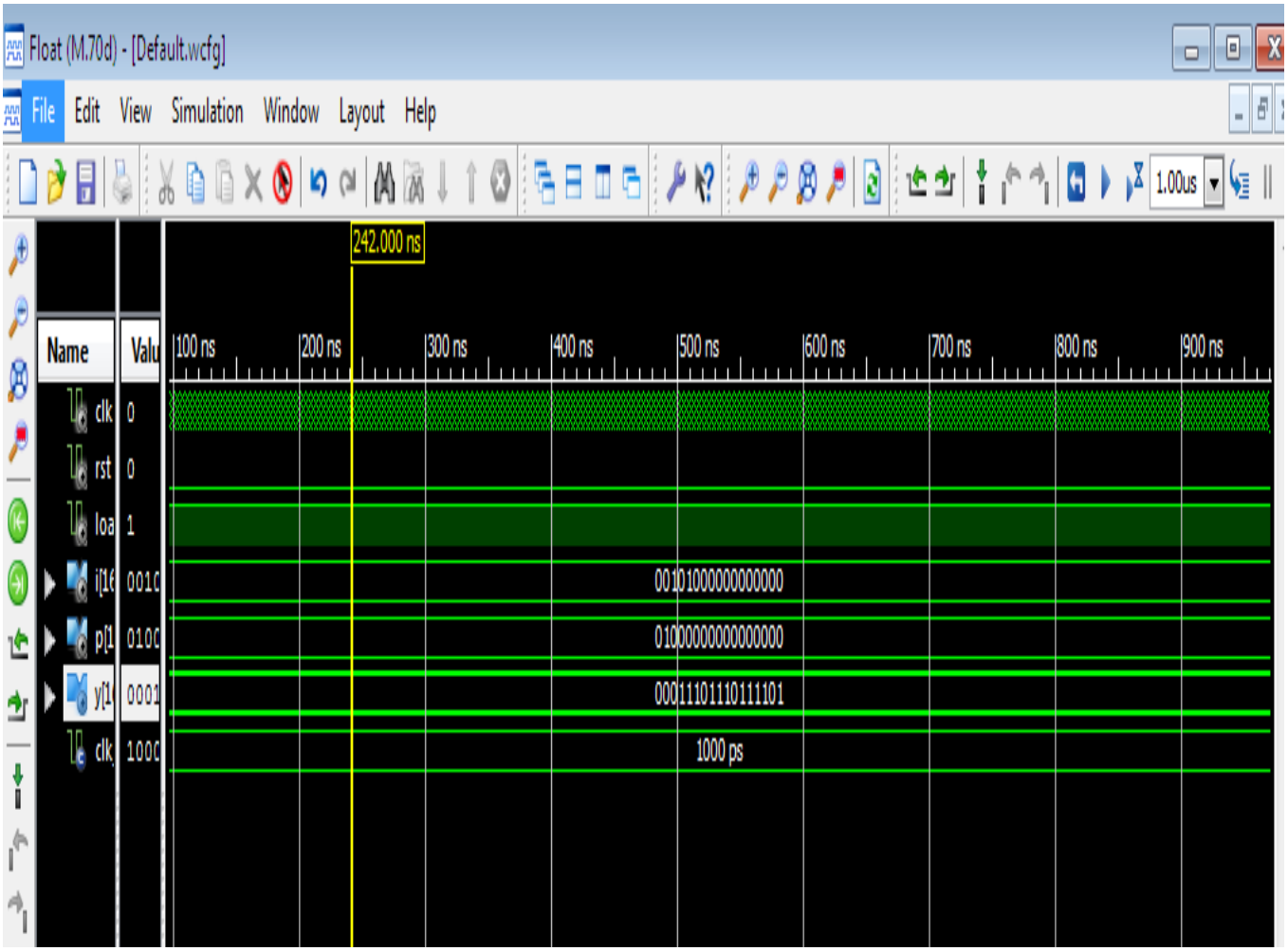
I(16:0)

P(16:0)

Output:

V(16:0)

4.4.4 TEST BENCH:



4.4.5 RESULTS:

Input:

I=0.5 ; P=1

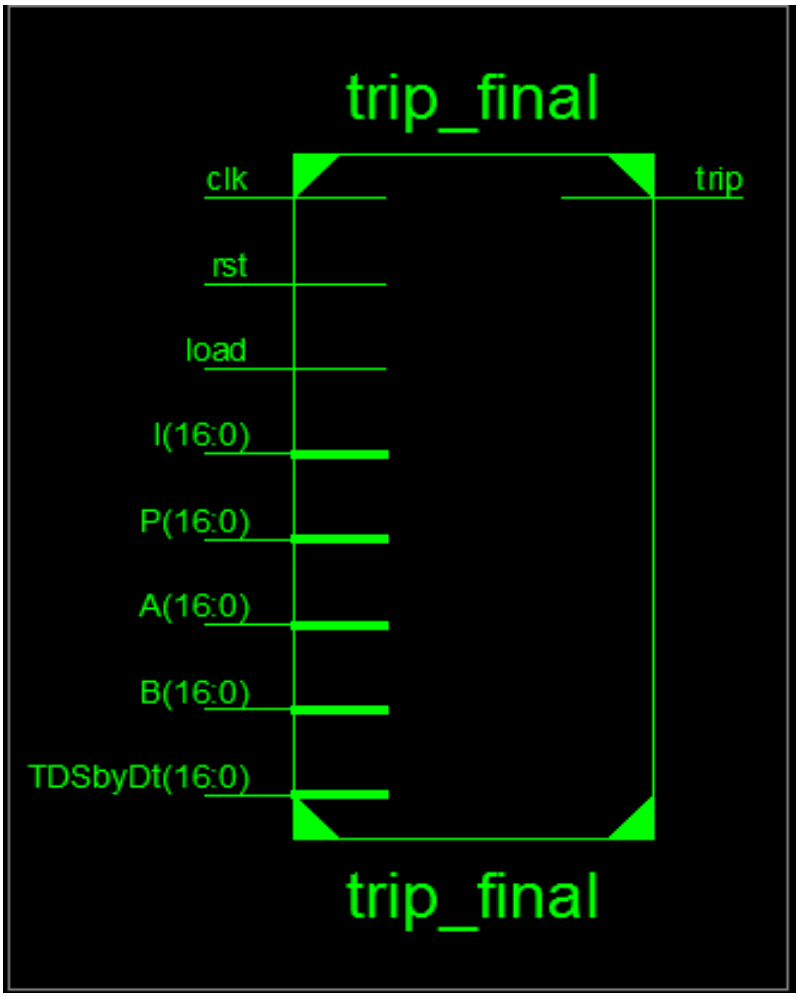
Output:

(I^P)

V=0.48

4.5 PROTECTIVE RELAY IMPLEMENTATION:

4.5.1 BLOCK DIAGRAM:



From table 3.1 we get the following values:

P=1.56

A=0.47

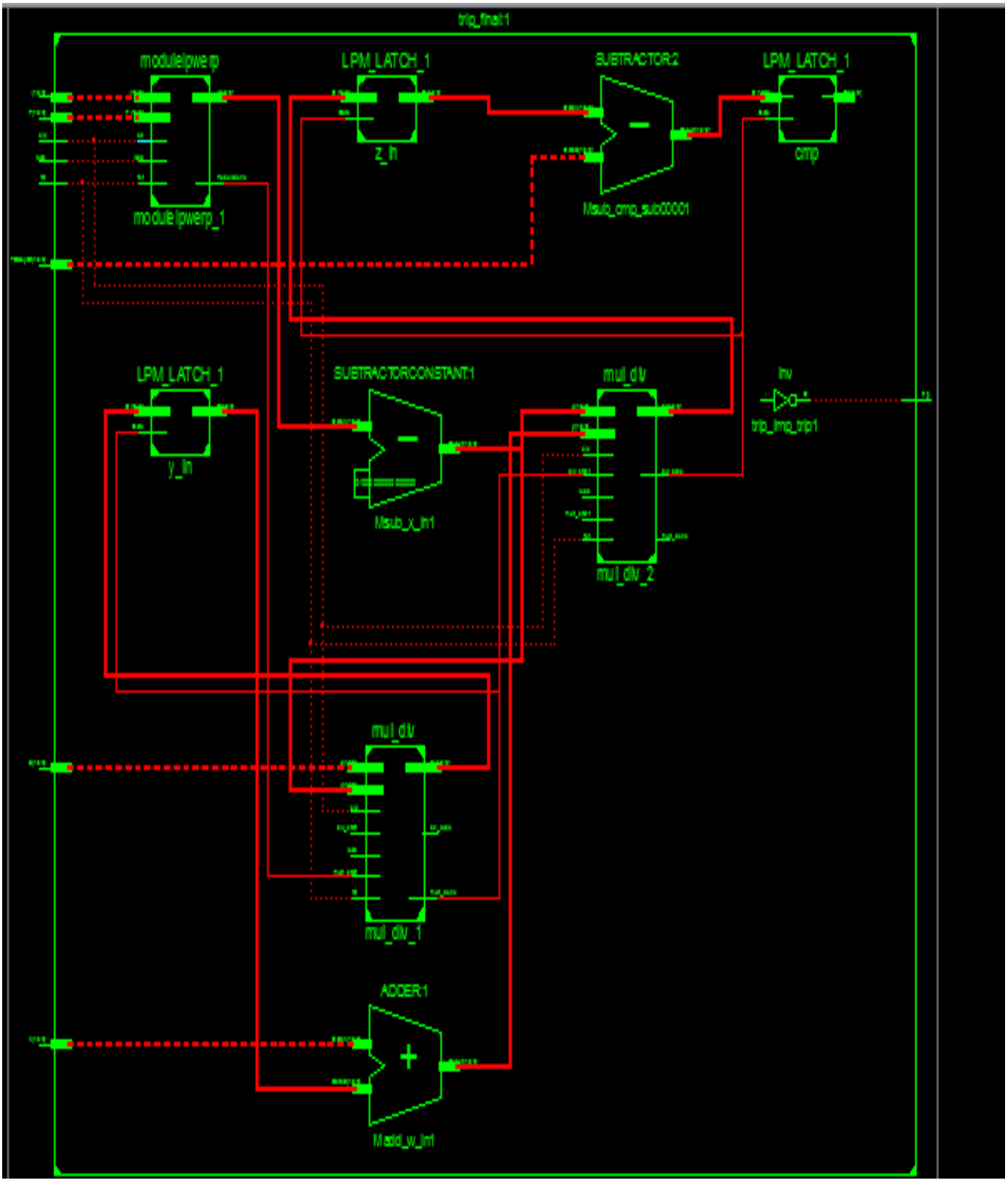
B=0.23

Q=2

Value of TDS/Δt =0.75

Above values are used to calculate trip condition using different Time Dial Setting and Current values.

4.5.2 RTL SCHEMATICS:



4.5.3 SUMMARY:

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	1030	5472	18%
Number of Slice Flip Flops	420	10944	3%
Number of 4 input LUTs	1970	10944	18%
Number of bonded IOBs	89	240	37%
Number of GCLKs	1	32	3%

4.5.4 RESULTS:

I1=0.5029 A, TRIP=0

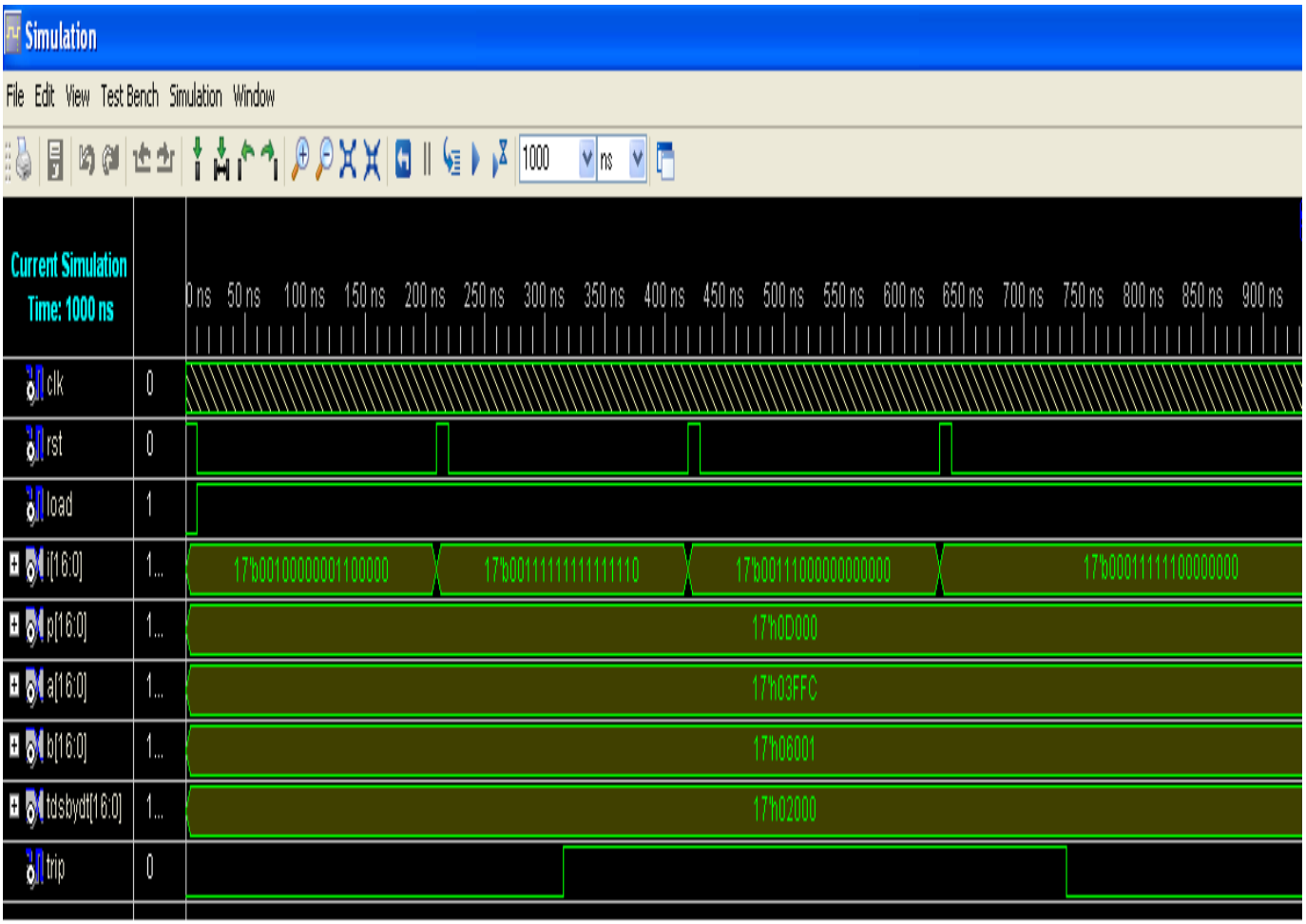
I2=1.9 A, TRIP=1

I3=0.875 A TRIP=1

I4=0.492187 A TRIP=0

At I1=0.5029 A, there is no tripping. When current is increased to I2=1.9 A, relay tripping occurs and continues for I3=0.875 A. Further decreasing current to I4=0.4921875 A deactivates relay.

4.5.5 TEST BENCH:



CHAPTER 5

CONCLUSION AND FUTURE WORK

CORDIC is a powerful tool to implement any complex equation using simple adders and shifters. As in this project, we have implemented various basic functions needed like exponentiation, logarithm, division and also multiplication. These modules were tested and found to be quite accurate. Implementation of overcurrent relay was also tested and results verified. The device utilization summary was found to be very minimal because of efficient use IN CORDIC.

The advantages of using CORDIC for this protective relay are:

- As only shift registers, adders and look-up tables are used, the cost and hardware requirement is very less if we implement CORDIC.
- As we reduce the hardware complexity, the FPGA footprint is considerably reduced
- It is quite simple to design and also to implement
- As we use no multiplication, but only shifting and addition, VLSI implementation is much simpler
- The delay is also quite less
- Hence, if there are no multipliers or there is necessity to economize on the number of gates used, this is a viable option.

However there are some disadvantages such as:

- To get the accurate results, large number of iterations required
- Power consumption can rise in some architecture types

FUTURE SCOPE:

The proper functionality of the relay developed can be tested using FPGA and further analysing the waveforms using ChipScopePro. Further it can be used to develop relays with more time current curves setting for better use. The accuracy for different time dial settings can be improved using more iteration and better time inverse characteristics can be tried upon. Other types of protective relays can also be developed using CORDIC algorithm.

REFERENCES:

[1]. Meher, Pramod, Valls, Javier, Juang, Tso-Bing, Sridharan, K and Maharatna, Koushik (2009) **50 Years of CORDIC Algorithms, Architectures, and Applications**. *IEEE* Transactions on Circuits and Systems - *I*, 56, (9), 1893-1907

[2]. Ray and Andraka, " **A Survey of CORDIC Algorithms for FPGA based Computers**", Andraka Consulting Group, Inc, North Kingstown, RI02852, 2011.

[3]. JACK E. VOLDERt. **The CORDIC Trigonometric Computing Technique.** IRE TRANSACTIONS ON ELECTRONIC COMPUTERS. September 1959.

[4]. Anis BOUDABOUS, Fahmi GHOZZI, M. Wajdi KHARRAT, Nouri MASMOUDI Laboratory of Electronics and Information Technology National Engineers School of Sfax (E.N.I.S.), BP W 3038 SFAX – TUNISIA. **Implementation of Hyperbolic Functions Using CORDIC Algorithm.** 0-7803-8656-6/04 ©2004 IEEE.

[5]. Using CORDIC methods for computation in micro-controllers

[6]. Daniel R. Llamocca-Obregón, Carla P. Agurto-Ríos. Grupo de Procesamiento Digital de Señales e Imágenes - Pontificia Universidad Católica del Perú Av. Universitaria s/n Cuadra 18 - Lima 32, Perú. **A FIXED-POINT IMPLEMENTATION OF THE EXPANDED HYPERBOLIC CORDIC ALGORITHM**

[7] J C Tan, University of Manitoba, Canada; P G McLaren, Florida State University, **USA;** R P Jayasinghe' P L Wilson, Manitoba HVDC Research Centre, Canada. **SOFTWARE MODEL FOR INVERSE TIME OVERCURRENT RELAYS INCORPORATING IEC *AND* IEEE STANDARD CURVES**. Proceedings of the 2002 IEEE Canadian Conference on Electrical & Computer Engineering.

[8]. Jong-wan Seo, School of Information and Communication Engineering. Sungkyunkwan university Suwon, Republic of Korea. Myong-Chul Shin, School of Information and Communication Engineering, Sungkyunkwan university Suwon, Republic of Korea. **A Study on an ASIC Design Technique for Digital Protective Relays.**

[9]. Tarlochan S. Sidhu, *Member*, IEEE, Mohindar S. Sachdev, Fellow, IEEE**,** and Hugh *C.* Wood, Member, IEEE. **A Computer-Aided Design Tool for Developing Digital Controllers and Relays**. IEEE TRANSACTIONS ON INDUSTRY APPLICATIONS, VOL. 28, NO. 6, NOVEMBER / DECEMBER 1992.

[10]. Jong Kang Park, Jong Tae Kim *, Myong-Chul Shin. **A CORDIC-based digital protective relay and its architecture**. School of Information and Communication Eng., Sungkyunkwan Univ., 300 Cheoncheon-dong Jangan-gu, Suwon, Gyeonggi-do 440-746, South Korea.