ES203
Cordic Processor

Overview

CORDIC (**Co**ordinate **R**otation **Di**gital **C**omputer) Algorithm uses straightforward and simple operations like shifting and addition along with look up tables to facilitate and speed up the process of computation. Not only it can be used to compute logarithmic, trigonometric and hyperbolic functions but also eigenvalue estimation, QR factorization, etc. In this project, we plan to implement a few of its diverse applications.

International Journal of Science and Research (IJSR) ISSN (Online): 2319-7064 | Index Copernicus Value (2013): 6.14 | Impact Factor (2015): 6.391

A Review Paper on CORDIC Algorithm and Its Applications for Current Technology

Why did we choose CORDIC?

The motivation behind choosing CORDIC as a subject of study comes from the idea of CORDIC itself. It is fascinating to think how arithmetic subtraction, addition and shift registers can form the basis of large number of functions which are usually calculated by computational intensive Taylor's expansion. CORDIC algorithm has a lot of scope in terms of image processing and waveform generation as it is a fast iterative algorithm which does not require multipliers or dividers for function generation.

Since our course deals with hardware design and coding, we believe it will be interesting to implement this algorithm that can be extended to wide applications like QR factorization, eigen estimation, 3D graphing and much more.

How Does Cordic Work?

$$x_R = x_{in}cos(\theta) - y_{in}sin(\theta)$$

$$y_R = x_{in}sin(\theta) + y_{in}cos(\theta)$$

$$\begin{bmatrix} x_R \\ y_R \end{bmatrix} = cos(\theta)\begin{bmatrix} 1 & -tan(\theta) \\ tan(\theta) & 1 \end{bmatrix}\begin{bmatrix} x_{in} \\ y_{in} \end{bmatrix}$$

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

Main concept of CORDIC

$$x[i+1] = x[i] - \sigma_i 2^{-i} y[i]$$

$$y[i+1] = y[i] + \sigma_i 2^{-i} x[i]$$

NOT the complete algorithm

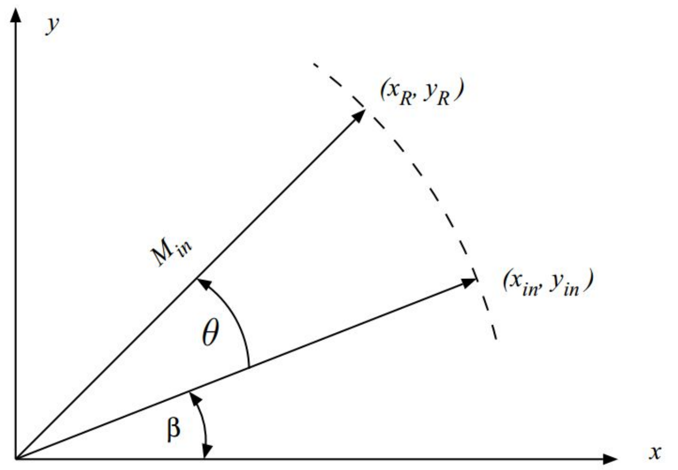


Image Source: https://www.allaboutcircuits.com/technical-articles/an-introduction-to-the-cordic-algorithm

Functions that use Taylor's expansion (hyperbolic, inverse functions, sine, cosine, exponential, logarithmic) can be evaluated using CORDIC in two modes:

1. Rotation Mode: In this mode, the angle is iteratively pushed to *approaching* zero. It is used for sine, cosine, hyperbolic functions, rotating 2D coordinates.

   For sine and cosine and hyperbolic, the y coordinate is set to zero.

2. Vectoring Mode: In this mode, then 'y' coordinate is iteratively pushed to *approaching* zero and the initial angle is set to 0. Any function which involves inverse functions (like arctan, arctanh) can be computed using this mode.

Design and Implementation of CORDIC Processor for Complex DPLL
Amritakar Mandal, B. K. Kaushik, Member, IEEE, Brijesh Kumar, Student Member, IEEE and R. P. Agarwal, Senior Member, IEEE | 978-1-4244-7882-8/11/$26.00 ©2011 IEEE

Advantages:

1. Cost of CORDIC is way less as only adders, subtractors and LUTs are required.
2. Either in absence of multiplier or if there is a need to optimise the overall design, CORDIC is the best algorithm.
3. Number of gates required in hardware implementation, such as on an FPGA, is minimum as hardware complexity is greatly reduced compared to other processors such as DSP multipliers.

Disadvantages:

1. The accuracy depends on the size of LUT and hence the number of iterations.
2. In Microprocessors which can handle multipliers well, taylor series computation is more effective.

International Journal of Advanced Research in Computer and Communication Engineering | Vol. 4, Issue 9, September 2015
Sine-Cosine Computation Using CORDIC Algorithm, Ranjita Naik, Riyazahammad Nadaf

Goals

1. **Finding angle:** To find the angle between a vector and a chosen axis using its coordinates.

2. **Trigonometric functions:** Computing the sine and cosine of the given angle using simple shift-add operations.

3. **Logarithm:** Finding the approximated natural log of a given number.

4. **Coordinate Rotation:** Finding the new coordinates after vector rotation by a given angle.

5. **Evaluation of DFT (using FFT Algorithm):** Using the built modules for DFT which relies on sine and cosine modules.

Week wise plan

1. **Week 1**: implement basic functions in python (sine, cosine, logarithmic functions). This requires use of division and addition functions to make the algorithm clear and easy to implement in verilog.

2. **Week 2**: implement them in verilog. Also, learn about how to use them in various applications.

3. **Week 3:** implement the applications in python (coordinate rotation, determining angle for input coordinates), along with implementation of DFT (based on FFT)

4. **Week 4**: present the final project after implementation of the applications in verilog and burning the code on FPGA or BASYS 3

Week 1

Python Codes

Status:
Implemented Successfully

GitHub Repository and Repl.it Codes

Computing angle of given coordinates:
Github link ○ https://repl.it/@hrushti/ES203-Cordic-Finding-Angle

Computing new coordinates:
Github link ○ https://repl.it/@hrushti/ES203-Cordic-Rotation

Computing Sine, Cosine:
Github link ○ https://repl.it/@hrushti/ES203-Cordic-Sine-Cosine

Computing Logarithm using atan hyperbolic:
Github link ○ https://repl.it/@hrushti/ES203-Cordic-Logarithm

Snippets of some results:

```
= RESTART: /Users/nipunmahajan/Desktop/Academ
 given coordinates.py
enter the x coordinate: 3.2567
enter the y coordinate: 648.4

Angle in radians:  1.565774234723597
Expected angle is:  1.5657736978396546
>>>
```

```
= RESTART: /Users/nipunmahajan/Desktop/Academics/ES 203/Project/Project Submissio
enter the x coordinate: 2
enter the y coordinate: 10
Enter the angle in degrees: 30
The expected values of X and Y are:  -3.2679491924311215 9.66025403784387
The CORDIC rotated values of X, Y are:  (-3.267949191916412 8, 9.6602540380185 1)
>>>
```

```
Enter the angle in degrees: 78.45
The CORDIC values of sine and cosine are:  (0.9797503501923033, 0.2002230039182804 3)
The expected value of sine and cosine is:  0.9797503501713428 0.200223004020844 6
>>>
================== RESTART: /Users/nipunmahajan/Desktop/Academics/ES 203/Project/sin_cos.py
Enter the angle in degrees: 0.8783475
The CORDIC values of sine and cosine are:  (0.01532945545429695, 0.9998824969942597)
The expected value of sine and cosine is:  0.01532945540414168 0.99988249699502 4
>>>
```

```
Enter value whose log has to be calculated 764.238237
The CORDIC value is 6.6388809054
The actual value of logarithm is:  6.63887956909213 9
>>>
======================= RESTART: /Users/nipunmahajan/Desktop/Ac
Enter value whose log has to be calculated -128
The actual value of logarithm is: INVALID
>>>
======================= RESTART: /Users/nipunmahajan/Desktop/Ac
Enter value whose log has to be calculated 0.08672
The CORDIC value is -2.4450703706
The actual value of logarithm is:  -2.4450707412908 01
```

Week 2

Verilog Codes

Status:
Implemented Successfully

Eda Playground

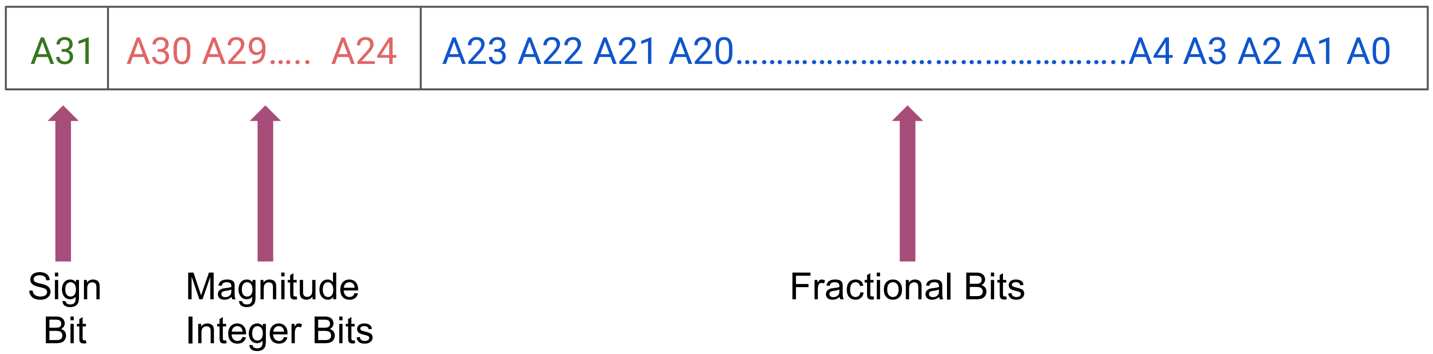
Computing angle of given coordinates:
https://edaplayground.com/x/WX_m

Computing new coordinates:
https://edaplayground.com/x/tU6w

Computing Sine, Cosine:
https://www.edaplayground.com/x/tTvQ

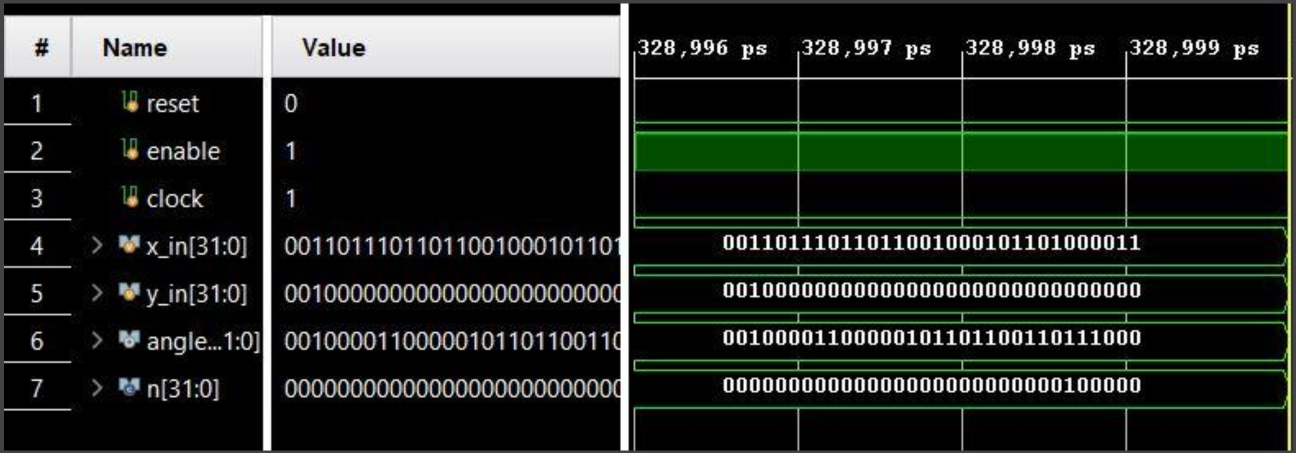
Computing Logarithm using atan hyperbolic:
https://www.edaplayground.com/x/G87Y

Fixed Point Representation: [32 24]



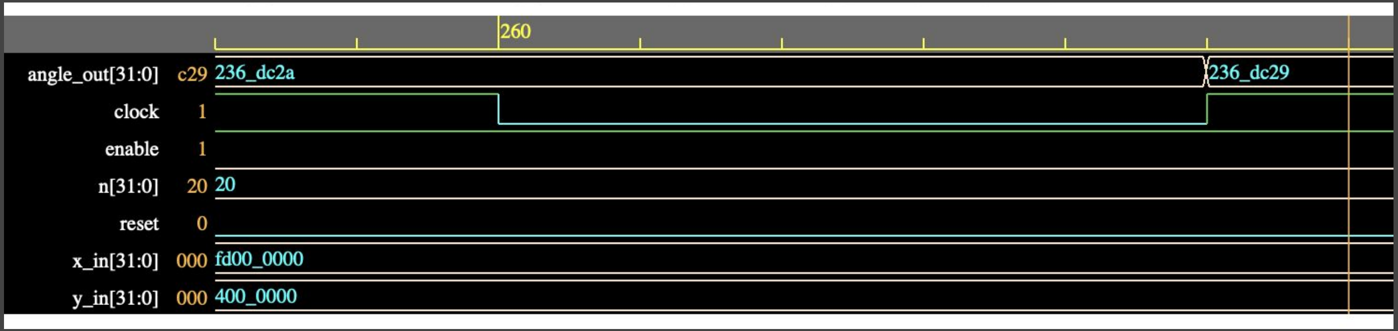
Range of I/O Values: **-128** to **127.99999994**

Snippets of some results: (Angular Outputs in Radians)



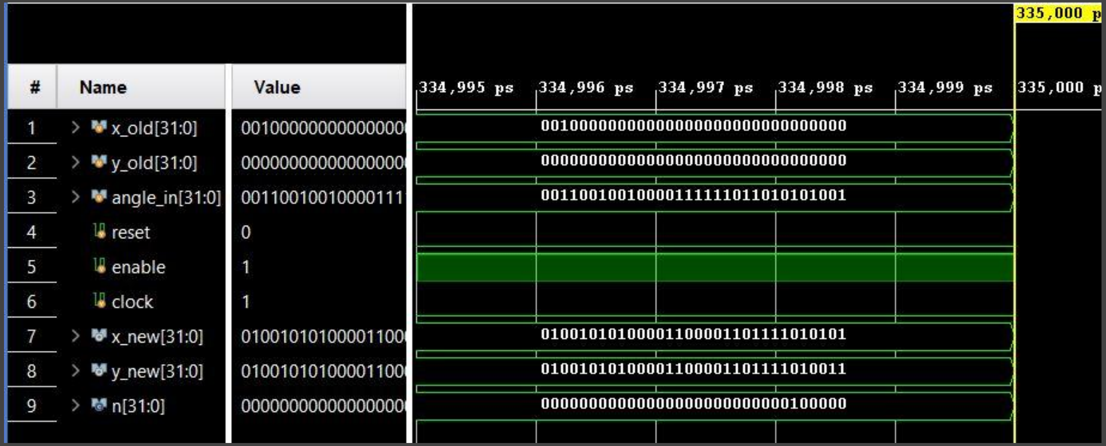
Finding the angle of a given vector

Coordinates: (√3/2, 1/2)
Output angle: 30°



Coordinates:
(-3, 4)

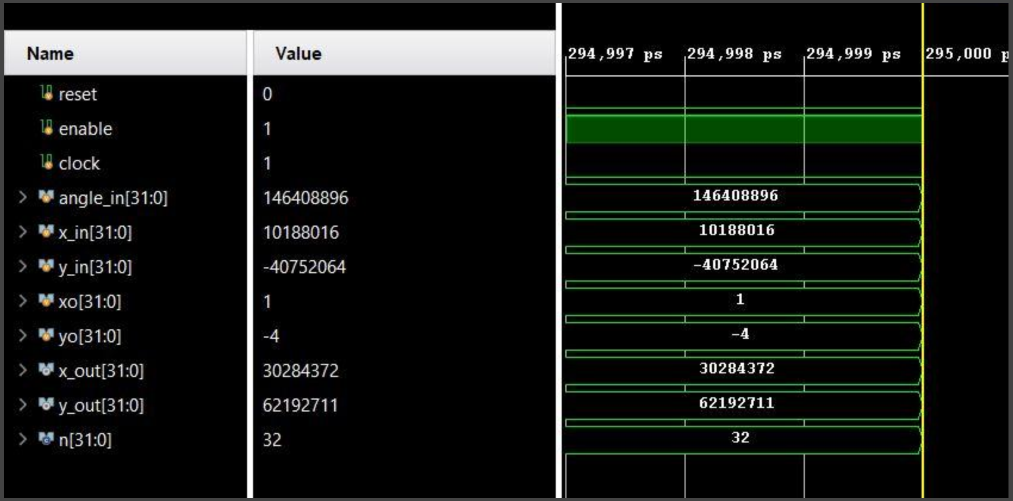
Output angle:
127°
(0236_dc29)



Rotating given coordinates by a given angle

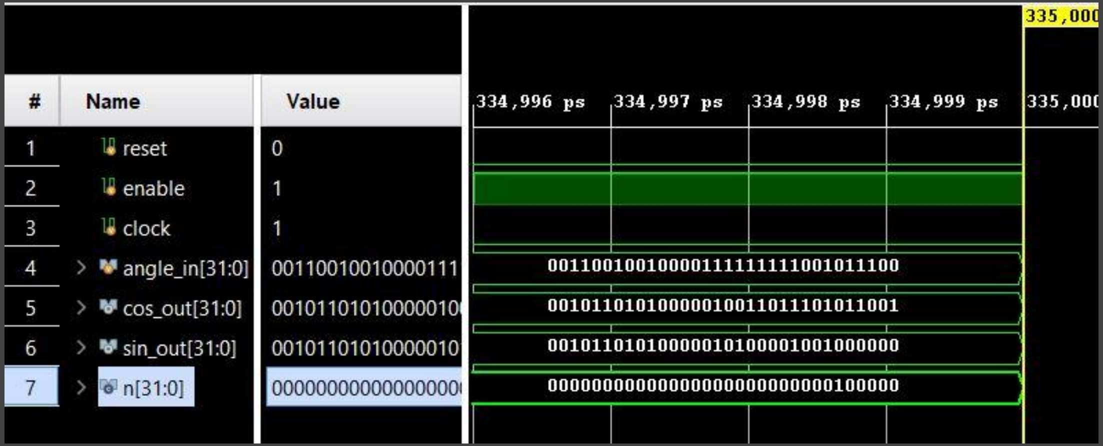
Coordinates: (1, 0)
Angle: $\pi/4$

X_out: 0.707
Y_out: 0.707



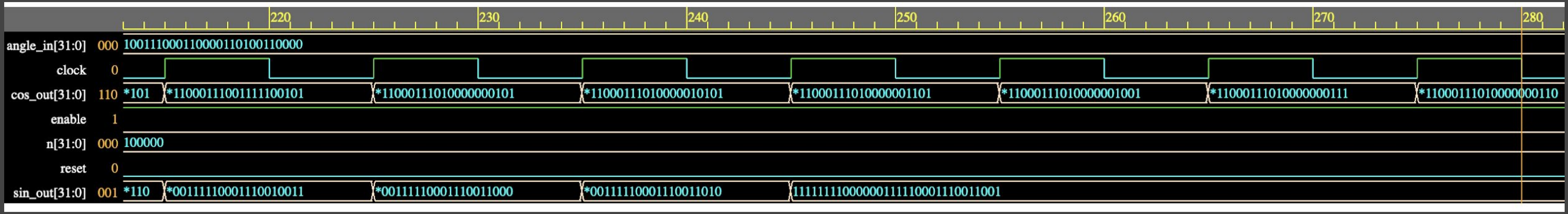
Coordinates: (1,- 4)
Angle: 500°

X_out: 1.805105
Y_out: 3.706965



Finding sin and cos of a given angle

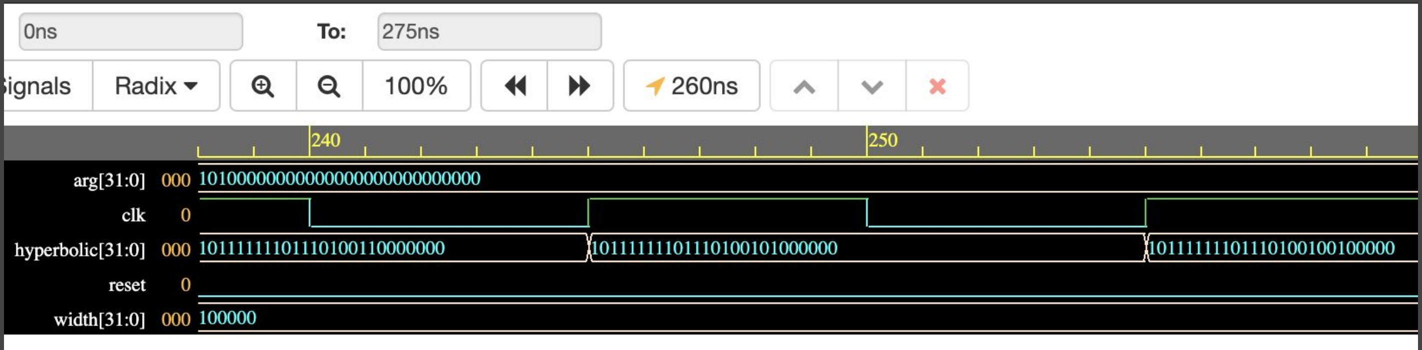
Angle: $\pi/4$



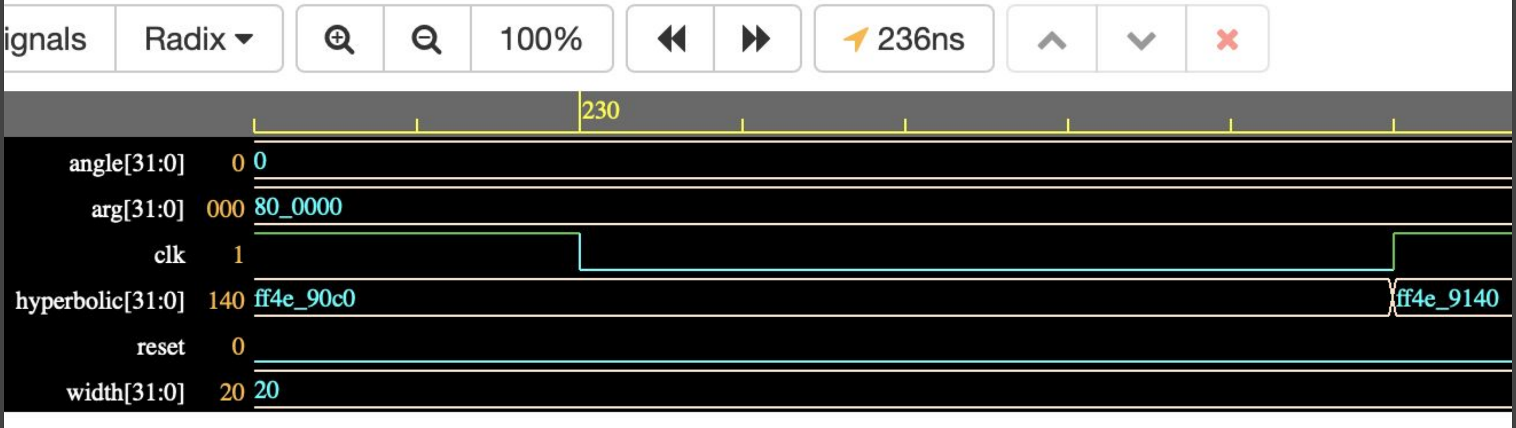
Input angle: 280°

Sin: -0.98480
Cos: 0.17364

Finding the logarithm of given number



Input: 20
Output: 2.995743



Input: 0.5
Output: -0.6931498

Week 3
DFT & FFT in python

Status:
Implemented Successfully

DFT

Repl link - https://repl.it/@hrushti/ES203-DFT

Snippets of results:

```
Enter the degree of the polynomial: 3
the polynomial has 4 terms
enter the term: -10
enter the term: 23
enter the term: 2
enter the term: 3
CORDIC real values         :  18.000000 -11.999956 -34.000000 -12.000050
expected real components :  18.000000 -12.000000 -34.000000 -12.000000
CORDIC imag values         :  0.00004 -20.00003 -0.00008 19.99997
expected imag components :  0.00000 -20.00000 0.00000 20.00000
>
```

https://cp-algorithms.com/algebra/fft.html#toc-tgt-4

```
Enter the degree of the polynomial: 2
the polynomial has 3 terms
enter the term: -8
enter the term: 4
enter the term: 6
CORDIC real values        :  2.000000 -13.999991 -6.000000 -14.000010
expected real components :  2.000000 -14.000000 -6.000000 -14.000000
CORDIC imag values        :  0.00000 -4.00003 -0.00001 3.99997
expected imag components :  0.00000 -4.00000 0.00000 4.00000
>
```

```
Enter the degree of the polynomial: 4
the polynomial has 5 terms
enter the term: 3
enter the term: 12
enter the term: 4
enter the term: 8
enter the term: 0
CORDIC real values        :  27.000000 5.828425 -0.999993 0.171552 -13.000000 0.171593 -1.000012 5.828429
expected real components :  27.000000 5.828427 -1.000000 0.171573 -13.000000 0.171573 -1.000000 5.828427
CORDIC imag values        :  0.00006 -18.14213 -4.00000 -10.14213 -0.00003 10.14214 4.00000 18.14214
expected imag components :  0.00000 -18.14214 -4.00000 -10.14214 0.00000 10.14214 4.00000 18.14214
```

FFT

Repl link - https://repl.it/@hrushti/ES203-FFT

Snippets of results:

```
Enter the degree of the polynomial: 3
the polynomial has 4 terms
enter the term: -10
enter the term: 23
enter the term: 2
enter the term: 3
CORDIC real values   :  18.000000 -11.999962 -34.000000 -12.000038
expected real components :  18.000000 -12.000000 -34.000000 -12.000000
CORDIC imag values   :  0.00007 -20.00000 -0.00006 20.00000
expected imag components :  0.00000 -20.00000 0.00000 20.00000
```

https://cp-algorithms.com/algebra/fft.html#toc-tgt-4

```
Enter the degree of the polynomial: 5
the polynomial has 6 terms
enter the term: 39
enter the term: 0
enter the term: 0
enter the term: 0
enter the term: 1
enter the term: 1
CORDIC real values   :  41.000000 37.292892 40.000004 38.707106 39.000000 38.707108 39.999996 37.292894
expected real components :  41.000000 37.292893 40.000000 38.707107 39.000000 38.707107 40.000000 37.292893
CORDIC imag values   :  0.00001 0.70710 -1.00000 0.70711 -0.00000 -0.70711 1.00000 -0.70711
expected imag components :  0.00000 0.70711 -1.00000 0.70711 0.00000 -0.70711 1.00000 -0.70711
>>> |

================= RESTART: /Users/nipunmahajan/Desktop/fft.py =================
Enter the degree of the polynomial: 7
the polynomial has 8 terms
enter the term: 2
enter the term: 4
enter the term: 5
enter the term: 0
enter the term: 7
enter the term: 34
enter the term: 2
enter the term: 3
CORDIC real values   :  57.000000 -24.091925 2.000140 14.091837 -25.000000 14.091929 1.999860 -24.091841
expected real components :  57.000000 -24.091883 2.000000 14.091883 -25.000000 14.091883 2.000000 -24.091883
CORDIC imag values   :  0.00022 20.33448 -35.00000 26.33454 -0.00014 -26.33451 35.00000 -20.33457
expected imag components :  0.00000 20.33452 -35.00000 26.33452 0.00000 -26.33452 35.00000 -20.33452
```

Week 4
Verilog Codes

Status:
Implemented Successfully

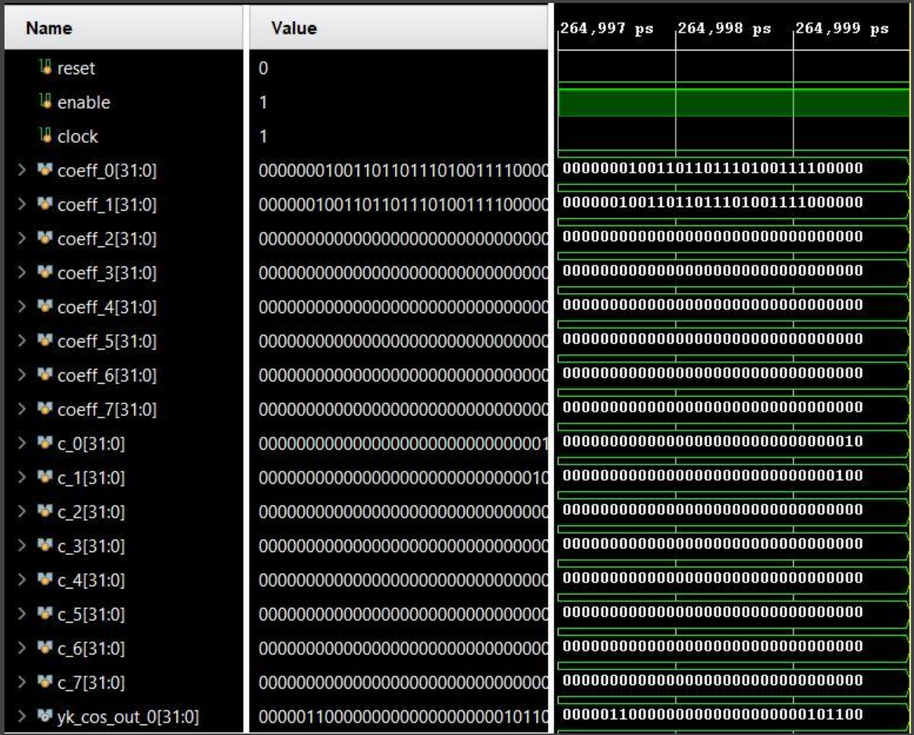
Eda Playground

Master Cordic:
https://www.edaplayground.com/x/D6KR

DFT:
https://www.edaplayground.com/x/kjSW

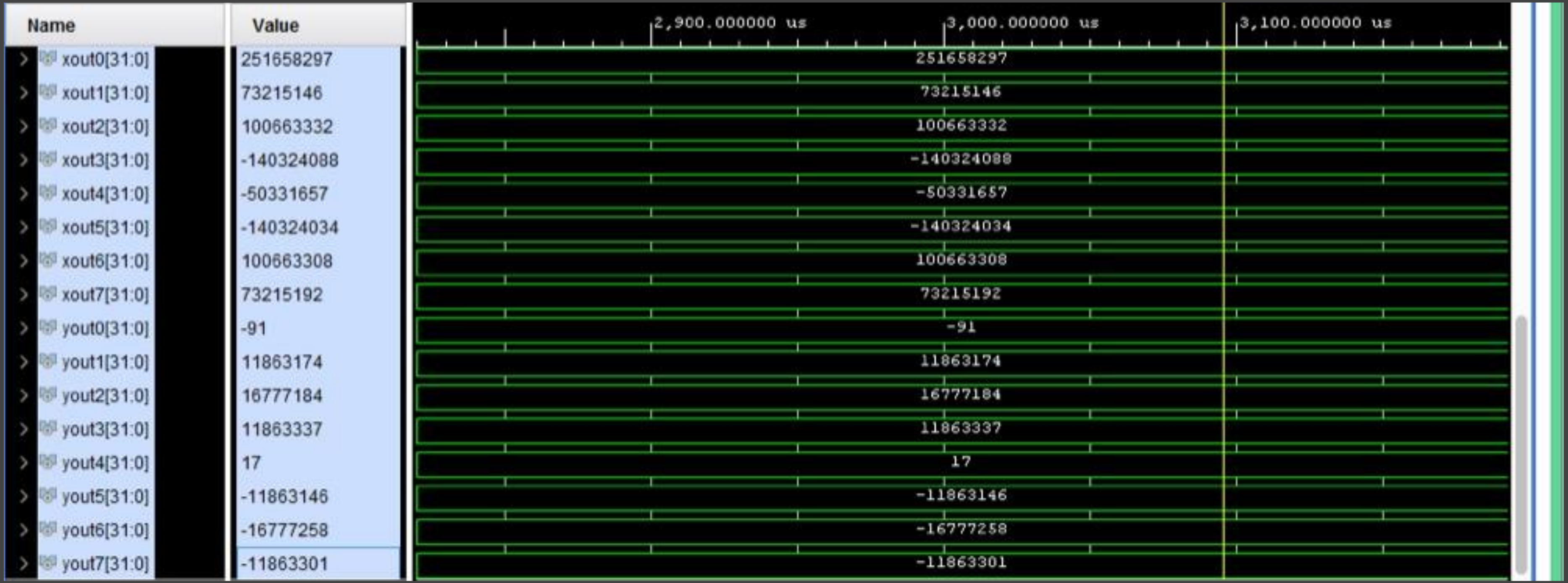
FFT:
https://www.edaplayground.com/x/84TY

Snippets of DFT results:



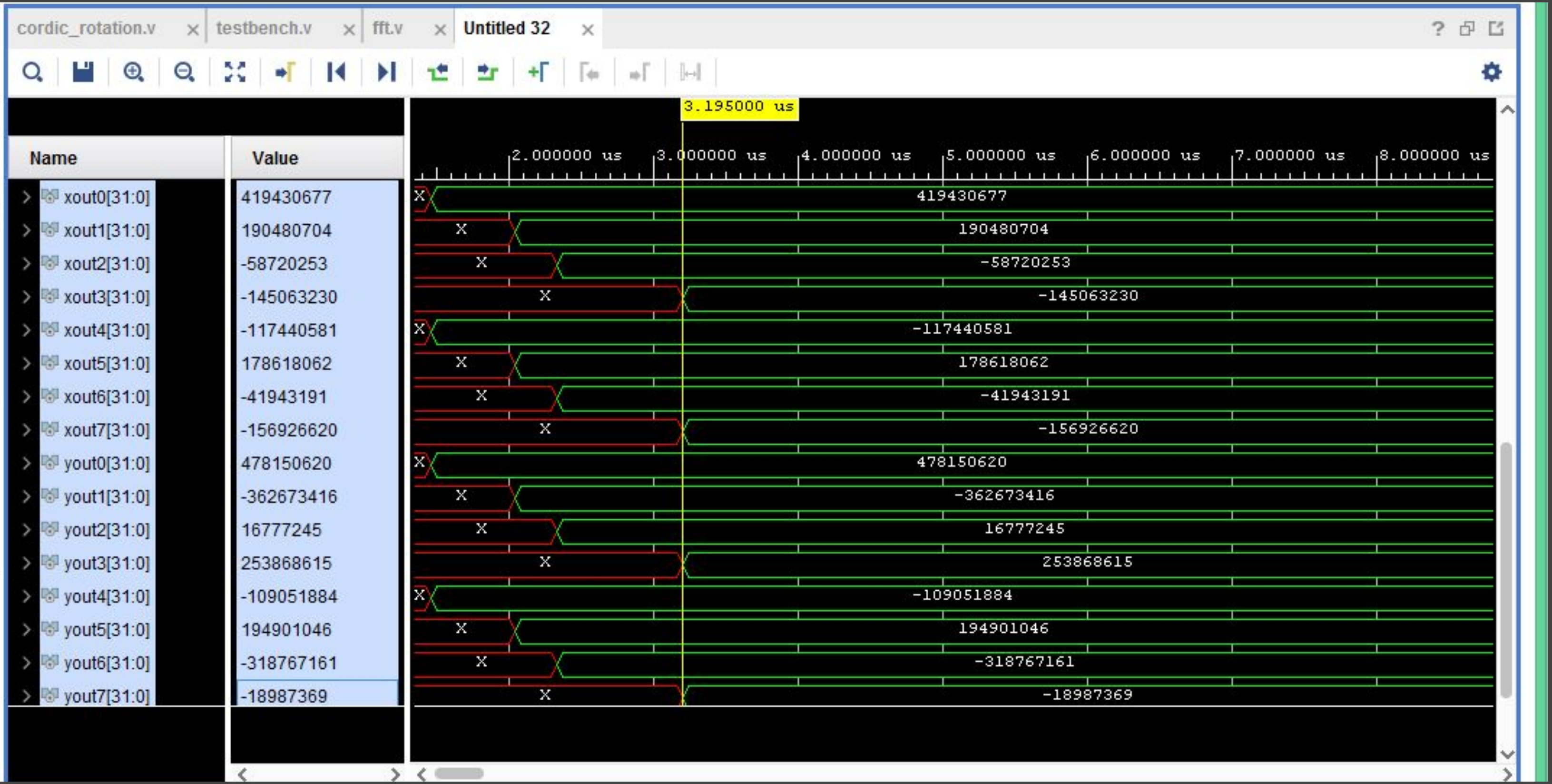
Snippet of FFT results:

Average Error : 1.107126984050883e-06



Inputs:

x = {2, 4, 0, 0, 4, 0, 0, 5}   y = {0, 0, 0, 0, 0, 0, 0, 0}



A0 =2+j

A1 =3

A2 =6+j10

A3 =8+j9

A4 =1

A5 =8.5j

A6 =0

A7 =5

Cordic FFT Real values:

25.00001651 11.353534698 -3.49999982 -8.646244229 -7.000000411 10.6464661359 -2.500009000 -9.3535554409

Cordic FFT Imaginary values:

28.499997854 -21.617020130 1.0000017285 15.13174861669 -6.49999888 11.617007613 -19.000003397 -1.131735384464

Python FFT Result (truncated to 5 decimal places):

```
================= RESTART: /Users/nipunmahajan/Desktop/trial.py ================
expected real components :  25.00000 11.35355 -3.50000 -8.64645 -7.00000 10.64645 -2.50000 -9.35355
expected imag components :  28.50000 -21.61701 1.00000 15.13173 -6.50000 11.61701 -19.00000 -1.13173
>>>
```

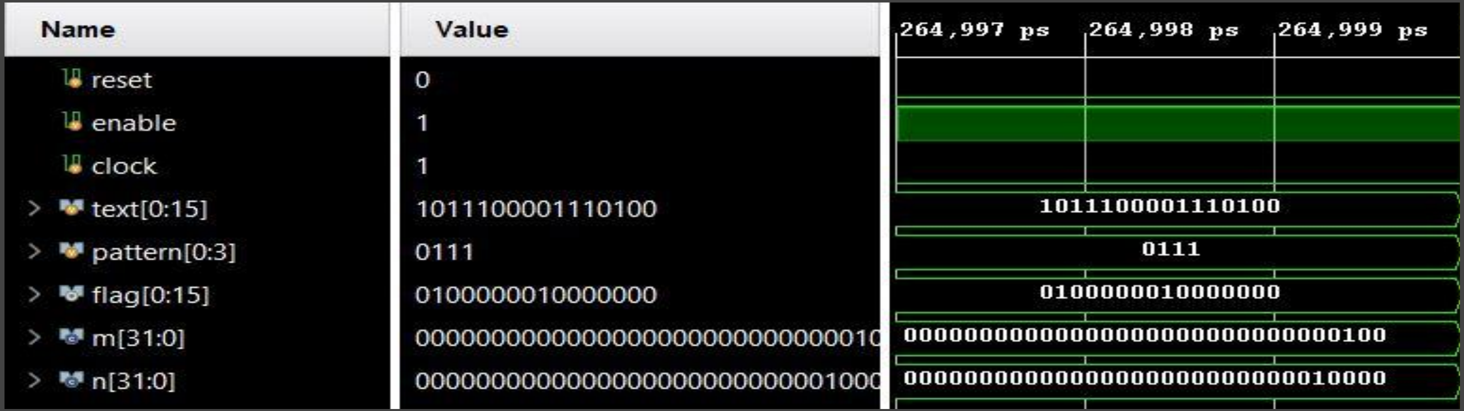
Buffer Time
String Matching

Status:
Implemented Successfully

String Matching

EDA Playground : https://edaplayground.com/x/Ltj_

Snippet of the result:



References:

Logarithm using CORDIC:

1. *Llamocca, Daniel & Agurto, Carla. (2006). A fixed-point implementation of the natural logarithm based on a expanded hyperbolic CORDIC algorithm.*
2. *Llamocca, Daniel & Agurto, Carla. (2007). A fixed-point implementation of the expanded hyperbolic CORDIC algorithm. Latin American Applied Research. 37. 83-91.*
3. *Coordinate rotation digital computer algorithm (CORDIC) to compute trigonometric and hyperbolic functions By Andrea Vitali / DT0085 Rev 1*

Sine, cosine using CORDIC:

1. *(IJITR) INTERNATIONAL JOURNAL OF INNOVATIVE TECHNOLOGY AND RESEARCH Volume No.2, Issue No. 2, February – March 2014, 891 – 895 / Calculation of Sine and Cosine of an Angle using the CORDIC Algorithm*
2. *Bhattarai, Bibek. (2013). FPGA Implementation of CORDIC Processor. 10.13140/RG.2.1.4432.1364.*

Computing tan inverse in CORDIC:

1. *Coordinate rotation digital computer algorithm (CORDIC) to compute trigonometric and hyperbolic functions By Andrea Vitali / DT0085 Rev 1*

Members

Hrushti Naik 19110088

Nipun Mahajan 19110127

Sakshi Jagtap 19110133

Shrreya Singh 19110136