CORDIC-based DDFS Architecture

Lecture 12

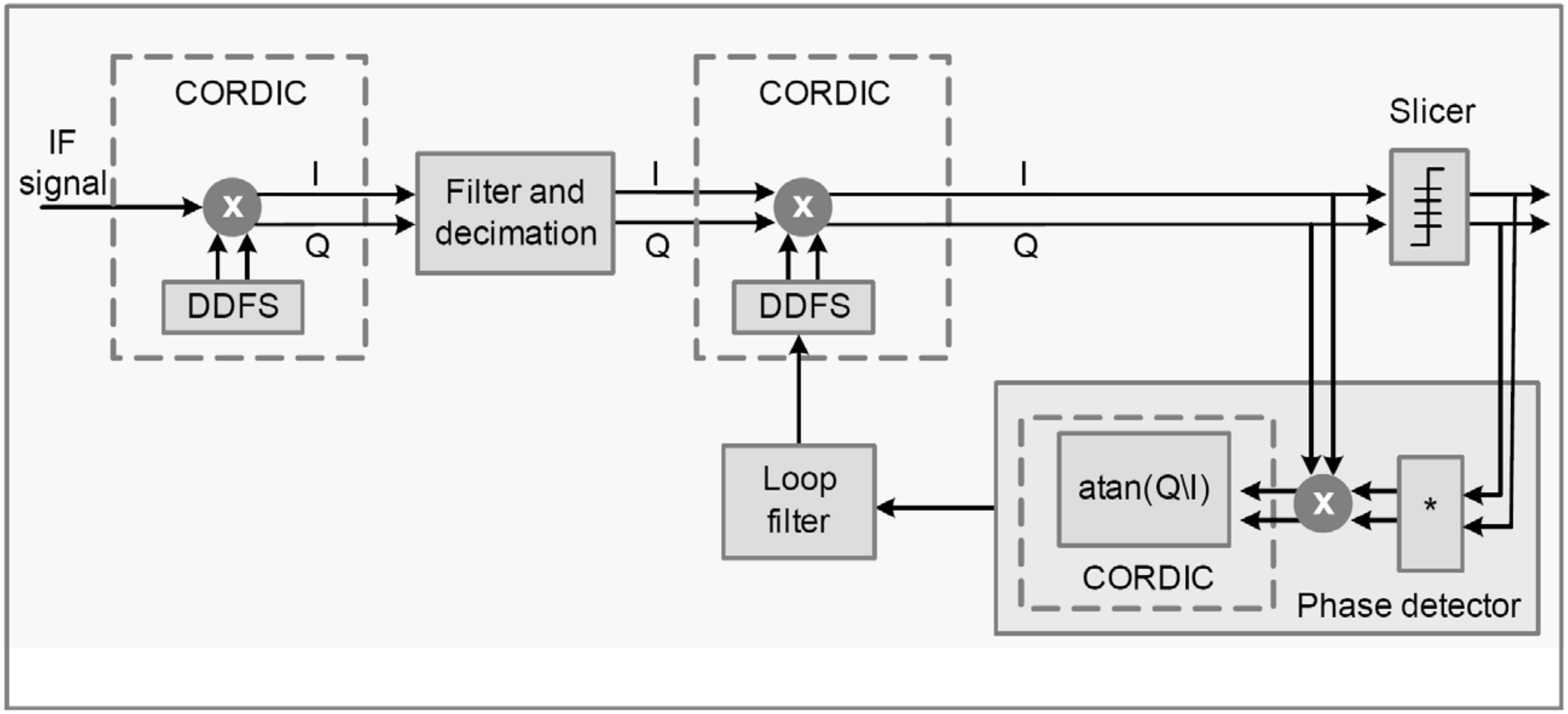
Dr. Shoab A. Khan

Direct Digital Frequency Synthesis (DDFS)

- Direct Digital Frequency Synthesis (DDFS) is used to produce sinusoid signals
    - High frequency resolution
    - Fast changes in frequency and phase
    - High spectral purity

DDFS

- A DDFS is an integral component of high performance communication systems

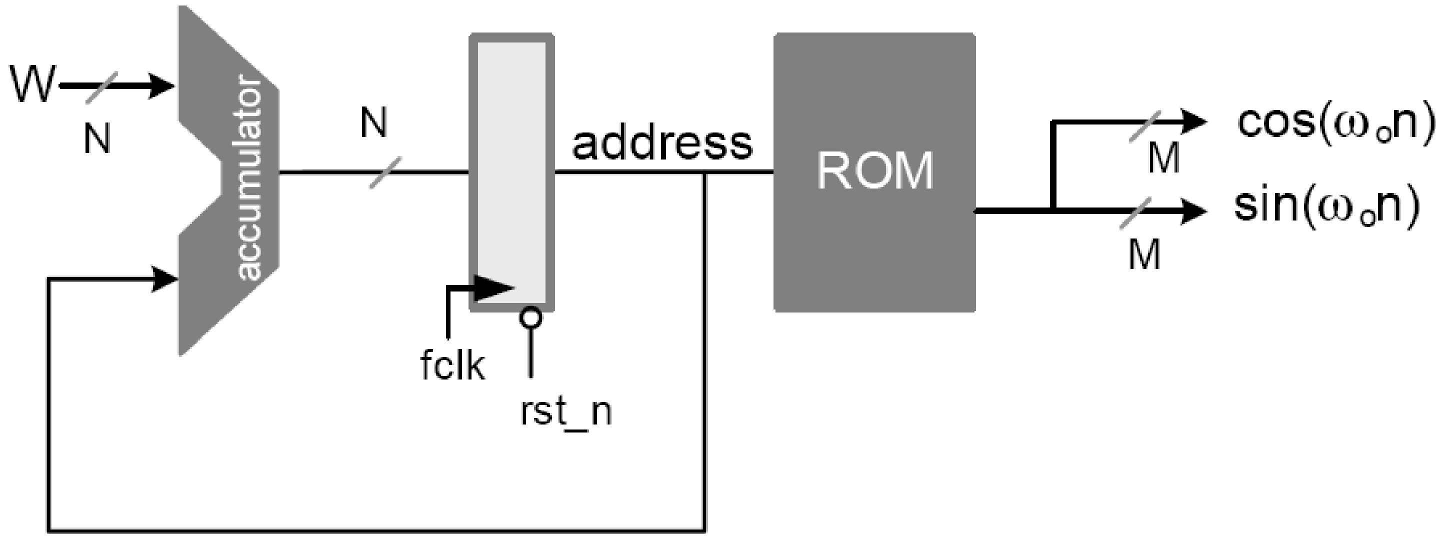


DDFS

- ADDFS is also critical in speed frequency and phase modulation systems
  - GMSK

$$s(t) = \sqrt{\frac{2E_b}{T}} \exp\left[j\pi \sum_{n \quad 0}^{k} \beta_n \theta(t-nt) \right]$$

Design of DDFS



2^N values of sin and cosine are stored in the ROM

DDFS

- The input to the accumulator is the frequency control word, *W*
- The output freq f_o depends on
  - W
  - f_{clk} clock freq

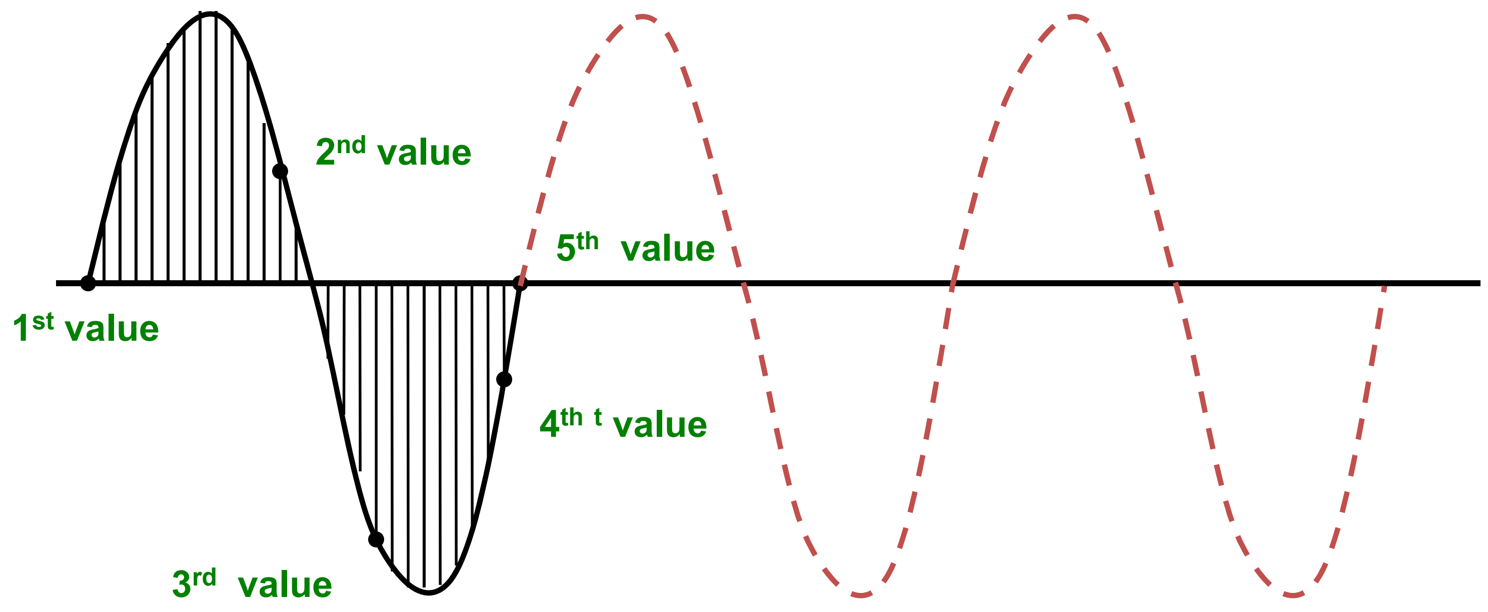
$$f_0 = \frac{f_{clk} W}{2^N}$$

- The phase accumulator produces a digital ramp out
  - acc_reg = acc_reg + W
- The ROM stores corresponding amplitude of sine and consine

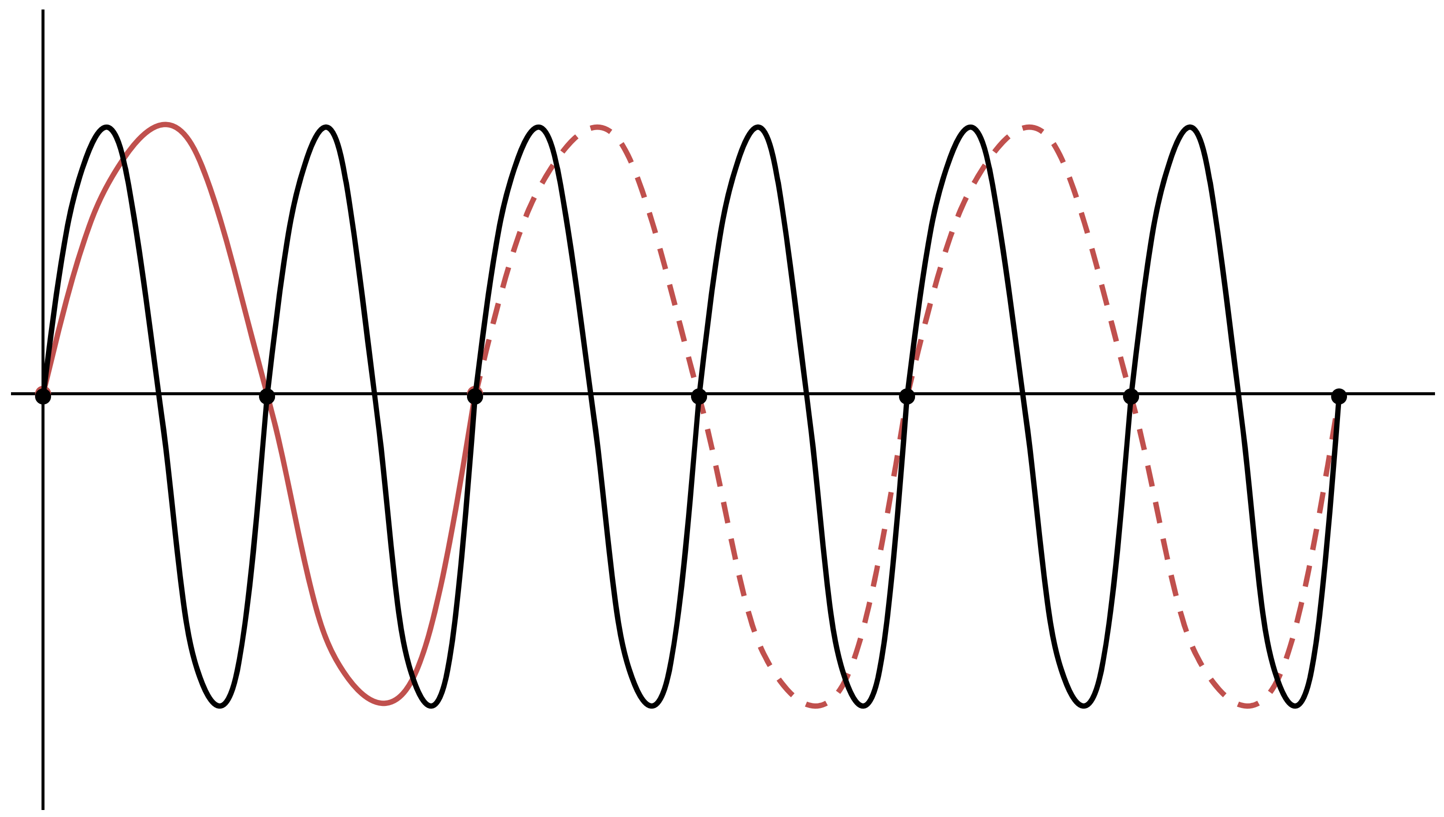
DDFS Accumulator: Verilog Code

```
always @(posedge clk or negedge rst_n)
begin
    if(!rst_n)              // all registers equal to 0 at reset
    begin
        acc_out <= 0;
        w_reg <= 0;
    end
    else if(load)
        w_reg <= w;     //load the input control word at load
    else
        acc_out <= acc_out + w_reg;
end
```

Generation of Sin and Cos



Generation of Sin and Cos

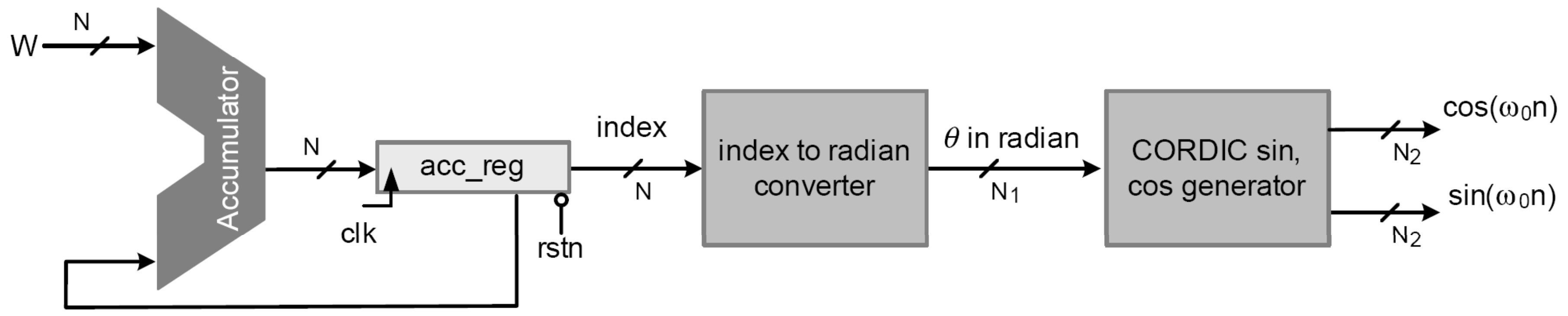


This waveform can be generated by giving an increment of 2

Generation of Sin and Cos

- In embedded system, a ROM can’t be afforded
- Algorithms are used
  - CORDIC

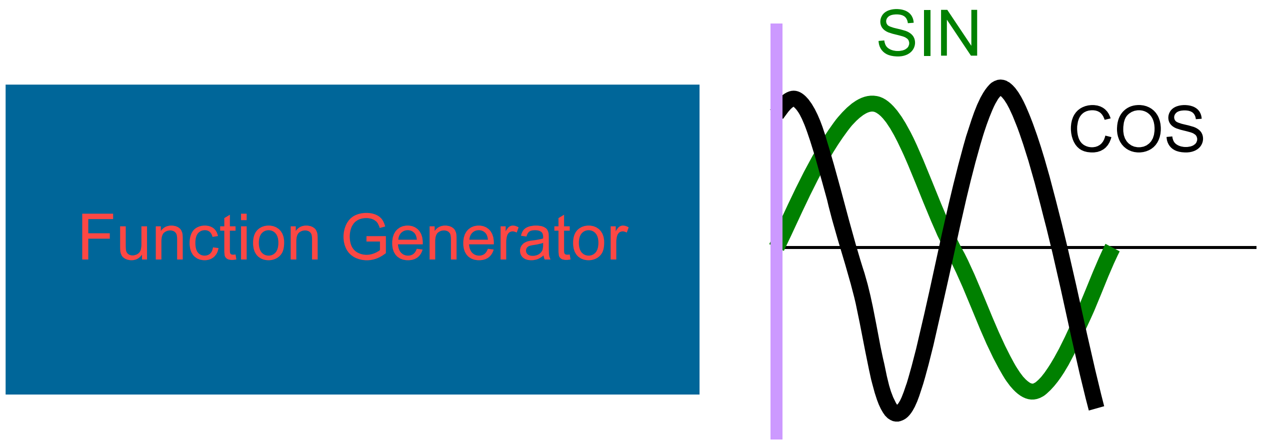
Generation of Sin and Cos



2^N index = 2π

1 index = $2\pi / 2^N$

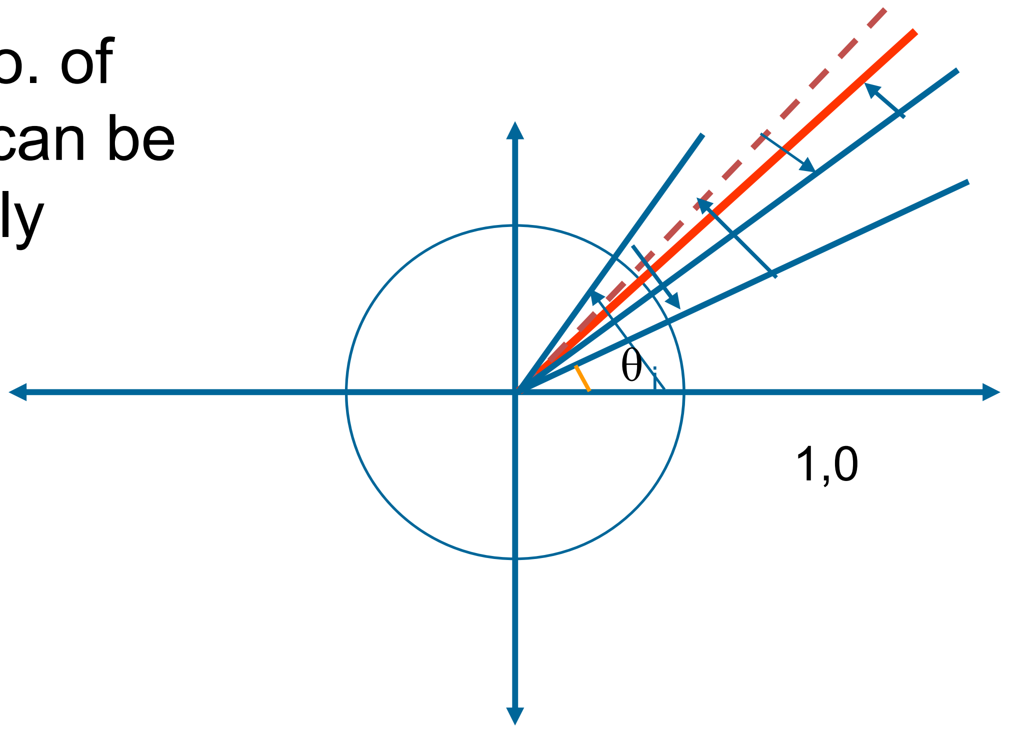
CORDIC as Function Generator



- Generates sin and cos digitally at the same time
- Performs Conversion from Cartesian to Polar Co-ordinates
- Acts as a DDFS
- Can also perform function like division and multiplication

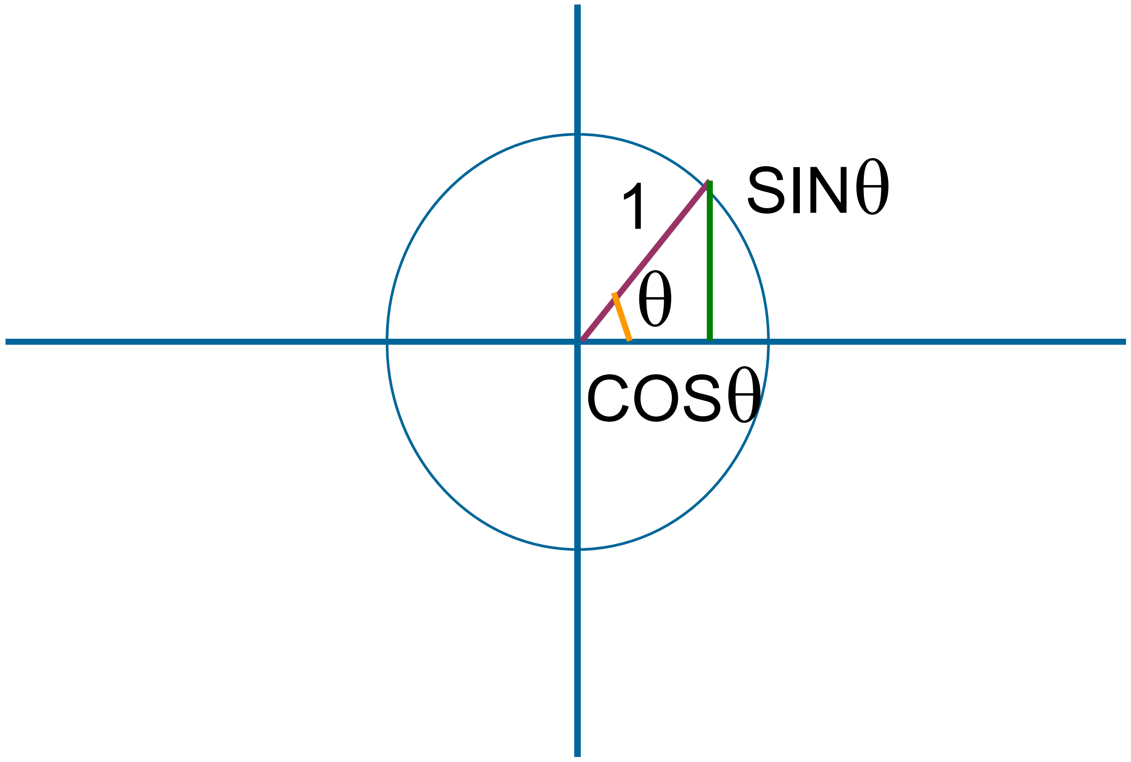
Basic Concept

- The cos and sin of an angle are evaluated by giving known recursive rotations
- Depending upon the No. of iterations, sin and cos can be generated very precisely



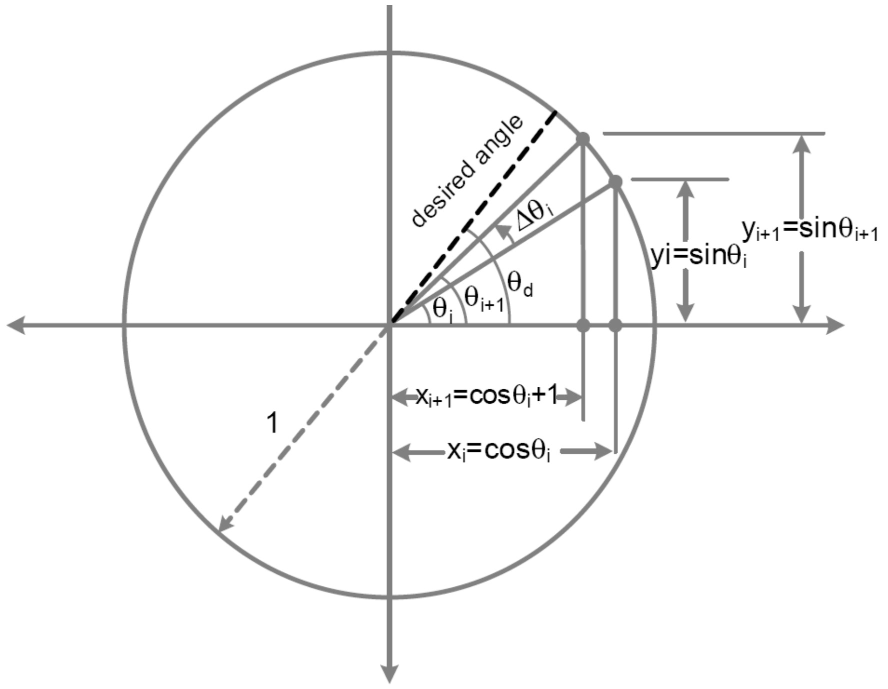
CorDiC Algorithm

- Basic idea
  - Rotate (1,0) by θ degree
    to get (x,y): x=cos θ y=sin θ



Formulation

$$\theta = \sum_{i\ \ 0}^{N\ \ 1} \sigma_i \Delta\theta_i \ \text{ for } \sigma_i = \begin{cases} 1 \text{ for positive rotation} \\ -1 \text{ for negative rotation} \end{cases}$$



$$\cos\theta_{i+1} = \cos(\theta_i + \sigma_i \Delta\theta_i) = \cos\theta_i \cos\Delta\theta_i - \sigma_i \sin\theta_i \sin\Delta\theta_i$$

$$\sin\theta_{i+1} = \sin(\theta_i + \sigma_i \Delta\theta_i) = \sin\theta_i \cos\Delta\theta_i - \sigma_i \cos\theta_i \sin\Delta\theta_i$$

Algorithm

$\cos\theta_{i+1} = \cos(\theta_i + \delta_i \Delta\theta_i)$

$$\delta_i \begin{cases} \text{+ve for +ve rotation} \\ \text{-ve for -ve rotation} \end{cases}$$

$\cos\theta_{i+1} = \cos\theta_i \cos\Delta\theta_i - \delta_i \sin\theta_i \sin\Delta\theta_i$

$\cos\theta_{i+1} = x_{i+1}$

$\cos\theta_i = x_i$

$\sin\theta_i = y_i$

For Cosine

General Formula

$x_{i+1} = x_i \cos \Delta\theta_i - \delta_i\, y_i \sin \Delta\theta_i \xrightarrow{\hspace{2cm}}$ **Eq 1**

For positive value

$$= x_i \cos \Delta\theta_i - y_i \sin \Delta\theta_i$$

For negative value

$$= x_i \cos \Delta\theta_i + y_i \sin \Delta\theta_i$$

For Sine

General Formula:

$$\sin\theta_{i+1} = \sin(\theta_i + \delta_i \Delta\theta_i)$$

For positive value

$$\sin\theta_i \cos\Delta\theta_i + \cos\theta_i \sin\Delta\theta_i$$

For negative value

$$\sin\theta_i \cos\Delta\theta_i - \cos\theta_i \sin\Delta\theta_i$$

$$x_{i+1} = x_i \cos \Delta\theta_i - \sigma_i y_i \sin \Delta\theta_i$$

$$y_{i+1} = \sigma_i x_i \sin \Delta\theta_i + y_i \cos \Delta\theta_i$$

$$\begin{pmatrix} x_{i+1} \\ y_{i+1} \end{pmatrix} = \begin{pmatrix} \cos \Delta\theta_i & -\delta_i \sin \Delta\theta_i \\ \delta_i \sin \Delta\theta_i & \cos \Delta\theta_i \end{pmatrix} \begin{pmatrix} x_i \\ y_i \end{pmatrix}$$

Rotation Matrix representation

Taking $\cos\Delta\theta_i$ common in the Rotation Matrix

$$= \cos\Delta\theta_i \begin{pmatrix} 1 & -\delta_i \tan\Delta\theta_i \\ \delta_i \tan\Delta\theta_i & 1 \end{pmatrix} \begin{pmatrix} x_i \\ y_i \end{pmatrix}$$

$\cos\theta = 1/\sqrt{1 + \tan^2\theta}$ [Trigonometric identity]

Basic Assumption of CORDIC

$$= 1/\sqrt{1+\tan^2\Delta\theta_i}\begin{pmatrix} 1 & -\delta_i \tan\Delta\theta_i \\ \delta_i \tan\Delta\theta_i & 1 \end{pmatrix}\begin{pmatrix} x_i \\ y_i \end{pmatrix}$$

$$\boxed{\tan\Delta\theta_i = 2^{-i}}$$

[Basic assumption of CorDiC algorithm]

$$\Delta\theta_i = \tan^{-1}(2^{-i})$$

Where i = 0,1,2,3,4,…. N-1

- So

$\Delta\theta_0 = \tan^{-1}(2^{0})$

$\Delta\theta_1 = \tan^{-1}(2^{-1})$

$\Delta\theta_2 = \tan^{-1}(2^{-2})$

$\Delta\theta_3 = \tan^{-1}(2^{-3})$

Hence considering $\tan\Delta\theta_i = 2^{-i}$ makes matrix multiplication easier and simpler

Computing $\Delta\theta_i$

Pre-computation of tan($\Delta\theta_i$)

- Find $\Delta\theta_i$ Such that tan($\Delta\theta_i$ = 2^{-i}: (or $\Delta\theta_i$=$\tan^{-1}(2^{-i})$)

i		tan($\Delta\theta_i$)	
0	45.0^0	1	$=2^{-0}$
1	26.6^0	0.5	$=2^{-1}$
2	14.0^0	0.25	$=2^{-2}$
3	7.0^0	0.125	$=2^{-3}$
4	3.6^0	0.0625	$=2^{-4}$
5	1.8^0	0.03125	$=2^{-5}$
6	0.9^0	0.015625	$=2^{-6}$
7	0.4^0	0.0078125	$=2^{-7}$
8	0.2^0	0.00390625	$=2^{-8}$
9	0.1^0	0.001953125	$=2^{-9}$

- **Note: decreasing** $\Delta\theta_i$
  - **Possible to write any angle** = $\theta \quad \pm\Delta\theta_0 \pm\Delta\theta_1 \pm\ldots\pm\Delta\theta_9$
    as long as -99.7$^0\leq$ **(which covers** -90..90**)**
  - **Convergence possible** θ

Concept

- The rotation by an angle θ is implemented as N micro-rotations during of step $\Delta\theta_i$ angles
- The angle θ can be represented to a certain accuracy by a set of N step angles $\Delta\theta_i$ for $i=0,1,2,\ldots,N-1$
- Specifying a direction of rotation, the sum of the step angles approximates the desired angle

$$\sum_{i=0,1,\ldots,N-1} \delta_i \, \Delta\theta_i$$

The Concept

- The sign of the difference between the desired angle and the partial sum of step angles determines the direction of rotation of the next micro angle rotation
  - Set θ_d to θ_0, and then subtracting or adding each micro rotation from the current angle depending on δ_i .

$$\theta_0 = \theta_d$$

$$\theta_{i+1} = \theta_i - \delta_i \Delta\theta_i$$

- To simplify the computation of rotation matrix, the step angles are chosen such that

$$\tan \Delta\theta_i = 2^{-i}$$

Three Equations for Rotation and Angle Computation

$$\delta$$

$$x_{i+1} = x_i - \delta_i \, 2^{-i} y_i$$

$$\theta \qquad \theta \qquad \delta \; \Delta\theta$$

$$y_{i+1} = {}_I \, 2^{-i} x_i + y_i$$

$${}_{i+1} = \quad \delta_i = \begin{array}{c} 1 \\ i \\ -1 \end{array} \left. \right\} \begin{array}{l} \theta_i \geq 0 \\ i \\ \theta_i < 0 \end{array}$$

Rotation Matrix for interaction i requiring only shift

$$\begin{pmatrix} x_{i+1} \\ \\ y_{i+1} \end{pmatrix} = \underbrace{\boxed{1/\sqrt{1+2^{-2i}}}}_{(Ki)} \underbrace{\begin{pmatrix} 1 & \boxed{-\delta_i}\, 2^{-i} \\ \\ \boxed{\delta_i}\, 2^{-i} & 1 \end{pmatrix}}_{\text{Rotation Matrix (Ri)}} \begin{pmatrix} x_i \\ \\ y_i \end{pmatrix}$$

Known quantity

Unknown quantity

Iteration Formulation

$$\begin{pmatrix} x_{i+1} \\ y_{i+1} \end{pmatrix} = K_i R_i \begin{pmatrix} x_i \\ y_i \end{pmatrix}$$

Starting from location 0 going to location 1:

$$\begin{pmatrix} x_1 \\ y_1 \end{pmatrix} = K_0 R_0 \begin{pmatrix} x_0 \\ y_0 \end{pmatrix}$$

This is the point where we are giving $\Delta\theta_0 = \tan^{-1}(2^{-0})$ rotation

$x_0 = 1$ and $y_0 = 0$

Tracking the angle traverse

Initializing θ_0 to the desired angle

$\theta_0 = \theta_d \longrightarrow$ **desired angle**

In every iteration compute the direction of the next rotation

$$\theta_1 = \theta_0 - \delta_0 \Delta\theta_0$$

$$\delta$$

$$1 = \left.\begin{matrix} +1 \\ -1 \end{matrix}\right\} \quad \theta_1 > 0$$

$$\theta_1 < 0$$

Series of Rotation starting from (1,0)

- Sign bit of the current angle tells us the direction of the rotation

$$\begin{pmatrix} x_2 \\ y_2 \end{pmatrix} = K_1R_1 \begin{pmatrix} x_1 \\ y_1 \end{pmatrix}$$

$$\begin{pmatrix} x_2 \\ y_2 \end{pmatrix} = K_1K_0R_1R_0 \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

$$\begin{pmatrix} x_3 \\ y_3 \end{pmatrix} = K_2K_1K_0R_2R_1R_0 \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

Complete algorithm

$$\begin{bmatrix} x_N \\ y_N \end{bmatrix} = k_{N-1}K_{N-2}K_0R_{N-1}R_{N-2}\ldots R_0 \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

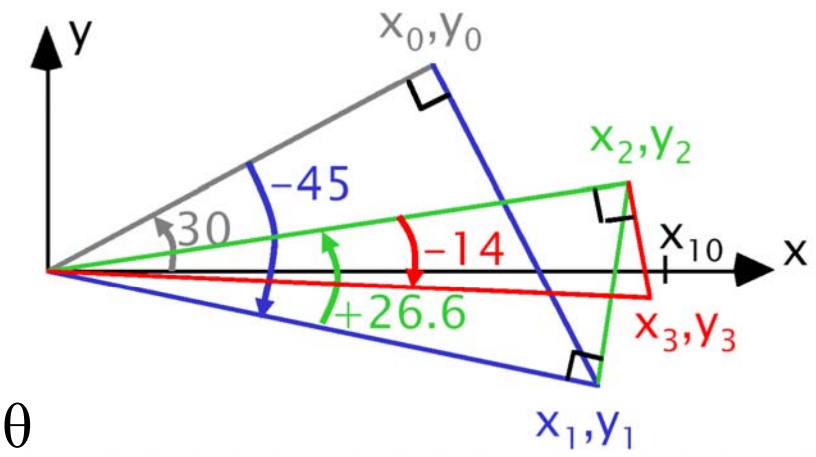
$$\begin{bmatrix} x_N \\ y_N \end{bmatrix} = {\color{red}K}_{,} R_{N-1}R_{N-2}\ldots R_0 \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

- Starting from (K, 0) instead of (1,0) in the first rotation will save multiplication by K of the final result

$$\begin{bmatrix} x_N \\ y_N \end{bmatrix} = R_{N-1}R_{N-2}\ldots R_0 \begin{bmatrix} K \\ 0 \end{bmatrix}$$

Hardware Realization: CORDIC Rotation Mode

- Algorithm: (θ is the current angle) θ_d θ_{i+1}
  - Mode: rotation: “each step, try to make θ_i zero”
  - Initialize x=0.607253, y=0, =
  - For i=0 $\rightarrow$ N
  - $\delta_i = 1$ when $\theta > 0$, else -1
  - $x_{i+1} = x_i - \delta_i \cdot 2^{-i} \cdot y_i$
  - $y_{i+1} = y_i + \delta_i \cdot 2^{-i} \cdot x_i$
  - $\theta_{i+1} = \theta_i - \delta_i \Delta\theta_i$
  - Result: $x_N = \cos\theta$, $y_N = \sin\theta$
  - Precision: n bits



Example: Rewriting Angles in Terms of α_i

θ_d

θ_0

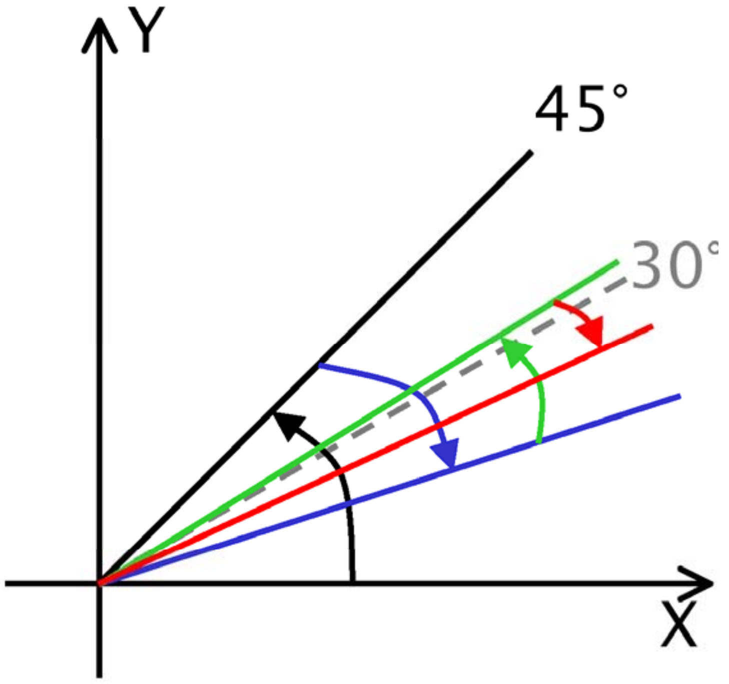
Example: $= 30.0^0$

- Start with $= 45.0$ (> 30.
- $45.0 - 26.6 = 18.4$ (< 30.
- $18.4 + 14.0 = 32.4$ (> 30.
- $32.4 - 7.1 = 25.3$ (< 30.
- $25.3 + 3.6 = 28.9$ (< 30.
- $28.9 + 1.8 = 30.7$ (> 30.
- θ . . .
- $\approx$ $= 30.0$

$45.0 - 26.6 + 14.0 - 7.1 + 3.6$

$+ 1.8 - 0.9 + 0.4 - 0.2 + 0.1$

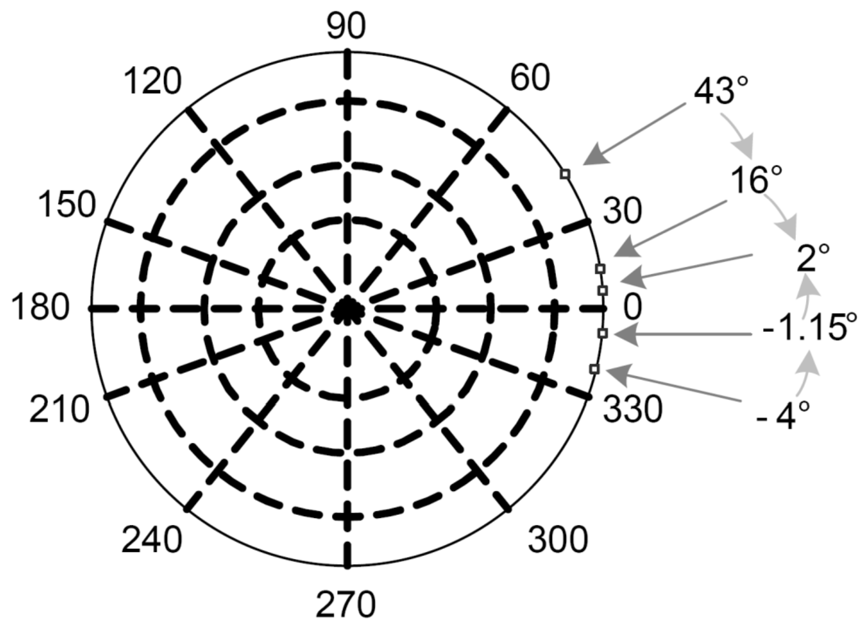
$= 30.1$



Iterations

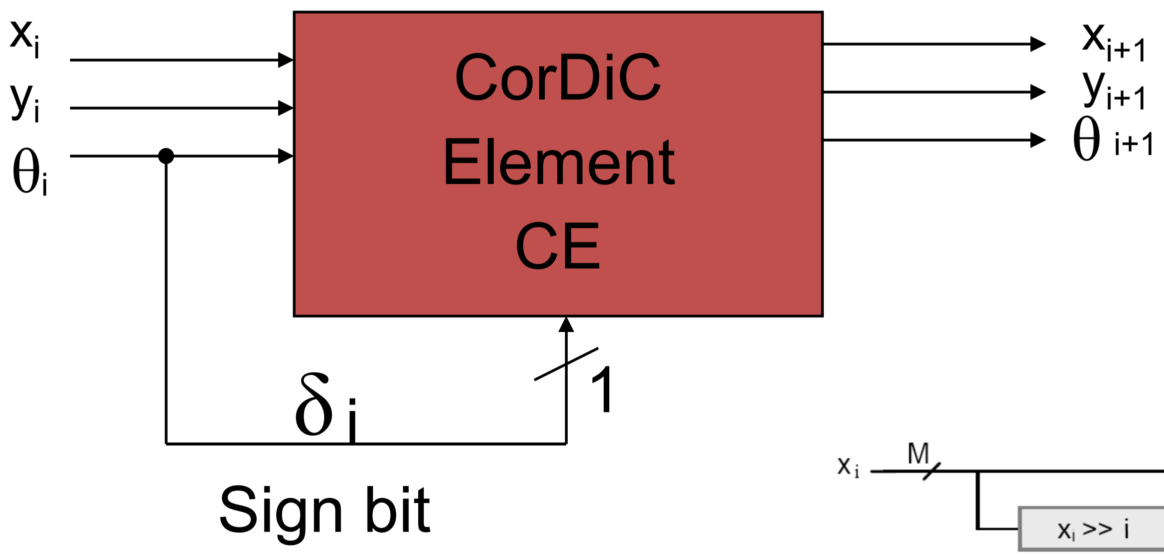
i	$\Delta\theta_i$ in degrees
0	43.0000
1	16.4349
2	2.3987
3	−4.7263
4	−1.1500
5	0.6399
6	−0.2552
7	0.1924
8	−0.0314
9	0.0805
10	0.0245
11	−0.0035
12	0.0105
13	0.0035
14	0.0000
15	−0.0017
16	−0.0008

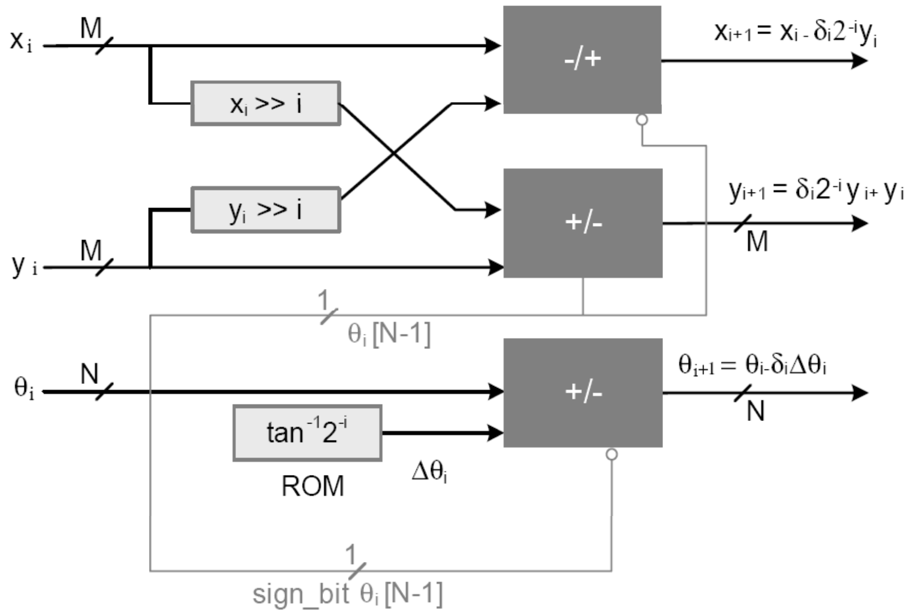
16 Iterations of CORDIC to compute cos and sin of 43°



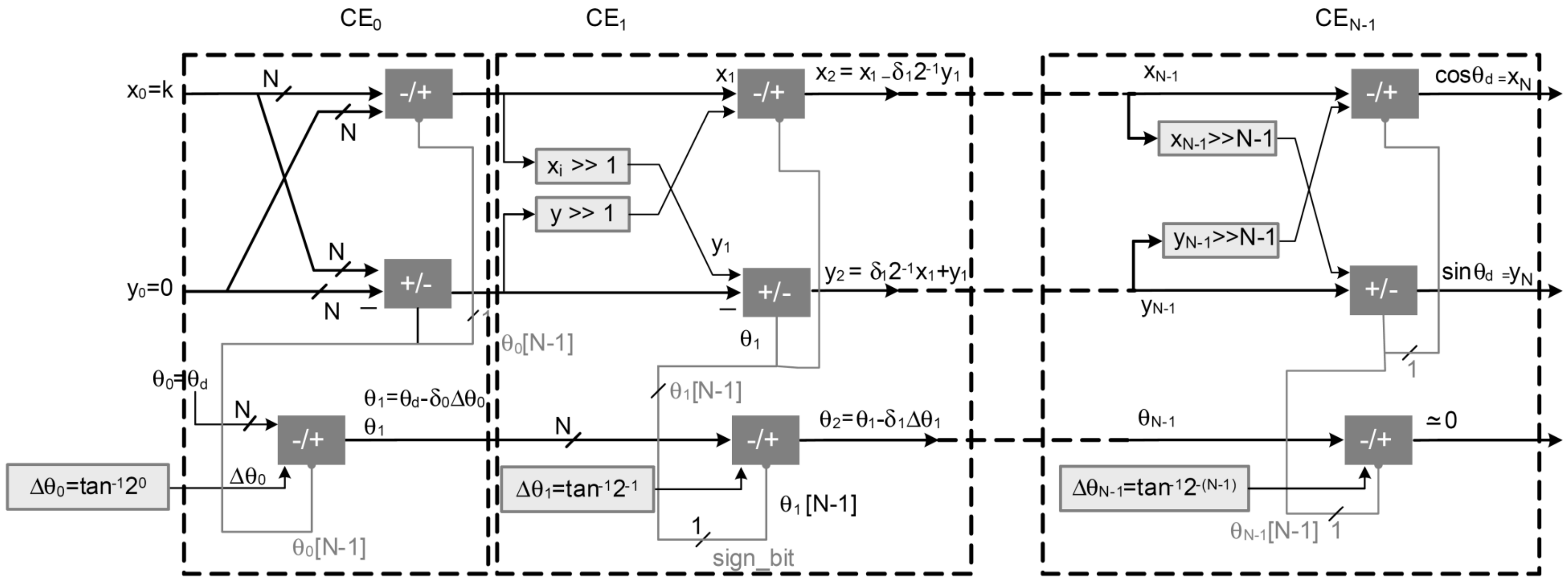
(b)

Architecture Mapping

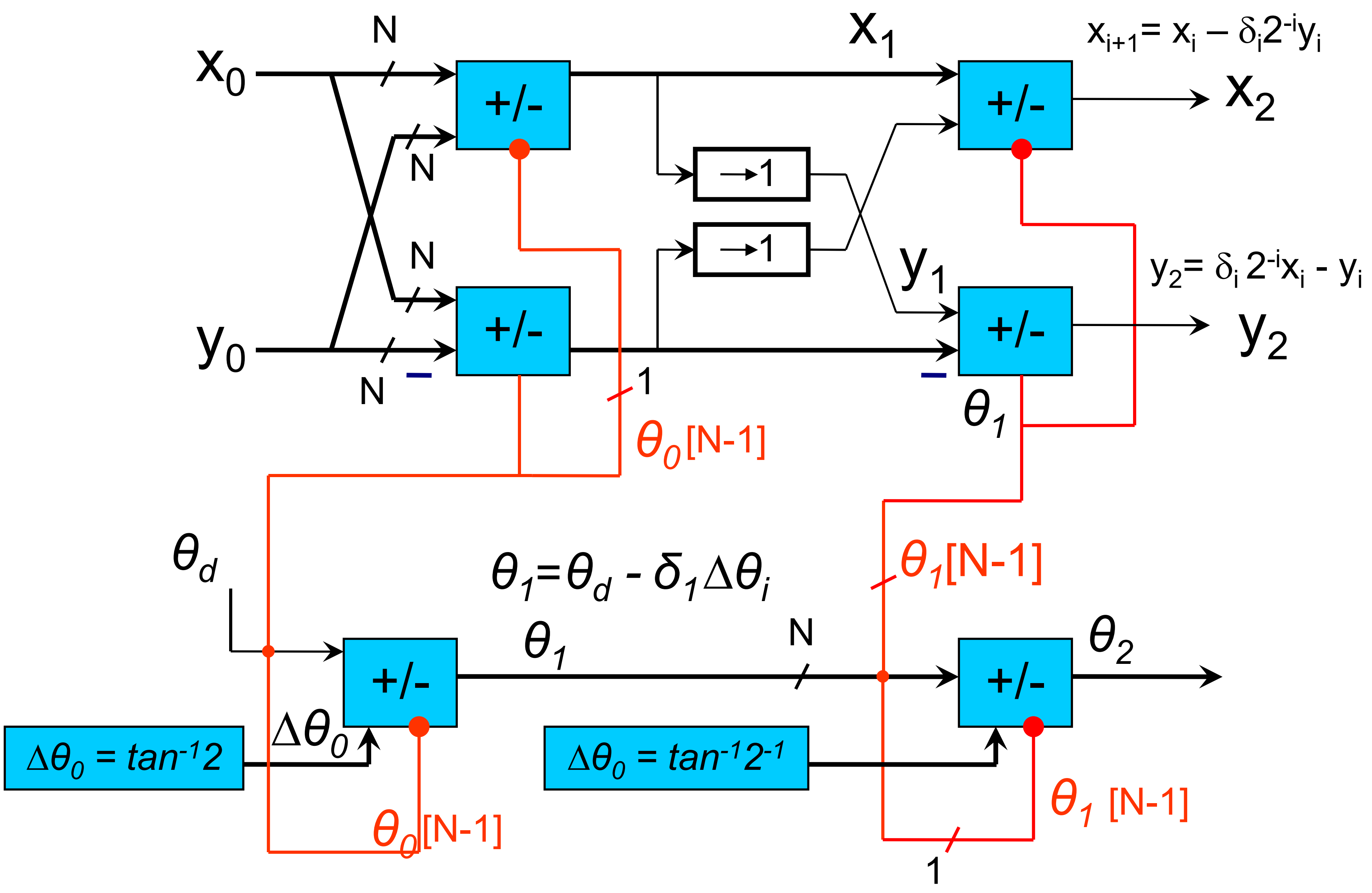



x_i
M
$x_i >> i$
-/+
$x_{i+1} = x_i - \delta_i 2^{-i} y_i$
$y_i >> i$
+/-
$y_{i+1} = \delta_i 2^{-i} y_i + y_i$
M
y_i
M
1
θ_i [N-1]
θ_i
N
+/-
$\theta_{i+1} = \theta_i - \delta_i \Delta\theta_i$
N
$\tan^{-1} 2^{-i}$
ROM
$\Delta\theta_i$
1
sign_bit θ_i [N-1]

CORDIC Architecture

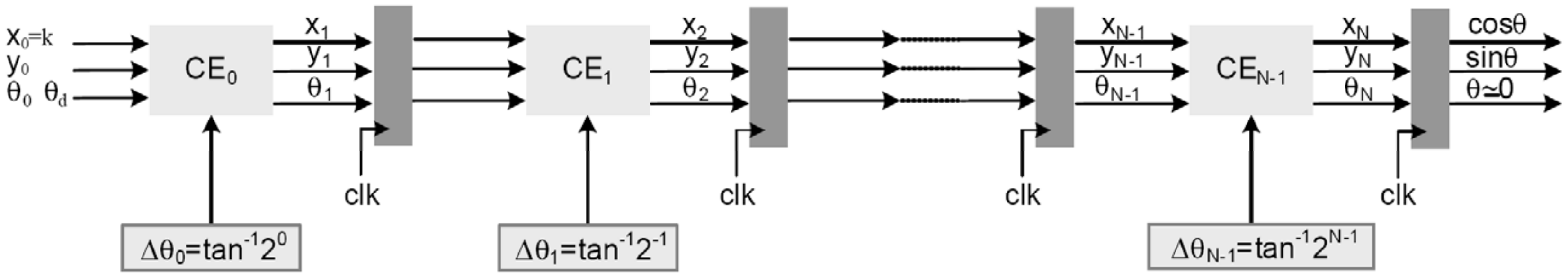


Example

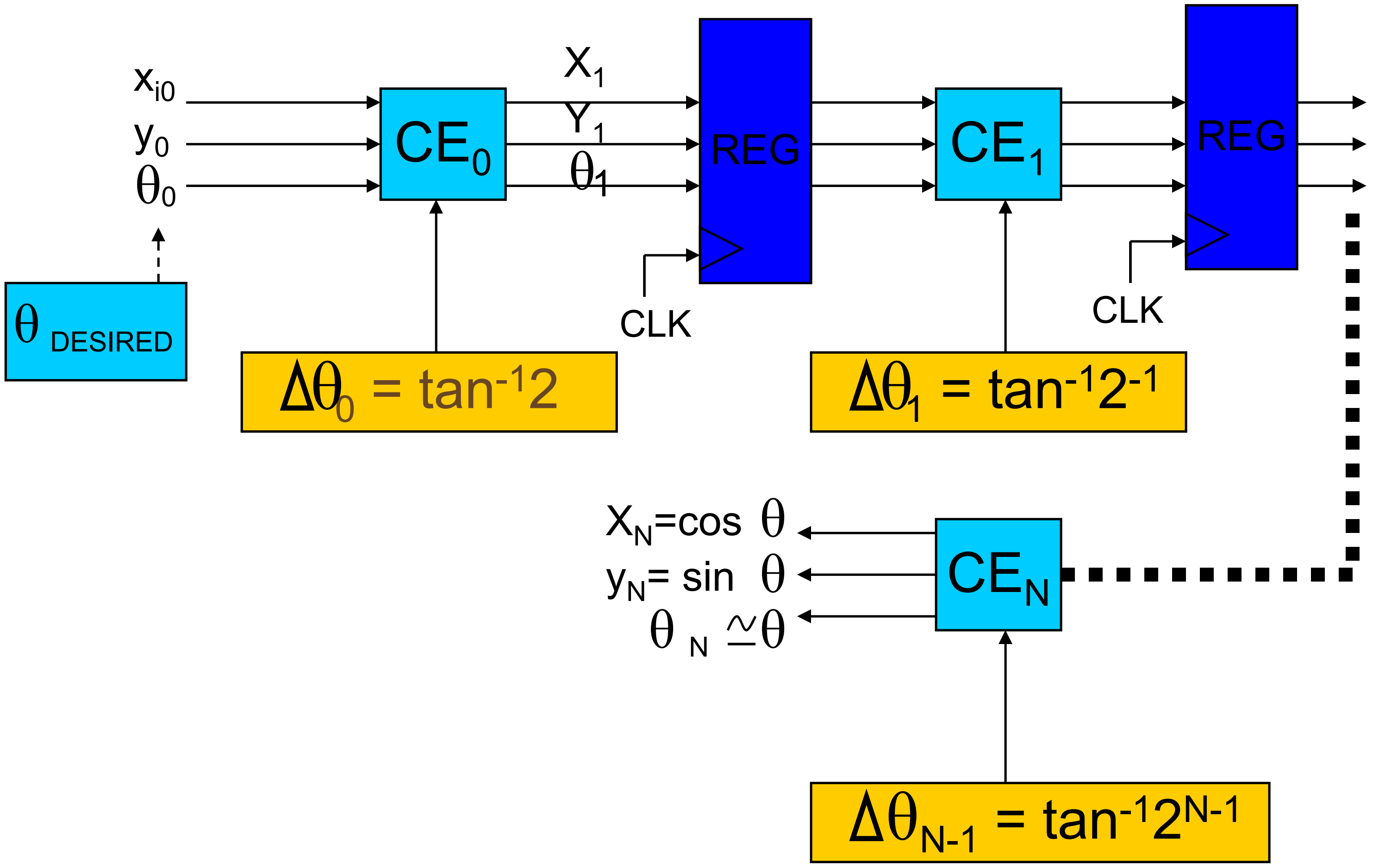


x_0
y_0
N
N
N
N
+/-
x_1
y_1
$\rightarrow 1$
$\rightarrow 1$
$x_{i+1} = x_i - \delta_i 2^{-i} y_i$
x_2
$y_2 = \delta_i\, 2^{-i} x_i - y_i$
y_2
–
1
θ_0[N-1]
θ_1
θ_d
$\theta_1 = \theta_d - \delta_1 \Delta\theta_i$
θ_1[N-1]
θ_1
N
θ_2
$\Delta\theta_0$
$\Delta\theta_0 = tan^{-1}2$
$\Delta\theta_0 = tan^{-1}2^{-1}$
θ_0[N-1]
θ_1 [N-1]
1

CORDIC Architecture



Pipelined Design

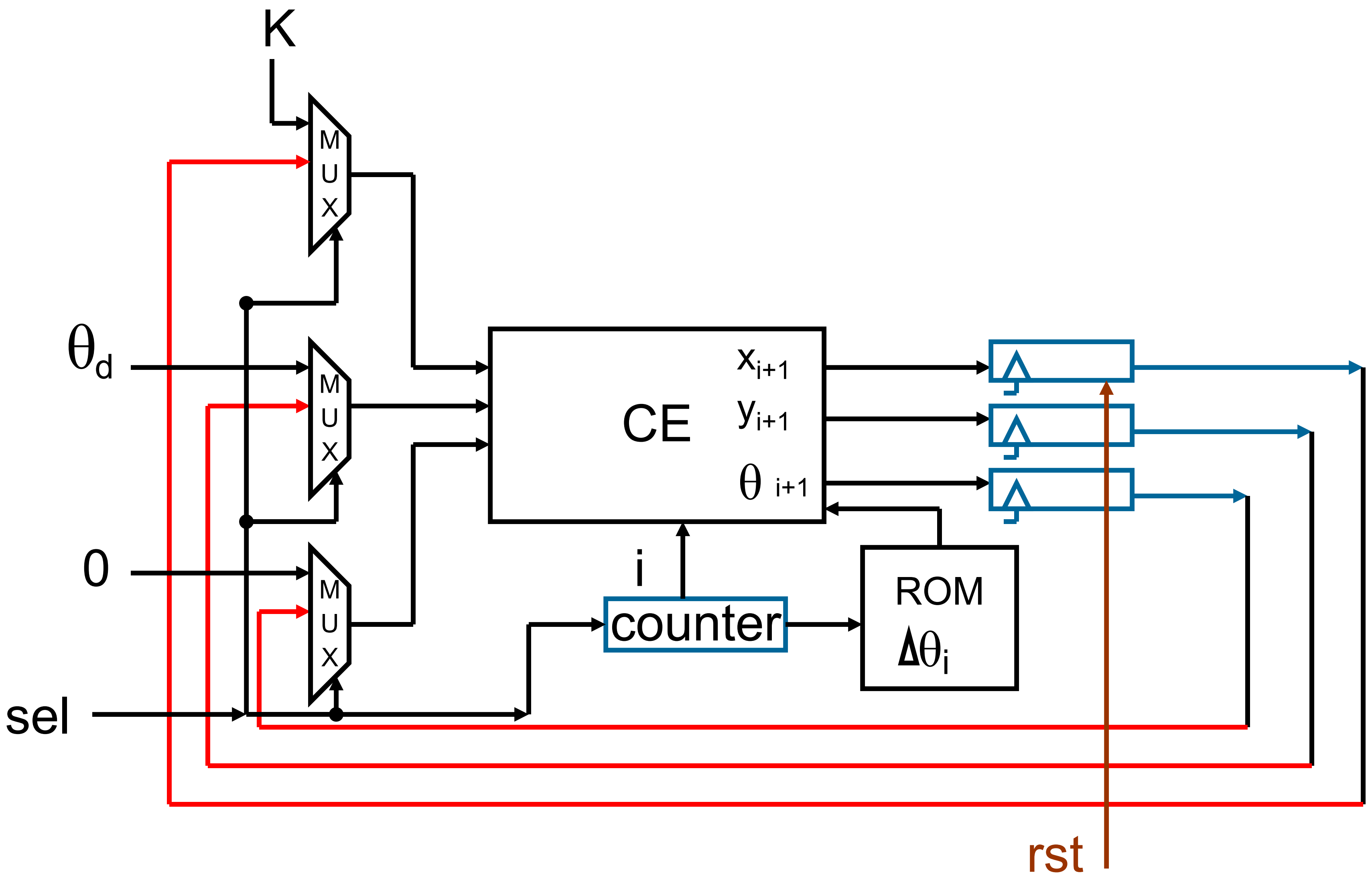


Verilog Code

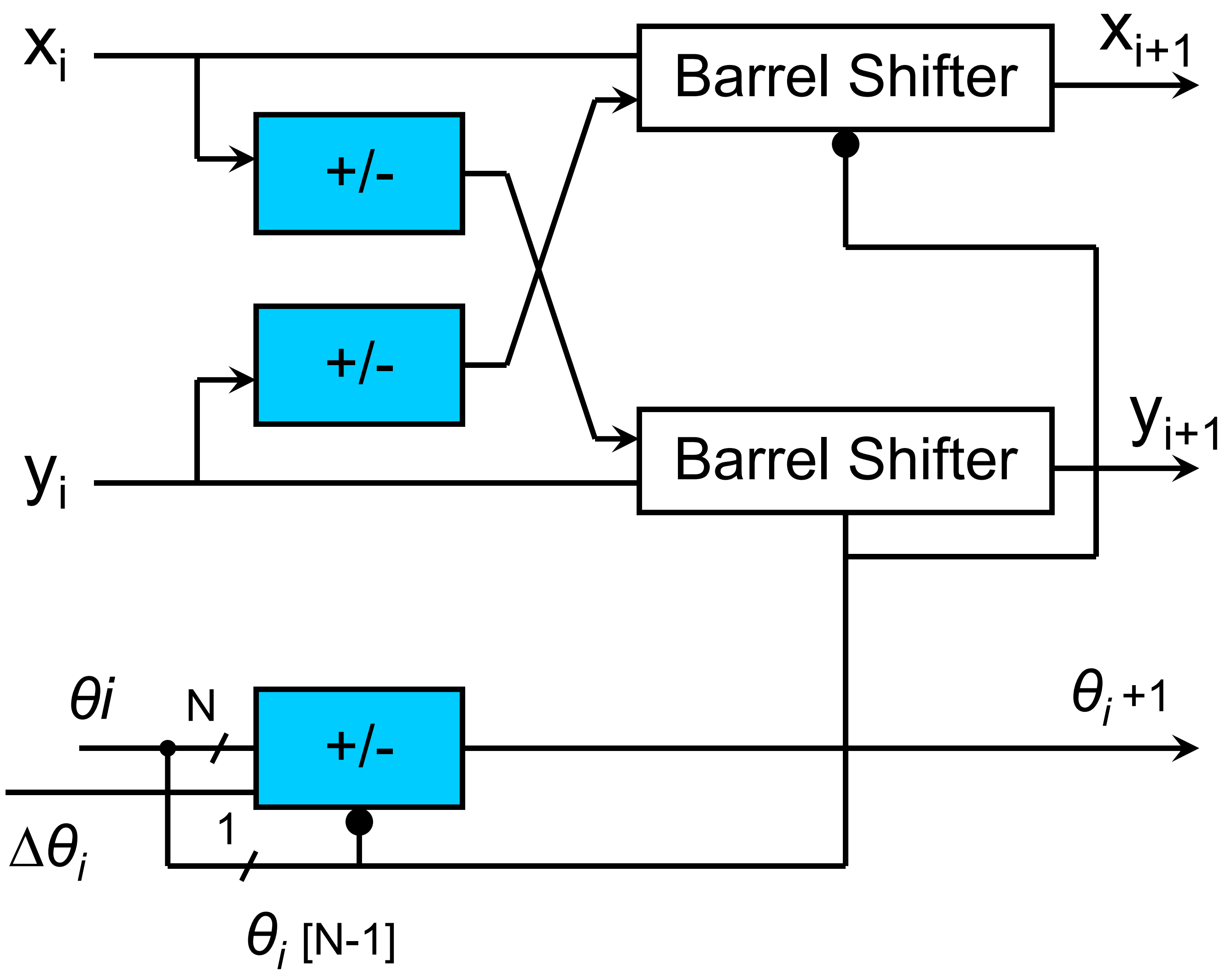
Define CorDiC Element as a task, having inputs x_0,y_0, $theta_0$, which are kept on being recalled in a for loop.

```
for(i=0; i<=N-1; i=i+1)
CEtask(x[i], y[i], theta[i], i, del_theta[i], x[i+1], y[i+1], theta[i+1])
always @(posedge clk)
for(i=0; i<=N-1; i=i+1) //Replication of hardware
begin
    x_reg[i+1] <= x[i];
    y_reg[i+1] <= y[i];
    theta_reg[i+1] <= theta[i];
end
```

Time Shared Architecture



CORDIC Element for computing x_{i+1} and y_{i+1}



Modified CORDIC Algorithm

$$\theta_i = 0\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ldots$$

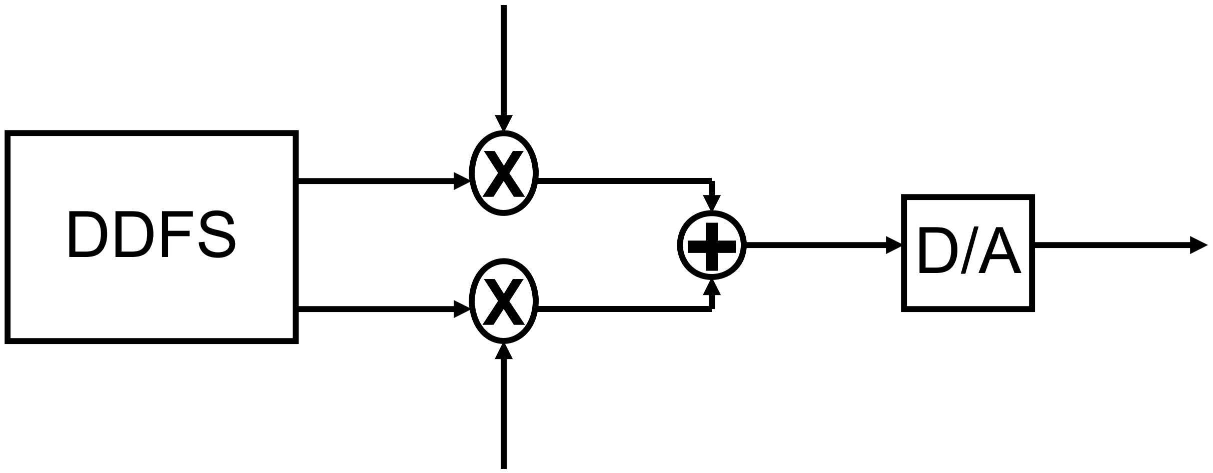
$$\theta_i = 0 + 2^{-1} + 2^{-6} + \ldots$$

$$\theta_i = \sum_{i=0}^{N-1} \Delta\theta_i 2^{-i}$$

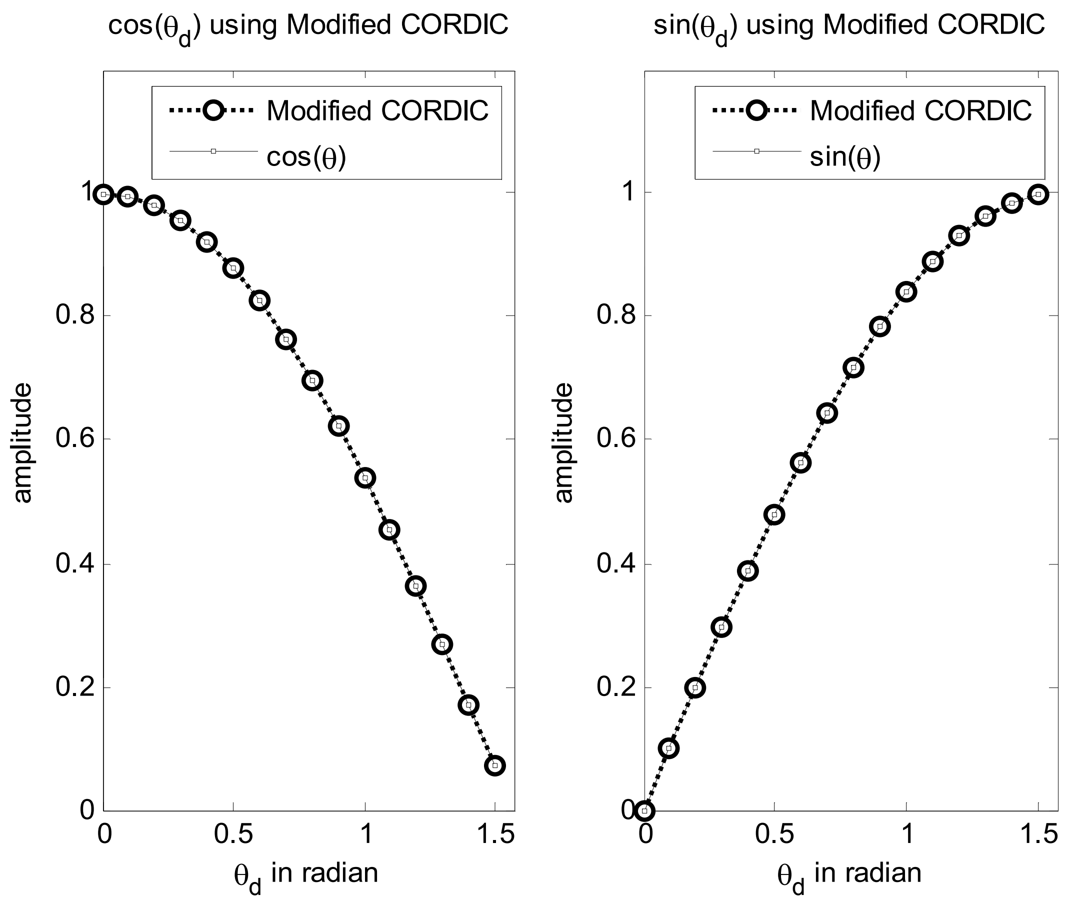
$$\Delta\theta_i = 0,1$$

$\theta_i = 0\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ldots$

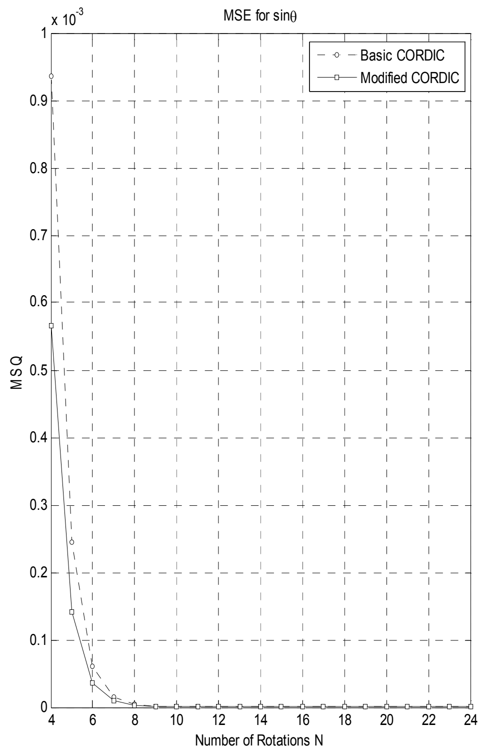
Where 1 gives that value i.e., rotate the weight of the bit, where 0's do not rotate hence we reach the desired angle

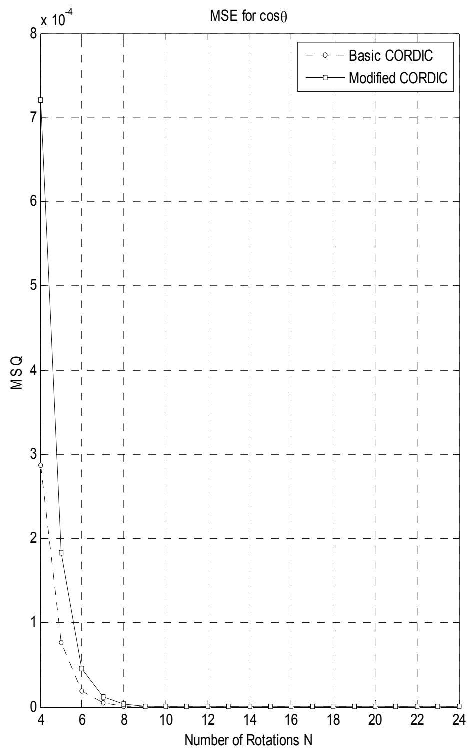


Results using CORDIC and modified CORDIC Algorithm



Hardware Mapping of Modified CORDIC Algorithm





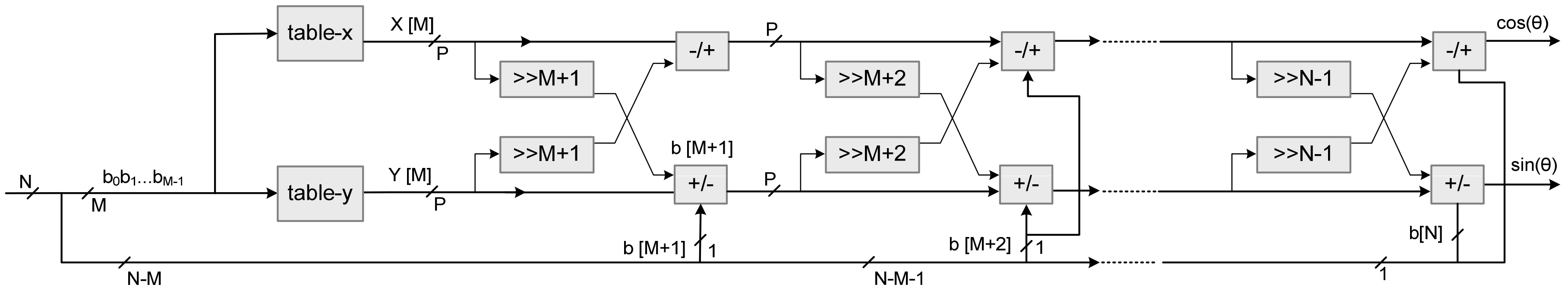
The MATLAB code

```
tableX=[ ];
tableY=[ ];

N = 16;
K = 1;
for i = 1:N
  K = K * cos(2^(-(i)));
end
 % the constant initial rotation
theta_init = (2)^0 - (2)^(-N);
x0 = K*cos(theta_init);
y0 = K*sin(theta_init);
cosine = [ ];
sine = [ ];

M = 4;
for index = 0:2^M-1
  for k=1:M
      b(M+1-k) = rem(index, 2);
      index = fix(index/2);
  end
```

Contd…

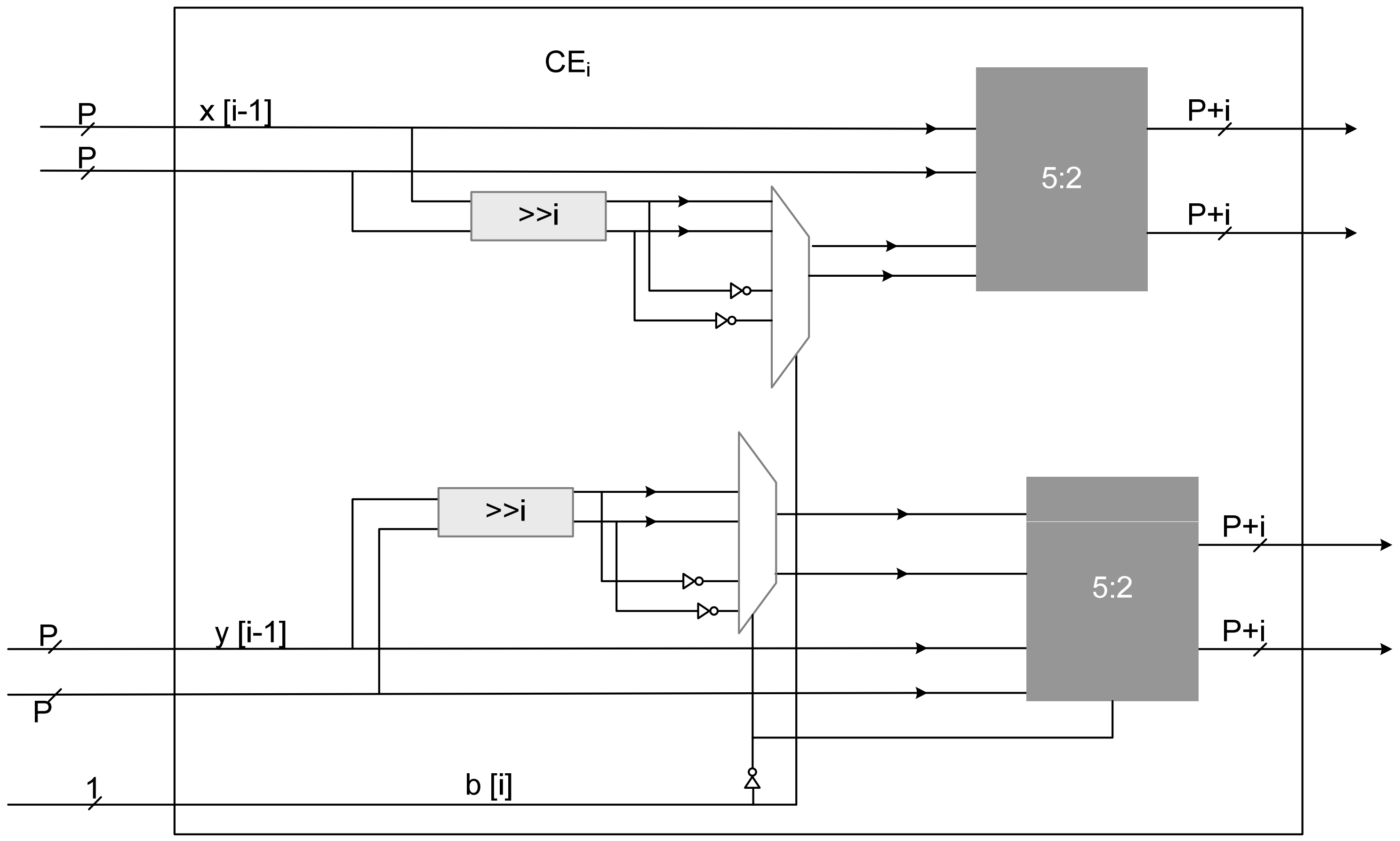
```
    % recoding b as r with +1,-1
    for k=1:M
        r(k) = 2*b(k) - 1;
    end
     % first Modified CORDIC rotation
    x(1) = x0 - r(1)*(tan(2^(-1)) * y0);
    y(1) = y0 + r(1)*(tan(2^(-1)) * x0);

    % rest of the Modified CORDIC rotations
    for k=2:M,
            x(k) = x(k-1) - r(k)* tan(2^(-k)) * y(k-1);
            y(k) = y(k-1) + r(k) * tan(2^(-k)) * x(k-1);
    end
    tableX = [tableX x(M)];
    tableY = [tableY y(M)];
end
```

Hardware Optimization

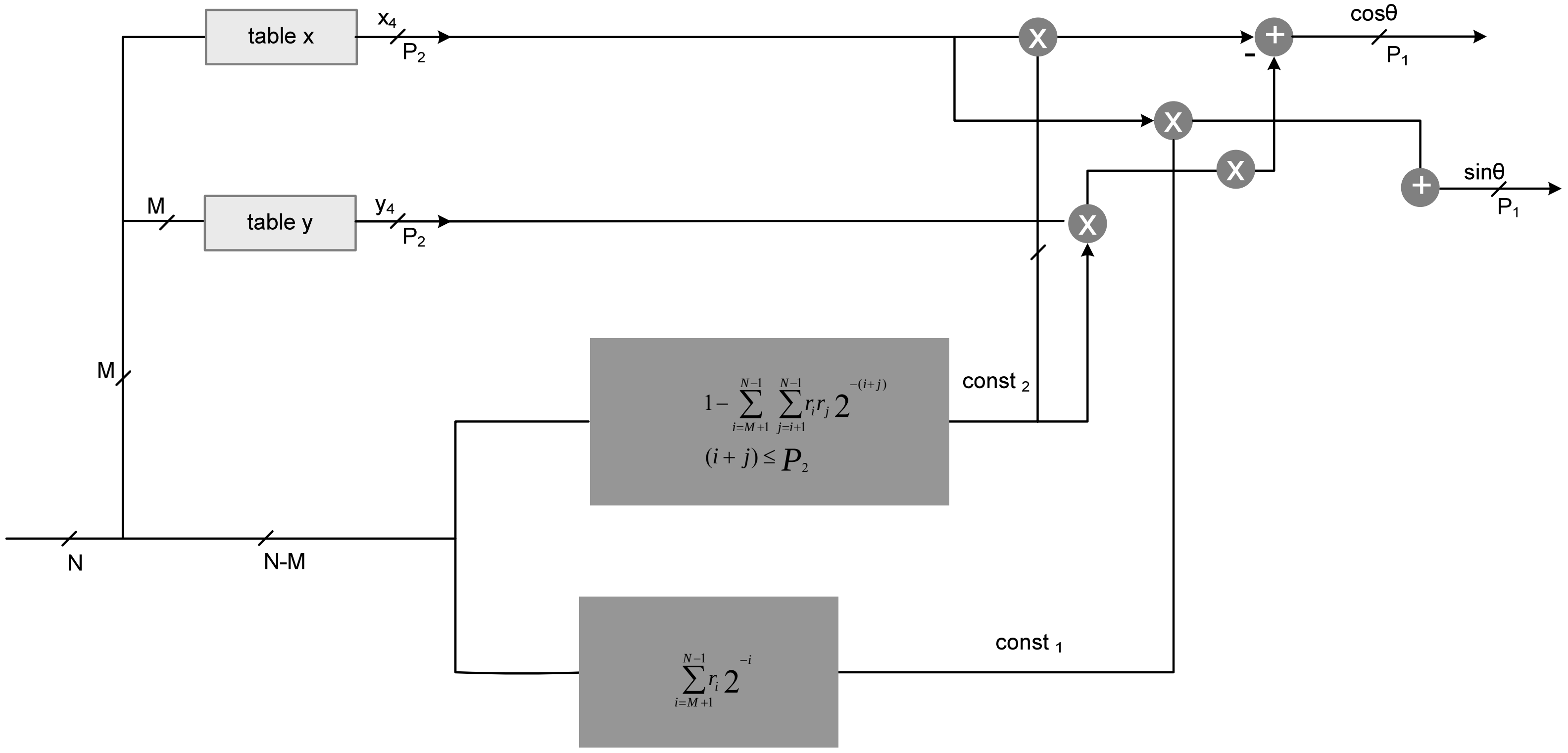


FDA of Modified CORDIC algorithm

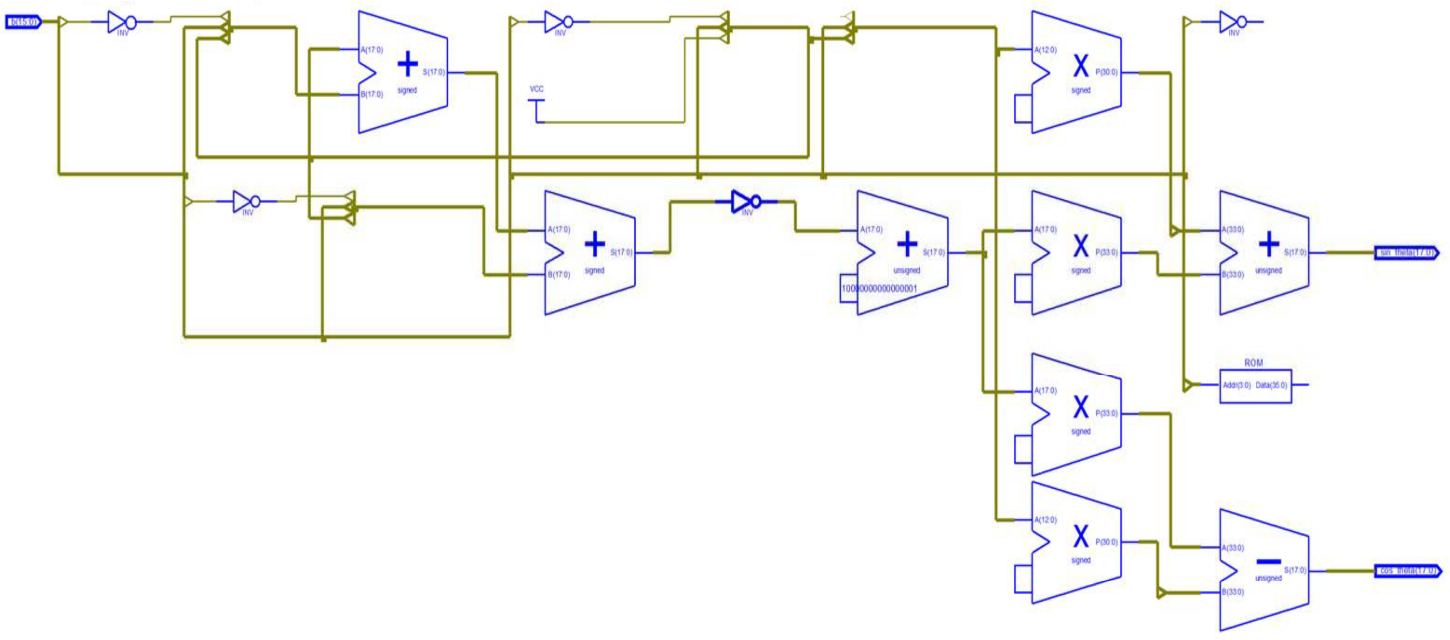
A CE with compression tree



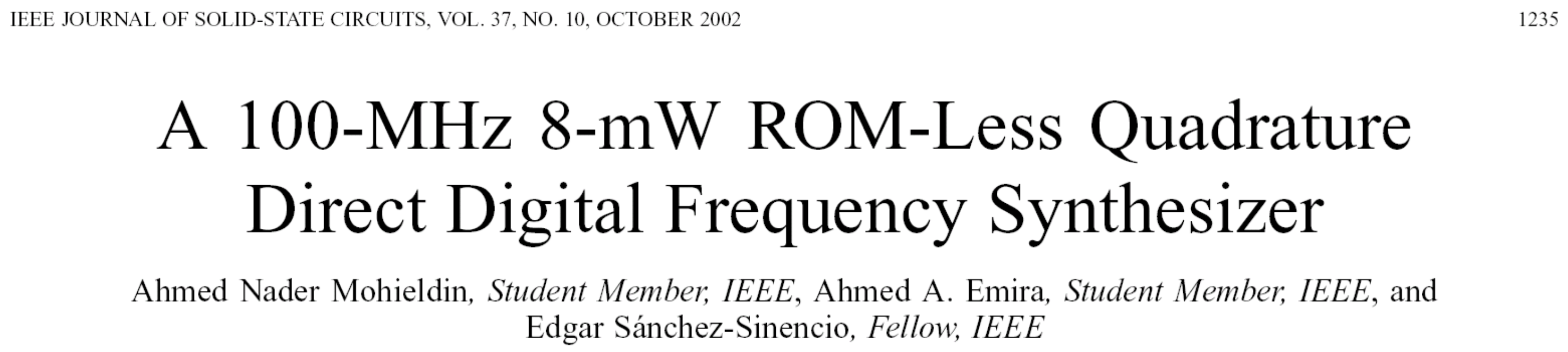
Optimal HW Design for Modified CORDIC Algorithm



Schematic of single-stage CORDIC design



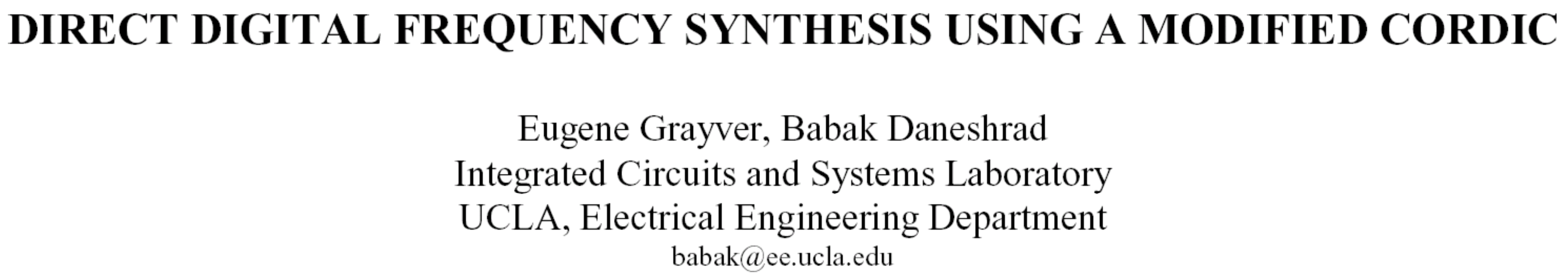
Publications



IEEE JOURNAL OF SOLID-STATE CIRCUITS, VOL. 37, NO. 10, OCTOBER 2002 1235

A 100-MHz 8-mW ROM-Less Quadrature Direct Digital Frequency Synthesizer

Ahmed Nader Mohieldin, *Student Member, IEEE*, Ahmed A. Emira, *Student Member, IEEE*, and Edgar Sánchez-Sinencio, *Fellow, IEEE*



DIRECT DIGITAL FREQUENCY SYNTHESIS USING A MODIFIED CORDIC

Eugene Grayver, Babak Daneshrad
Integrated Circuits and Systems Laboratory
UCLA, Electrical Engineering Department
babak@ee.ucla.edu

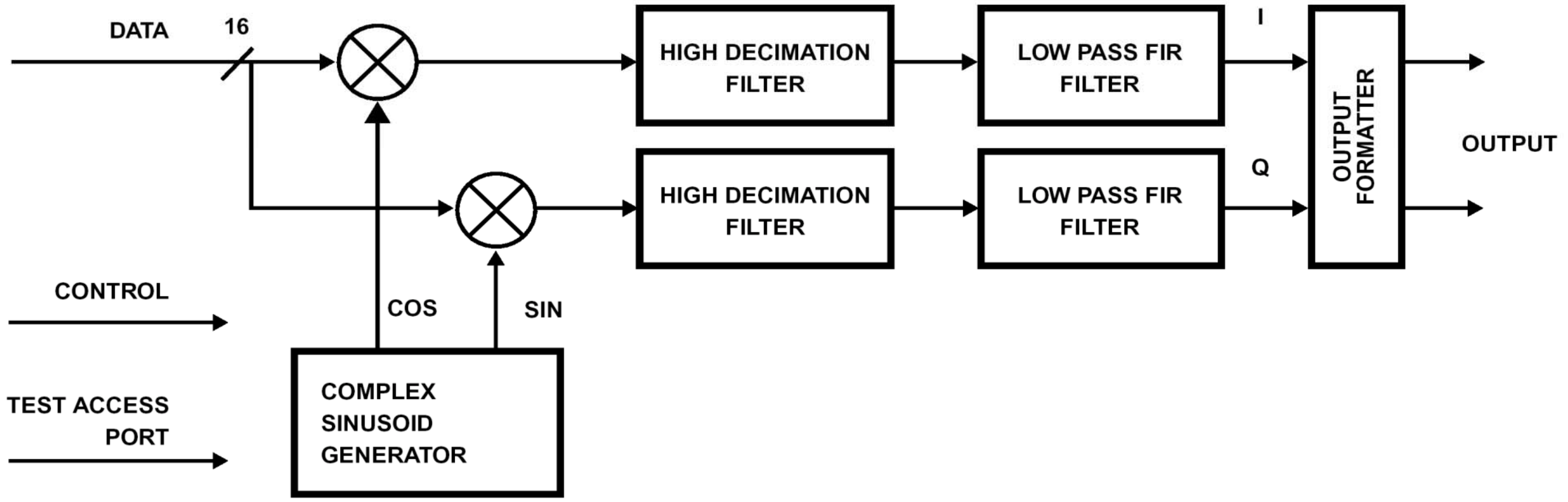
Henry Nicholas PhD Work

IEEE JOURNAL OF SOLID-STATE CIRCUITS, VOL. 26, NO. 12, DECEMBER 1991 1959

A 150-MHz Direct Digital Frequency Synthesizer in 1.25-μm CMOS with -90-dBc Spurious Performance

Henry T. Nicholas, III, and Henry Samueli, *Member, IEEE*

BLOCK DIAGRAM OF HSP50016 DIGITAL DOWN CONVERTER



Questions/Feedback