CORDIC
(COordinate Rotation DIgital Computer)

For Advanced VLSI

Major Reference: Y. H. Hu, "CORDIC-based VLSI architectures for digital signal processing," *IEEE Signal Processing Mag.*, pp.16-35, July 1992.

2002/9/25

Outline

- Introduction
- Conventional CORDIC Algorithm
- Enhancement of CORDIC
  - MVR-CORDIC Algorithm
  - EEAS-Based CORDIC Algorithm
- Vector Rotational CORDIC Family

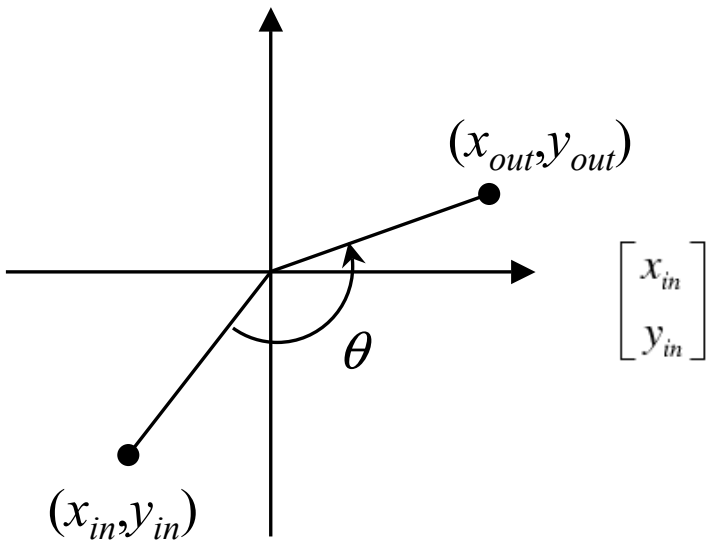
Introduction

- Vector rotation is the kernel of various digital signal processing (DSP) applications, including
  - Digital filters:
    - Orthogonal digital filters, and adaptive lattice filters.
  - Linear transformation:
    - DFT, Chirp-Z transform, DHT, and FFT.
  - Matrix-based digital signal processing algorithms:
    - QR factorization, with applications to Kalman filtering and QR decomposition (QRD) based adaptive filtering.
  - Linear system solvers
    - such as Toeplitz and covariance system solvers,……,*etc.*

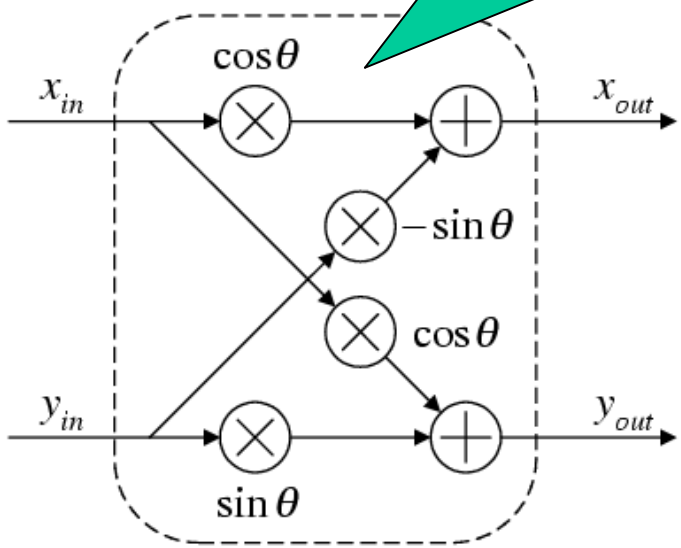
Rotational Operation

$$\begin{bmatrix} x_{out} \\ y_{out} \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \cdot \begin{bmatrix} x_{in} \\ y_{in} \end{bmatrix} \triangleq \mathbf{R} \cdot \begin{bmatrix} x_{in} \\ y_{in} \end{bmatrix}.$$

Each vector rotation takes **4 multiplications** and **2 additions**



$\begin{bmatrix} x_{in} \\ y_{in} \end{bmatrix}$

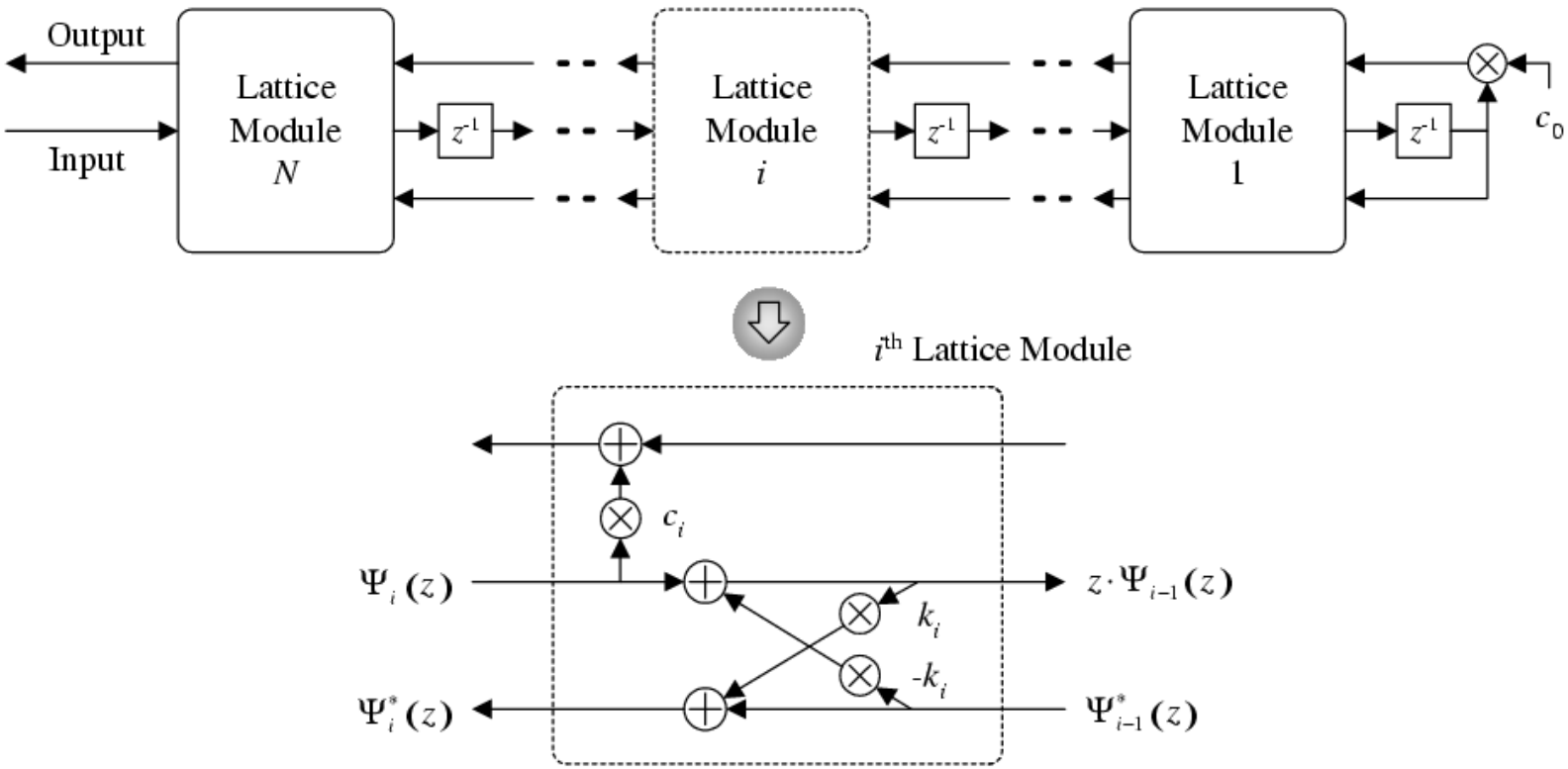


$\begin{bmatrix} x_{out} \\ y_{out} \end{bmatrix}$

Rotational Circuit

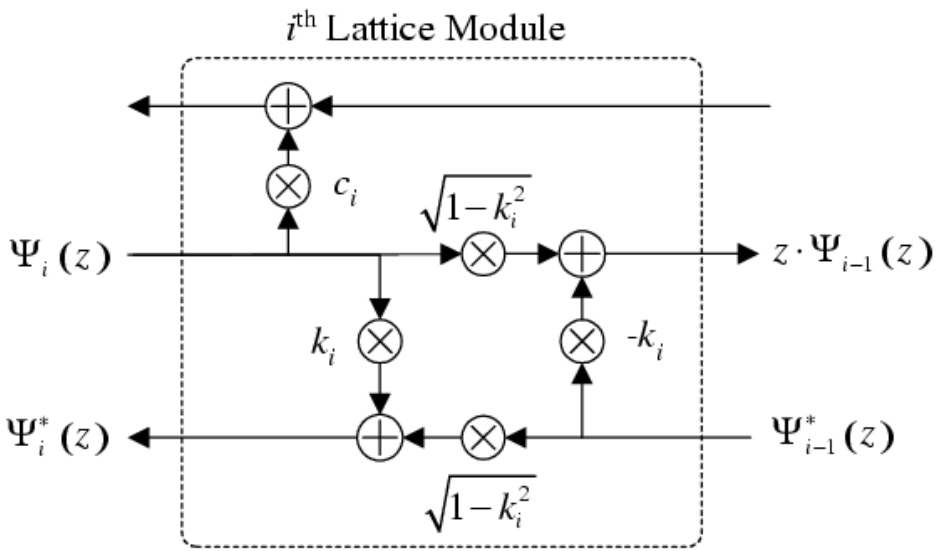
Digital Lattice Filter

- Low-sensitivity to coefficient quantization error
- Smaller dynamic range



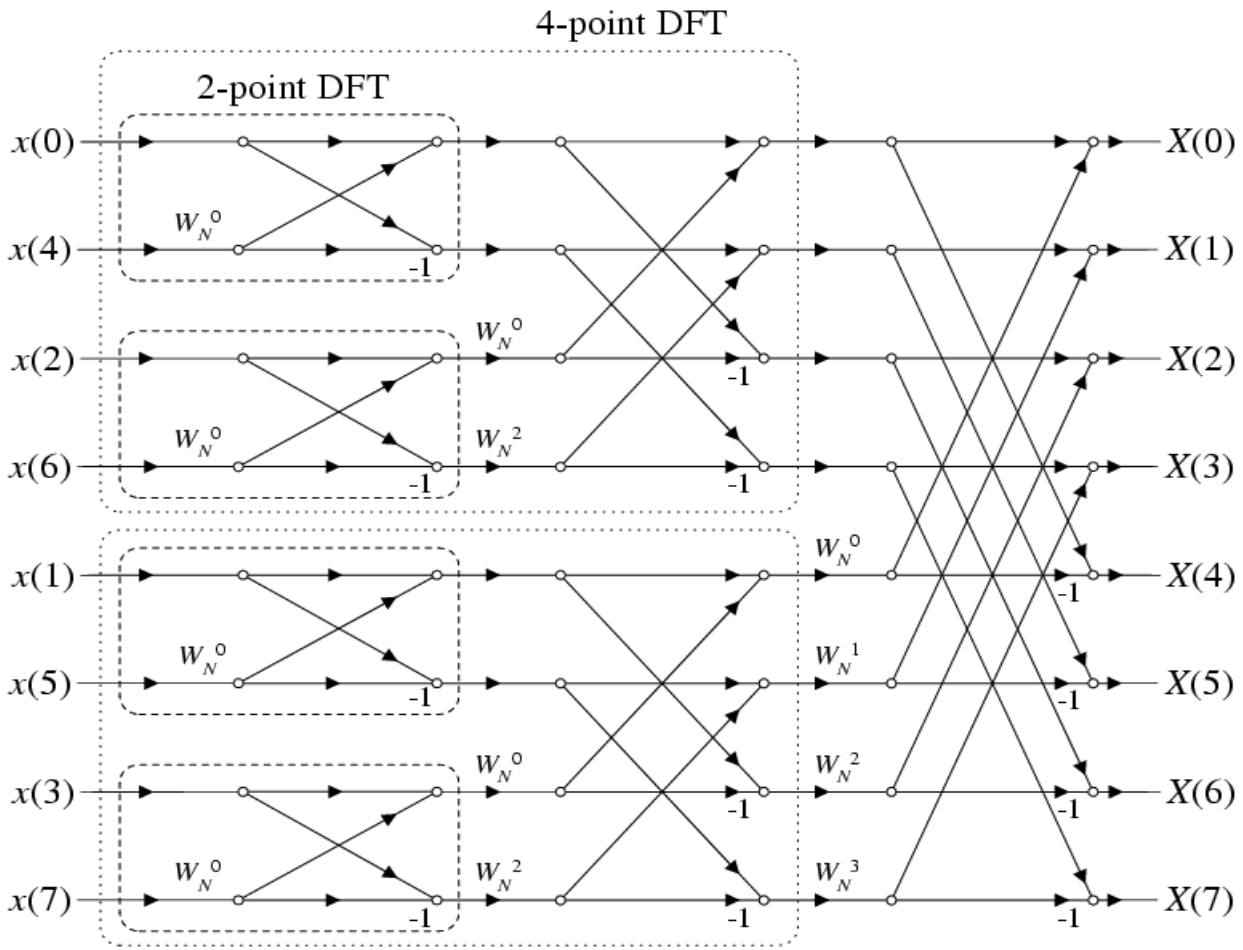
Normalized Lattice Section

- Givens Rotation



$$\begin{bmatrix} z\Psi_{i-1}(z) \\ \Psi_i^*(z) \end{bmatrix} = \begin{bmatrix} \sqrt{1-k_i^2} & -k_i \\ k_i & \sqrt{1-k_i^2} \end{bmatrix} \cdot \begin{bmatrix} \Psi_i(z) \\ \Psi_{i-1}^*(z) \end{bmatrix}.$$

Fast Fourier Transformation



Fast Fourier Transformation

- Twiddle factor

$$W_N^{nk} = e^{-j(2\pi nk/N)}$$

$$\triangleq e^{-j\varphi} = \cos\varphi + j\sin\varphi,$$

$$G = S \cdot W_N^{nk} = (s_r + j \cdot s_i) \cdot (\cos\varphi + j\sin\varphi)$$

$$= (s_r \cdot \cos\varphi - s_i \cdot \sin\varphi) + j\,(s_r \cdot \sin\varphi + s_i \cdot \cos\varphi)\,.$$

$$\begin{bmatrix} g_r \\ g_i \end{bmatrix} = \begin{bmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{bmatrix} \cdot \begin{bmatrix} s_r \\ s_i \end{bmatrix}.$$

Outline

- Introduction
- ✓ Conventional CORDIC Algorithm
- Enhancement of CORDIC
  - MVR-CORDIC Algorithm
  - EEAS-Based CORDIC Algorithm
- Vector Rotational CORDIC Family

Basic Concept of CORDIC Operation

- To decompose the **desired rotation angle (θ)** into the weighted sum of a set of ***predefined* elementary rotation angles ($\theta(i)$, $I=0,1,2,\ldots W$)**

- Such that the rotation through each of them can be accomplished with **simple *shift-and-add* operation**.

Conventional CORDIC Algorithm

$$\begin{bmatrix} x(i+1) \\ y(i+1) \end{bmatrix} = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \cdot \begin{bmatrix} x(i) \\ y(i) \end{bmatrix}$$

$$\Downarrow$$

$$\begin{bmatrix} x(i+1) \\ y(i+1) \end{bmatrix} = \cos\alpha \cdot \begin{bmatrix} 1 & -\tan\alpha \\ \tan\alpha & 1 \end{bmatrix} \cdot \begin{bmatrix} x(i) \\ y(i) \end{bmatrix}$$

$$\Downarrow$$

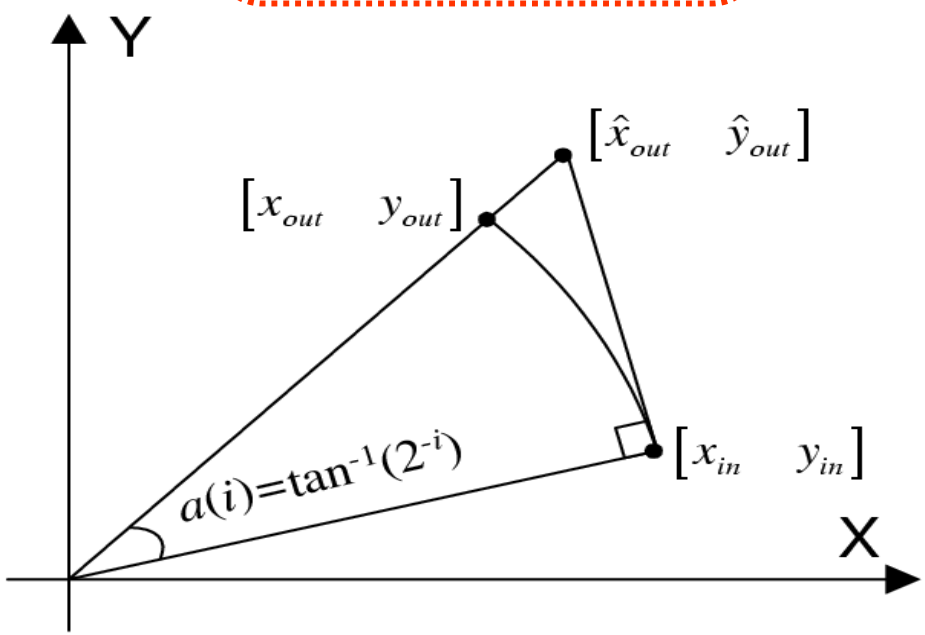
$$\begin{bmatrix} x(i+1) \\ y(i+1) \end{bmatrix} = \begin{bmatrix} 1 & -\mu_i 2^{-i} \\ \mu_i 2^{-i} & 1 \end{bmatrix} \cdot \begin{bmatrix} x(i) \\ y(i) \end{bmatrix}$$

$$where\ (\alpha_i = \tan^{-1}\mu_i 2^{-i} = \mu_i \tan^{-1} 2^{-i})$$

Conventional CORDIC Algorithm

- Ease-to-implementation (shift-and-add only)

$$\begin{bmatrix} x_{out} \\ y_{out} \end{bmatrix} = \frac{1}{\sqrt{1+2^{-2i}}} \cdot \begin{bmatrix} 1 & -2^{-i} \\ 2^{-i} & 1 \end{bmatrix} \begin{bmatrix} x_{in} \\ y_{in} \end{bmatrix} \triangleq \frac{1}{\sqrt{1+2^{-2i}}} \cdot \begin{bmatrix} \hat{x}_{out} \\ \hat{y}_{out} \end{bmatrix}$$



Elementary Angle Set of Conventional CORDIC

Iteration Index	Elementary Angle	Value in Radius
$i = 0$	$a(0) = \tan^{-1}(2^{-0})$	0.785398
$i = 1$	$a(1) = \tan^{-1}(2^{-1})$	0.463648
$i = 2$	$a(2) = \tan^{-1}(2^{-2})$	0.244979
$i = 3$	$a(3) = \tan^{-1}(2^{-3})$	0.124355
$i = 4$	$a(4) = \tan^{-1}(2^{-4})$	0.062419
$i = 5$	$a(5) = \tan^{-1}(2^{-5})$	0.031240
$i = 6$	$a(6) = \tan^{-1}(2^{-6})$	0.015624
$i = 7$	$a(7) = \tan^{-1}(2^{-7})$	0.007812

Conventional CORDIC with N=W=8

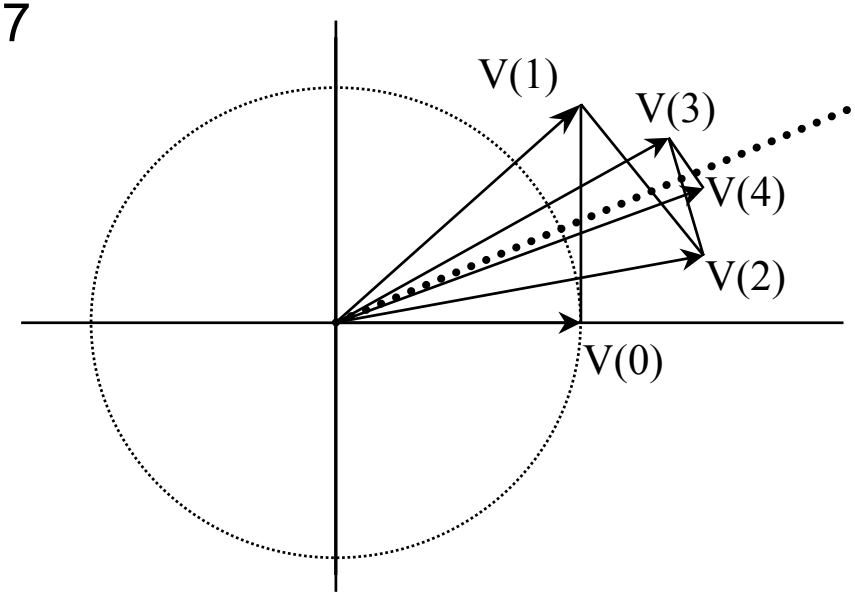
Conventional CORDIC Algorithm

- Sequentially performing of sub-angles, a(i)

$$\xi_{m,CORDIC} \equiv \theta - \left[\sum_{i=0}^{N-1} \mu(i)a(i)\right], \quad \mu(i) = \{+1, -1\}$$

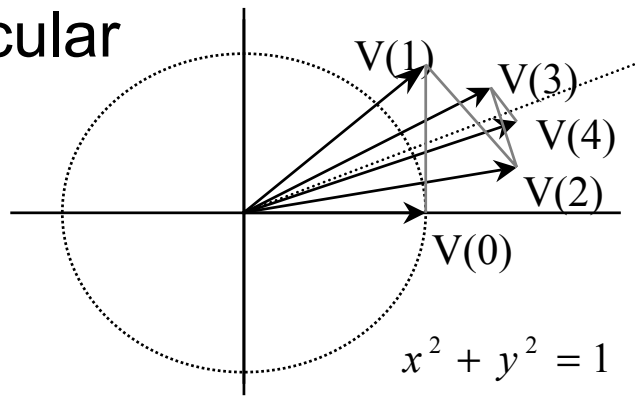
- Example:
  - Rotation angle: θ = π/8 = 0.3927

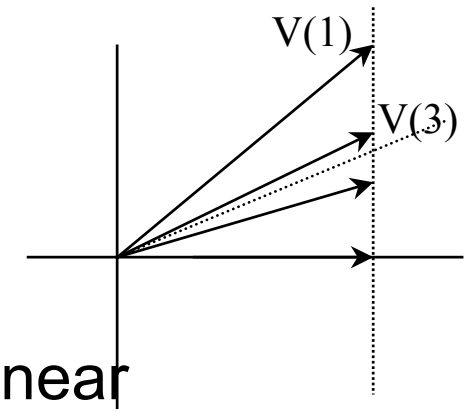
$$\theta \approx \mu(0)a(0) + \mu(1)a(1) + \mu(2)a(2) + \mu(3)a(3) + \cdots$$



Generalized CORDIC

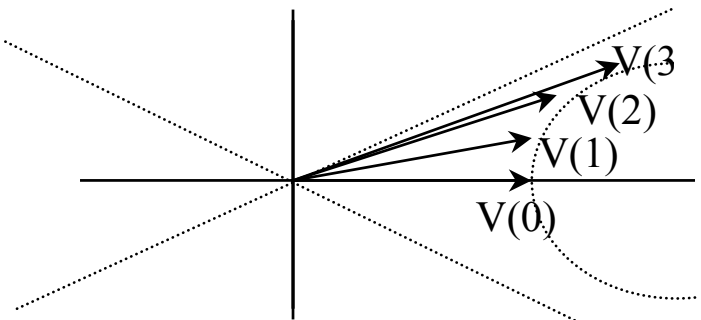
Circular





Linear

$$a_m(i) = \frac{1}{\sqrt{m}} \tan^{-1}[\sqrt{m}\,2^{-s(m,i)}] = \begin{cases} -2^{s(0,1)} & m \to 0 \\ \tan^{-1} 2^{-s(1,i)} & m = 1 \\ \tanh^{-1} 2^{-s(-1,i)} & m = -1 \end{cases}$$



Hyperbolic

m→0 , Linear system ;

m=1 , Circular system ;

m=-1 , Hyperbolic system.

Summary of CORDIC Algorithm

- Both micro-rotation and scaling phases

% Initialization

Given $x(0), y(0)$, and $z(0)$

% Micro-rotation phase

FOR $i = 0$ to $N-1$

$$\begin{bmatrix} x(i+1) \\ y(i+1) \end{bmatrix} = \begin{bmatrix} 1 & -\mu(i)2^{-i} \\ \mu(i)2^{-i} & 1 \end{bmatrix} \begin{bmatrix} x(i) \\ y(i) \end{bmatrix}$$

% Angle updating

$z(i+1) = z(i) - \mu(i)a(i)$, where $a(i) = \arctan(2^{-i})$

END

% Scaling phase

$$\begin{bmatrix} x_f \\ y_f \end{bmatrix} = P \begin{bmatrix} x(N) \\ y(N) \end{bmatrix} = \frac{1}{\prod_{i=0}^{N-1} \sqrt{1+2^{-2i}}} \begin{bmatrix} x(N) \\ y(N) \end{bmatrix}$$

Modes of CORDIC Operations

- **Vector rotation mode** (θ is given)
  - ***The objective*** is to compute the final vector $[x_f, y_f]^T$

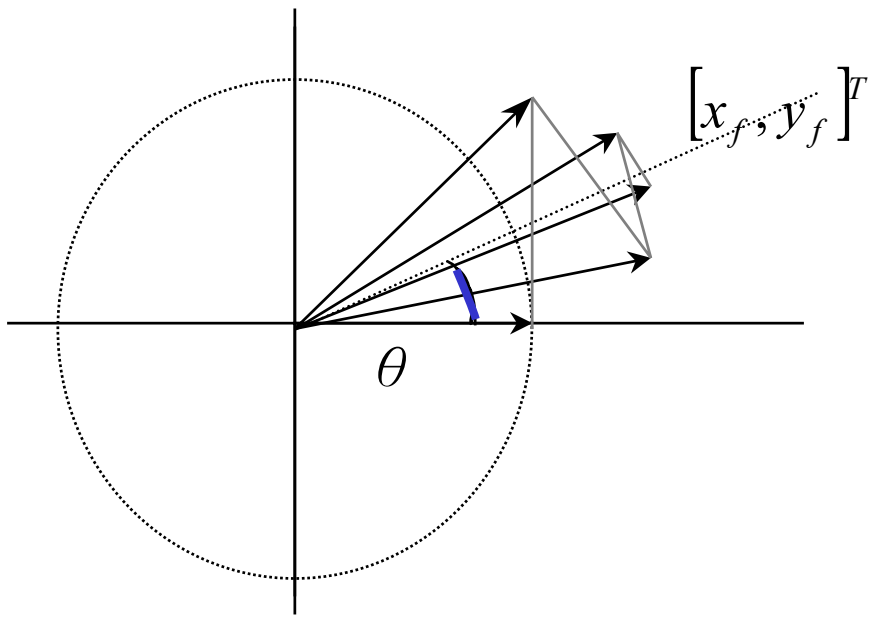
- Usually , we set z(0)= θ

$$z(0)-z(N)=\theta-z(N)=\sum_{i=0}^{N-1}\mu_1 a_m(i)$$

$|z(N)| \to 0$ after the N-th iteration

Rotational Sequence

μ_i= sign of z(i)



Usage of Vector Rotation Mode

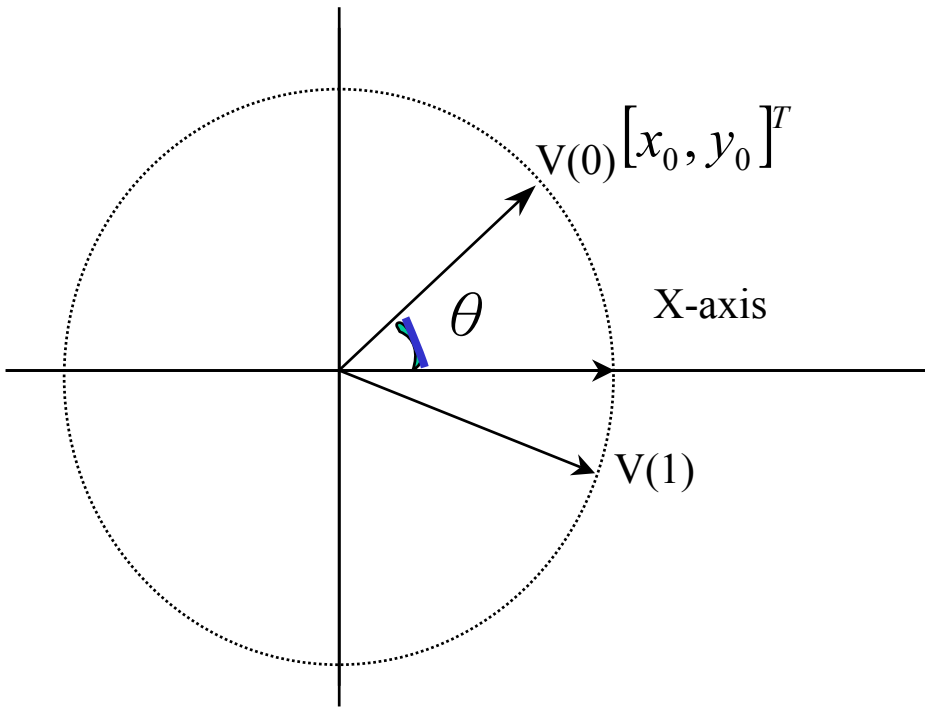
- In below situations, one may choose to evaluate angle by computing the **rotational sequence** in advance (off-line).
  - For many DSP problem, θ, is know in advance (e.g., twiddle factor)
  - The same θ will be used over and over.

- Store the corresponding set of μ_i, instead of θ, in the memory.
  - A particular advantage is that there will be no need to implement the angle updating formula resulting in a one-third cost saving of hardware.

$$Z_{i+1} = Z_i - \mu_i a_m(i)$$

Modes of Operations (cont'd)

- **Angle accumulation mode** (θ is not given)
  - ***Objective*** is to rotate the given initial vector $[x_0, y_0]^T$ back to x-axis, and the angle can be accrued.
  - Set z(0) = 0;
  - Rotational Sequence
  - $\mu i = -\text{sign}\ x(i)y(i)$



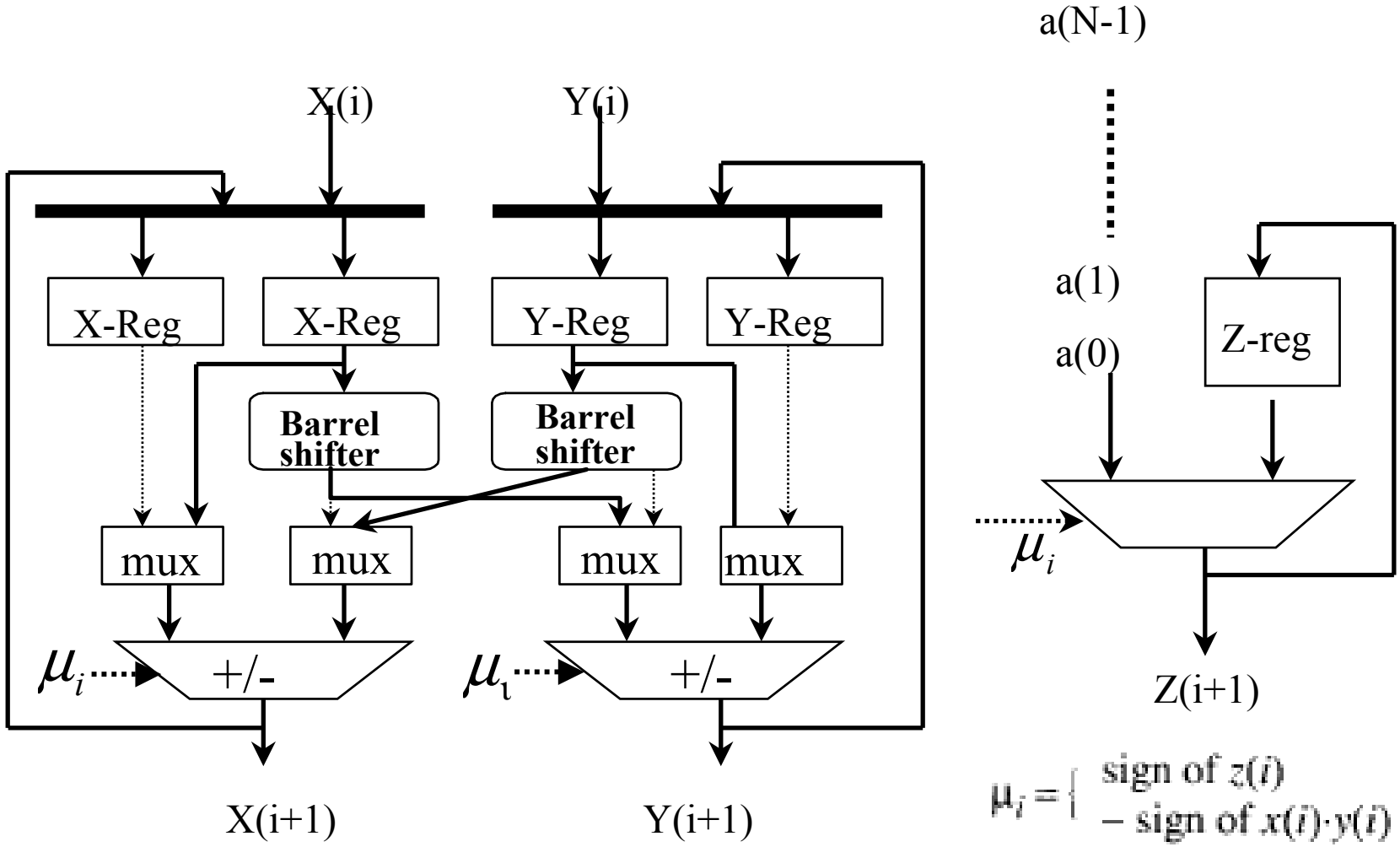
Rotational Sequence μ_i v.s. θ

- The selection criteria for μ_i are:

$$\mu_i = \begin{cases} sign\ \ of\ \ z(i) & \text{vector rotation mode} \\ -sign\ \ of\ \ x(i) \bullet y(i) & \text{angle accumulation mode} \end{cases}$$

- Note that the set of $\{\mu_i; i = 0\ to\ n-1\}$ can be regard as an alternative representation of the rotation angle θ.
- Hence, even in the angle accumulation mode, often it is not necessary to explicitly compute θ

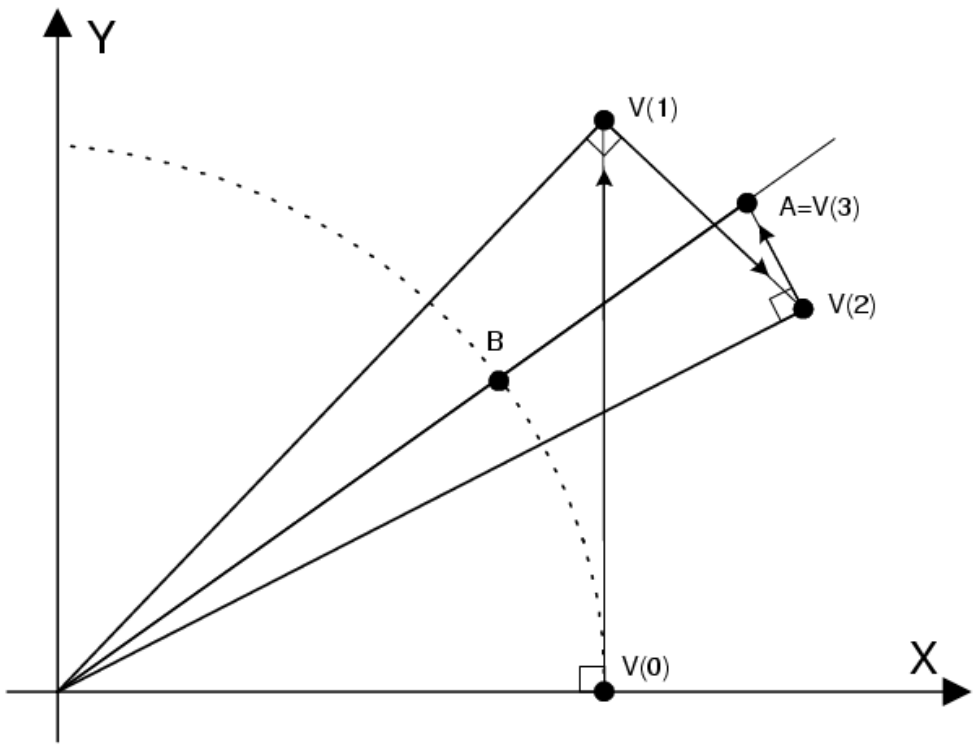
Basic processor for Micro-Rotation



Scaling Operation

- Scaling operation is used to maintain the "norm" of the original vector

$$P = \left(\prod_{i=0}^{R_m - 1} \sqrt{1 + 2^{-2s(i)}} \right)^{-1}.$$



Conventional Scaling Operation

Type I: $$\hat{P} = \sum_{j=0}^{R_s-1} k_j \cdot 2^{-q_j},$$

Type II: $$\hat{P} = \prod_{j=0}^{R_s-1} \left(1 + k_j \cdot 2^{-q_j}\right),$$

$$\xi_s \triangleq \left|1 - \frac{\hat{P}}{P}\right|.$$

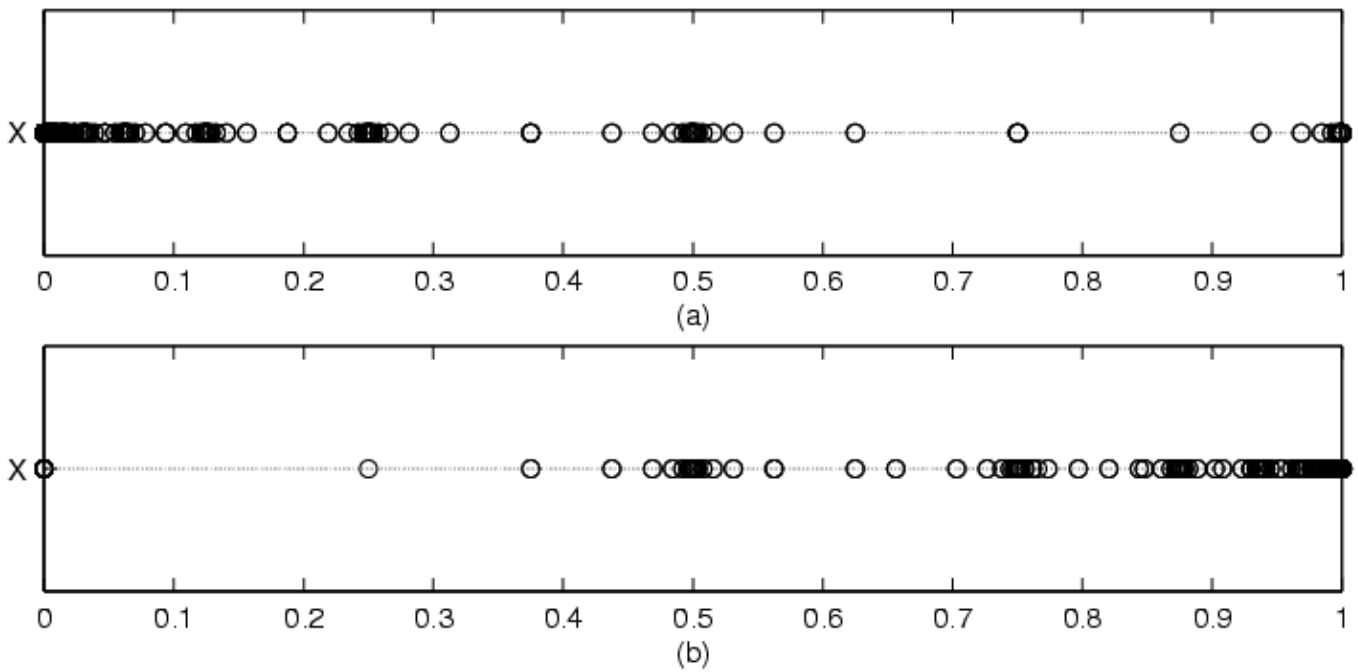
$$SQNR(dB) = 10 \cdot \log_{10}\left(\frac{1}{\xi_m^2 + \xi_s^2}\right).$$

Type I:
$$\begin{cases} \tilde{x}(j+1) = \tilde{x}(j) + k_j \cdot 2^{-q_j} \cdot x(R_m) \\ \tilde{y}(j+1) = \tilde{y}(j) + k_j \cdot 2^{-q_j} \cdot y(R_m) \end{cases}$$

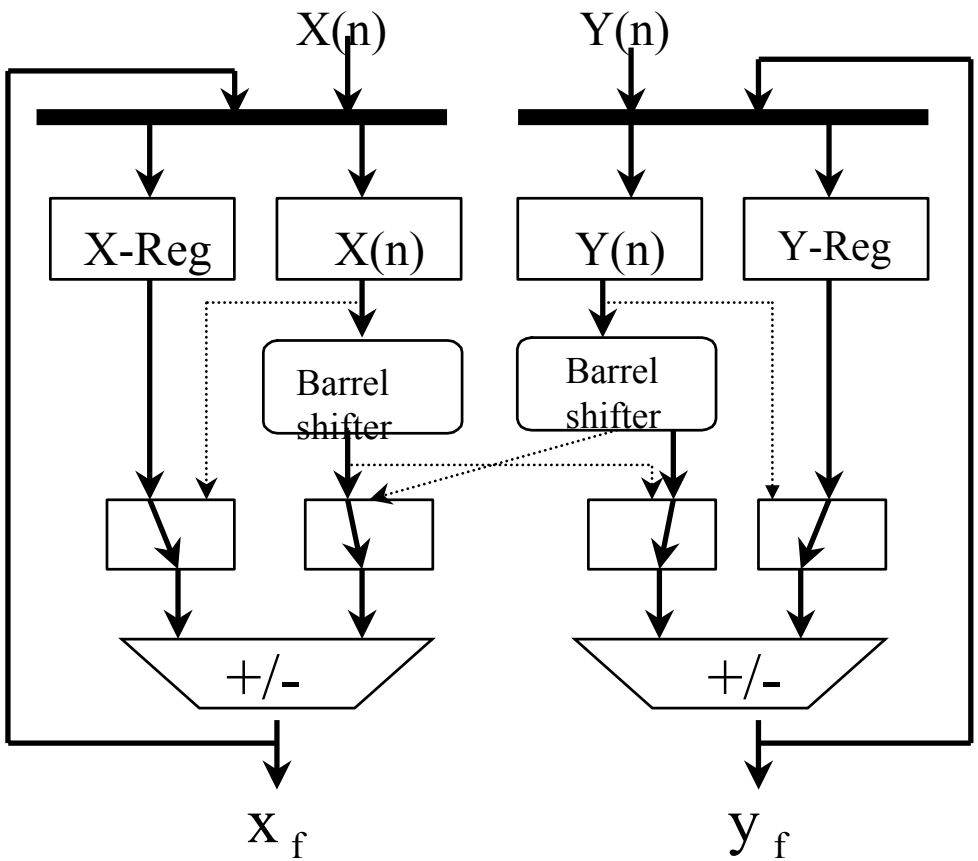
Type II:
$$\begin{cases} \tilde{x}(j+1) = \tilde{x}(j) + k_j \cdot 2^{-q_j} \cdot \tilde{x}(j) \\ \tilde{y}(j+1) = \tilde{y}(j) + k_j \cdot 2^{-q_j} \cdot \tilde{y}(j) \end{cases}$$

Selective Scaling Operation

- Combine Type-I and Type-II scaling operation by featuring
  - Complementary distribution
  - Off-line operation



Basic processor for Scaling Operation



$$Type\ 1:$$

$$x'(i+1) = x'(i) + 2^{-i_p}\, x(n)$$

$$y'(i+1) = y'(i) + 2^{-i_p}\, y(n)$$

$$Type\ 2:$$

$$x'(i+1) = x'(i) + 2^{-i_q}\, x'(n)$$

$$y'(i+1) = y'(i) + 2^{-i_q}\, y'(n)$$

Advantages and disadvantages

-Simple Shift-and-add Operation.

(2 adders+2 shifters v.s. 4 mul.+2 adder)

-Small area.

-It needs *n* iterations to obtain *n*-bit precision.

-Slow carry-propagate addition.

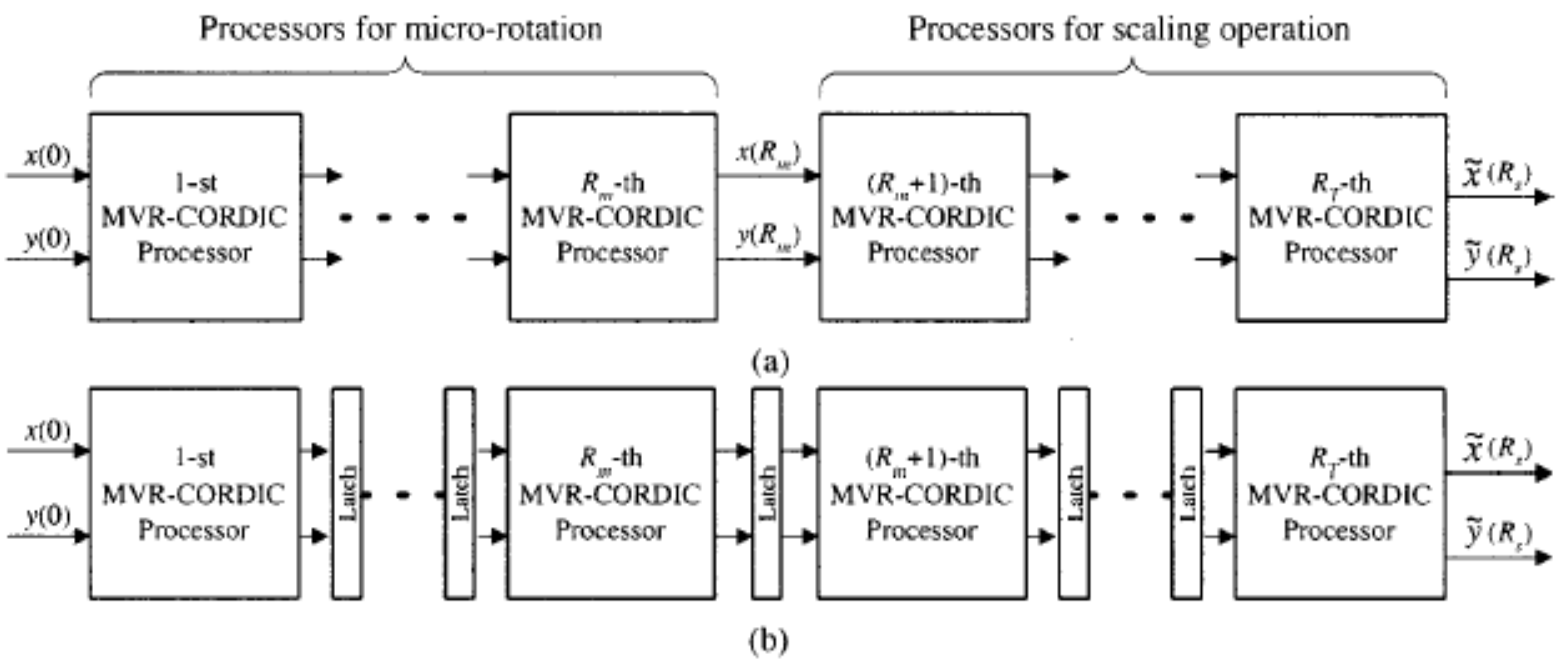
-Area consuming shifts (Barrel Shifter)

Enhancement of CORDIC

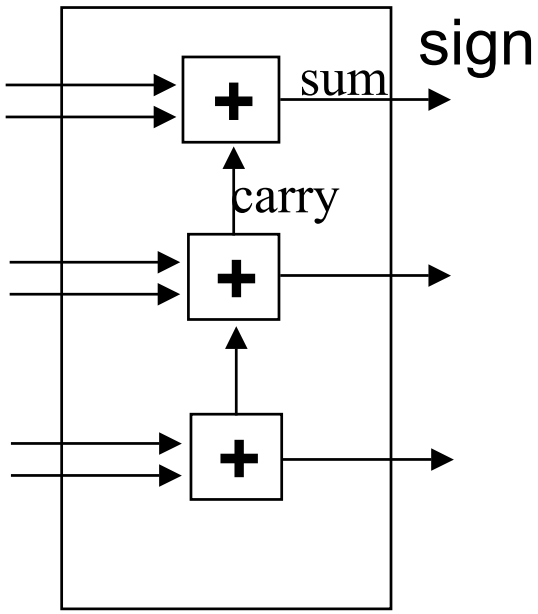
- Architecture
  - Pipelined Architecture
  - Faster Adder (CSA)
- Algorithm
  - Radix-4 CORDIC
  - ***MVR-CORDIC***
  - ***EEAS-CORDIC***

Pipelined architecture

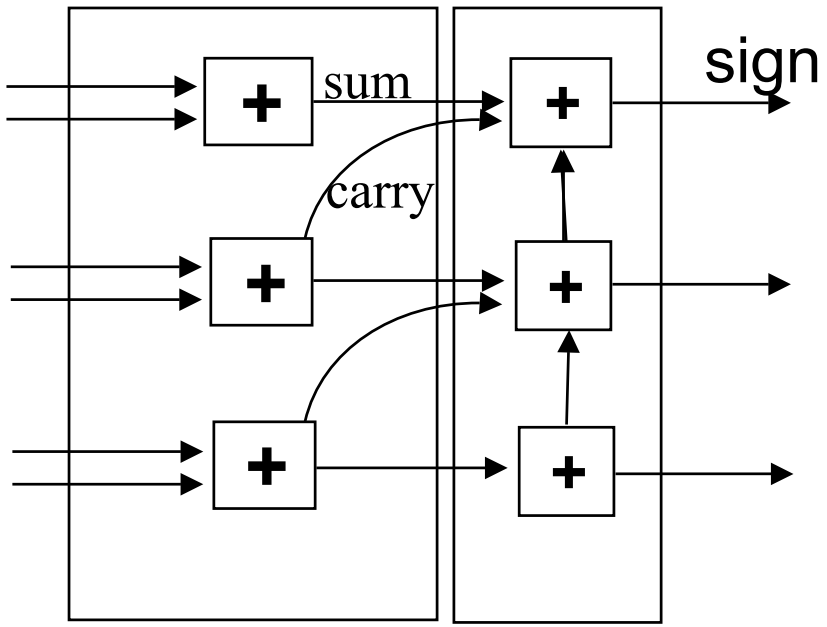
- Expand folded CORDIC processor to achieve the pipelined architecture
- Shifting can be realized by wiring



Faster Adder (CSA)



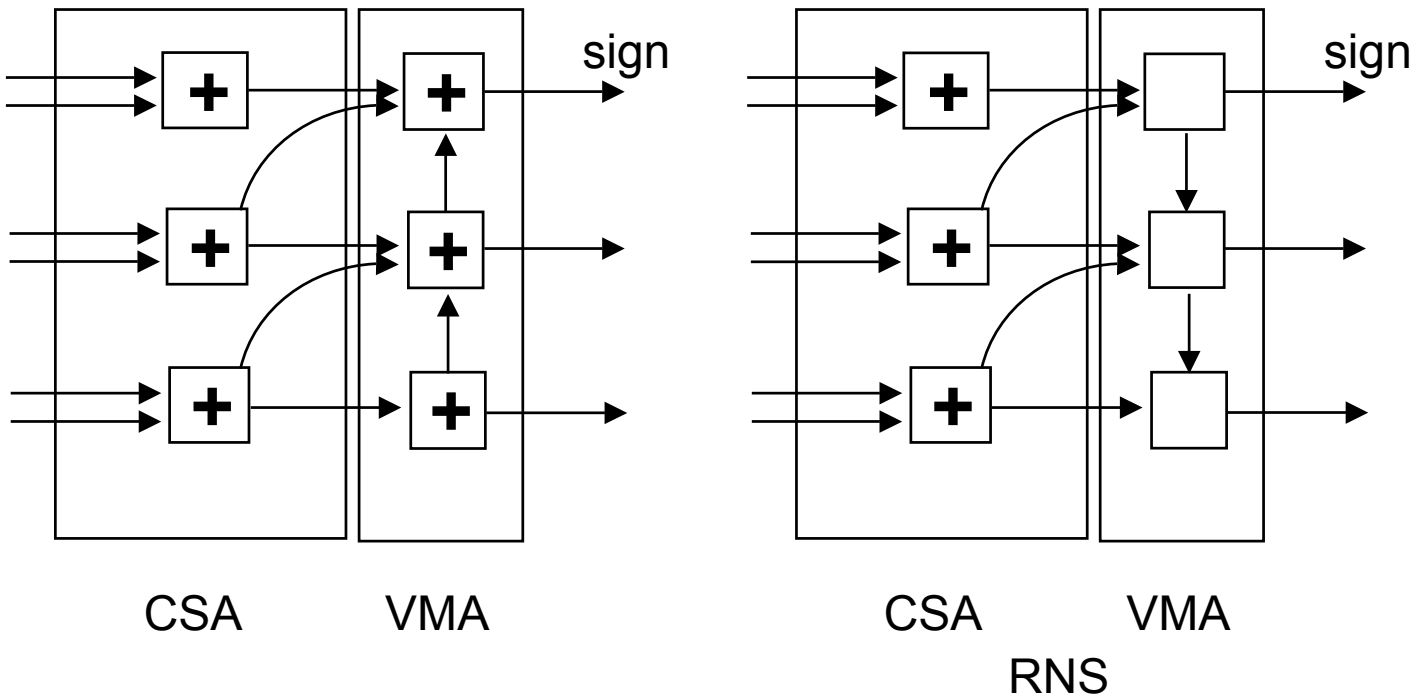
Ripple Adder
and its sign calculation



CSA VMA
and its sign calculation

Critical Path of CSA

- In on-line approach, we want to get sign bit as soon as possible !
- *Redundant Number System* to be used to obtain the sign bit quickly



Radix-4 CORDIC

- Reduce Iteration Numbers
  - High radix CORDIC.(e.g. Radix-4, Radix-8)
- 1 stage of Radix-4 = two stages of Radix-2
  - Faster computation at higher cost
- Employ the Radix-4 micro-rotations to
  - Reduce the stage number.
- But K_m may not be constant.

$$K_m = \prod_{i=0}^{n-1} \sqrt{1 + m\mu_i^2 4^{-2s(m,i)}} \qquad \mu = \{0, \pm 1, \pm 2\}$$

Outline

- Introduction
- Conventional CORDIC Algorithm
- Enhancement of CORDIC
  - ✓ Modified Vector Rotation (MVR)-CORDIC Algorithm
  - EEAS-Based CORDIC Algorithm
- Vector Rotational CORDIC Family

Modification of CORDIC Algorithm

- Skip some micro-rotation angles
  - For certain angles, we can only only reduce the iteration number but also improve the error performance.
  - For example, $\theta=\pi/4$

Conventional CORDIC

$$\bar{\mu} = [1, \quad -1, \quad 1, \quad 1, \quad 1, \quad \cdots]$$

$$\Rightarrow \quad \xi_m = 7.2*10^{-3}$$

MVR-CORDIC

$$\bar{\mu} = [1, \quad 0, \quad 0, \quad 0, \quad 0, \quad \cdots]$$

$$\Rightarrow \quad \xi_m = 0$$

Modification of CORDIC Algorithm

- Repeat some micro-rotation angles
  - Each micro-rotation angle can be performed repeatedly
  - For example, $\theta = \pi/2$: execute the micro-rotation of a(0) twice

- Confine the number of micro-rotations to R_m
  - In conventional CORDIC, number of iteration=W
  - In the MVR-CORDIC, $R_m << W$
  - Hardware/timing alignment

Modification of CORDIC Algorithm

- With above three modification

$$\theta = \sum_{i=0}^{R_m - 1} \alpha(i)\theta\left(s(i)\right) + \xi_m,$$

where

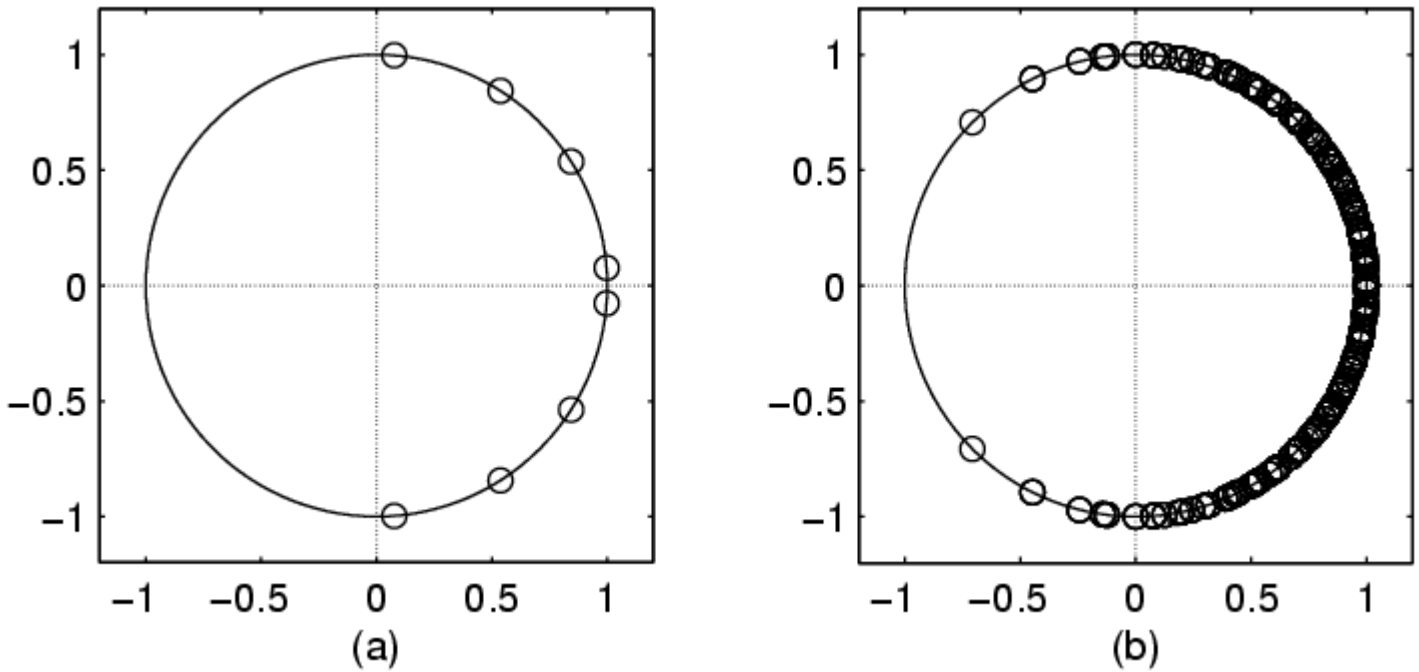
- s(i) ∈ {0, 1, 2, …, W} is the rotational sequence that determines the micro-rotation angle in the i^{th} iteration
- α(i) ∈ {-1, 0 ,1} is the directional sequence that controls the direction of the i^{th} micro-rotation

Elementary Angle Set of Conventional CORDIC

Iteration Index	Elementary Angle	Value in Radius
$i = 0$	$a(0) = \tan^{-1}(2^{-0})$	0.785398
$i = 1$	$a(1) = \tan^{-1}(2^{-1})$	0.463648
$i = 2$	$a(2) = \tan^{-1}(2^{-2})$	0.244979
$i = 3$	$a(3) = \tan^{-1}(2^{-3})$	0.124355
$i = 4$	$a(4) = \tan^{-1}(2^{-4})$	0.062419
$i = 5$	$a(5) = \tan^{-1}(2^{-5})$	0.031240
$i = 6$	$a(6) = \tan^{-1}(2^{-6})$	0.015624
$i = 7$	$a(7) = \tan^{-1}(2^{-7})$	0.007812

Conventional CORDIC with N=W=8

Constellation of Reachable Angles



(a) Conventional CORDIC with N=W=4
(b) MVR-CORDIC with W=4 and R_m=3

Summary of MVR-CORDIC Algorithm

- Both micro-rotation and scaling phases

% Initialization

Given $x(0)$ and $y(0)$

% Micro-rotation phase

FOR $i = 0$ to $R_m - 1$

$$\begin{bmatrix} x(i+1) \\ y(i+1) \end{bmatrix} = \begin{bmatrix} 1 & -\alpha(i)2^{-s(i)} \\ \alpha(i)2^{-s(i)} & 1 \end{bmatrix} \begin{bmatrix} x(i) \\ y(i) \end{bmatrix}$$

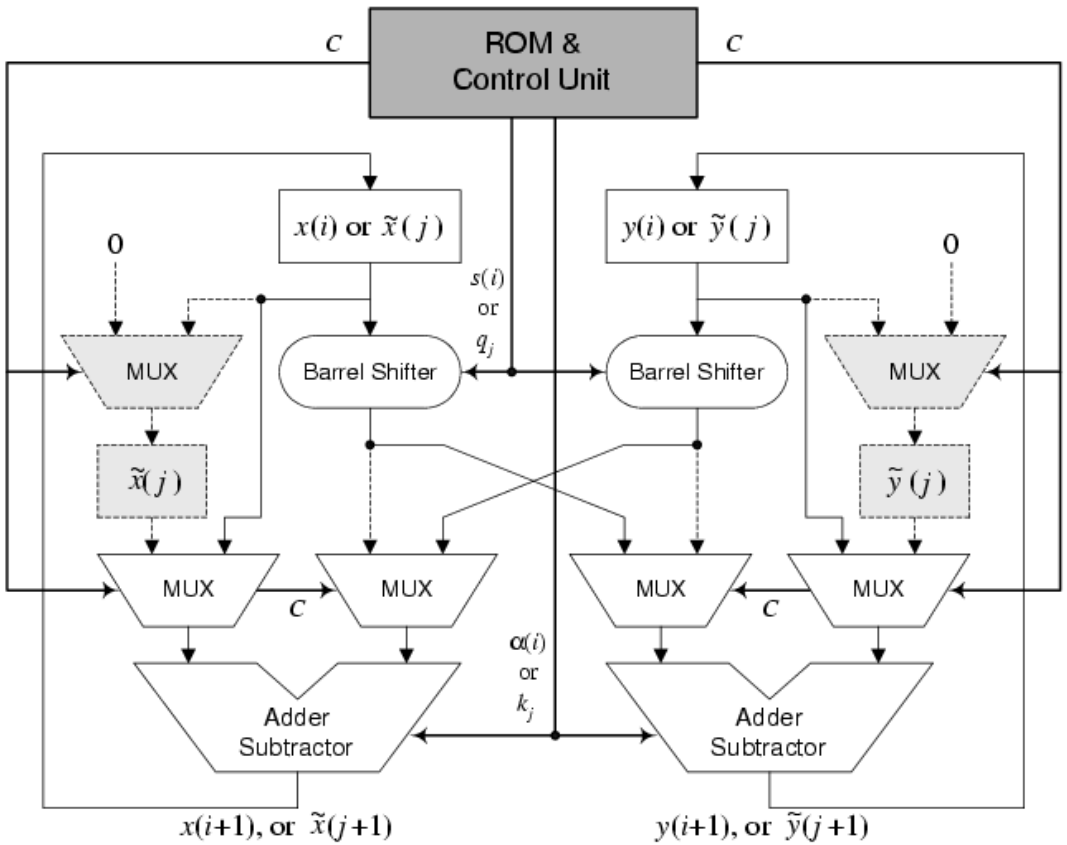
END

% Scaling phase

$$\begin{bmatrix} x_f \\ y_f \end{bmatrix} = P \begin{bmatrix} x(R_m) \\ y(R_m) \end{bmatrix} = \frac{1}{\prod_{i=0}^{R_m-1} \sqrt{1+2^{-2s(i)}}} \begin{bmatrix} x(R_m) \\ y(R_m) \end{bmatrix}$$

VLSI Architecture

- Iterative structure



Outline

- Introduction
- Conventional CORDIC Algorithm
- Enhancement of CORDIC
  - MVR-CORDIC Algorithm
  - ✓ EEAS-Based CORDIC Algorithm
- Vector Rotational CORDIC Family

Extended Elementary Angle Set (EEAS)

- Apply relaxation on EAS of

$$\mathcal{S}_1 = \left\{ \tan^{-1}\left(\alpha^\star \cdot 2^{-s^\star}\right) : \alpha^\star \in \{-1, 0, 1\},\ s^\star \in \{0, 1, \cdots, N-1\} \right\}.$$

- EAS is comprised of arctangent of single singed-power-of-two (SPT) term
- Effective way to extend the EAS is to employ more SPT terms

$$\mathcal{S}_2 = \left\{ \tan^{-1}\left(\alpha_0^\star \cdot 2^{-s_0^\star} + \alpha_1^\star \cdot 2^{-s_1^\star}\right) : \right.$$
$$\left. \alpha_0^\star, \alpha_1^\star \in \{-1, 0, 1\},\ s_0^\star, s_1^\star \in \{0, 1, \cdots, W-1\} \right\}.$$

Example of EAS and EEAS

Elementary Angle	Value in Radius	Elementary Angle	Value in Radius
$r(1) = \tan^{-1}(-2^{-0})$	-0.785398	$r(5) = \tan^{-1}(2^{-2})$	0.244979
$r(2) = \tan^{-1}(-2^{-1})$	-0.463648	$r(6) = \tan^{-1}(2^{-1})$	0.463648
$r(3) = \tan^{-1}(-2^{-2})$	-0.244977	$r(7) = \tan^{-1}(2^{-0})$	0.785398
$r(4) = \tan^{-1}(0)$	0.0		

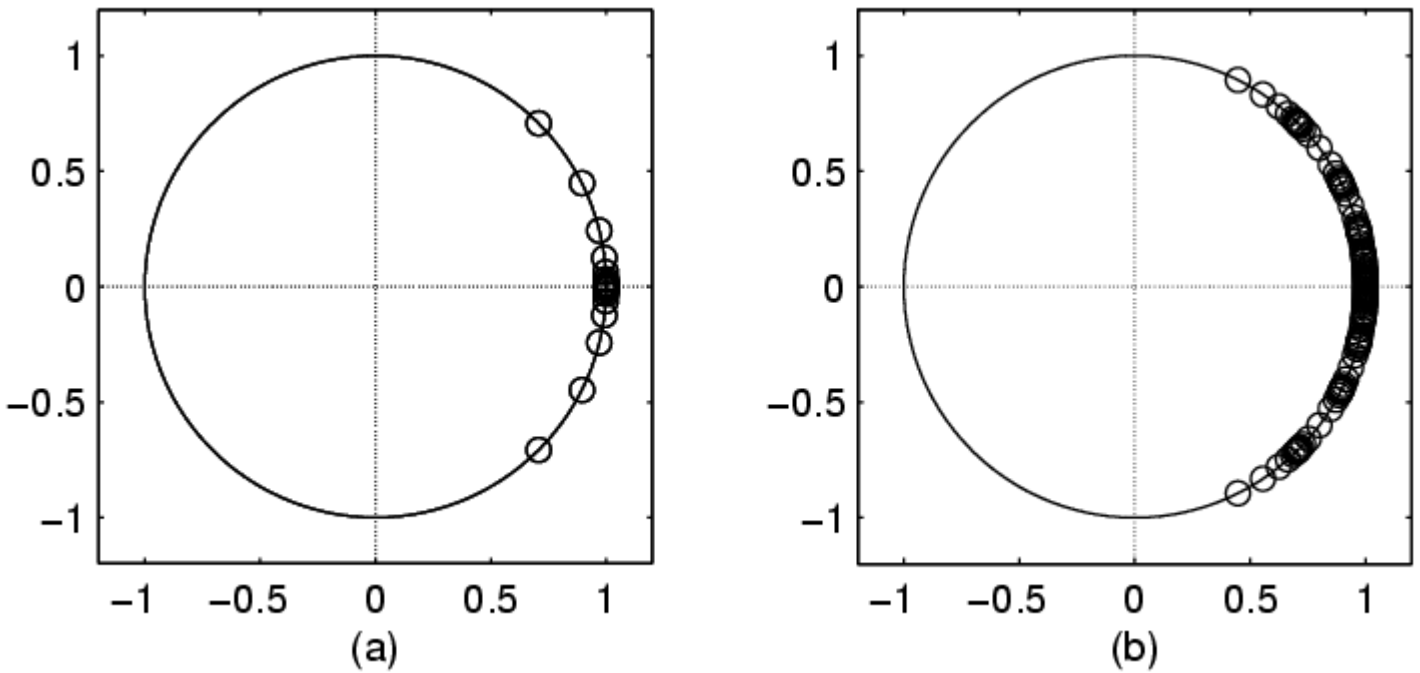
(a)

Elementary Angle	Value in Radius	Elementary Angle	Value in Radius
$r(1) = \tan^{-1}(-2^{-0} - 2^{-0})$	-1.107149	$r(9) = \tan^{-1}(2^{-2})$	0.244979
$r(2) = \tan^{-1}(-2^{-0} - 2^{-1})$	-0.982793	$r(10) = \tan^{-1}(2^{-1})$	0.463648
$r(3) = \tan^{-1}(-2^{-0} - 2^{-2})$	-0.896055	$r(11) = \tan^{-1}(2^{-1} + 2^{-2})$	0.643501
$r(4) = \tan^{-1}(-2^{-0})$	-0.785398	$r(12) = \tan^{-1}(2^{-0})$	0.785398
$r(5) = \tan^{-1}(-2^{-1} - 2^{-2})$	-0.643501	$r(13) = \tan^{-1}(2^{-0} + 2^{-2})$	0.896055
$r(6) = \tan^{-1}(-2^{-1})$	-0.463647	$r(14) = \tan^{-1}(2^{-0} + 2^{-1})$	0.982793
$r(7) = \tan^{-1}(-2^{-2})$	-0.244979	$r(15) = \tan^{-1}(2^{-0} + 2^{-0})$	1.107149
$r(8) = \tan^{-1}(0)$	0.0		

(b)

Example of elementary angles of (a) EAS S_1, (b) EEAS S_2, with wordlength W=3.

Constellation of EEAS



Constellation of elementary angles of (a) EAS S_1, (b) EEAS S_2, with wordlength W=8.

Summary of EEAS Scheme

- Both micro-rotation and scaling phases

% Initialization

Given initial vector of $[x(0), y(0)]^T$

% Micro-rotation phase

FOR $j = 0$ to $R_m - 1$

$$\begin{bmatrix} x(j+1) \\ y(j+1) \end{bmatrix} = \begin{bmatrix} 1 & -\alpha_0(j) \cdot 2^{-s_0(j)} - \alpha_1(j) \cdot 2^{-s_1(j)} \\ \alpha_0(j) \cdot 2^{-s_0(j)} + \alpha_1(j) \cdot 2^{-s_1(j)} & 1 \end{bmatrix} \begin{bmatrix} x(j) \\ y(j) \end{bmatrix}$$

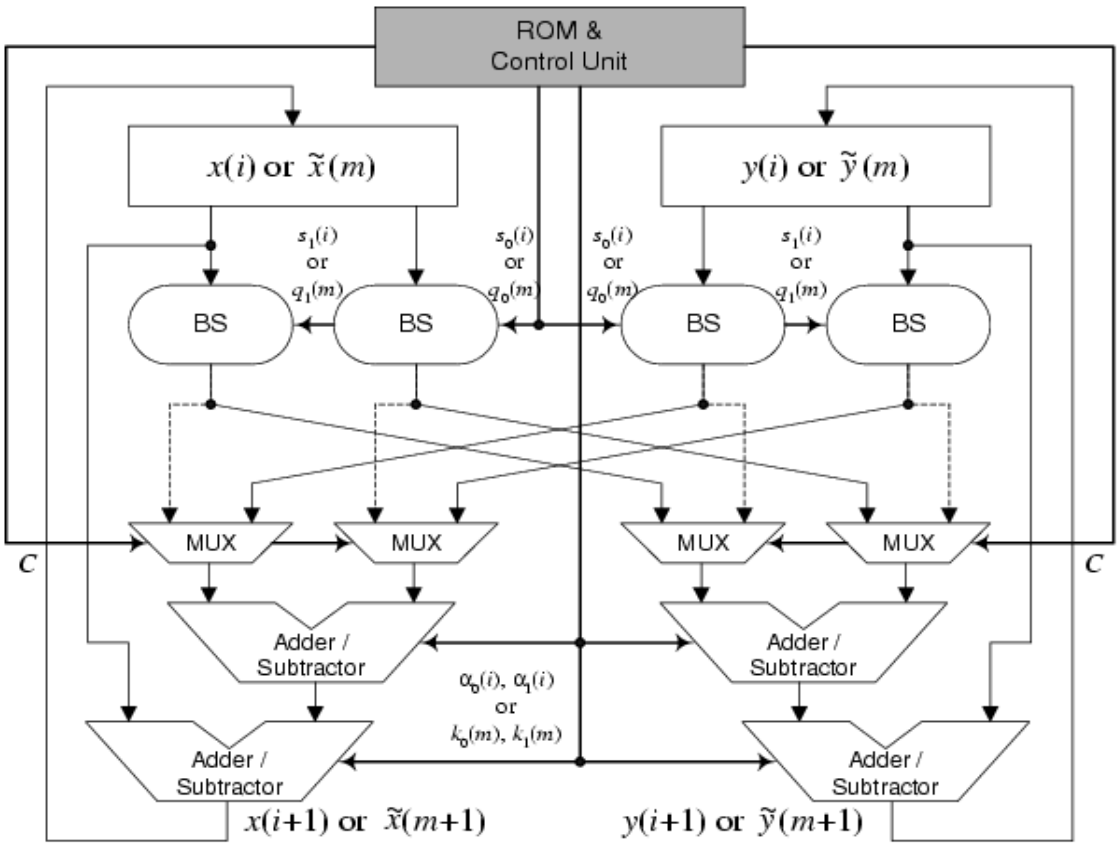
END

% Scaling phase

$$\begin{bmatrix} x_f \\ y_f \end{bmatrix} = P \begin{bmatrix} x(R_m) \\ y(R_m) \end{bmatrix} = \frac{1}{\prod_{j=0}^{R_s-1} \sqrt{1 + \left[\alpha_0(j) \cdot 2^{-s_0(j)} + \alpha_1(j) \cdot 2^{-s_1(j)}\right]^2}} \begin{bmatrix} x(R_m) \\ y(R_m) \end{bmatrix}$$

VLSI Architecture

- Iterative structure for one-stage rotational operation



Comparison of Rotation Schemes

	Hardware Requirement	Full Adder (FA) Count	SQNR Performance
Direct Implementation	4 Multipliers, 2 Adders	1,056	98.7dB
Conventional CORDIC Algorithm	About 43 Adders	688	97.4dB
Angle Recoding (AR) Technique (R_m=6, R_s=6)	24 Adders	384	93.3dB
EEAS-based CORDIC Algorithm (R_m=2, R_s=2)	16 Adders	256	95.1dB

Comparison of existing approaches/algorithms performing vector rotation in 2D plane, where the wordlength, *W*, is 16.

Outline

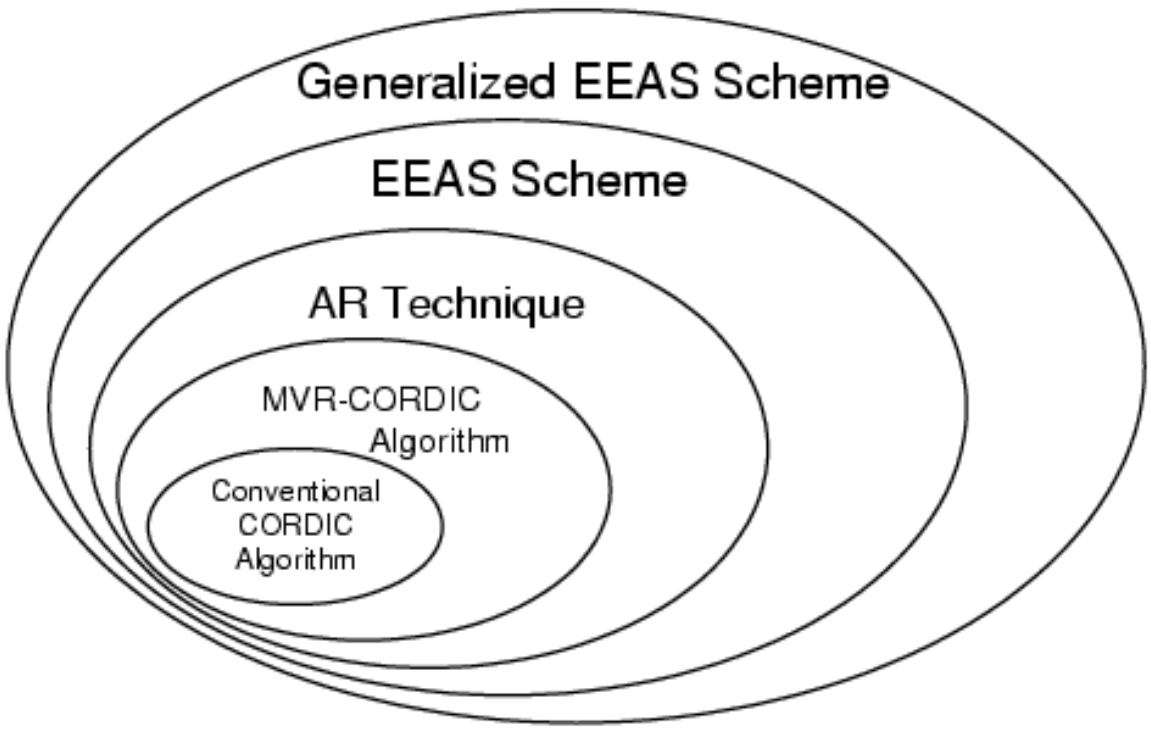
- Introduction
- Conventional CORDIC Algorithm
- Enhancement of CORDIC
  - MVR-CORDIC Algorithm
  - EEAS-Based CORDIC Algorithm
- ✓ Vector Rotational CORDIC Family

Family of Vector Rotational CORDIC Algorithm

Vector Rotation Algorithms	Rotation Sequence	Elementary Angle Set	Micro-rotations	Angle Quantization	
				θ_i	N_A
CORDIC Algorithm	$\mu \in \{-1,\ 1\}$	$EAS\ \mathcal{S}$	Complete	$\theta_i = \mu(i)a(i)$	W Fixed
Angle Recoding Technique	$\mu \in \{-1,\ 0,\ 1\}$	$EAS\ \mathcal{S}_1$	Selective	$\theta_i = \tan^{-1}\left(\alpha(i) \cdot 2^{-2s(i)}\right)$	N' Variable
MVR-CORDIC Algorithm	$\alpha \in \{-1,\ 0,\ 1\}$	$EAS\ \mathcal{S}_1$	Selective	$\theta_i = \tan^{-1}\left(\alpha(i) \cdot 2^{-2s(i)}\right)$	R_m Fixed
EEAS-based CORDIC Alg.	$\alpha_0,\ \alpha_1$ $\in \{-1,\ 0,\ 1\}$	$EEAS\ \mathcal{S}_2$	Selective	$\theta_i = \tan^{-1}\left(\alpha_0(i) \cdot 2^{-2s_0(i)} + \alpha_1(i) \cdot 2^{-2s_1(i)}\right)$	R_m Fixed
Generalized EEAS Alg.	$\alpha_0, \alpha_1, \cdots, \alpha_{d-1}$ $\in \{-1,\ 0,\ 1\}$	$EEAS\ \mathcal{S}_d$ $d \geq 3$	Selective	$\theta_i = \tan^{-1}\left(\alpha_0(i) \cdot 2^{-2s_0(i)} \cdots + \alpha_{d-1}(i) \cdot 2^{-2s_{d-1}(i)}\right)$	R_m Fixed

Set Diagram of VR CORDIC Family

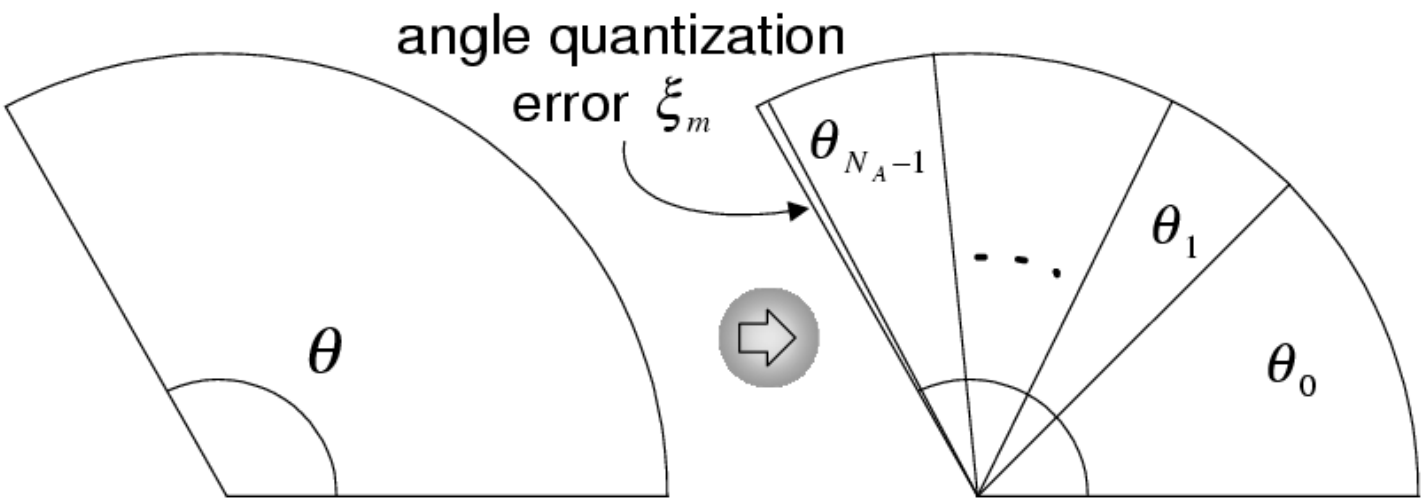
- Relationship among members in Vector Rotational (VR) CORDIC family can be represented as



Angle Quantization

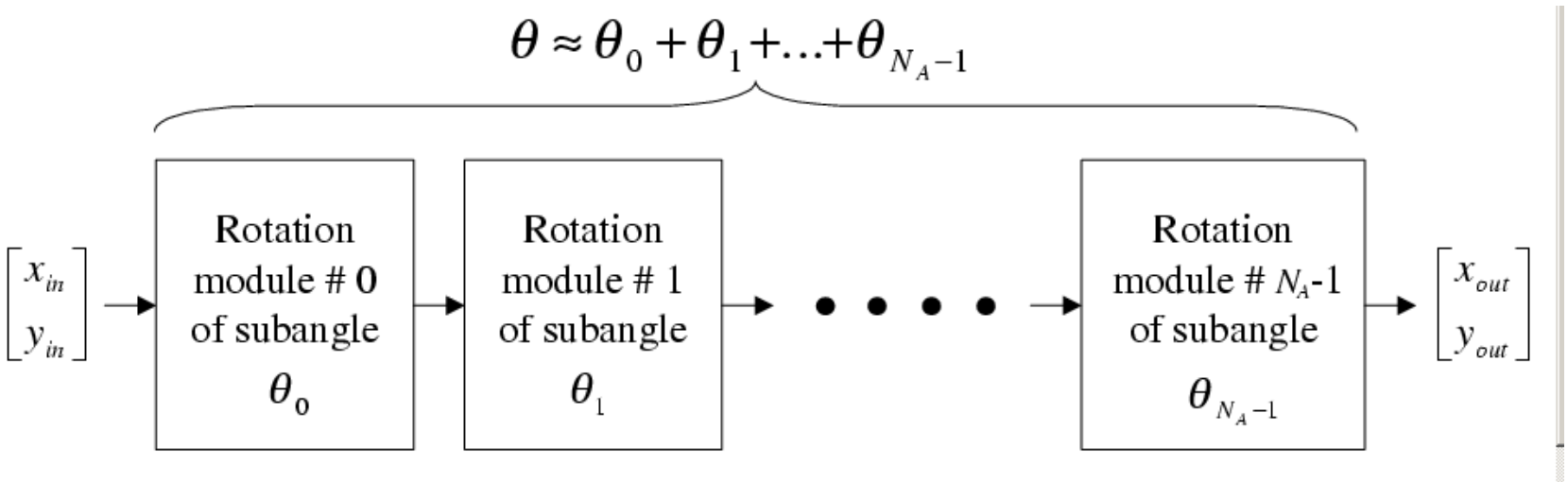
- Approximation is applied on “ANGLE”
- Decompose target angle into sub-angles

$$\xi_m \triangleq \theta - \sum_{i=0}^{N_A-1} \theta_i,$$



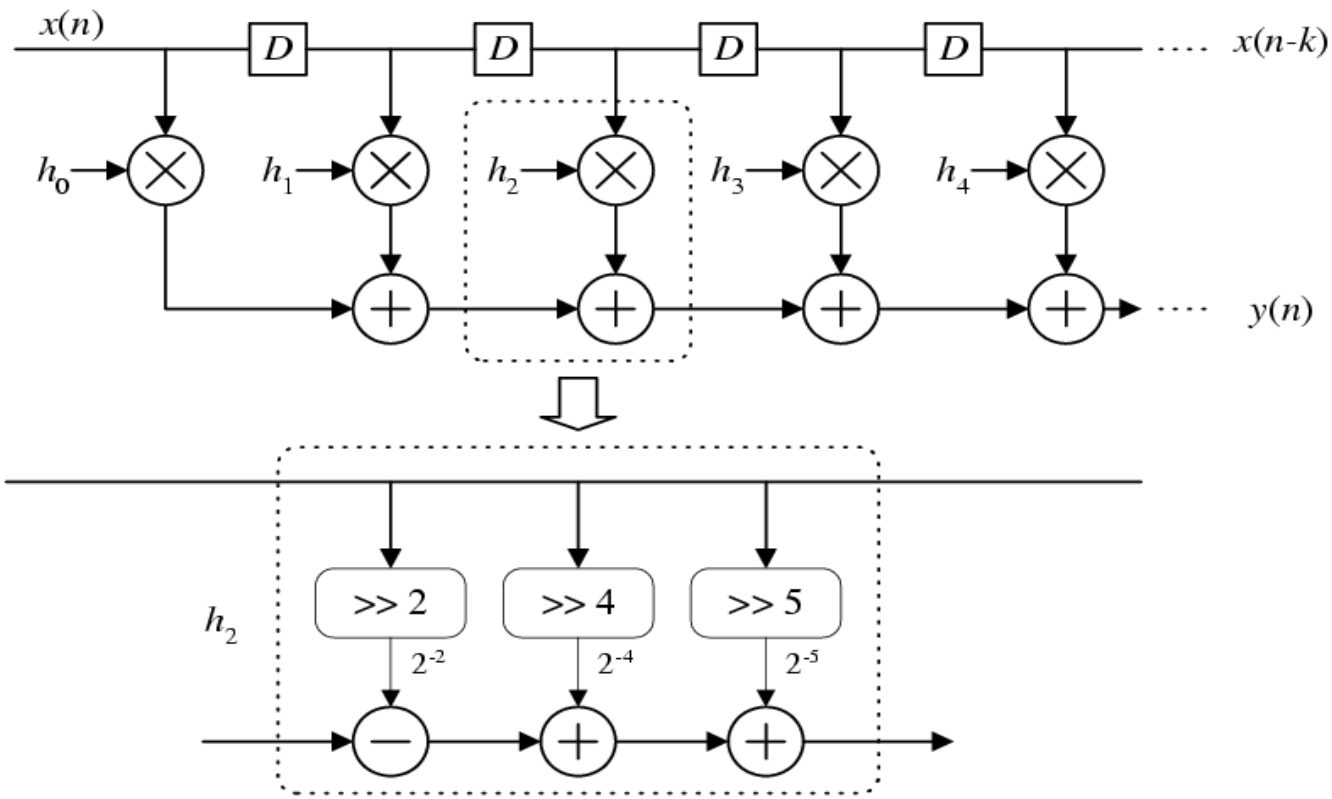
Angle Quantization (Cont.)

- Two key design issues in AQ process
  1) Determine (construct) the sub-angle, and each sub-angle needs to be easy-to-implementation in practical implementation.
  2) How to combine sub-angles such that angle quantization can be minimized.



Angle Quantization (Cont.)

- AQ process is conceptually similar to Sum-of-Power-ot Two (SPT) quantization



Angle Quantization (Cont.)

- Correspondences between AQ process and SPT (CSD) quantization process

	Canonical Signed Digit (CSD)	Angle Quantization (AQ)
Target of Approximation	Coefficient, h_k	Rotation angle, θ
Basic element	Non-zero digit, 2^{-i}	Sub-angle, θ_i
Basic operation	Shift-and-add operation	Shift-and-add operation
Approximation equation	$h_k \approx \sum_{j=0}^{N_D-1} g_j \cdot 2^{-d_j}$	$\theta \approx \sum_{j=0}^{N_A-1} \theta_j$

Conclusion

- Rotational Engine plays an important role in modern DSP systems
- The conventional CORDIC is an interesting idea to reduce VLSI hardware cost in implementing the rotation operation (as well as other arithmetic operations).
- The idea of Angle Quantization is firstly introduced in Ref. [4,5]. It is analogous to the SPT concept in quantizing floating-point numbers.
- Most CORDIC-related algorithms can be explained using the AQ design framework

Reference

1. *Y.H. Hu,, "*CORDIC-based VLSI architectures for digital signal processing", *IEEE Signal Processing Magazine* , Vol. 9, Issue: 3, pp. 16 -35, July 1992.
2. *E. Antelo, J. Villalba, J.D. Bruguera, and E.L. Zapata,"* High performance rotation architectures based on the radix-4 CORDIC algorithm", *IEEE Transactions on Computers*, Vol. 46, Issue: 8, Aug. 1997.
3. C.S Wu and A.Y. Wu, "Modified vector rotational CORDIC (MVR-CORDIC) algorithm and architecture", *IEEE Trans. Circuits and systems-II: analog and digital signal processing*, vol. 48, No. 6, June 2001.
4. A.Y. Wu and C.S. Wu, "A unified design framework for vector rotational CORDIC family based on angle quantization process", *In Proc. of 2001 IEEE International Conference on Acoustics, Speech, and Signal Processing*, Vol. 2, pp. 1233 -1236.
5. A. Y. Wu and Cheng-Shing Wu, "A Unified View for Vector Rotational CORDIC Algorithms and Architectures based on Angle Quantization Approach," to appear in *IEEE Trans. Circuits and Systems Part-I: Fundamental Theory and Applications,* Oct. 2002.
6. C. S. Wu and A. Y. Wu, "A novel trellis-based searching scheme for EEAS-based CORDIC algorithm" In Proc. of 2001 IEEE International Conference on Acoustics, Speech, and Signal Processing, Vol. 2 , pp. 1229 -1232.