DO NOT SHARE SLIDES AND CLASS MATERIALS ON ONLINE SITES

Advanced Logic Design
Computer Arithmetic: CORDIC

Mingoo Seok
Columbia University

BV: no related section

CORDIC

- What is CORDIC ?
  - <u>CO</u>ordinate <u>R</u>otation <u>DI</u>gital <u>C</u>omputer

- Why do we use CORDIC ?
  - CORDIC can be used to calculate hyperbolic, trigonometric, exponential functions, logarithms, multiplications, divisions, and square root
  - While saving hardware cost
  - Particularly good when hardware multiplier is not available

CORDIC History

1959

The CORDIC Trigonometric Computing Technique*

JACK E. VOLDER†

1971

A unified algorithm for elementary functions

by J. S. WALTHER

Hewlett-Packard Company
Palo Alto, California

1977

Algorithms and Accuracy in the HP-35

A lot goes on in that little machine when it's computing a transcendental function.

By David S. Cochran

Personal Calculator Algorithms II: Trigonometric Functions

A detailed explanation of the algorithms used by HP hand-held calculators to compute sine, cosine, and tangent.

by William E. Egbert

Personal Calculator Algorithms III: Inverse Trigonometric Functions

A detailed description of the algorithms used in Hewlett-Packard hand-held calculators to compute arc sine, arc cosine, and arc tangent.

by William E. Egbert

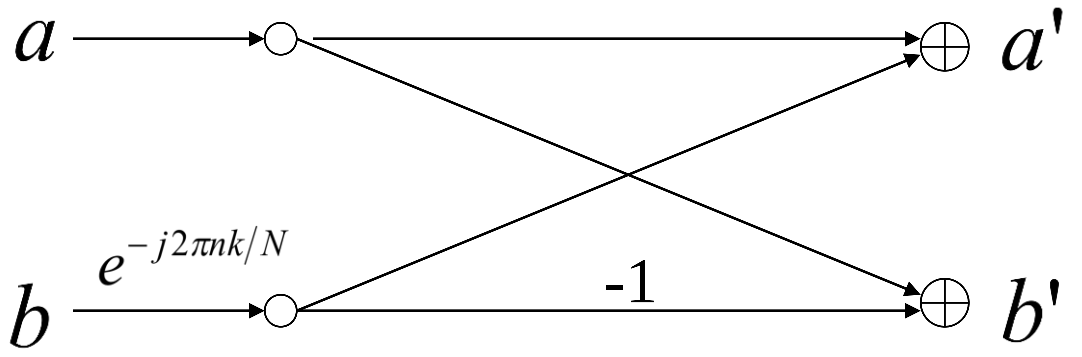
- Invented a while ago
- Initial application is a hand-held calculator where reducing silicon area is key for cost reduction

Rotation Operation

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} . \begin{bmatrix} x(i) \\ y(i) \end{bmatrix}$$

- Perform a rotation for an input vector by an angle

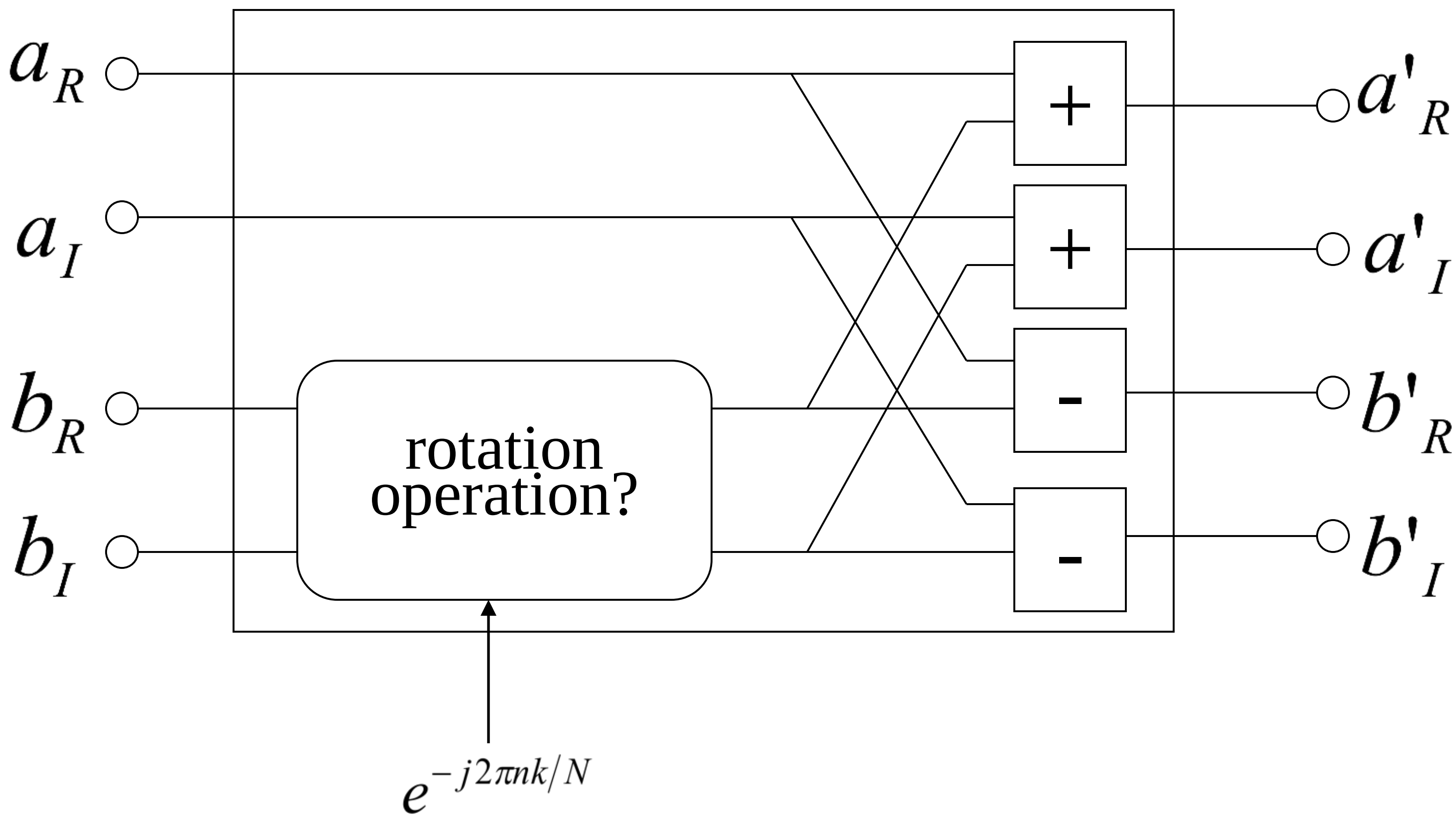
Butterfly in Fast Fourier Transform



$$a' = a + b \cdot e^{-j2\pi nk/N}$$

$$b' = a - b \cdot e^{-j2\pi nk/N}$$

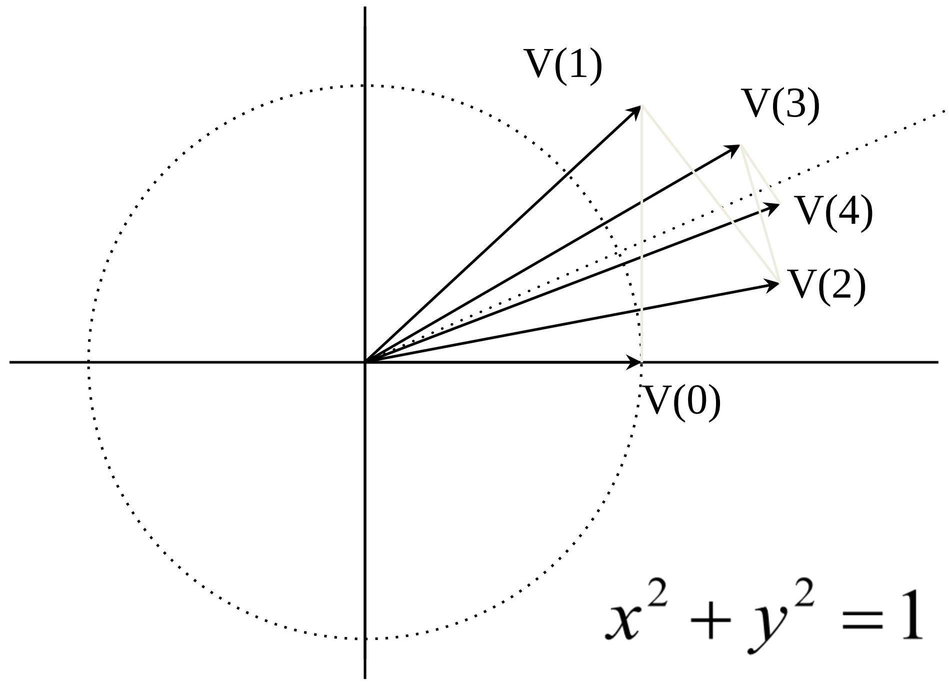
Complex-Value Butterfly



CORDIC Rotation Mode

- Steps
  - Decompose the desired rotation angle (θ) into the weighted sum of a set of predefined elementary rotation angles ($a_m(i)$)
  - Rotate through each of them
  - Each rotation is done by a simple shift-and-add operation.

Behavior of CORDIC



- Want to rotate the input vector V(0) by $e^{-j2\pi nk/N}$
- CORDIC algorithms can achieve this via iterations
  - V(0) → V(1) → V(2) → …

Single Iteration

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

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

$$\cos a_m = \frac{1}{\sqrt{1+\tan^2 a_m}} = K_m$$

CORDIC Algorithm

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

θ	Tan θ	K	i
45°	1	1	0
26.565°	2^{-1}	2	1
14.036°	2^{-2}	3	2
7.125°	2^{-3}	4	3
3.576°	2^{-4}	5	4
1.790°	2^{-5}	6	5
0.895°	2^{-6}	7	6

- Use only carefully selected θs such that tan(θ) is a power-of-2 number. This allows the most of the CORDIC operation involves only a shift-&-add operation
- μ_i is either +1 or -1, depending on the rotation direction

Scaling

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

$$K(n) = \prod_{i=0}^{n-1} K_i = \prod_{i=0}^{n-1} 1/\sqrt{1+2^{-2i}}$$

$$K = \lim_{n \to \infty} K(n) \approx 0.6072529350088812561694$$

- K_m values can be pre-calculated and stored in the lookup table
- If the iteration number is fixed, only a single K value is sufficient.

For $i=0$ to $n-1$,Do

/*CORDIC iteration equation */

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

/*Angle updating equation*/

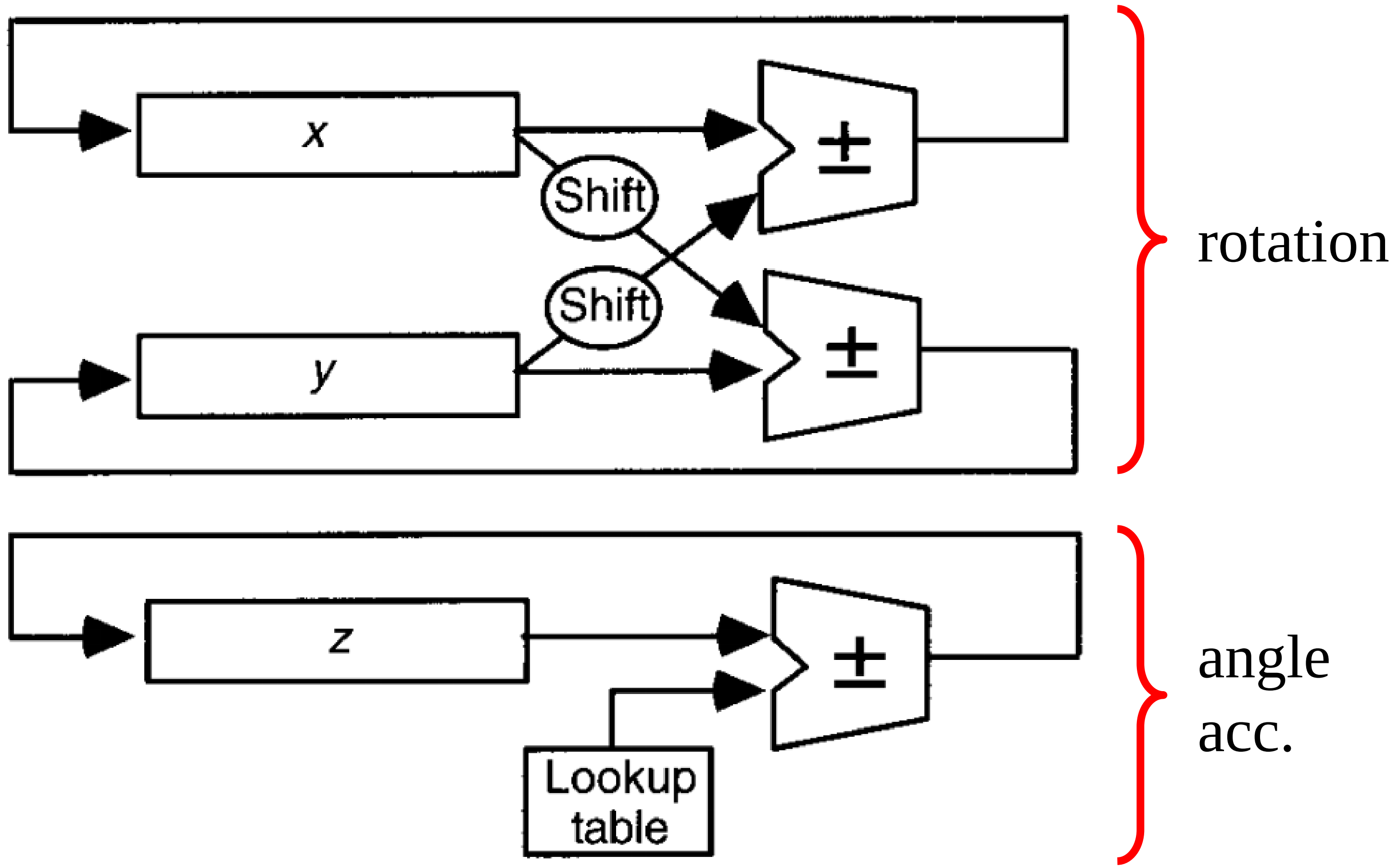
$$\mathbf{z(i+1) = z(i) - \mu_i a_m(i)}$$

End i loop

/*Scaling Operation (required for m=±1 only)*/

$$\begin{bmatrix} x_f \\ y_f \end{bmatrix} = \frac{1}{K_m(n)} \begin{bmatrix} x(n) \\ y(n) \end{bmatrix} \qquad K_m(n) = \frac{1}{\prod_{i=0}^{n-1} \cos a_m}$$

CORDIC HW Architecture



- Hardware complexity?

Summary

- Simple shift-and-add operation.
  - Save areas
  - 2 adders + 2 shifters v.s. 4 mul.+ 2 adders
- It needs *n* iterations to obtain *n*-bit precision.
  - Slow carry-propagate additions.
  - Low throughput rate