Introduction to the CORDIC algorithm

Signal Processing Systems Fall 2025

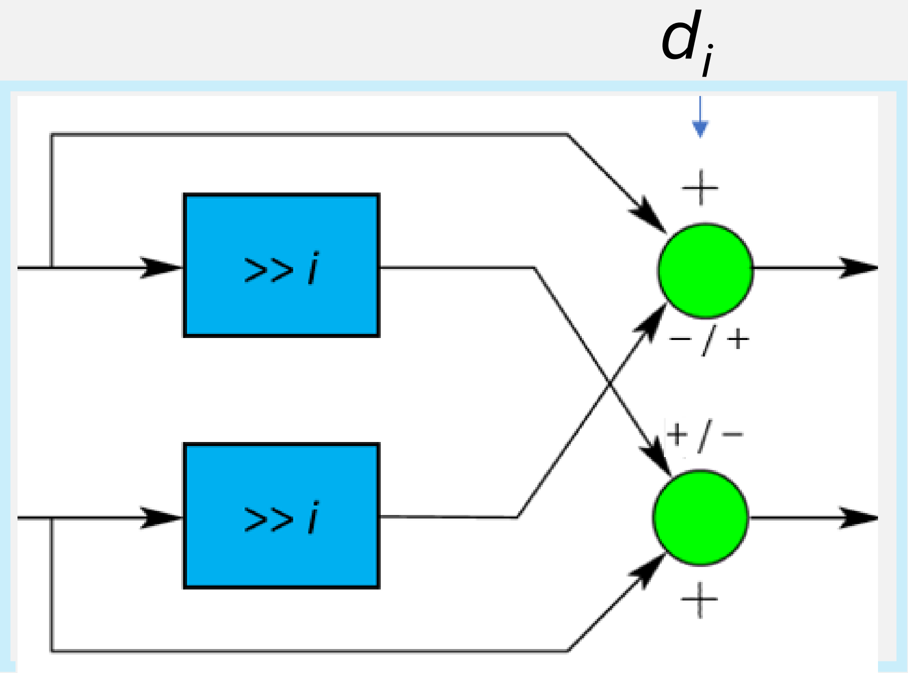
Lecture 6 (Thursday 13.11.)

Outline

- Givens transform
- From Givens transform to CORDIC structure
  - Derivation of the structure
  - Rotation control, angle range coverage, gain issue
  - Numeric example
- Polar transform with CORDIC
  - Principle, numeric example

CORDIC

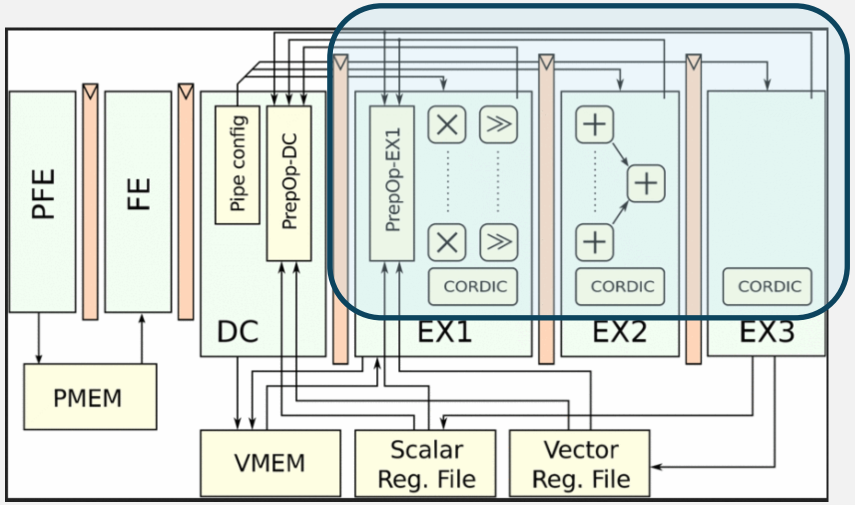
- COordinate Rotation DIgital Computer (Volder's algorithm)
- Jack E. Volder (1956) - real-time digital solution for aircraft navigation
- A shift-and-add algorithm
  - Commonly used when no HW multiplier is available
  - Requires only additions, subtractions, arithmetic shifts and possibly table lookup operations
  - In some cases, hardwired implementations possible
  - Essentially a fixed-point technique
- Computing the Givens transform is the basic application, but can be used for other purposes too (unified CORDIC)



Basic computation step in CORDIC: arithmetic shifts to right (i bits) are applied to inputs, followed by addition/subtraction operations to produce two outputs

From Lecture 3 slides

- Fixed-point SIMD core design for MIMO-OFDM detection: CORDIC blocks included in 3 pipeline stages



[Guenther et al. 2014]

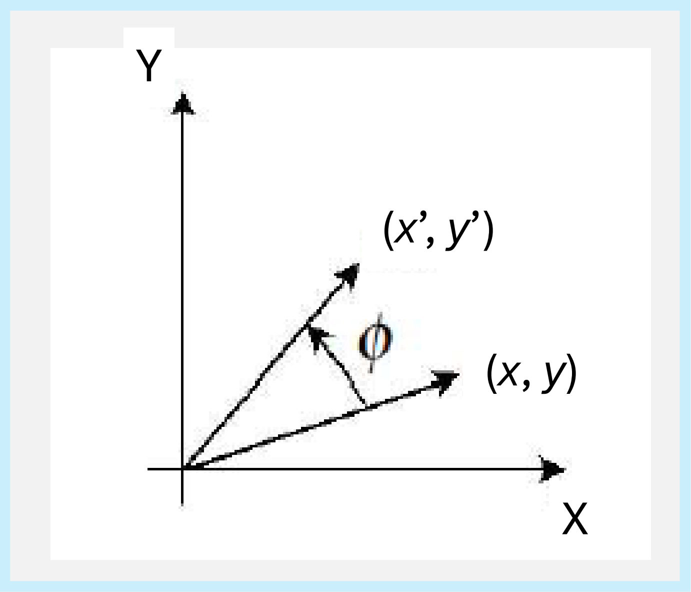
Givens transform

- Maps coordinates (x, y) to coordinates (x’, y’) according to

$$x' = x\cos\phi - y\sin\phi$$
$$y' = y\cos\phi + x\sin\phi$$

where ϕ is the rotation angle.

- This rotator is a common operation in signal processing algorithms.
  - Example. Calculating the product of two complex numbers
- GT can be computed with multipliers and adders
  - When ϕ is known beforehand, HW complexity is **4 multiplications and 2 additions** as $\cos\phi$ and $\sin\phi$ can be evaluated and stored to a lookup table
- Another approach: CORDIC
  - HW complexity of a CORDIC rotator is about **one multiplication**
  - In computation-intensive DSP applications, such a difference in complexity is significant

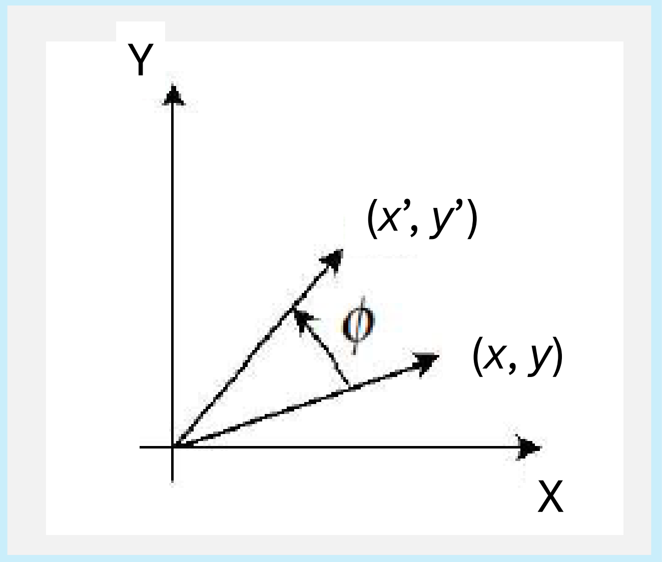


Multiplication of complex signal $x + iy$ by a complex coefficient $\cos\phi + i\sin\phi$
=> result is $x' + iy'$

CORDIC computes the transform **up to a scale factor A** (it computes ***A****x*’ and ***A****y*’). But, there are means to deal with this scaling or it may even be irrelevant.

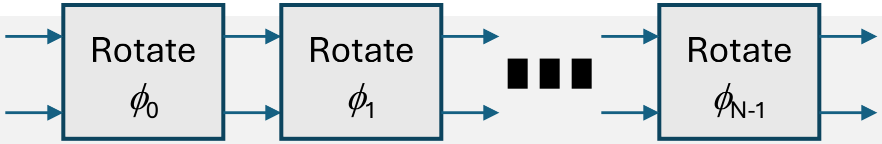
From Givens transform to CORDIC structure

- Let us see how we get it



STEP 1. Rotation by ϕ can be done by several elementary rotations.

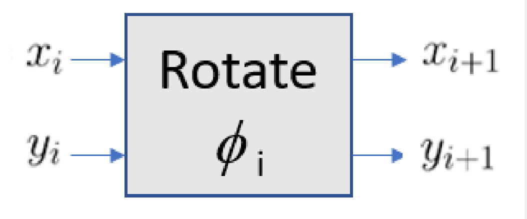
As a first step towards the CORDIC implementation, we note that if $\phi = \phi_a + \phi_b$, we may first map (x, y) to (x'', y'') using the angle ϕ_a, and then map (x'', y'') to (x', y') using the angle ϕ_b. So, it is possible to concatenate mappings for angles ϕ_i, $(i = 0, \ldots, N-1)$ in order to evaluate the mapping for $\phi = \sum_{i=0}^{N-1} \phi_i$.



STEP 2. Focusing to a single elementary rotation ϕ_i.

In the following, we will denote with (x_i, y_i) and (x_{i+1}, y_{i+1}) the input and output to the rotation by ϕ_i:

$$\begin{aligned} x_{i+1} &= x_i \cos\phi_i - y_i \sin\phi_i \\ y_{i+1} &= y_i \cos\phi_i + x_i \sin\phi_i \end{aligned} \tag{5}$$



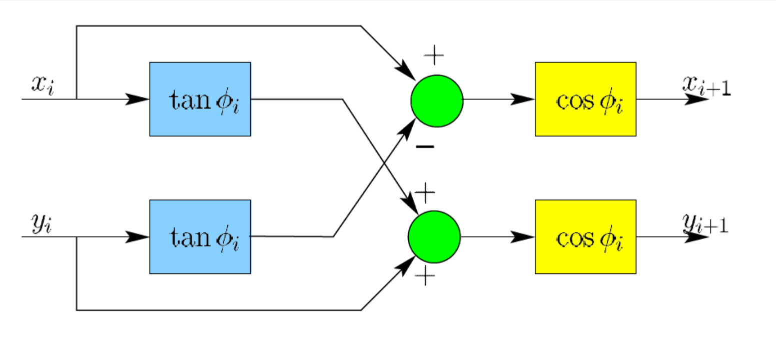
From Givens to CORDIC …

STEP 3. Rewriting the equations related to rotation by ϕ_i.

To proceed, let us assume that $-\pi/2 < \phi_i < \pi/2$. Using $\tan\phi = \sin\phi/\cos\phi$, equation (5) can be rewritten as

$$x_{i+1} = x_i \cos\phi_i - y_i \sin\phi_i$$
$$y_{i+1} = y_i \cos\phi_i + x_i \sin\phi_i$$

$$x_{i+1} = \cos\phi_i(x_i - y_i \tan\phi_i)$$
$$y_{i+1} = \cos\phi_i(y_i + x_i \tan\phi_i) \qquad (6)$$

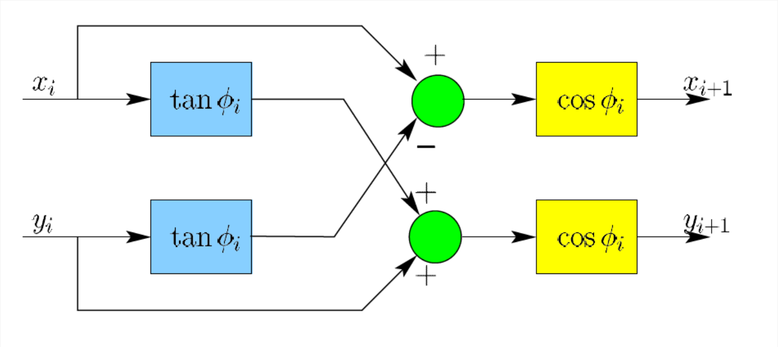


From Givens to CORDIC ...

STEP 3. Rewriting the equations related to rotation by ϕ_i.

To proceed, let us assume that $-\pi/2 < \phi_i < \pi/2$. Using $\tan\phi = \sin\phi/\cos\phi$, equation (5) can be rewritten as

$$\begin{aligned} x_{i+1} &= \cos\phi_i(x_i - y_i \tan\phi_i) \\ y_{i+1} &= \cos\phi_i(y_i + x_i \tan\phi_i) \end{aligned} \tag{6}$$

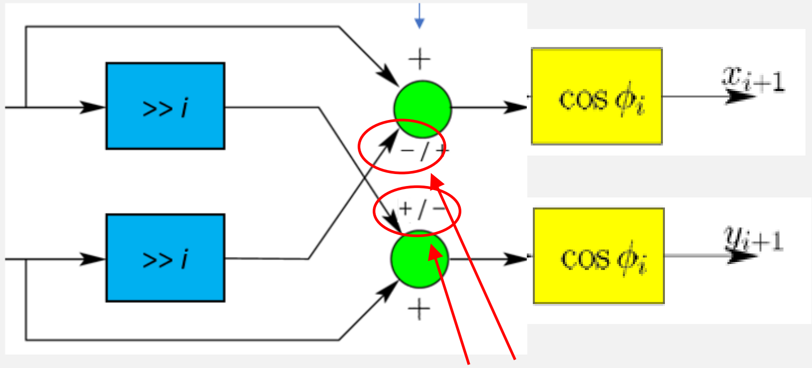


STEP 4. Restrict ϕ_i so that $\tan\phi_i = \pm 2^{-i}$ (i : integer ≥ 0).

Under this condition, (6) becomes

$$\begin{aligned} x_{i+1} &= \cos\phi_i(x_i - d_i \cdot y_i \cdot 2^{-i}) \\ y_{i+1} &= \cos\phi_i(y_i + d_i \cdot x_i \cdot 2^{-i}) \end{aligned} \tag{7}$$

where $d_i = +1$, if $\phi_i > 0$, and $d_i = -1$, if $\phi_i < 0$. Thus, substituting $d_i = -1$ for $d_i = +1$ corresponds to swapping of signs of the second terms within parentheses, that is, subtraction becomes addition and vice versa.



d_i controls the mode

Rotations whose tangent = arithmetic shift

Multiplication by 2^{-i} corresponds to **arithmetic shift** to right by i bits.

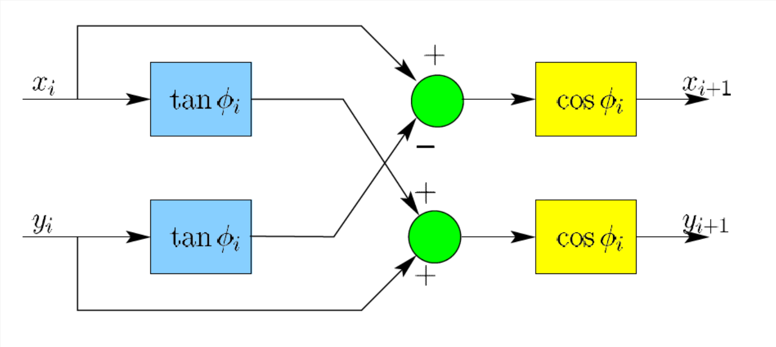
i	$\phi_i = \arctan(2^{-i})$ [deg]
0	45.00
1	26.57
2	14.04
3	7.13
4	3.58
5	1.79
6	0.90
7	

From Givens to CORDIC ...

STEP 3. Rewriting the equations related to rotation by ϕ_i .

To proceed, let us assume that $-\pi/2 < \phi_i < \pi/2$. Using $\tan\phi = \sin\phi / \cos\phi$, equation (5) can be rewritten as

$$\begin{aligned} x_{i+1} &= \cos\phi_i(x_i - y_i \tan\phi_i) \\ y_{i+1} &= \cos\phi_i(y_i + x_i \tan\phi_i) \end{aligned} \tag{6}$$

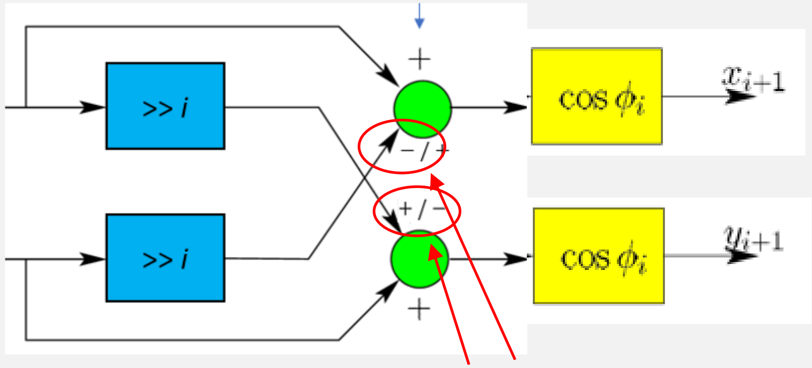


STEP 4. Restrict ϕ_i so that $\tan\phi_i = \pm 2^{-i}$ (i : integer ≥ 0).

Under this condition, (6) becomes

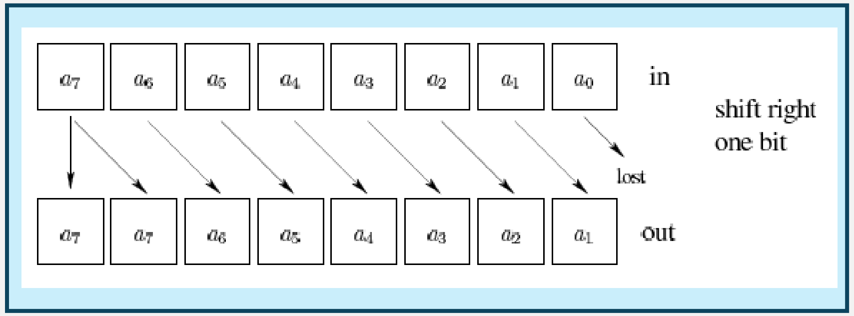
$$\begin{aligned} x_{i+1} &= \cos\phi_i(x_i - d_i \cdot y_i \cdot 2^{-i}) \\ y_{i+1} &= \cos\phi_i(y_i + d_i \cdot x_i \cdot 2^{-i}) \end{aligned} \tag{7}$$

where $d_i = +1$, if $\phi_i > 0$, and $d_i = -1$, if $\phi_i < 0$. Thus, substituting $d_i = -1$ for $d_i = +1$ corresponds to swapping of signs of the second terms within parentheses, that is, subtraction becomes addition and vice versa.



d_i controls the mode

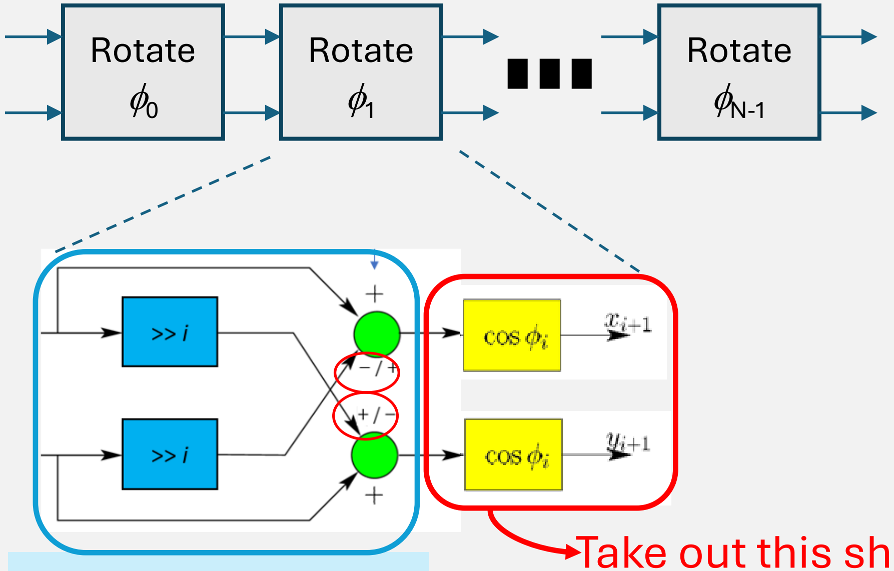
Multiplication by 2^{-i} corresponds to **arithmetic shift** to right by i bits. Exactly so, if discarded least significant bits are all zero! Otherwise, works like truncation (floor operation).



1110000111000000
>> 5
1111111100001110

From Givens to CORDIC …

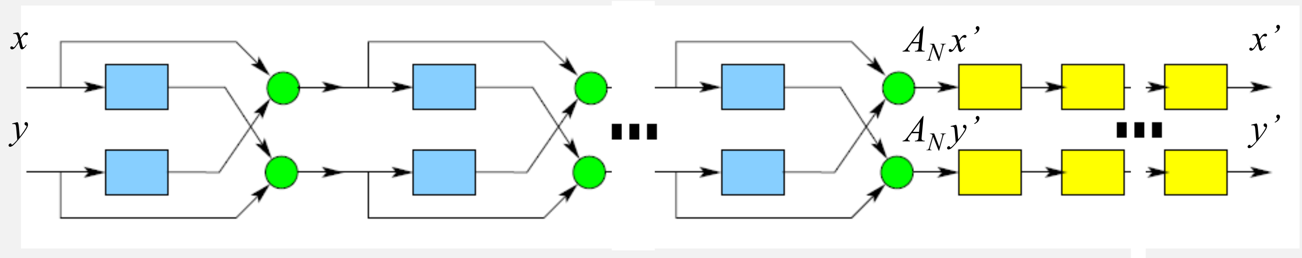
STEP 5. Chaining of rotations and taking multiplications by $\cos\phi_i$ out.



Gain of shift-add:

$a_i = \frac{1}{\cos\phi_i} = \sqrt{1+2^{-2i}}$

Take out this shift-add
"gain compensation"
and handle it elsewhere!

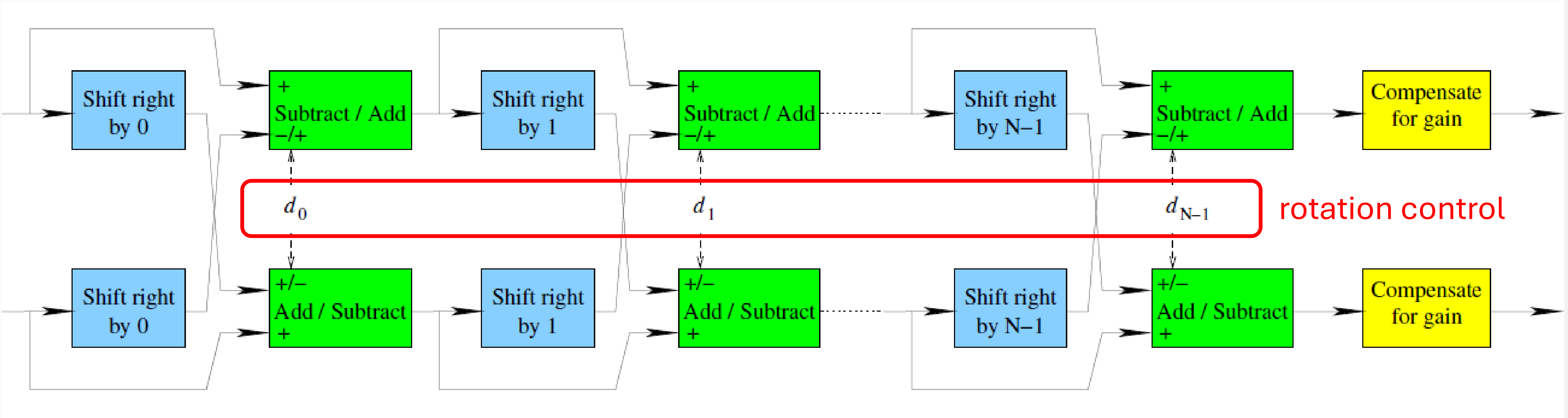


Shift-add stages

Gain compensations

From Givens to CORDIC …

RESULT. The derived fixed-point computation structure for Givens transform.



Gain issue. Total gain over shift-add stages is $A_N = \prod_{i=0}^{N-1}(\cos\phi_i)^{-1} = \prod_{i=0}^{N-1}\sqrt{1+2^{-2i}}$.
The gain is constant, that is, it does not depend on the target angle.

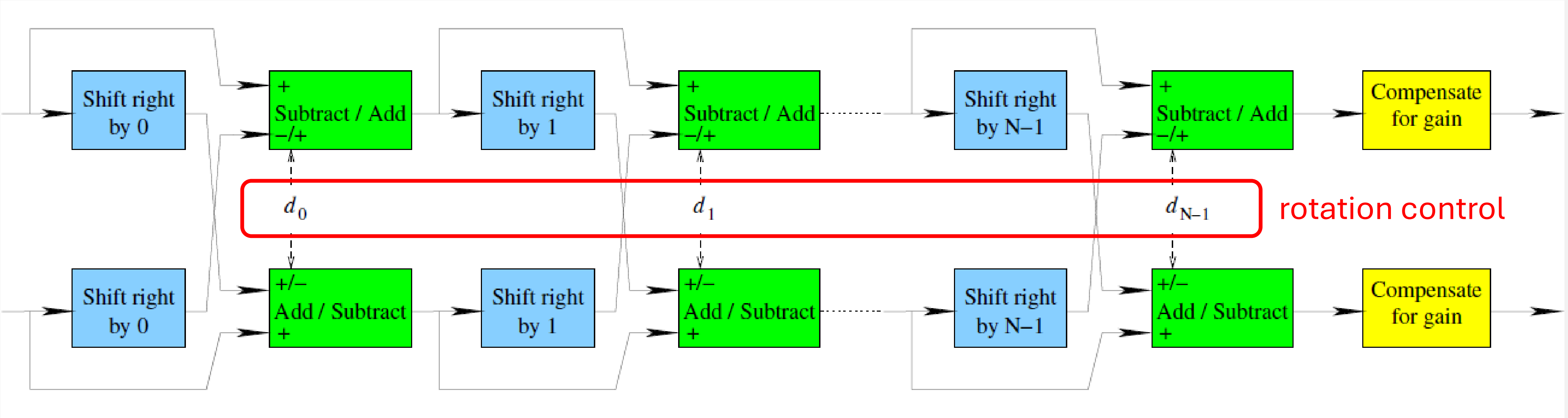
$$A_N = \prod_{i=0}^{N-1} (\cos \phi_i)^{-1} = \prod_{i=0}^{N-1} \sqrt{1 + 2^{-2i}}$$

i	$\phi_i = \arctan(2^{-i})$ [deg]	$(\cos \phi_i)^{-1}$
0	45.00	$\sqrt{1 + 2^{-0}} = \sqrt{2}$
1	26.57	$\sqrt{1 + 2^{-2}} = \sqrt{5/4}$
2	14.04	$\sqrt{1 + 2^{-4}} = \sqrt{17/16}$
3	7.13	etc.
4	3.58	For large i, $\sqrt{1 + 2^{-i}} \approx 1$
5	1.79	
6	0.90	
7		

product $\rightarrow$ A_N

From Givens to CORDIC …

RESULT. The derived fixed-point computation structure for Givens transform.



Gain issue. Total gain over shift-add stages is $A_N = \prod_{i=0}^{N-1}(\cos\phi_i)^{-1} = \prod_{i=0}^{N-1}\sqrt{1+2^{-2i}}$.
The gain is constant, that is, it does not depend on the target angle.

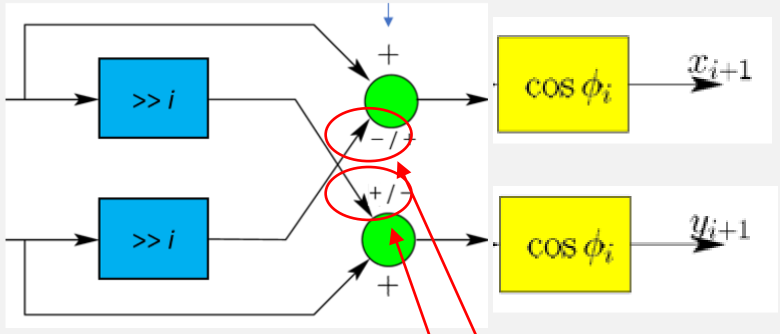
Compensation for A_N is done in some appropriate place of the signal processing chain.
Can happen as pre-processing or post-processing.
Example: JPEG coding with CORDIC based DCT : gain compensation included in the final coefficient quantization step.

If it is sufficient to have output values just in right ratio, gain compensation is not even needed.

Rotation control for angle ϕ

The angle ϕ of a composite rotation is uniquely defined by the sequence of elementary rotation directions,

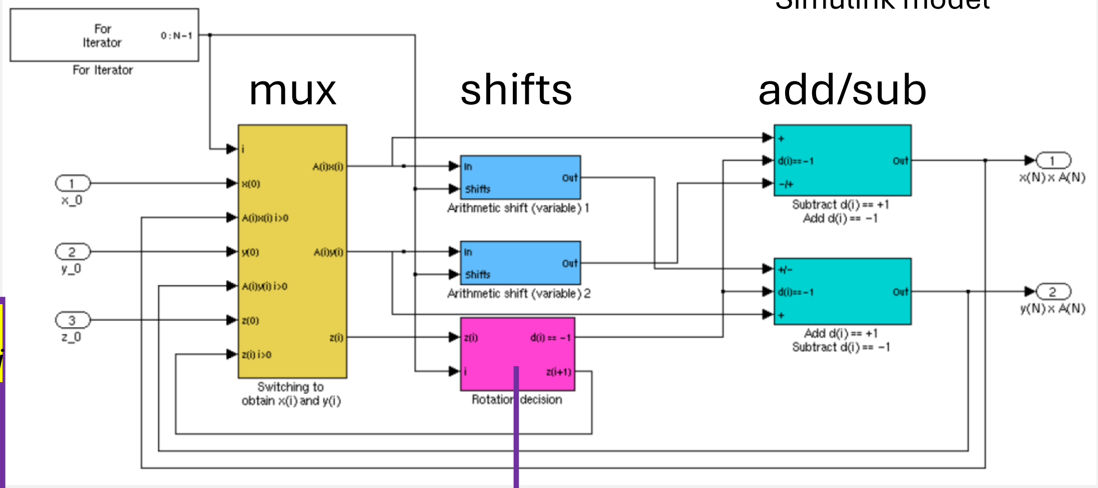
$$(d_0, d_1, \ldots, d_{N-1}).$$



d_i controls the mode

1. If some fixed set of composite rotations are needed, one can **precalculate** decision sequences and store them as bit strings into some lookup table. Note: exhaustive search for best rotation combination is possible.

Simulink model



2. If the rotation directions must be determined **on the fly**, one possibility is to do iterative computations as shown on the right. Here, decisions must be done sequentially. This is done

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

REMAINING ANGLE z_i

where z_i $(i = 0, 1, \ldots)$ denotes the remaining rotation before performing the rotation by ϕ_i $(z_0 = \phi)$. The decision rule is

$$d_i = \begin{cases} -1 & \text{if } z_i < 0 \\ +1 & \text{otherwise} \end{cases} \tag{12}$$

Angles $\arctan(2^{-i}) = \phi_i$ are stored into a **lookup table**, which is used within the decision block.

Numeric example

- Rotation of the vector (x,y) = (1,0) by ϕ=40 degrees. Precision requirement: absolute error less than 0.5 degrees.

1. Rotation sequence

Calculated sequentially using the rule on the right (shown in the previous slide).

$$z_{i+1} = z_i - d_i \cdot \arctan(2^{-i}) \tag{11}$$

where z_i $(i = 0, 1, \ldots)$ denotes the remaining rotation before performing the rotation by ϕ_i $(z_0 = \phi)$. The decision rule is

$$d_i = \begin{cases} -1 & \text{if } z_i < 0 \\ +1 & \text{otherwise} \end{cases} \tag{12}$$

Target angle = remaining angle in the beginning

i	z_i [deg]	d_i	$\arctan(2^{-i})$ [deg]	z_{i+1} [deg]
0	$+40.00$	$+1$	45.00	-5.00
1	-5.00	-1	26.57	$+21.57$
2	$+21.57$	$+1$	14.04	$+7.53$
3	$+7.53$	$+1$	7.13	$+0.40$
4	$+0.40$	$+1$	3.58	-3.18
5	-3.18	-1	1.79	-1.39
6	-1.39	-1	0.90	-0.49
7	-0.49			

Remaining angles after each iteration, inputs to the next iterations.

Angles in the decision block's **lookup table**.

Numeric example

2. Intermediate coordinates in computation

Note: gain is included in the computed values. Computation can be expressed as the rule shown on the right. The inputs to iteration *i* are $x_i A_i$ and $y_i A_i$. The outputs are $x_{i+1} A_{i+1}$ and $y_{i+1} A_{i+1}$. Note that $x_0 A_0 = x$ and $y_0 A_0 = y$.

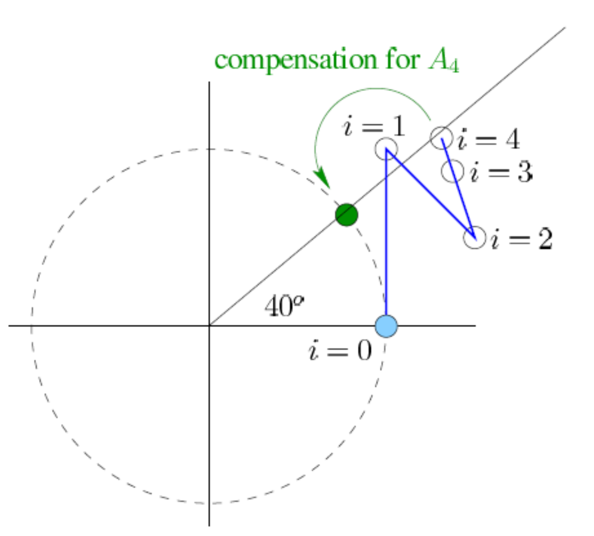
$$x_{i+1}A_{i+1} = \boxed{x_i A_i} - d_i \cdot \boxed{y_i A_i} \cdot 2^{-i}$$
$$y_{i+1}A_{i+1} = \boxed{y_i A_i} + d_i \cdot \boxed{x_i A_i} \cdot 2^{-i}$$

First iterations, results shown with full precision (2's complement in parentheses) ...

The point to be rotated, (x, y)

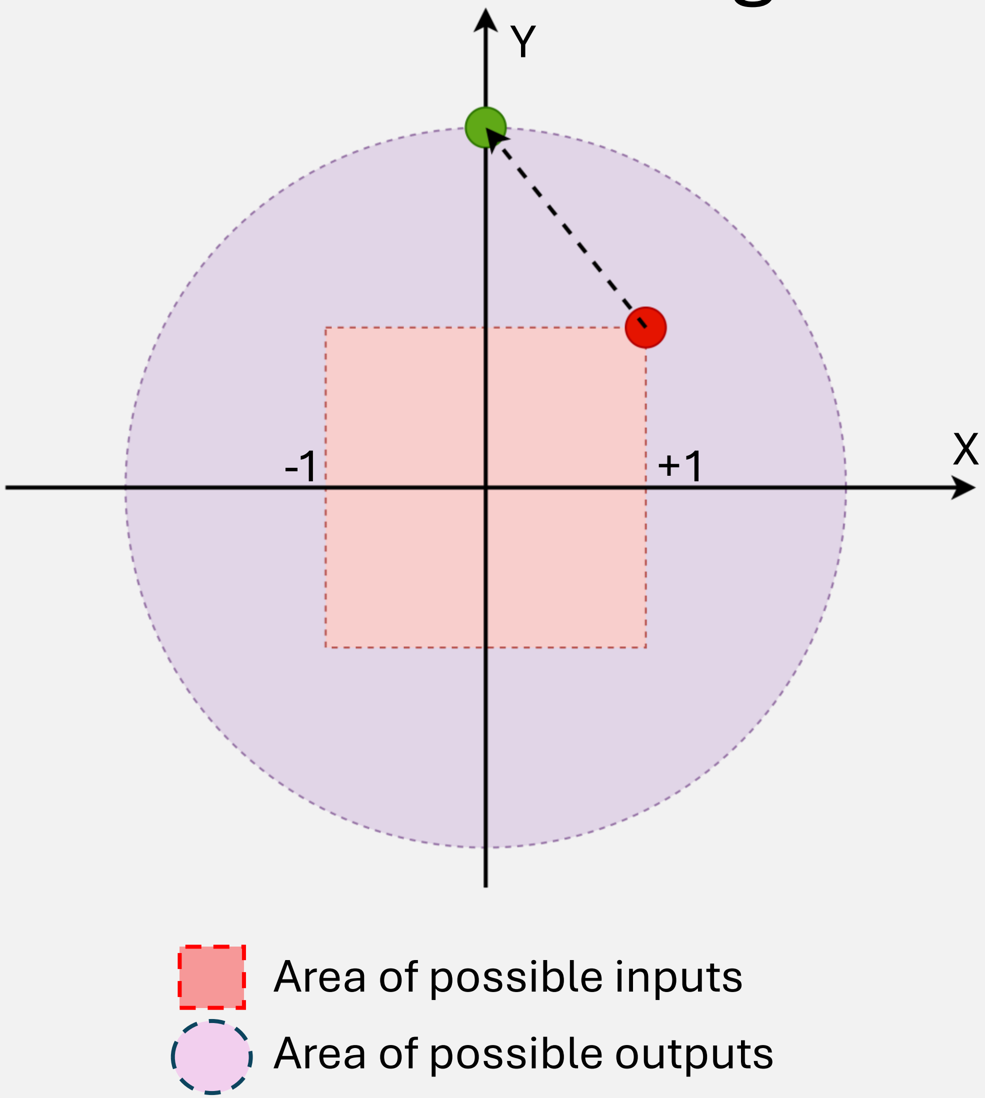
i	$x_i A_i$	$y_i A_i$	d_i	$x_{i+1} A_{i+1}$	$y_{i+1} A_{i+1}$
0	1	0	+1	1 (01.)	1 (01.)
1	1	1	−1	1.5 (01.1)	0.5 (00.1)
2	1.5	0.5	+1	1.375 (01.011)	0.875 (00.111)
3	1.375	0.875	+1	1.265625 (01.010001)	1.046875 (01.000011)
4	1.265625	1.046875			

The number of fraction bits increases fast in the full precision result.



CORDIC gain puts the result out of circle (in general, extra integer bits may be needed).

Effect of CORDIC gain on word length



Input coordinates within the range (-1,1), format **s**p.(p-1)

Consider red point (has input radius $\approx \sqrt{2}$).

Rotating it by 45 degrees with CORDIC maximizes output Y.

The green point is the output location:

$$Y = \text{CORDIC gain x point radius} = 1.64676 \times \sqrt{2} = 2.32887$$

The output fixed-point format is

s(p+k+2).(p-1+k)

where k is the fraction length increase due to arithmetic shifts and increment of the word length by 2 comes from this output radius analysis: **two extra integer bits are needed**.

Reachability of angles

Note 1. Restricting ϕ_i so that $\tan\phi_i = \pm 2^{-i}$ is ok from the viewpoint of angles. A feature of this is that

$$2\phi_{i+1} > \phi_i$$

We cover by *N* rotations the angles in the **range** $[-\Sigma_N, +\Sigma_N]$, where $\Sigma_N = \sum_{i=0}^{N-1} \phi_i$. By increasing *N*, we can increase the **density** of reachable angles.

For large N, each increment of N corresponds to one-bit increment in the precision of calculations.

i	$\tan\phi_i$	ϕ_i **[deg]** $\approx$
0	± 1	± 45
1	± 0.5	± 26.565
2	± 0.25	± 14.036
3	± 0.125	± 7.125
4	± 0.0625	± 3.576
5	± 0.03125	± 1.790

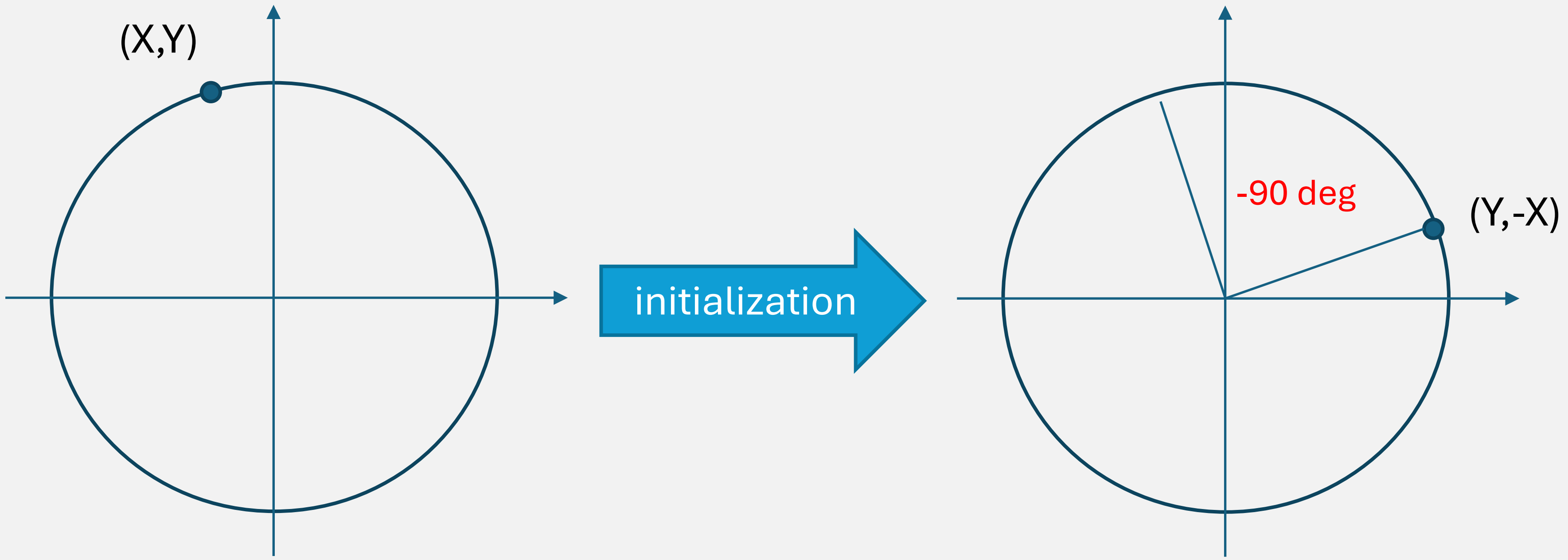
Note 2. $\lim_{N\to\infty} \Sigma_N \approx 99.83$ degrees. We **cannot cover the whole range of rotation angles**, [-180°, +180°].

If it is needed, we can add an extra **initialization step**, where we rotate the coordinates by ±90° which corresponds to input coordinate swapping as shown on the right.

Target angle ϕ	**Rotation by**	**Substitutions**
≤ 0	+90°	$x \leftarrow y$ $y \leftarrow -x$
≥ 0	-90°	$x \leftarrow -y$ $y \leftarrow x$

Note. Gain = 1 for this step. No compensation needed.

Example of initialization



Target: rotate by -145 degrees the point (X, Y)

New target rotation angle -145+90 = -55 deg

The coordinate to be rotated by CORDIC iterations (Y, -X)

Summary

To compute the Givens transform of (x,y) with the CORDIC for **any angle ϕ in the range [-180, 180] degrees**,

1. Compute the initial values for CORDIC iterations using the table on the right.

2. CORDIC loop:

   Iterate N times ($i = 0, \ldots, N-1$):

   (a) Determine the rotation direction d_i

   (b) Compute $x_{i+1}A_{i+1}$ and $y_{i+1}A_{i+1}$ using shift-add arithmetic

   (c) Compute the remaining angle z_{i+1}

3. Output of previous is $x_N A_N$ and $y_N A_N$. If necessary, compensate for the gain A_N somewhere in the full signal processing chain (before or after CORDIC).

Target angle ϕ	Rotation by	Initialization
≤ 0	+90°	$x_0A_0 \leftarrow +y$ $y_0A_0 \leftarrow -x$ $z_0 \leftarrow \phi + 90^o$
≥ 0	-90°	$x_0A_0 \leftarrow -y$ $y_0A_0 \leftarrow +x$ $z_0 \leftarrow \phi - 90^o$

$$d_i = \begin{cases} -1 & \text{if } z_i < 0 \\ +1 & \text{otherwise} \end{cases}$$

$$x_{i+1} \cdot A_{i+1} = x_i A_i - d_i \cdot y_i A_i \cdot 2^{-i}$$
$$y_{i+1} \cdot A_{i+1} = y_i A_i + d_i \cdot x_i A_i \cdot 2^{-i},$$

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

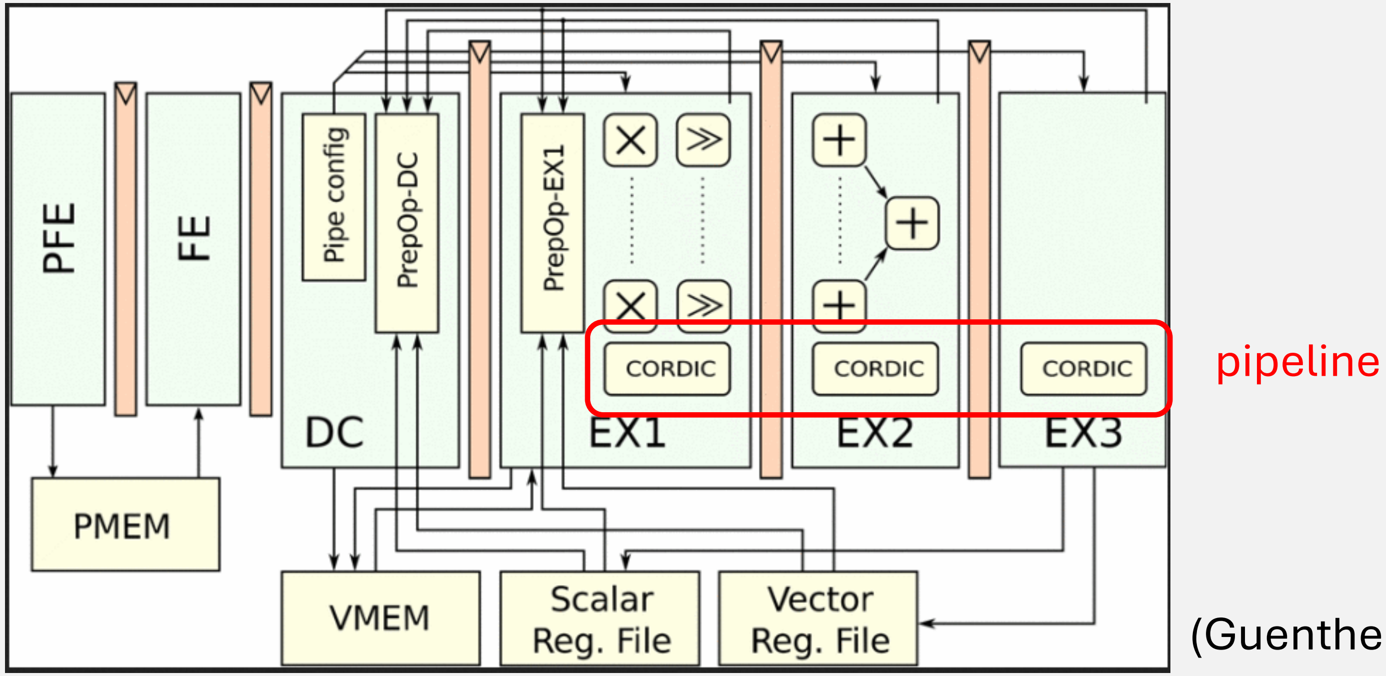
$$x_N A_N \times (1/A_N)$$
$$y_N A_N \times (1/A_N)$$

Note on implementation

The iteration loop can be unrolled, which gives a processor consisting of a chain of N units, each performing dedicated iteration.

An advantage of such a solution is **that shifts are fixed for each unit which allows hardwired implementation** of them (Andraka 1998).

Unrolled loop can also be mapped to a pipelined implementation.



(Guenther et al. 2014)

Design task 3

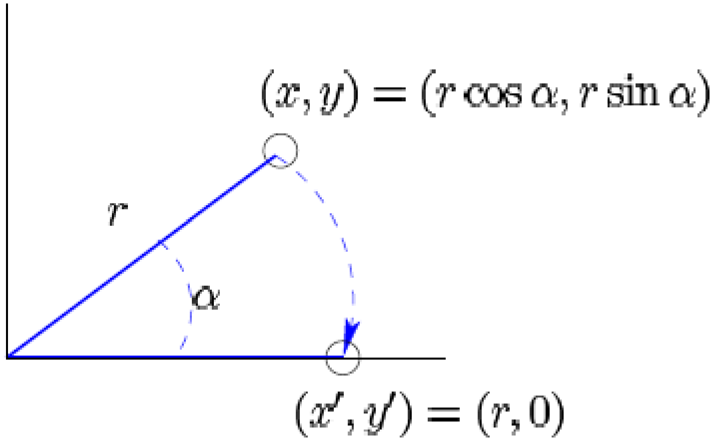
- T1. Givens rotation for a specific angle using CORDIC (2p)
  - Rotation sequence (d_i), intermediate values $x_i A_i$ and $y_i A_i$, gain computation
- T2. Simulating CORDIC using Matlab (1p)
  - Determination of sufficient fixed-point formats by simulation
- T3. Exercise on 3-valued CORDIC (1p)
  - Determining rotator for an 8-point DCT implementation
  - 3-v CORDIC & DCT presented next Monday

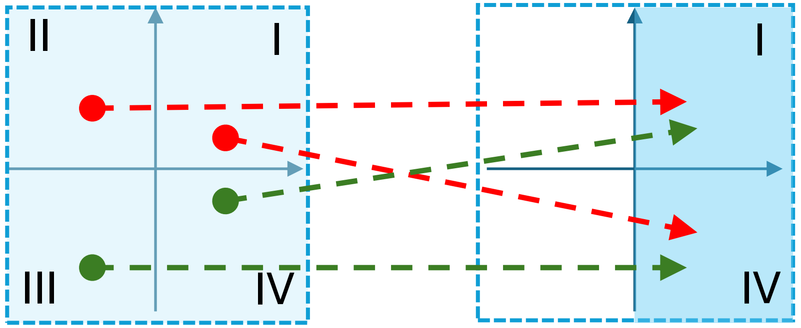
Polar transform & CORDIC

- Map coordinate (x,y) to polar representation (r, α)

- CORDIC shift-add stages can be used to compute ($A_N r$, α)
  - The idea is to perform rotations **until the coordinate** $A_i y_i$ **goes to zero**. Then, $A_i x_i$ converges to $A_n r$.
  - The sum of performed rotations in z_i provides α.

- Implementation requires
  - **Decision logic for rotations**: in Givens transform (sequential) decisions were based on the remaining angles z_i, now it is based on the sign of the coordinate $\boldsymbol{A_i y_i}$
  - **Initialization logic** to cover the whole coordinate space: ±90° rotation of the coordinate maps coordinate to I or IV quadrant





Suitable inputs for the CORDIC loop

Algorithm

1. Compute the initial values of $x_0 = x_0A_0$, $y_0 = y_0A_0$ and z_0 using (14) and (15).
2. Iterate N times ($i = 0, \ldots, N-1$):
   (a) Determine the rotation direction d_i using (13).
   (b) Compute $x_{i+1}A_{i+1}$ and $y_{i+1}A_{i+1}$ using shift-add arithmetic based on

$$\begin{aligned} x_{i+1}A_{i+1} &= x_iA_i - d_i \cdot y_iA_i \cdot 2^{-i} \\ y_{i+1}A_{i+1} &= y_iA_i + d_i \cdot x_iA_i \cdot 2^{-i}, \end{aligned} \qquad (16)$$

   as described in Sec. 2.
   (c) Update the sum of angles, z_{i+1}, using (12).
3. Compensate for the gain A_N in the result to obtain x_N, which corresponds to the unknown radius r. z_N gives the unknown angle α.

$$d_{\text{init}} = \begin{cases} -1 & \text{if } y > 0 \\ +1 & \text{otherwise.} \end{cases} \qquad (14)$$

$$\begin{aligned} x_0 &= -d_{\text{init}} \cdot y \\ y_0 &= d_{\text{init}} \cdot x \\ z_0 &= -d_{\text{init}} \cdot \pi/2 \end{aligned} \qquad (15)$$

$$d_i = \begin{cases} -1 & \text{if } y_iA_i > 0 \\ +1 & \text{otherwise.} \end{cases} \qquad (13)$$

$$z_{i+1} = z_i - d_i \cdot \arctan(2^{-i}) \qquad (12)$$

Numeric example

Mapping (x,y) = (3,4) to polar coordinates

- Initializing rotation -90 deg ($d_{init} = -1$)
  - => CORDIC loop begins from (4, -3)
- First iterations

i	x_iA_i	y_iA_i	z_i [deg]	d_i	$\arctan(2^{-i})$ [deg]
0	$+4.00$	-3.00	$+90.00$	$+1$	45.00
1	$+7.00$	$+1.00$	$+45.00$	-1	26.57
2	$+7.50$	-2.50	$+71.57$	$+1$	14.04
3	$+8.13$	-0.63	$+57.53$	$+1$	7.13
4	$+8.20$	$+0.39$	$+50.40$	-1	3.58
5	$+8.23$	-0.12	$+53.98$		

$\sqrt{3^2 + 4^2} = 5$

Converges to
5 x Cordic gain

Converges to
+53.13 deg

$3 = 5\cos 53.13^o$

$4 = 5\sin 53.13^o$

Summary

- Two applications of CORDIC explained
  - Givens transform
  - Polar transform

Trigonometric operations

- Next lecture
  - Unified CORDIC
  - DCT & CORDIC in its implementation