EE290C - Fall 2018

Advanced Topics in Circuit Design

VLSI Signal Processing

Lecture 3: Finite Wordlength Effects

CORDIC

Announcements

- Assignment 1 – due today
- Assignment 2 – due on Thursday 9/6
  - Chisel bootcamp 2.3-2.5, 3.1-3.2

Projects

- OFDM software radio (LTE, WiFi)
- Bluetooth modem
- GPS receiver
- Smart NIC
- Wellness monitor

- #students on project = #unique blocks

Reading

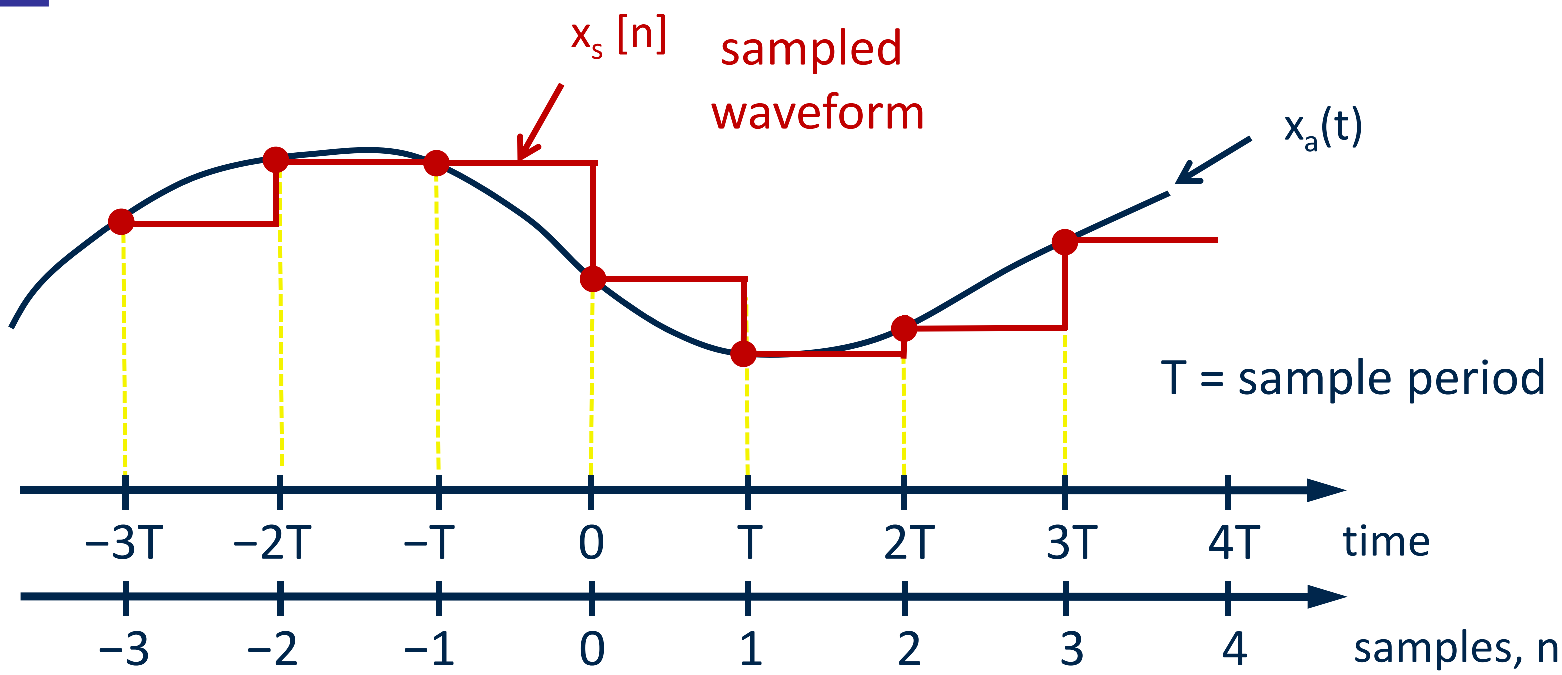
- Numbers
  - Markovic, Brodersen, DSP Architecture Design Essentials, 2012, (Ch. 5)
- CORDIC
  - Markovic, Brodersen, DSP Architecture Design Essentials, 2012, (Ch. 6)
  - Any other book above

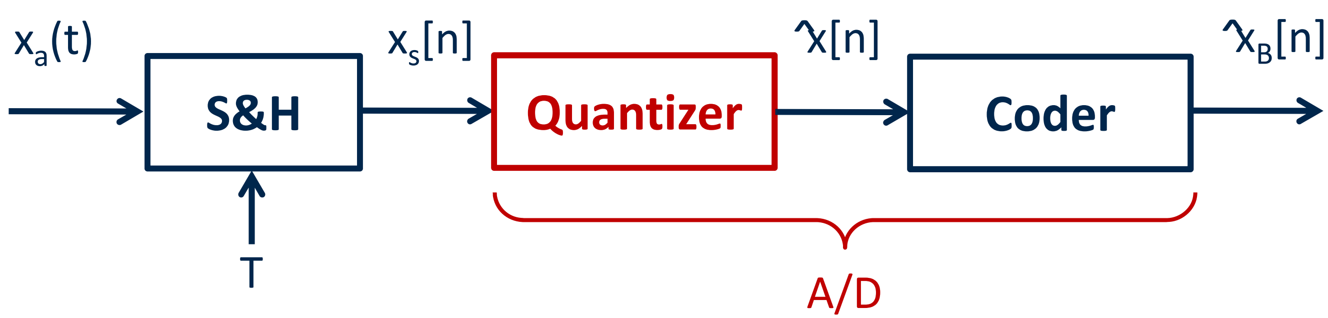
Advanced Topics in Circuit Design

VLSI Signal Processing

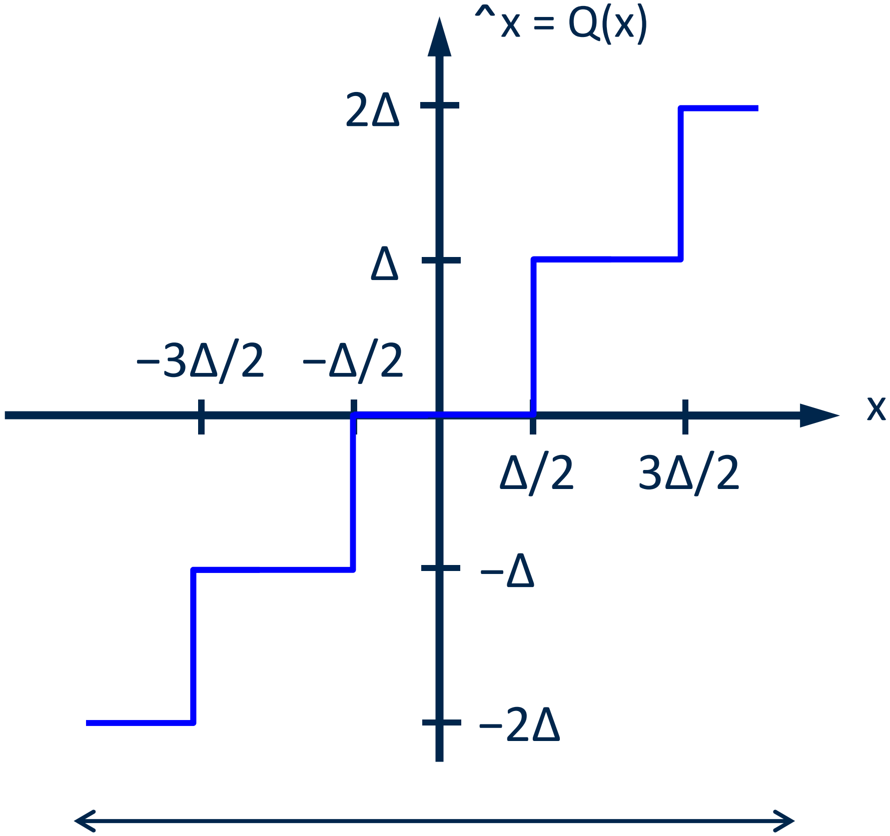
Finite Wordlength Effects

Quantization Effects

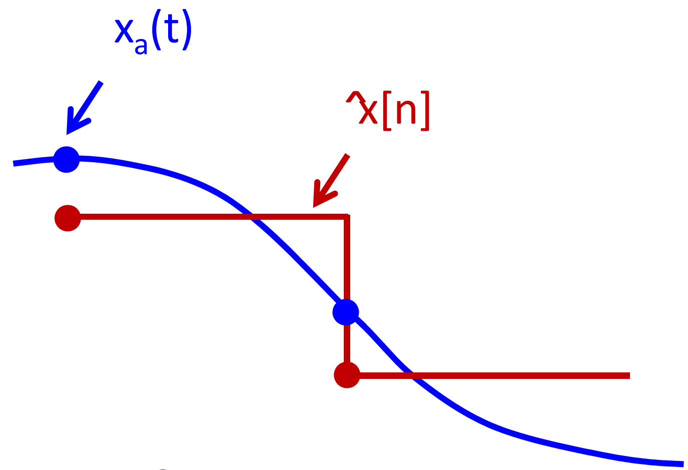




Quantization



Full range: $2x_m$



$$\Delta = \frac{2x_m}{2^{B+1}} = \frac{x_m}{2^B}$$

B = # bits of quantization

$$e[n] = \hat{x}[n] - x[n]$$

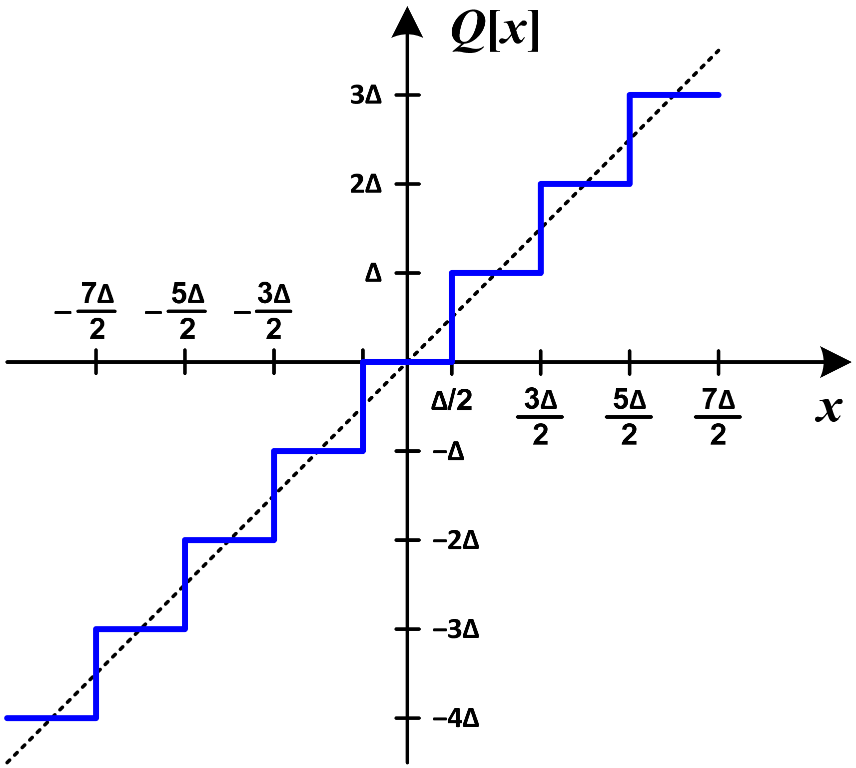
$$-\Delta/2 < e[n] \leq \Delta/2$$

(AWN process)

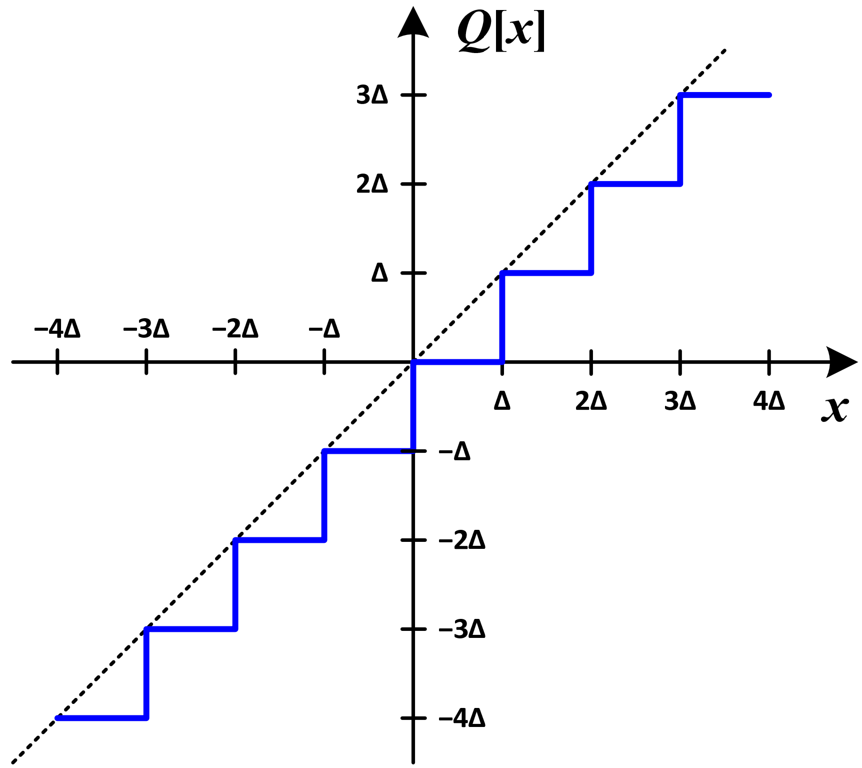
2's complement representation

Quantization: Rounding, Truncation

Rounding

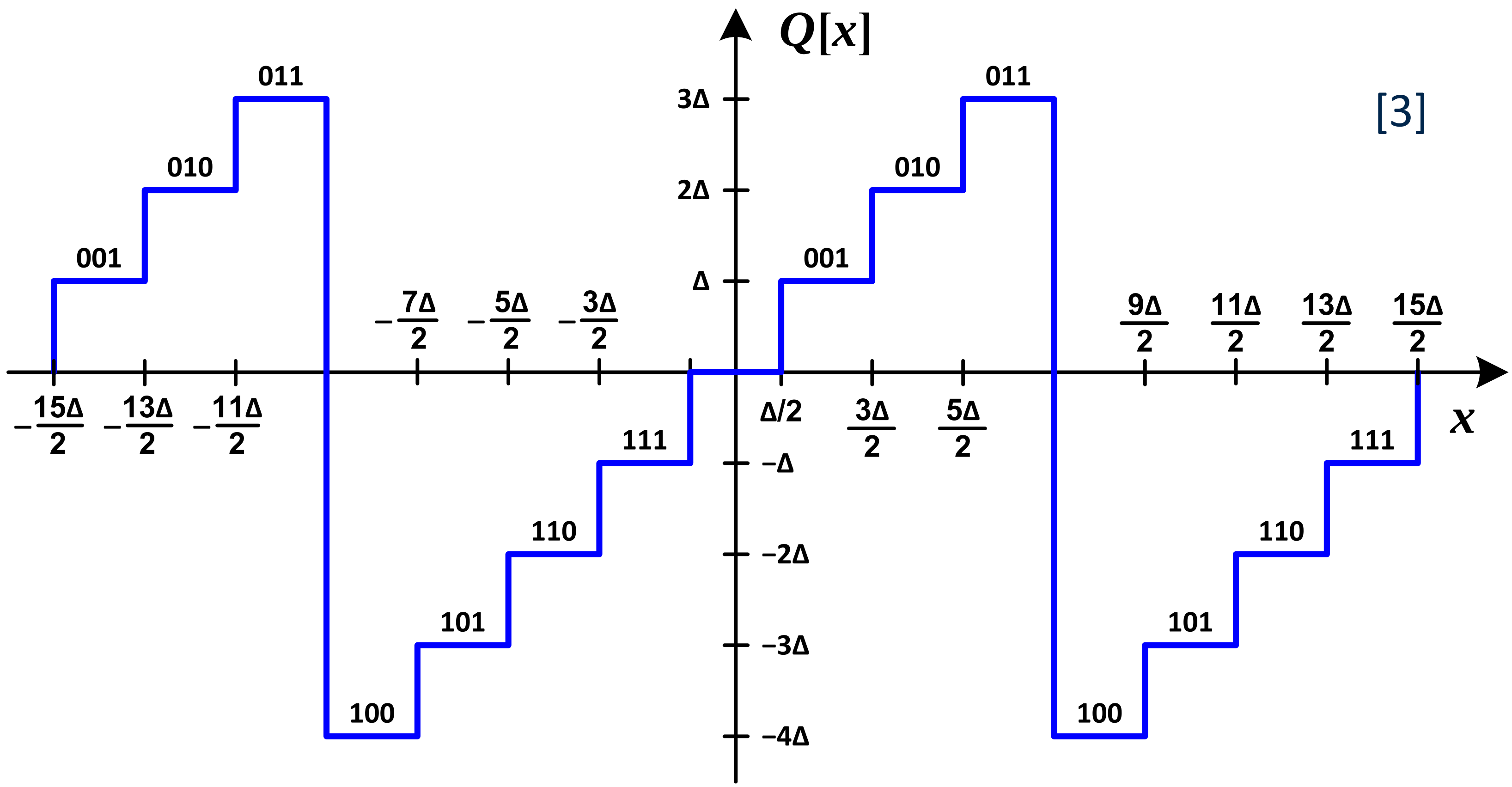


Truncation



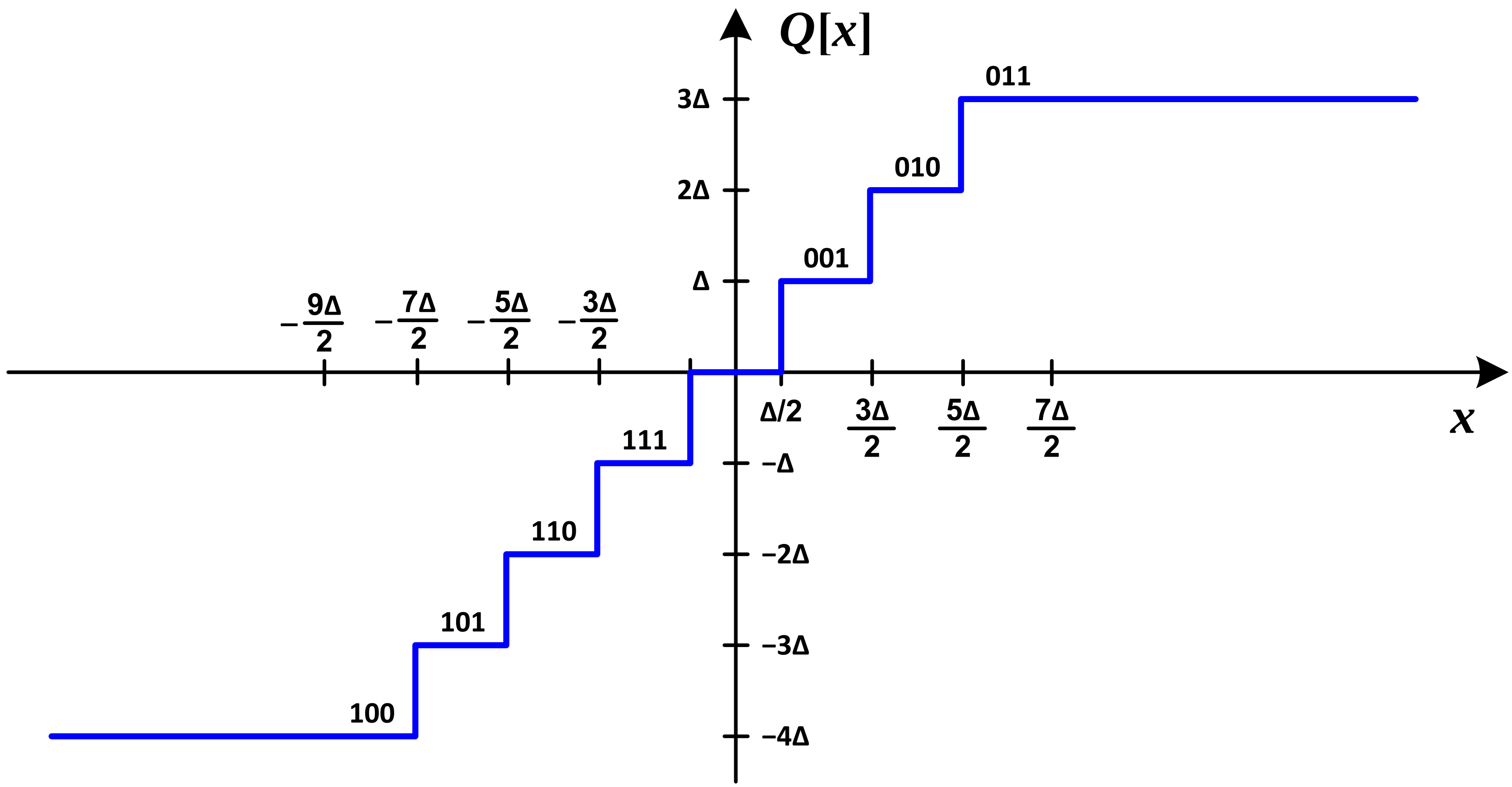
Feedback systems use rounding

Overflow Modes: Wrap Around



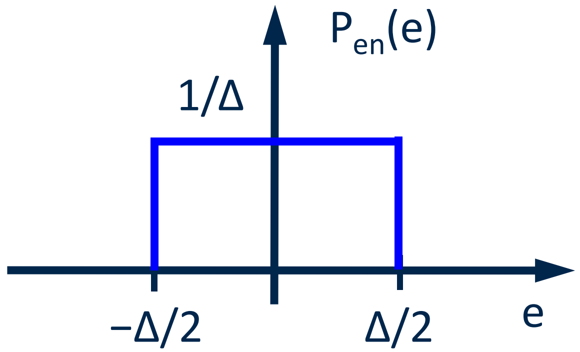
[3] A.V. Oppenheim, R.W. Schafer, with J.R. Buck, Discrete-Time Signal Processing, (2nd Ed), Prentice Hall, 1998.

Overflow Modes: Saturation



Feedback systems use saturation

Quantization Noise



$$\sigma_e^2 = \int_{-\Delta/2}^{\Delta/2} e^2 \frac{1}{\Delta} de = \frac{\Delta^2}{12}$$

x_m: full-scale signal

B + 1 quantizer

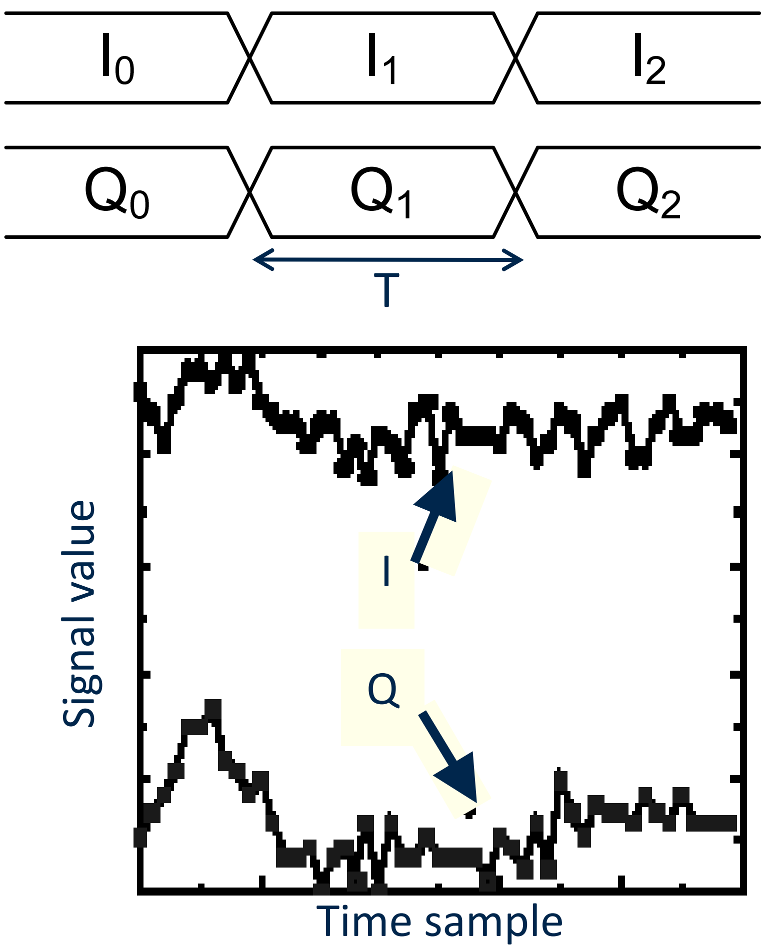
$$\sigma_e^2 = \frac{2^{-2B} x_m^2}{12}$$

$$SQNR = 10log_{10}\left(\frac{\sigma_x^2}{\sigma_e^2}\right) = 10log_{10}\left(\frac{12 \cdot 2^{2B} \cdot \sigma_x^2}{x_m^2}\right)$$

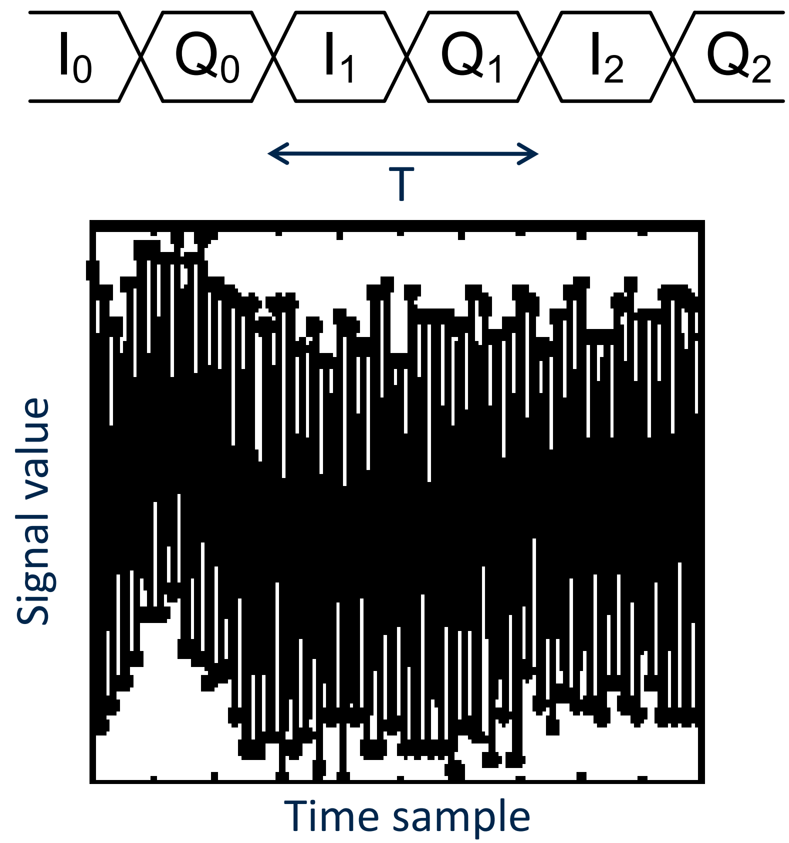
$$= 6.02 \cdot B + 10.8 - 20log_{10}\left(\frac{x_m}{\sigma_x}\right)$$

Time-Multiplexed Architectures

Parallel bus for I,Q

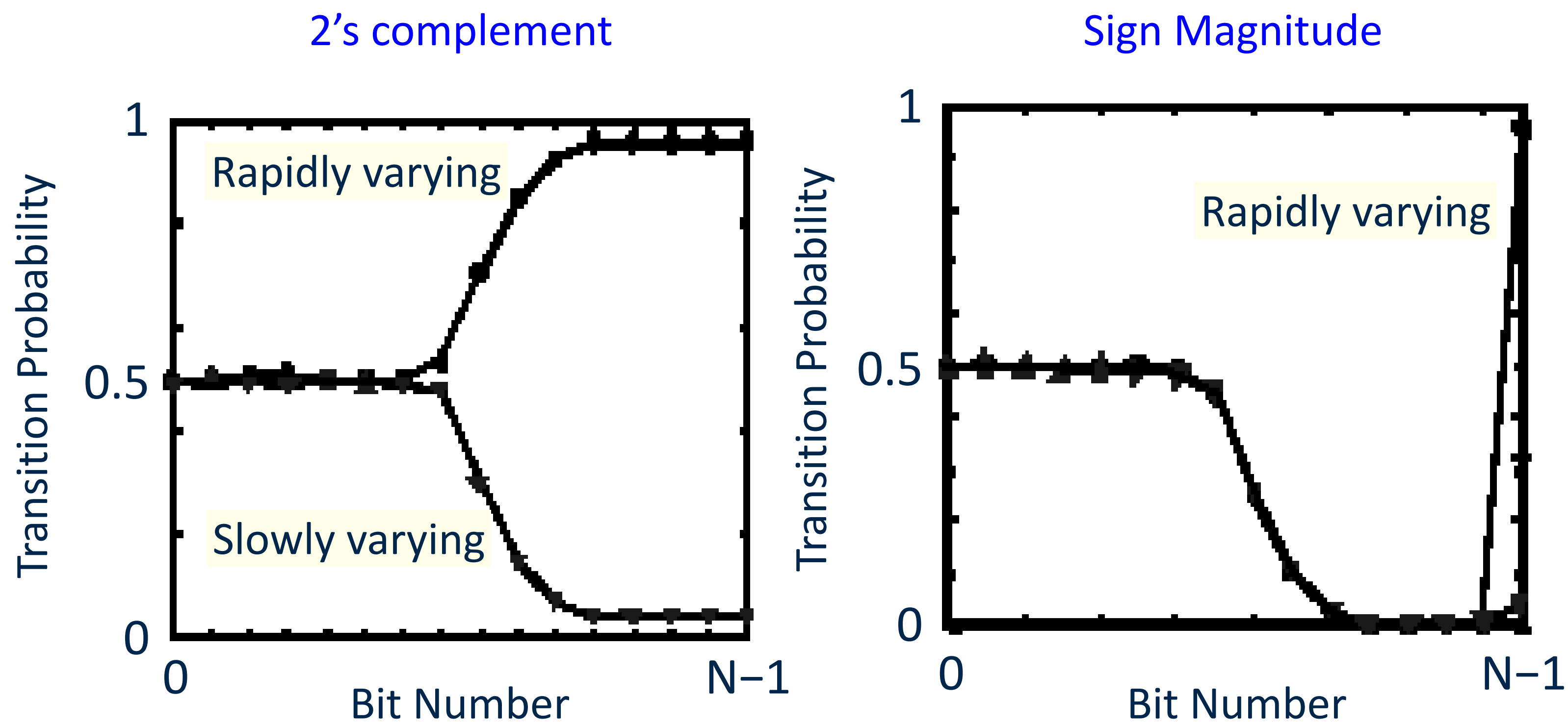


Time-shared bus for I,Q



- **Time-shared bus destroys signal correlations and increases switching activity**

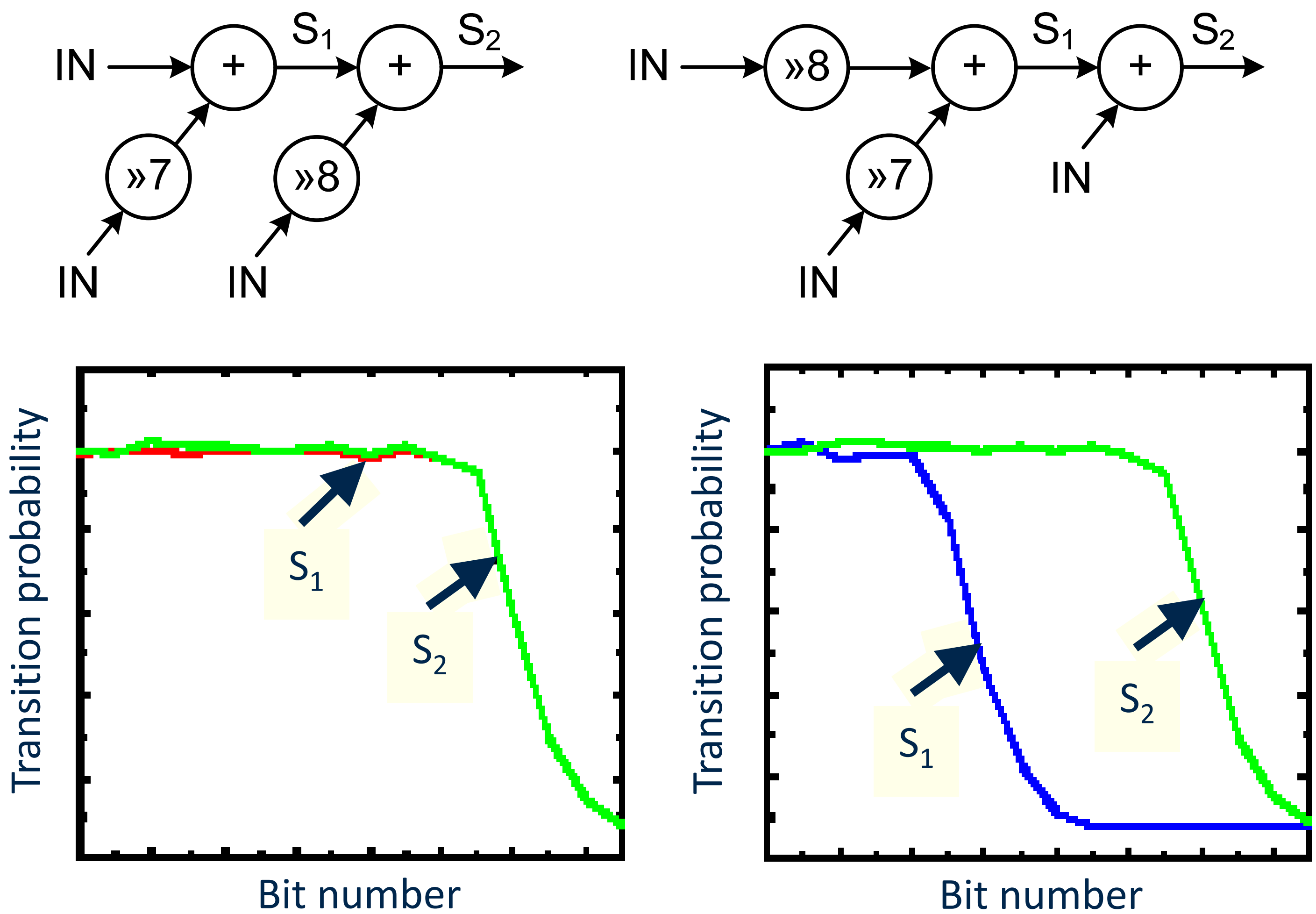
Number Representation



- **Sign-extension activity is significantly reduced using sign-magnitude representation** [5]

[5] A. Chandrakasan, Low Power Digital CMOS Design, Ph.D. Thesis, University of California, Berkeley, 1994.

Reducing Activity by Reordering Inputs



30% reduction in switching energy

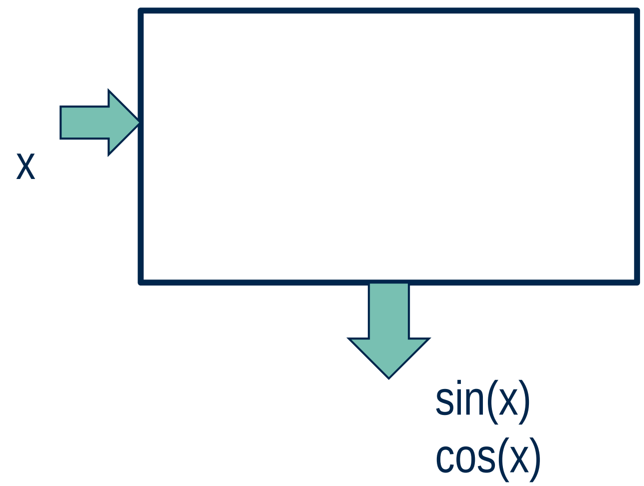
Advanced Topics in Circuit Design

VLSI Signal Processing

CORDIC Algorithm and Implementation

Trigonometric Functions

- How to implement sine, cosine, etc?
- Lookup table:



- Or CORDIC

CORDIC

To perform the following transformation

$$\boxed{y(t) = y_R + j \cdot y_I \rightarrow |y| \cdot e^{j\phi}}$$

and the inverse, we use the CORDIC algorithm

CORDIC:

COordinate Rotation DIgital Computer

CORDIC: Idea

Use rotations to implement a variety of functions

Examples:

$$x + j \cdot y \Leftrightarrow |\sqrt{x^2 + y^2}\,|\,e^{j \cdot tan^{-1}(y/x)}$$

$$z = \sqrt{x^2 + y^2} \qquad z = \cos(y/x) \qquad z = \tan(y/x)$$

$$z = x/y \qquad z = \sin(y/x) \qquad z = \sinh(y/x)$$

$$z = tan^{-1}(y/x) \qquad z = cos^{-1}(y)$$

CORDIC: How to Do It?

- Start with general rotation by φ

$$\mathbf{x' = x \cdot \cos(\phi) - y \cdot \sin(\phi)}$$
$$y' = y \cdot \cos(\phi) + x \cdot \sin(\phi)$$

$$x' = \cos(\phi) \cdot [x - y \cdot \tan(\phi)]$$
$$y' = \cos(\phi) \cdot [y + x \cdot \tan(\phi)]$$

- The key is to only do rotations by values of $\tan(\varphi)$ which are powers of 2

CORDIC: An Iterative Process

- To rotate to any arbitrary angle, we do a sequence of rotations to get to that value

Rotation Number

φ	$\tan(\varphi)$	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

Basic CORDIC Iteration

Gain Direction Angle

$$\mathbf{x_{i+1} = (K_i) \cdot [x_i - y_i \cdot d_i \cdot 2^{-i}]}$$

$$y_{i+1} = (K_i) \cdot [y_i + x_i \cdot d_i \cdot 2^{-i}]$$

$$\mathbf{K_i = \cos(\tan^{-1}(2^{-i})) = 1/(1 + 2^{-2i})^{0.5}}$$

$$d_i = \pm 1 \text{ (rotate by } \pm\phi)$$

- If we don't multiply by K_i we get a gain error which is independent of the direction of the rotation
  - The error converges to 0.61
  - May not need to compensate for it

Basic CORDIC Iteration

Gain Direction Angle

$$x_{i+1} = (K_i) \cdot [x_i - y_i \cdot d_i \cdot 2^{-i}]$$
$$y_{i+1} = (K_i) \cdot [y_i + x_i \cdot d_i \cdot 2^{-i}]$$

$$K_i = \cos(\tan^{-1}(2^{-i})) = 1/(1 + 2^{-2i})^{0.5}$$

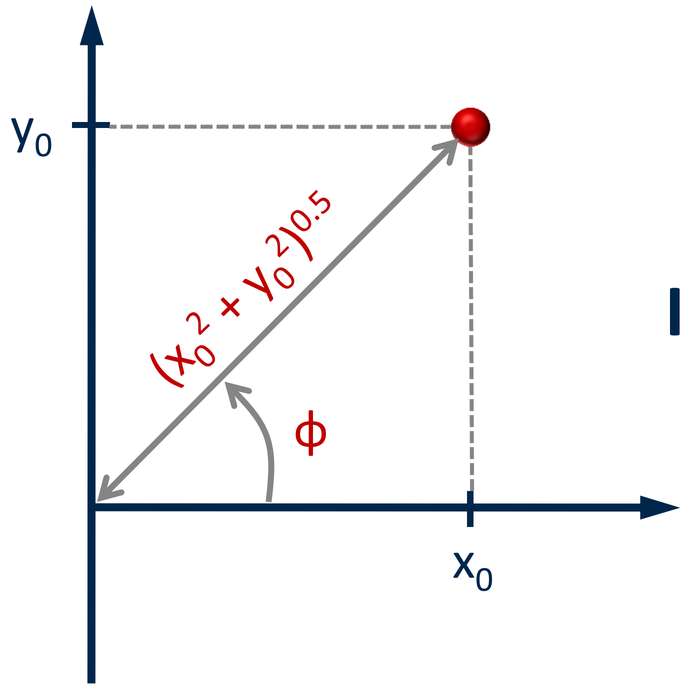
$d_i = \pm 1$ (rotate by $\pm\phi$)

- We can also accumulate the rotation angle:

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

Cartesian to Polar Conversion

Initial vector is described by x_0 and y_0 coordinates

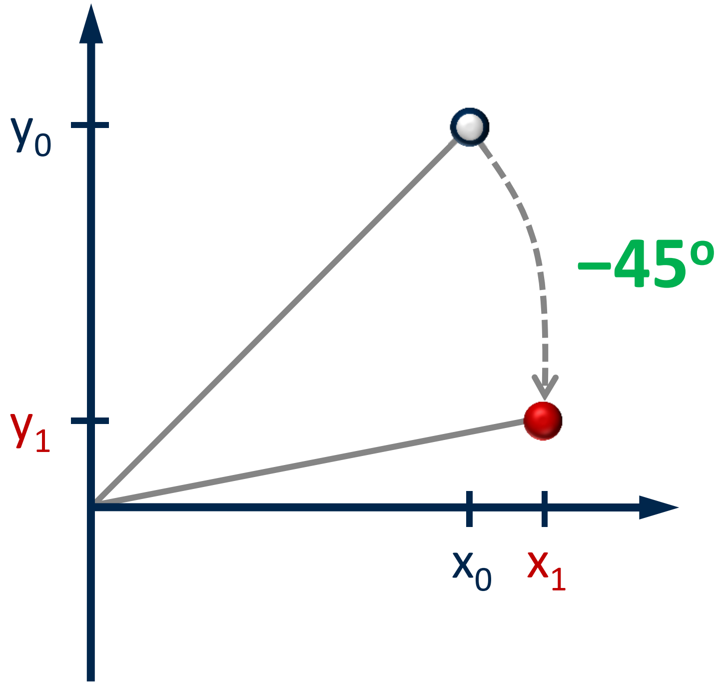


Initial vector

Find ϕ and $(x_0^2 + y_0^2)^{0.5}$

Step 1: Check the Angle / Sign of y_0

- If positive, rotate by **–45º**
- **If negative, rotate by +45º**

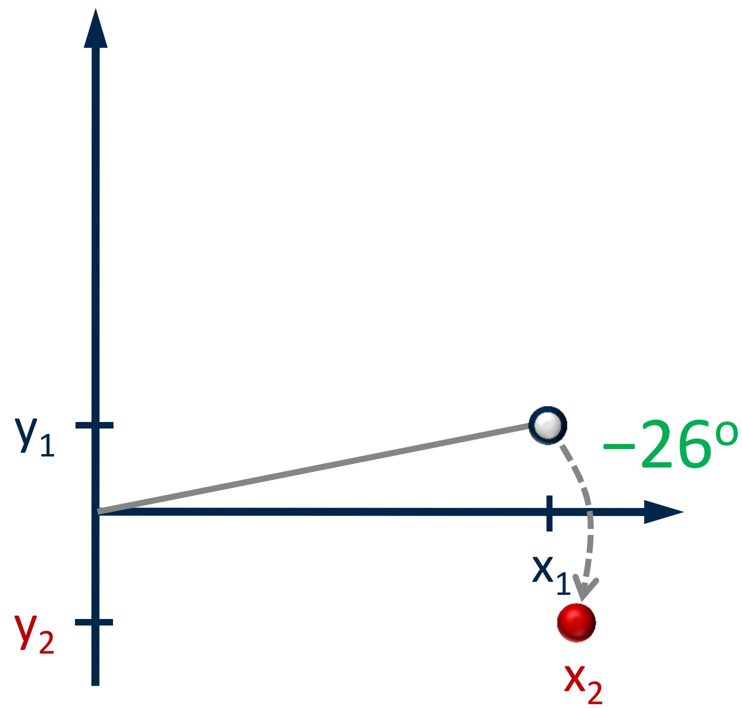


$d_1 = -1$ **(y_0 > 0)**

$\mathbf{x_1 = x_0 + y_0}$
$y_1 = y_0 - \mathbf{x_0}$

Step 2: Check the Sign of y_1

- **If positive, rotate by –26.57º**
- **If negative, rotate by +26.57º**

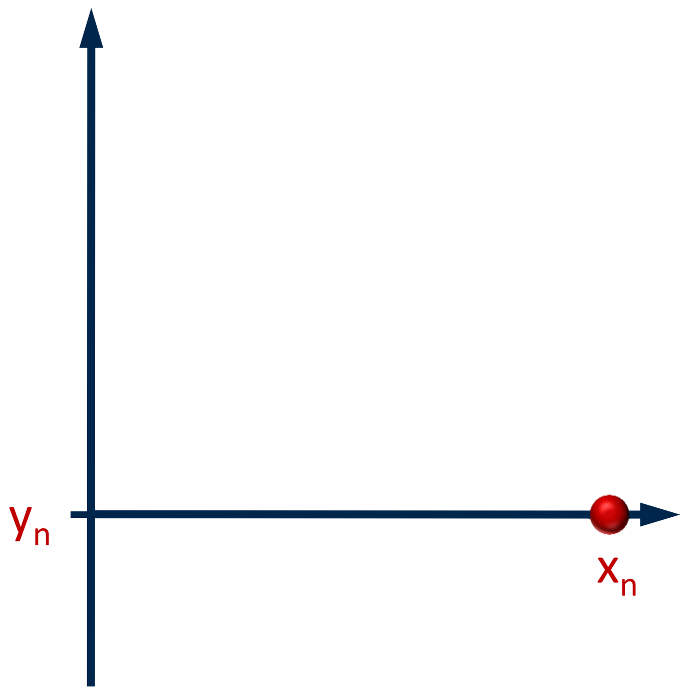


$d_2 = -1$ ($y_1 > 0$)

$x_2 = x_1 + y_1 /2$
$y_2 = y_1 - x_1 /2$

Repeat Step 2 for Each Rotation *k*

- **Until $y_n = 0$**



$y_n = 0$

$x_n = A_n \cdot (x_0^2 + y_0^2)^{0.5}$

accumulated gain

The Gain Factor

- Gain accumulation (when you don't multiply by K_i)

  G_0 = 1
  $G_0G_1 = 1.414$
  $G_0G_1G_2 = 1.581$
  $G_0G_1G_2G_3 = 1.630$
  $G_0G_1G_2G_3G_4 = 1.642$

- **So, start with x_0, y_0; end up with:**

  Shift & adds of x_0, y_0

  $z_4 = 71^o$
  $(x_0^2 + y_0^2)^{0.5} = 1.642$ (...)

- **We did the Cartesian-to-polar coordinate conversion**

Polar-to-Rectangular Conversion

Start with vector on x-axis

$A = |A| \cdot e^{j\phi}$

$x_0 = |A|$

$y_0 = 0,\ z_0 = \phi$

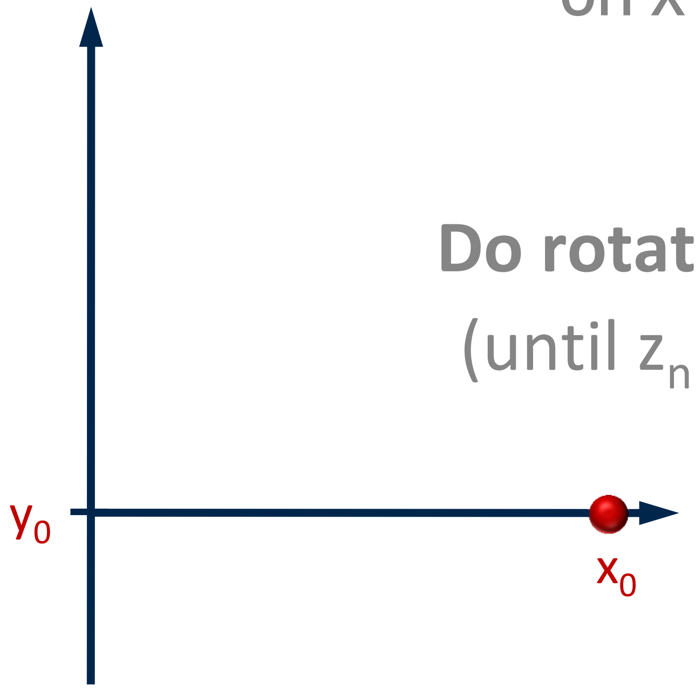
Do rotations

(until $z_n = 0$)

$\mathbf{z_i < 0,\ d_i = -1}$

$z_i > 0,\ d_i = +1$

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



CORDIC Algorithm

$$x_{i+1} = x_i - y_i \cdot d_i \cdot 2^{-i}$$
$$y_{i+1} = y_i + x_i \cdot d_i \cdot 2^{-i}$$
$$z_{i+1} = z_i - d_i \cdot \tan^{-1}(2^{-i})$$

$$d_i = \begin{cases} -1, & z_i < 0 \\ +1, & z_i > 0 \end{cases}$$

$$d_i = \begin{cases} -1, & y_i > 0 \\ +1, & y_i < 0 \end{cases}$$

Rotation mode	**Vectoring mode**
(rotate by specified angle) **Minimize residual angle**	(align with the x-axis) **Minimize y component**

Rotation mode:

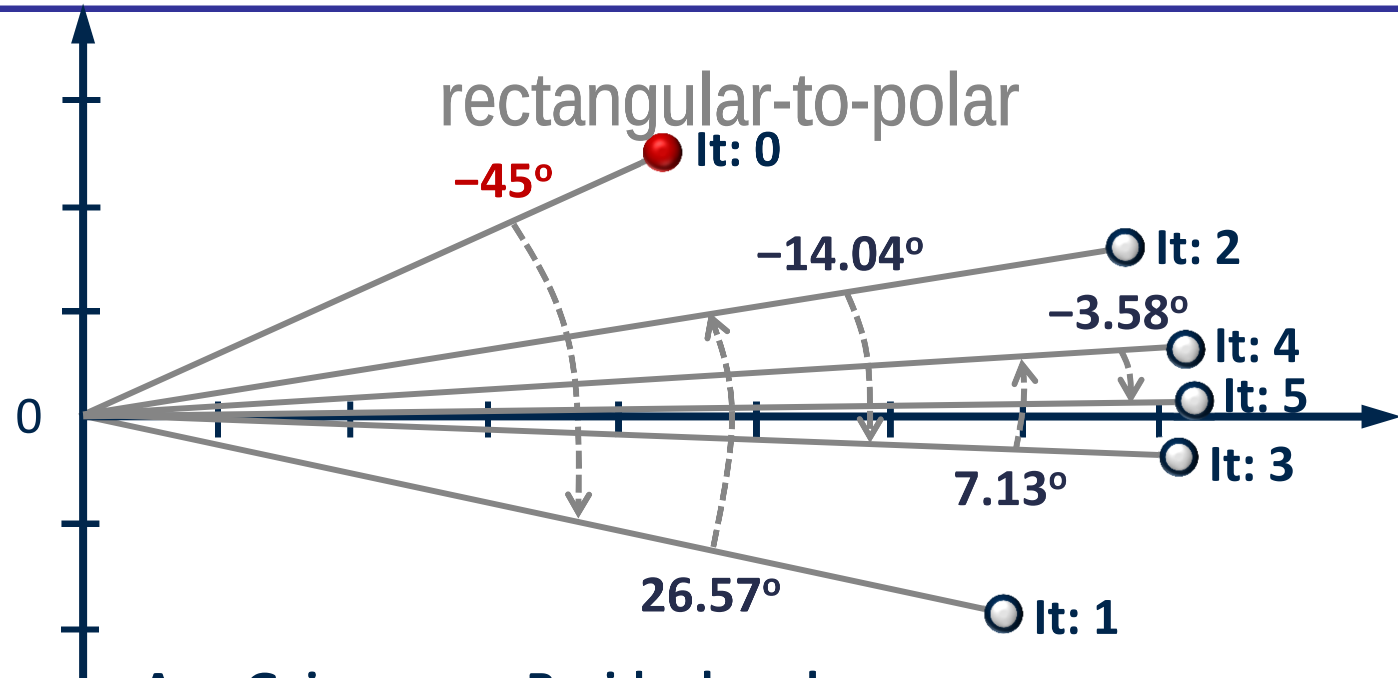
$$x_n = A_n \cdot [x_0 \cdot \cos(z_0) - y_0 \cdot \sin(z_0)]$$
$$y_n = A_n \cdot [y_0 \cdot \cos(z_0) + x_0 \cdot \sin(z_0)]$$
$$z_n = 0$$

Vectoring mode:

$$x_n = A_n \cdot (x_0^2 + y_0^2)^{0.5}$$
$$y_n = 0$$
$$z_n = z_0 + \tan^{-1}(y_0/x_0)$$

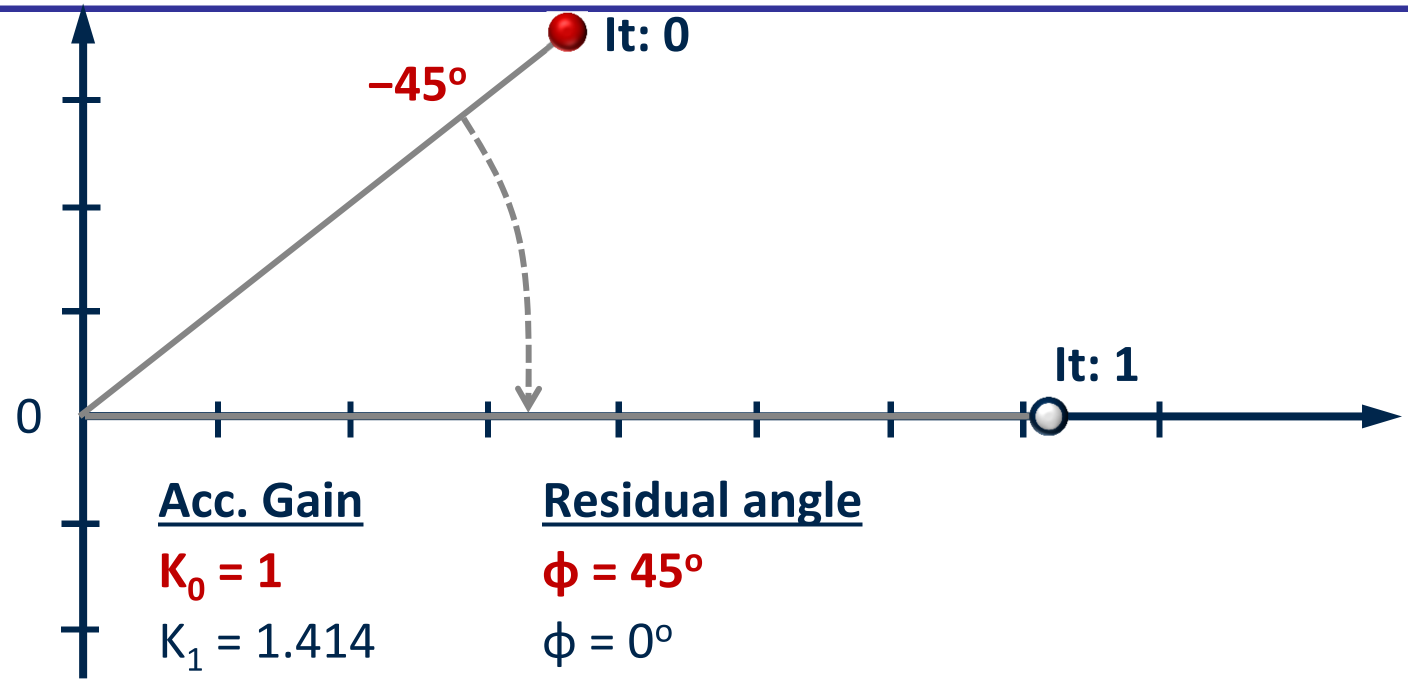
$$A_n = \prod^{n}(1+2^{-2i}) \cdot 0.5 \rightarrow 1.647$$

Vectoring Example



Acc. Gain	Residual angle
$K_0 = 1$	$\phi = 30^o$
$K_1 = 1.414$	$\phi = -15^o$
$K_2 = 1.581$	$\phi = 11.57^o$
$K_3 = 1.630$	$\phi = -2.47^o$
$K_4 = 1.642$	$\phi = 4.65^o$
$K_5 = 1.646$	$\phi = 1.08^o$
Etc.	Etc.

Vectoring: Best-Case Convergence

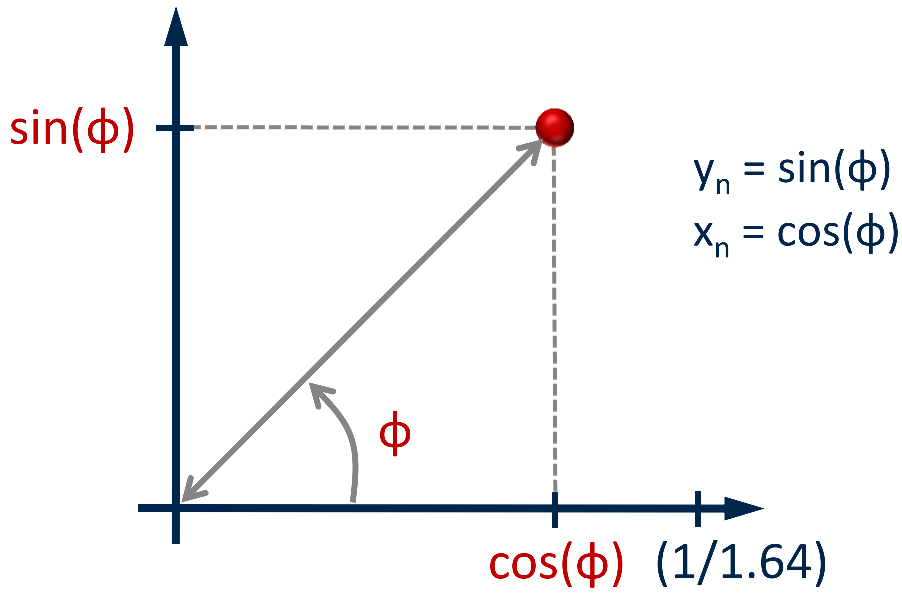


Acc. Gain	Residual angle
$K_0 = 1$	$\phi = 45^\circ$
$K_1 = 1.414$	$\phi = 0^\circ$
$K_2 = 1.581$	$\phi = 0^\circ$
$K_3 = 1.630$	$\phi = 0^\circ$
$K_4 = 1.642$	$\phi = 0^\circ$
$K_5 = 1.646$	$\phi = 0^\circ$
Etc.	

In the best case ($\varphi = 45^\circ$), we can converge in 1 iteration

Calculating Sine and Cosine

- Start with $\boldsymbol{x}_0 = 1/1.64$, $\boldsymbol{y}_0 = 0$
- Rotate by $\boldsymbol{\varphi}$



Functions

Rotation mode

sin/cos

z_0 = angle

$y_0 = 0,\ x_0 = 1/A_n$

$x_n = \boxed{A_n \cdot x_0} \cdot \cos(z_0)$

$y_n = \boxed{A_n \cdot x_0} \cdot \sin(z_0)$

(=1)

Vectoring mode

$\tan^{-1}$

$z_0 = 0$

$z_n = z_0 + \tan^{-1}(y_0/x_0)$

Vector/Magnitude

$x_n = A_n \cdot (x_0^2 + y_0^2)^{0.5}$

Polar → Rectangular

$x_n = r \cdot \cos(\phi)$

$y_n = r \cdot \sin(\phi)$

$x_0 = r$

$z_0 = \phi$

$y_0 = 0$

Rectangular → Polar

$r = (x_0^2 + y_0^2)^{0.5}$

$\phi = \tan^{-1}(y_0/x_0)$

CORDIC Divider

- To do a divide, change CORDIC rotations to a linear function calculator

$$x_{i+1} = x_i - \boxed{0} \cdot y_i \cdot d_i \cdot 2^{-i} = x_i$$

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

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

Generalized CORDIC

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

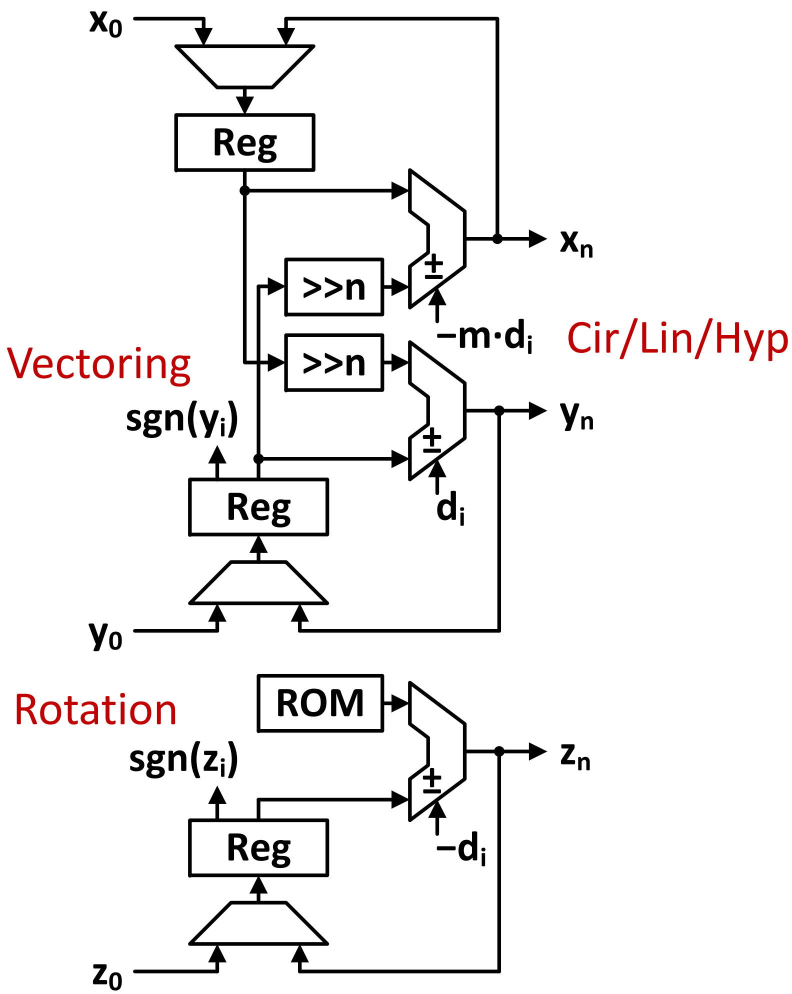
$$y_{i+1} = y_i + x_i \cdot d_i \cdot 2^{-i}$$

$$z_{i+1} = z_i - d_i \cdot \boxed{e_i}$$

d_i	
Rotation	**Vectoring**
$d_i = -1, z_i < 0$ $d_i = +1, z_i > 0$	$d_i = -1, y_i > 0$ $d_i = +1, y_i < 0$
$\mathrm{sign}(z_i)$	$-\mathrm{sign}(y_i)$

Mode	m	e_i
Circular	+1	$\tan^{-1}(2^{-i})$
Linear	0	2^{-i}
Hyperbolic	−1	$\tanh^{-1}(2^{-i})$

An FPGA Implementation

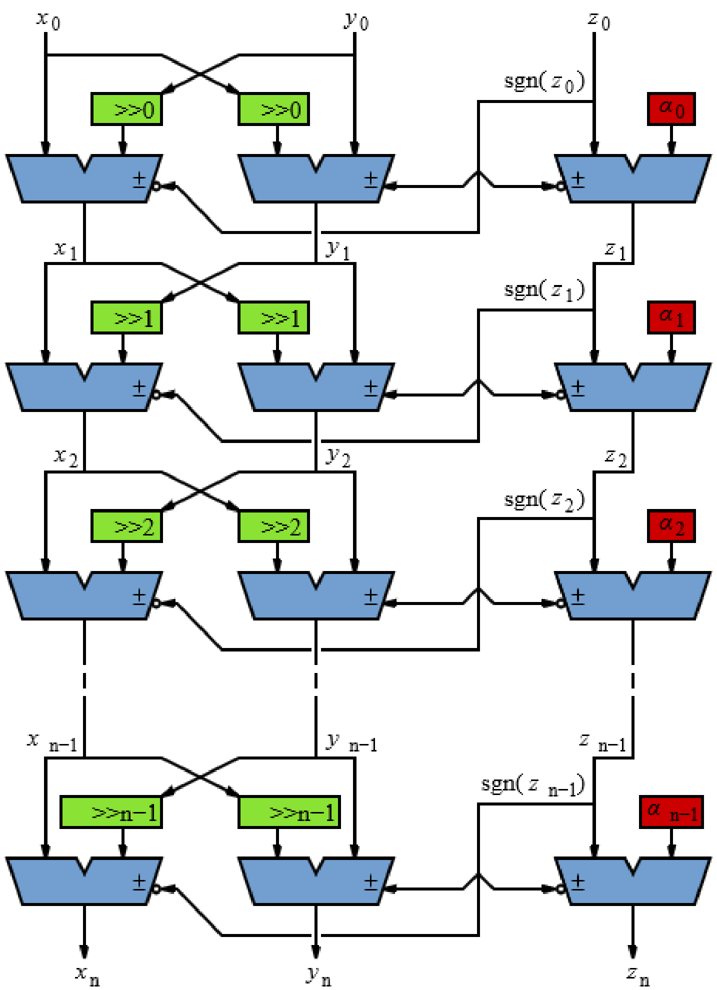


R. Andraka, "A survey of CORDIC algorithms for FPGA based computers," in Proc. Int. Symp. FPGAs, Feb. 1998, pp. 191-200.

- Three difference equations directly mapped to hardware
- The decision d_i is driven by the sign of the y or z register
  - Vectoring: $d_i = -sign(y_i)$
  - Rotation: $d_i = sign(z_i)$
- The initial values loaded via muxes
- On each clock cycle
  - Register values are passed through shifters and add/sub and the values placed in registers
  - The shifters are modified on each iteration to cause the desired shift (state machine)
  - Elementary angle stored in ROM
- Last iteration: results read from reg

Unrolled CORDIC

- Single-cycle loop - unroll



Next Lecture

- Processor interfacing