

# DSP Design – Lecture 2

Number representation, scaling, quantization  
and round-off Noise

**Steffen Malkowsky**

**[steffen.malkowsky@eit.lth.se](mailto:steffen.malkowsky@eit.lth.se)**



# Representation of Numbers

# Numbers is a way to use symbols to describe and model the world!

However, numbers may be represented in many different ways and there are considerations to make when it comes to digital signal processing!



# Definition: Radix

In a **mathematical numeral system**, the **radix** or **base** is the number of **unique digits**, including zero, used to represent numbers in a **positional numeral system**.

Example: In the decimal system (the most common system in use today) the **radix is 10**, because it uses the ten digits from 0 through 9. The number 47 may then be written as:  **$47_{10}$**

**0 1 2 3 4 5 6 7 8 9, 10 11 12 13 14 ...**



**Base 10**

## Definition: Radix

Each position have its on “weight” depending on the current base.

$$125_{10} = 1*100 + 2*10 + 5*1 = 1*10^2 + 2*10^1 + 5*10^0$$

Representing fractions is a simple extension of this idea.

$$25.43_{10} = 2*10 + 5*1 + 4*0.1 + 3*0.01 = 2*10^1 + 5*10^0 + 4*10^{-1} + 3*10^{-2}$$

# Unsigned Number Representation

General description of a fixed radix (base) systems  
(fixed point positional number system)

The digits  $a \in \{0, 1, 2, \dots, r-1\}$  in a radix  $r$  system:

$$\sum_{i=k-1}^{-l} r^i \times a_i =$$

digit set

$$= r^{k-1} a_{k-1} + r^{k-2} a_{k-2} \cdots r^1 a_1 + r^0 a_0 + r^{-1} a_{-1} \cdots r^{-l} a_{-l}$$

described in a fixed point positional number system:

$$\underbrace{a_i a_{i-1} \cdots a_1 a_0}_{\text{Whole part}} \cdot \underbrace{a_{-1} \cdots a_{-l}}_{\text{Fractional part}}$$

# What about numbers in DSP?

In DSP, numbers are encoded by means of **binary digits** or **bits {0,1}**.

For binary digits the base = 2.

All (base 10) numbers can be represented by groups of bits, for example

$$86_{10} = 1 * 2^6 + 0 * 2^5 + 1 * 2^4 + 0 * 2^3 + 1 * 2^2 + 1 * 2^1 + 0 * 2^0$$

or

$$86_{10} = 1010110_2$$

# Binary numbers (base=2)

## Representing unsigned integers

**MSB =**  
**Most Significant Bit**

**LSB =**  
**Least Significant Bit**

$2^2$	$2^1$	$2^0$	In base 10
0	0	0	(0)
0	0	1	(1)
0	1	0	(2)
0	1	1	(3)
1	0	0	(4)
1	0	1	(5)
1	1	0	(6)
1	1	1	(7)

**N bits**  
↓  
 **$2^N$  words**

**Ex.  $101_2 = 5_{10}$**



# Example: Unsigned Number

$$\sum_{i=k-1}^{-l} 10^i a_i = \{a \in \{0, 1, 2, \dots, 9\} \text{ in radix } 10\}$$

Radix 10 = decimal system

$$= 10^{k-1} a_{k-1} + 10^{k-2} a_{k-2} \cdots 10^1 a_1 + 10^0 a_0 + 10^{-1} a_{-1} \cdots 10^{-l} a_{-l}$$

$$\sum_{i=k-1}^{-l} 2^i a_i = \{a \in \{0, 1\} \text{ in radix } 2\}$$

Radix 2 = binary system

$$= 2^{k-1} a_{k-1} + 2^{k-2} a_{k-2} \cdots 2^1 a_1 + 2^0 a_0 + 2^{-1} a_{-1} \cdots 2^{-l} a_{-l}$$

# Signed Numbers and Fractions

But we also need to represent numbers like  $-34_{10}$  and  $25.43_{10}$  in binary.

Devote 1 bit to indicate sign:

$$1101\ 1101_2 = -128 + 64 + 0 + 16 + 8 + 4 + 0 + 1 = -35.$$

Fractions can be expressed like this

$$0.625_{10} = 1*0.5 + 0*0.25 + 1*0.125 = 1* 2^{-1} + 0* 2^{-2} + 1* 2^{-3} = 0.101_2$$

**However there are also other possibilities!**

# Number Representation

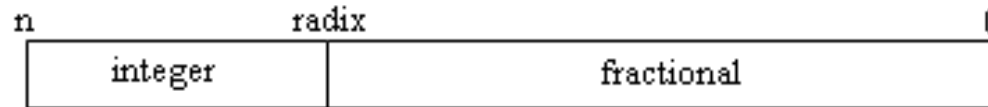
Fixed point & Floating point

# Fixed point & Floating point

- Digital signal processing number representation can be separated into two categories - *fixed point* and *floating point*. These designations refer to the format used to store and manipulate numeric representations of data.
- *Fixed-point DSPs* are designed to represent and manipulate integers – positive and negative whole numbers.
- *Floating-point DSPs* represent and manipulate rational numbers.

# Fixed point

- In fixed point we only have a single value ( $n$ )
- In fixed-point representation, a specific radix point (decimal point in English “.”) is chosen so there is a fixed number of bits to the right (fractional bits) and a fixed number of bits to the left (integer bits) of the radix point.



# Floating point

In floating point a value is represented by

- mantissa ( $m$ ) determining the resolution/precision
- exponent ( $e$ ) determining the dynamic range



$$m \times b^e$$

In floating-point representation, the placement of the decimal point can 'float' relative to the significant digits of the number.

# Floating vs. Fixed point - Design Considerations

With fixed-point notation, the gaps between adjacent numbers are always equal a value of one, whereas in floating-point notation, gaps between adjacent numbers are not uniformly spaced (ANSI/IEEE Std. 754 standard format), with large gaps between large numbers and small gaps between small numbers.

Floating point gives higher dynamic range but often high cost in:

- energy
- area
- calculation time

For energy efficient implementations fixed point is preferred, however fixed-point numbers are more susceptible to common numeric computational inaccuracies.

When choosing number representation you also need to consider

- Cost
- Performance
- Ease of implementation/development
- Cell library?

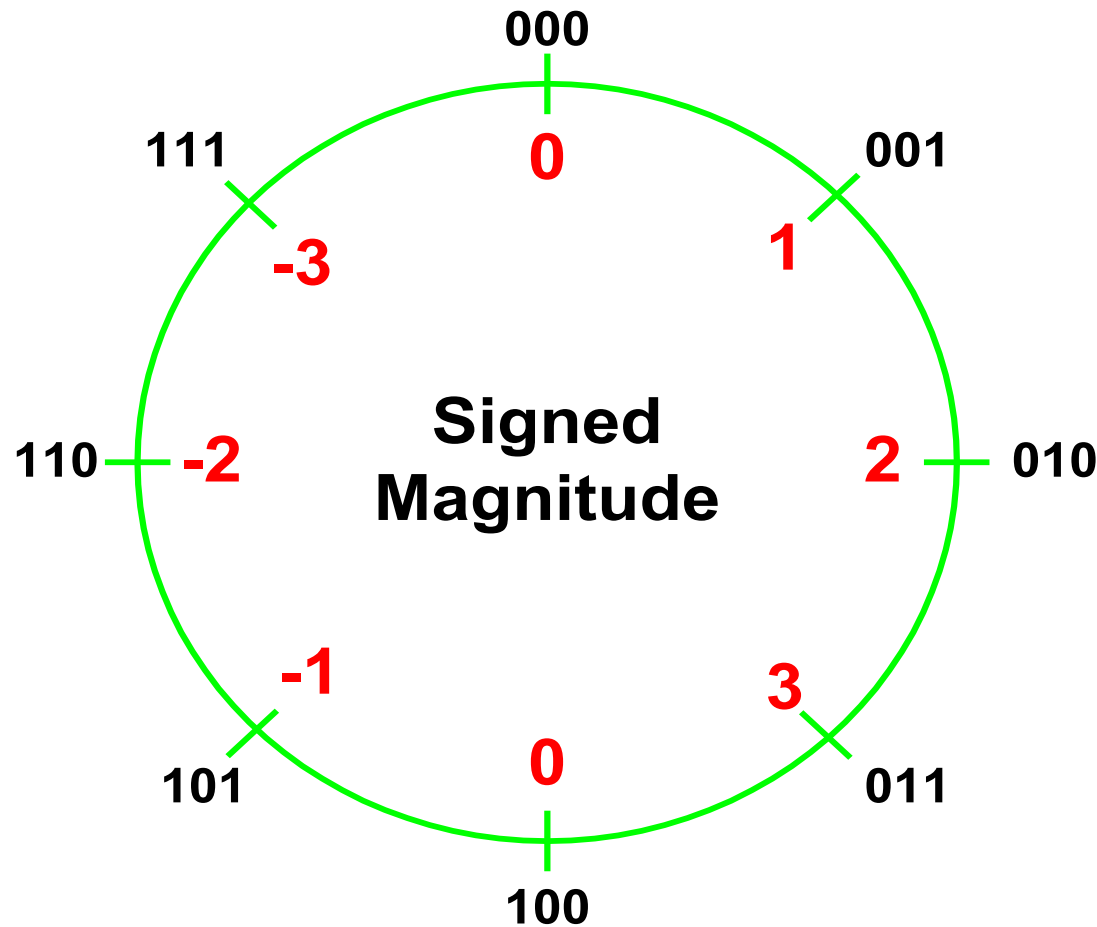
# There are different fixed point numbering systems!

Sign Magnitude  
One's Complement  
Two's Complement  
Etc.



# Signed Magnitude

Unsigned numbers with a sign-bit (MSB = sign bit)



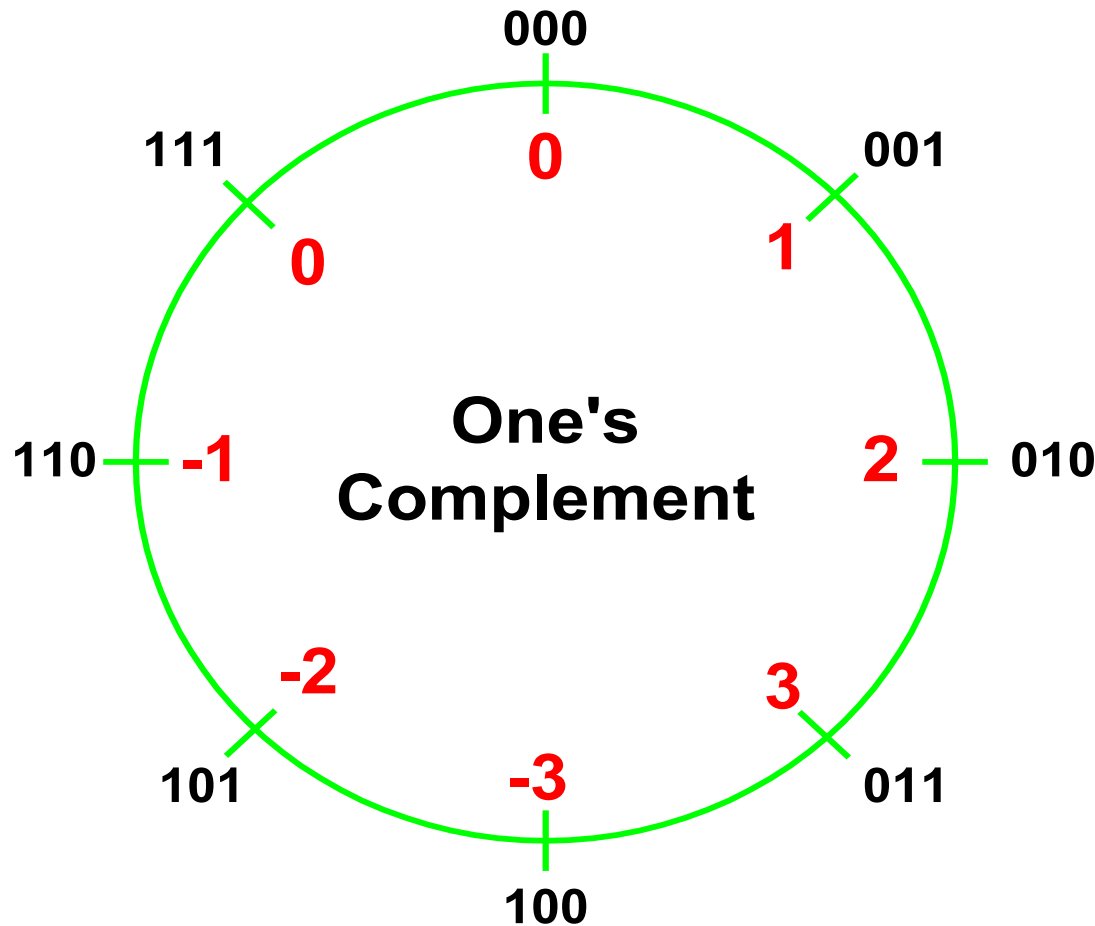
- Two Zeros

+ Low Power?

+ Easy to convert to Negative (flip bit)

# One's Complement

Signed numbers by inverting (Complement)

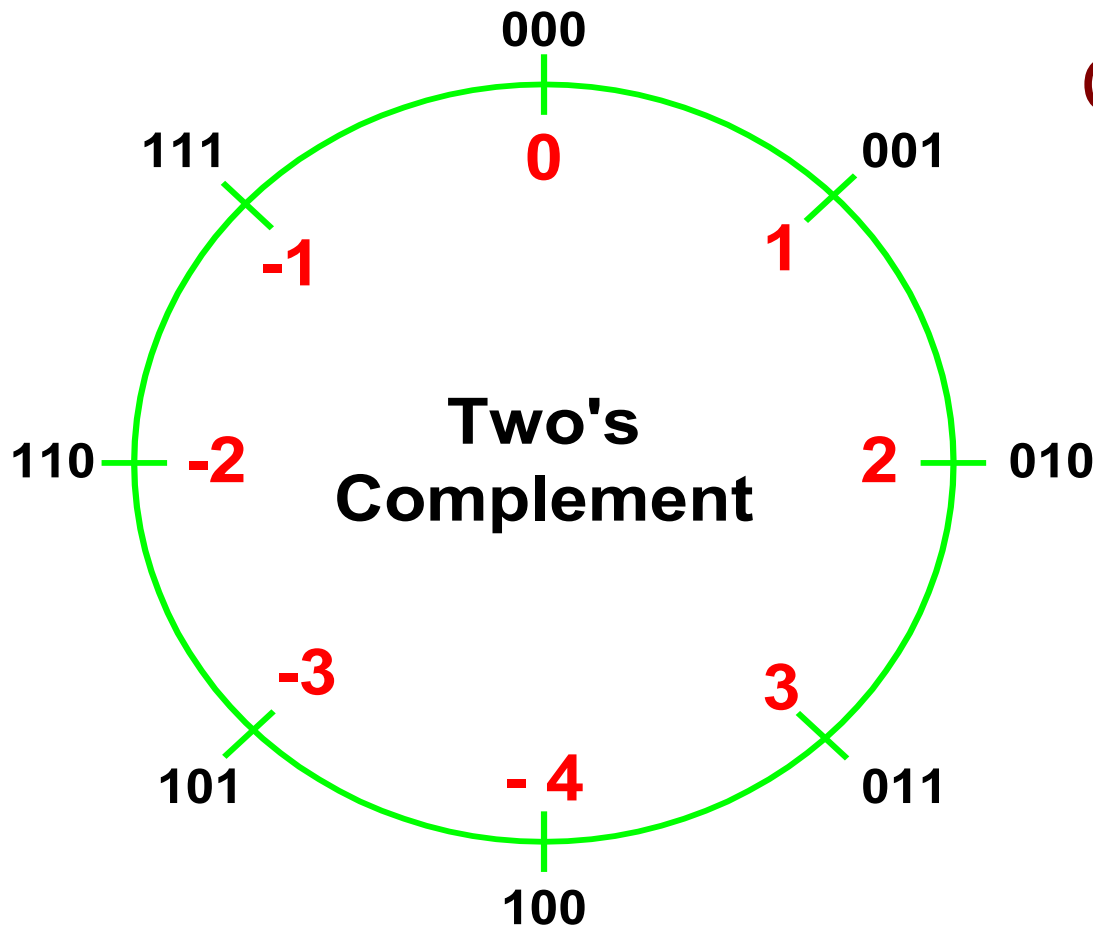


- Two Zeros

+ Easy to convert to Negative (invert)

# Two's Complement

Most widely used fixed point numbering system



Complement + LSB  
+ One Zero

+ Easy Add/Sub

Add operation for negative numbers is identical to the add operation for positive numbers, so no additional logic is required to handle the negative case.

- Not so easy to convert to Neg.

# Table of number representations

Binary	Unsigned	Sign and magnitude	Ones' complement	Two's complement	Excess-8	Base -2
0000	0	0	0	0	-8	0
0001	1	1	1	1	-7	1
0010	2	2	2	2	-6	-2
0011	3	3	3	3	-5	-1
0100	4	4	4	4	-4	4
0101	5	5	5	5	-3	5
0110	6	6	6	6	-2	2
0111	7	7	7	7	-1	3
1000	8	-0	-7	-8	0	-8
1001	9	-1	-6	-7	1	-7
1010	10	-2	-5	-6	2	-10
1011	11	-3	-4	-5	3	-9
1100	12	-4	-3	-4	4	-4
1101	13	-5	-2	-3	5	-3
1110	14	-6	-1	-2	6	-6
1111	15	-7	-0	-1	7	-5

Wikipedia

# Considerations

- **Much to consider when choosing between fixed point or floating point representation. Type of application, development time, cost, tools, etc. will influence your choice!**
- **Choosing different fixed point numbering systems will also influence your design!**

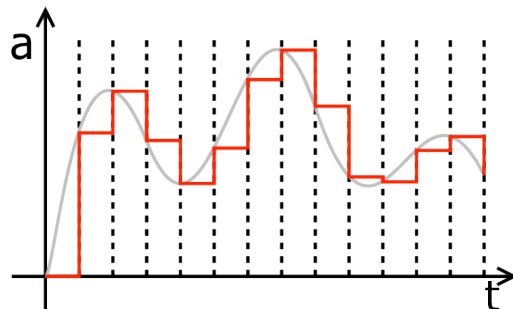
# Quantization

**Ch. 11.1 – 11.4 in Phari**

**Overview of Effects of Finite Register  
Length in DSP** (download from homepage)

# What is Quantization?

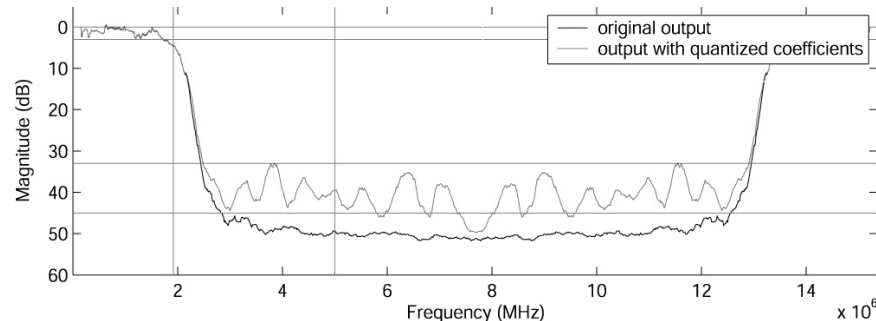
- **Quantization**, in mathematics and in DSP, is the process of mapping a large set of input values to a (countable) smaller set.
- The operations *rounding* and *truncation* are typical examples of quantization processes.
- Quantization is involved to some degree in nearly all digital signal processing, as the process of representing a signal in digital form ordinarily involves rounding.



# Quantization

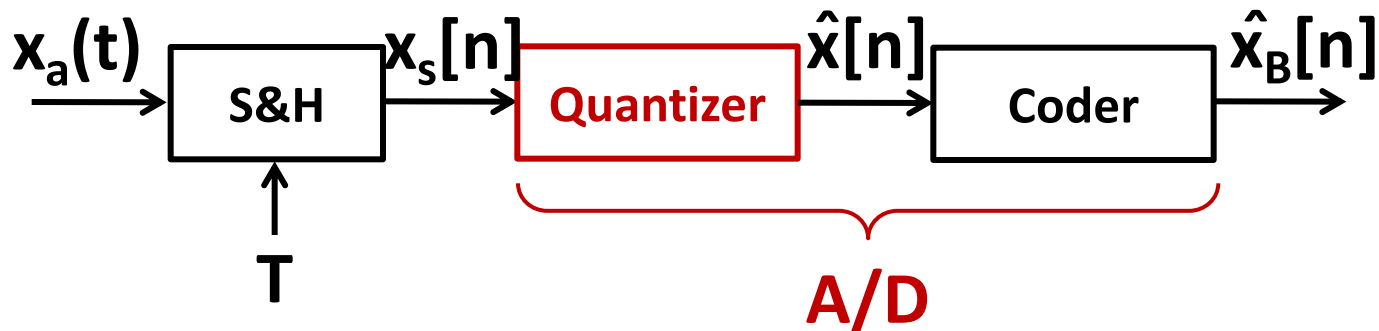
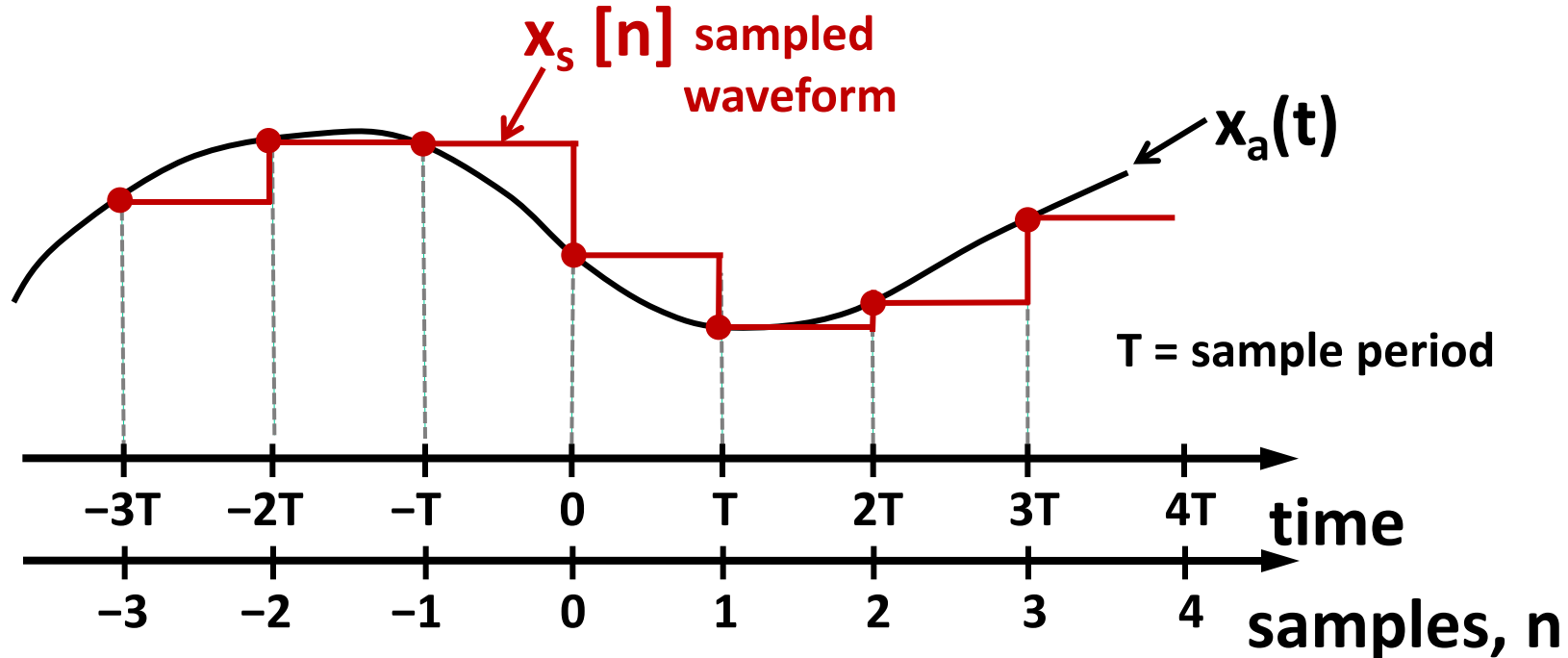
Mainly two types:

- **Signal Quantization**
- **Coefficient Quantization**

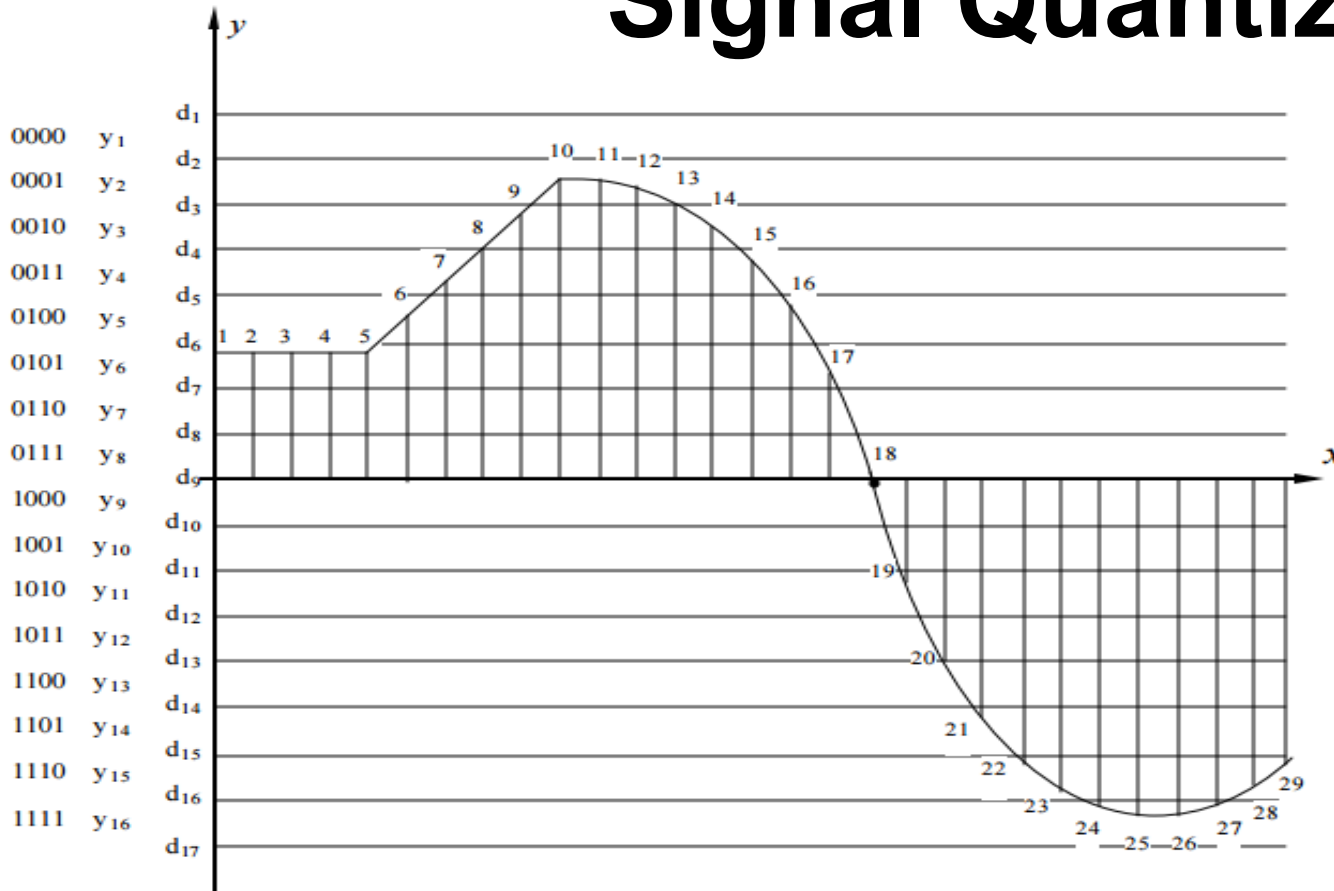




# Signal Quantization



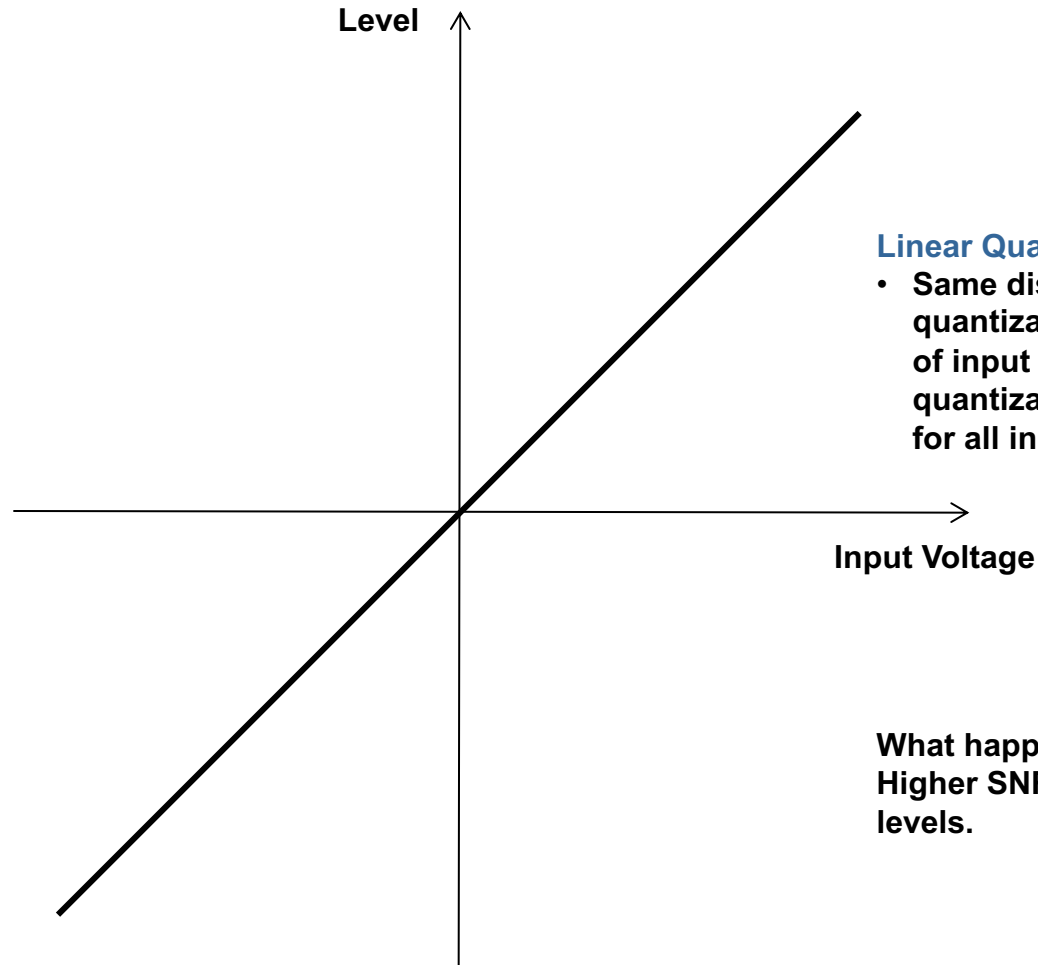
# Signal Quantization



Output code (from left to right, from top to bottom):

0101	0101	0101	0101	0101	0100	0011	0011	0010	0001	0001	0001	0010	0010	0011
0100	0101	1000	1010	1100	1101	1110	1110	1111	1111	1111	1111	1110	1110	

# Different quantization techniques can be applied

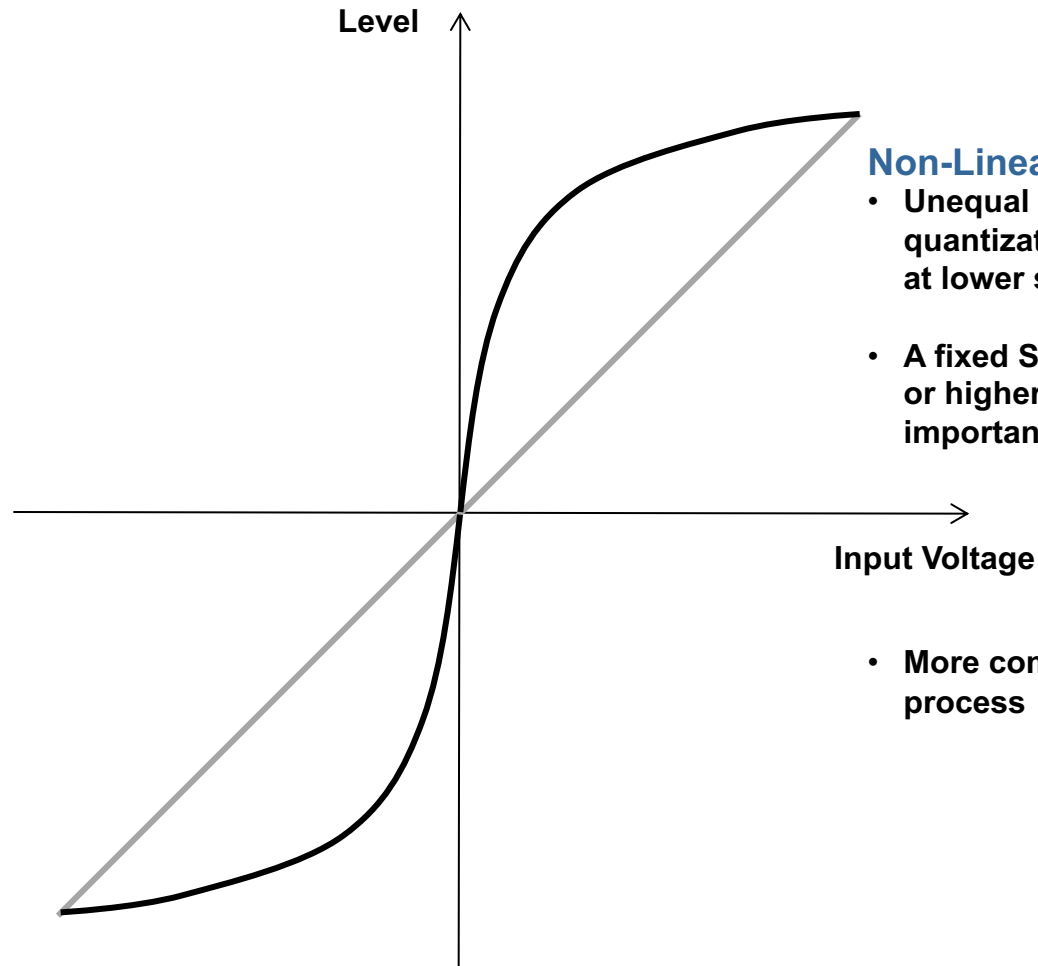


## Linear Quantization:

- Same distance between quantization levels independent of input voltage, i.e. quantization error is the same for all input signal.

What happens with the SNR?  
Higher SNR for high signal levels.

# Different quantization techniques can be applied



## Non-Linear Quantization:

- Unequal distance between quantization, e.g. closer levels at lower signal levels.
- A fixed SNR can be achieved or higher SNR in most important areas.
- More complex quantization process

# Quantization effects

## Round-off Noise

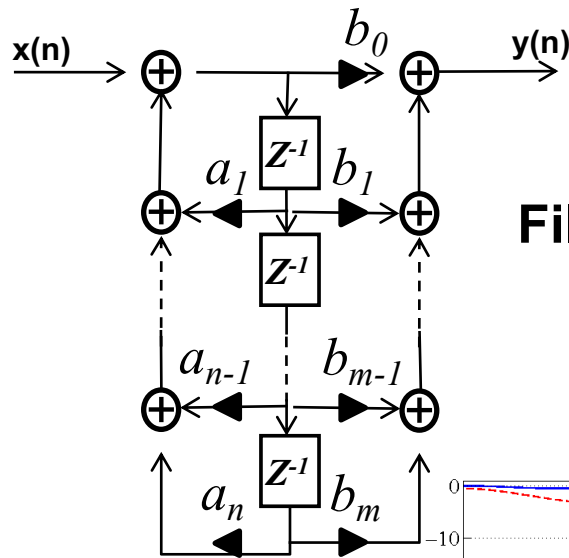
- Affect the output as a random disturbance

## Limit Cycle Oscillations

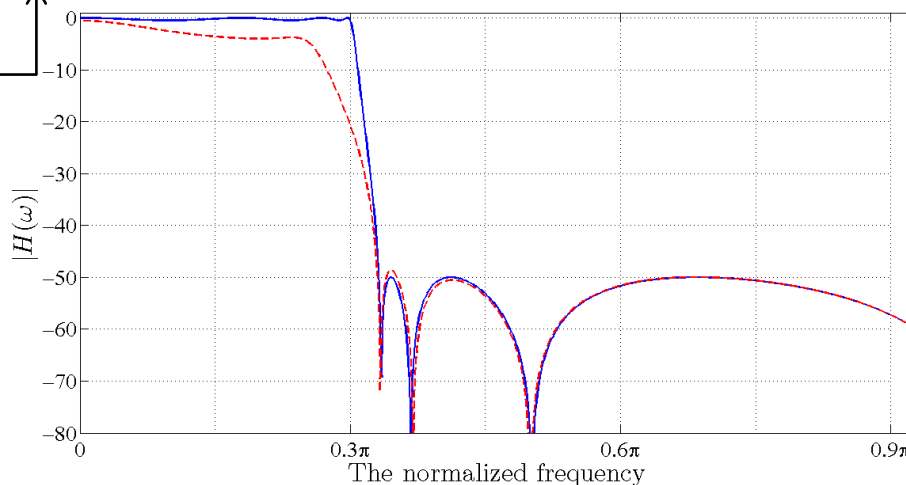
- Undesired periodic components due to non-linear behavior in the feedback (rounding or overflow)

# Coefficient Quantization

IIR filter

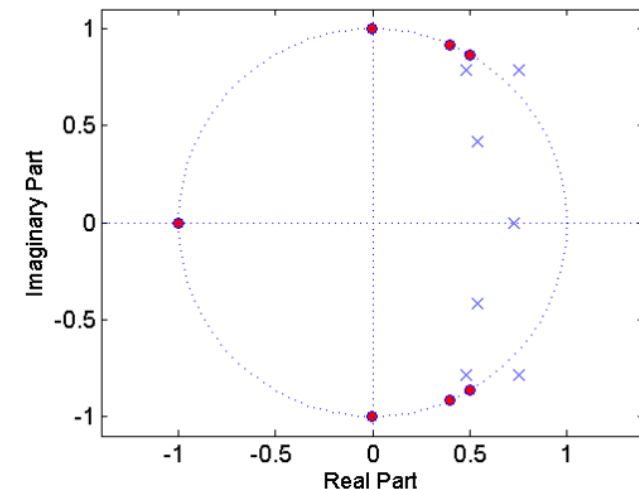


**Filter becomes unstable!  
Non-ideal transfer  
function!**



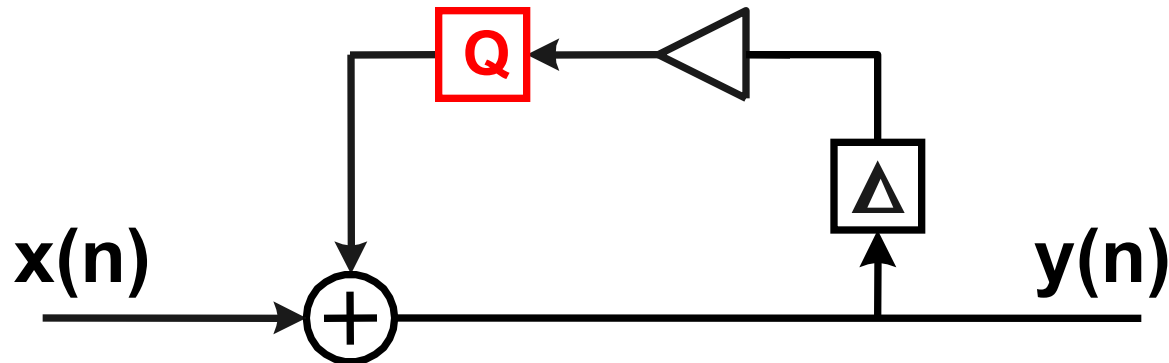
k	$b_k(\text{unquantized})$	$a_k(\text{unquantized})$
0	0.012218357882143	1.000000000000000
1	-0.009700754662078	-4.288900601525732
2	0.024350450826845	9.216957436091198
3	0.002532504848041	-12.195350561406707
4	0.002532504848041	10.633166152311462
5	0.024350450826845	-6.062798190498858
6	-0.009700754662078	2.098067018562072
7	0.012218357882143	-0.342340135743532

k	$b_k(\text{quantized})$	$a_k(\text{quantized})$
0	0.0122	1.0000
1	-0.0097	-4.2813
2	0.0244	9.2188
3	0.0025	-12.1875
4	0.0025	10.6250
5	0.0244	-6.0625
6	-0.0097	2.0938
7	0.0122	-0.3438



# Rounding & Truncation

Rounding/Truncation is “always” there!  
Especially necessary in recursive systems

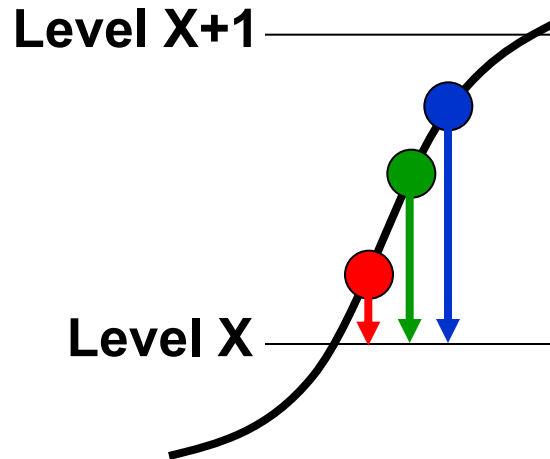


Without quantization - infinite wordlength

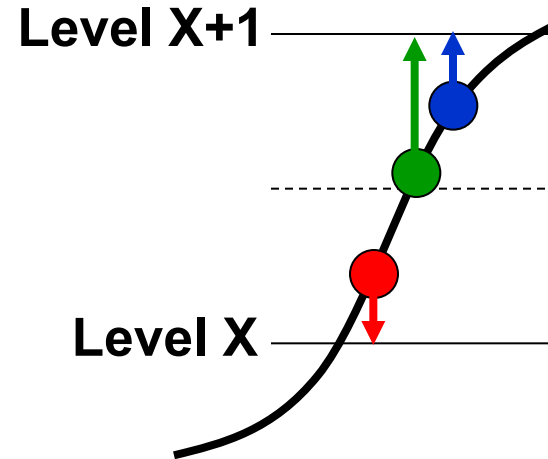
Multiplication  $\Rightarrow n+m$  output bits

Addition  $\Rightarrow n+1$  output bits

# Truncation & Rounding



**Truncation**  
All values approximated  
in the same direction  
  
Max error = 1LSB



**Rounding**  
Values approximated  
up or down  
  
Max error = 1/2 LSB



# Different rounding schemes

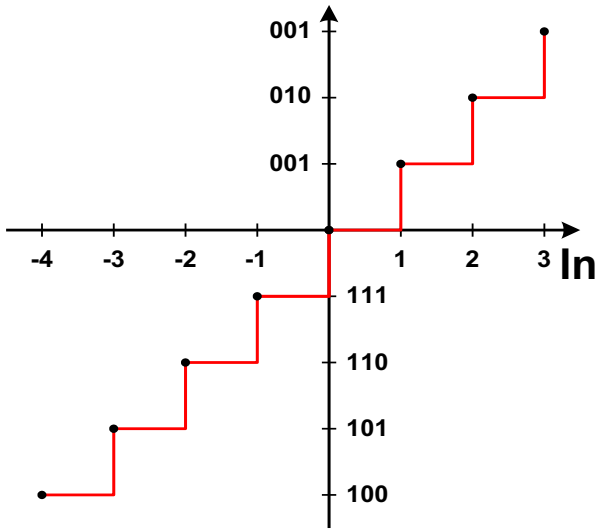
$y$	round down (towards $-\infty$ )	round up (towards $+\infty$ )	round towards zero	round away from zero	round to nearest
+23.67	+23	+24	+23	+24	+24
+23.50	+23	+24	+23	+24	+24
+23.35	+23	+24	+23	+24	+23
+23.00	+23	+23	+23	+23	+23
0	0	0	0	0	0
-23.00	-23	-23	-23	-23	-23
-23.35	-24	-23	-23	-24	-23
-23.50	-24	-23	-23	-24	-24
-23.67	-24	-23	-23	-24	-24

# Rounding & Truncation

No energy added to the system

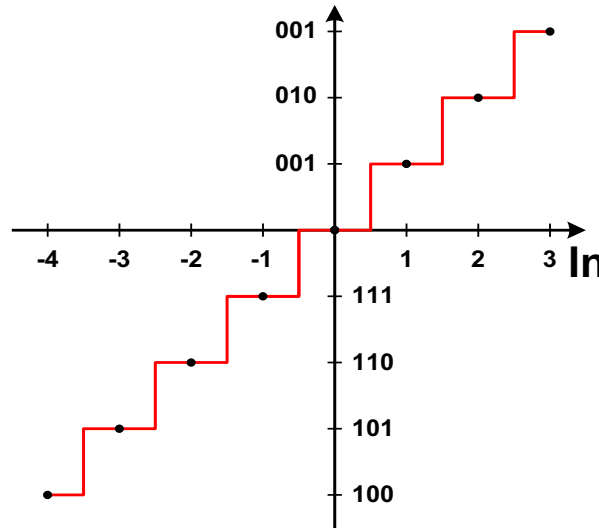
Often used in recursive algorithms

Truncation



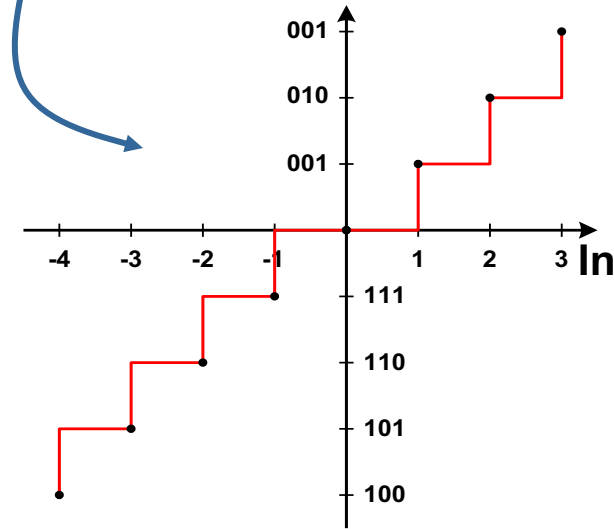
DC error  
All values goes towards -infinity

Rounding



”Rounded to even”

Truncation towards zero



Add LSB before truncation if negative

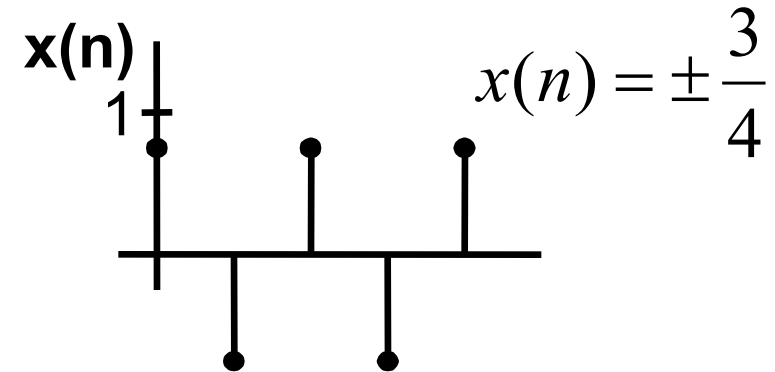
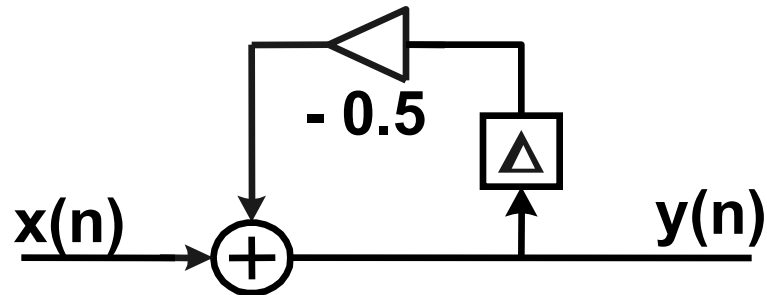
# Scaling

**Scaling is about adjusting the signal range to fit the hardware.**

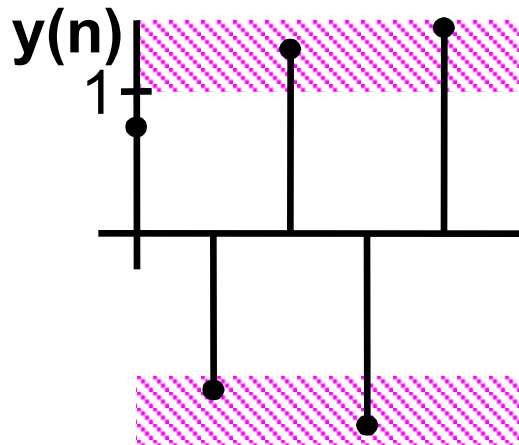
**Unchanged transfer function (scaled coefficients might move the pole-zeros)**

**However you might loose in precision!**

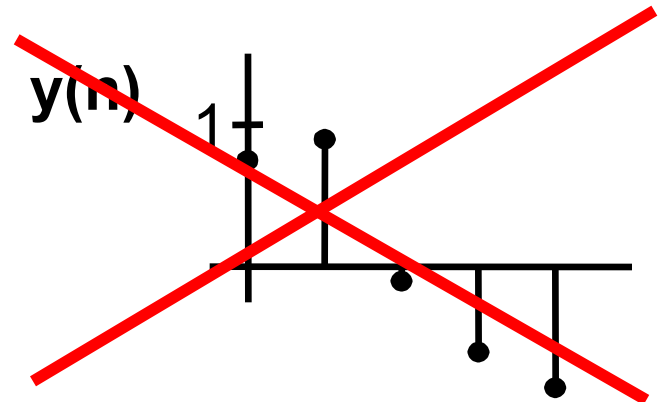
# An example where scaling is needed



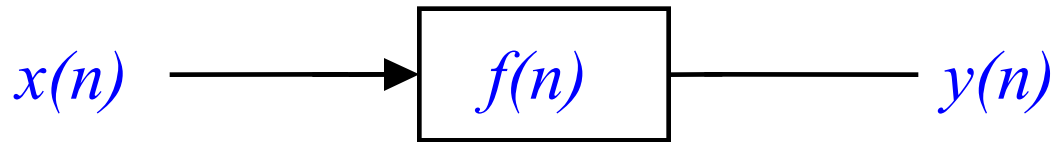
$$-1 \leq x, y < 1$$



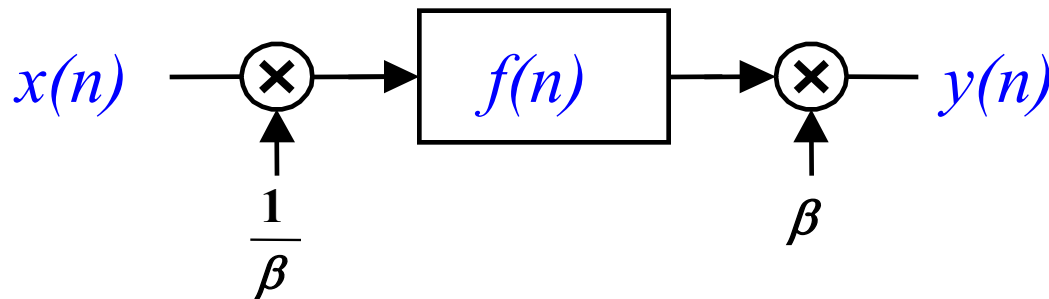
**Overflow**  $\Rightarrow$



# Scaling to avoid overflow



**Safe scaling if**



$$\beta = \sum_{i=0}^{\infty} |f(i)|$$

( $l_1$  – scaling in the book)

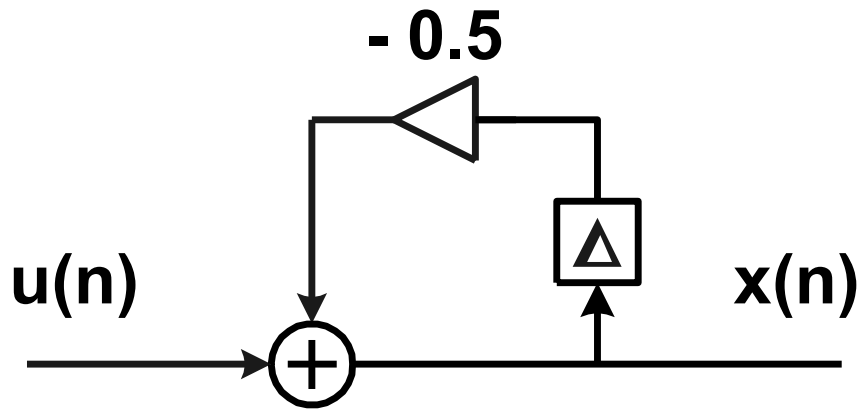
**Where  $f(i)$  is the unit sample response**

# Example 1: Safe Scaling

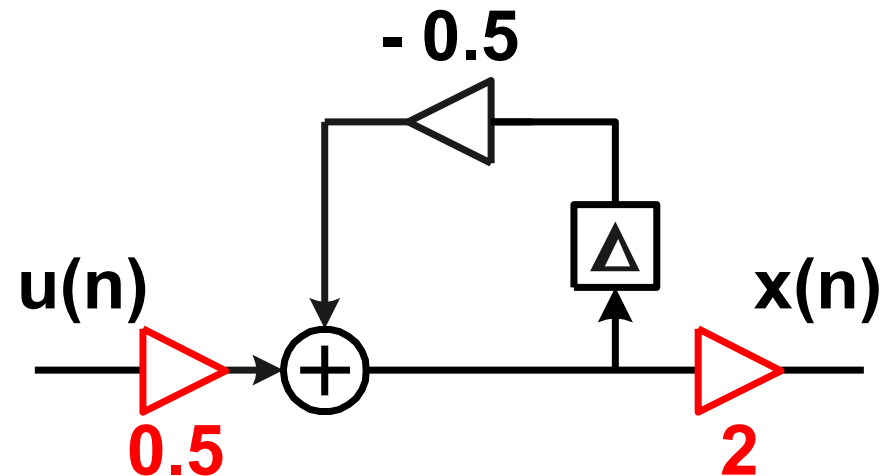
$$\beta = \sum_{i=0}^{\infty} |f(i)| = \sum_{i=0}^{\infty} |(-0.5)^i| =$$

Geometric series

$$|(-0.5)^0| + |(-0.5)^1| + |(-0.5)^2| + \dots = \frac{1}{1 - 0.5} = 2$$



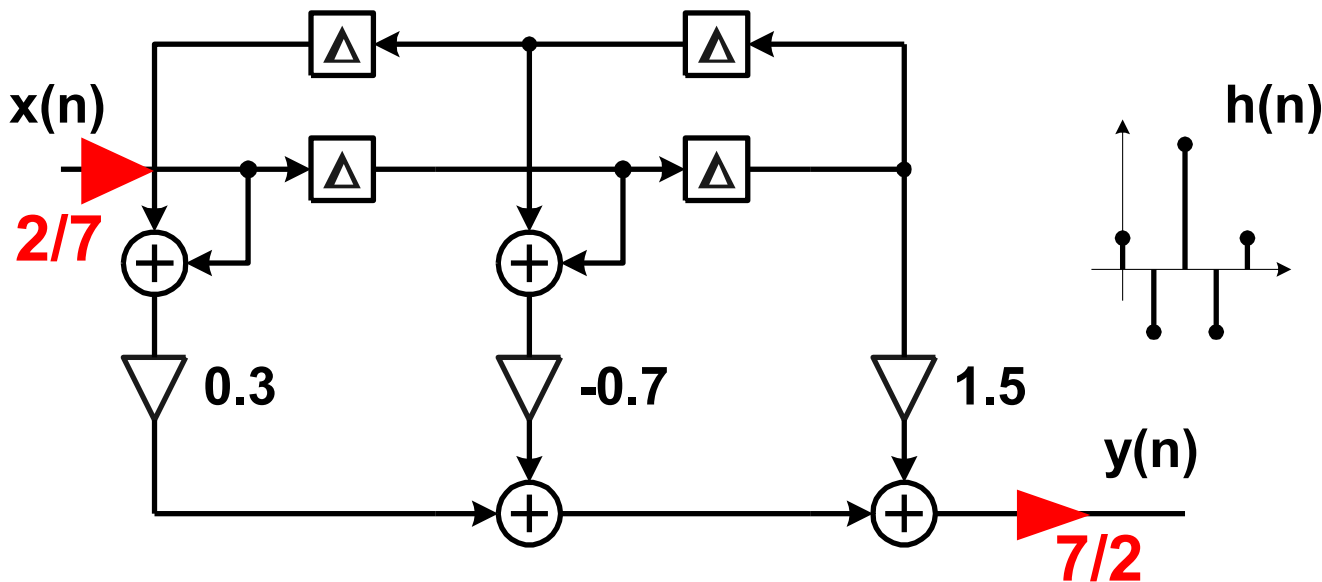
Original filter with overflow



# Example 2: Safe Scaling $\frac{2}{7}x(n)$ and $\frac{7}{2}y(n)$

*give safe scaling*

$$\beta = \sum_{i=0}^{\infty} |f(i)| = 0.3 \times 2 + |-0.7 \times 2| + 1.5 = 3.5$$



**Increased roundoff noise**  
**Internal scaling might improve**

**(Linear phase FIR)**

# Scaling

- **Safe scaling is pessimistic**

- Alternative is scaling with

$$\beta = \sqrt{\sum_{i=0}^{\infty} (|f(i)|^2)} \quad (\ell_2 - \text{scaling in the book})$$

- **In practice: Scaling with  $\beta = 2^{\pm n}$**

- Easy to do - a shift

- **Increased internal wordlength an alternative**

**However, overflow may occur!!**



# Summary Scaling Techniques

$$l_1 \text{ - scaling } \quad \beta = \sum_{i=0}^{\infty} |f(i)|, \textit{ Safe - scaling}$$

$$l_2 \text{ - scaling } \quad \beta = \delta \sqrt{\sum_{i=0}^{\infty} f^2(i)}, \textit{ possible overflow}$$

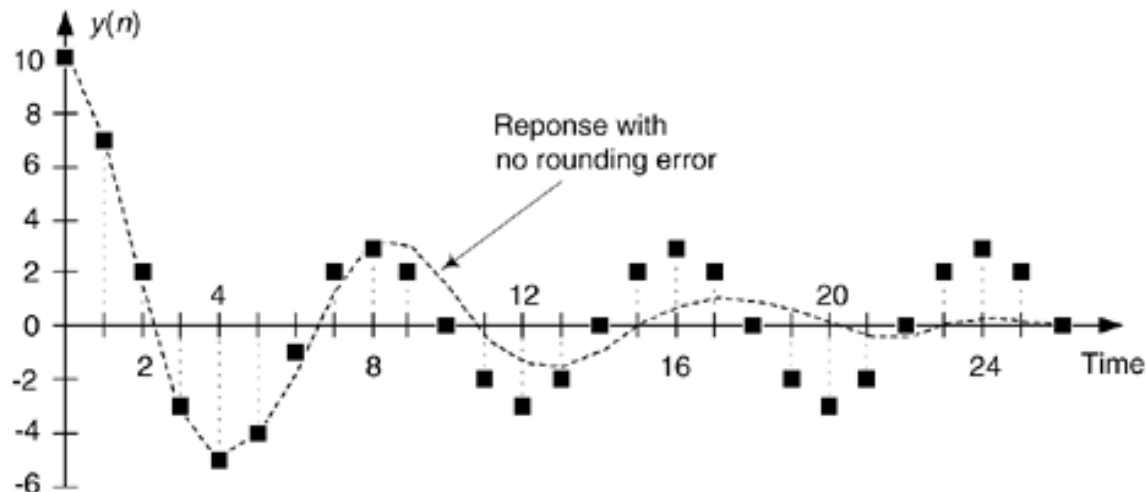
$f(i)$  = unit sample response,  $\sum_{i=0}^{\infty} f^2(i)$  = Variance white noise input

- “Safe scaling” but not guaranteed
- $\delta$  sets the probability for an overflow
- Typically one overflow every  $10^6$  sample is accepted in audio [Wanhammar]

# Limit Cycles

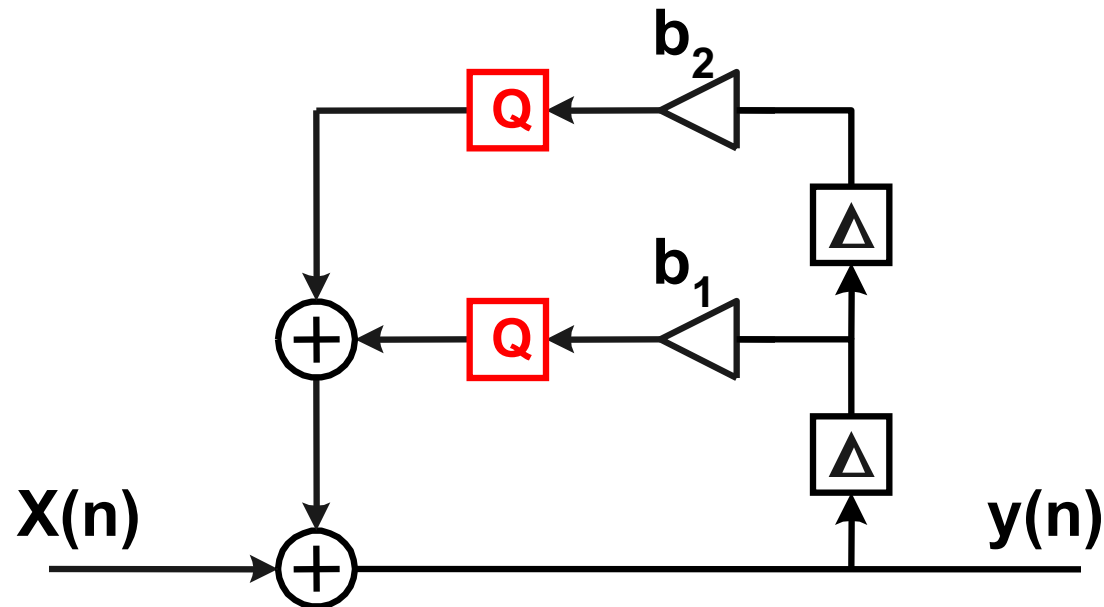
# Limit Cycles Oscillations

Limit cycle oscillations can be described as the undesirable periodic components at the output due to the fact that quantization is a nonlinear operation.



# Limit Cycles Oscillations

**Example: zero input oscillations in 2nd order IIR**

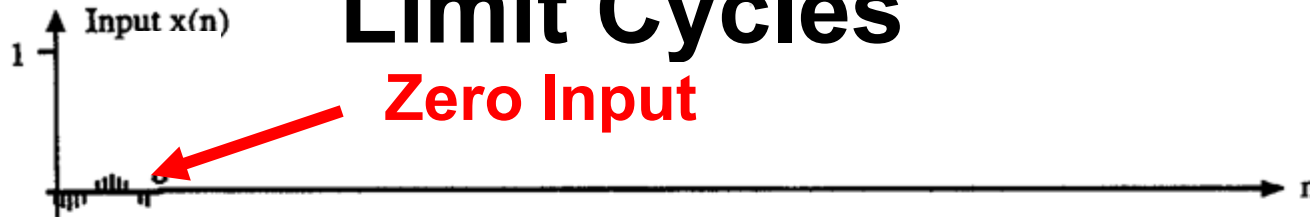


$$b_1 = \frac{489}{256} = 1.91015625; \quad b_2 = -\frac{15}{16} = -0.9375$$

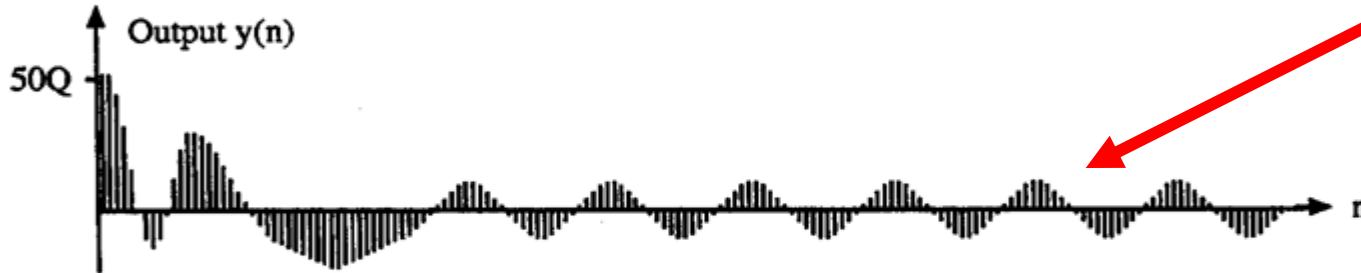
Example: zero input oscillations

# Limit Cycles

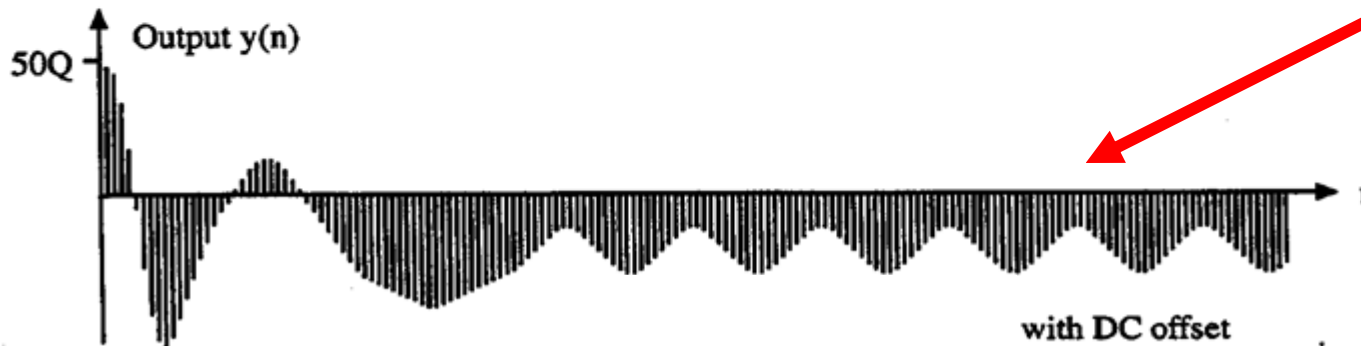
Zero Input



Rounding after multiplication



Truncation after multiplication



Source: Lars Wanhammar, "DSP Integrated circuits"

# Limit Cycles

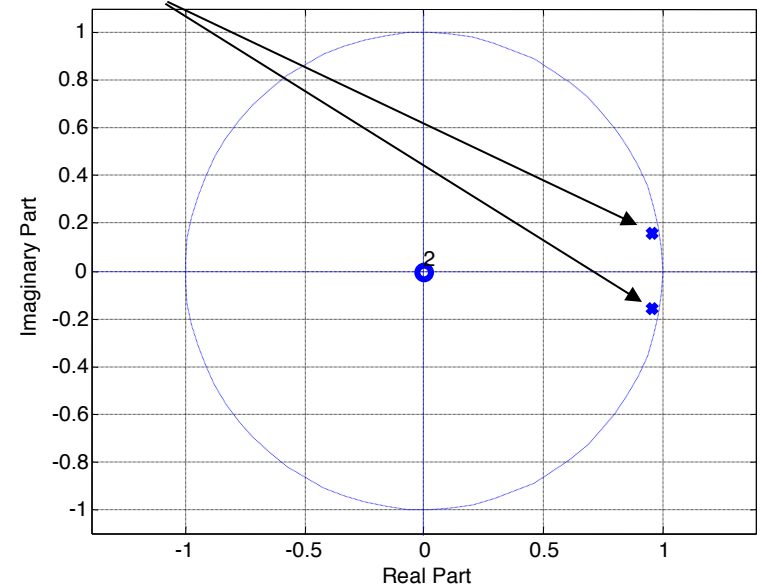
## Zero input oscillations

- Not accepted in some applications such as in audio applications.

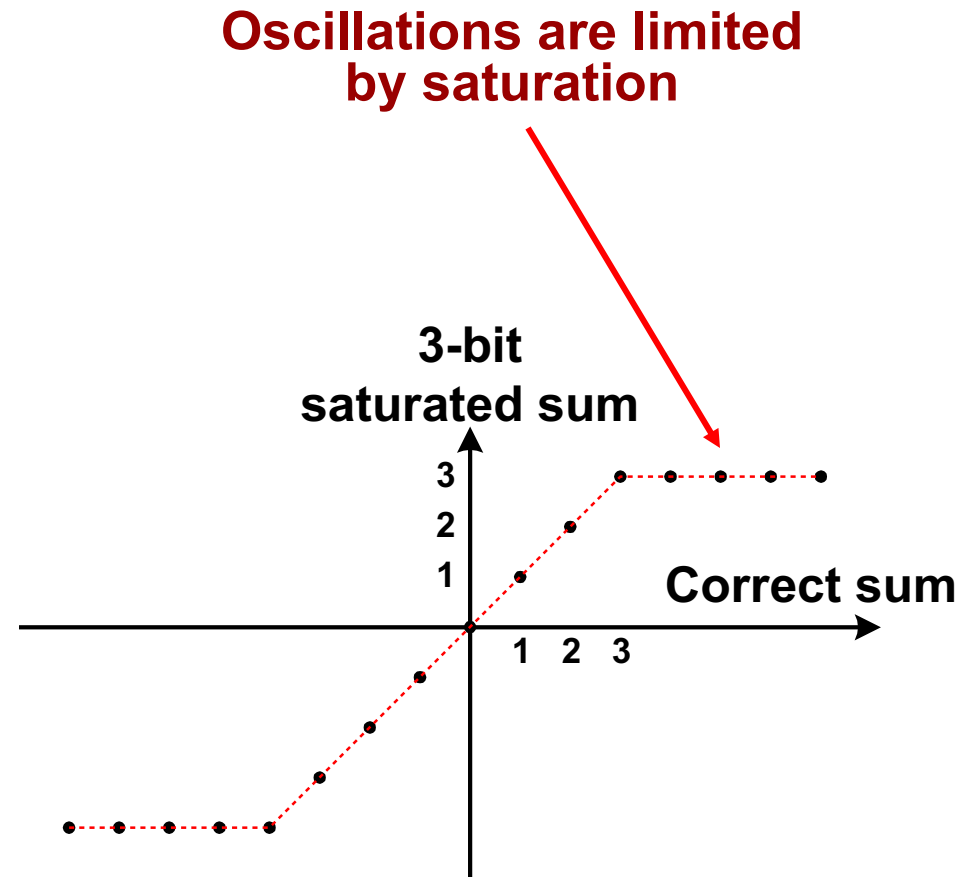
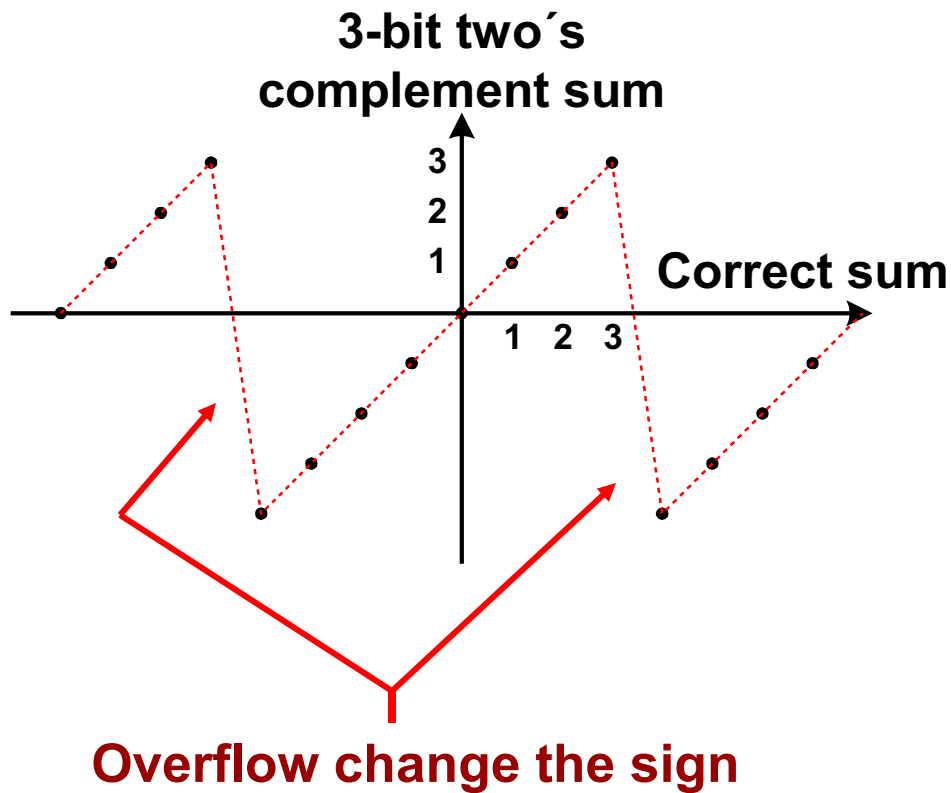
## Very difficult problem

- In general, no solutions for structures > 2<sup>nd</sup> order
- Can be limited by increased internal wordlength
- Can in some 2<sup>nd</sup> order structures be eliminated by pole positioning
- 2<sup>nd</sup> order Wave Digital Filters are free from parasitic oscillations

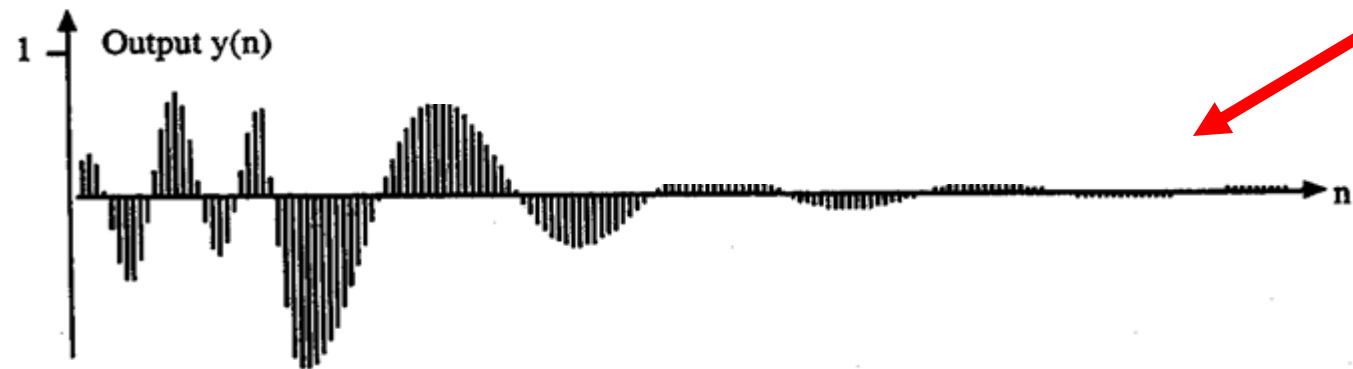
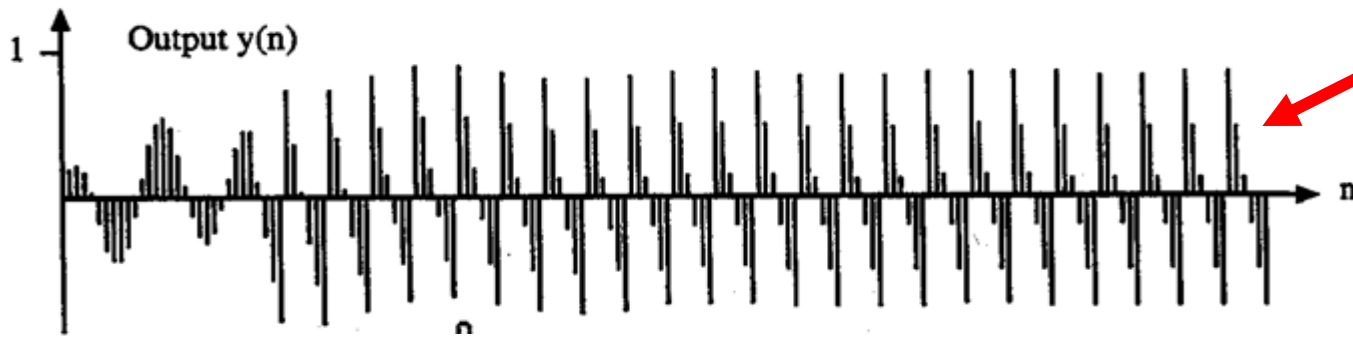
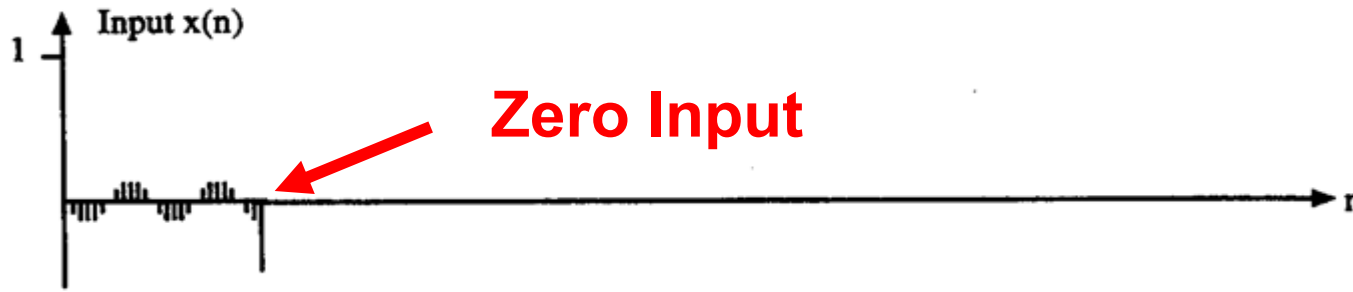
Poles close to the unity circle  
 Matlab: `zplane(1,[1 -1.91015625 0.9375])`



# We may also get Overflow Oscillations



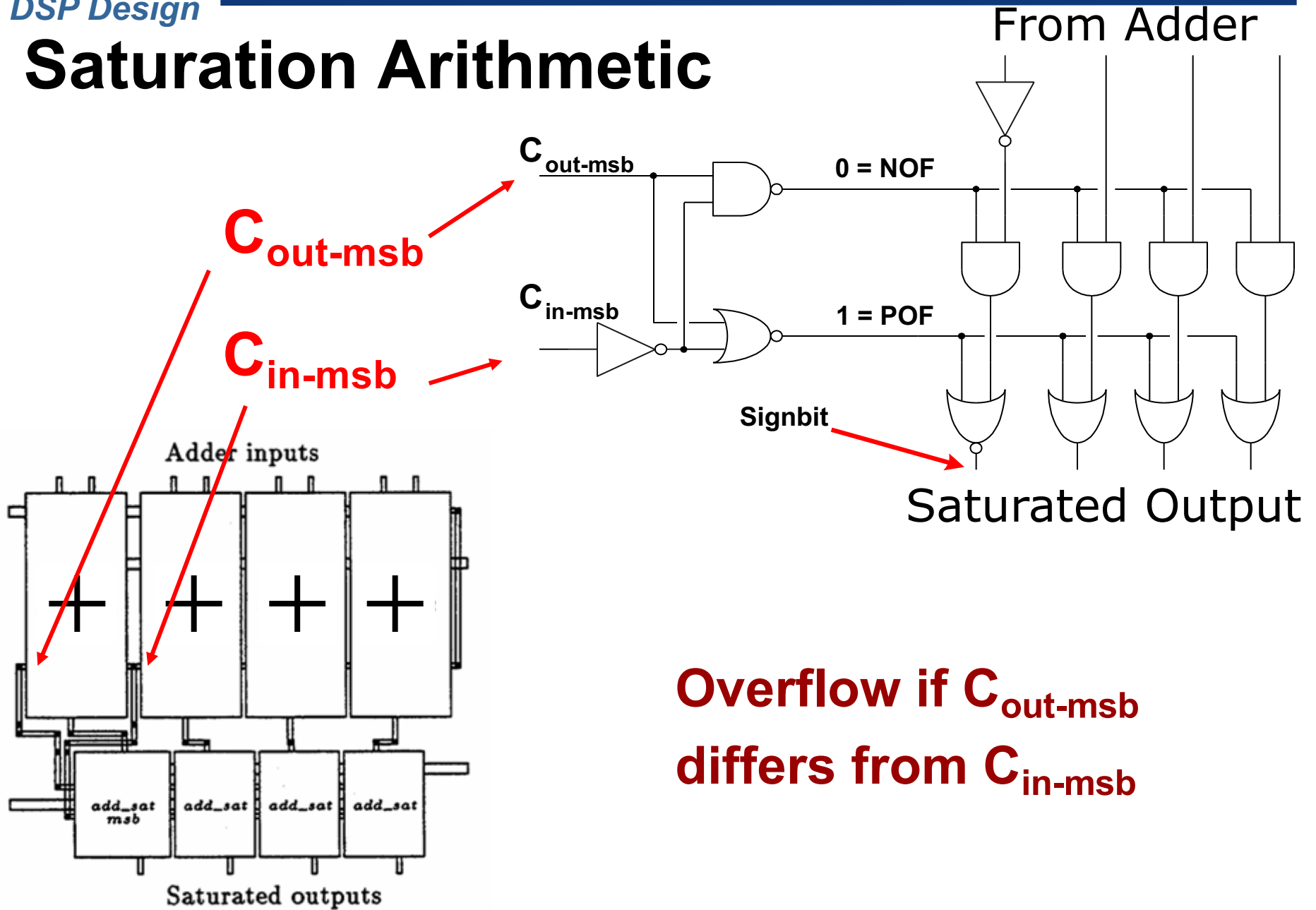
# Limit Cycles due to overflow



Source: Lars Wanhammar, "DSP Integrated circuits"



# Saturation Arithmetic



**Overflow if  $C_{out-msb}$  differs from  $C_{in-msb}$**

# Simple Noise Analysis

# Quantization Analysis

## Using “real” rounding, truncation, and overflow

- Gives exact result
- Tricky - need integer representation

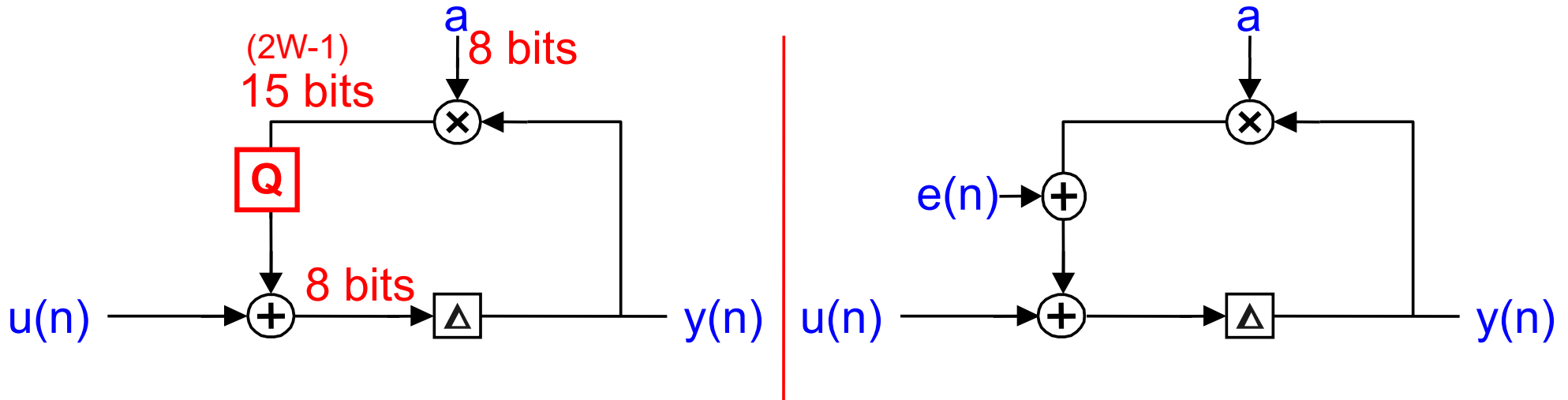
## Using noise models

- Floating point representation can still be used
- Suitable for Matlab, C/C++ ...

# Rounding

# Model

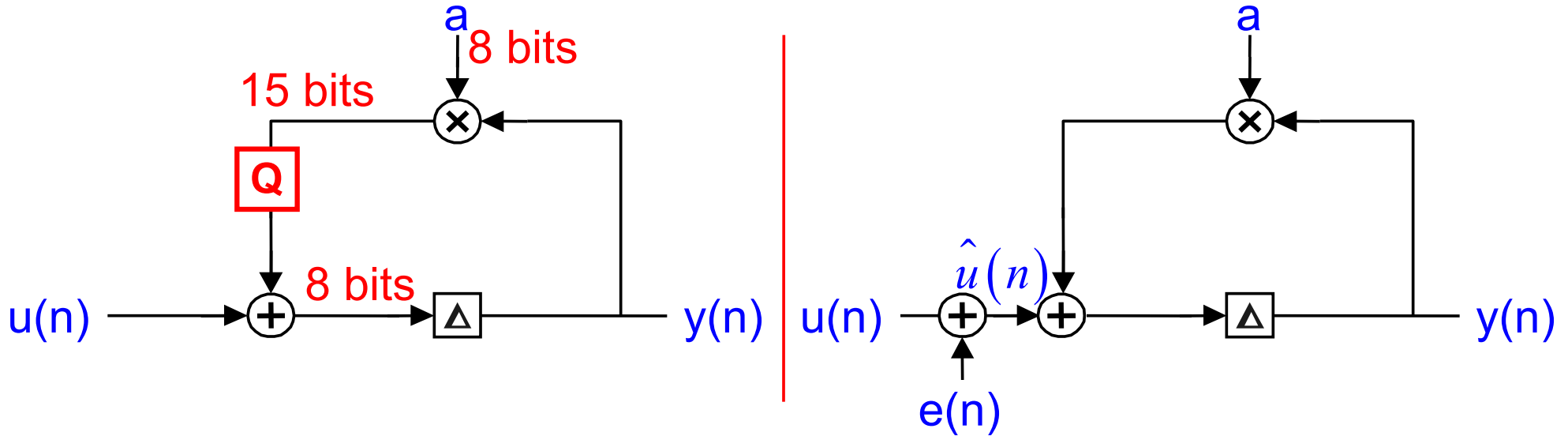
Wordlength (W) = 8 bits



Wordlength (W) = 8 bits

# Rounding

# Model



$$e(n) = \hat{u}(n) - u(n)$$

**Modeled with added noise as an input error**

# Analysis of Round-off Noise

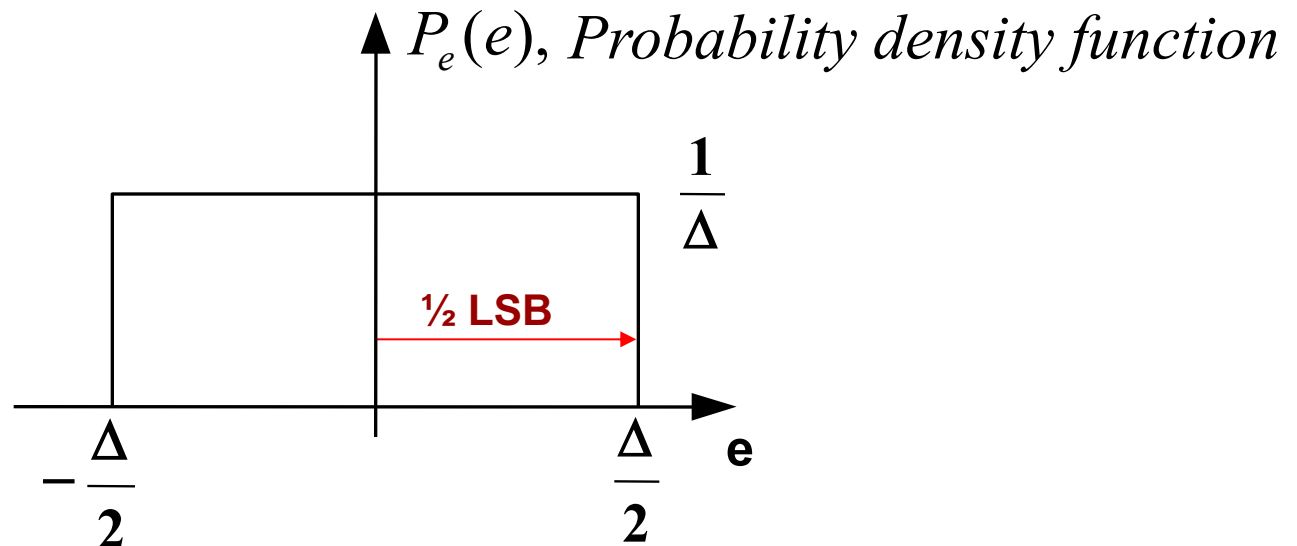
- **We need to analyze the round-off noise to determine its effect on the output signal!**
- **If the noise variance is not negligible in comparison with the output signal level the wordlength must be increased or some low-noise structure must be used!**

# Roundoff Noise

If the quantization error probability is uniformly distributed in the interval

$$-\frac{\Delta}{2} \leq e(n) \leq \frac{\Delta}{2} \text{ where } \Delta = 2^{-(W-1)}$$

$W$  is the number of bits after the rounding



# Statistical model of $e(n)$

The statistical model is based on the following assumptions:

1.  $e(n)$  is a sample sequence of a wide-sense stationary random process.
2.  $e(n)$  is uncorrelated with  $x(n)$
3. The error is a white noise process
4. The probability distribution of the error process is uniform over the range of the quantization error.

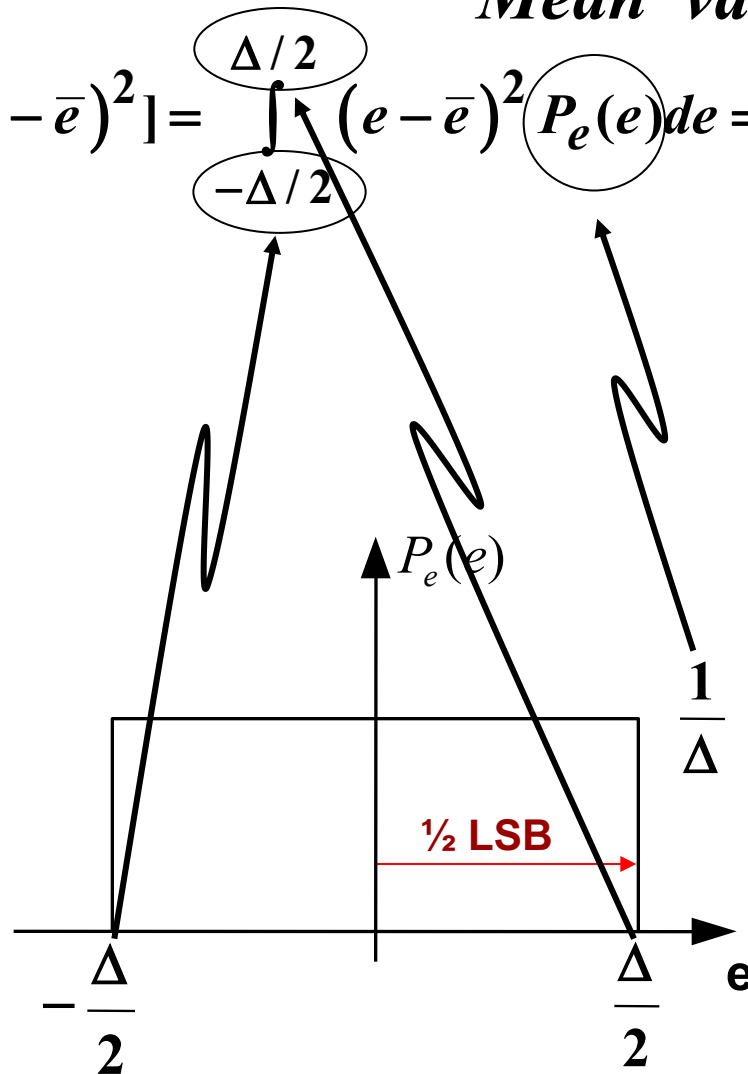
It is easy to find situations where this is not valid, e.g. if  $x(n)$  is a square wave. However, when  $x(n)$  is complicated, e.g. speech or music, the assumptions are realistic.



# Roundoff Noise

Mean value  $\bar{e} = E[e(n)] = 0$

$$\text{Variance} = E[(e - \bar{e})^2] = \int_{-\Delta/2}^{\Delta/2} (e - \bar{e})^2 P_e(e) de = [\bar{e} = 0]$$

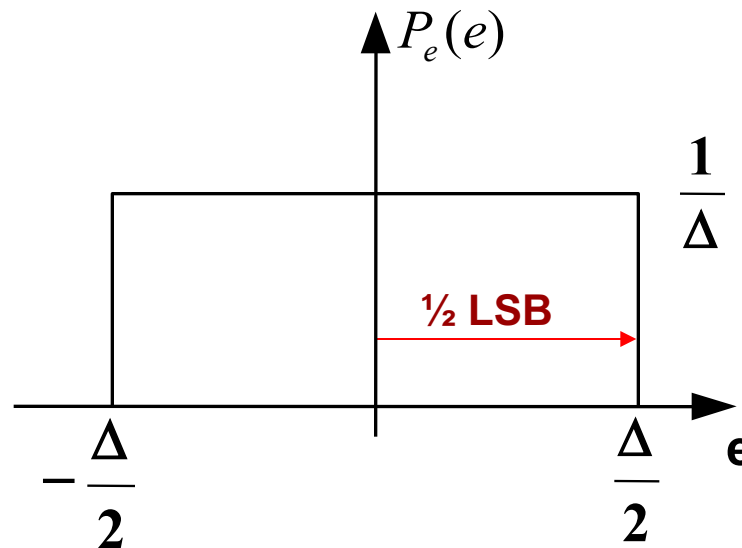


# Roundoff Noise

Mean value  $\bar{e} = E[e(n)] = 0$

$$\text{Variance} = E[(e - \bar{e})^2] = \int_{-\Delta/2}^{\Delta/2} (e - \bar{e})^2 P_e(e) de = [\bar{e} = 0]$$

$$\int_{-\Delta/2}^{\Delta/2} e^2 \frac{1}{\Delta} de = \left[ \frac{1}{\Delta} \frac{e^3}{3} \right]_{-\Delta/2}^{\Delta/2}$$

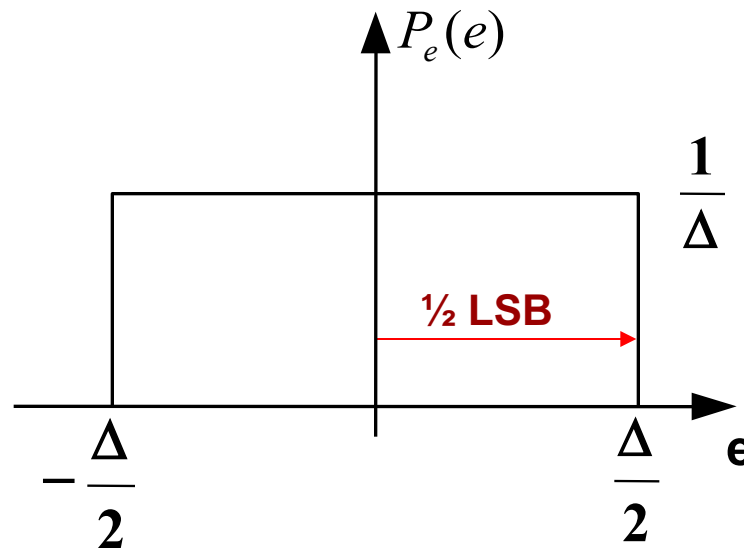


# Roundoff Noise

Mean value  $\bar{e} = E[e(n)] = 0$

$$\text{Variance} = E[(e - \bar{e})^2] = \int_{-\Delta/2}^{\Delta/2} (e - \bar{e})^2 P_e(e) de = [\bar{e} = 0]$$

$$\int_{-\Delta/2}^{\Delta/2} e^2 \frac{1}{\Delta} de = \left[ \frac{1}{\Delta} \frac{e^3}{3} \right]_{-\Delta/2}^{\Delta/2} = \frac{1}{\Delta} \left[ \frac{\Delta^3}{8} - \frac{-\Delta^3}{8} \right]$$

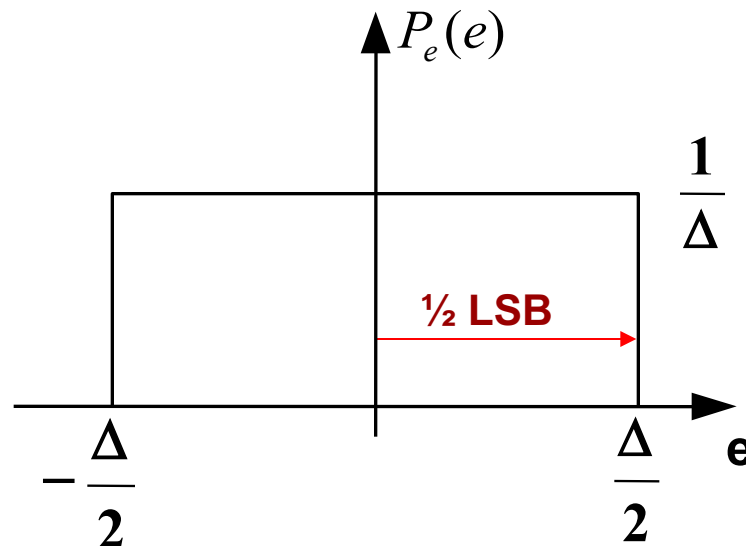


# Roundoff Noise

Mean value  $\bar{e} = E[e(n)] = 0$

$$\text{Variance} = E[(e - \bar{e})^2] = \int_{-\Delta/2}^{\Delta/2} (e - \bar{e})^2 P_e(e) de = [\bar{e} = 0]$$

$$\int_{-\Delta/2}^{\Delta/2} e^2 \frac{1}{\Delta} de = \left[ \frac{1}{\Delta} \frac{e^3}{3} \right]_{-\Delta/2}^{\Delta/2} = \frac{1}{\Delta} \left[ \frac{\Delta^3/8}{3} - \frac{-\Delta^3/8}{3} \right] = \frac{\Delta^2}{12} = \frac{2^{-2W}}{3}$$



## Example: Roundoff Noise

In the case of rounding ( $mean=0$ ) the variance and the average power are the same, i.e. if a value is rounded the quantization noise becomes:

$$\sigma_e^2 = \frac{2^{-2W}}{3}$$

If we scale down one bit:

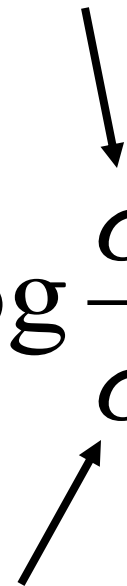
$$\frac{2^{-2(W-1)}}{3} = \frac{2^2 \times 2^{-2W}}{3} = 4\sigma_e^2$$

# Analysis of Round-off Noise

- **Not only the noise gain at the output needs to be determined!**
- **We also need to determine the SNR at the output to get the full picture!**

# Signal to Noise Ratio (SNR)


Signal power (variance)

$$SNR = 10 \log \frac{\sigma_x^2}{\sigma_e^2} = 10 \log \frac{3}{2^{-2W}} \sigma_x^2$$


Roundoff error power (variance)

# Signal to Noise Ratio (SNR)

One extra bit reduces quantization error by a factor 4


$$SNR = 10 \log \frac{4\sigma_e^2}{\sigma_e^2} = 6.02 \text{ dB}$$

Good to remember: 6 dB increase in SNR per bit



# Signal to Noise Ratio (SNR)

**Example: Full scale sinus wave rounded to 8 bits**

$$\mathbf{SNR = 10 \log \frac{3}{2^{-2 \times 8}} \left( \frac{A}{\sqrt{2}} \right)^2 = 50 \text{ dB}; -1 \leq A \leq 1}$$

**Equivalent: 100 000**

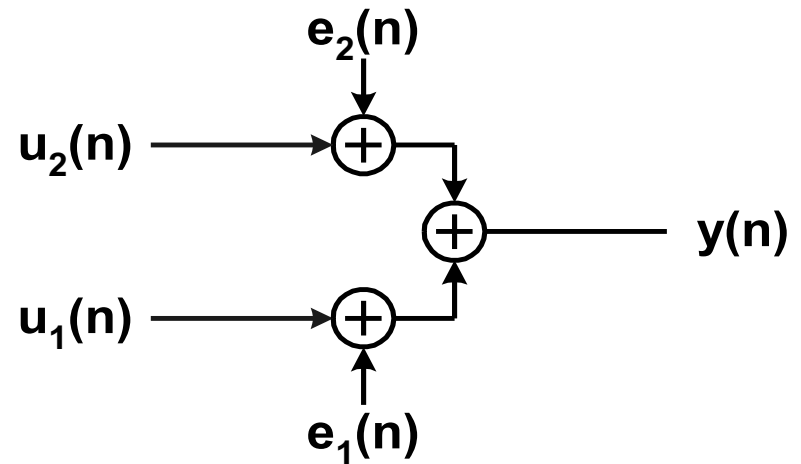
# Roundoff Noise: Addition

$$E[(e_1 + e_2)^2] = E[e_1^2 + 2e_1e_2 + e_2^2] =$$

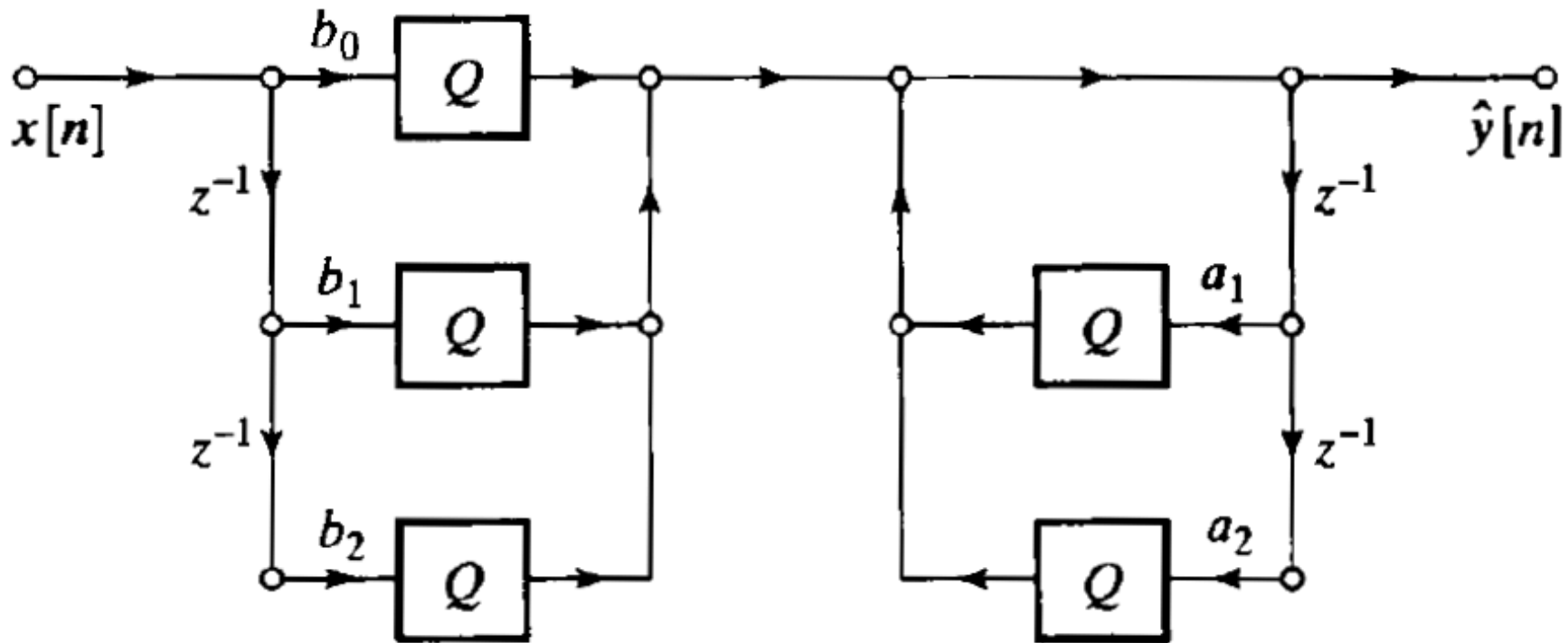
$$= E[e_1^2] + \underbrace{E[2e_1e_2]}_{\text{zero if } u_1 \text{ and } u_2 \text{ independent}} + E[e_2^2] =$$

*zero if  
 $u_1$  and  $u_2$   
independent*

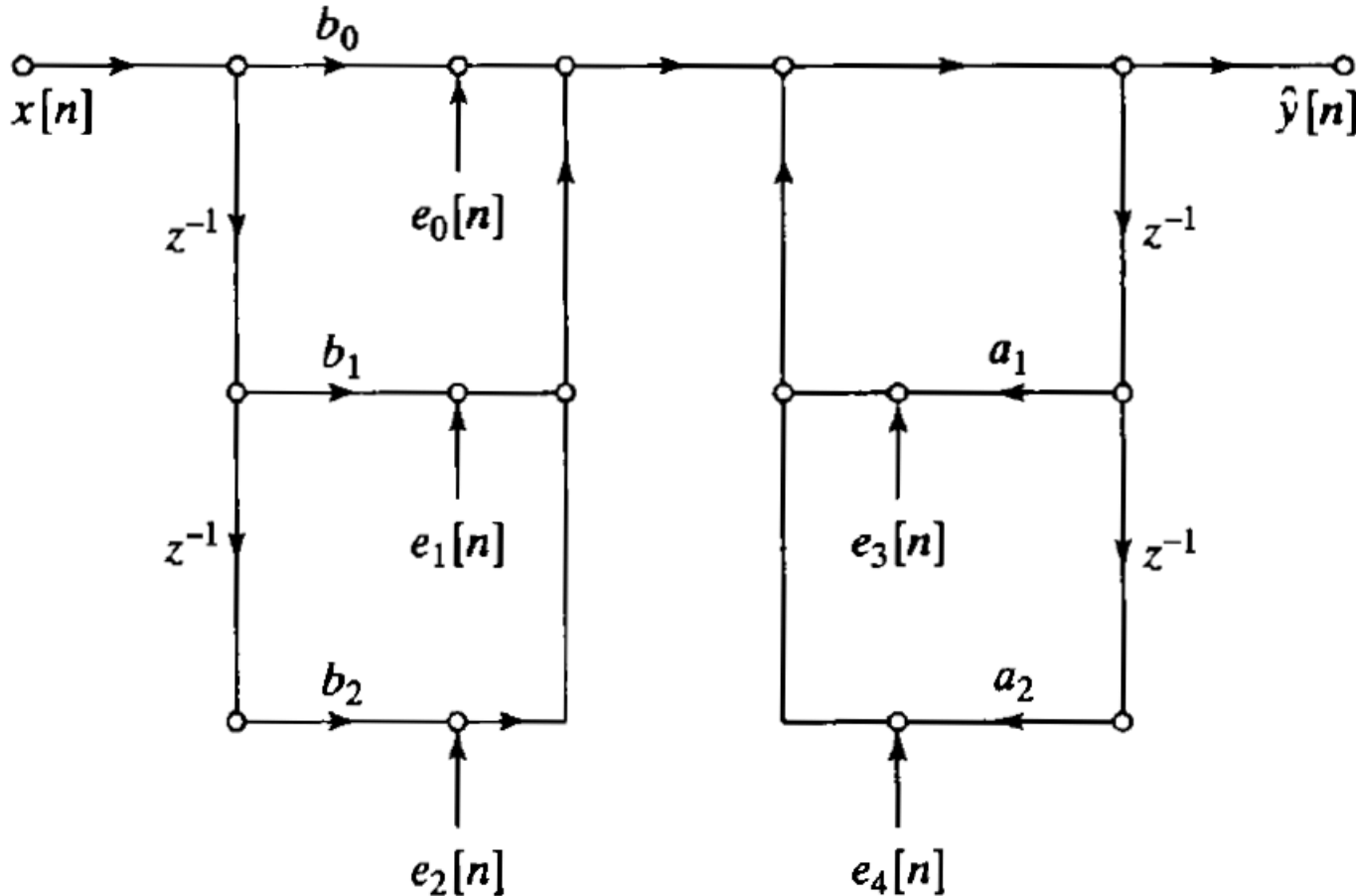
$$= E[e_1^2] + E[e_2^2]$$



# A larger IIR-filter with several quantization noise sources (I)



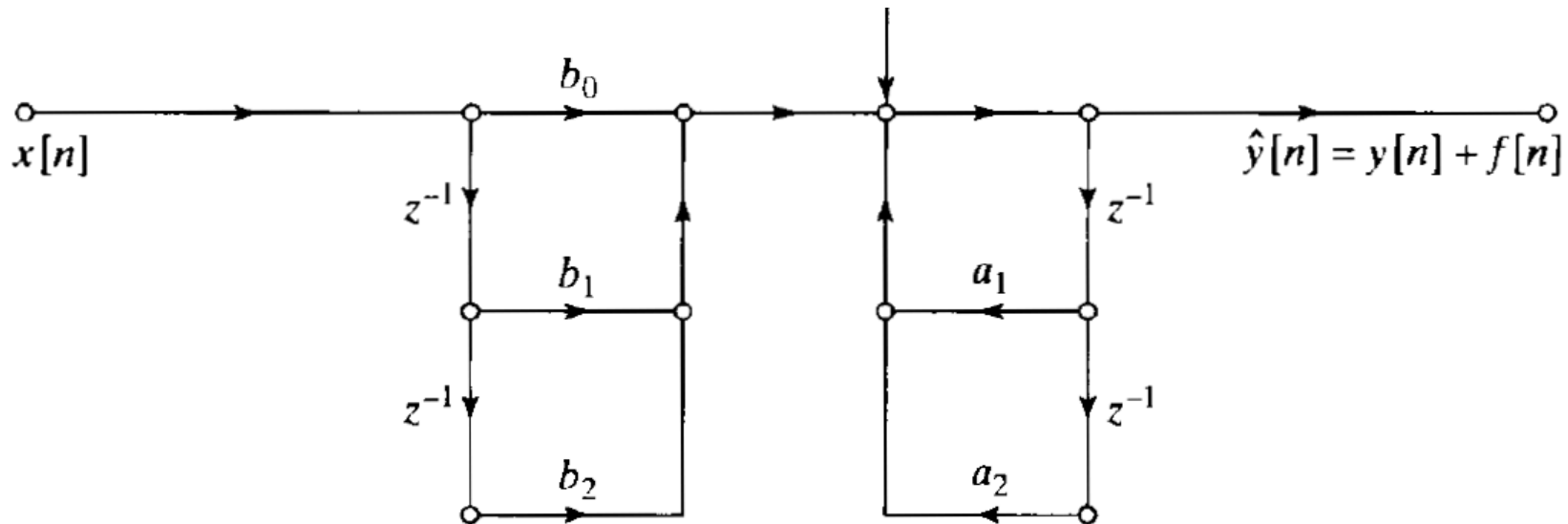
# A larger IIR-filter with several quantization noise sources (II)



# A larger IIR-filter with several quantization noise sources (III)

Assumption: noise sources are independent of (a) the input and (b) each other  $\Rightarrow$  can be added  $\Rightarrow$

$$\sigma_e^2 = \sigma_{e0}^2 + \sigma_{e1}^2 + \sigma_{e2}^2 + \sigma_{e3}^2 + \sigma_{e4}^2$$



# End of Lecture 2