Electronic Systems

Digital Representations

Ing. Luca Zulberti – luca.zulberti@phd.unipi.it
Prof. Massimiliano Donati – massimiliano.donati@unipi.it
Prof. Luca Fanucci – luca.fanucci@unipi.it

Agenda

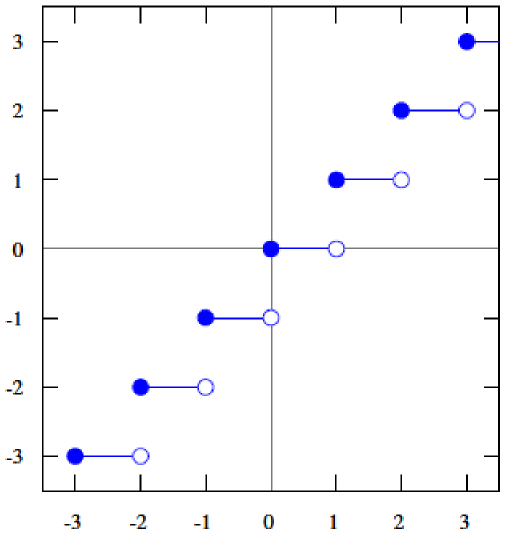
1) Integer Representations
   I. Unsigned
   II. Signed
2) Real Representations
   I. Floating-Point
   II. Fixed-Point
3) Quantization Strategies in Digital Design
4) Bit-true implementation: Alpha-Max Beta-Min example

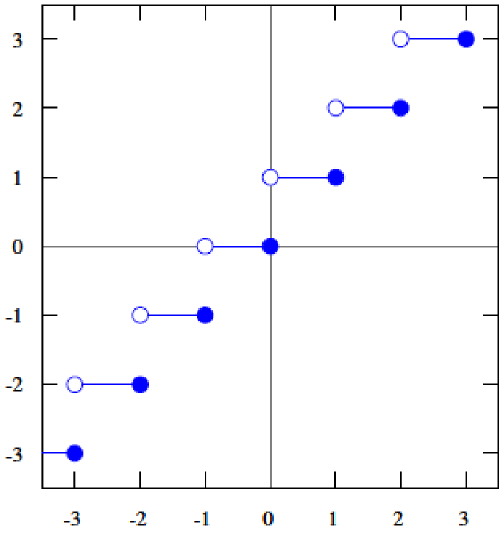
Agenda

1) Integer Representations
   I. Unsigned
   II. Signed
2) Real Representations
   I. Floating-Point
   II. Fixed-Point
3) Quantization Strategies in Digital Design
4) Bit-true implementation: Alpha-Max Beta-Min example

Rounding $\mathbb{R} \to \mathbb{Z}$

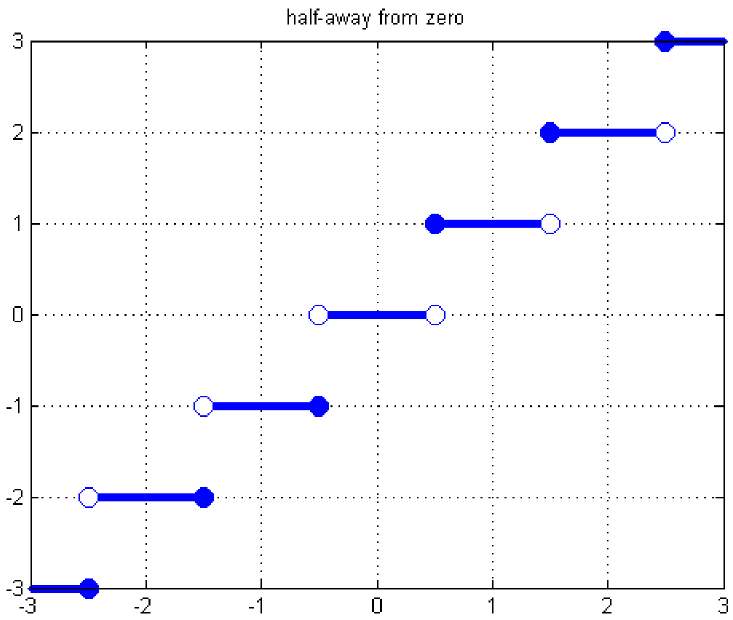
- Greatest preceding integer $\quad y = \lfloor x \rfloor = floor(x) = \max(y \in \mathbb{Z} : y \leq x)$
- Least succeeding integer $\quad y = \lceil x \rceil = ceil(x) \quad = \min(y \in \mathbb{Z} : y \geq x)$





Rounding $\mathbb{R} \to \mathbb{Z}$

- Round half-up $\quad y = round(x) = \lfloor x + 0{,}5 \rfloor$
- Round half-down $\quad y = round(x) = \lceil x - 0{,}5 \rceil$
- **Round half-away** $\quad y = round(x) = sgn(x)\lfloor |x| + 0{,}5 \rfloor$
- Round half-towards $\quad y = round(x) = sgn(x)\lceil |x| - 0{,}5 \rceil$



Rounding $\mathbb{R} \to \mathbb{Z}$

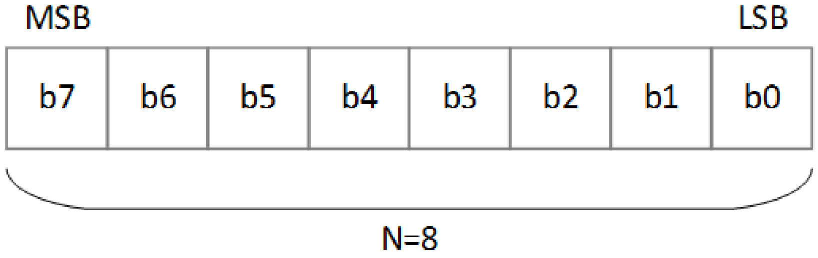
```
import math
```

```
#include <math.h>
```

`y = floor(x)`	`y = math.floor(x)`	`y = floor(x)`	`=ARROTONDA.DIFETTO(A1;1)`
`y = ceil(x)`	`y = math.ceil(x)`	`y = ceil(x)`	`=ARROTONDA.ECCESSO(A1;1)`
`y = round(x)`	`y = round(x)`	`y = round(x)`	`=ARROTONDA(A1;0)`
`y = fix(x)`	`y = int(x)`	`y = (int)x`	`=ARROTONDA.PER.DIF(A1;0)`

Unsigned

- Given N-bit
- 2^N representations
- $UN(N) = [0: 2^N - 1]$



$$x = b_{N-1}b_{N-2} \dots \dots b_1 b_0 = \sum_{k=0}^{N-1} b_k 2^k$$

$$N = \lceil \log_2(x+1) \rceil$$

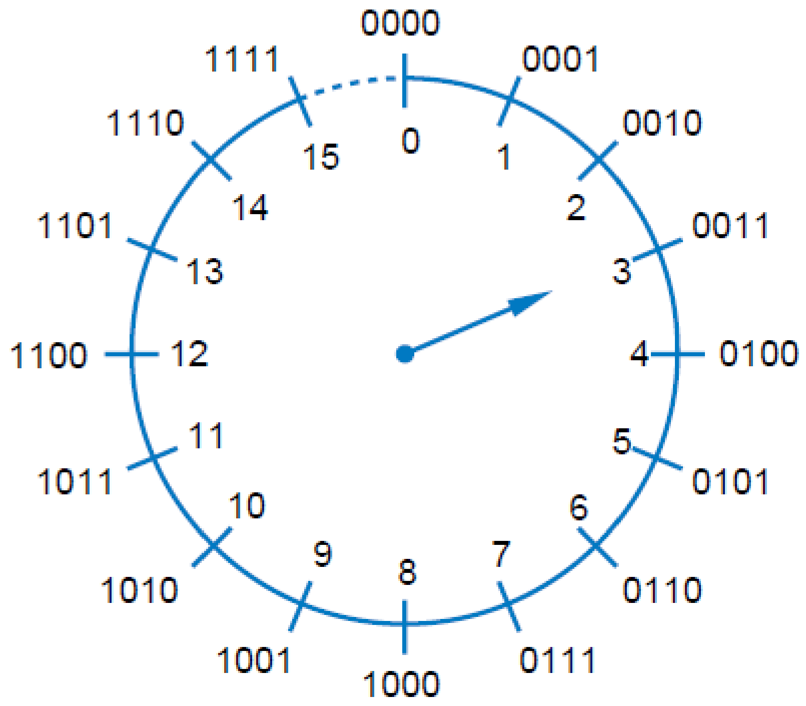
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	10
1011	11
1100	12
1101	13
1110	14
1111	15

Unsigned

- Addition needs one bit more (MSB)
  - Avoid overflow or carry-out (wrap-around)
- $UN(N) + UN(N) \rightarrow UN(N+1)$
- $UN(N) + UN(M) \rightarrow UN(\max(N, M) + 1)$

	1	0	1	1	0	0	1	0
		1	1	1	1	0	1	1
1	0	0	1	0	1	1	0	1

$178 + 123 = 301$



Unsigned

- Multiplication is representable on the sum of word length
- $UN(N) \times UN(N) \rightarrow UN(2N)$
- $UN(N) \times UN(M) \rightarrow UN(N+M)$

					1	0	1	1	
					1	1	0	1	
					1	0	1	1	
				0	0	0	0		Partial Products
			1	0	1	1			
		1	0	1	1				
11 x 13 = 143	1	0	0	0	1	1	1	1	

Signed

- Sign and Magnitude
  - The MSB is the sign (0: positive, 1: negative)
  - Others are the magnitude

$$x = (-1)^{b_{N-1}} \times \sum_{k=0}^{N-2} b_k 2^k$$

- $N = 8$

b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
$\pm$	b_6	b_5	b_4	b_3	b_2	b_1	b_0

Signed

- Biased or Excess-K
  - The «all zero» representation is centered on $-K$
  - Representation as translation

$$x = -K + \sum_{k=0}^{N-1} b_k 2^k$$

- $N = 8$

b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	
b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	$-K$

Signed

- One's complement
  - Negative integers are bitwise negation of positives
  - Bitwise negation is the same of $2^N - 1 - |x|$

$$x = -b_{N-1}(2^N - 1) + \sum_{k=0}^{N-2} b_k 2^k$$

- $N = 8$

b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	
b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	$-b_7(2^7 - 1)$

Signed

- Two's complement
  - Negative integers are bitwise negation of positives+1
    - right to left, until first '1', then flip the rest
  - Is the same of $2^N - |x|$

$$x = -b_{N-1}2^{N-1} + \sum_{k=0}^{N-2} b_k 2^k$$

- $N = 8$

b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	
b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	$-b_7 2^7$

Signed

- Two's complement (signed de facto)
- $SG(N) = [-2^{N-1}: 2^{N-1} - 1]$

$$N = \lceil \log_2(x + 1) \rceil + 1 \qquad if\ x \geq 0$$
$$N = \lceil \log_2(|x|) \rceil + 1 \qquad if\ x < 0$$

- C2 Balanced representation
- $SG_B(N) = [-2^{N-1} + 1: 2^{N-1} - 1]$

1000	-8
1001	-7
1010	-6
1011	-5
1100	-4
1101	-3
1110	-2
1111	-1
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7

Word length extension

- Unsigned Extension
  - Insert ‘0’ as new MSB

0	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0

- Signed (C2) Extension
  - Copy the MSB as new MSB
  - From $x_{C2_{N+1}} = |x|_{2^{N+1}}$

b_7	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0

Addition

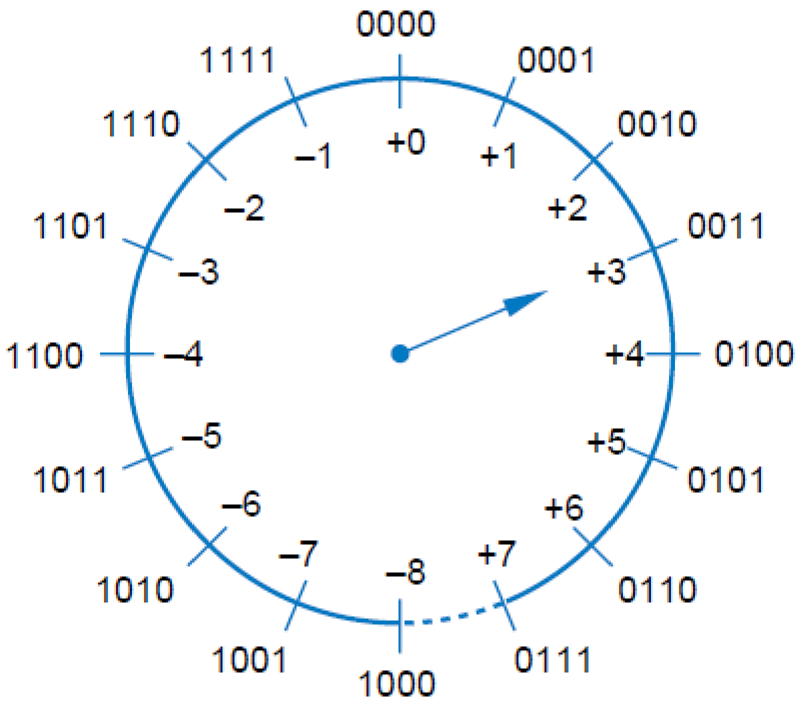
- Addition "needs" one bit more (MSB copied)
  - Avoid C2 overflow
  - Digital native $(x \pm y)_{C2} = |x_{C2} \pm y_{C2}|_{2^N}$
- $SG(N) + SG(N) \rightarrow SG(N+1)$
- $SG(N) + SG(M) \rightarrow SG(\max(N,M)+1)$

Extended

	1	1	0	1	1	0	0	1	0
	0	0	1	1	1	1	0	1	1
~~1~~	0	0	0	1	0	1	1	0	1

$-78 + 123 = 45$

Don't mind the carry out in C2



Multiplication

- Multiplication is representable on the sum of word length
- $SG(N) \times SG(N) \rightarrow SG(2N)$ $\quad$ $SG_B(N) \times SG_B(N) \rightarrow SG_B(2N-1)$
- $SG(N) \times SG(M) \rightarrow SG(N+M)$ $\quad$ $SG_B(N) \times SG_B(M) \rightarrow SG_B(N+M-1)$

!!! Digital circuits **DO NOT** work intrinsically with C2 representation for multiplication !!!

- Absolute values multiplication (unsigned)
- Dedicated Architectures (Booth's algorithm, Baugh-Wooley multiplier…)

Shift Operation

- Shift Right Logical (srl)
  - Discard k bit from right and insert k zeros from left ($\left\lfloor \frac{x}{2^k} \right\rfloor$ on unsigned)

$$b_7\ b_6\ b_5\ b_4\ b_3\ b_2\ b_1\ b_0 \gg 3 \quad 0\ 0\ 0\ b_7\ b_6\ b_5\ b_4\ b_3$$

- Shift Right Arithmetical (sra)
  - Discard k bit from right and insert k MSB from left ($\left\lfloor \frac{x}{2^k} \right\rfloor$ on signed)

$$b_7\ b_6\ b_5\ b_4\ b_3\ b_2\ b_1\ b_0 \gg 3 \quad b_7\ b_7\ b_7\ b_7\ b_6\ b_5\ b_4\ b_3$$

- Shift Left Logical (sll)
  - Discard k bit from left and insert k zeros from right ($x \times 2^k$ on unsigned and signed)
  - Overflow Possibilities!!

$$b_7\ b_6\ b_5\ b_4\ b_3\ b_2\ b_1\ b_0 \ll 3 \quad b_4\ b_3\ b_2\ b_1\ b_0\ 0\ 0\ 0$$

Agenda

1) Integer Representations
   I. Unsigned
   II. Signed
2) Real Representations
   I. Floating-Point
   II. Fixed-Point
3) Quantization Strategies in Digital Design
4) Bit-true implementation: Alpha-Max Beta-Min example

Real Representations

Two models

- **Floating-Point**
- **Fixed-Point**

Properties

- **Precision**: Number of bit used.
- **Resolution**: Smallest positive number representable, zero excluded.
- **Range**: The set given by the minimum negative number and maximum positive number.
- **Accuracy**: Maximum error from the real number and its representation (usually = Resolution/2).
- **Dynamic Range**: Ratio of maximum positive over minimum positive, zero excluded (dB).

Floating-Point

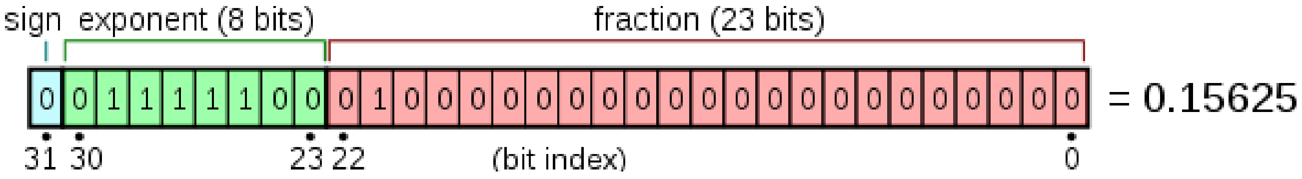
$$x = M \times \beta^E$$

- M = Significand, signed fraction
- β = Base
- E = Exponent, signed integer

- "Normalized Form" → M representation with the MSDigit different from zero.

Floating-Point

- Standard IEEE 754
  - Single Precision (32 bit)
  - Double Precision (64 bit)



- M = Significand, 24 bit of magnitude +1 bit for sign
  - F = Fractional, 23 bit stored + 1 implicit, for normalized form 1.fff…
  - s = Sign bit
- β = Base 2
- E = Exponent, 8-bit integer with excess-127, biased (0 and 255 used for special cases)

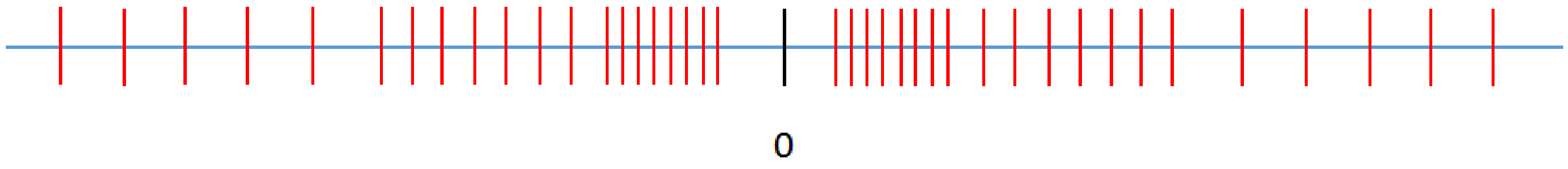
$$x = (-1)^s \times \left(1 + \sum_{k=1}^{23} b_{23-k} 2^{-k}\right) \times 2^{E-127}$$

Floating-Point

- Standard IEEE 754

	S	M	E	bias	~Min	~Max	DR	Max ϵ_r
Single	1 bit	1+23 bit	8 bit [-126:127]	127	10^{-38}	10^{+38}	1529 dB	$< 6 \times 10^{-8}$
Double	1 bit	1+51 bit	11 bit [-1022:1023]	1023	10^{-308}	10^{+308}	12312 dB	$< 3 \times 10^{-16}$

- Non-Uniform numbers distribution



Floating-Point

- Standard IEEE 754
  - Special cases

± 0

denormalized $|x| < 2^{-127}$

	F = 0	F ≠ 0
E = 0	$(-1)^S \cdot 0$	$x = (-1)^S (0.F) 2^{-126}$
E = 255	$(-1)^S \cdot \infty$	NaN

$\pm\infty$

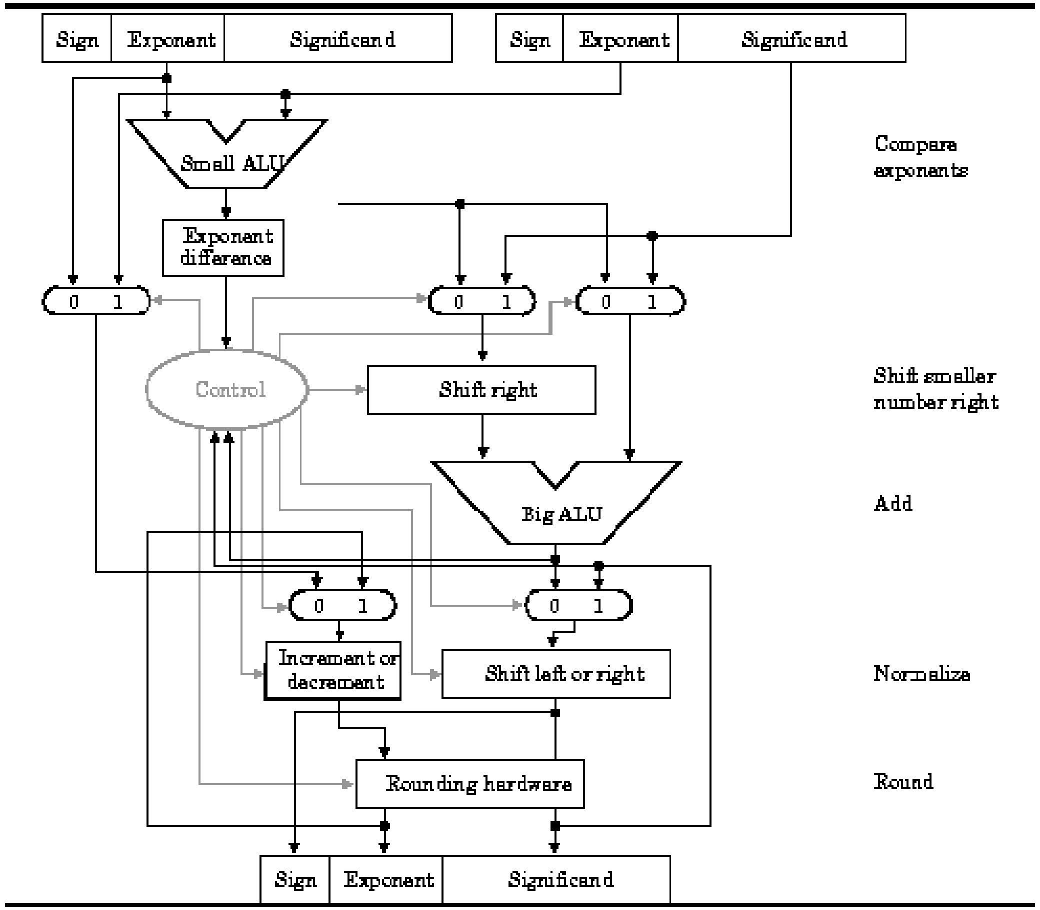
Not a Number (e.g. $\frac{0}{0}$)

Special cases must be managed with the same performance as normal cases.

Floating-Point

- Standard IEEE 754
- **Addition and Subtraction**
  - Align Fractional parts based on Exponents
  - Compute the result
  - Normalization and Round-off
    - Overflow possibility (±∞)
    - Underflow possibility (denormalized)
- **Multiplication**
  - Add the exponents
  - Multiply Fractional
  - Normalization of result
  - XOR for sign bit

MIPS FPA for add, sub



Fixed-Point

- Represent less order of magnitude than the floating-point ranges.
- But follow the integer arithmetic!!

$$\boldsymbol{x = N \times LSB_{value}}$$

- If the $\mathrm{LSB_{value}}$ is a power of 2, we can effectively see the decimal point.

$$x = \underbrace{b_{M-1} b_{M-2} \dots \dots b_1 b_0}_{integer\ part}\, \underbrace{b_{-1} \dots \dots b_{-(F-1)} b_{-F}}_{fractional\ part} = \sum_{k=-F}^{M-1} b_k 2^k$$

1	0	0	1	0	0	0	1

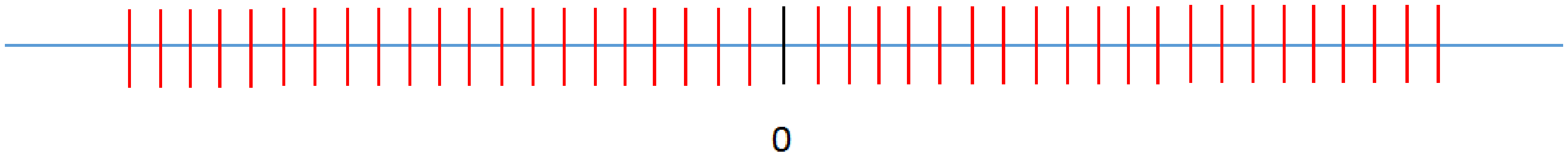
$$= 9.0625 = 145 \times 2^{-4}$$

- Fixed Point works exactly as integer representation but scaled by the weight of the LSB.

Fixed-Point

	Precision	Resolution	Accuracy	Range	DR
UN	8 bit	LSB	LSB/2	[0:255]	≈ 48 dB
SG	8 bit	LSB	LSB/2	[−128:127]	≈ 42 dB
UN	16 bit	LSB	LSB/2	[0:65535]	≈ 96 dB
SG	16 bit	LSB	LSB/2	[−32768:32767]	≈ 90 dB
UN	32 bit	LSB	LSB/2	[0:4294967295]	≈ 193 dB
SG	32 bit	LSB	LSB/2	[−2147483648:2147483647]	≈ 187 dB

- Uniform numbers distribution



Agenda

1) Integer Representations
   I. Unsigned
   II. Signed

2) Real Representations
   I. Floating-Point
   II. Fixed-Point

3) **Quantization Strategies in Digital Design**

4) Bit-true implementation: Alpha-Max Beta-Min example

Quantization

$$x \rightarrow Q[x]$$

- $Q[x]$ is the quantization function.
- Should be one of the rounding (or floor) techniques showed.

$$Q[x] = \underbrace{floor \mid round \left(\frac{x}{LSB}\right)}_{\text{Integer}} \times LSB$$

- absolute error is $\epsilon_A = |Q[x] - x|$

Quantization

- How to select the LSB or the Precision for a signal, or a variable(s), representation?
- Two cases: **starts from the LSB** or reach the LSB.
- If the LSB is a power of 2 and a given ϵ_A is required (accuracy requirements)

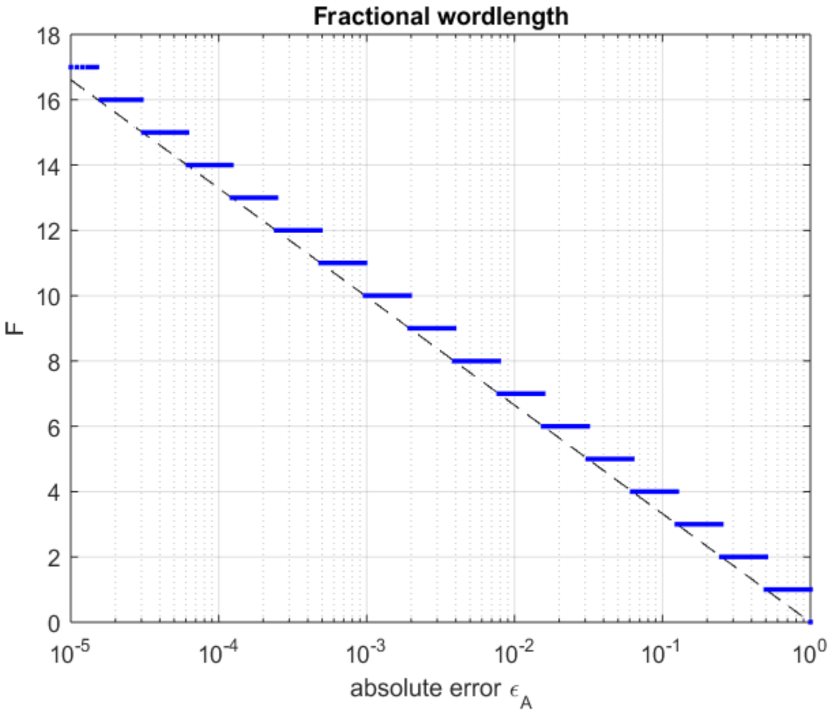
$$LSB = 2^{-F} \leq \epsilon_A \qquad F = \left\lceil \log_2\left(\frac{1}{\epsilon_A}\right)\right\rceil$$

Example

Represent $e = 2.718281828 \ldots$ with max $\epsilon_A = 10^{-3}$

$$F = \left\lceil \log_2\left(\frac{1}{\epsilon_A}\right)\right\rceil = 10 \qquad \rightarrow LSB = 2^{-10}$$

$$Q[e] = \left[\frac{e}{2^{-10}}\right] \times 2^{-10} = 2784 \times 2^{-10} = 2.71875$$



Quantization

- How to select the LSB or the Precision for a signal, or a variable(s), representation?
- Two cases: starts from the LSB or **reach the LSB**.
- If a range of reals has to be represented on N bits, we may normalize, remap, this range.

Hp: $\max x > 0$;
$\min x < 0$

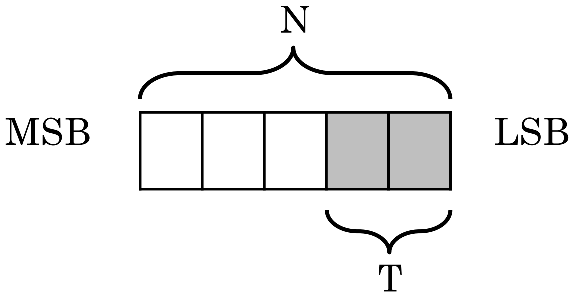
$$LSB = \frac{\max x}{2^{N-1} - 1} \quad or \quad \frac{|\min x|}{2^{N-1}}$$

Example

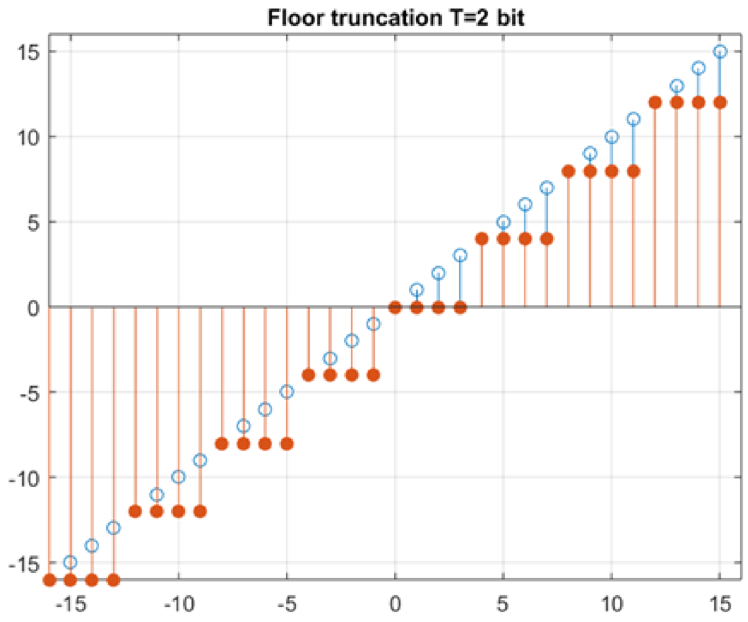
Represent $[-\pi, \pi)$ on $N = 8$ bit $\quad \rightarrow \; LSB = \frac{\pi}{2^{N-1}} = 0{,}024543692606 \ldots$

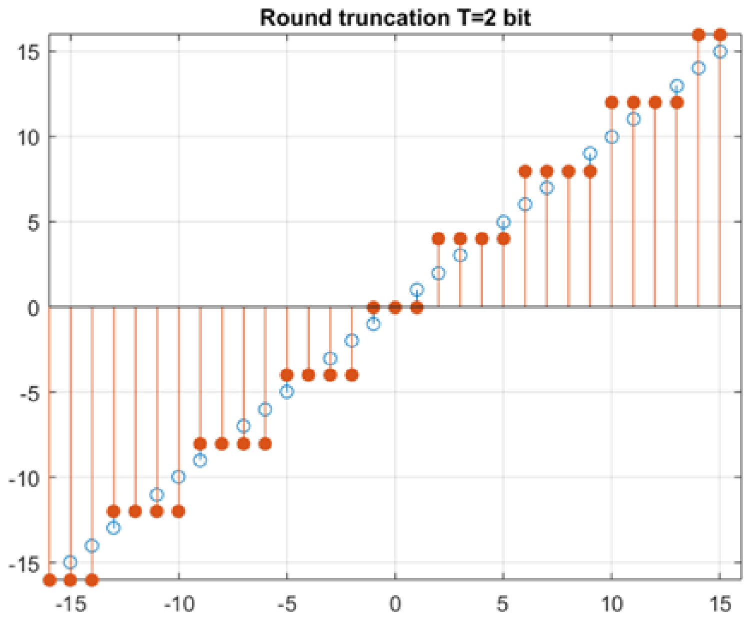
$$[-\pi, \pi) \rightarrow [-128{:}\, 127] \times LSB$$

Quantization – Truncation



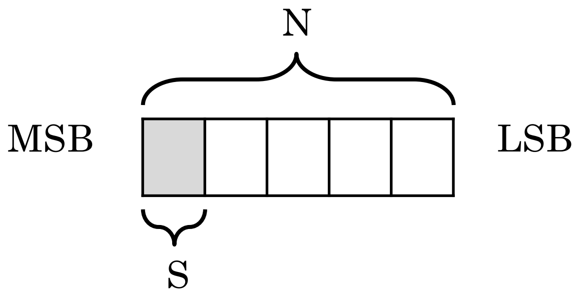
- **Truncation** discard *T* bit from right (LSBs).
- LSB changes value $LSB_{x_T} = LSB_x \cdot 2^T$
  - $x_T = floor\left(\frac{x}{2^T}\right)$ Simple, cut away *T* LSBs wires. Mean value altered.
  - $x_T = round\left(\frac{x}{2^T}\right)$ More Complex but does not alter the mean value.

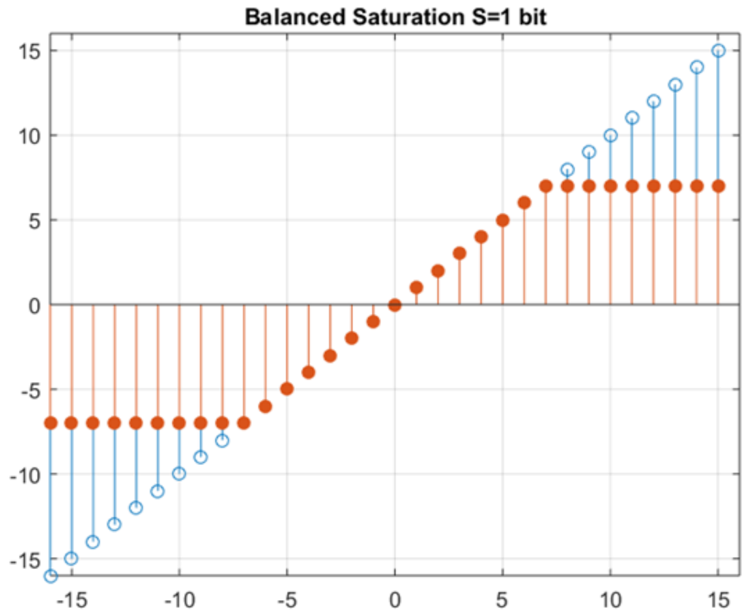




Quantization – Saturation

- **Saturation** discard S bit from left (MSBs) but implements a clipping strategies on data.
- LSB does not change value $LSB_{x_S} = LSB_x$
  - $$x_S = \begin{cases} x & x \in [-2^{N-S-1}+1 : 2^{N-S-1}-1] \\ -2^{N-S-1}+1 & x < -2^{N-S-1}+1 \\ 2^{N-S-1}-1 & x > 2^{N-S-1}-1 \end{cases}$$





Agenda

1) Integer Representations
   I. Unsigned
   II. Signed
2) Real Representations
   I. Floating-Point
   II. Fixed-Point
3) Quantization Strategies in Digital Design
4) Bit-true implementation: Alpha-Max Beta-Min example

Bit-true implementation

- Starting from a Problem “P” described by math formulation.
- The Design goes through many levels of abstraction to implement a digital circuit.

Math
- Problem formulation

Floating-Point
- Solving Algorithm ϕ
  - MATLAB
  - C/C++
  - Python
  - …

Algorithm

Fixed-Point
- Approx Algorithm ϕ^*
  - MATLAB
  - C/C++
  - Python
  - …

Implementation
- Bit-true model

RTL
- HW definition for ϕ^*
  - VHDL
  - Verilog
  - …

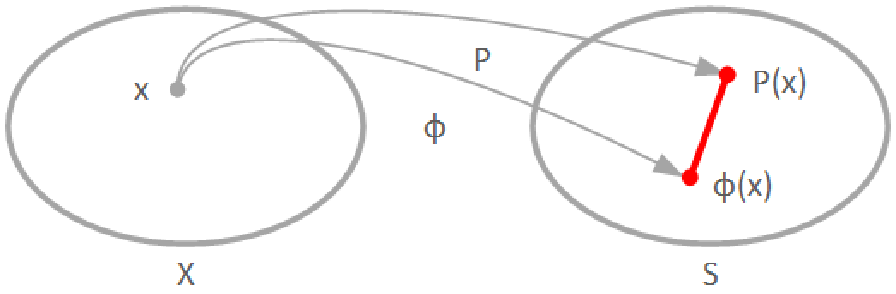
Architecture
- Bit-true model
- Cycle-true model
- Blocks Partitioning

Alpha-max Beta-min

- Compute the **magnitude of a complex number**
- Problem $P(a,b) = \sqrt{a^2 + b^2}$
- Square root operator is not easy in hardware, we need an approximated algorithm.

$$\boldsymbol{\phi}(\boldsymbol{a},\boldsymbol{b}) = \boldsymbol{\alpha}\,\mathbf{max}(|\boldsymbol{a}|,|\boldsymbol{b}|) + \boldsymbol{\beta}\,\mathbf{min}(|\boldsymbol{a}|,|\boldsymbol{b}|)$$

- Where α and β are real constant.



Alpha-max Beta-min

Math | Float | Fixed | RTL

- How to **measure the approximation of $\boldsymbol{\phi}(\boldsymbol{a}, \boldsymbol{b})$** vs the real problem $P(a, b)$?
- First, we have to select one metric to optimize (cost function). Common examples:
  - Maximum Error

$$MAX_E = \max_X(|\phi(\boldsymbol{x}) - P(\boldsymbol{x})|)$$

  - Mean Squared Error (MSE)

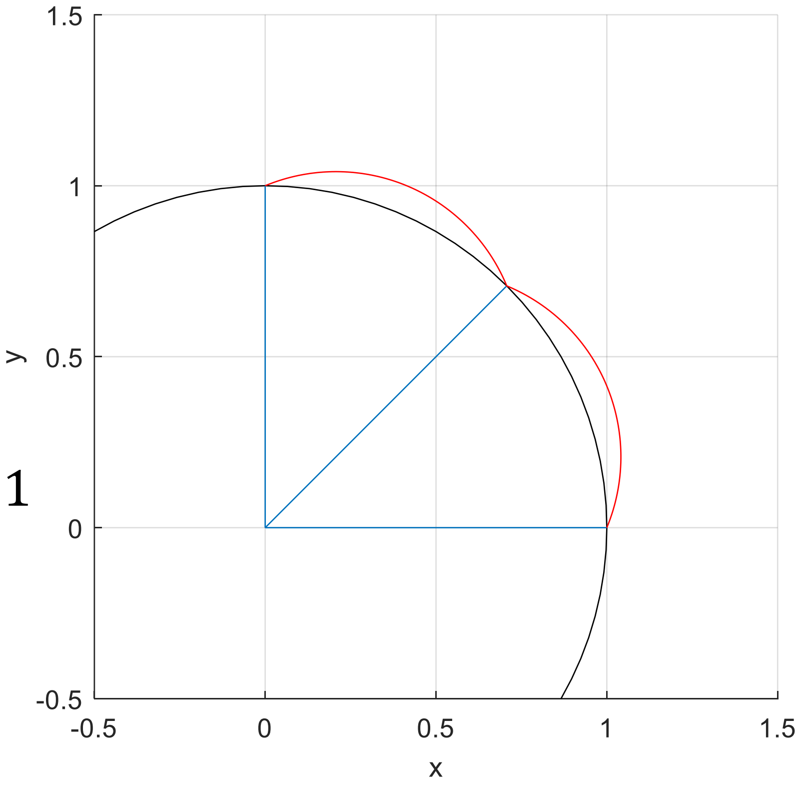
$$MSE = \frac{1}{card(X)^*} \sum_X \left(\phi(\boldsymbol{x}) - P(\boldsymbol{x})\right)^2$$

*cardinality/dimension of vector x

GOAL: Are there some α and β exact values that "minimize" a cost function?

Alpha-max Beta-min

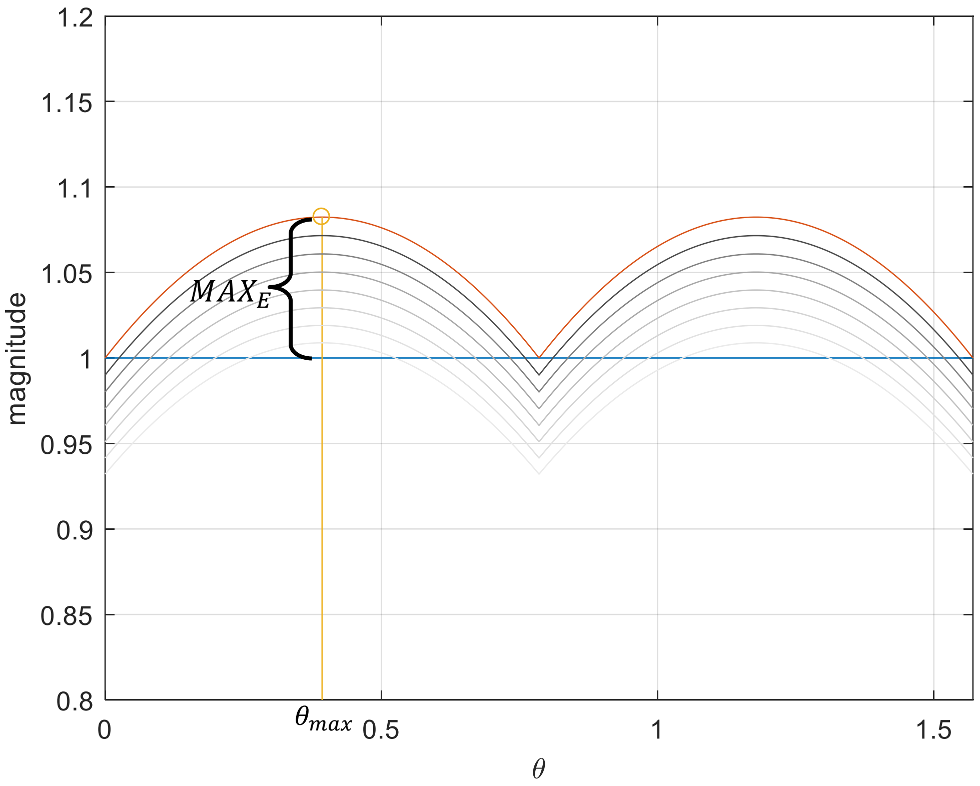
- $\phi(x, y) = \alpha \max(|x|, |y|) + \beta \min(|x|, |y|)$
- Let's take points over the unitary radius for $0 \le \theta \le \frac{\pi}{4}$
  - $\phi(\theta) = \alpha \cos(\theta) + \beta \sin(\theta) \quad$ for $\quad 0 \le \theta \le \frac{\pi}{4}$
- For $\theta = 0$ we hit $\phi = \alpha$
- For $\theta = \frac{\pi}{4}$ we hit, symmetrically, $\phi = \frac{\sqrt{2}}{2}(\alpha + \beta)$
- First empirical closed form $\alpha = 1$ then $\beta = \sqrt{2} - 1$



Alpha-max Beta-min

Math · Float · Fixed · RTL

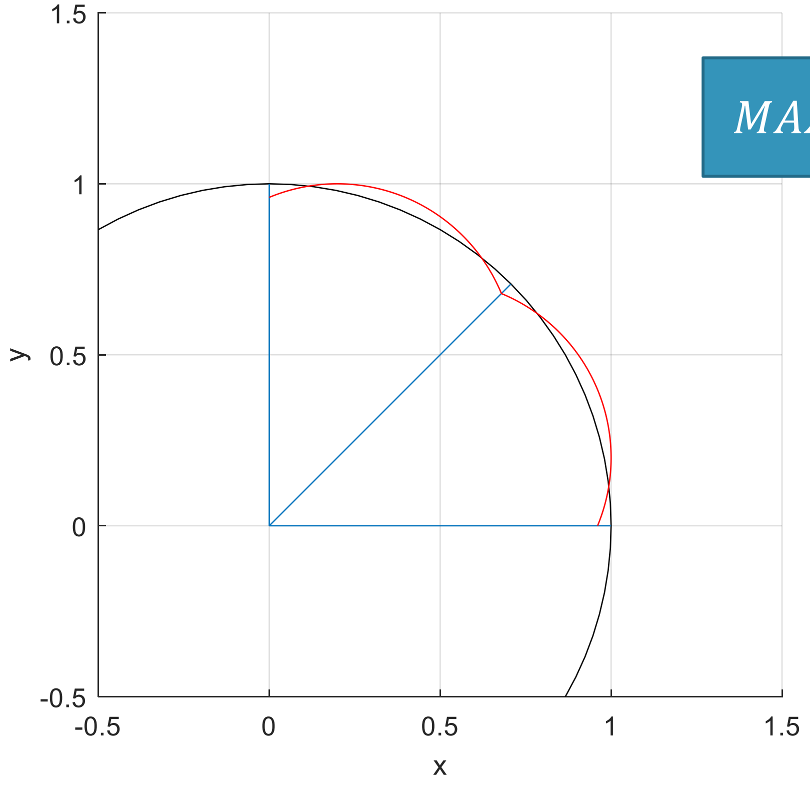
- $\phi(x,y) = \alpha \max(|x|,|y|) + \beta \min(|x|,|y|)$
- If $\alpha = 1$ $\beta = \sqrt{2} - 1$ the max error is on $\theta_{max} = \text{atan}\frac{\beta}{\alpha} = \frac{\pi}{8}$
  $MAX_E \approx 0.0824$ (8.24%)
- We can reduce α, β and balance the errors
- $E(\theta) = |\alpha \cos(\theta) + \beta \sin(\theta) - 1|$
  - $E(0) = 1 - \alpha$
  - $E\left(\frac{\pi}{8}\right) = \alpha \cos\frac{\pi}{8} + \beta \cos\frac{\pi}{8} - 1$
  - $E\left(\frac{\pi}{4}\right) = 1 - \alpha \cos\frac{\pi}{4} - \beta \cos\frac{\pi}{4}$
- We want to find α, β so that: $\begin{cases} E\left(\frac{\pi}{8}\right) = E(0) \\ E\left(\frac{\pi}{4}\right) = E(0) \end{cases}$



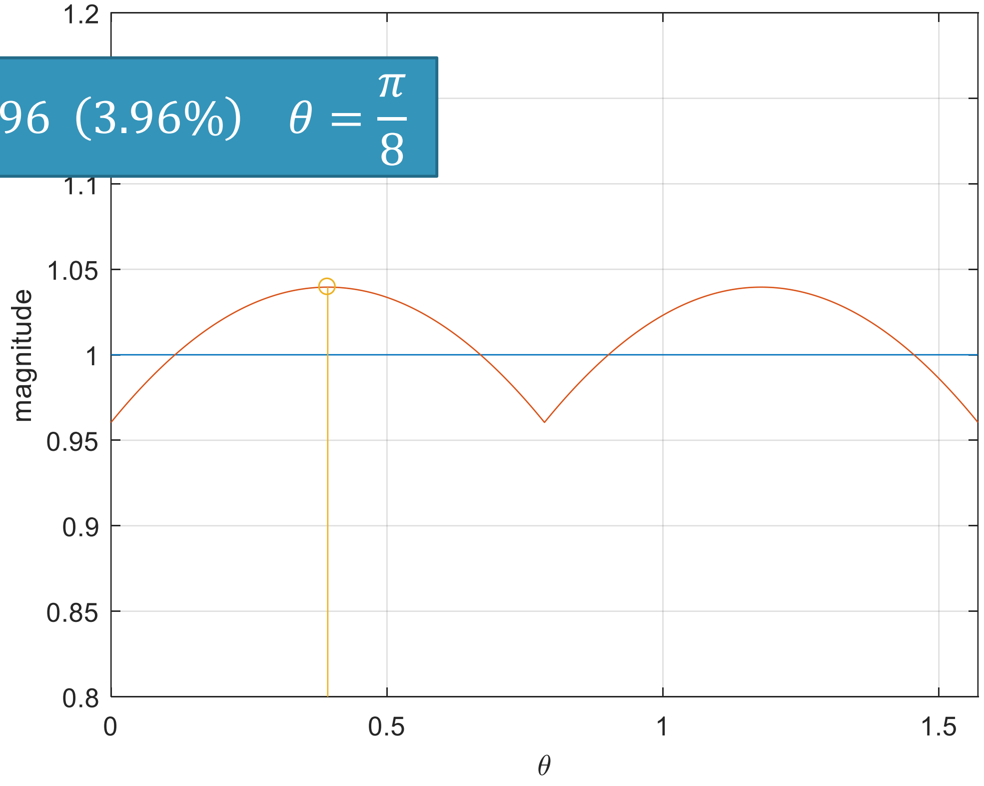
Alpha-max Beta-min

Math | Float | Fixed | RTL

$$\alpha = \frac{2\cos\frac{\pi}{8}}{1+\cos\frac{\pi}{8}} \approx 0.96043\ldots \qquad \beta = \frac{2\sin\frac{\pi}{8}}{1+\cos\frac{\pi}{8}} \approx 0.39782\ldots$$

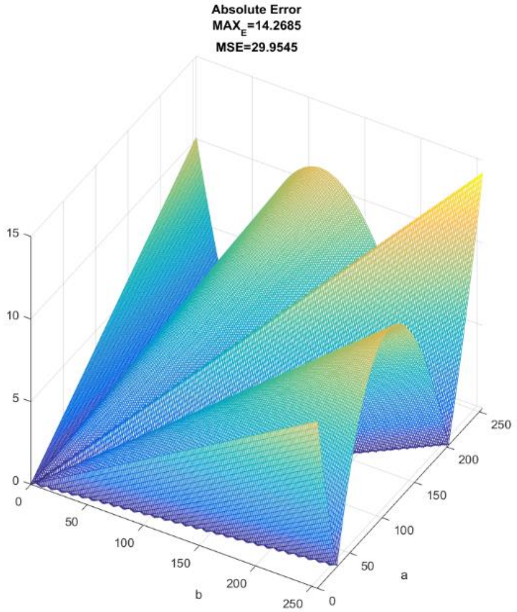


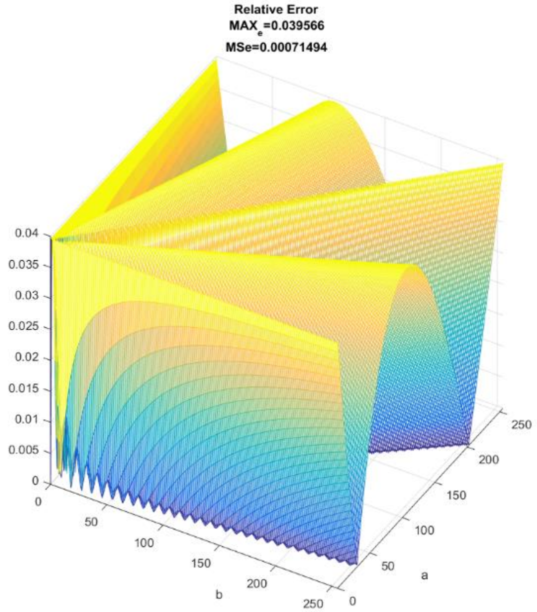
$MAX_E \approx 0.0396\ (3.96\%) \quad \theta = \frac{\pi}{8}$

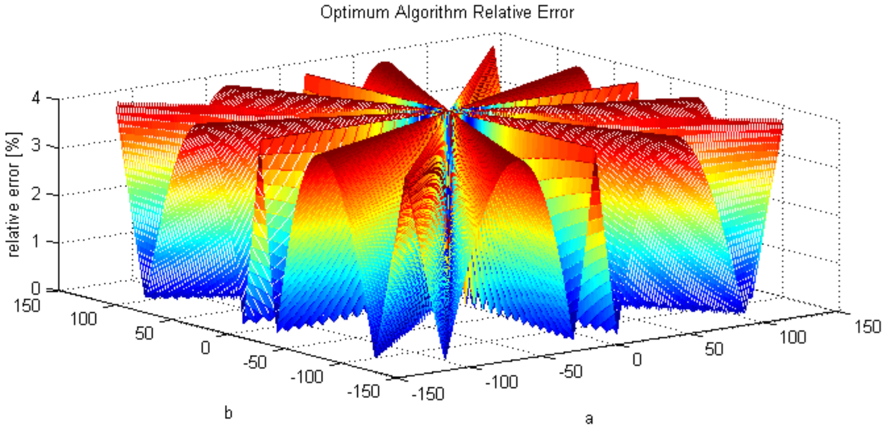


Alpha-max Beta-min

- However, we can write a floating-point model
- Find the optimum (min) of a cost function on 2D space with numerical computations.



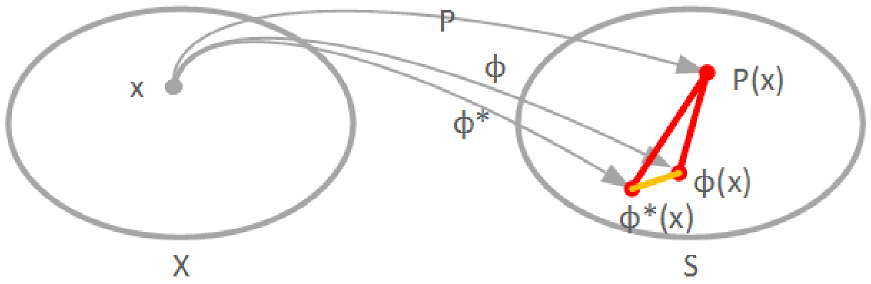




Alpha-max Beta-min

Math | Float | Fixed | RTL

- To achieve bit-true model we have to quantize α, β
- $Q[\alpha] = round\left(\frac{\alpha}{2^{-F}}\right) \times 2^{-F}$
- $Q[\beta] = round\left(\frac{\beta}{2^{-F}}\right) \times 2^{-F}$

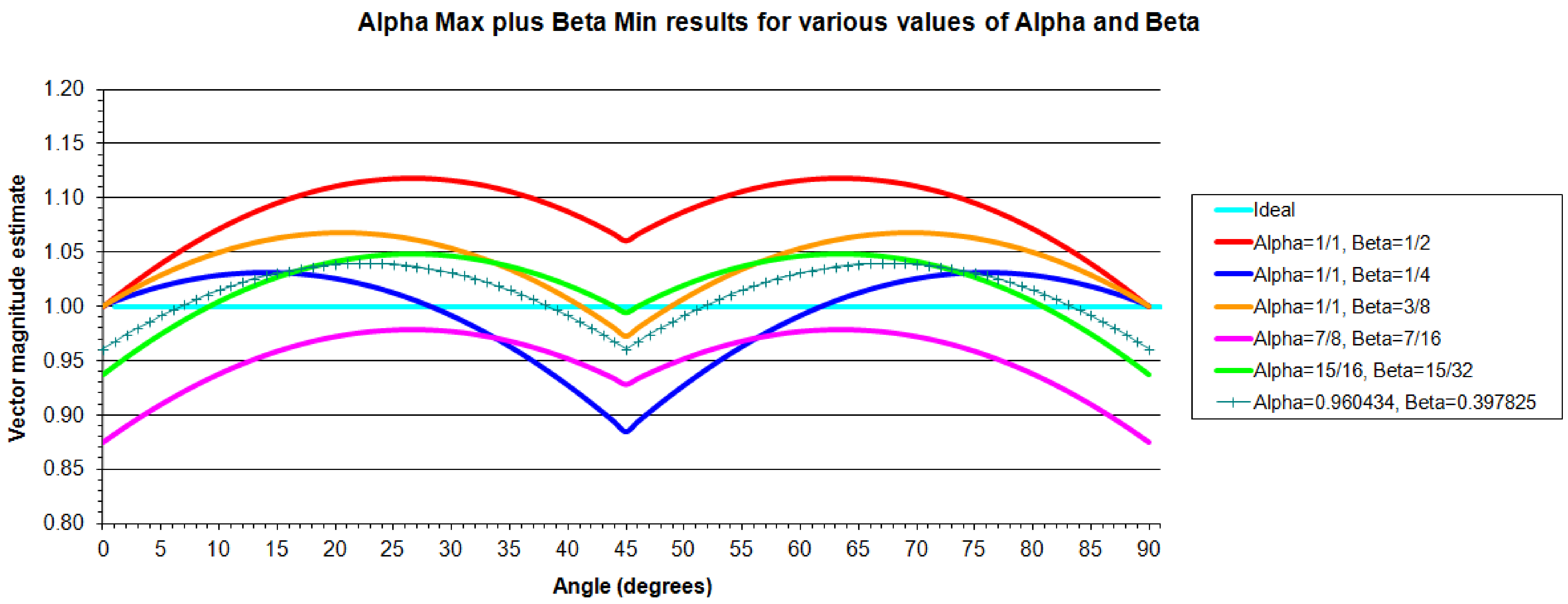



```
F = 1   alpha = 2/2     beta = 1/2      error = 11.8034%
F = 2   alpha = 4/4     beta = 2/4      error = 11.8034%
F = 3   alpha = 8/8     beta = 3/8      error = 6.8%
F = 4   alpha = 15/16   beta = 6/16     error = 7.1891%
F = 5   alpha = 31/32   beta = 13/32    error = 5.0484%
F = 6   alpha = 61/64   beta = 25/64    error = 4.9794%
F = 7   alpha = 123/128 beta = 51/128   error = 4.0266%
F = 8   alpha = 246/256 beta = 102/256  error = 4.0266%
```

MATLAB python C/C++

Alpha-max Beta-min

Math Float Fixed RTL



End

1) Integer Representations
   I. Unsigned
   II. Signed

2) Real Representations
   I. Floating-Point
   II. Fixed-Point

3) Quantization Strategies in Digital Design

4) Bit-true implementation: Alpha-Max Beta-Min example