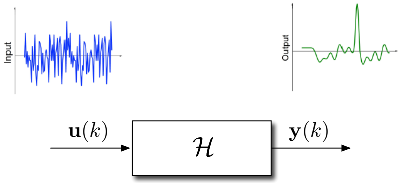
Towards reliable code generation for filters

Anastasia Volkova

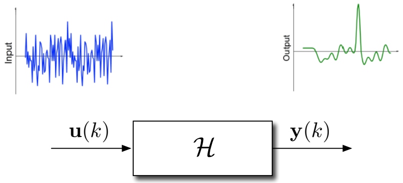
PhD Director: Jean Claude Bajard
PhD Advisors: Thibault Hilaire, Christoph Lauter

May 11, 2016

Context: digital filters



Context: digital filters

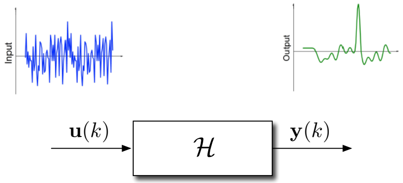


On the one hand

- LTI filter with Infinite Impulse Response
- Its transfer function:

$$H(z) = \frac{\sum_{i=0}^{n} b_i z^{-i}}{1 + \sum_{i=1}^{n} a_i z^{-i}}$$

Context: digital filters



On the one hand

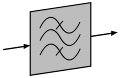
- LTI filter with Infinite Impulse Response
- Its transfer function:

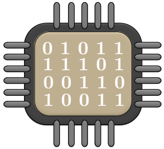
$$H(z) = \frac{\sum_{i=0}^{n} b_i z^{-i}}{1 + \sum_{i=1}^{n} a_i z^{-i}}$$

On the other hand

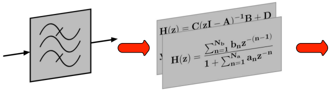
- Hardware or Software target
- Implementation in Fixed-Point Arithmetic

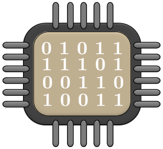
Context: implementation of LTI filters





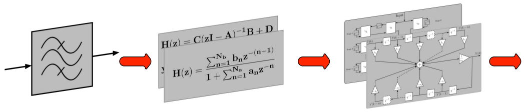
Context: implementation of LTI filters

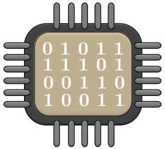

$H(z) = C(zI - A)^{-1}B + D$
$H(z) = \frac{\sum_{n=1}^{N_b} b_n z^{-(n-1)}}{1 + \sum_{n=1}^{N_a} a_n z^{-n}}$



- Transfer function generation

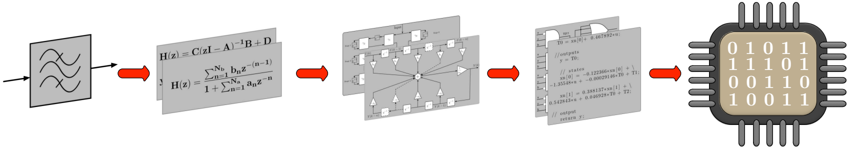
Context: implementation of LTI filters


$H(z) = C(zI - A)^{-1}B + D$
$H(z) = \frac{\sum_{n=1}^{N_b} b_n z^{-(n-1)}}{1 + \sum_{n=1}^{N_a} a_n z^{-n}}$



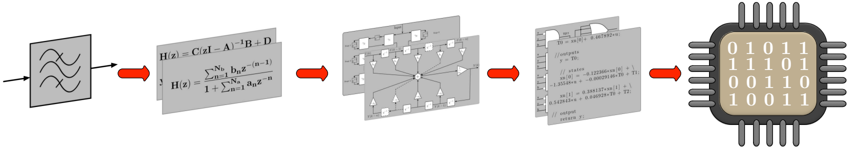
- Transfer function generation
- Algorithm choice: State-space, Direct Form I, Direct Form II, . . .

Context: implementation of LTI filters


$H(z) = C(zI - A)^{-1}B + D$
$H(z) = \frac{\sum_{n=1}^{N_b} b_n z^{-(n-1)}}{1 + \sum_{n=1}^{N_a} a_n z^{-n}}$
0 1 0 1 1
1 1 1 0 1
0 0 1 1 0
1 0 0 1 1

- Transfer function generation
- Algorithm choice: State-space, Direct Form I, Direct Form II, ...
- Software or Hardware implementation

Context: implementation of LTI filters



- Transfer function generation
  - ! Coefficient quantization
- Algorithm choice: State-space, Direct Form I, Direct Form II, . . .
  - ! Large variety of structures with no common quality criteria
- Software or Hardware implementation
  - ! Constraints: power consumption, area, error, speed, etc.
  - ! Computational errors due to finite-precision implementation

Filter-to-code generator

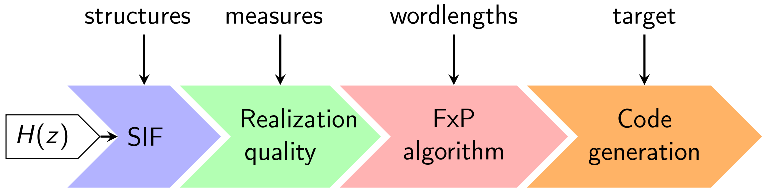


Figure: Automatic filter generator flow.

Filter-to-code generator

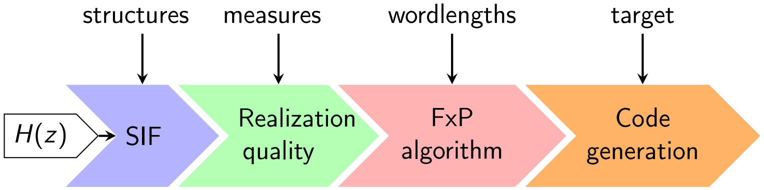


Figure: Automatic filter generator flow.

Before this thesis:

Stage 1: analytical filter realization representation

Stage 2: filter quality measures

Stage 3: fixed-point algorithm (naive approach, computational errors not taken into account)

Stage 4: Fixed-Point Code Generator

Filter-to-code generator

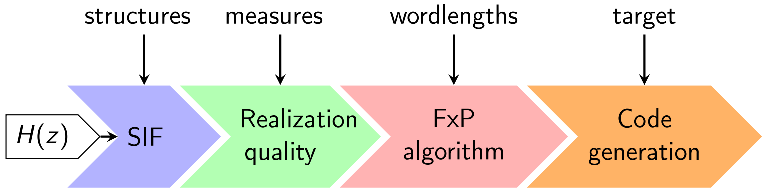


Figure: Automatic filter generator flow.

During this thesis:

Stage 1: analytical filter realization representation

Stage 2: filter quality measures

Stage 3: fixed-point algorithm (~~naive~~ **rigorous** approach, computational errors ~~not~~ **taken into account**)

Stage 4: Fixed-Point Code Generator

Representing Lattice Wave Digital Filters with Specialized Implicit Framework[1].

[1] A.V. et al., "Fixed-Point Implementation of Lattice Wave Digital Filter: Comparison and Error Analysis", in 23rd European Signal Processing Conference, EUSIPCO 2015

Reliable implementation of digital filters in Fixed-Point Arithmetic

LTI filters

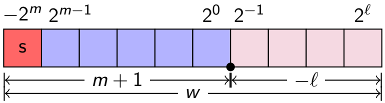
Let $\mathcal{H} := (\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D})$ be a LTI filter:

$$\mathcal{H} \left\{ \begin{array}{rcl} \boldsymbol{x}(k+1) & = & \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}(k) \\ \boldsymbol{y}(k) & = & \boldsymbol{C}\boldsymbol{x}(k) + \boldsymbol{D}\boldsymbol{u}(k) \end{array} \right.$$

The filter $\mathcal{H}$ is considered Bounded Input Bounded Output stable if

$$\rho(\boldsymbol{A}) < 1$$

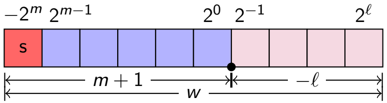
Two's complement Fixed-Point arithmetic



$$y = -2^m y_m + \sum_{i=\ell}^{m-1} 2^i y_i$$

- Wordlength: w
- **M**ost **S**ignificant **B**it position: m
- **L**east **S**ignificant **B**it position: $\ell := m - w + 1$

Two's complement Fixed-Point arithmetic



$$y = -2^m y_m + \sum_{i=\ell}^{m-1} 2^i y_i$$

- $y(k) \in \mathbb{R}$
- wordlength w bits
- minimal Fixed-Point Format (FPF) is the least m:

$$\forall k, \quad y(k) \in [-2^m; 2^m - 2^{m-w+1}]$$

Reliable Fixed-Point implementation

Input:

- $\mathcal{H} = (\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D})$
- bound on the input interval
- wordlength constraints

Determine: the Fixed-Point Formats s.t.

- the least MSBs
- no overflows occur

  $\rightsquigarrow$ must take into account computational errors

Reliable Fixed-Point implementation

Input:

- $\mathcal{H} = (\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D})$
- bound on the input interval
- wordlength constraints

Determine: the Fixed-Point Formats s.t.

- the least MSBs
- no overflows occur

  $\rightsquigarrow$ must take into account computational errors

How to proceed:

1. determine the output interval of all variables
2. analyze propagation of the error in filter implementation and determine the FxPF

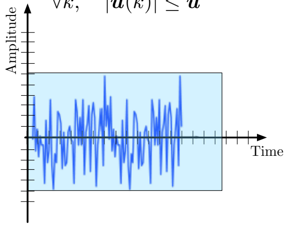
Deducing the output interval[2]

[2] A.V. et al., "Reliable Evaluation of the Worst-Case Peak Gain Matrix in Multiple Precision", ARITH22, 2015

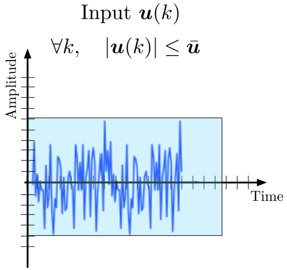
Basic brick: the Worst-Case Peak Gain theorem

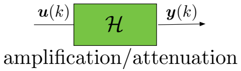
Input $\boldsymbol{u}(k)$

$$\forall k, \quad |\boldsymbol{u}(k)| \leq \bar{\boldsymbol{u}}$$

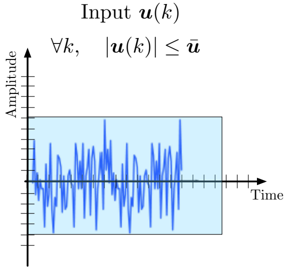


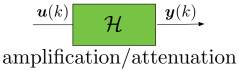
Basic brick: the Worst-Case Peak Gain theorem

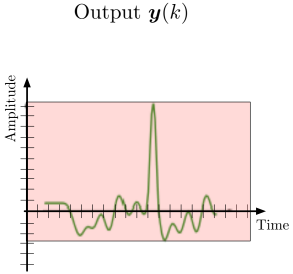




Basic brick: the Worst-Case Peak Gain theorem



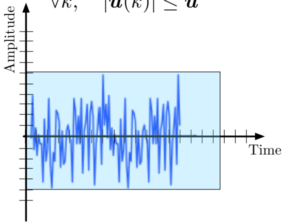


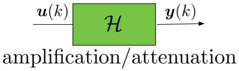


Basic brick: the Worst-Case Peak Gain theorem

Input $\boldsymbol{u}(k)$

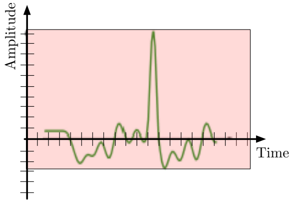
$\forall k, \quad |\boldsymbol{u}(k)| \leq \bar{\boldsymbol{u}}$





Output $\boldsymbol{y}(k)$

$\forall k, \quad |\boldsymbol{y}(k)| \leq \langle\langle\mathcal{H}\rangle\rangle\, \bar{\boldsymbol{u}}$



Worst-Case Peak Gain

$$\langle\langle\mathcal{H}\rangle\rangle = |\mathbf{D}| + \sum_{k=0}^{\infty} |\mathbf{C}\mathbf{A}^k\mathbf{B}|$$

Computing the Worst-Case Peak Gain

Problem: compute the Worst-Case Peak Gain with arbitrary precision.

$$\langle\langle \mathcal{H} \rangle\rangle = |\boldsymbol{D}| + \sum_{k=0}^{\infty} \left| \boldsymbol{C}\boldsymbol{A}^k \boldsymbol{B} \right|$$

- Cannot sum infinitely $\Rightarrow$ need to truncate the sum
- Once the sum is truncated, evaluate it in multiple precision

Truncation

$$\sum_{k=0}^{\infty} \left| \boldsymbol{C}\boldsymbol{A}^k\boldsymbol{B} \right| \quad \longrightarrow \quad \sum_{k=0}^{N} \left| \boldsymbol{C}\boldsymbol{A}^k\boldsymbol{B} \right|$$

Truncation

$$\left| \sum_{k=0}^{\infty} \left| \boldsymbol{C}\boldsymbol{A}^k\boldsymbol{B} \right| \quad - \quad \sum_{k=0}^{N} \left| \boldsymbol{C}\boldsymbol{A}^k\boldsymbol{B} \right| \right| \leq \varepsilon_1$$

Compute an approximate lower bound on truncation order N such that the truncation error is smaller than ε_1.

Lower bound on truncation order N

$$N \geq \left\lceil \frac{\log \frac{\varepsilon_1}{\|\boldsymbol{M}\|_{min}}}{\log \rho(\boldsymbol{A})} \right\rceil, \quad \text{with } \boldsymbol{M} := \sum_{l=1}^{n} \frac{|\boldsymbol{R}_l|}{1 - |\boldsymbol{\lambda}_l|} \frac{|\boldsymbol{\lambda}_l|}{\rho(\boldsymbol{A})}$$

where

$\boldsymbol{\lambda}$ – eigenvalues of matrix $\boldsymbol{A}$

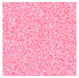
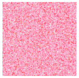
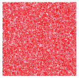
$\boldsymbol{R}_l$ – l^{th}residue matrix computed out of $\boldsymbol{C}, \boldsymbol{B}, \boldsymbol{\lambda}$

Powering

$$\sum_{k=0}^{N} \left| \boldsymbol{C} \boldsymbol{A}^k \boldsymbol{B} \right|$$

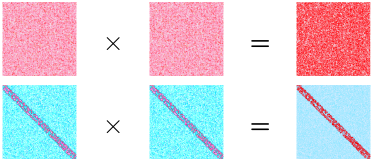
Powering

$$\sum_{k=0}^{N} \left| C A^k B \right|$$

 $\times$  $=$  cancellation

Powering

$$\sum_{k=0}^{N} \left| \boldsymbol{C}\boldsymbol{A}^k\boldsymbol{B} \right|$$

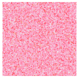
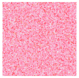
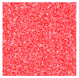


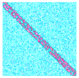
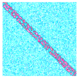
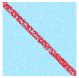
cancellation

less cancellation

Powering

$$\sum_{k=0}^{N} \left| \boldsymbol{C} \boldsymbol{A}^{k} \boldsymbol{B} \right|$$

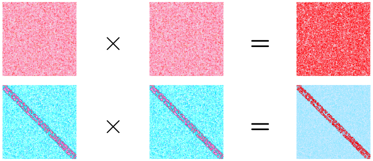
 $\times$  $=$  cancellation

 $\times$  $=$  less cancellation

$$\boldsymbol{A} = \boldsymbol{X} \boldsymbol{E} \boldsymbol{X}^{-1}$$

Powering

$$\sum_{k=0}^{N} \left| \boldsymbol{C} \boldsymbol{A}^k \boldsymbol{B} \right|$$



cancellation

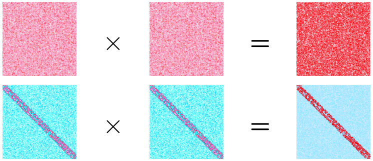
less cancellation

$\boldsymbol{A} = \boldsymbol{X}\boldsymbol{E}\boldsymbol{X}^{-1}$

$\boldsymbol{V} \approx \boldsymbol{X}$ and $\boldsymbol{T} \approx \boldsymbol{E}$

Powering

$$\sum_{k=0}^{N} \left| \boldsymbol{C} \boldsymbol{A}^{k} \boldsymbol{B} \right|$$



cancellation

less cancellation

$$\begin{tikzcd}
\boldsymbol{A} = \boldsymbol{X}\boldsymbol{E}\boldsymbol{X}^{-1} \arrow[dr] & & \boldsymbol{V} \approx \boldsymbol{X} \text{ and } \boldsymbol{T} \approx \boldsymbol{E} \arrow[dl] \\
& \boldsymbol{T} \approx \boldsymbol{V}^{-1} \times \boldsymbol{A} \times \boldsymbol{V} \arrow[d] & \\
& \boldsymbol{A}^{k} \approx \boldsymbol{V} \times \boldsymbol{T}^{k} \times \boldsymbol{V}^{-1} &
\end{tikzcd}$$

Powering

$$\left| \sum_{k=0}^{N} \left| \boldsymbol{C} \boldsymbol{A}^k \boldsymbol{B} \right| \quad - \quad \sum_{k=0}^{N} \left| \boldsymbol{C} \boldsymbol{V} \boldsymbol{T}^k \boldsymbol{V}^{-1} \boldsymbol{B} \right| \right| \leq \varepsilon_2$$

Given matrix $\boldsymbol{V}$ compute $\boldsymbol{T}$ such that the error of substitution of the product $\boldsymbol{V}\boldsymbol{T}^k\boldsymbol{V}^{-1}$ instead of $\boldsymbol{A}^k$ is less than ε_2.

Further steps

$$\left| \sum_{k=0}^{N} \left| \boldsymbol{C} \boldsymbol{A}^k \boldsymbol{B} \right| \quad - \quad \sum_{k=0}^{N} \left| \boldsymbol{C} \boldsymbol{V} \boldsymbol{T}^k \boldsymbol{V}^{-1} \boldsymbol{B} \right| \right| \leq \varepsilon_2$$

Apply the same approach for the other steps:

$$\left| \sum_{k=0}^{N} \left| \boldsymbol{C} \boldsymbol{V} \boldsymbol{T}^k \boldsymbol{V}^{-1} \boldsymbol{B} \right| - \sum_{k=0}^{N} \left| \boldsymbol{C}' \boldsymbol{T}^k \boldsymbol{B}' \right| \right| \leq \varepsilon_3$$

$$\left| \sum_{k=0}^{N} \left| \boldsymbol{C}' \boldsymbol{T}^k \boldsymbol{B}' \right| - \quad \sum_{k=0}^{N} \left| \boldsymbol{C}' \boldsymbol{P}_k \boldsymbol{B}' \right| \right| \leq \varepsilon_4$$

$$\left| \sum_{k=0}^{N} \left| \boldsymbol{C}' \boldsymbol{P}_k \boldsymbol{B}' \right| - \sum_{k=0}^{N} \left| \boldsymbol{L}_k \right| \right| \leq \varepsilon_5$$

$$\left| \sum_{k=0}^{N} \left| \boldsymbol{L}_k \right| - \quad \boldsymbol{S}_N \right| \leq \varepsilon_6$$

Further steps

$$\left| \sum_{k=0}^{N} \left| \boldsymbol{C} \boldsymbol{A}^{k} \boldsymbol{B} \right| \quad - \quad \sum_{k=0}^{N} \left| \boldsymbol{C} \boldsymbol{V} \boldsymbol{T}^{k} \boldsymbol{V}^{-1} \boldsymbol{B} \right| \right| \leq \varepsilon_2$$

Apply the same approach for the other steps:

$$\left| \sum_{k=0}^{N} \left| \boldsymbol{C} \boldsymbol{V} \boldsymbol{T}^{k} \boldsymbol{V}^{-1} \boldsymbol{B} \right| - \sum_{k=0}^{N} \left| \boldsymbol{C}' \boldsymbol{T}^{k} \boldsymbol{B}' \right| \right| \leq \varepsilon_3$$

$$\left| \textstyle\sum_{k=0}^{N} \left| \boldsymbol{C}' \boldsymbol{T}^{k} \boldsymbol{B}' \right| - \quad \textstyle\sum_{k=0}^{N} \left| \boldsymbol{C}' \boldsymbol{P}_k \boldsymbol{B}' \right| \right| \leq \varepsilon_4$$

$$\left| \textstyle\sum_{k=0}^{N} \left| \boldsymbol{C}' \boldsymbol{P}_k \boldsymbol{B}' \right| - \textstyle\sum_{k=0}^{N} \left| \boldsymbol{L}_k \right| \right| \leq \varepsilon_5$$

$$\left| \textstyle\sum_{k=0}^{N} \left| \boldsymbol{L}_k \right| - \quad \boldsymbol{S}_N \right| \leq \varepsilon_6$$

We can determine the output interval of a filter with arbitrary precision.

Determining the Fixed-Point Formats[3]

[3] A.V. et al., "Determining Fixed-Point Formats for a Digital Filter Implementation using the Worst-Case Peak Gain Measure", Asilomar 49, 2015

Determining the Fixed-Point Formats

$$\mathcal{H} \left\{ \begin{array}{rcl} \boldsymbol{x}(k+1) & = & \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}(k) \\ \boldsymbol{y}(k) & = & \boldsymbol{C}\boldsymbol{x}(k) + \boldsymbol{D}\boldsymbol{u}(k) \end{array} \right.$$

We know that if $\forall k, |\boldsymbol{u}_i(k)| \leq \bar{\boldsymbol{u}}_i$, then

$$\forall k, \quad |\boldsymbol{y}_i(k)| \leq \left(\langle\langle\mathcal{H}\rangle\rangle\, \bar{\boldsymbol{u}}\right)_i .$$

Determining the Fixed-Point Formats

$$\mathcal{H} \begin{cases} \boldsymbol{x}(k+1) &= \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}(k) \\ \boldsymbol{y}(k) &= \boldsymbol{C}\boldsymbol{x}(k) + \boldsymbol{D}\boldsymbol{u}(k) \end{cases}$$

We know that if $\forall k, |\boldsymbol{u}_i(k)| \leq \bar{\boldsymbol{u}}_i$, then

$$\forall k, \quad |\boldsymbol{y}_i(k)| \leq \left(\langle\langle \mathcal{H} \rangle\rangle \, \bar{\boldsymbol{u}}\right)_i .$$

We need to find the least $\boldsymbol{m}_y$ such that

$$\forall k, \quad |\boldsymbol{y}_i(k)| \leq 2^{\boldsymbol{m}_{y_i}} - 2^{\boldsymbol{m}_{y_i} - \boldsymbol{w}_{y_i} + 1}.$$

Determining the Fixed-Point Formats

$$\mathcal{H} \left\{ \begin{array}{rcl} \boldsymbol{x}(k+1) & = & \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}(k) \\ \boldsymbol{y}(k) & = & \boldsymbol{C}\boldsymbol{x}(k) + \boldsymbol{D}\boldsymbol{u}(k) \end{array} \right.$$

We know that if $\forall k, |\boldsymbol{u}_i(k)| \leq \bar{\boldsymbol{u}}_i$, then

$$\forall k, \quad |\boldsymbol{y}_i(k)| \leq \left(\langle\langle \mathcal{H} \rangle\rangle\, \bar{\boldsymbol{u}}\right)_i .$$

We need to find the least $\boldsymbol{m}_y$ such that

$$\forall k, \quad |\boldsymbol{y}_i(k)| \leq 2^{\boldsymbol{m}_{y_i}} - 2^{\boldsymbol{m}_{y_i} - \boldsymbol{w}_{y_i} + 1}.$$

We have shown that $\boldsymbol{m}_y$ can be computed with

$$\boldsymbol{m}_{y_i} = \left\lceil \log_2 \left(\langle\langle \mathcal{H} \rangle\rangle\, \bar{\boldsymbol{u}}\right)_i - \log_2 \left(1 - 2^{1-\boldsymbol{w}_{y_i}}\right) \right\rceil .$$

Taking the quantization errors into account

The exact filter $\mathcal{H}$ is:

$$\mathcal{H} \left\{ \begin{array}{rcl} \boldsymbol{x}(k+1) & = & \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}(k) \\ \boldsymbol{y}(k) & = & \boldsymbol{C}\boldsymbol{x}(k) + \boldsymbol{D}\boldsymbol{u}(k) \end{array} \right.$$

Taking the quantization errors into account

The actually implemented filter $\mathcal{H}^\Diamond$ is:

$$\mathcal{H}^\Diamond \left\{ \begin{array}{rcl} \boldsymbol{x}^\Diamond(k+1) & = & \Diamond_{m_x}(\boldsymbol{A}\boldsymbol{x}^\Diamond(k) + \boldsymbol{B}\boldsymbol{u}(k)) \\ \boldsymbol{y}^\Diamond(k) & = & \Diamond_{m_y}(\boldsymbol{C}\boldsymbol{x}^\Diamond(k) + \boldsymbol{D}\boldsymbol{u}(k)) \end{array} \right.$$

where $\Diamond_m$ is some operator ensuring faithful rounding:

$$|\Diamond_m(x) - x| \leq 2^{m-w+1}.$$

Taking the quantization errors into account

The actually implemented filter $\mathcal{H}^\diamondsuit$ is:

$$\mathcal{H}^\diamondsuit \left\{ \begin{array}{rcl} \boldsymbol{x}^\diamondsuit(k+1) & = & \boldsymbol{A}\boldsymbol{x}^\diamondsuit(k) + \boldsymbol{B}\boldsymbol{u}(k) \ + \varepsilon_x(k) \\ \boldsymbol{y}^\diamondsuit(k) & = & \boldsymbol{C}\boldsymbol{x}^\diamondsuit(k) + \boldsymbol{D}\boldsymbol{u}(k) \ + \varepsilon_y(k) \end{array} \right.$$

with

$$|\varepsilon_x(k)| \leq 2^{\boldsymbol{m}_x - \boldsymbol{w}_x + 1} \quad \text{and} \quad |\varepsilon_y(k)| \leq 2^{\boldsymbol{m}_y - \boldsymbol{w}_y + 1}.$$

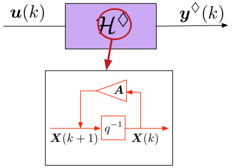
Taking the quantization errors into account

The actually implemented filter $\mathcal{H}^\diamondsuit$ is:

$$\mathcal{H}^\diamondsuit \begin{cases} \boldsymbol{x}^\diamondsuit(k+1) = \boldsymbol{A}\boldsymbol{x}^\diamondsuit(k) + \boldsymbol{B}\boldsymbol{u}(k) + \varepsilon_x(k) \\ \boldsymbol{y}^\diamondsuit(k) = \boldsymbol{C}\boldsymbol{x}^\diamondsuit(k) + \boldsymbol{D}\boldsymbol{u}(k) + \varepsilon_y(k) \end{cases}$$

with

$$|\varepsilon_x(k)| \le 2^{\boldsymbol{m}_x - \boldsymbol{w}_x + 1} \quad \text{and} \quad |\varepsilon_y(k)| \le 2^{\boldsymbol{m}_y - \boldsymbol{w}_y + 1}.$$



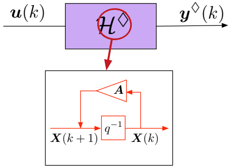
Taking the quantization errors into account

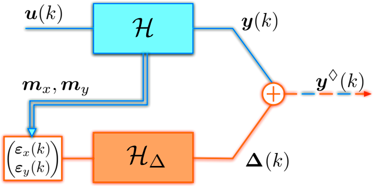
The actually implemented filter $\mathcal{H}^{\Diamond}$ is:

$$\mathcal{H}^{\Diamond} \begin{cases} \boldsymbol{x}^{\Diamond}(k+1) = \boldsymbol{A}\boldsymbol{x}^{\Diamond}(k) + \boldsymbol{B}\boldsymbol{u}(k) + \varepsilon_x(k) \\ \boldsymbol{y}^{\Diamond}(k) = \boldsymbol{C}\boldsymbol{x}^{\Diamond}(k) + \boldsymbol{D}\boldsymbol{u}(k) + \varepsilon_y(k) \end{cases}$$

with

$$|\varepsilon_x(k)| \le 2^{\boldsymbol{m}_x - \boldsymbol{w}_x + 1} \quad \text{and} \quad |\varepsilon_y(k)| \le 2^{\boldsymbol{m}_y - \boldsymbol{w}_y + 1}.$$





$\boldsymbol{u}(k)$
$\mathcal{H}$
$\boldsymbol{y}(k)$
$\boldsymbol{m}_x, \boldsymbol{m}_y$
$\boldsymbol{y}^{\Diamond}(k)$
$\begin{pmatrix}\varepsilon_x(k)\\ \varepsilon_y(k)\end{pmatrix}$
$\mathcal{H}_{\Delta}$
$\boldsymbol{\Delta}(k)$

Numerical example

Example:

- Random filter with 3 states, 1 input, 1 output
- $\bar{u} = 5.125$, wordlengths set to 7 bits

	states			output
	$\boldsymbol{x}_1(k)$	$\boldsymbol{x}_2(k)$	$\boldsymbol{x}_3(k)$	$\boldsymbol{y}(k)$
Step 1	6	7	5	6
Step 2	6	7	6	6
Step 3	6	7	6	6

Table: Evolution of the MSB positions

Numerical examples

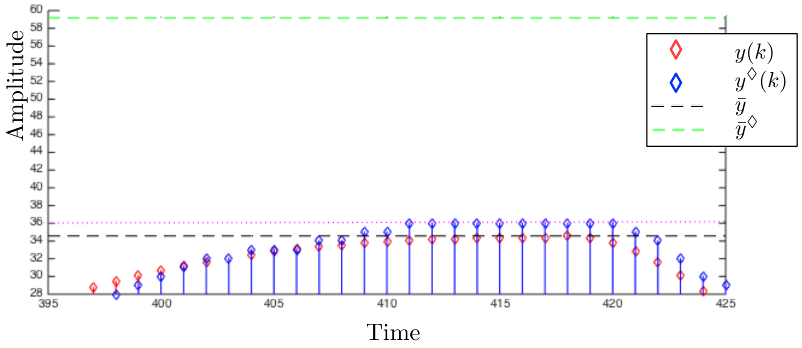


Figure: The exact and quantized outputs of the example.
Quantized output does not pass over to the next binade.

Numerical examples

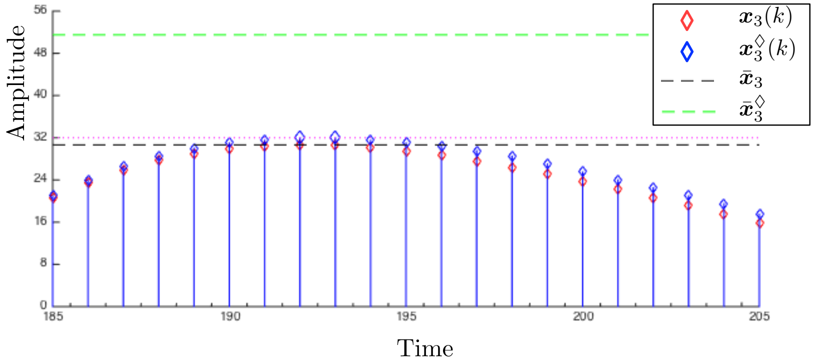


Figure: The exact and quantized third state of the example.

Quantized state passes over to the next binade.

Conclusion

- Represented a new structure in the SIF formalism
- Provided reliable evaluation of the WCPG measure
- Applied the WCPG measure to determine the FxPF that guarantee no overflow

Publications:

"Reliable Evaluation of the Worst-Case Peak Gain Matrix in Multiple Precision", ARITH22

"Determining Fixed-Point Formats for a Digital Filter Implementation using the Worst-Case Peak Gain Measure", Asilomar 49

"Fixed-Point Implementation of Lattice Wave Digital Filter: Comparison and Error Analysis", EUSIPCO 2015

Perspectives

For filter implementation...

- Solve the off-by-one problem for the MSBs
- Get a deeper insight on the behavior of rounding errors
  - $\rightsquigarrow$ determine the probability distribution function
- Plug all the stages of the generator into optimization routines
- Accuracy of the algorithms for the design of IIR filters
  - $\rightsquigarrow$ take coefficient quantization errors into account in the filter error analysis
  - $\rightsquigarrow$ consider coefficients as intervals and make necessary adaptations in the algorithms within the filter

Perspectives

For filter implementation...

- Solve the off-by-one problem for the MSBs
- Get a deeper insight on the behavior of rounding errors
  - $\rightsquigarrow$ determine the probability distribution function
- Plug all the stages of the generator into optimization routines
- Accuracy of the algorithms for the design of IIR filters
  - $\rightsquigarrow$ take coefficient quantization errors into account in the filter error analysis
  - $\rightsquigarrow$ consider coefficients as intervals and make necessary adaptations in the algorithms within the filter

Mathematical function implementation...

- Draw parallels between filter and elementary function implementation.

Merci !

Thank you!

Спасибо!

Дякую!