Fixed-Point implementation of Lattice Wave
Digital Filter: comparison and error analysis

Anastasia Volkova, Thibault Hilaire

Sorbonne Universités, University Pierre and Marie Curie, LIP6,
Paris, France

EUSIPCO 15
September 2, 2015
Motivation

Need to deal with

- Discretize functions and coefficients
  - parametric errors
  - computational errors
- Implementation under constraints
  - software implementation
  - hardware implementation
Motivation

Different filter structures:

- Direct Form I, Direct Form II
- State-space
- Wave, Lattice Wave, ...
- $\rho$-operator: $\rho$DFIIIt, $\rho$State-space...
- LGS, LCW, etc.

Problem:

They are equivalent in *infinite* precision but no more in *finite* precision. The finite precision degradation depends on the realization.
Motivation

Given transfer function and a target, we want:

- Represent various realizations (in an easy way)
- Evaluate finite precision degradation (a priori/a posteriori)
- Find an optimal realization (need to compare realizations)

Tradeoff:

- Error
- Quality
- Power consumption
- Area
- Speed

\[
\text{w.r.t. exact filter}
\]

\[
\text{resources}
\]
Motivation

Given transfer function and a target, we want:

- Represent various realizations (in an easy way)
- Evaluate finite precision degradation (a priori/a posteriori)
- Find an optimal realization (need to compare realizations)

Tradeoff:

- Error
- Quality
- Power consumption
- Area
- Speed

w.r.t. exact filter

resources

Specialized Implicit Framework (SIF)
Outline

1. Motivation
2. Specialized Implicit Framework
3. Lattice Wave Digital Filters
4. LWDF-to-SIF conversion
5. Example and comparison
6. Summary
**SIF: Specialized Implicit Framework**

SIF is:
- Macroscopic description
- Based on state-space
- Explicit all the computations and their order
- Any DFG can be transformed to this form
- Analytical derivation of measures

\[
\begin{align*}
\mathcal{H} \quad J_t(k + 1) & = M x(k) + N u(k) \\
\quad x(k + 1) & = K t(k + 1) + P x(k) + Q u(k) \\
\quad y(k) & = L t(k + 1) + R x(k) + S u(k)
\end{align*}
\]

Denote \( Z \) the matrix containing all the coefficients

\[
Z \triangleq \begin{pmatrix} -J & M & N \\ K & P & Q \\ L & R & S \end{pmatrix}
\]
Measures

- **a priori** measures
  - transfer function sensitivity (based on $\frac{\partial H}{\partial Z}$)
    $\rightarrow$ stochastic measure, takes into account coefficient wordlengths
  - poles or zeros sensitivity (e.g. based on $\frac{\partial |\lambda_i|}{\partial Z}$ for a pole $\lambda_i$)
    $\rightarrow$ stochastic measure, takes into account coefficient wordlengths
  - RNG, ...

- **a posteriori** measures
  - Signal to Quantization Noise Ratio
  - output error
SIF: Worst-Case Peak Gain theorem

\[ \forall k \ |u(k)| \leq \bar{u} \]
SIF: Worst-Case Peak Gain theorem

\[
\langle \mathcal{H} \rangle = ||h||_{l_1}
\]

Input interval

\[\forall k \ |u(k)| \leq \bar{u}\]
SIF: Worst-Case Peak Gain theorem

\[ \forall k \ |u(k)| \leq \bar{u} \]

Worst-Case Peak Gain
\[ \langle \langle \mathcal{H} \rangle \rangle = \|h\|_{l_1} \]

Output interval
\[ \forall k \ |y(k)| \leq \langle \langle \mathcal{H} \rangle \rangle \bar{u} \]
WCPG theorem permits to determine:

- the output error interval

\[
\begin{align*}
    u(k) & \xrightarrow{\mathcal{H}} y(k) \\
    \varepsilon(k) & \xrightarrow{\mathcal{H}_\varepsilon} \Delta y(k) + y^*(k)
\end{align*}
\]

- the Most Significant Bit, therefore Fixed-Point Formats

\[
m_y = \left\lfloor \log_2 (\langle \mathcal{H} \rangle \bar{u}) \right\rfloor + 1
\]

**Equivalent technique:** WCPG-scaling, it guarantees that no overflows occur.

**Fixed Point Code Generator (FiPoGen)**

- Generates bit-accurate fixed-point algorithms
- Optimizes the wordlength under certain criteria (e.g. area)
SIF: from transfer function to Fixed-Point code
Outline

1 Motivation

2 Specialized Implicit Framework

3 Lattice Wave Digital Filters

4 LWDF-to-SIF conversion

5 Example and comparison

6 Summary
Lattice Wave Digital Filters

Motivation SIF LWDF LWDF-to-SIF Example and comparison Summary

Lattice Wave Digital Filters

Stage 0
Stage 1
Stage 2
Stage (n - 1)

Input

High-pass output

Low-pass output

\[ n = 0, 1, 2, \ldots, \frac{N - 1}{2} \]
Lattice Wave Digital Filters

Stage 0
Stage 1
Stage 2
Stage (n – 1)
Stage n

Low-pass output
1/2

High-pass output

n = 0, 1, 2, ..., \frac{N - 1}{2}

\gamma

\text{Input} + \frac{1}{2}

\frac{1}{2}

\text{Output}
Lattice Wave Digital Filters

Two-port adaptor: Richard’s structures

Type 1:
\[
\gamma < \frac{1}{2} \\
\alpha = 1 - \gamma
\]

Type 2:
\[
0 < \gamma \leq \frac{1}{2} \\
\alpha = 1 + \gamma
\]

Type 3:
\[
-\frac{1}{2} \leq \gamma < 0 \\
\alpha = -\gamma
\]

Type 4:
\[
-1 < \gamma < -\frac{1}{2} \\
\alpha = \gamma
\]
Lattice Wave Digital Filters

Positive sides

- parallelizable
- modular, convenient for VLSI
- often referred to as stable

Drawbacks

- Studies of Fixed-Point implementation include complicated infinite-precision optimization
- Comparison is difficult

Objectives

- Represent LWDF in terms of SIF
- Perform *rigorous* error analysis
- Instantly compare with other structures
Outline

1 Motivation

2 Specialized Implicit Framework

3 Lattice Wave Digital Filters

4 LWDF-to-SIF conversion

5 Example and comparison

6 Summary
LWDF-to-SIF conversion
LWDF-to-SIF conversion

Motivation
SIF
LWDF
LWDF-to-SIF
Example and comparison
Summary

Stage 0
Stage 1
Stage 2
Stage (n−1)
Stage (n−1)−1

Low-pass output
1/2
Input
+ 1/2

High-pass output
1/2

γ0
γ1
γ2
γ3
γ4
γ5
γ6
γ2·(n−1)−1
γ2·(n−1)

γ2·n
γ2·n−1

16/23
LWDF-to-SIF conversion

![Diagram of LWDF-to-SIF conversion](image-url)
LWDF-to-SIF conversion

Input

Stage 0

γ0

Stage 1

γ1

Stage 2

γ2

γ4

Stage (n − 1)

γ2−(n−1)

γ2−(n−1)−1

High-pass output

1/2

Low-pass output

1/2

+ +

+ 1/2

γ5

γ3

γ6

Stage 3

Stage n

Stage n
LWDF-to-SIF conversion

![Diagram of LWDF-to-SIF conversion process]

- **Stage 0**:\( \gamma_0 \)
- **Stage 1**:\( \gamma_1 \), \( \gamma_2 \), \( \gamma_3 \)
- **Stage 2**:\( \gamma_4 \)
- **Stage 3**:\( \gamma_5 \), \( \gamma_6 \)
- **Stage \( n \)**:\( \gamma_{2n-1} \), \( \gamma_{2n} \)
- **Stage \( (n-1) \)**:\( \gamma_{2(n-1)} \), \( \gamma_{2(n-1)-1} \)

---

**Input** → **High-pass output**

\[ \frac{1}{2} \]

**Low-pass output** → \( -1 \) → \( \frac{1}{2} \)

---

**Motivation** | **SIF** | **LWDF** | **LWDF-to-SIF** | **Example and comparison** | **Summary**
---|---|---|---|---|---

**LWDF-to-SIF conversion: example**

Convert DFGs of two adaptors into SIFs:

The matrices and equations for the conversions are as follows:

\[
\begin{pmatrix}
1 & 0 & 0 & 0 & 0 \\
-\alpha & 1 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 \\
1 & -1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
t(k+1) \\
x(k+1) \\
y(k)
\end{pmatrix}
= \begin{pmatrix}
0 & 0 & 0 & -1 & 1 \\
0 & 0 & 0 & 0 & -1 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0
\end{pmatrix}
\begin{pmatrix}
t(k) \\
x(k) \\
y(k)
\end{pmatrix}
\]

\[
\begin{pmatrix}
1 & 0 & 0 & 0 & 0 \\
-\alpha & 1 & 0 & 0 & 0 \\
0 & -1 & 1 & 0 & 0 \\
1 & -1 & 0 & 1 & 0
\end{pmatrix}
\begin{pmatrix}
t(k+1) \\
x(k+1) \\
y(k)
\end{pmatrix}
= \begin{pmatrix}
0 & 0 & 1 & -1 & 0 \\
0 & 0 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0
\end{pmatrix}
\begin{pmatrix}
t(k) \\
x(k) \\
y(k)
\end{pmatrix}
\]
LWDF-to-SIF conversion: example

Convert DFGs of two adaptors into SIFs:

LWDF-to-SIF conversion: example

Convert DFGs of two adaptors into SIFs:

\[
\begin{align*}
Z_A & \triangleq \begin{pmatrix}
-J_A & M_A & N_A \\
K_A & P_A & Q_A \\
L_A & R_A & S_A
\end{pmatrix} = \\
&= \begin{pmatrix}
-1 & 0 & 0 & -1 & 1 \\
\alpha & -1 & 0 & 0 & -1 \\
0 & -1 & 0 & 0 & 0 \\
-1 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0
\end{pmatrix},
\end{align*}
\]

\[
\begin{align*}
Z_B & \triangleq \begin{pmatrix}
-J_B & M_B & N_B \\
K_B & P_B & Q_B \\
L_B & R_B & S_B
\end{pmatrix} = \\
&= \begin{pmatrix}
-1 & 0 & 1 & -1 \\
\alpha & -1 & 1 & 0 \\
0 & -1 & 0 & 0 \\
-1 & 1 & 0 & 0
\end{pmatrix}.
\end{align*}
\]
<table>
<thead>
<tr>
<th>Outline</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Motivation</td>
</tr>
<tr>
<td>2 Specialized Implicit Framework</td>
</tr>
<tr>
<td>3 Lattice Wave Digital Filters</td>
</tr>
<tr>
<td>4 LWDF-to-SIF conversion</td>
</tr>
<tr>
<td>5 Example and comparison</td>
</tr>
<tr>
<td>6 Summary</td>
</tr>
</tbody>
</table>
Example and comparison

Reference filter: low-pass 5\textsuperscript{th} order Butterworth filter with cutoff frequency 0.1.

Structures for the comparison:

- LWDF
- state-space
- $\rho$-Direct Form II transposed
- Direct Form I

Normalized (\textit{i.e.} all coefficients have the same wordlength) measures:

- transfer function error: $\tilde{\sigma}_\Delta^2 H$
- pole error: $\tilde{\sigma}_\Delta^2 |\lambda|$
- output error: $\Delta y$
Example and comparison

\[ Z = \begin{pmatrix} \end{pmatrix} \]

LWDF, Z is 22 × 22

\[ Z = \begin{pmatrix} \end{pmatrix} \]

State-Space, Z is 12 × 12

\[ Z = \begin{pmatrix} \end{pmatrix} \]

DFI, Z is 12 × 12
## Example and comparison

| Realization | size(Z) | coeff. | $\bar{\sigma}^2_{\Delta H}$ | $\bar{\sigma}^2_{\Delta |\lambda|}$ | $\bar{\Delta}_y$ |
|-------------|---------|--------|------------------|------------------|-----------------|
| LWDF        | 22×22   | 5      | 0.3151           | 0.56             | 122.9           |
| state-space | 6×6     | 36     | 1.15             | 5.75             | 23.33           |
| $\rho$DFIIt | 11×11   | 11     | 0.09             | 0.45             | 94.3            |
| DFI         | 12×12   | 11     | 1.42e+6          | -                | 7.961           |
Conclusion and perspectives

Conclusion:

- LWDF converted to SIF
- Normalized sensitivity and output error measures applied
- Comparison with several popular structures presented

Perspectives:

- Use VHDL code generator (FloPoCo) to compare hardware implementations
- Apply $\rho$-operator to LWDF
Thank you!

Questions?
SIF: the rigorous filter error bound

Exact filter:

\[
\mathcal{H} \left\{ \begin{align*}
J_t (k + 1) &= M x (k) + N u(k) \\
x (k + 1) &= K_t (k + 1) + P x (k) + Q u(k) \\
y (k) &= L_t (k + 1) + R x (k) + S u(k)
\end{align*} \right. 
\]

where \( \varepsilon_t (k) \), \( \varepsilon_x (k) \) and \( \varepsilon_y (k) \) are the computational errors.

The output error \( \Delta y (k) \equiv y^* (k) - y (k) \) can be seen as the output of a MIMO filter \( H_{\varepsilon} \).
SIF: the rigorous filter error bound

Implemented filter:

\[ H^* \begin{cases} J^*(k + 1) = Mx^*(k) + Nu(k) + \varepsilon_t(k) \\ x^*(k + 1) = Kt^*(k + 1) + Px^*(k) + Qu(k) + \varepsilon_x(k) \\ y^*(k) = Lt^*(k + 1) + Rx^*(k) + Su(k) + \varepsilon_y(k) \end{cases} \]

where \( \varepsilon_t(k) \), \( \varepsilon_x(k) \) and \( \varepsilon_y(k) \) are the computational errors.
SIF: the rigorous filter error bound

Implemented filter:

\[
\mathcal{H}^* \left\{ \begin{array}{l}
J^*(k+1) = Mx^*(k) + Nu(k) + \varepsilon_t(k) \\
x^*(k+1) = Kt^*(k+1) + Px^*(k) + Qu(k) + \varepsilon_x(k) \\
y^*(k) = Lt^*(k+1) + Rx^*(k) + Su(k) + \varepsilon_y(k)
\end{array} \right.
\]

where \( \varepsilon_t(k) \), \( \varepsilon_x(k) \) and \( \varepsilon_y(k) \) are the computational errors. The output error

\[
\Delta y(k) \triangleq y^*(k) - y(k)
\]

can be seen as the output of a MIMO filter \( \mathcal{H}_{\varepsilon} \).
SIF: the rigorous filter error bound

Implemented filter:

\[
\mathcal{H}^* \begin{cases}
Jt^*(k + 1) = Mx^*(k) + Nu(k) + \varepsilon_t(k) \\
x^*(k + 1) = Kt^*(k + 1) + Px^*(k) + Qu(k) + \varepsilon_x(k) \\
y^*(k) = Lt^*(k + 1) + Rx^*(k) + Su(k) + \varepsilon_y(k)
\end{cases}
\]

where \( \varepsilon_t(k) \), \( \varepsilon_x(k) \) and \( \varepsilon_y(k) \) are the computational errors. The output error

\[
\Delta y(k) \triangleq y^*(k) - y(k)
\]

can be seen as the output of a MIMO filter \( \mathcal{H}_\varepsilon \).
SIF: the rigorous filter error bound

Implemented filter:

\[ \mathcal{H}^* \left\{ \begin{array}{l}
Jt^*(k + 1) = Mx^*(k) + Nu(k) + \varepsilon_t(k) \\
x^*(k + 1) = Kt^*(k + 1) + Px^*(k) + Qu(k) + \varepsilon_x(k) \\
y^*(k) = Lt^*(k + 1) + Rx^*(k) + Su(k) + \varepsilon_y(k)
\end{array} \right. \]

where \( \varepsilon_t(k) \), \( \varepsilon_x(k) \) and \( \varepsilon_y(k) \) are the computational errors. The output error

\[ \Delta y(k) \triangleq y^*(k) - y(k) \]

can be seen as the output of a MIMO filter \( \mathcal{H}_\varepsilon \).

WCPG theorem on \( \mathcal{H}_\varepsilon \) gives the output error interval.