A PAM-4 behavioral model using Laguerre-Volterra feed forward neural network and its implementation in IBIS-AMI

Xinying Wang, Thong Nguyen, José E Schutt-Ainé

European Virtual IBIS Summit at Virtual SPI2021
May 12, 2021
Outline

• Introduction and motivation
• Volterra series and Laguerre-Volterra feed forward neural network (LVFFN)
• PAM-4 behavior modeling using LVFFN
• IBIS-AMI model implementation and ezAMI software
• Summary
Introduction

- **High speed link technology**
  - Non-Return-to-Zero (NRZ)
  - Pulse Modulation 4-level (PAM-4)

- **Behavioral model**
  - A compact mathematical model
  - Fast and accurate
  - IP protection

- **Artificial neural network**
  - A mathematical mapping of input to output
  - FFN, CNN, RNN
IBIS-AMI Model Basics

- An interface to model SerDes IP in commercial EDA simulators
- Interoperable, flexible, high performance, and IP protection

**AMI_Init()**
Impulse response processing

**AMI_GetWave()**
Waveform Processing

**AMI_Close()**
Clean-up and exit

Diagram:
- Analog part (.snp)
- Algorithmic part (.dll .ami)
- Tx DLL -> channel -> Rx DLL
- Equalized impulse response
- Equalized waveform
- Clock “ticks”
Motivation and focus of this work

- High demand for rapid data transmission.
  - 100Gb/s -> 400Gb/s

- Simulation and validation are desirable to cut development time and cost
  - A high-performance model is the key

- Behavioral model using machine learning is emerging
  - Large model size
  - Lack of interoperability and transportability

- High speed link behavioral model development
  - A machine learning behavioral model
  - Take nonlinearity into accountable
  - Fast and accurate

- IBIS-AMI model generation
  - Enhance model interoperability and transportability
  - Compatible with commercial circuit simulator

- IBIS-AMI model generation software
Volterra Series

- A versatile model for nonlinear systems with memory

\[
y(t) = y_0 + \sum_{n=1}^{\infty} y_n(t)
\]

\[
y_n(t) = \int_{-\infty}^{+\infty} \cdots \int_{-\infty}^{+\infty} h_n(\tau_1, \tau_2, \ldots, \tau_n) u(t - \tau_1) u(t - \tau_2) \ldots u(t - \tau_n) d\tau_1 d\tau_2 \ldots d\tau_n
\]

In discrete time

\[
y(t) = h_0 + \sum_{\tau_1=0}^{\infty} h_1(\tau_1) u(t - \tau_1) + \sum_{\tau_1=0}^{\infty} \sum_{\tau_2=0}^{\infty} h_2(\tau_1, \tau_2) u(t - \tau_1) u(t - \tau_2) + \ldots
\]

\[h_0, h_1, h_2, \ldots, h_n\] are the Volterra kernels
Monomial Power Series Neural Network

\[
y[t] = \sum_{i=1}^{M} c_i \sigma \left( b_i + \sum_{j=1}^{M} w_{ji} u[t - j] \right)
\]

\[
y(t) = h_0 + \sum_{\tau_1=0}^{M} h_1(\tau_1) u(t - \tau_1) + \sum_{\tau_1=0}^{M} \sum_{\tau_2=0}^{M} h_2(\tau_1, \tau_2) u(t - \tau_1) u(t - \tau_2) + \ldots
\]

X Wang, T Nguyen, J Schutt-Aine, EMC Sapporo & APHEME 2019
T Nguyen, X Wang, J Schutt-Aine, ECTC 2019
Challenges with Volterra Series

- Large number of parameters when nonlinearity order goes high

<table>
<thead>
<tr>
<th>Number of Volterra Kernels</th>
<th>1\textsuperscript{st} order</th>
<th>2\textsuperscript{nd} order</th>
<th>3\textsuperscript{rd} order</th>
<th>4\textsuperscript{th} order</th>
</tr>
</thead>
<tbody>
<tr>
<td>M = 10</td>
<td>10</td>
<td>110</td>
<td>1,110</td>
<td>11,110</td>
</tr>
<tr>
<td>M = 20</td>
<td>20</td>
<td>420</td>
<td>8,420</td>
<td>168,420</td>
</tr>
<tr>
<td>M = 30</td>
<td>30</td>
<td>930</td>
<td>27,930</td>
<td>837,930</td>
</tr>
<tr>
<td>M = 40</td>
<td>40</td>
<td>1,640</td>
<td>641,640</td>
<td>3,201,640</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Number of Laguerre Parameters</th>
<th>1\textsuperscript{st} order</th>
<th>2\textsuperscript{nd} order</th>
<th>3\textsuperscript{rd} order</th>
<th>4\textsuperscript{th} order</th>
</tr>
</thead>
<tbody>
<tr>
<td>R = 2</td>
<td>2</td>
<td>6</td>
<td>14</td>
<td>30</td>
</tr>
<tr>
<td>R = 3</td>
<td>3</td>
<td>12</td>
<td>39</td>
<td>120</td>
</tr>
<tr>
<td>R = 4</td>
<td>4</td>
<td>20</td>
<td>84</td>
<td>340</td>
</tr>
<tr>
<td>R = 5</td>
<td>5</td>
<td>30</td>
<td>155</td>
<td>780</td>
</tr>
</tbody>
</table>
Laguerre-Volterra Feed Forward Neural Network (LVFFN)

- Reduce dimension through projecting Volterra kernels on finite number of Laguerre Functions

\[ h(t) = \sum_{r=1}^{R} \theta_r \varphi_r(t) \]

- Typically, \( R \) is much smaller compared to the memory length \( M \)
Discrete Time Laguerre Functions

\[
\varphi_r(t) = \alpha \frac{t-r}{2} (1 - \alpha)^{\frac{1}{2}} \sum_{k=0}^{r} (-1)^k \binom{t}{k} \binom{r}{k} \alpha^{r-k} (1 - \alpha)^k
\]

- Guaranteed to converge at infinity
- Functions are orthogonal basis
  \[< \varphi_i, \varphi_j > = \begin{cases} 
  0 & i \neq j \\
  \sigma & i = j
\end{cases} \]
- \( \alpha \) is the decay factor which takes value \((0,1)\)
- \( r \) is the function order
From Volterra Space to Laguerre Space

Volterra Series

\[ y(t) = h_0 + \sum_{\tau_1=0}^{M} h_1(\tau_1)u(t - \tau_1) + \sum_{\tau_1=0}^{M} \sum_{\tau_2=0}^{M} h_2(\tau_1, \tau_2)u(t - \tau_1)u(t - \tau_2) + \cdots \]

Laguerre-Volterra model

\[ y(t) = \theta_0 + \sum_{r_1=1}^{R} \theta_1(r_1)v_{r_1}(t) + \sum_{r_1=1}^{R} \sum_{r_2=1}^{R} \theta_2(r_1, r_2)v_{r_1}(t)v_{r_2}(t) + \cdots \]
Identification of Laguerre Coefficients

X Wang, T Nguyen, J Schutt-Aine, TCPMT 2020
X Wang, T Nguyen, J Schutt-Aine, LASCAS 2020
Modeling PAM-4 system

- LVFFN architecture: one hidden layer with 10 neurons
- FFN architecture: one hidden layer with 150 neurons
- RNN architecture:
  - 100 memory length
  - 6 stacked layers
  - 20 neurons for each layer
Model Size Reduction and Computation Efficiency

Model size comparison

- Volterra: $1,698,826$
- RNN: $484,000$
- FFN: $22,801$
- LVFFN: $111$

Computation efficiency comparison

- Volterra: $3,397,650$
- RNN: $485,200$
- FFN: $22,650$
- LVFFN: $1610$
Volterra Kernel Extraction with LVFFN

\[ h_0 = \theta_0 \]

\[ h_1(\tau) = \sum_{r=1}^{r=R} \theta_r \phi_r(\tau) \]

\[ h_2(\tau_1, \tau_2) = \sum_{r_1=1}^{r_1=R} \sum_{r_2=1}^{r_2=R} \theta_{r_1, r_2} \phi_{r_1}(\tau_1) \phi_{r_2}(\tau_2) \]

\[ h_n(\tau_1, ..., \tau_n) = \sum_{r_1=1}^{r_1=R} ... \sum_{r_n=1}^{r_n=R} \theta_{r_1, ..., r_n} \prod_{l=1}^{n} \phi_l(\tau_l) \]
Implementation in IBIS-AMI

Simulating 1 million bits takes 142s!
IBIS-AMI Model Generation

- Model generation requires cross-disciplinary knowledge
- ezAMI software facilitates the model generation

Why AMI-model generation takes so long?

Typical Signal Integrity Engineers are NOT programmers

...they are having “Nightmares” in trying to develop AMI models

- Cryptic Matlab/C++ code passed from System-Architectures → AMI Modeler (if lucky)
- Challenge to Convert Algorithm design Code → AMI format

0 months
AMI 101, Decipher Code → 8 months → First-model to Customer
Nightmare Begins → 4 months → Early Model prototypes → 12 months

ezAMI software

- IBIS-AMI model generator and simulator
- Supports model pre-generation verification
- Allows on-site development and debug
- Supports NRZ and PAM-4 simulation
- Project-based development
- User-friendly GUI system
Software interface

A. Project Navigator

B. Code Snippet:
   ```c
   #include "ami.h"
   
   long AMI_Init(double *impulse_matrix,
                  long row_size,
                  long agxTensors,
                  double sample_interval,
                  double bit_time,
                  char **AMI_parameters_in,
                  char **AMI_parameters_out,
                  void **AMI_memory_handle,
                  char **msg) {

      return 1;
   }
   
   /**********************************************************/
   long AMI_GetWave(double *wave,
                  long wae_size,
                  long agxTensors,
                  double *clock_times,
                  char **AMI_parameters_out,
                  void *AMI_memory) {

      return 1;
   }
   
   /**********************************************************/
   long AMI_Close(void *AMI_memory) {

      return 1;
   }
   ```

C. Diagram of AMI Model

D. Flowchart of AMI Process
Software download links

- [https://gitlab.engr.illinois.edu/xinying/ezami](https://gitlab.engr.illinois.edu/xinying/ezami)
- [https://github.com/WXY163/ezAMI](https://github.com/WXY163/ezAMI)
- Installer: [https://github.com/WXY163/ezAMI/tree/master/installer](https://github.com/WXY163/ezAMI/tree/master/installer)
Quick demo…
Summary

- Proposed a Laguerre-Volterra feed forward neural network (LVFFN) which can significantly reduce the model size and enhance the computation efficiency for modeling PAM-4 systems
- Implemented the PAM-4 LVFFN model in IBIS-AMI and simulated it in industrial EDA tools
- Developed an IBIS-AMI model generation software ezAMI which can help developing non-traditional SERDES IBIS-AMI model.
Thank you!