1A5 Behavioral Model Development for High-Speed Links

PIs: M. Swaminathan, P. Franzon & J. Schutt-Aine
Post Doc: M. Larbi & K. Roy
Students: H. Yu, M. Ahadi, T. Nguyen and B. Li
1A5 Behavioral Model Development for High-Speed Links

Multiple University Project

1. Building models using time domain data, neural networks and polynomial chaos (new)
   M. Swaminathan, GT

2. Building models using X-parameters and neural networks
   J. Schutt-Ainé, UIUC

3. Building receiver models using system identification and system modeling
   P. Franzon, NCSU
1A5 Behavioral Model Development for High-Speed Links (Time Domain)

Project Definition

- Develop methods that use ML for the automated behavioral modeling of complex circuits that:
  1. Has same accuracy as transistor-level models.
  2. Protects IP. Requires >50X less CPU time.
  3. Is significantly more accurate than IBIS.
  4. Is parameterized, accounts for varying channel conditions and compatible with existing tools
- Relevance to Industry: Accurate models required for system level simulation that protects IP; IBIS models are inaccurate

Results and Significance

- Developed parametrized behavioral models of I/O circuits for signal integrity analysis (Ex: Qualcomm); 1000X speed-up; Quantified accuracy
- Behavioral modeling that combines signal & power integrity in progress
- Developed a preliminary polynomial chaos based surrogate model to estimate the statistics of output (e.g. jitter, BER) depending on input random variables (input pulse); Applied to simple & complex circuits (Ex: Qualcomm)

Future Outlook

- Continued behavioral modeling of transistor circuits that combines SI & PI (Qualcomm)
- Estimation of the statistics of outputs (including all sources of noise and jitter) depending on input random variables. Application to HSSCDR (IBM). Possibility for Qualcomm drivers (in discussion)

Milestones: Progress

✓ Year 1: Preliminary methods that use ML for the generation of beh. models for circuits used in high-speed signaling. Comparison for accuracy/speed.
- Year 2: Compatible (w/ commercial tools) automated software for behavioral model and SPICE netlist generation; Tech. transfer
- Deliverables: Year 1: Beh. models w/ accuracy, speed, and memory for 2 ports; Year 2: Software that automates behavioral modeling and SPICE netlist generation (>2 ports; >50X speed-up)
- Publications: EPEPS ’17 (pub); IMS ‘18 (accepted); EMC+SIPI ‘18 (acc); ITC ‘18 (sub), ICCAD ‘18 (pro)

Presentation: CDNLive ’18 Silicon Valley
Best Presentation Award, Academic Track

Relevance to Industry: Accurate models required for system level simulation that protects IP; IBIS models are inaccurate
Parametrized behavioral models

- Model equations

\[ I_o(k) = w_1(k) \cdot I_{o1}(k) + w_2(k) \cdot I_{o2}(k) \]

Submodels for HIGH and LOW logic state

\[ I_{oi}(k) = f_{i}^{RNN} \left( \begin{array}{c} V_o(k), V_o(k-1), V_o(k-2), \\ I_{oi}(k-1), I_{oi}(k-2), \end{array} \right) \]

Control Parameters

\[ w_i(t) = g_{i}^{NN} \left( \begin{array}{c} w_{i,\text{start}}, t_{\text{transit}}, \end{array} \right) \]

Control Parameters

Objectives and Accomplishment
- Developed parametrized behavioral models of transistor circuits that are compatible with circuit simulators
- Applied to examples provided by Qualcomm
- Note: Qualcomm cannot share transistor level circuits but can share simulation data; Qualcomm has provided 40K simulations (Mentors: Jaemin Shin & Tim Michalka)
Objectives and Accomplishment

- Developed parametrized behavioral models of transistor circuits that are compatible with circuit simulators
- Applied to examples provided by Qualcomm
- Note: Qualcomm cannot share transistor level circuits but can share simulation data; Qualcomm has provided 40K simulations (Mentor: Jaemin Shin & Tim Michalka)

Parametrized behavioral models (cont.)

Example
- Corner MCPVT = -1
- Control parameters:
  - Ron.pu = 60
  - Ron.pd = 34
  - EQ2 = 2
  - EQ1 = 4
- Load:
  - TL = 20 mm
  - R = 60 Ohm
  - C = 0.7 pF

- Driver behavioral model achieves ~1000X simulation speed-up
Figure of Merit

Defined by Qualcomm to quantify accuracy of Behavioral Model

<table>
<thead>
<tr>
<th>Test case 1</th>
<th>Near-end</th>
<th>Far-end</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Full voltage range</td>
<td>Restricted voltage range</td>
</tr>
<tr>
<td>FOM1</td>
<td>99.305768</td>
<td>98.344409</td>
</tr>
<tr>
<td>FOM2</td>
<td>8.215724</td>
<td>7.054134</td>
</tr>
<tr>
<td>FOM3</td>
<td>0.029090</td>
<td>0.024977</td>
</tr>
</tbody>
</table>

FOM1: \[ FOM = 100 \cdot \left[ 1 - \frac{\sum_{i=1}^{N} |X_i(golden) - X_i(DUT)|}{\Delta X \cdot N} \right] \]

\( \Delta X \): range of data

\( N \): number of data points

FOM2: \[ FOM2 = 100 \cdot \left[ \frac{\max |X_i(golden) - X_i(DUT)|}{\Delta X} \right] \]

FOM3: \[ FOM3 = \max [ |X_i(golden) - X_i(DUT)| ] \]
Stochastic analysis (Preliminary)

- **Objectives and Accomplishment**
  - Developed a polynomial chaos (PC) surrogate model to estimate the statistics of output (e.g. jitter, BER) considering input pulse as random variables.
  - Being applied to examples provided by Qualcomm (Mentors: Jaemin Shin & Tim Michalka).
  - Being applied to HSSCDR from IBM for different topologies (Mentors: Jose Hejase & Dale Becker).

**High frequency signal transmission**

- Input pulse modeled by random variables.
- Variability of outputs (e.g. jitter, BER).

**Output: Near-end pulse**

- Training data = 8000 bits,
  Order of PC expansion = 2
- Bits considered for ISI = 20,
  Testing data = 100k
Objectives and Accomplishment

- Developed a polynomial chaos (PC) surrogate model to estimate the statistics of output (e.g. jitter, BER) considering input pulse as random variables
- Applied to an example provided by Qualcomm (Mentors: Jaemin Shin & Tim Michalka)
- Applied to HSSCDR from IBM for different topologies (Mentors: Jose Hejase & Dale Becker)

Training data = 8000 bits, Order of PC expansion = 2
- bits considered for ISI = 20, testing data = 100k

Near-end pulse statistics

<table>
<thead>
<tr>
<th>PC</th>
<th>MC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rising edge jitter RMS</td>
<td>1.20 ps</td>
</tr>
<tr>
<td>Falling edge jitter RMS</td>
<td>2.68 ps</td>
</tr>
<tr>
<td>Number of bits</td>
<td>8000 (Training data)</td>
</tr>
</tbody>
</table>
### Project Definition
- Use X-parameters and ML for buffer models
- Relevance to Industry: Alternative to IBIS
- Identify Volterra kernels from frequency-domain measurements of X parameters
- Use X-parameters to improve/enhance IBIS models
- Generation of nonlinear stamps for time-domain simulator (e.g. SPICE)

### Results and Significance
- X-parameters contain information relevant to IBIS models (X2ibis)
- K-mean method for IBIS-IV extraction shows robustness
- Volterra kernel (VK) extraction will lead to more accurate simulation
- 2-tiered approach (rational and impulse function approximation)

### Progress
- Milestones: proof of concept established
- Pole/residue learner and rational matrix builder
- Extraction of IBIS static curves from X-parameters through ML
- Deliverables: in progress (python and C/C++ codes)
- 2 conference papers (Radio F17, and APEMC-18)

### Future Outlook
- Extend K-mean method for IBIS Vt curve extraction
- Identify Volterra kernel coefficients.
- Define SPICE stamps from kernels
- Outlook is to offer a neural-based behavioral model extraction module that is pluggable into existing EDA simulation tools
X-parameter for Buffer Modeling

Cost function

S

Optim

Pole/residue learner

Rational matrix builder

H₂ cost

S, f

X-Parameters

Machine Learning

Volterra Kernel Extraction

Low-Order Nonlinear Model

Behavioral Model

K-mean classification

X-Parameters for Buffer Modeling
Heat Map of X-Parameters

One-port X-parameter data is vectorized for all types and all frequencies and mapped to two-dimensional image.

X-parameter – DC term

X-parameter – RF term

X-parameter – very high frequency term
X-parameter for IBIS generation

- X-parameter as in the neural network’s eye

Pull-up

Pull-down
• X parameter for static IBIS curve is generated.
• An unsupervised classification algorithm (as simple as K-mean) is used to verify that pull-up and pull-down data is separable.
• Watch out for potential outliers: recollect data, unusual dynamic nature between pull-up and pull-down.
• Then the X-parameters of pull-up or pull-down configurations and its corresponding static curves are used to train a FFN.
X-parameter for IBIS generation

- Use importance sampling with assumed Gaussian distribution to bootstrap the result due to limited number of training samples.
- Need more (a lot more) practical data for further investigations.
1A5 Behavioral Model Development for High-Speed Links (Receivers)

Project Definition
- Predict a signal accurately at the output of the receiver (RX) based on the input signal.
- Use system identification to add power-supply-induced jitter into the input signal and build a receiver model.
- For both simulation and measured systems

Results and Significance
- Compared linear System Identification models, ARX, ARMAX, State Space; Among linear System Identification models, State Space models show better accuracy.
- Compared State Space model and nonlinear system identification models. Nonlinear models have the best performance.

Progress
- Measured receiver input and output data and use input-output data pair to build different types of system identification models.
- Compared different model performances and select the best one for future use.
- Turned to nonlinear modeling.

Future Outlook
- Adjoint State-Space Dynamic Neural Network Technique for Nonlinear Modeling.
- Power supply induced jitter classification.
Objectives and Experimental Plan (Receivers)

• Improve model performance using Adjoint State-Space Dynamic Neural Network Technique.

• Determine how to classify PSIJ

• Determine how to model PSIJ
### Project Definition

- In modern high-speed chip to chip SerDes (Serializer-Deserializer) link, the eye diagram at the receiver input is often closed. A receiver behavioral model is needed to predict the inner signal at the output of the receiver.
- Our goal is to predict RX output waveform in a short time using receiver models.

### Results and Significance

- Preferred modeling technique changes with size of training set:
  - Linear models can be reasonably accurate with small training sets.
  - Non-linear models can be more accurate but only with large data sets (many hours of sim time).

### Progress

- Evaluated different model types for one specific receiver (Avago chip).
- Evaluated different model types for a RX modeled behaviorally, CppSim.
- Investigating how to model jitter impact of power supply noise.

### Future Outlook

- Continue analyze power supply induced jitter model in the receiver model in CppSim.
- Use SPICE data (from Qualcomm) to build machine learning models.
- Compare current models with different techniques (esp. RNNs).
Build a receiver model with signal generator and channel in CppSim.

- **Signal generator** uses PRBS data (10G) including jitter.
- **Channel** is represented by a simple 3-pole low-pass filter. It also describes a transmission line, with the potential for loss, impedance mismatches, and reflections.
- **Receiver** has CTLE, CDR and DFE.
CDR behavior model is structured as 4 parts: phase detector, charge pump, loop filter, and VCO. In VCO, noise can be added and act as power supply induced jitter.

Sweep jitter frequency, compare the relative magnitude of the input and output jitter at the frequencies of interest. The jitter transfer function of CDR is shown on the right.
In vco_with_noise block, one possible way to inject PSIJ is to add random noise in the input of VCO.

VCO noise output:

\[
out = \sqrt{\frac{f_{\text{offset}}^2}{Kv^2} \times 10^{\frac{\text{noise}_{\text{at}}_{\text{offset}}}{10}} / T_s} \times N(0,1)
\]

Where \(N(0,1)\) is the Gaussian random noise, which the mean is 0 and the variance is 1. 
\(\text{noise}_{\text{at}}_{\text{offset}}\) is output noise of vco at offset frequency. 
\(Kv\) is determined by VCO implementation.
The number of data points is 8000. Different size of training data (400, 800, 2400, 4000, 5600) are used for training models. And use the last 2400 data as testing.

- With small size of training data set, linear model shows better performance.
- With large size of training data set, nonlinear model is better.
Compare model performance using different orders with different model types.

- If we have a small number of training data (400 sample points), some linear models (e.g. Output Error, ARMAX) and some nonlinear models (e.g. Hammerstein-Wiener) show better performance.

- If we have a large number of training data (4000 sample points), some nonlinear models (e.g. NNARX, NNARMAX) are much better than other models.

<table>
<thead>
<tr>
<th>Model</th>
<th>Order = 5</th>
<th>Order = 10</th>
<th>Order = 15</th>
<th>Order = 20</th>
<th>Order = 25</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output Error</td>
<td>81.73%</td>
<td>81.78%</td>
<td>81.70%</td>
<td>75.75%</td>
<td>72.62%</td>
</tr>
<tr>
<td>ARMAX</td>
<td>75.63%</td>
<td>80.16%</td>
<td>80.87%</td>
<td>80.93%</td>
<td>81.06%</td>
</tr>
<tr>
<td>State Space</td>
<td>76.99%</td>
<td>76.99%</td>
<td>76.99%</td>
<td>76.99%</td>
<td>76.99%</td>
</tr>
<tr>
<td>Hammerstein-Wiener</td>
<td>81.73%</td>
<td>80.16%</td>
<td>80.87%</td>
<td>80.93%</td>
<td>81.06%</td>
</tr>
<tr>
<td>Nonlinear ARX</td>
<td>63.81%</td>
<td>80.78%</td>
<td>81.49%</td>
<td>81.71%</td>
<td>81.73%</td>
</tr>
<tr>
<td>NNARX</td>
<td>81.35%</td>
<td>80.33%</td>
<td>80.18%</td>
<td>74.29%</td>
<td>72.76%</td>
</tr>
<tr>
<td>NNARMAX</td>
<td>-12.96%</td>
<td>-0.37%</td>
<td>69.55%</td>
<td>69.50%</td>
<td>74.49%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model</th>
<th>Order = 5</th>
<th>Order = 10</th>
<th>Order = 15</th>
<th>Order = 20</th>
<th>Order = 25</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output Error</td>
<td>68.85%</td>
<td>15.74%</td>
<td>68.97%</td>
<td>0.06%</td>
<td>63.53%</td>
</tr>
<tr>
<td>ARMAX</td>
<td>66.61%</td>
<td>68.69%</td>
<td>68.46%</td>
<td>68.92%</td>
<td>68.69%</td>
</tr>
<tr>
<td>State Space</td>
<td>69.22%</td>
<td>69.22%</td>
<td>69.22%</td>
<td>69.22%</td>
<td>69.22%</td>
</tr>
<tr>
<td>Hammerstein-Wiener</td>
<td>0.09%</td>
<td>0.87%</td>
<td>68.99%</td>
<td>0.28%</td>
<td>0.04%</td>
</tr>
<tr>
<td>Nonlinear ARX</td>
<td>66.94%</td>
<td>67.46%</td>
<td>67.59%</td>
<td>67.75%</td>
<td>67.99%</td>
</tr>
<tr>
<td>NNARX</td>
<td>83.76%</td>
<td>86.51%</td>
<td>85.63%</td>
<td>88.20%</td>
<td>88.93%</td>
</tr>
<tr>
<td>NNARMAX</td>
<td>76.74%</td>
<td>83.86%</td>
<td>84.29%</td>
<td>78.27%</td>
<td>83.82%</td>
</tr>
</tbody>
</table>
Summary

- Focus is on behavioral modeling of driver and receiver circuits
- Project consists of three parts:
  - Based on time domain data (Swaminathan, GT)
  - Based on X-Parameters (Schutt-Aine, UIUC)
  - Receiver Modeling (Franzon, NCSU)
- Time Domain
  - Applied to Qualcomm SI example (2 ports) – Good results
  - Extending to Qualcomm SI/PI example (4 ports) – Ongoing
  - PC based Jitter and BER estimation making excellent good progress
- X-Parameters
  - Applied to Buffer and IBIS modeling
- Receiver Modeling
  - Applied to PSIJ in VCO
  - Compared various ML methods
  - In discussions with Qualcomm for application to receiver circuits
- Overall good progress in achieving goals
Outline

Enter text here