IBIS-Compatible Macromodel and Interconnect Simulation Techniques

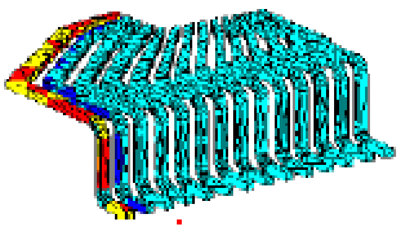
José E. Schutt-Ainé

University of Illinois
at
Urbana-Champaign

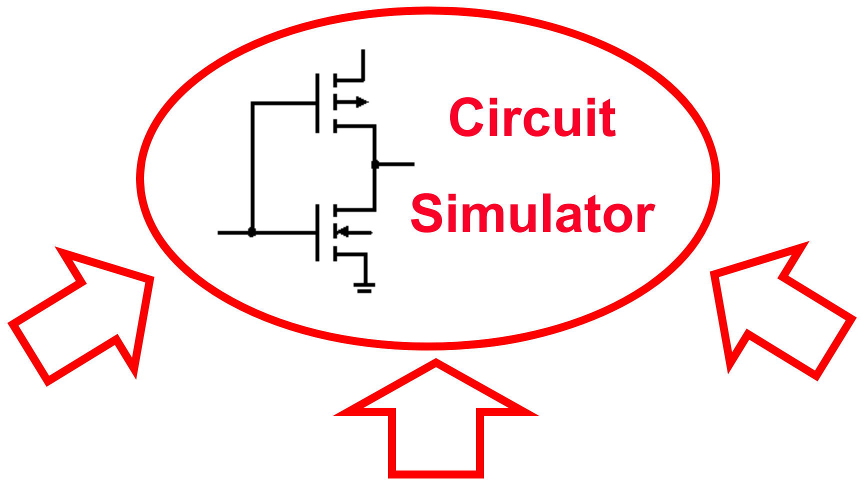
IBIS Summit
May 25, 2018
Brest France

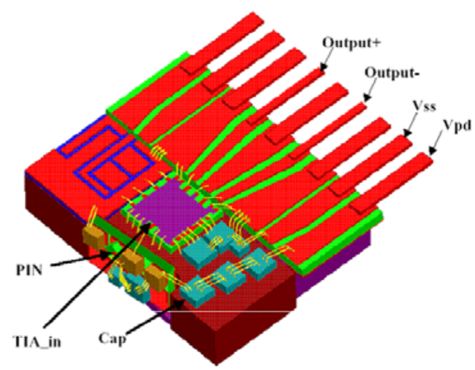
Interconnect Structures

Crosstalk
Couplings
Reflections
Losses
Dispersion

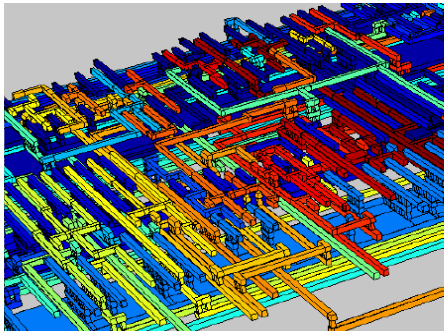


Packages





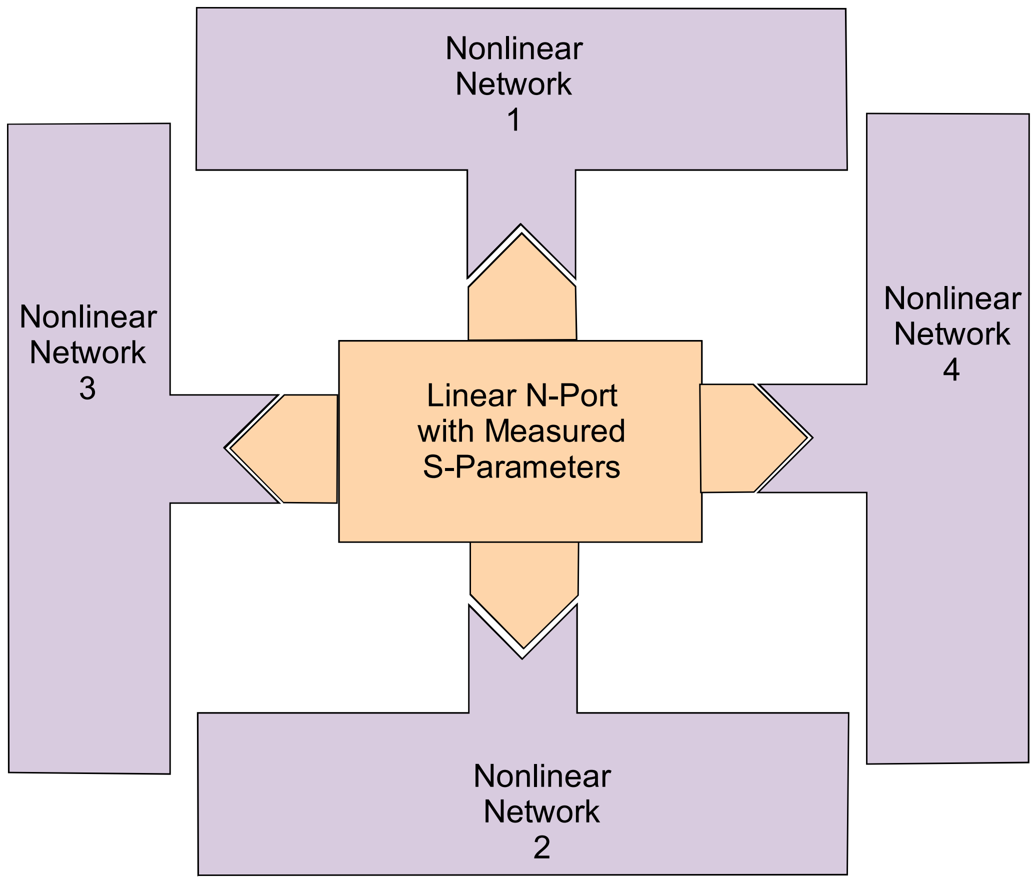
Ground Noise
Nonlinear effects
Radiation, EMI



Interconnects

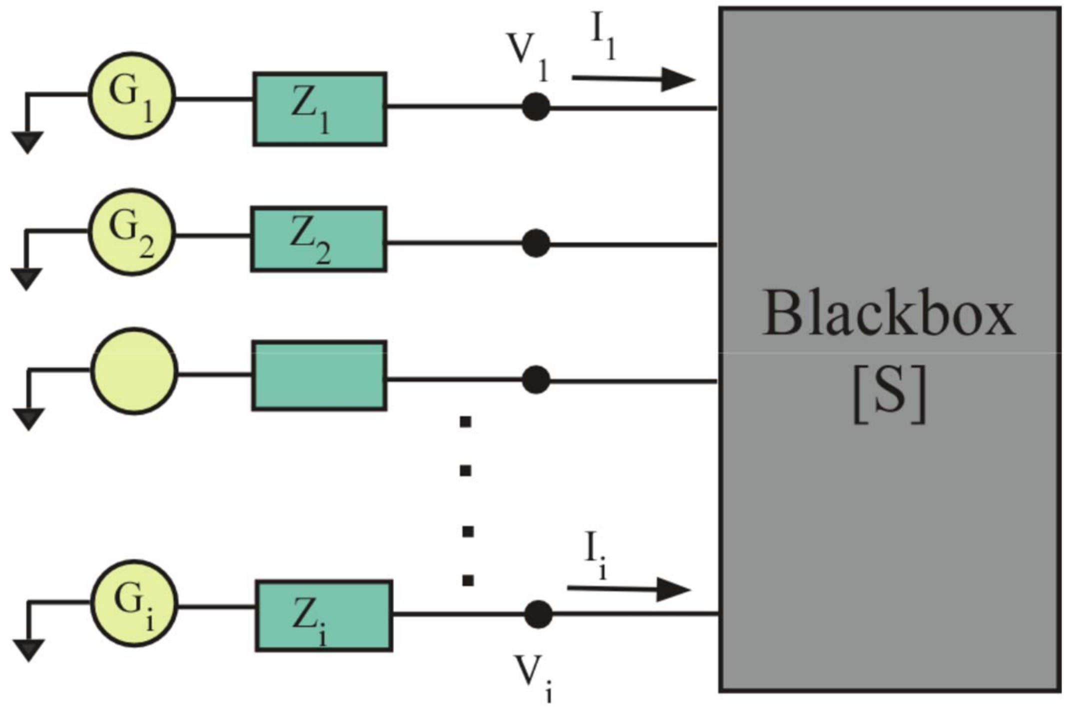
Courtesy of http://www.ansoft.com/hfworkshop03/Weimin_Sun_Vitesse.pdf

Blackbox Macromodeling



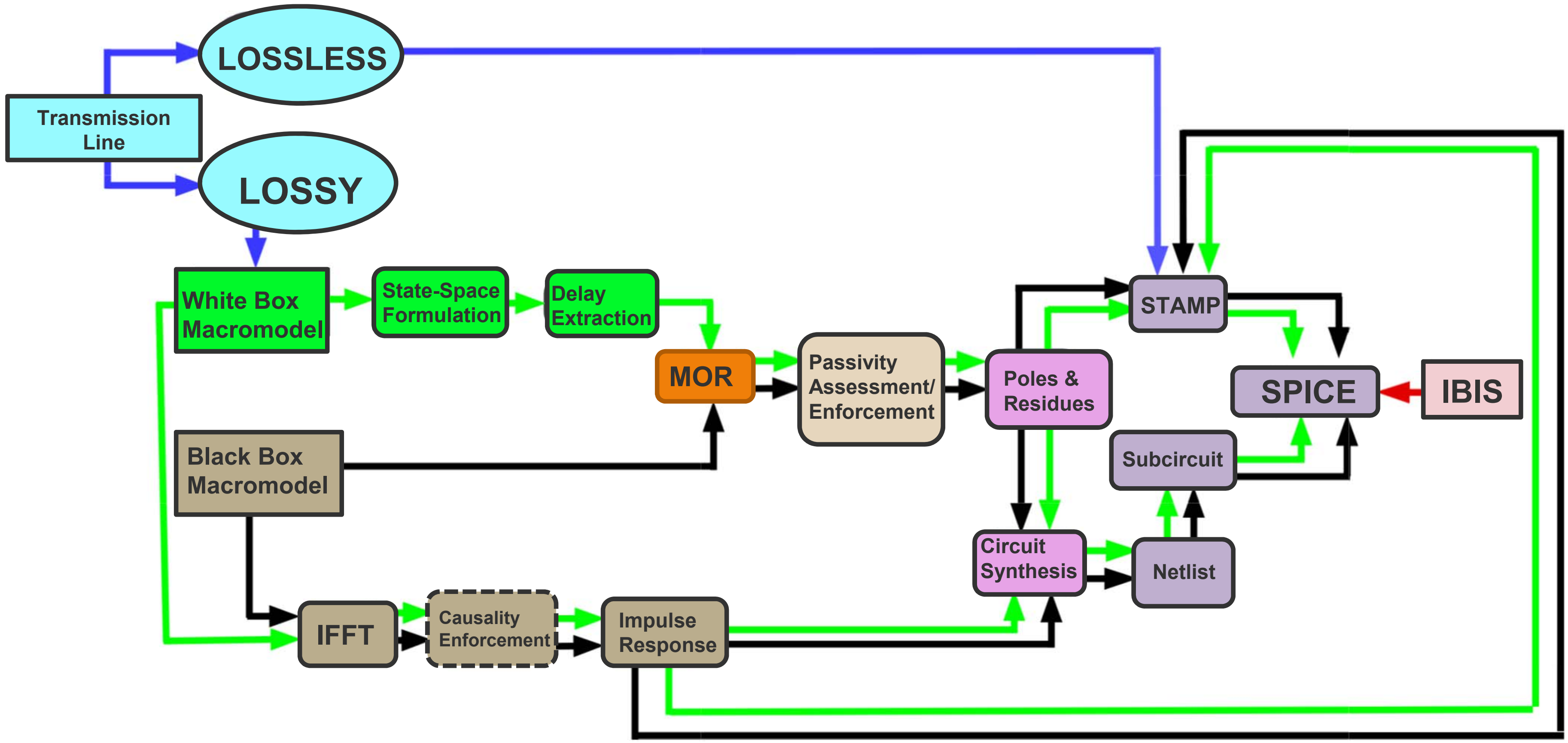
Objective:
Perform time-domain simulation of composite network to determine timing waveforms, noise response or eye diagrams

Advantages of Macromodels



- Protect intellectual property (IP)
- Reduce complexity (fewer poles, fewer ports)
- Capture frequency dependence

Interconnect/Macromodel Modeling Strategies



Model Order Reduction

Objective: Approximate frequency-domain transfer function to take the form:

$$H(\omega) = \left[A_1 + \sum_{i=1}^{L} \frac{a_{1i}}{1 + j\omega / \omega_{c1i}} \right]$$

Methods

- AWE – Pade
- Pade via Lanczos (Krylov methods)
- Rational Function
- Chebyshev-Rational function
- Vector Fitting Method

Model Order Reduction (MOR)

Question: Why use a rational function approximation?

Answer: because the frequency-domain relation

$$Y(\omega) = H(\omega)X(\omega) = \left[d + \sum_{k=1}^{L} \frac{c_k}{1 + j\omega / \omega_{ck}} \right] X(\omega)$$

will lead to a time-domain *recursive* convolution:

$$y(t) = dx(t-T) + \sum_{k=1}^{L} y_{pk}(t)$$

where

$$y_{pk}(t) = a_k x(t-T)\left(1 - e^{-\omega_{ck}T}\right) + e^{-\omega_{ck}T} y_{pk}(t-T)$$

which is very fast!

State-Space Representation

The State space representation of the transfer function is given by

$$\dot{x}(t) = Ax(t) + Bu(t)$$

$$y(t) = Cx(t) + Du(t)$$

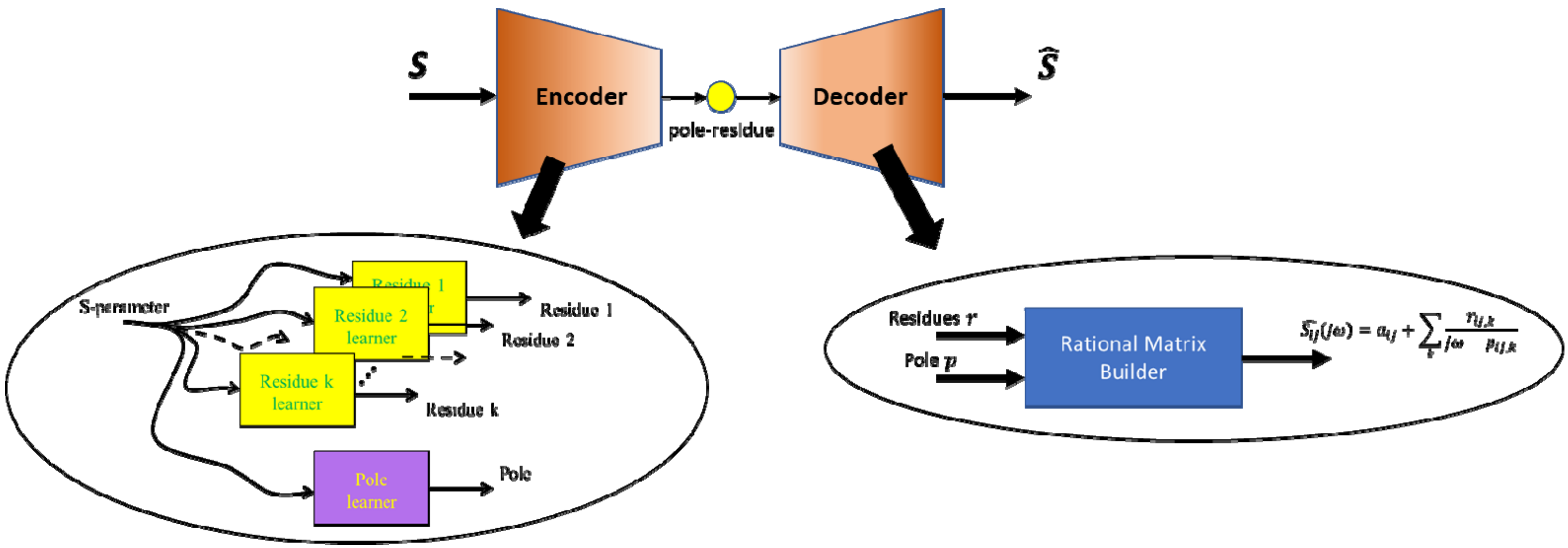
The transfer function is given by

$$S(s) = C\left(sI - A\right)^{-1} B + D$$

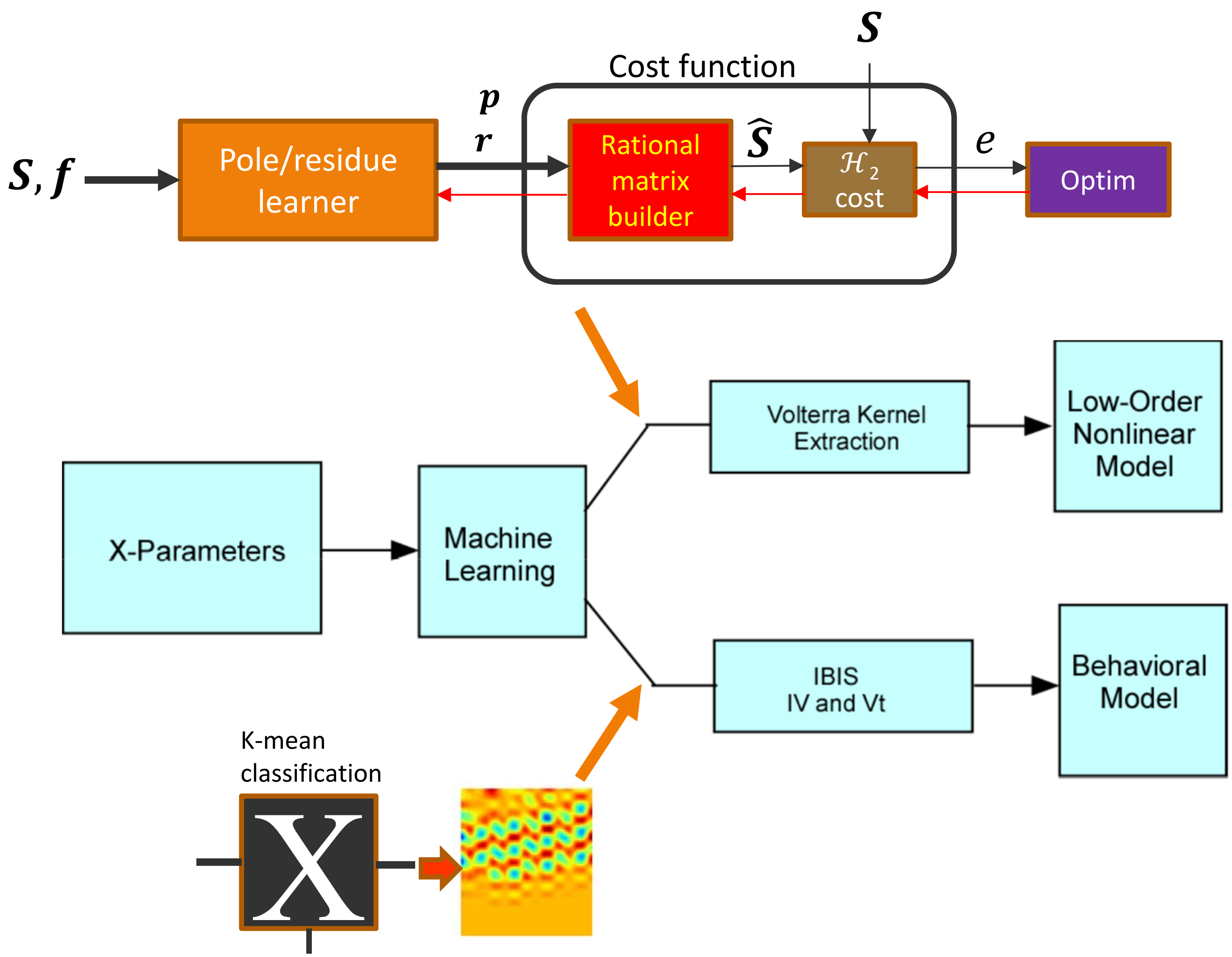
A, B, C and *D* are constructed from poles and residues

Looking Forward
Machine Learning for Macromodeling

- S-parameter feed forward neural network (SFNN) works like an **auto encoder**
  - No need to prepare pole/residue for training. It trains on the input it sees itself.
  - This is the beginning for GAN (Generative Adversarial Networks) for stochastic modeling.

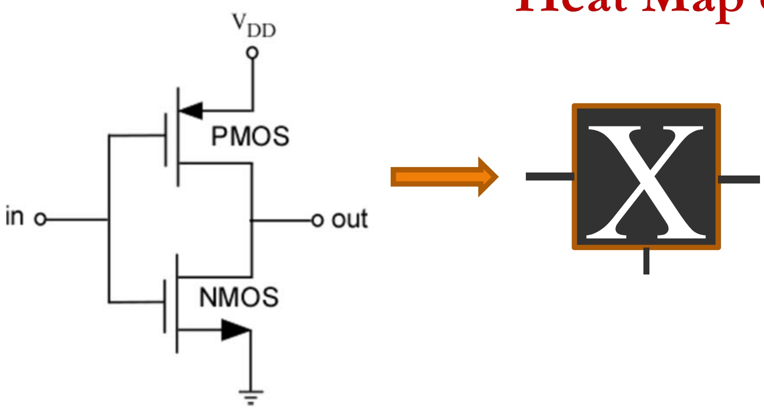


Looking Forward
Machine Learning for Buffer Modeling



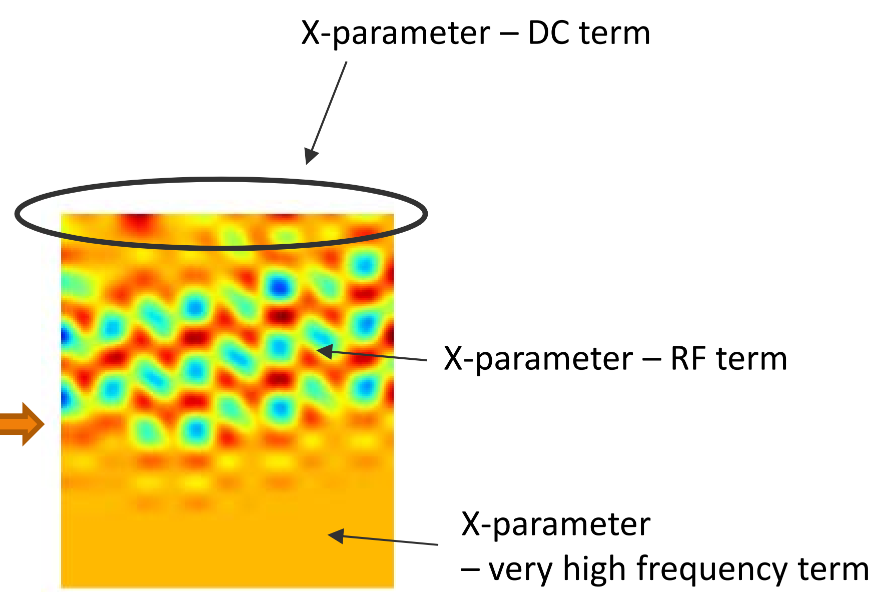
Looking Forward
Machine Learning for Buffer Modeling

Heat Map of X-Parameters



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

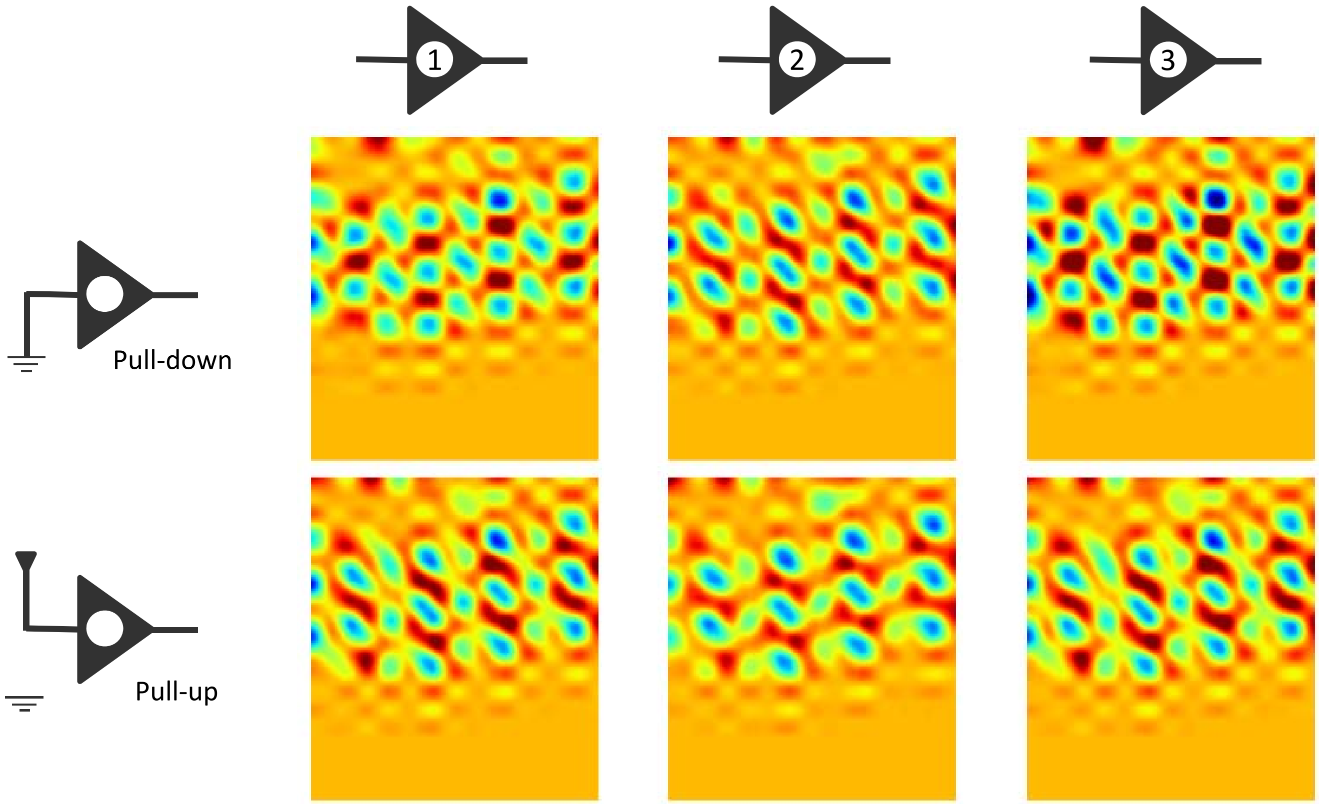
```
% AN_1_1(real)  FI_31(real)  FB_1_1(complex)  FB_1_2(complex)  FB_1_3(complex)
% FB_2_1(complex)  FB_2_2(complex)  FB_2_3(complex)  S_1_1_1_1(complex)  T_1_1_1_1(complex)
% S_1_2_1_1(complex)  T_1_2_1_1(complex)  S_1_3_1_1(complex)  T_1_3_1_1(complex)  S_2_1_1_1(complex)
% T_2_1_1_1(complex)  S_2_2_1_1(complex)  T_2_2_1_1(complex)  S_2_3_1_1(complex)  T_2_3_1_1(complex)
% XY_3_1_1(complex)  S_1_1_1_2(complex)  T_1_1_1_2(complex)  S_1_2_1_2(complex)  T_1_2_1_2(complex)
% S_1_3_1_2(complex)  T_1_3_1_2(complex)  S_2_1_1_2(complex)  T_2_1_1_2(complex)  S_2_2_1_2(complex)
% T_2_2_1_2(complex)  S_2_3_1_2(complex)  T_2_3_1_2(complex)  XY_3_1_2(complex)  S_1_1_1_3(complex)
% T_1_1_1_3(complex)  S_1_2_1_3(complex)  T_1_2_1_3(complex)  S_1_3_1_3(complex)  T_1_3_1_3(complex)
% S_2_1_1_3(complex)  T_2_1_1_3(complex)  S_2_2_1_3(complex)  T_2_2_1_3(complex)  S_2_3_1_3(complex)
% T_2_3_1_3(complex)  XY_3_1_3(complex)  S_1_1_2_1(complex)  T_1_1_2_1(complex)  S_1_2_2_1(complex)
% T_1_2_2_1(complex)  S_1_3_2_1(complex)  T_1_3_2_1(complex)  S_2_1_2_1(complex)  T_2_1_2_1(complex)
% S_2_2_2_1(complex)  T_2_2_2_1(complex)  S_2_3_2_1(complex)  T_2_3_2_1(complex)  XY_3_2_1(complex)
% S_1_1_2_2(complex)  T_1_1_2_2(complex)  S_1_2_2_2(complex)  T_1_2_2_2(complex)  S_1_3_2_2(complex)
% T_1_3_2_2(complex)  S_2_1_2_2(complex)  T_2_1_2_2(complex)  S_2_2_2_2(complex)  T_2_2_2_2(complex)
% S_2_3_2_2(complex)  T_2_3_2_2(complex)  XY_3_2_2(complex)  S_1_1_2_3(complex)  T_1_1_2_3(complex)
% S_1_2_2_3(complex)  T_1_2_2_3(complex)  S_1_3_2_3(complex)  T_1_3_2_3(complex)  S_2_1_2_3(complex)
% T_2_1_2_3(complex)  S_2_2_2_3(complex)  T_2_2_2_3(complex)  S_2_3_2_3(complex)  T_2_3_2_3(complex)
```



Looking Forward
Machine Learning for Buffer Modeling

X-parameter for IBIS generation

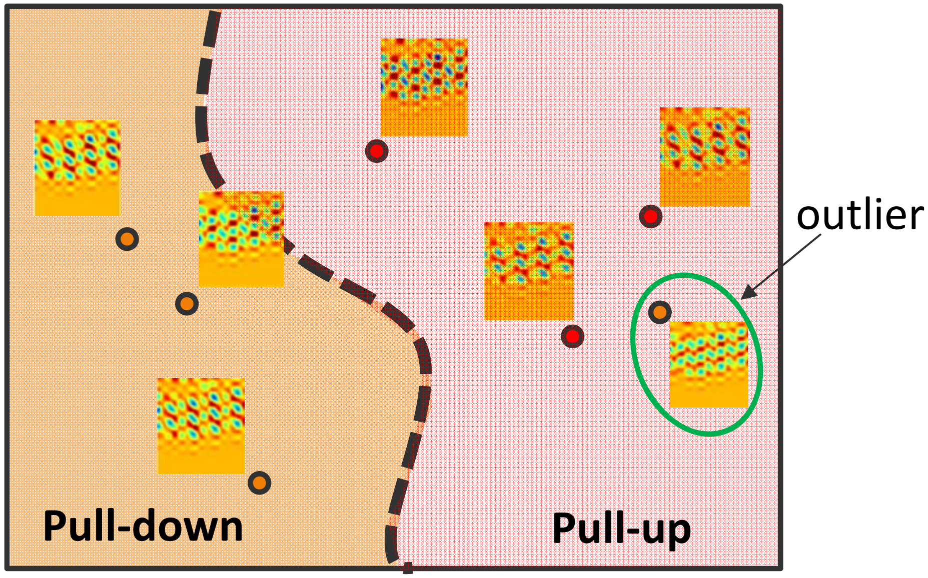
- X-parameter as in the neural network's eye



Looking Forward
Machine Learning for Buffer Modeling

X-parameter for IBIS generation

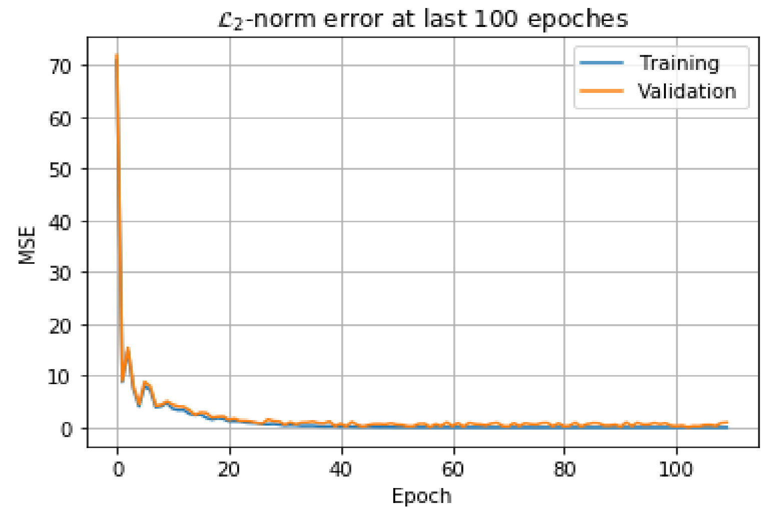
- **X parameter for static IBIS curve is generated.**
- **An unsupervised classification algorithm (as simple as K-mean) is used to verify that pull-up and pull-down data is separable.**
- **Watch out for potential outliers: recollect data, unusual dynamic nature between pull-up and pull-down.**
- **Then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

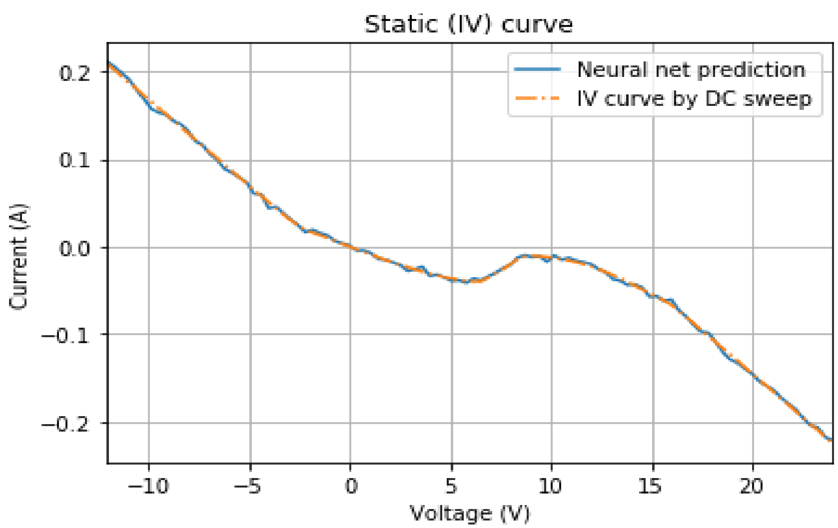


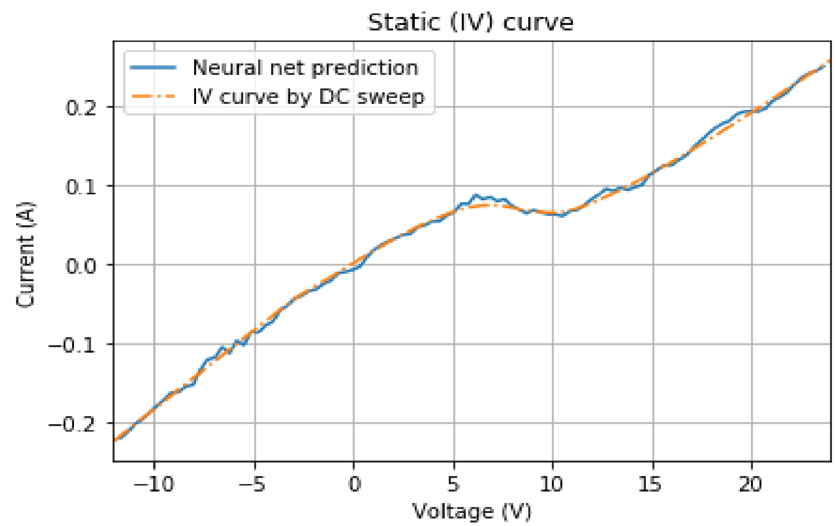
X-parameter in separable hyperspace

Looking Forward
Machine Learning for Buffer Modeling

- 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.







L2-norm error: ~ e-4

L2-norm error: ~ e-4

Looking Forward

- Machine learning approach to extract poles and residues
- X-parameters for IBIS models
- Machine learning and X parameters for IBIS
- Volterra series expansion of X parameters