Using IBIS-AMI in COM Analysis

DesignCon IBIS Summit
Santa Clara, California
February 2nd, 2018

Wei-hsing Huang, SPISim
Wei-hsing.Huang@spisim.com

Agenda:

- Motivation
- Background
- Using AMI in COM Flow
- Results
- Summary
- Q & A

Motivation

- **AMI model development :**
  - Model is not an executable, it needs driver
  - Spawn child (simulation) processes is tricky to debug
  - Optimization/flow is beyond model developer's control

- **Open source link-analysis platforms**
  - Includes useful building blocks (e.g. Figure of Merits, BER)
  - Mostly use generic Tx/Rx EQ blocks/algorithms
  - Can be adapted to use IBIS-AMI models
  - Can shorten AMI modeling design cycle
  - E.g. COM (1), (2) & PyBERT (3)

Background 1/3

- COM (Channel operating Margin)
  - Is a IEEE 802.3bj Spec (Annex 93A)
  - Published codes, well documented and maintained
  - Is a simplified version of BER analysis
  - Figure of merit based channel optimization and analysis
  - Jitter, Noise etc are also included

$$COM = 20\log_{10}(A_s/A_{ni})$$

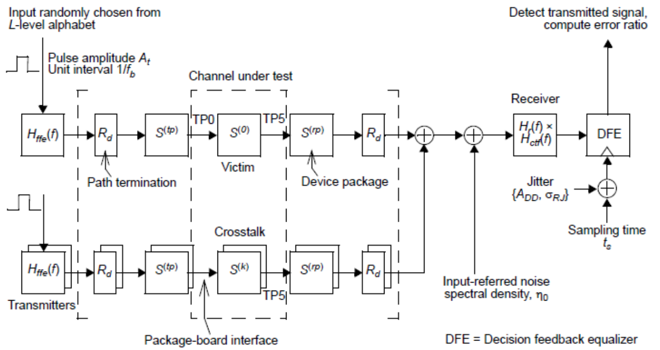
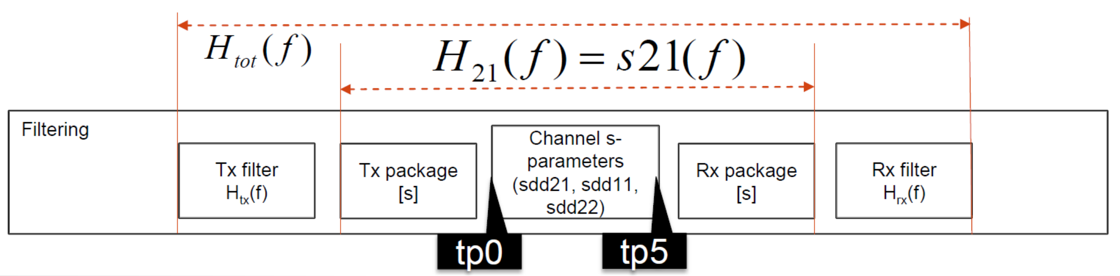


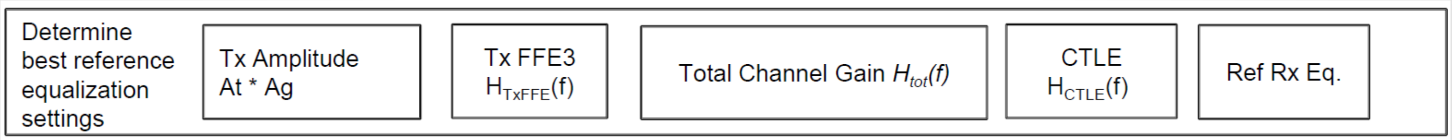
Figure 93A–1—COM reference model

Background 2/3

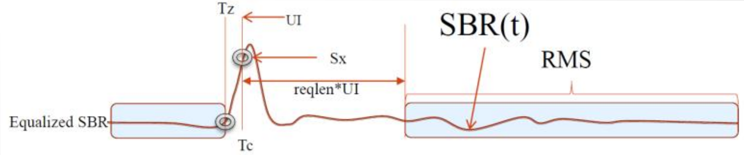
- COM has channel components and conditioning algorithms



- Use FOM to find FFE, CTLE settings, then apply DFE for BER



- Single-bit-response based



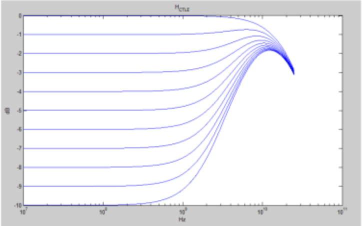
Background 3/3

- COM use exhaustic search for FFE + CTLE [4]
  - Generic implementations
  - CTLE is gdc only
  - DFE is not optimized together

CTLE

One degree of freedom: G_{DC}

$$H_{CTLE}(f) = f_b \frac{j \cdot f + 0.25 \cdot f_b 10^{\frac{G_{DC}}{20}}}{(j \cdot f + 0.25 \cdot f_b) \cdot (j \cdot f + f_b)}$$

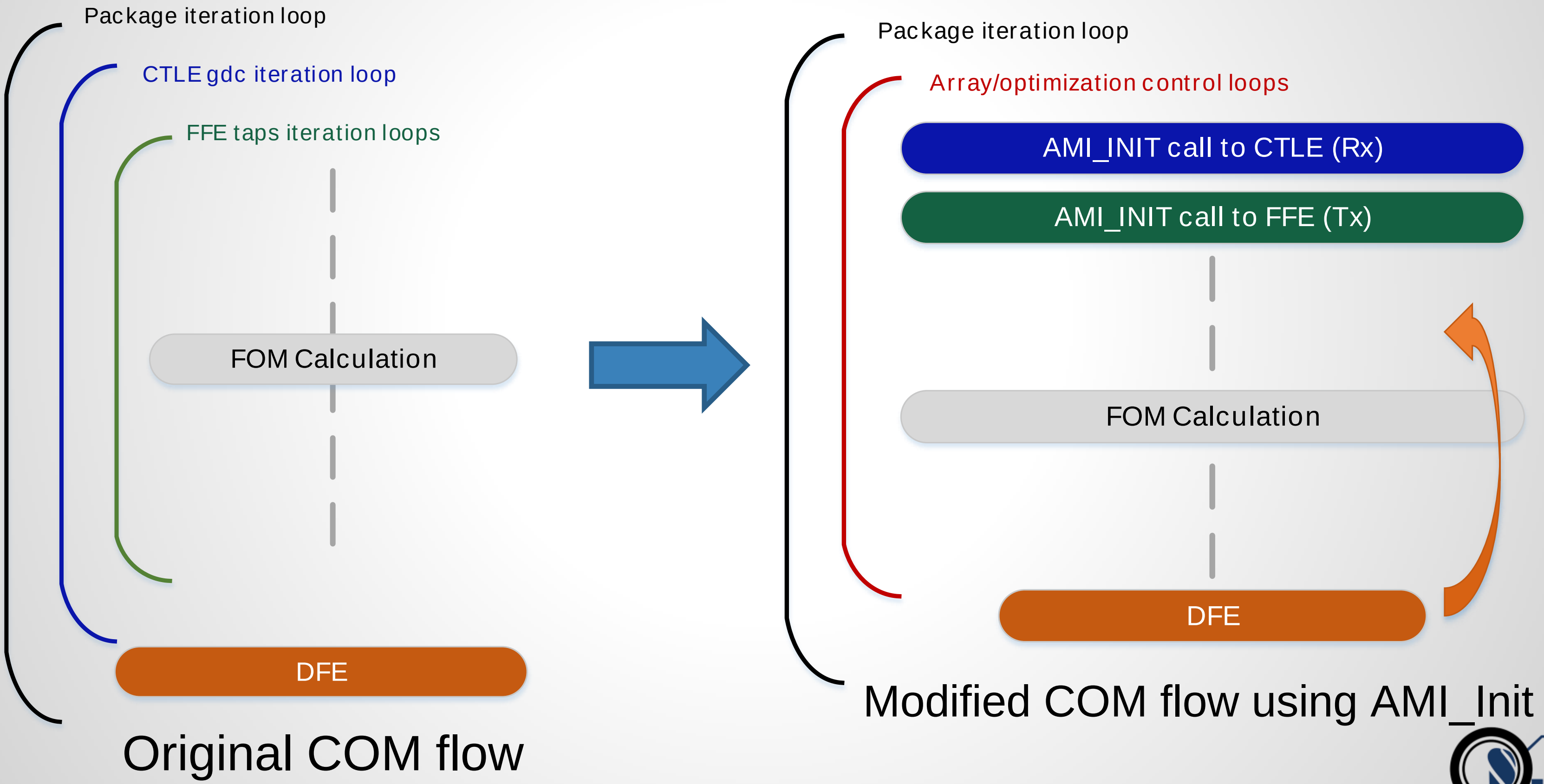


G_{DC} is DC gain in dB

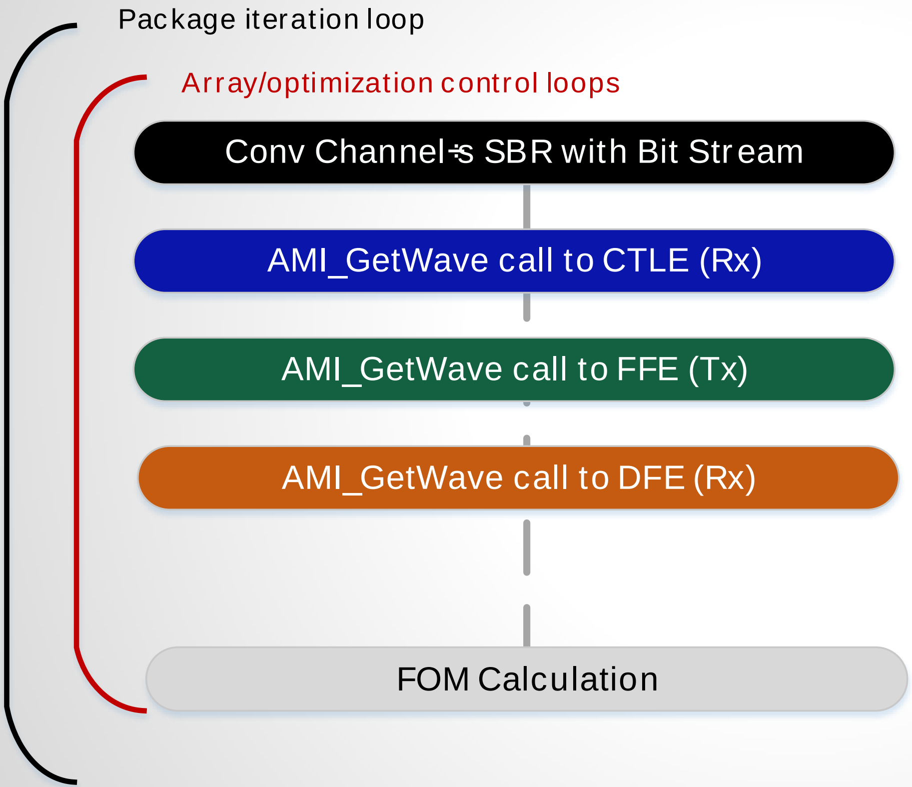
```
for ctle_index=1:length(gdc_values)
    g_dc = gdc_values(ctle_index);
    kacdc = 10^(g_dc/20);
    CTLE_fp1 = param.CTLE_fp1(ctle_index);
    CTLE_fp2 = param.CTLE_fp2(ctle_index);
    CTLE_fz = param.CTLE_fz(ctle_index);
    switch param.CTLE_type
```

```
for k_cm2=1:length(cm2_values)
    cm2=cm2_values(k_cm2);
    for k_cm1=1:length(cm1_values)
        cm1=cm1_values(k_cm1);
        for k_cp1=1:length(cp1_values)
            cp1=cp1_values(k_cp1);
            for k_cp2=1:length(cp2_values)
                cp2=cp2_values(k_cp2);
                for k_cp3=1:length(cp3_values)
                    cp3=cp3_values(k_cp3);
                    pxi=pxi+1;
                    progress = pxi/(length(cm2_values)* length(cm1_values)*length(cp1_values)*length(gdc_values)*length(cp2_values)*length(cp3_values)*lf_indx )
                    txffe = [cm2, cm1, 1-abs(cm2)-abs(cm1)-abs(cp1)-abs(cp2)-abs(cp3), cp1, cp2, cp3]
                    % Skip combinations with small values of c(0), not guaranteed to be supported by all transmitters.
                    if txffe(3)<param.tx_ffe_c0_min
                        continue;
                    end
```

Use AMI models in COM 1/2 (5)



Use AMI model in COM 2/2



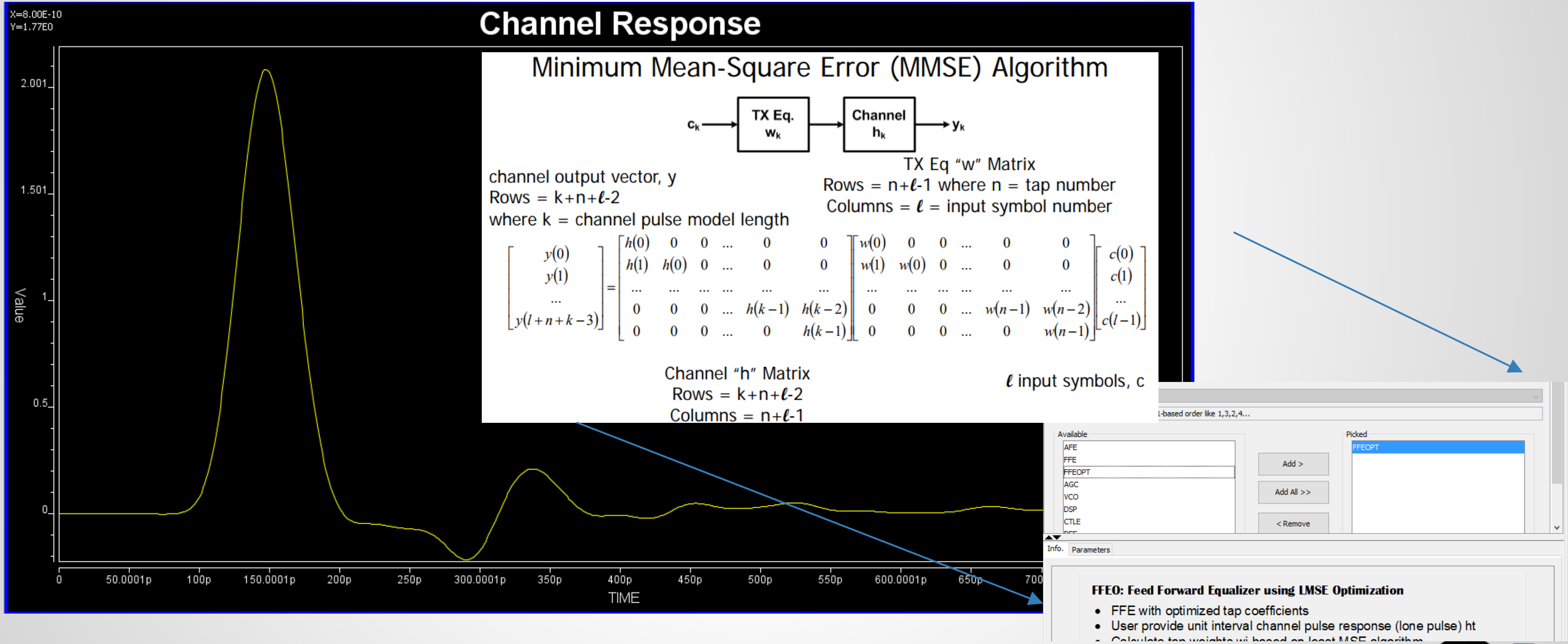
Modified COM flow using AMI_GetWave (Bit-by-bit)

- Use loadlibrary mechanism
- AMI parameters can be pre-assembled
- Example library loading/calling in COM

```
mex -setup
load('SPISimAMI_WIN64.dll', 'ami.h')
libisloaded('SPISimAMI_WIN64')
libfunctions('SPISimAMI_WIN64')
calllib('SPISimAMI_WIN64','ami_init', htInput, rowSize, numAggr...)
unloadlibrary('SPISimAMI_WIN64')
```

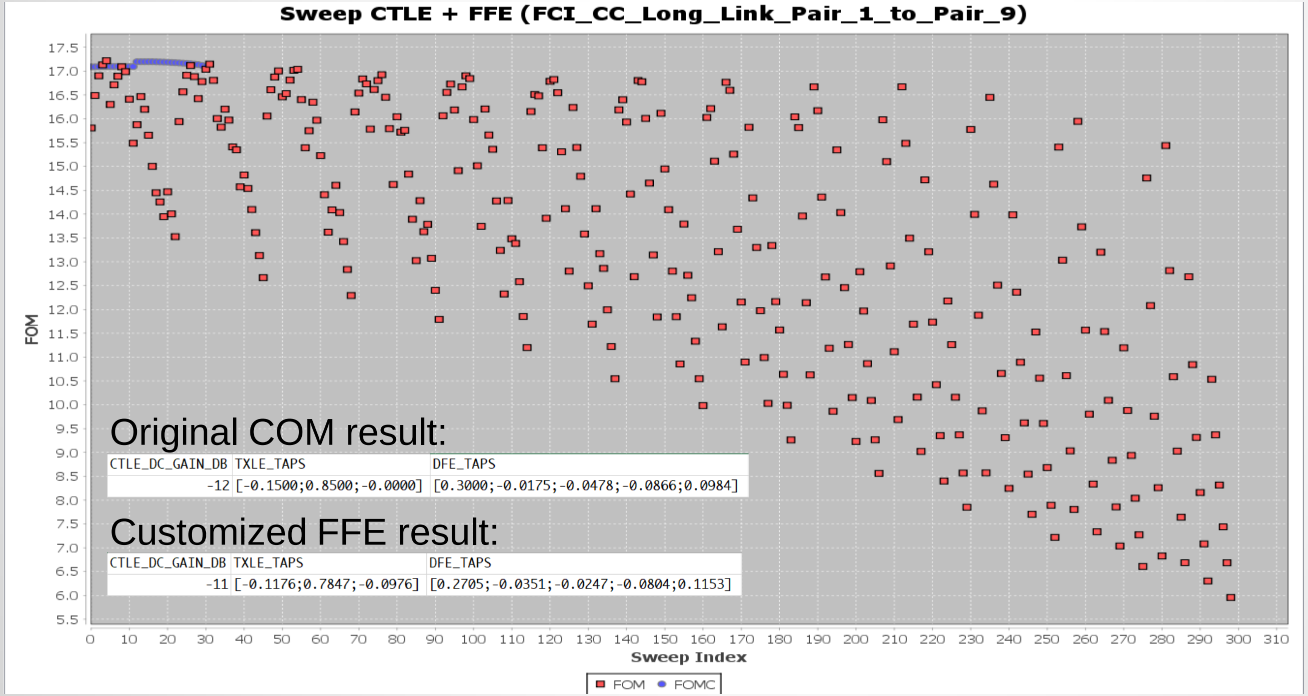
Example Results 1 [(6), (7)]

- Replace COM's FFE with self-optimization FFE



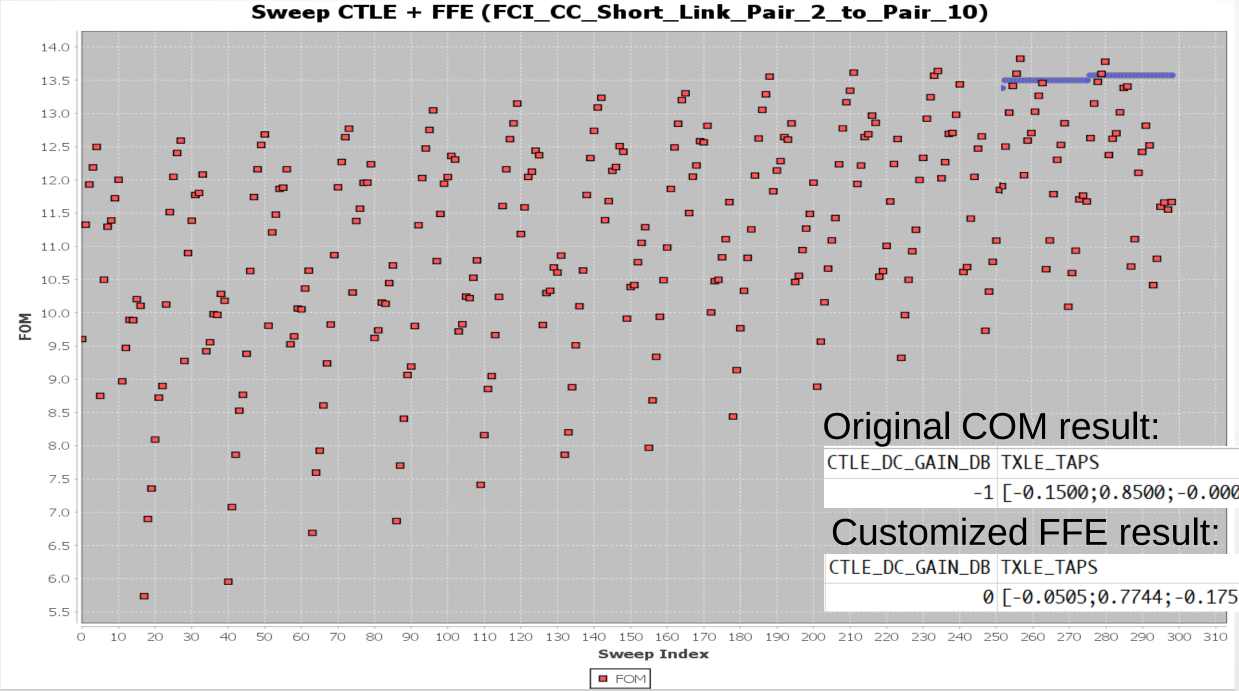
Example Results 1

- 13 gdc * 24 FFE sweep **(red)** vs customized FFE **(blue)**



Example Results 2

- 13 gdc * 24 FFE sweep **(red)** vs customized FFE **(blue)**



Original COM result:

CTLE_DC_GAIN_DB	TXLE_TAPS	DFE_TAPS
-1	[-0.1500;0.8500;-0.0000]	[0.3000;0.1017;0.0660;0.0171;0.0480]

Customized FFE result:

CTLE_DC_GAIN_DB	TXLE_TAPS	DFE_TAPS
0	[-0.0505;0.7744;-0.1751]	[0.3000;0.0591;0.0395;0.0322;0.0224]

Summary:

- AMI model can be used in COM analysis:
  - COM is a great open platform for link analysis/AMI development
  - Replaces multi-level CTLE and FFE loop with AMI call
  - Can pull-in DFE for co-optimization

- Considerations:
  - Original COM flow supports AMI_Init type LTI only
    - AMI_GetWave based flow needs SBR $\otimes$ BitStream first
  - AMI parser is not necessarily needed
    - Parameters can be pre-assembed as strings
  - Can be used for back-channel analysis development

References:

1. **IEEE Std 802.3bj-2014, Specification**, Annex 93A
2. **Channel Operating Margin (COM)**, Richard Mellitz, DesignCon 2013
3. **PyBERT**: https://pypi.python.org/pypi/PyBERT
4. **COM tools**: http://www.ieee802.org/3/bj/public/tools.html
5. **IBIS V6.1 Spec. Section 10** http://ibis.org/ver6.1/
6. **New SI Techniques for Large System Performance Tuning**, Donald Telian, DesignCon 2016
7. **Sam Palermo, ECEN 720, High-Speed Link Circuits & Systems, Texas A&M**

Q & A

EDA Expertise in Signal, Power Integrity & Simulation