

# Automated AMI Model Generation & Validation



**José Luis Pino**  
**Amolak Badesha**

**Manuel Luschas**  
**Antonis Orphanou**  
**Halil Civit**

**Agilent EEsof EDA**



**Asian IBIS Summit, Shenzhen China November 9, 2010**  
**(Presented previously at the IBIS Summit on June 15, 2010)**

# Agenda

- AMI Model Generation Barriers
- Automated AMI Model-generation flow
  - Example-1: 6.25 Gb/s
  - Example-2: 10.3125 Gb/s
- TX Model Correlation Study
  - with Transistor Simulations
  - with Measurements
- Benefits of Automated AMI flow

# #1 AMI modeling barrier

## *Model Generation Time*



AMI Modeling suppose to Speed-up System Design Cycle,  
BUT, Model-generation takes Significant Time & Resources



....System Vendors have to wait a LONG  
time before accurate AMI models become  
available

Note: Vendors with **NO** experience in AMI modeling are spending 6-12+ months to come up with first-generation models

Models come very late in Design Cycle → used only for Validation, NOT Design

# Why AMI-model generation takes so long?

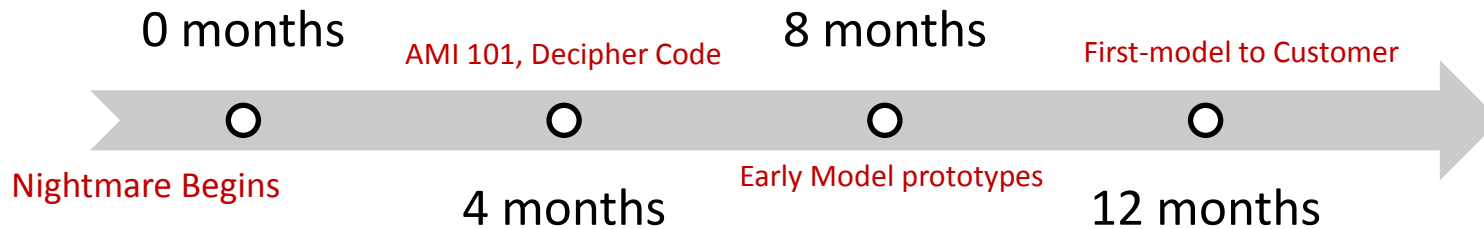


Typical Signal Integrity Engineers are NOT programmers

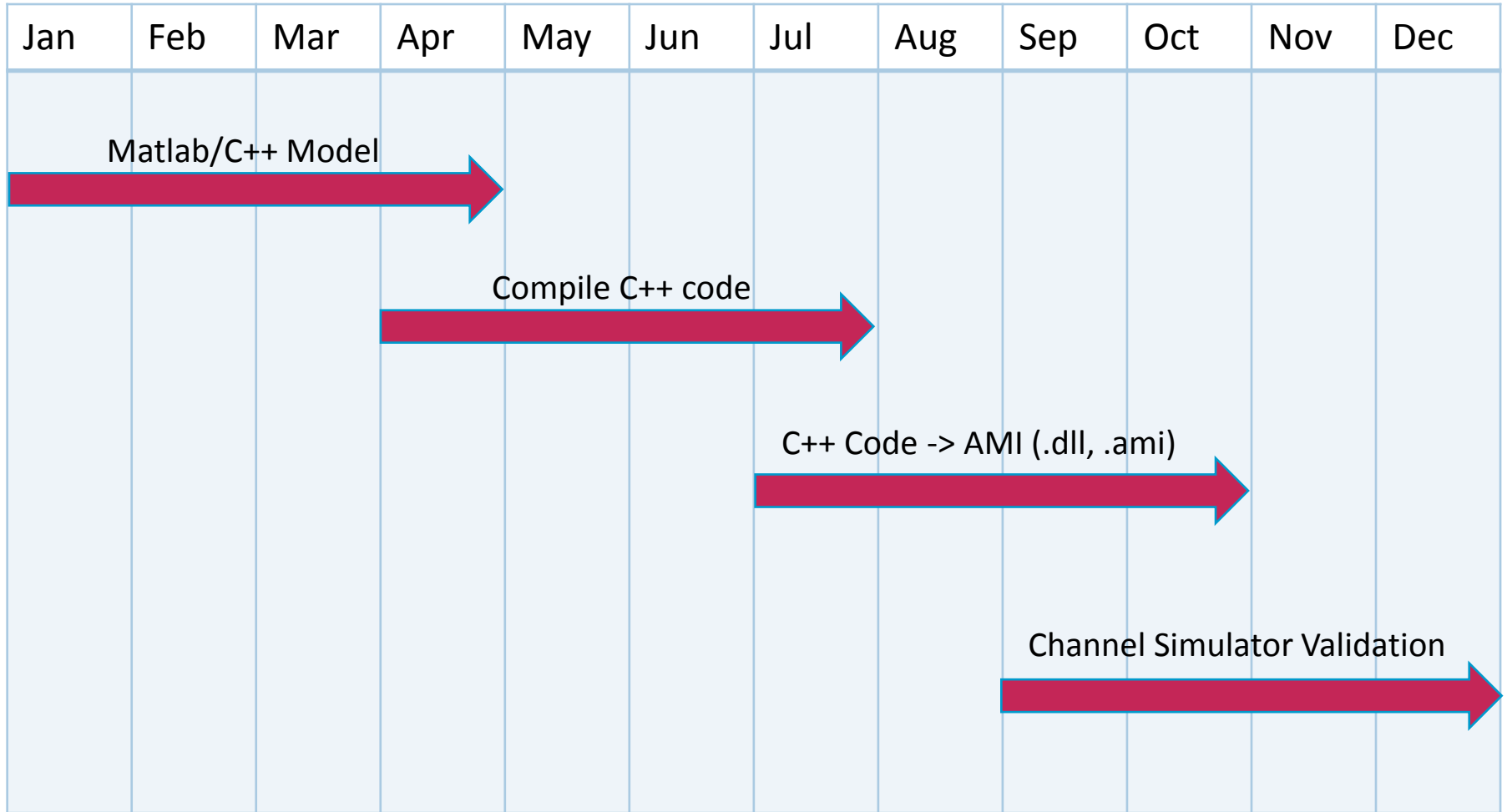


...they are having “Nightmares” in trying to develop AMI models

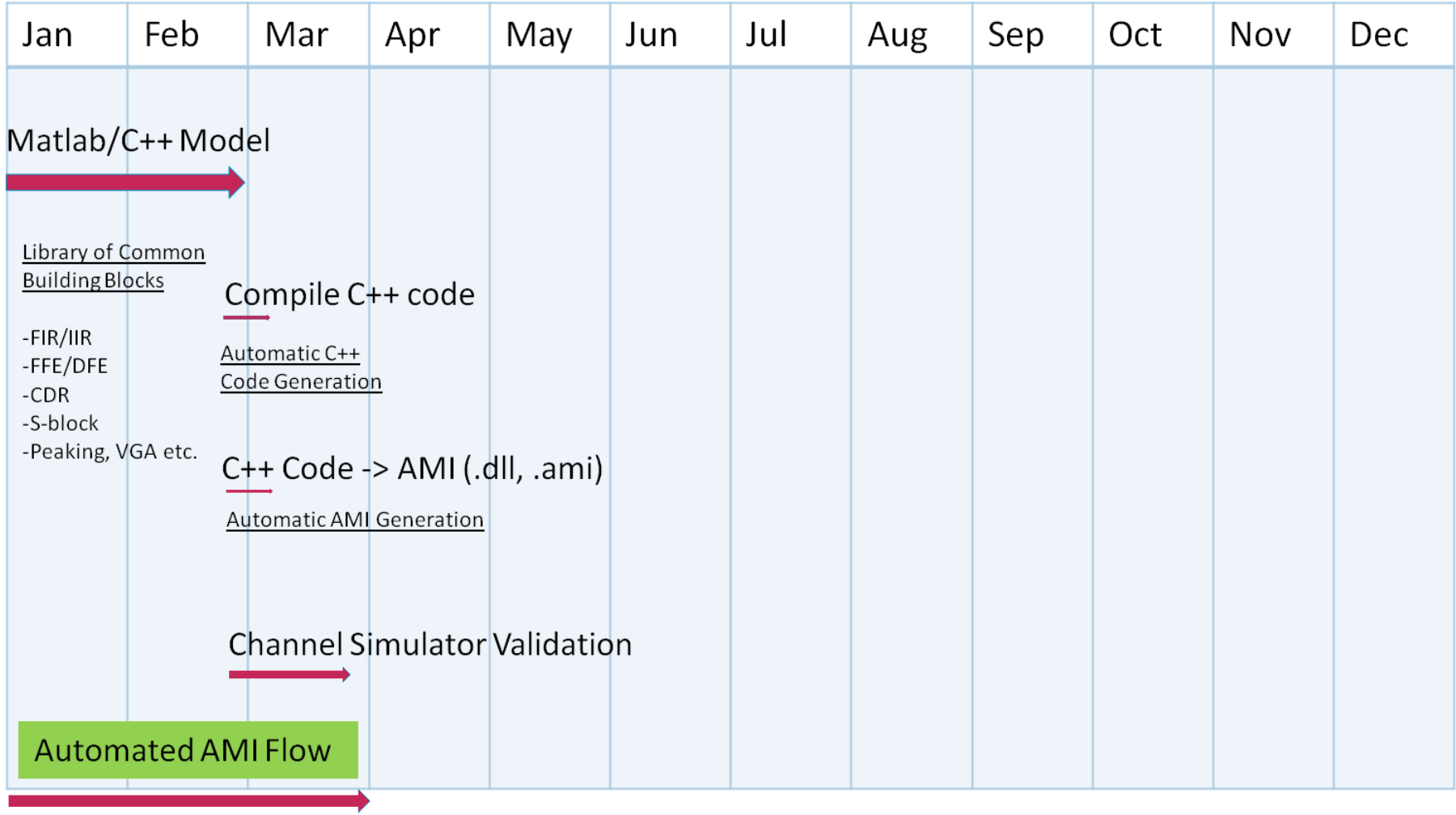
- Cryptic Matlab/C++ code passed from System-Architectures → AMI Modeler (if lucky)
- Challenge to Convert Algorithm design Code → AMI format



# Typical AMI model generation flow...



# Automated AMI model generation flow...



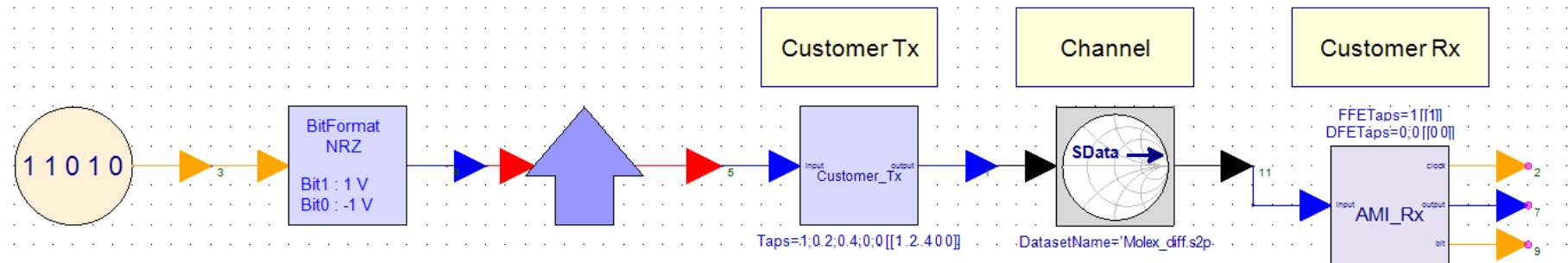
# ESL flow for Automated AMI Modeling

**Electronic System Level (ESL)** design and verification is an emerging electronic design methodology that focuses on the higher abstraction level concerns first and foremost.

ESL flow facilitates utilization of appropriate abstractions in order to increase comprehension about a system, and to enhance the probability of a successful implementation of functionality in a cost-effective manner

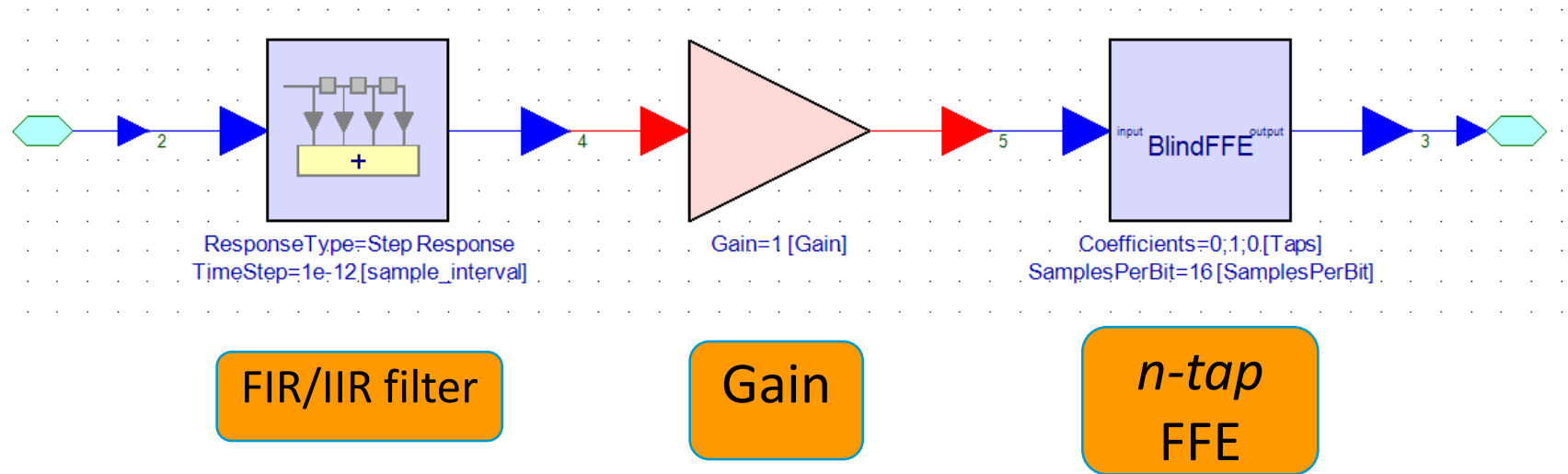


Here is an Example of SerDes modeling using ESL flow-



# ESL flow: TX Modeling Example (1)

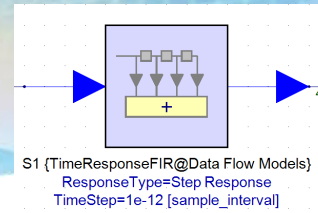
## Step-1: Starting Architecture Design with Generic Model



Different blocks represent high-level TX architecture

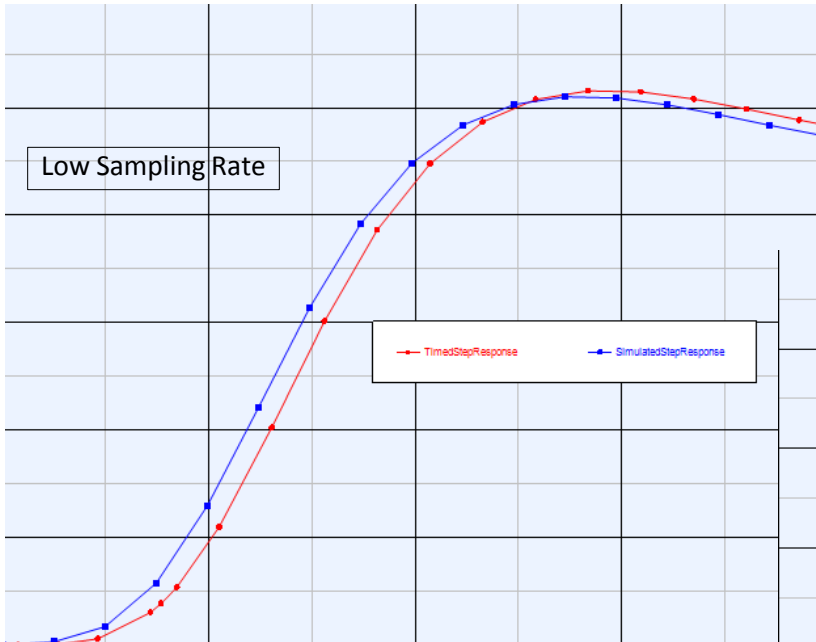
# More on FIR Filter...

How to bring in Spice or Measured data?

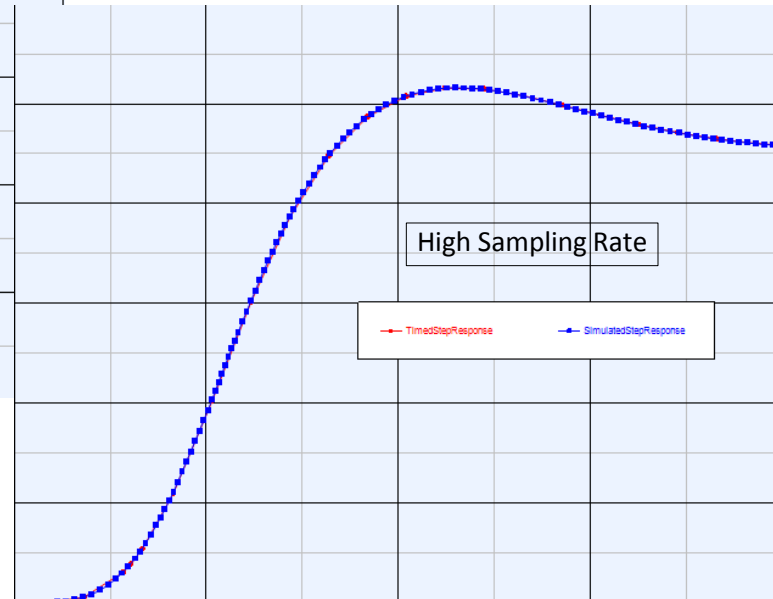


## Challenges:

1. Typical Simulation and Measured Data is not equally time-stepped



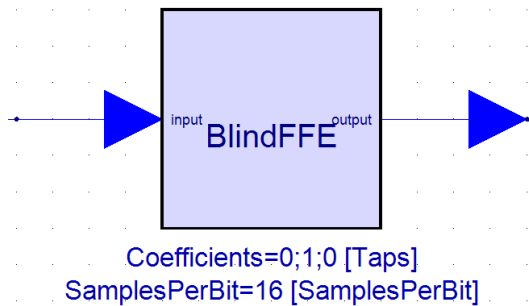
Sampling Rate determines Simulation Accuracy



FIR model should support "Arbitrary" Sampling Rate

# ESL flow: TX Modeling Example (2)

## Step-2: Customize IP -> Bring in Math Lang or C++ Code



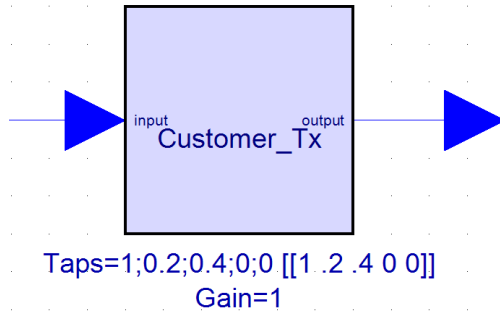
The screenshot shows the Agilent Model Composer interface for a 'Blind Feed-Forward Equalizer' model. The Designator is 'B2' and the Description is 'Blind Feed-Forward Equalizer'. The Model is set to 'MathLang@Data Flow Models'. The 'Equations' tab is active, showing the following code:

```
1 persistent dSamples;  
2 persistent numSamples;  
3 persistent taps;  
4  
5 if isempty(dSamples)  
6     % first time we hit this routine  
7     numSamples = length(Coefficients) * SamplesPerBit;  
8     dSamples = zeros(1, numSamples);  
9     dSamples(1) = input;  
10    taps = Coefficients';  
11 else  
12     dSamples = [input, dSamples(1:numSamples-1)];  
13 end  
14  
15 output = dSamples(1:SamplesPerBit:numSamples) * taps;
```

Fine-tune and Customize models with Math Lang and/or C++ code

# ESL flow: TX Modeling Example (3)

## Step-3: One-click AMI Code-Generation



Define Reserved and Model Specific Parameters -> Automatically configure appropriate AMI wrapper

The 'Shell Configuration' dialog box is shown. It has a 'Shell Type' dropdown set to 'IBIS Algorithmic Modeling Interface' and an 'AMI Model' dropdown set to 'customer\_tx'. Below these are three tabs: 'AMI Configuration', 'AMI Reserved Parameters', and 'AMI Model Specific Parameters'. The 'AMI Model Specific Parameters' tab is active and contains several sections: 'Model Type' with radio buttons for 'LTI' (selected) and 'NLTV'; 'Serdes Tx/Rx' with radio buttons for 'Tx' (selected) and 'Rx'; 'AMI\_Init Arguments' with dropdowns for 'Impulse Matrix', 'Sample Interval', and 'Bit Time'; and 'Output Port Mapping' with dropdowns for 'Waveform' (set to 'output') and 'Clock' (set to 'output'). At the bottom, there is a 'Generate Now' button with a gear icon, and 'OK', 'Cancel', and 'Help' buttons.

One-click AMI Code-generation

# ESL flow: TX Modeling Example (4)

## Step-4: Automatically Generated .ami and Visual-Studio project

The screenshot displays the Visual Studio 2008 Express Edition interface. The main window shows the source code for `sv_ami_tx.ami` and `sv_ami_tx.h`. The code defines a model with reserved parameters and model-specific coefficients. The Solution Explorer on the left shows the project structure, including header files, IBIS-AMI files, source files, and XML files. A context menu is open over the 'Build Solution' option in the Solution Explorer, listing options such as 'Rebuild Solution', 'Clean Solution', 'Batch Build...', 'Configuration Manager...', 'Add', 'Set StartUp Projects...', 'Paste', 'Rename', and 'Properties'.

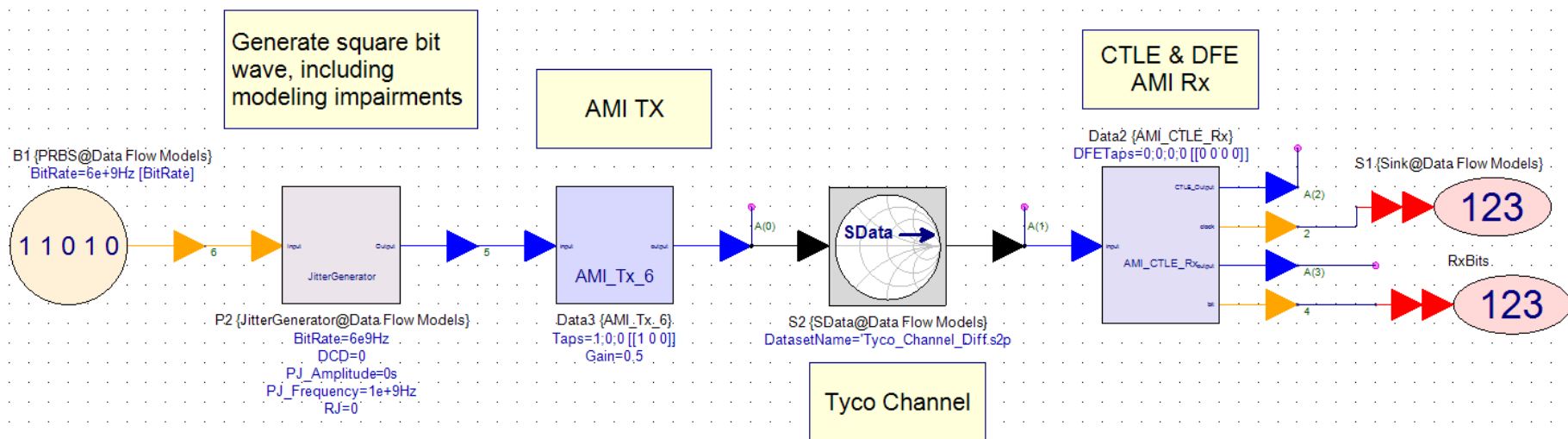
```
(sv_ami_tx
  (Reserved_Parameters
    (Init_Returns_Impulse (Usage Info) (Type Tap) (pulse True))
    (GetWave_Exists (Usage Info) (Type Tap) (pulse True))
    (Use_Init_Output (Usage Info) (Type Tap) (pulse True))
  )
  (Model_Specific
    (Coefficients
      (index0 (Usage In) (Type Tap) (pulse True))
      (index1 (Usage In) (Type Tap) (pulse True))
      (index2 (Usage In) (Type Tap) (pulse True))
    )
  )
)
```

The visual studio project automatically created -> One click to create .dll

# Example #1

## 6.0 Gb/s (SATA 3.0)

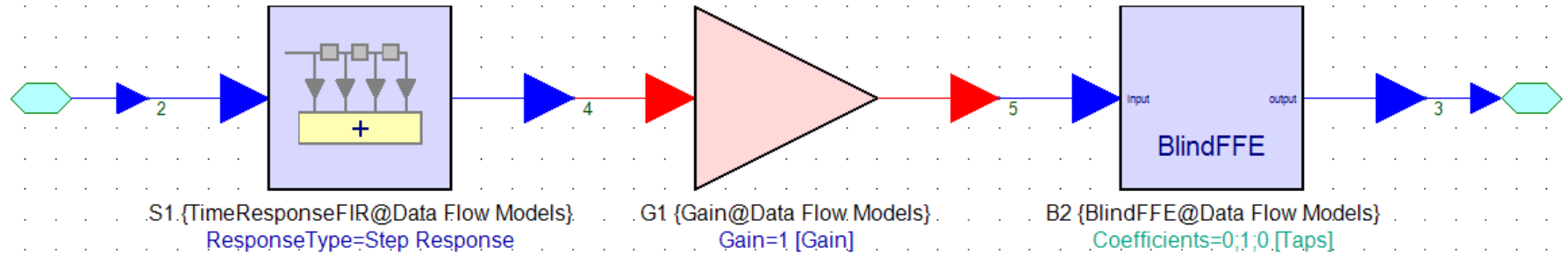
### 6.0 Gb/s SATA 3.0 SerDes



# TX Modeling

## 6.0 Gb/s (SATA 3.0)

### TX Architecture



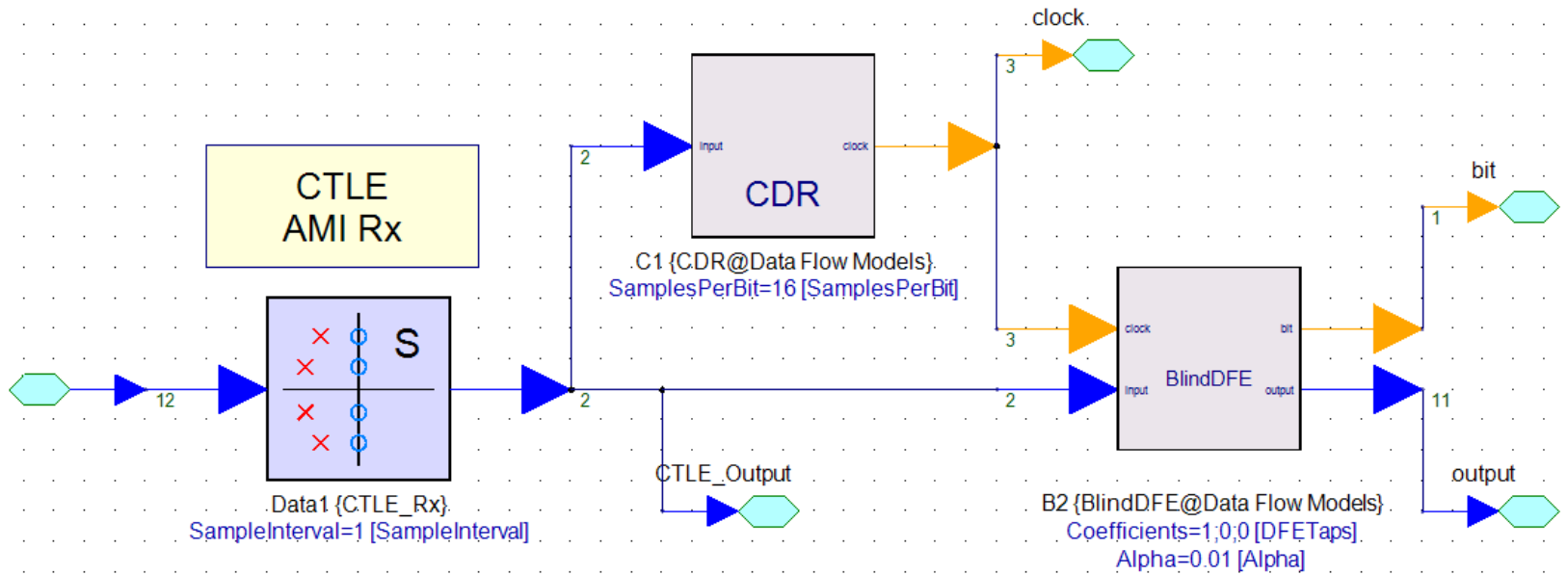
FIR filter

3-tap  
FFE

# RX Modeling

## 6.0 Gb/s (SATA 3.0)

### RX Architecture



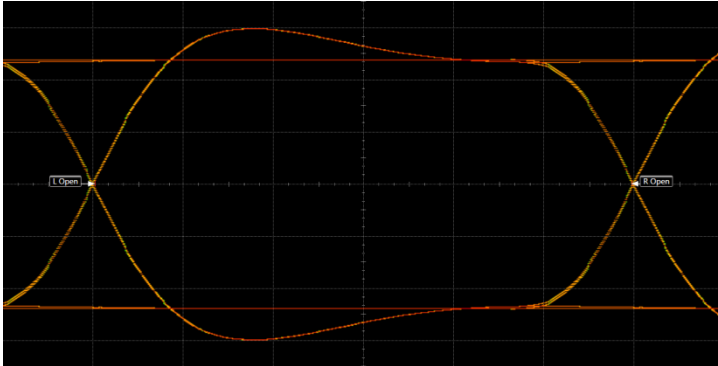
S-domain filter

3-tap DFE

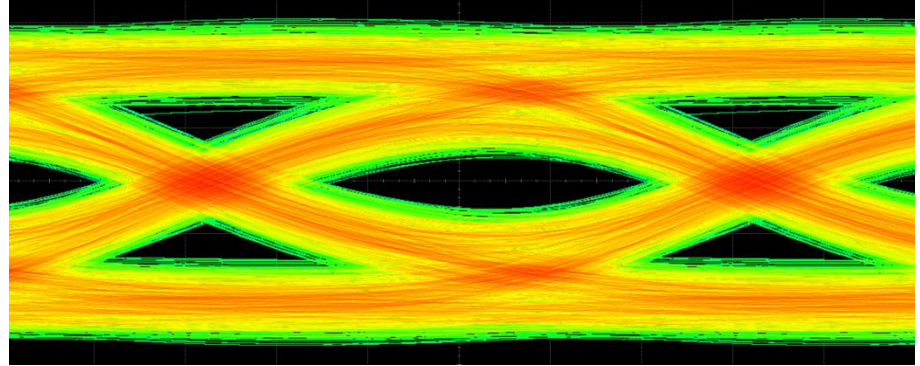
# Results

## 6.0 Gb/s (SATA 3.0)

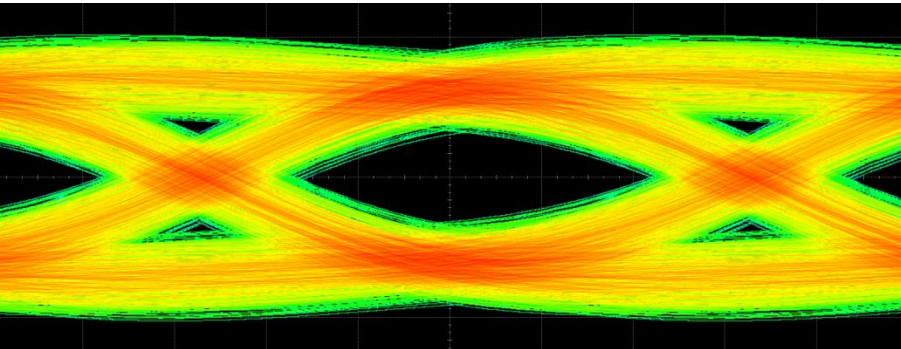
TX Output



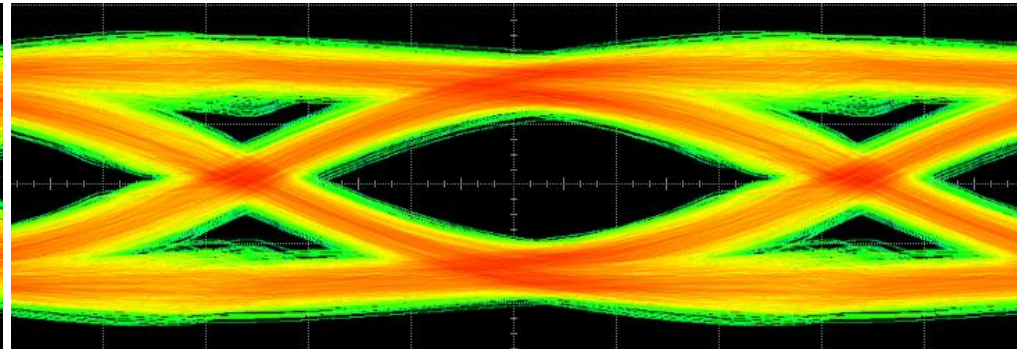
After Channel



After CTLE EQ



After CTLE+DFE EQ

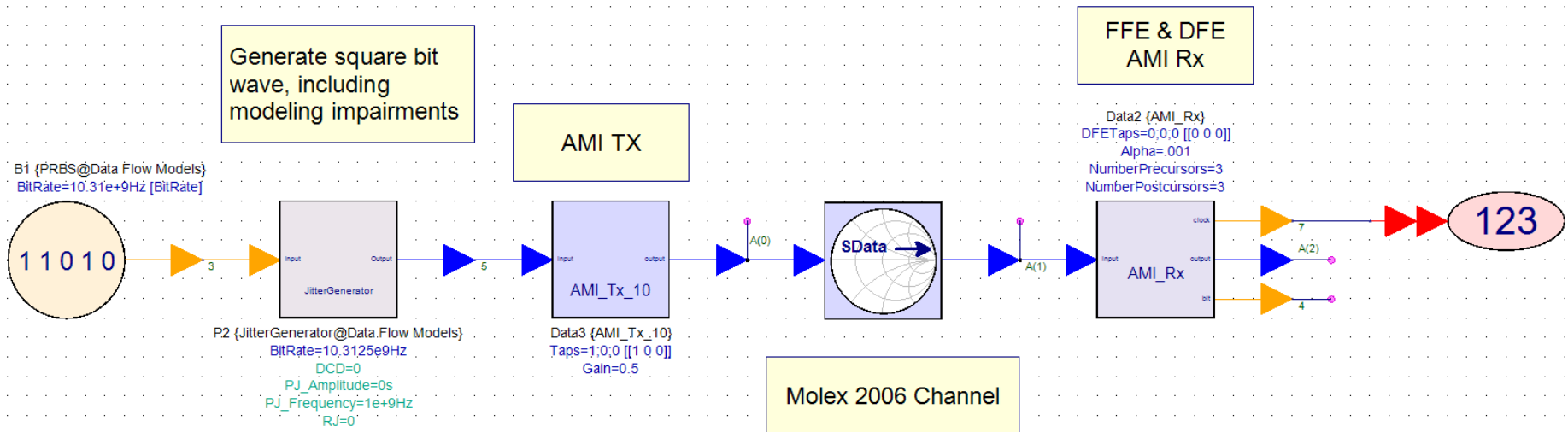


\*Note: EQ taps not optimized for maximum eye

# Example #2

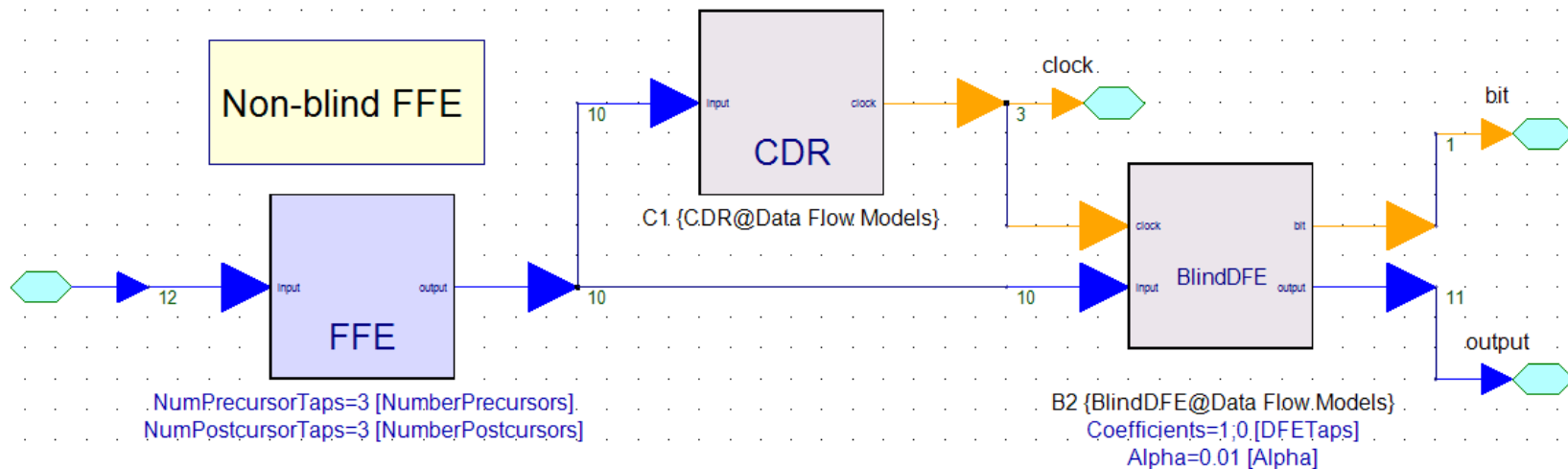
## 10.3125 Gb/s (10-GB Ethernet)

### 10.3125 Gb/s SerDes



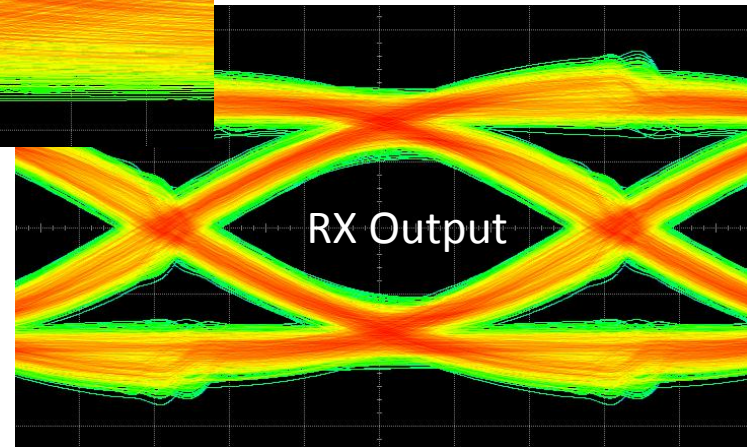
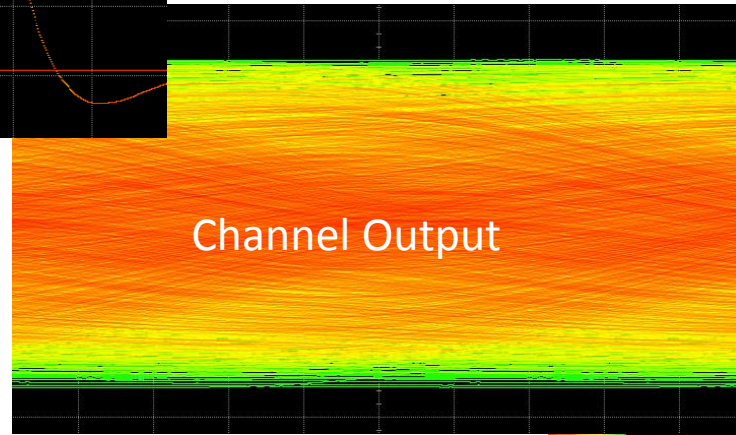
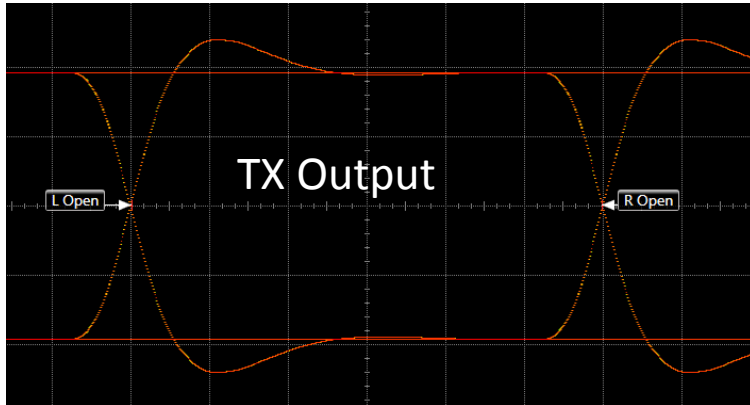
# RX Modeling

## 10.3125 Gb/s (10-GB Ethernet)



# Example #2

## 10.3125 Gb/s (10-GB Ethernet)



# TX 10.3125 Gb/s

## *AMI model correlation study*



### Strategy

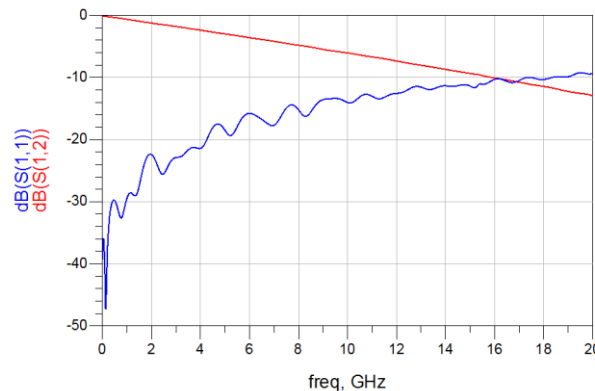
1. Correlate Transistor Simulation vs. AMI model
2. Correlate Measured vs. AMI model

# Transistor Simulation vs. AMI Model

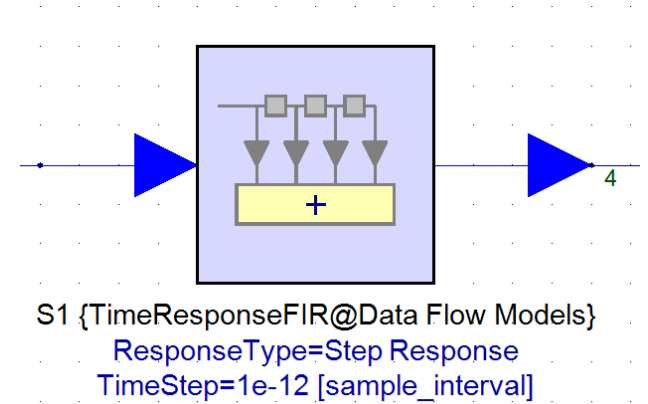
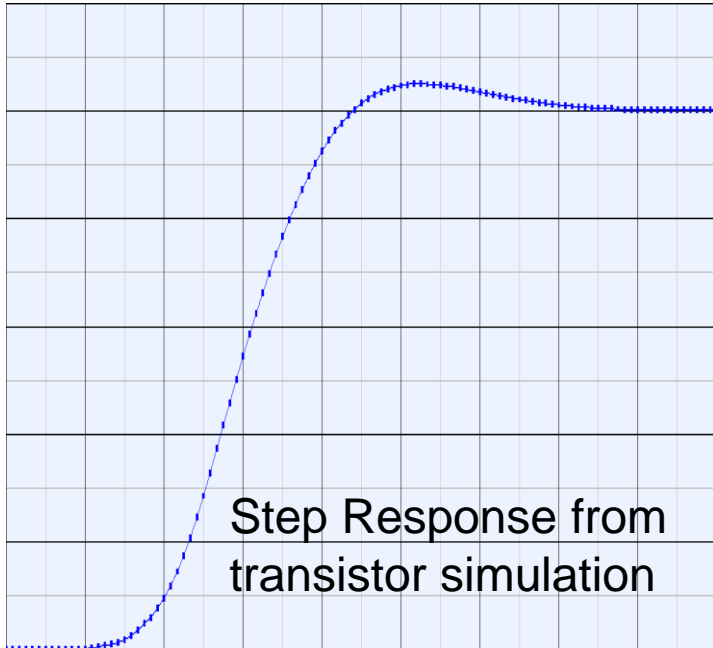


## Steps-

1. Generate Step Response from transistor simulation
2. Generate AMI model using EDA tool
3. Compare



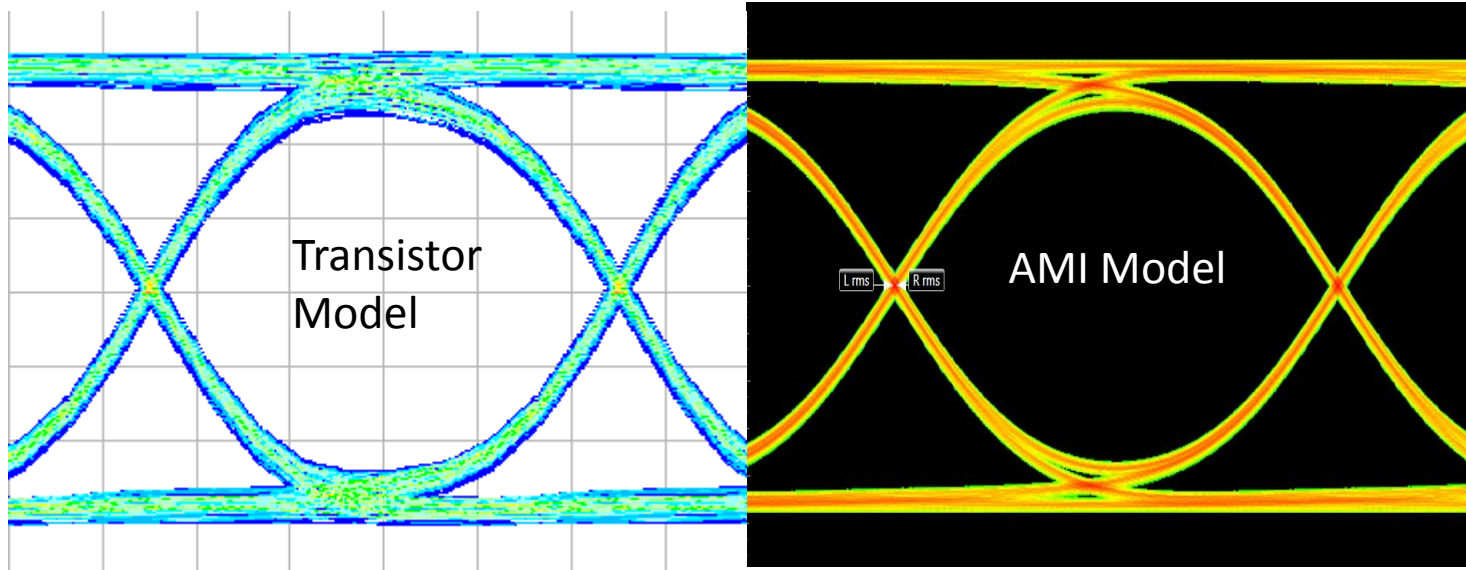
# Step Response Model



FIR filter with Step Response Input

# Correlation

## *transistor model vs. AMI model*



Excellent match between transistor simulation and AMI model

Good faith in model-generation methodology!

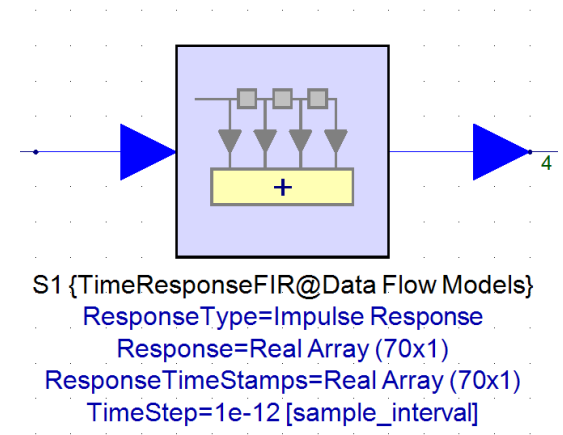
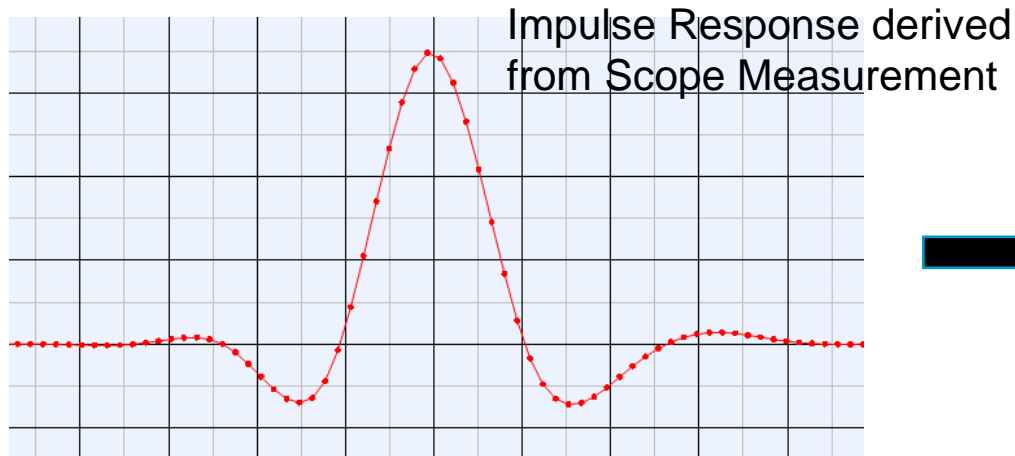
# Measurement vs. AMI Model



## Steps-

1. Measure waveform
2. De-embed Channel
3. Output Impulse response

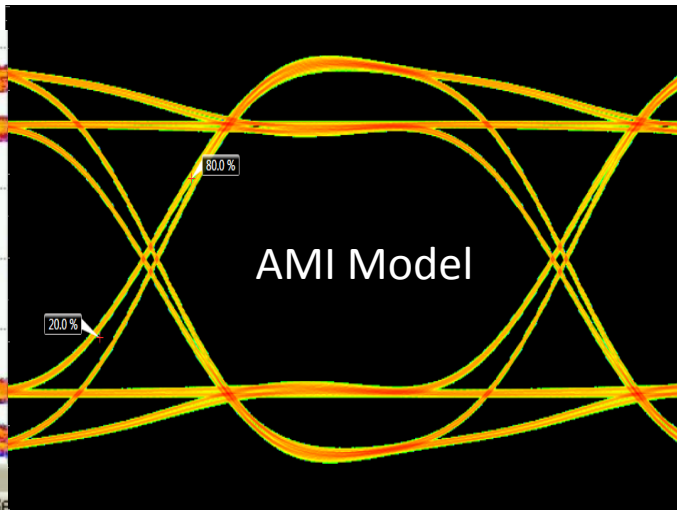
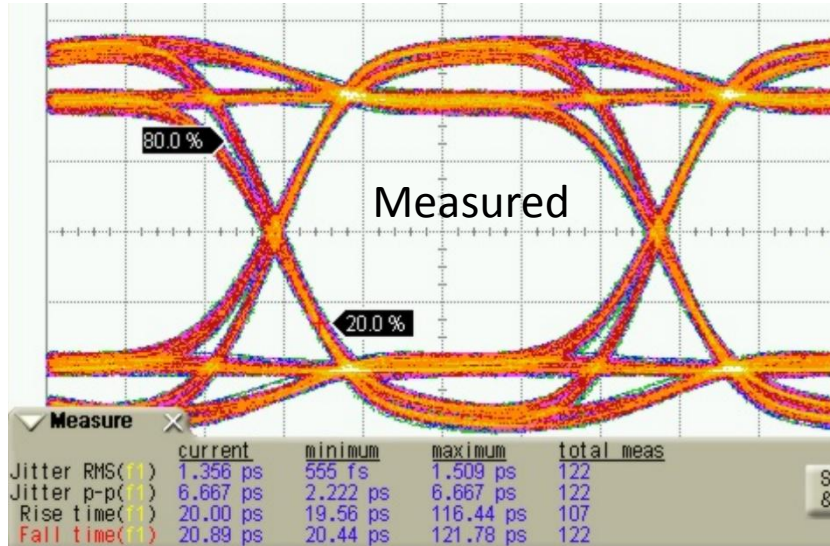
# Impulse Response Model



FIR filter with Impulse Response Input

# TX Correlation Measured

*emphasis #1: tap 0, 1, -0.2*

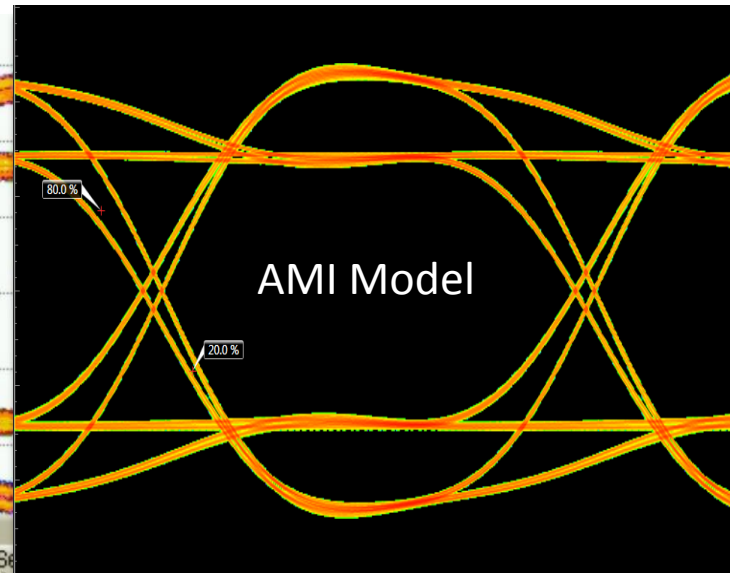
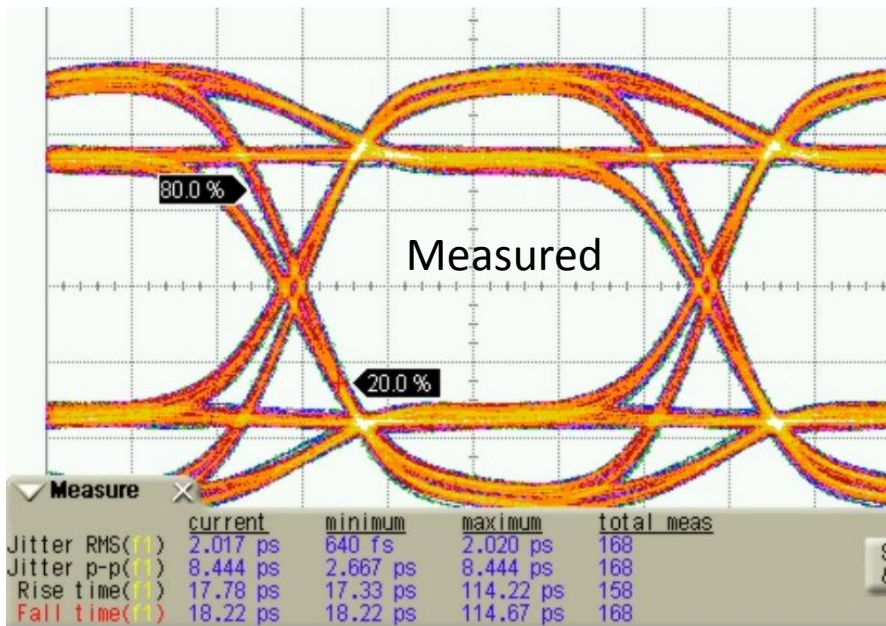


Measure	current	minimum	maximum	total meas
Jitter RMS(f1)	1.356 ps	555 fs	1.509 ps	122
Jitter p-p(f1)	6.667 ps	2.222 ps	6.667 ps	122
Rise time(f1)	20.00 ps	19.56 ps	116.44 ps	107
Fall time(f1)	20.89 ps	20.44 ps	121.78 ps	122

Measurement	Current	Minimum	Maximum
Jitter[rms]	1B 1.632 ps	1.632 ps	1.632 ps
Jitter[p-p]	1B 5.386 ps	5.386 ps	5.386 ps
Rise Time	1B 21.8 ps	21.8 ps	21.8 ps
Fall Time	1B 21.8 ps	21.8 ps	21.8 ps

# TX Correlation Measured

*emphasis #2: tap 0, 1, -0.25*



	current	minimum	maximum	total meas
Jitter RMS(f1)	2.017 ps	640 fs	2.020 ps	168
Jitter p-p(f1)	8.444 ps	2.667 ps	8.444 ps	168
Rise time(f1)	17.78 ps	17.33 ps	114.22 ps	158
Fall time(f1)	18.22 ps	18.22 ps	114.67 ps	168

Measurement	Current	Minimum	Maximum
Jitter[rms]	1B 2.160 ps	2.154 ps	2.162 ps
Jitter[p-p]	1B 6.464 ps	6.248 ps	6.464 ps
Rise Time	1B 20.4 ps	20.4 ps	20.4 ps
Fall Time	1B 20.4 ps	20.4 ps	20.4 ps

# Benefits of ESL Design Flow

## Automated AMI-Model Generation

1. Complete “Automation” of Code-generation and Model Compilation  
*a task that routinely takes months because of its complexity*
2. Basic building blocks that can used to start model development  
*FIR/IIR filters, FFE, DFE, CDR etc.*
3. Easily customize models to include custom IP  
*Custom C++ and Math-Lang*