Automating AMI Model-Generation using ESL (Electronic-System-Level) Design Flow

Amolak S. Badesha
Senior Application Expert
José Luis Pino
Principal Engineer

Agilent EEsof EDA

Manuel Luschas
Signal Integrity Manager
Antonis Orphanou
Signal Integrity Engineer

NetLogic Microsystems

European IBIS-Summit
Hildesheim, Germany
May 12th, 2010

Outline

-AMI Modeling Barriers

- Complex Code-generation process
- Limited Engineering Resources for Modeling

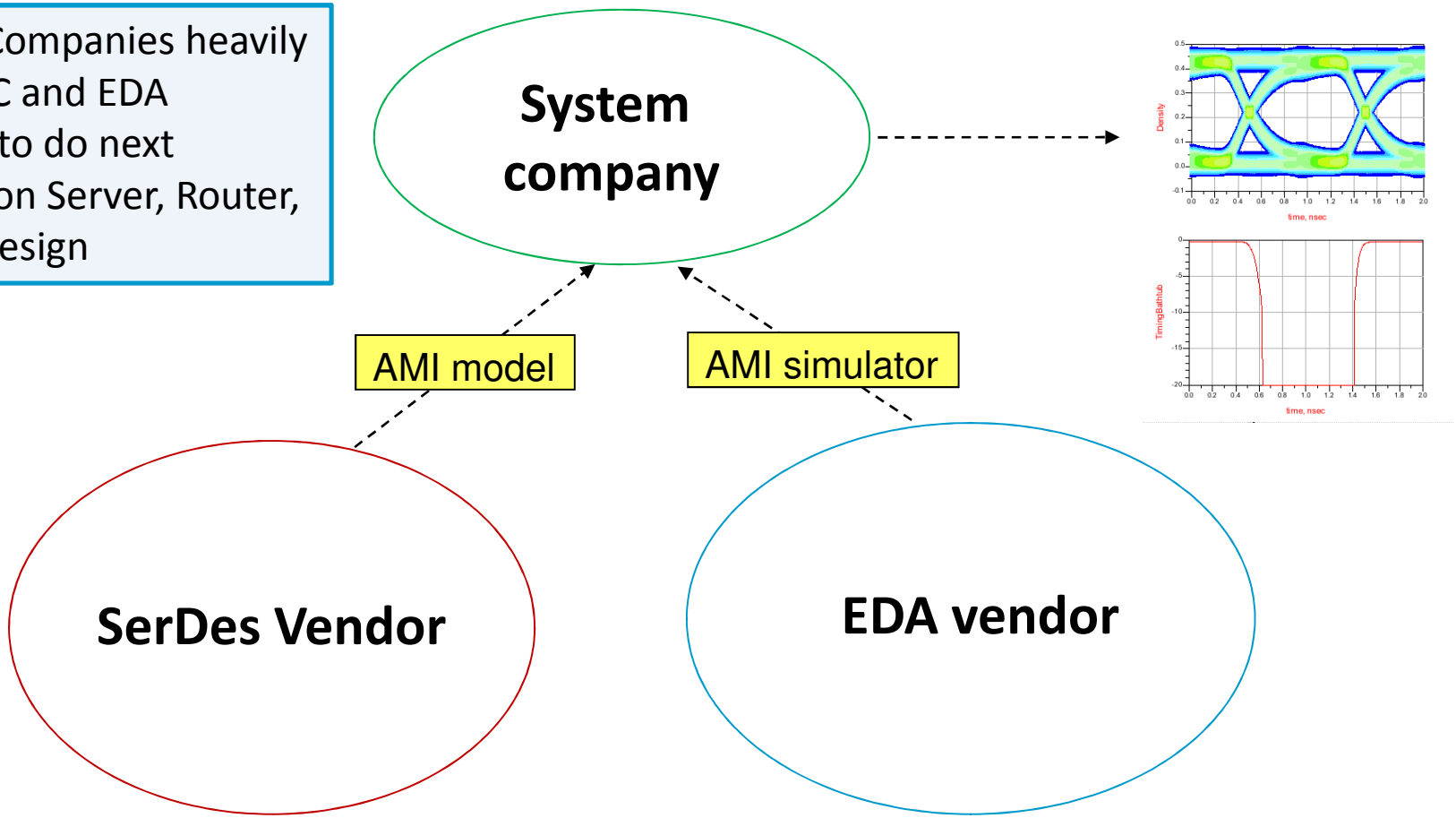
-Improved AMI Model-generation flow

- Model-based Architecture Design
- Automated AMI-Model Generation

-Business Implications

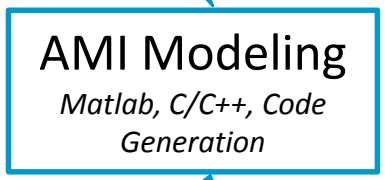
The AMI food-chain

System Companies heavily rely on IC and EDA vendors to do next generation Server, Router, Switch Design



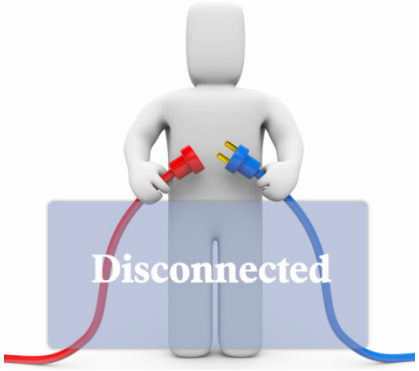
The AMI Model Lifecycle…disconnected flows

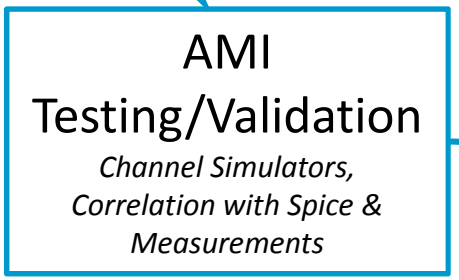
System Architect, IC Designer



Modeling Engineer

AMI-Modeling adds significant engineering requirement for coding, without any major benefit for IC vendor





SI/Application Engineer

System Design Engineer

Platform & Compiler Version dependencies, Correlation with Spice and Measurements

#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 Vendor models become available

Note: Vendors with NO experience in AMI modeling are spending 8-16+ months to come up with first-generation models

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

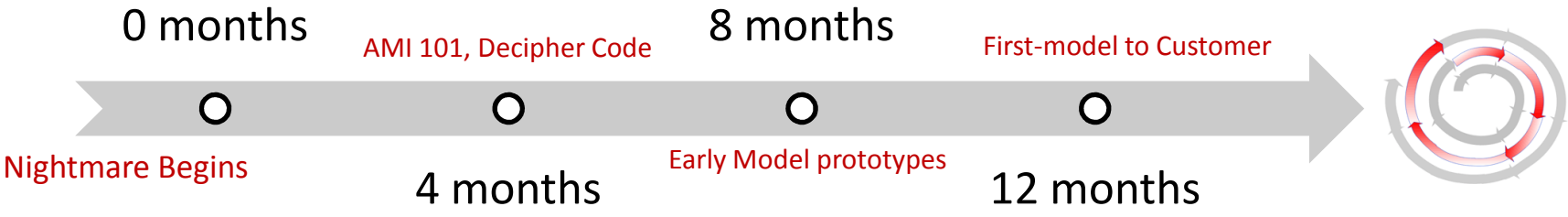
Why is it taking so long?

Model Generation Time

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



How to Speed up AMI Model Development?

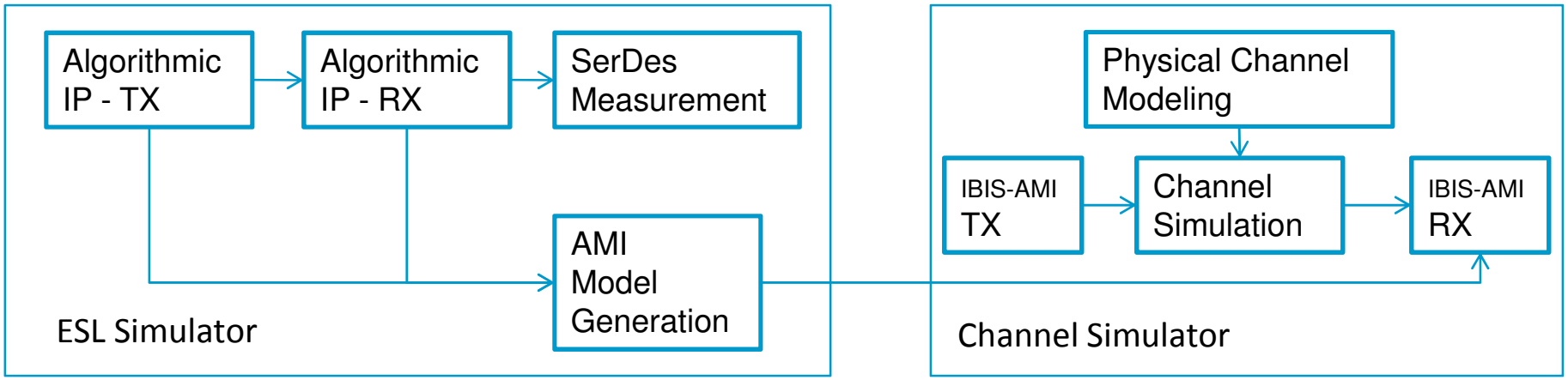
Can this be done?

Yes, EDA companies are good at “Automating” routine tasks -> that is why we exist

-Two key requirements:

-Efficient Tops-down “**E**lectronic-**S**ystem-**L**evel” Design Methodology

-Automate Code-generation



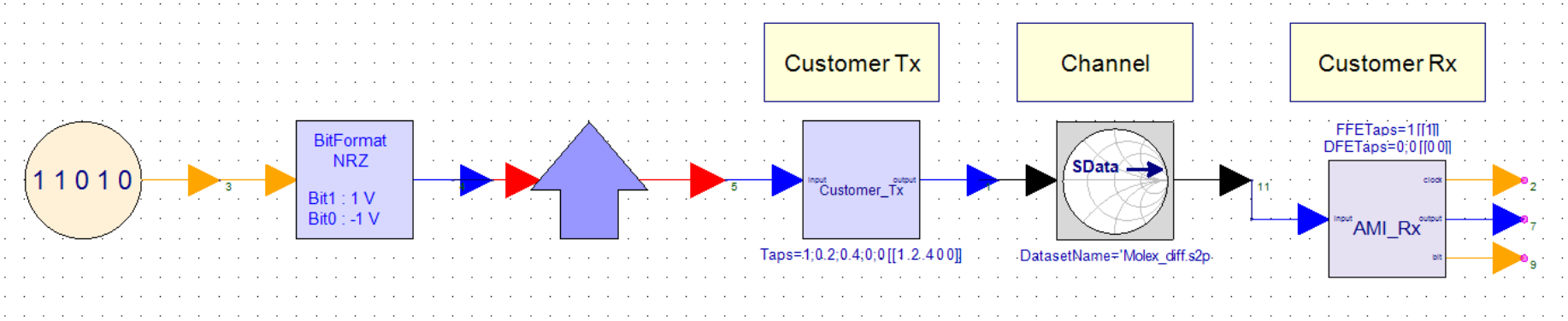
ESL Design Flow with Automatic AMI model-generation

“Electronic-System-Level” (ESL) Design Flow

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

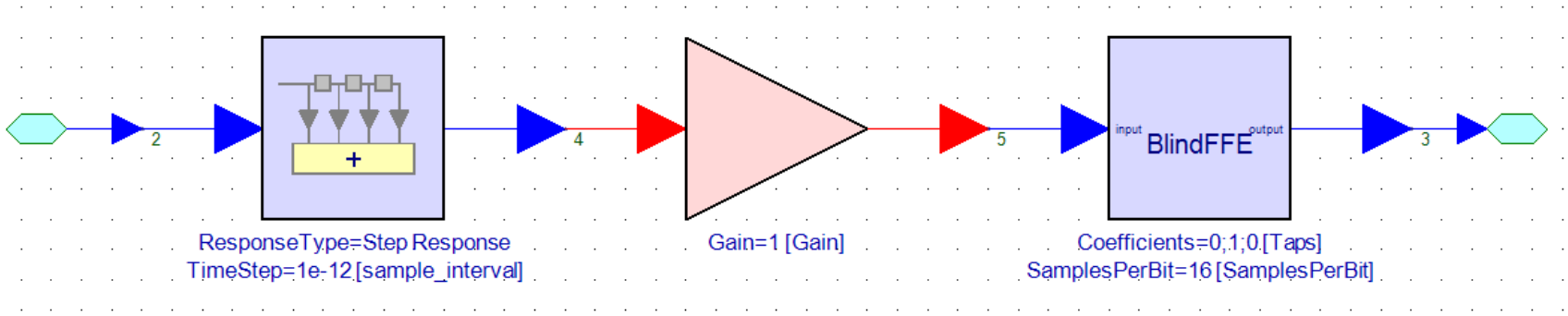
ESL flow facilitates utilization of appropriate abstractions in order to *increase comprehension about a system*, and to enhance the probability of a successful implementation of functionality in a *cost-effective* manner

Here is an Example of SerDes modeling using ESL flow-



ESL flow: TX Modeling Example (1)

Step-1: Starting Architecture Design with Generic Model



FIR/IIR filter

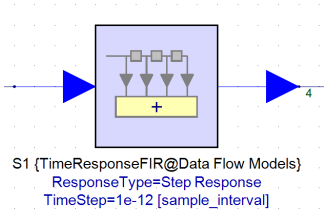
Gain

n-tap FFE

Different blocks represent high-level TX architecture

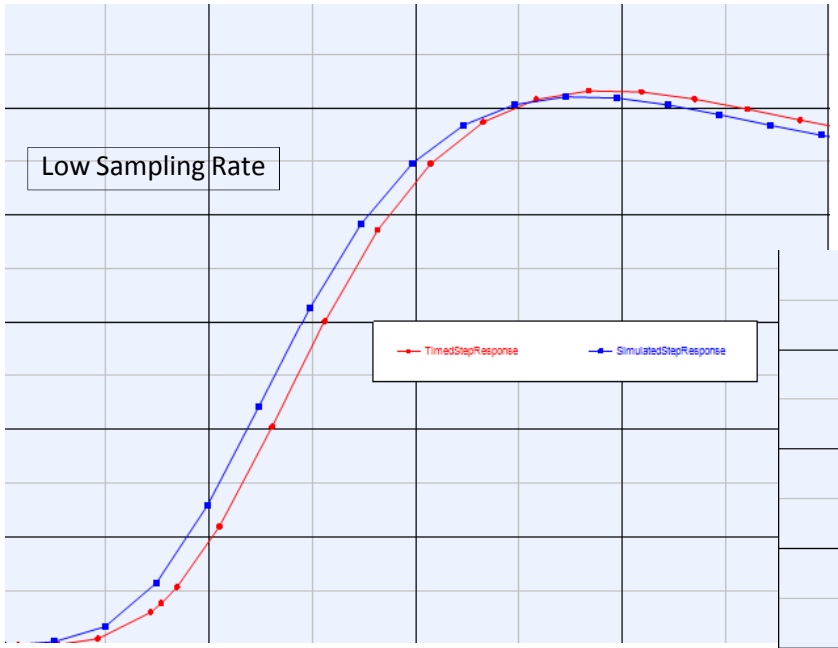
More on FIR Filter…

How to bring in Spice or Measured data?

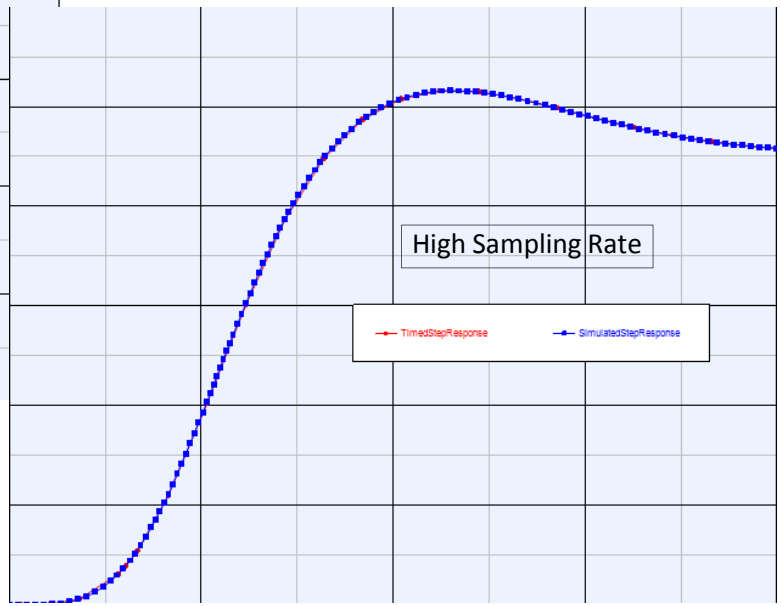


Challenges:

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



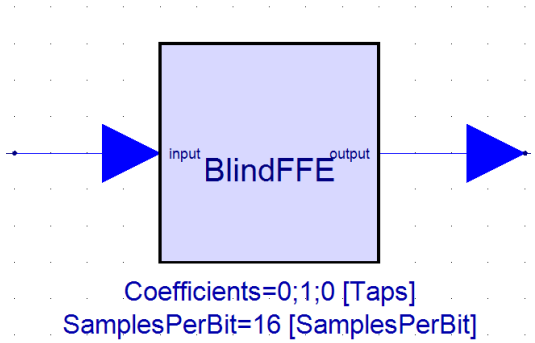
Sampling Rate determines Simulation Accuracy



FIR model should support "Arbitrary" Sampling Rate

ESL flow: TX Modeling Example (2)

Step-2: Customize IP -> Bring in M-code or C++ Code



Fine-tune and Customize models with Matlab Syntax and/or C++ code

Designator: B2 — Show Designator

Description: Blind Feed-Forward Equalizer

Model: MathLang@Data Flow Models — Show Model

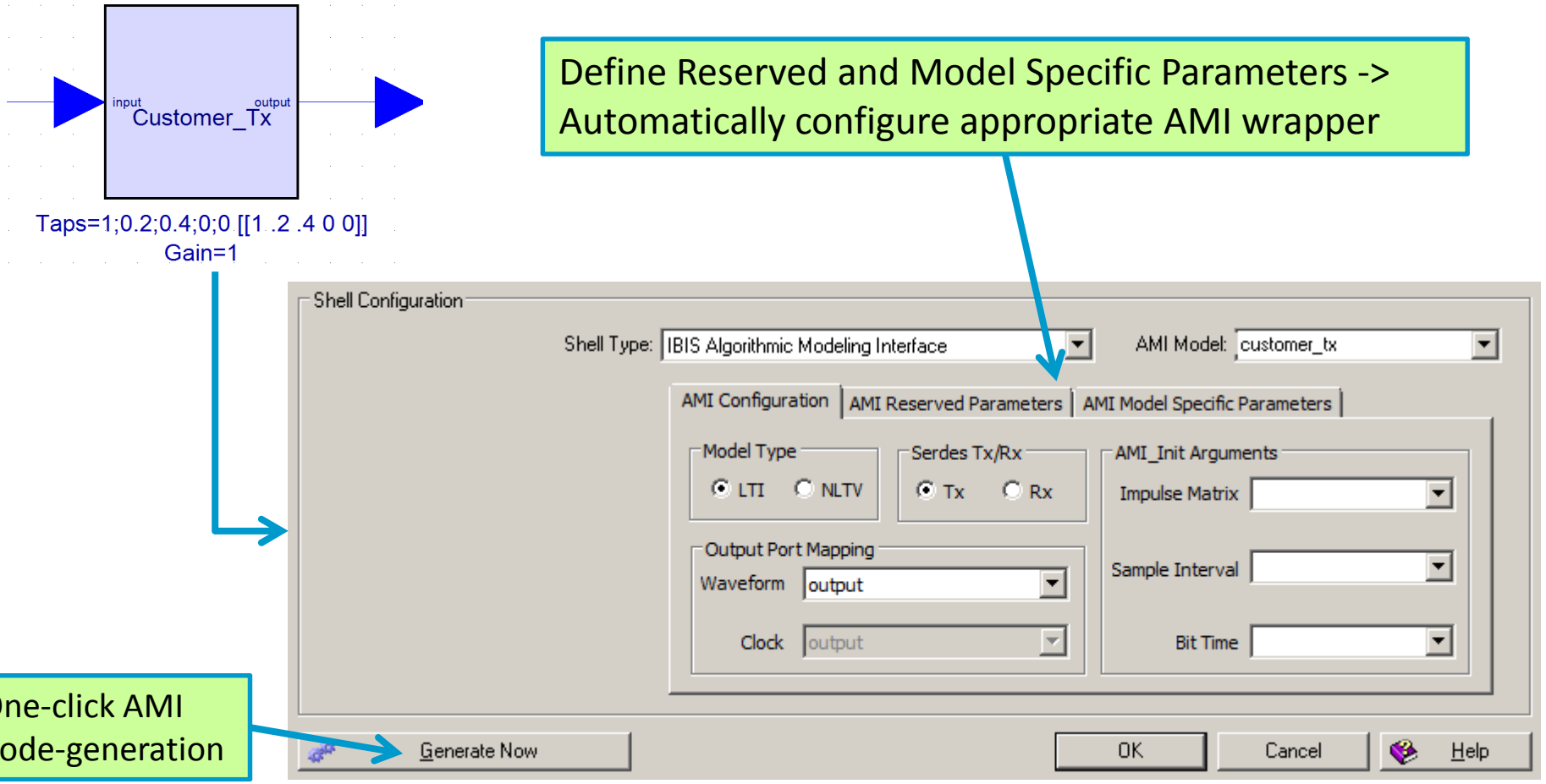
Manage Models... | Model Help | Use Model

Equations | I/O | Custom Parameters

```
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;
```

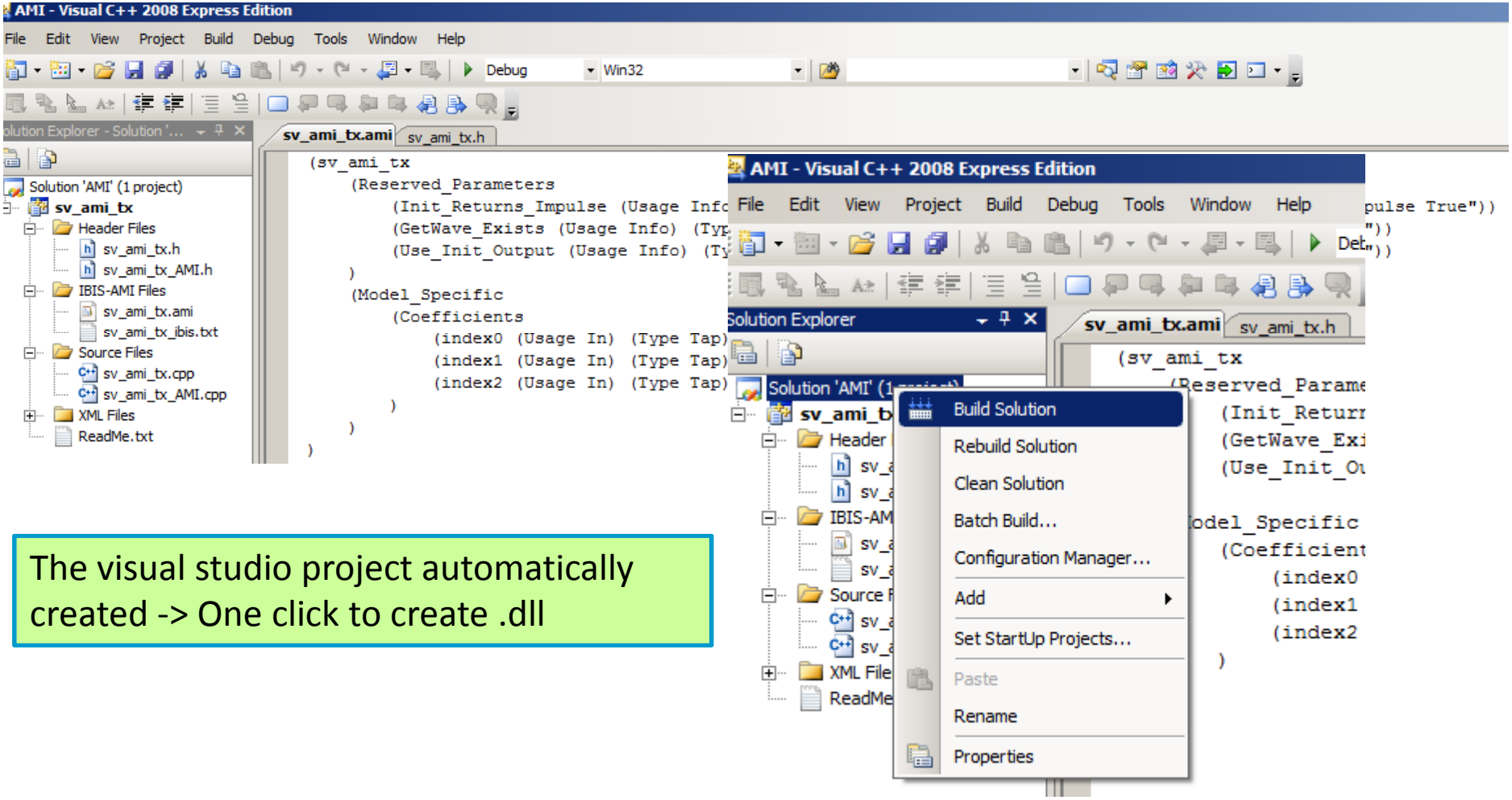
ESL flow: TX Modeling Example (3)

Step-3: One-click AMI Code-Generation



ESL flow: TX Modeling Example (4)

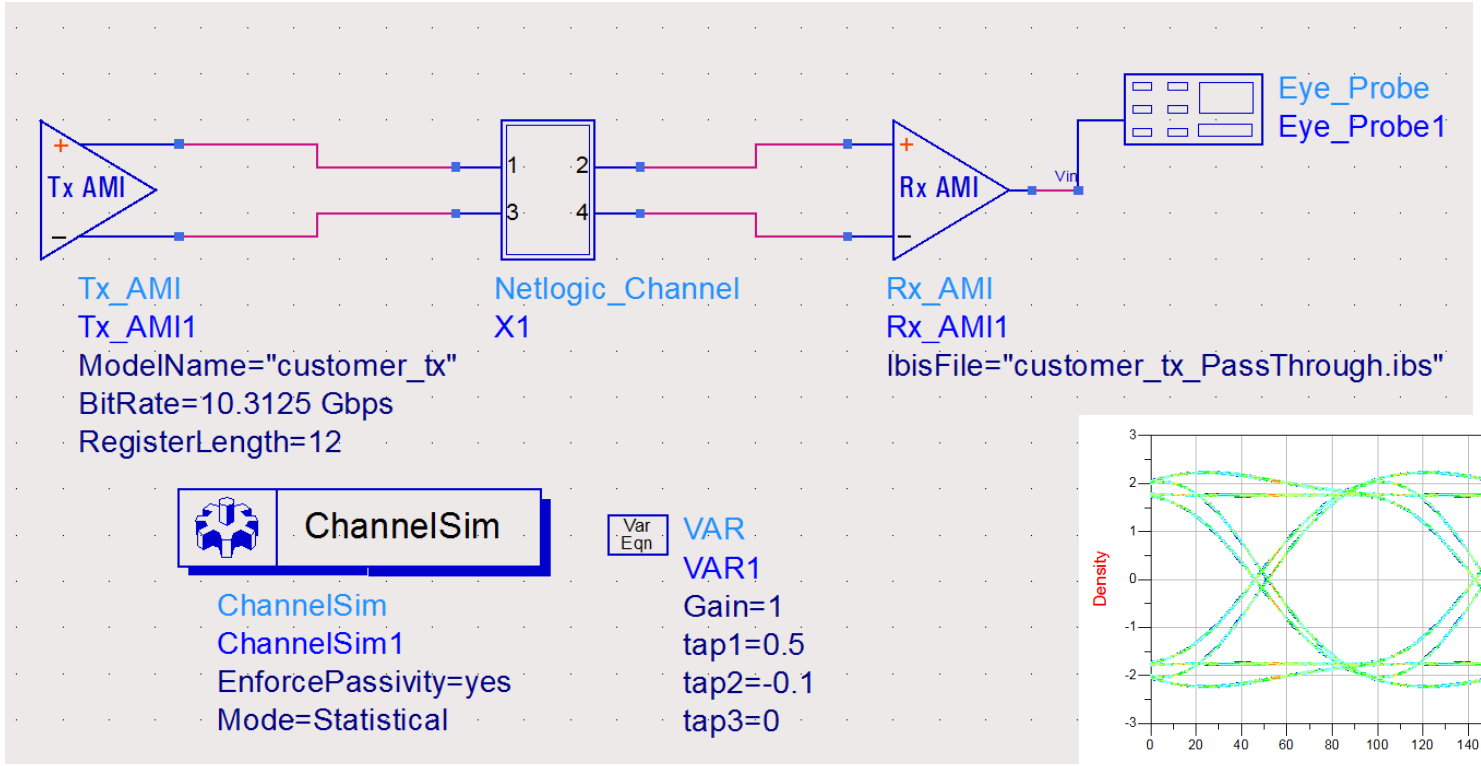
Step-4: Automatically Generated .ami and Visual-Studio project



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

Validating AMI-Model in Channel Simulator

AMI models produced using ESL-flow should support ALL Channel Simulators that are IBIS-5.0 compliant



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 in include custom IP
   Custom C++ and M-code

Benefits of ESL Design Flow for SERDES Design

- ✓Rapidly optimize the signal processing at the appropriate level of abstraction
  - ✓HDL simulators and SPICE aren't signal processing data flow tools
- ✓Implement the optimized architecture one time
  - ✓No need to iterate at the implementation level
- ✓AMI is a natural by-product of the architectural model
  - ✓Automatic C/C++ code generation and compilation from the model
  - ✓IBIS AMI wrapper enables compliance with the standard's API

Business Implications

Imagine AMI-models can be generated in Days, instead of Months

-> accelerate System-Design Innovation

-> level the playing field between SerDes vendors

(currently, big companies have advantage since they can pour lot of money/resources into modeling, whereas small competitors struggle)

-> SerDes companies focus on IP creation, not model generation

-> System companies can test “new-innovative” algorithms from IP vendors much earlier in design phase

Automated ESL AMI model-generation flow can significantly speed-up High-Speed System Design

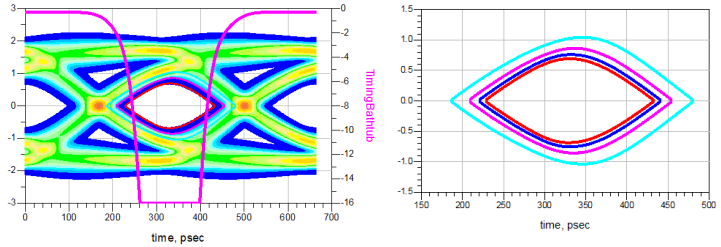
Thank You!

Amolak S. Badesha
amolak_badesha@agilent.com
408-553-2606

Backup Slides

Top Three reasons AMI Standard exists…

1. Fast Serial-Link Simulations (alternative to Spice, AMS)



2. Model Inter-operability between multiple SerDes vendors

3. IP protection for SerDes vendor