A New Method For Developing IBIS-AMI Models
Executive Summary

- Model simulation speed is about five times faster
- Model achieves good matching to the circuit simulation
- Runs in various simulators (ADS, QCD, HyperLynx, …)
Outline

Motivation and Introduction

IBIS-AMI Model Development Challenges

Proposed IBIS-AMI Development Flow

Engineering Trade-offs

An IBIS-AMI Model Development Example

Results, Verification, and Correlation

Conclusions
Motivation and Introduction

Accurate and fast link simulation is required at early system design stages to avoid over-design or under-design and to assess margin.

SerDes vendors desire a convenient IBIS-AMI model development flow that deals with the development challenges.
IBIS-AMI Model Development Challenges

- Architecture development vs. IBIS-AMI model generation
- Circuit vs. IBIS-AMI model representation
- Model abstraction
- Model development
- Model compilation
- Model debug
IBIS-AMI Model Development Challenges (Cont’d)

- Architecture development vs. AMI model generation
  - AMI model is more than architecture model
  - Need to match design implementation
  - Need to reflect silicon behavior across PVT
IBIS-AMI Model Development Challenges (Cont’d)

Circuit vs. IBIS-AMI model representation

- Identify and categorize circuit into LTI and NLTV systems
- Frequency domain vs. time domain LTI modeling
- Modeling of nonlinear circuit behavior
- Modeling of time varying circuits
IBIS-AMI Model Development Challenges (Cont’d)

IBIS-AMI model abstraction
- Decide which parts of the model to place .ibs or .dll (.so)
- Identify signal flow and control flow
- Code signal flow into interconnected LTI and NLTV blocks
- Parameterize relevant control signals
AMI model development requires many skills

- Master C/C++ coding skills
- Guarantee coding compatibility across platforms
- Compile and link program in both Windows and Linux
- In depth understanding of IBIS-AMI standard
AMI model debug

- Need to maximize code development efficiency
  - Debug algorithmic code within one model development platform, independently from use in any Channel Simulator

- Need to minimize delivered code debugging
  - Only debug ported AMI model in Channel Simulator for possible code interface problems
Proposed IBIS-AMI Development Flow

- Model hierarchy partition
- Modularization and component reutilization
- Debugging hooks embedding
- Scripting and automation
Proposed IBIS-AMI Development Flow (Cont’d)

Model hierarchy partition
- Model nonlinear analog circuit in AMI portion
- Match circuit hierarchy closely
- Separate analog modeling from digital modeling
- Separate data flow path from control flow path
Proposed IBIS-AMI Development Flow (Cont’d)

- **Modularization and component reutilization**
  - Minimizes coding maintenance and verification load
  - Reduces model development cycle
  - Helps in hierarchy building

- **Debugging hooks insertion**
  - Early planning
  - Debug source code vs. DLL/SO
Proposed IBIS-AMI Development Flow (Cont’d)

Scripting and automation

– Have a wrapper script to generate IBIS-AMI ready code from the core code
– Automate the .ami file generation based on model parameter list
– Eliminate redundant work
– Guarantee the robustness of the compilation and build process
– Ensure IBIS-AMI compliance
Engineering Trade-offs

- Accuracy vs. speed
- Custom models vs. library models
- Hierarchy levels
Engineering Trade-offs (Cont’d)

Accuracy vs. speed

- Higher accuracy typically comes with lower speed and longer development cycle
- Balance the accuracy level and the simulation speed of each component to achieve optimal accuracy and acceptable simulation speed
Engineering Trade-offs (Cont’d)

Library models
- save development time
- reduce the chance of error
- Good for initial model generation and proof of concepts.

Custom models
- improve model accuracy
- enhance programming flexibility
- requires full verification
- increases development time
Hierarchy levels

- Maximize coding flexibility
- Code reuse and maintenance
- For design match up and correlation
- For debugging and model enhancement
An IBIS-AMI Model Development Example (RX)

- Six modules
- Customized modeling
- Speed & mem optimization
An IBIS-AMI Model Development Example (Cont’d)

- Assembled and debugged in SystemVue
- Compilation process is automated

Diagram:
- Visual Studio C++
  - Create
- C++ Custom Models
  - Load
- Top Level C++ Model
  - Build
- Auto gen
- SystemVue
  - IBIS-AMI Model
  - .ibs
  - .ami
  - .dll
- Compile
- SystemVue model
  - Probe and debug
Results Verification and Correlation

- Both TX and RX models are generated by the same flow
- The models verified on multiple EDA tools
- Results correlate with design
Results Verification and Correlation (Cont’d)

Example 1:
- Simulated at 32Gbps
- IL = 22dB at 16 GHz
- RL = 17dB at 16 GHz
Results Verification and Correlation (Cont’d)

- 32 Gbps, 22dB loss channel
- BER < 1E-18 with 120mV margin
Example 2:
- Simulated at 28Gbps
- IL = 30dB at 14 GHz
- RL = 18dB at 14 GHz
Results Verification and Correlation (Cont’d)

- 28 Gbps, 30dB loss channel
- BER<1E-18 with 80mV margin
Results Verification and Correlation (Cont’d)

- Verified across various EDA tools on Linux and Windows
- Achieved general matching
  - Matched TX output waveform with design transient simulation
  - Matched RX internal waveform with design transient simulation
- Simulation takes about 6 minutes per one million bits
- Will correlate with and match to silicon in the future
Conclusions

- Challenges and trade-offs often encountered in IBIS-AMI model development are discussed and addressed in this presentation.
- Hierarchy partition, modularization and reutilization, debugging hooks embedding, scripting and automation form the practical IBIS-AMI development flow proposed.
- This flow not only helps to reduce model generation cycle, but also assures acceptable simulation speed and good accuracy.