



IBIS-AMI Modeling Recommendations - European IBIS Summit 2010

May 12, 2010 Hildesheim, Germany

Kumar Keshavan Ken Willis

Presented by Srdjan Djordjevic





- When is AMI required?
- IBIS-AMI key concepts
- General AMI recommendations
- Real-world AMI examples



When is AMI required?

- IBIS-AMI key concepts
- General recommendations
- Real-world AMI examples



When is AMI required?

- AMI is required when *adaptive filtering* is done by the Serdes Tx or Rx
- This means that the filtering automatically adjusts to the specific channel, based on its own specific algorithm
- For applications that use "static" filtering (ex. PCI Express Gen 1), the behavior can be represented in a circuit model, and AMI is NOT mandatory



AMI - Required vs. Convenient

• Despite that fact that AMI may not be **REQUIRED** for static preemphasis, it can be CONVENIENT to do so •Folding the preemphasis filtering into an algorithmic model is convenient because filter settings can be modified without requiring additional characterizations to be run for the analog channel





- When is AMI required?
- IBIS-AMI key concepts
- General recommendations
- Real-world AMI examples



IBIS-AMI Key Concepts

- Circuit and algorithmic models
- "Tap" terminology
- IBIS-AMI data flow and APIs





Circuit Models and Algorithmic Models

- The Tx to Rx pathway is composed of 3 separate entities
 - Tx algorithmic part
 - Analog (i.e. "circuit") channel part
 - Rx algorithmic part
- Three "decoupled" parts can be *independently* solved in time domain
 - Underlying assumption is HIGH IMPEDANCE connection between analog and algorithmic elements





"Tap" Terminology



As typically seen in "FFE" implementations



IBIS-AMI Data Flow and APIs





AMI APIs - Impulse Response or Waveforms

- AMI_Init
 - Takes in the impulse response of the channel
 - Algorithm in DLL decides how to best filter it
 - The filtered (and hopefully improved) "modified" impulse response is passed back to the tool
- AMI_GetWave WWW/WWW
 - Takes in raw waveforms of the channel
 - Algorithm in DLL decides how to best filter it, "real time"
 - The filtered "modified" waveform is passed back to the tool, along with the clock ticks (sampling information)



- When is AMI required?
- IBIS-AMI key concepts
- General AMI recommendations
- Real-world AMI examples



General Recommendations

- Circuit vs. algorithmic model content
- When to use AMI_Init vs. AMI_GetWave
- Statistical analysis and AMI_Getwave
- Using ibischk5 for .ibs and .ami files
- IBIS-AMI and vendor-independence





Circuit vs. Algorithmic Model Content

- Don't try to put circuit parasitics into the algorithmic portion of the model
- Leaving out circuit parasitics means you will miss reflections from impedance discontinuities that exist between the Tx output / Rx input and the interconnect channel
- These should get captured in the impulse response
- If you leave these out you will not correlate back to golden data from circuit models (ex. transistorlevel IO models)



Using AMI_Init vs. AMI_GetWave

- Basic principle K.I.S.S.
 - "Keep it simple, SI people!"



- AMI_Init > modifies the impulse response
 - If the filtering functionality sets up the coefficients once based on the channel, this API is the simplest implementation
- AMI_GetWave > modifies the raw waveforms
 - If the filtering functionality has real-time, dynamic adaptation of its coefficients based on the incoming waveforms, you need this API
 - If the algorithm includes clock and data recovery (CDR) functionality, this API is needed
- Bottom line > use AMI_Init if it will do the job, otherwise use AMI_GetWave
- Avoid extraneous functionality, and unnecessary complexity





Statistical Analysis and AMI_GetWave

- Pure Statistical Analysis is <u>not</u> generally compatible with AMI models using AMI_GetWave
- Should not assume anything about inner workings of a "black box" DLL "AMI_GetWave" algorithm
- Could have non-LTI behavior
 - Usually Receiver Models
 - Adaptive DFE
 - Pattern Dependent Equalization
 - Time Domain Clock and Data Recovery
- Only <u>limited</u> Statistical Analysis is possible
 - Ex. post-processing of time domain data







Using ibischk5 for .ibs and .ami Files

- The IBIS5.0 golden parser "ibischk5" can operate on .ibs and .ami files (with –ami switch), ex:
 - ibischk5 <ibis_file.ibs>
 - Ibischk5 –ami <ami_file.ami>
- This should be run on all IBIS-AMI model kits delivered by model developers to users
- Users should run this on incoming models







IBIS-AMI and Vendor Independence

- The purpose of defining a standard is to enable a vendor-neutral format that users can consume with their EDA tool of choice
- Sigrity has seen many "IBIS-AMI models" that are <u>full</u> of vendor-specific content, and will only run in a specific tool
- This violates the spirit of the IBIS standard





- When is AMI required?
- IBIS-AMI key concepts
- General recommendations
- Real-world AMI examples







Real-World AMI Examples

- FFE Feed Forward Equalizer
- DFE Decision Feedback Equalizer
- Advanced DFE

All can be implemented with existing IBIS 5.0 functionality!







- FFE stands for Feed Forward Equalizer
- Typically used in Tx
- Mathematically

•
$$\mathbf{y}_{n} = \Sigma \mathbf{w}_{i}^{*} \mathbf{x}_{i}$$

Yn – output



DFE

- DFE stands for Decision Feedback Equalizer
- Removes inter-symbol interference (ISI) by adding corrections to the input based on previous decisions
- $\mathbf{y}_{n} = \mathbf{x}_{n} + \Sigma \mathbf{w}_{i}^{*}\mathbf{d}_{i}$
 - y_n output
 - x_n input
 - d_i previous 'i_{th}' decision
 - w_i i_{th} tap weight







Advanced DFE



- Optional tuning of decision point
- Optionally include precursor



Thank You!

