IBIS-AMI Modeling Using Scripts and Spice Models

Asian IBIS Summit
Shanghai, China
November 13th, 2017
(Previously presented October 18th, 2017)

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

Agenda:

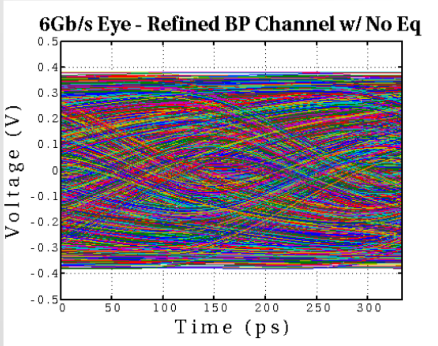
- Motivation
- Background
- IBIS-AMI Modeling Flow
- Modeling with Scripts
- Modeling with Spice models
- Summary
- Q & A

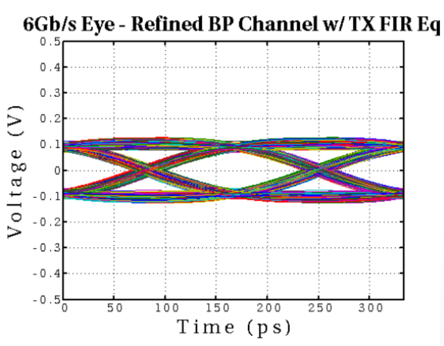
Motivation

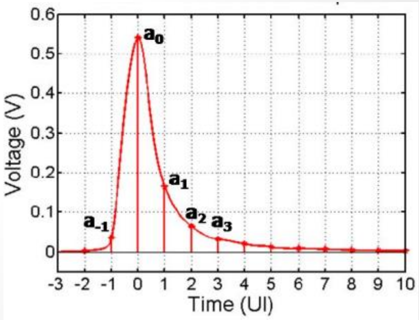
- Channel analysis usually requires IBIS-AMI:
  - For internal analysis and/or external model release
- AMI Modeling is technically challenging
  - Requires cross domain expertise
  - Take longer to ramp-up and develop comparing to IBIS
- Can we lower the AMI modeling barriers?
  - Use scripting languages
  - Use existing spice models

Background 1/3

- Channel analysis: [1]
  - Mostly have stages beyond traditional IBIS (e.g. Tx/Rx EQ)
  - Analyis methodologies [2]
    - Statistical: for LTI (Linear Time Invariant) circuit, using superposition
    - Time-domain: for NLTV (Non-Linear, Time Variant) circuit, using convolution







Background 2/3

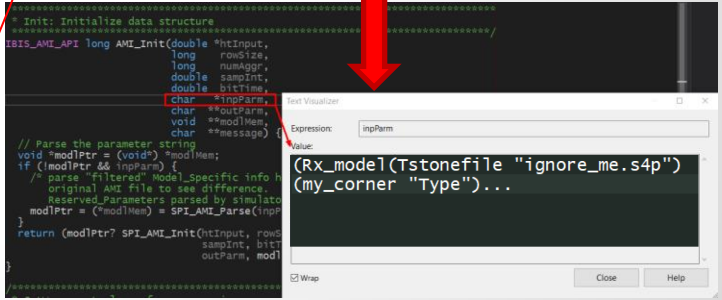
- AMI Model: [3]
  - Includes an .ibs, .ami and .dll/.so files
  - .ibs specifies .ami and .dll/.so files
  - .ami is a plain text file

```
[Algorithmic Model]
| The Model_type for the associated [Model] must be "I/O"
| "I/O_open_drain", "I/O_open_sink", "I/O_open_source", or "I/O_ECL".
|
Executable_Tx Windows_VisualStudio_32 tx_getwave.dll tx_getwave_params.ami
Executable_Tx Solaris_cc_32 libtx_getwave.so tx_getwave_params.ami
|
Executable_Rx Windows_VisualStudio_32 rx_getwave.dll rx_getwave_params.ami
Executable_Rx Solaris_cc_32 libtx_getwave.so rx_getwave_params.ami
|
[End Algorithmic Model]
```

This part is for simulator

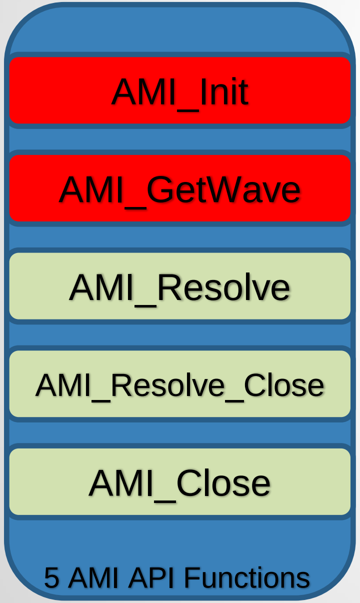
This part is for AMI model, passed into .dll/.so as name-value pairs

```
(Rx_model
  (Reserved_Parameters
    (Resolve_Exists (Usage Info) (Type Boolean) (Value True)
      (Description "Indicates whether the executable model implements
        AMI_Resolve."))
    (Model_Name (Usage In) (Type String) (Value "ignore_me")
      (Description "IBIS model name"))
    (Rx_Receiver_Sensitivity (Usage Out) (Type Float) (Range 0.0 0.0 0.01)
      (Description "Value depends on OP_mode and data rate"))    …
  )
  (Model_Specific
    (Tstonefile (Usage Dep) (Type String) (Value "ignore_me.s4p")
      (Description "Rx analog model. Value depends on OP_mode"))
    (my_corner (Usage In) (Type String) (Corner "Typ" "Min" "Max")
      (Description "Informs the executable model what corner is selected by
        user"))
    (OP_mode (Usage In) (Type Integer) (List 0 1 2 3)
      (Description "Operation mode"))
    …
  )
)
```



Background 3/3

- AMI Model:
  - .dll/.so may implements these API functions



AMI_Init → For LTI processing, when AMI_GetWave does not exist

AMI_GetWave → For LTI or NLTV processing, when AMI_GetWave does exist

These arrays serve as both waveform data input and output!

```
long AMI_Init (double *impulse_matrix,
               long number_of_rows,
               long aggressors,
               double sample_interval,
               double bit_time,
               char *AMI_parameters_in,
               char **AMI_parameters_out,
               void **AMI_memory_handle,
               char **msg)
```

```
long AMI_GetWave (double *wave,
        long wave_size,
        double *clock_times,
        char **AMI_parameters_out,
        void *AMI_memory)
```

AMI Modeling Flow

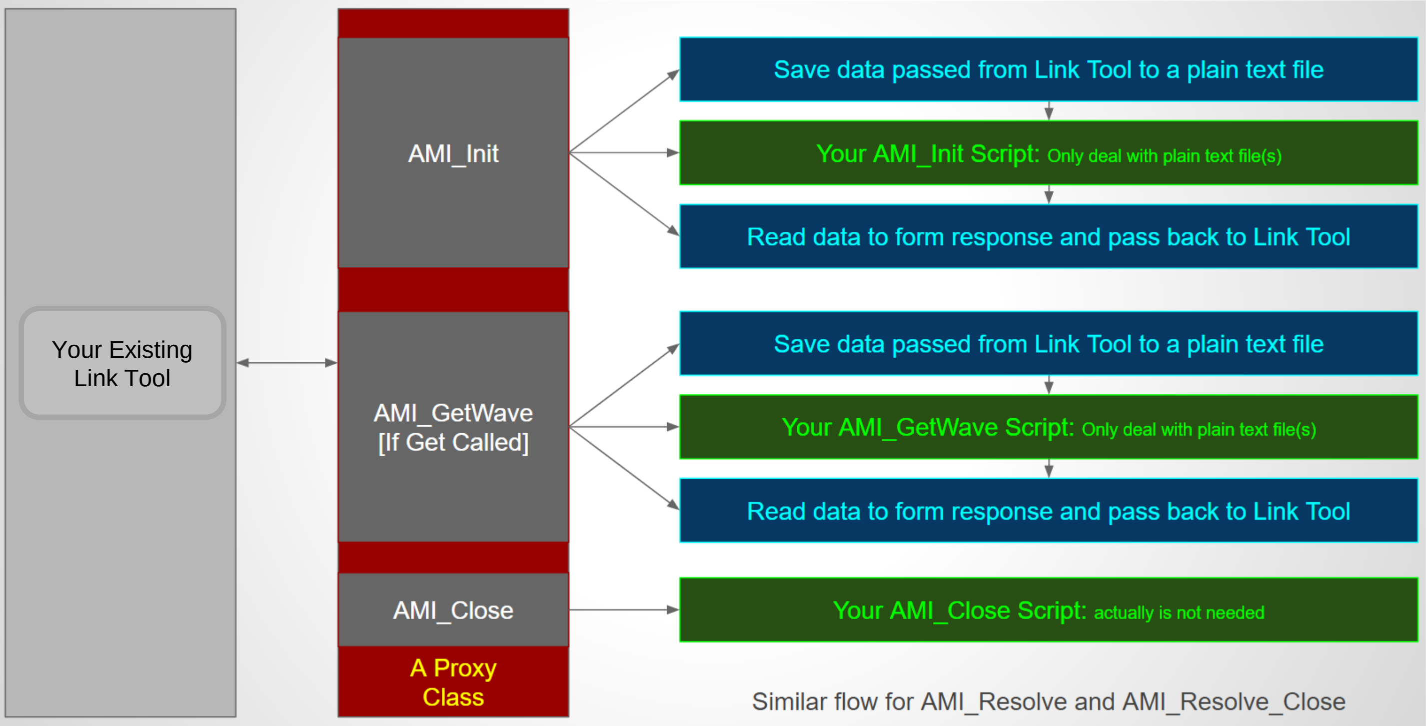
- Identify model behavior(s)
  - From mathematical equation, simulation or measurements

- Code the behaviors and IBIS-AMI API
  - API implemented MUST follow IBIS spec. and in C
  - Core processing can be in written other languages

- Compile and link as .dll or .so
  - Check library dependencies, different OS bits & linux distros

With Script and/or Spice models for core processing, this AMI model is very reuseable!

Modeling with Scripts: Flow



[4]

Script path and arguments are passed via .ami file

Modeling with Scripts: Example

Store calling arguments into text file

Call external scripts for processing

Parse resulting text file and return

An implement neutural AMI_Init() function in C

```
clear
clc

%% file to load data from
inpFile = 'AMI_Init_Inp.txt';
%% file to save data to
outFile = 'AMI_Init_Out.txt';

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%% Parse waveform data passed from the simulator
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
wave = parseInput(inpFile);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%% Perform AMI_Init using Matlab
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
sample_per_bit = floor(bit_time / sample_interval);
preTap   = -0.05;
posTap   = -0.1;
mainSig = 1 - abs(preTap) - abs(posTap);
waveInp = wave;
ht = [[1 zeros(1, sample_per_bit-1)] * preTap ...
      [1 zeros(1, sample_per_bit-1)] * mainSig ...
      [1 zeros(1, sample_per_bit-1)] * posTap];
out=conv(wave, ht);
wave = out(1:size(wave, 2));


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%% Store waveform data to return to the simulator
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
storeOutput(wave, outFile)
```

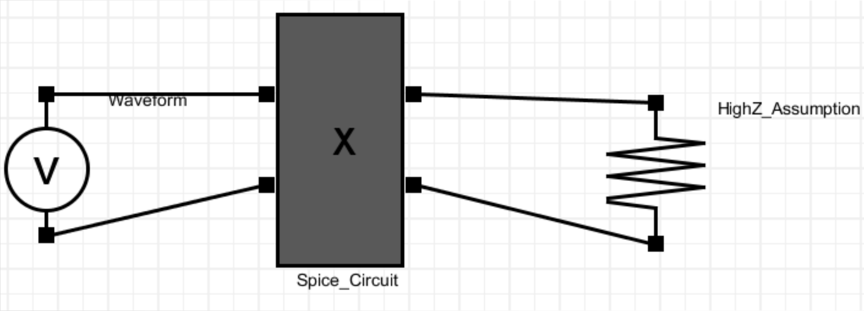
Modeling with Scripts: Consideration

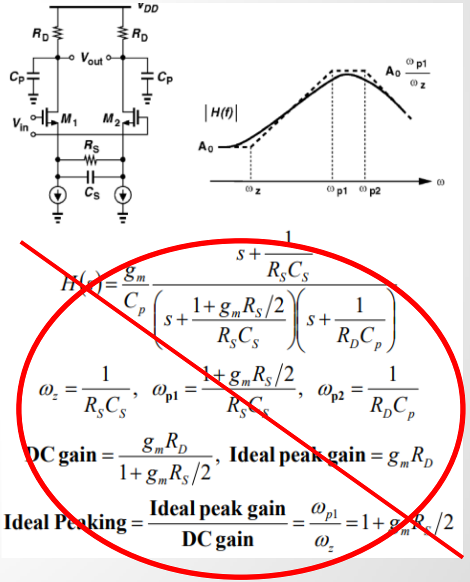
- Performance and distribuability:
  - Intepretor performance.
  - Redistributable (license)?
  - Does it require model user to install intrepreter?

- Consider Python! [5]
  - SciPy, NumPy etc for numerical analysis.
  - Embedded python: a single zip file together with AMI models.
  - Performance and extendability.

Modeling with Spice: Concept

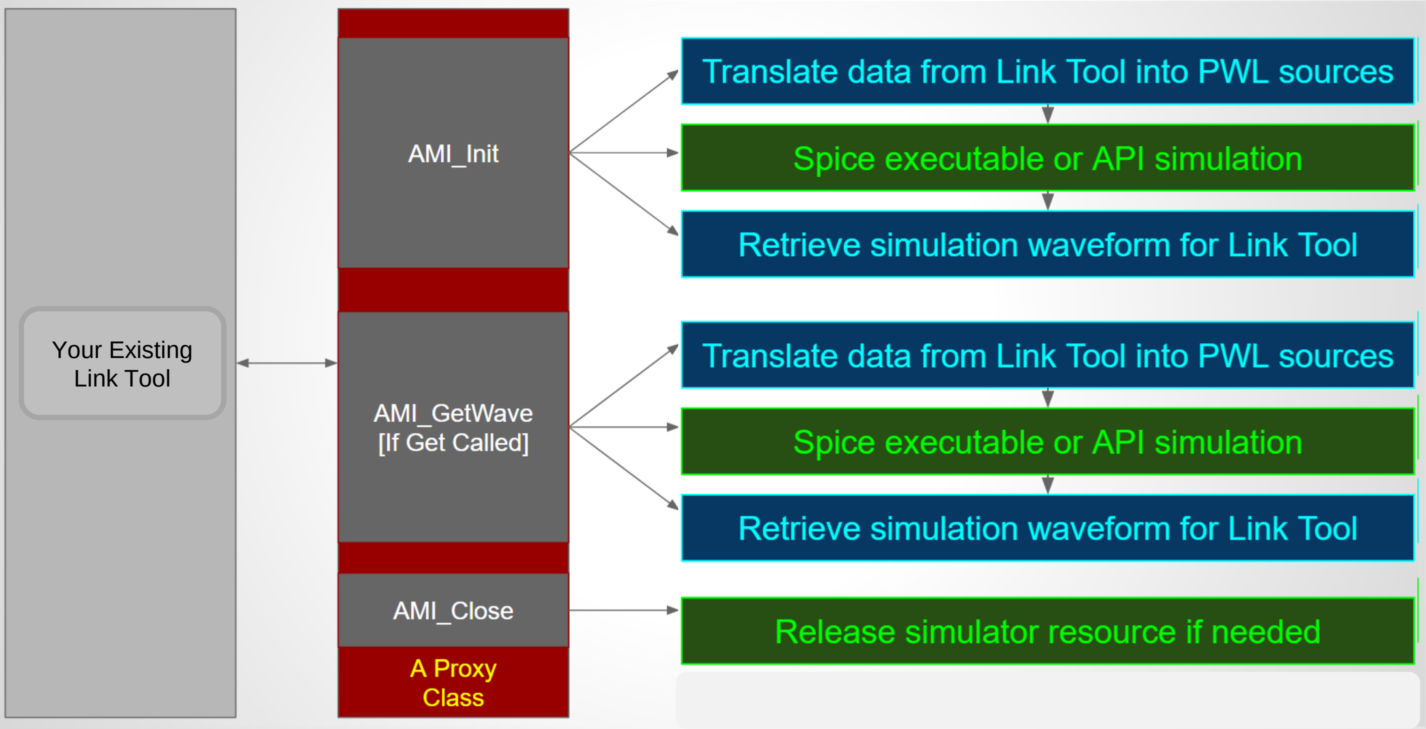
- Dynamically generated PWL inputs:
  - High-Z assumption
  - Simulate
    - Circuit may need to provide GND.
  - Retrieve waveform and return



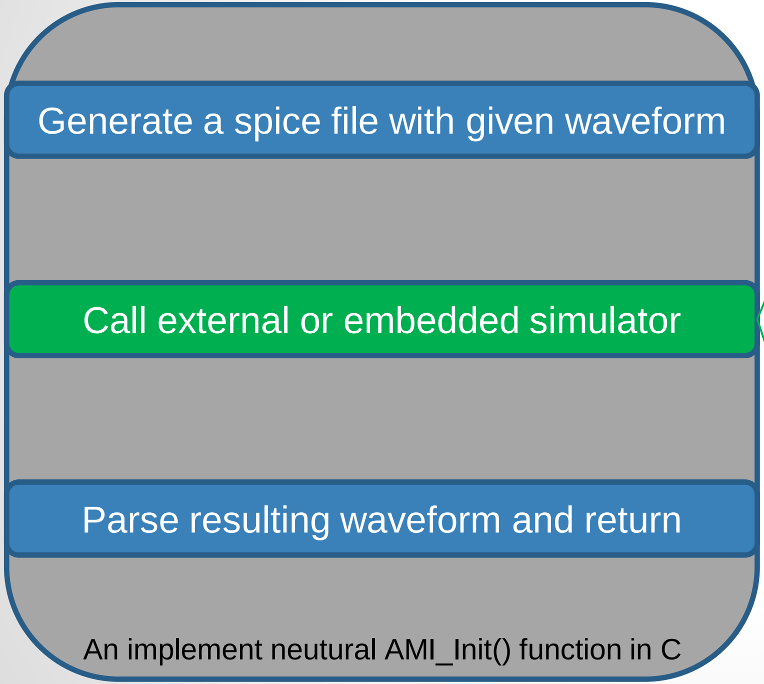


No need to code circuit behavior!

Modeling with Spice: Flow



Modeling with Spice: Example




```
A Spice AMI deck
.tran 1e-11 2.001e-08
.probe VOUT=par('V(OUTP)-V(OUTN)')
.probe VINP=par('V(INPP)-V(INPN)')
.option noinit acct
.option rshunt=1E12
.option method=gear
.option RELTOL=0.01
.option ABSTOL=1N VNTOL=1M
.option ITL4=500
* The following two lines are for HSpice compatibility
.OPTION POST_VERSION=9601
.OPTION PROBE POST
* INPUT
VINP INPP INPN PWL(
+ 0 0.5
+ 1e-11 0.5
.....
+ 1.998e-08 -0.499034
+ 1.999e-08 -0.499035
+ 2e-08 -0.499035)
* AMI Subckt
.INC D:\WorkDir\CTLE.sp
XAMI INPP INPN OUTP OUTN RC_CTLE  CTLE_R1=0.01234 CTLE_C1=5.678E-12
* High Z Load
RHIZ OUTP OUTN 1E6
.end
```

Modeling with Spice: Consideration

- Performance and distribuability:
  - Availability of device models?
  - Redistributable (license)?
  - Does model user need specific simulator?

- Consider open source simulator!
  - NgSpice, QUCS etc all supports API/Shared library [6]
  - The AMI model is basically a circuit simulator
    - Implement once, use many times!
  - Performance vs Accuracy

Summary:

- AMI model using scripts and spice circuit:
  - Doable! (Has been implemented! Example included.)
  - Can reduce AMI modeling time significantly
  - Can serve as an intermediate step toward full C/C++ implementation.
- Considerations:
  - Performance:
    - Not a concern if only AMI_Init is needed (called only once)
  - Model release:
    - Can the model be distributed and used easily?
  - A simple wrapper IBIS-AMI model is needed as a proxy.

References:

1. **High-speed Links Circuits and Systems** http://www.ece.tamu.edu/~spalermo/ecen720.html
2. **Simulating High-Speed Serial Channels with IBIS-AMI Models** http://literature.cdn.keysight.com/litweb/pdf/5990-9111EN.pdf?id=2095655
3. **IBIS V6.1 Spec. Section 10** http://ibis.org/ver6.1/
4. **AMI Analysis Using a Proxy Class** http://ibis.org/summits/feb17/
5. **Embedding python in another application** https://docs.python.org/3/extending/embedding.html
6. **NgSPice as a shared library** http://ngspice.sourceforge.net/shared.html

Q & A

EDA Expertise in Signal, Power Integrity & Simulation