Automated AMI Model Generation & Validation

José Luis Pino
Amolak Badesha
Manuel Luschas
Antonis Orphanou
Halil Civit

Asian IBIS Summit, Taipei, Taiwan November 12, 2010
(Presented previously at the IBIS Summits on June 15, 2010 and November 9, 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

0 months  AMI 101, Decipher Code
4 months  Early Model prototypes
8 months  First-model to Customer
12 months

Nightmare Begins
Typical AMI model generation flow...

<table>
<thead>
<tr>
<th>Jan</th>
<th>Feb</th>
<th>Mar</th>
<th>Apr</th>
<th>May</th>
<th>Jun</th>
<th>Jul</th>
<th>Aug</th>
<th>Sep</th>
<th>Oct</th>
<th>Nov</th>
<th>Dec</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matlab/C++ Model</td>
<td>Compile C++ code</td>
<td>C++ Code -&gt; AMI (.dll, .ami)</td>
<td>Channel Simulator Validation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Automated AMI model generation flow...

<table>
<thead>
<tr>
<th>Jan</th>
<th>Feb</th>
<th>Mar</th>
<th>Apr</th>
<th>May</th>
<th>Jun</th>
<th>Jul</th>
<th>Aug</th>
<th>Sep</th>
<th>Oct</th>
<th>Nov</th>
<th>Dec</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matlab/C++ Model</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Compile C++ code</td>
<td>Library of Common Building Blocks</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C++ Code -&gt; AMI (.dll, .ami)</td>
<td>-FIR/IIR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>-FFE/DFE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>-CDR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>-S-block</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>-Peaking, VGA etc.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Channel Simulator Validation</td>
<td>Automatic C++ Code Generation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Automatic AMI Generation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Automated AMI Flow</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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:
**Step-1:** Starting Architecture Design with Generic Model

Different blocks represent high-level TX architecture.

- **FIR/IIR filter**
- **Gain**
- **n-tap FFE**

Parameters:
- **Response Type:** Step Response
- **Time Step:** 1e-12 [sample_interval]
- **Gain:** 1 [Gain]
- **Coefficients:** 0; 0 [Taps]
- **Samples Per Bit:** 16 [Samples Per Bit]
More on FIR Filter...
How to bring in Spice or Measured data?

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

- Low Sampling Rate
- High Sampling Rate

FIR model should support “Arbitrary” Sampling Rate

Sampling Rate determines Simulation Accuracy
Step-2: Customize IP -> Bring in Math Lang or C++ Code

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

```
persistent dSamples;
persistent numSamples;
persistent taps;
if isempty(dSamples)
  % first time we hit this routine
  numSamples = length(Coefficients) * SamplesPerBit;
  dSamples = zeros(1, numSamples);
  dSamples(1) = input;
  taps = Coefficients';
else
  dSamples = [input, dSamples(1:numSamples-1)];
end
output = dSamples(1:SamplesPerBit:numSamples) * taps;
```
ESL flow: TX Modeling Example (3)

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

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

One-click AMI Code-generation
Step-4: Automatically Generated .ami and Visual-Studio project

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

- Generate square bit wave, including modeling impairments
- AMI TX
- CTLE & DFE AMI RX

Tyco Channel

123 123

1 1 0 1 0

B1 [PRBS(Data Flow Models)]
BitRate=5e+1GHz [BitRate]

P2 [JitterGenerator(Data Flow Models)]
BitRate=600Hz
DCD=0
Pj_Amplitude=0s
Pj_Frequency=1e+8Hz
RJ=0

Data3 [AMI_Tx_6]
Tap=1.0,0,1.0]
Gain=0.5

Data2 [AMI_CTE_Rx]
DFETaps=0.0,0,0,0 [0 0 0 0]

S1 [Sink(Data Flow Models)]
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)

*Note: EQ taps not optimized for maximum eye
Example #2
10.3125 Gb/s (10-GB Ethernet)

10.3125 Gb/s SerDes

Generate square bit wave, including modeling impairments

B1 [PRBS(Data Flow Models)]
BitRate=10.3125e9 [bit/second]

P1 [JitterGenerator(Data Flow Models)]
BitRate=10.3125e9 [bit/second]
DC[0]
PhaseAmplitude[0]
PhaseFrequency[1e+01]
PhaseRotation[0]
Gain[0.5]

AMI TX

Data3 [AMI_TX_10]
Taps=[1 0 0] [1 0 0]
Gain=0.5

SData

Molex 2006 Channel

FFE & DFE AMI Rx

Data2 [AMI_Rx]
DFETaps=[0 0 0] [0 0 0]
Alpha=0.001
NumberPrecursors=3
NumberPostcursors=3

123
RX Modeling
10.3125 Gb/s (10-GB Ethernet)
Example #2
10.3125 Gb/s (10-GB Ethernet)
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

Step Response from transistor simulation

FIR filter with Step Response Input

S1 {TimeResponseFIR@Data Flow Models} ResponseType=Step Response TimeStep=1e-12 [sample_interval]
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

Impulse Response derived from Scope Measurement

FIR filter with Impulse Response Input
TX Correlation Measured
emphasis #1: tap 0, 1, -0.2

Measured

AMI Model

<table>
<thead>
<tr>
<th>Measurement</th>
<th>Current</th>
<th>Minimum</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jitter [rms]</td>
<td>18</td>
<td>1.632 ps</td>
<td>1.632 ps</td>
</tr>
<tr>
<td>Jitter [p-p]</td>
<td>18</td>
<td>5.386 ps</td>
<td>5.386 ps</td>
</tr>
<tr>
<td>Rise Time</td>
<td>18</td>
<td>21.8 ps</td>
<td>21.8 ps</td>
</tr>
<tr>
<td>Fall Time</td>
<td>18</td>
<td>21.8 ps</td>
<td>21.8 ps</td>
</tr>
</tbody>
</table>
TX Correlation Measured

emphasis #2: tap 0, 1, -0.25

Measured

AMI Model

<table>
<thead>
<tr>
<th>Measurement</th>
<th>Current</th>
<th>Minimum</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jitter[rms]</td>
<td>1B</td>
<td>2.160 ps</td>
<td>2.154 ps</td>
</tr>
<tr>
<td>Rise Time</td>
<td>1B</td>
<td>20.4 ps</td>
<td>20.4 ps</td>
</tr>
<tr>
<td>Fall Time</td>
<td>1B</td>
<td>20.4 ps</td>
<td>20.4 ps</td>
</tr>
</tbody>
</table>
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*