**BUFFER ISSUE RESOLUTION DOCUMENT (BIRD)**

**BIRD NUMBER: (Draft 0)**

**ISSUE TITLE:** AMI Simulation Reference Flow Enhancement

**REQUESTOR:**  Fangyi Rao, Keysight Technologies, Inc.

Walter Katz, Signal Integrity Software, Inc.

Vladimir Dmitriev-Zdorov, Mentor Graphics, Inc.

Arpad Muranyi, Mentor Graphics, Inc.

Yunong Guan, Broadcom Corp.

Dong Yang, Broadcom Corp.

**DATE SUBMITTED:**

**DATE REVISED:**

**DATE ACCEPTED:**

**DEFINITION OF THE ISSUE:**

Current simulation reference flows does not properly handle the case where Tx has AMI\_Getwave and Rx does not. In particular, the Rx equalization filter can not be obtained by deconvolution when DFE is present.

In current simulation reference flows for channels with Redriver, the through channel impulse response sent to the downstream Rx AMI\_Init only represents the downstream channel. As a result, the optimization in the downstream Rx AMI\_Init can not take into account effects of the upstream channel loss and the Redriver Rx setting.

**SOLUTION REQUIREMENTS:**

The IBIS specification must meet these requirements:

Table : Solution Requirements

|  |  |
| --- | --- |
| Requirement | Notes |
| 1. For channels without Repeater, the case where the Tx has AMI\_GetWave and the Rx does not must be handled properly. |  |
| 1. For channels with Redrivers, the impulse response from the terminal Tx to the Rx must be available to the Rx AMI\_Init. |  |

**SUMMARY OF PROPOSED CHANGES:**

For review purposes, the proposed changes are summarized as follows:

Table : IBIS Keywords, Subparameters, AMI Reserved\_Parameters, and AMI functions Affected

|  |  |  |
| --- | --- | --- |
| Specification Item | New/Modified/Other | Notes |
| AMI simulation reference flows for channels without and with Repeaters are enhanced. | Modified |  |
| The impulse response matrix in the Rx AMI\_Init is extended. | Modified |  |
| The Reserved Parameter Init\_Supports\_Extended\_Impulse\_Matrix is added. | New |  |
| The Reserved Parameter Impulse\_Matrix\_Is\_Extended is added. | New |  |

**PROPOSED CHANGES:**

*Sections “10.2.2.3.1 STATISTICAL SIMULATION REFERENCE FLOW” and “10.2.2.3.2 TIME DOMAIN SIMULATION REFERENCE FLOW” on pages 177-179 of the existing IBIS 6.1 specification should be changed from:*

10.2.2.3.1 STATISTICAL SIMULATION REFERENCE FLOW

Step 1. The EDA tool obtains the impulse response for the analog channel. This represents the combined impulse response of the transmitter’s analog output, the channel and the receiver’s analog front end. The transmitter’s output or receiver’s input characteristics must not include any filtering effects, for example equalization, in this impulse response, although it may include any parasitics which are included in the Tx or Rx analog model.

Step 2. The output of Step 1 is presented to the Tx executable model file’s AMI\_Init function and the Tx AMI\_Init function is executed. The impulse response returned by the Tx AMI\_Init function is passed onto Step 3.

Step 3. The output of Step 2 is presented to the Rx executable model file’s AMI\_Init function and the Rx AMI\_Init function is executed. The impulse response returned by the Rx AMI\_Init function is passed onto Step 4.

Step 4. The EDA tool completes the rest of the simulation/analysis using the impulse response calculated in Step 3 by the Rx executable model file’s AMI\_Init function which is a complete representation of the behavior of a given [Algorithmic Model] combined with the channel.

10.2.2.3.2 TIME DOMAIN SIMULATION REFERENCE FLOW

Step 1. The EDA tool obtains the impulse response for the analog channel. This represents the combined impulse response of the transmitter’s analog output, the channel and the receiver's analog front end. The transmitter’s output or receiver’s input characteristics must not include any filtering effects, for example equalization, in this impulse response, although it may include any parasitics which are included in the Tx or Rx analog model.

Step 2. The output of Step 1 is presented to the Tx executable model file’s AMI\_Init function and the Tx AMI\_Init function is executed. The Tx AMI\_Init function may modify the impulse response or choose to leave it unchanged.

Step 3. The output of Step 2 is presented to the Rx executable model file’s AMI\_Init function and the Rx AMI\_Init function is executed. The Rx AMI\_Init function may modify the impulse response or choose to leave it unchanged. Under certain circumstances, for example when the Rx AMI\_Init function includes an optimization algorithm, the impulse response presented to the Rx AMI\_Init function must include the Tx equalization effects for the optimization to work correctly. However, when the Tx AMI model contains an AMI\_GetWave function that performs a similar or better equalization than the Tx AMI\_Init function, there is a possibility for “double-counting” the equalization effects in the Tx executable model file. To allow for such models to work correctly, the EDA tool can operate in one of several ways, two of which are documented here:

• not utilize the Tx AMI\_GetWave functionality, by treating the Tx AMI model as if the Tx GetWave\_Exists was False.

• use deconvolution to obtain the impulse response of the Rx filter. Since the AMI\_Init function contains a linear and time invariant algorithm, the Rx equalization can be represented as an impulse response. Since the output of the Rx AMI\_Init function (output of Step 3) is an impulse response modified by the Rx equalization (e.g., by convolving the input of the Rx AMI\_Init function with the impulse response of the Rx filter), the impulse response of the Rx filter can be obtained by deconvolving the output of Step 3 with the input presented to Step 3.

Note: The Rx executable model file writer should keep in mind that it is not guaranteed that the impulse response that is presented to the Rx AMI\_Init function will always include the effects of the Tx filter. Therefore the Rx AMI\_Init function may not be able to perform accurate optimization under all circumstances. For this reason, the parameters of the Rx AMI\_Init function should always default to valid values or have a mechanism to accept user-defined coefficients and allow the user to turn off any automatic optimization routines to ensure successful simulations.

Step 4. The EDA tool produces a digital stimulus waveform. A digital stimulus waveform is 0.5 when the stimulus is "high", -0.5 when the stimulus is "low", and may have a value between -0.5 and 0.5 such that transitions occur when the stimulus crosses 0.

Step 5. If Tx GetWave\_Exists is True the output of Step 4 is presented to the Tx executable model file’s AMI\_GetWave function and the Tx AMI\_GetWave function is executed. The output of the Tx AMI\_GetWave function is passed on to Step 6.

Step 6a. If Tx GetWave\_Exists is True and Rx GetWave\_Exists is True, the output of Step 5 is convolved with the output of Step 1 by the EDA tool and the result is passed on to Step 7.

Step 6b. If Tx GetWave\_Exists is False and Rx GetWave\_Exists is True, the output of Step 4 is convolved with the output of Step 2 by the EDA tool and the result is passed on to Step 7.

Step 6c. If Tx GetWave\_Exists is False and Rx GetWave\_Exists is False, the output of Step 4 is convolved with the output of Step 3 by the EDA tool and the result is passed on to Step 8.

Step 6d. If Tx GetWave\_Exists is True and Rx GetWave\_Exists is False, the output of Step 5 is convolved with the output of Step 1 and the Impulse Response of the Rx filter by the EDA tool andthe result is passed on to Step 8. (The Impulse Response of the Rx filter may be obtained by deconvolving the output of Step 3 by the input of Step 3).

Note: For the scenario where the Tx AMI\_Init function does NOT include equalization effects (i.e., does not modify the impulse response of the channel), Step 6d is functionally equivalent to simply convolving the output of Step 5 with the output of Step 3.

Step 7. If Rx GetWave\_Exists is True the output of Step 6 is presented to the Rx executable model file’s AMI\_GetWave function and the Rx AMI\_GetWave function is executed. The output of the Rx AMI\_GetWave function is passed on to Step 8.

Step 8. The output of Step 6c, 6d or 7 becomes the simulation waveform output at the Rx decision point. Step 7 optionally may also return clock ticks, which may be post-processed by the simulation tool or presented to the user as is.

Steps 4 through 8 can be called once or can be called multiple times to process the full analog waveform. Splitting up the full analog waveform into multiple calls reduces the memory requirements when doing long simulations, and allows AMI\_GetWave to return model status every so many bits. Once all blocks of the input waveform have been processed, Tx AMI\_Close and Rx AMI\_Close are called to perform any final processing and release allocated memory.

*… to:*

10.2.2.3.1 STATISTICAL SIMULATION REFERENCE FLOW

Step 1. The EDA tool obtains the impulse response for the analog channel. This represents the combined impulse response of the transmitter’s analog output, the channel and the receiver’s analog front end. The transmitter’s output or receiver’s input characteristics must not include any filtering effects, for example equalization, in this impulse response, although it may include any parasitics which are included in the Tx or Rx analog model.

Step 2. The output of Step 1 is presented to the Tx executable model file’s AMI\_Init function and the Tx AMI\_Init function is executed. The impulse response returned by the Tx AMI\_Init function is passed onto Step 3.

Step 3. One of the two steps described in Steps 3.1 and 3.2 is executed.

Step 3.1. The output of Step 2 is presented to the Rx executable model file’s AMI\_Init function, and the Rx AMI\_Init function is executed. The impulse response returned by the Rx AMI\_Init function is passed onto Step 4.

Step 3.2. Three impulse responses, denoted by h1, h2, and h3, are presented to the Rx executable model file’s AMI\_Init function. Both impulses h1 and h2 are the output of step 2. Impulse h3 is an empty placeholder for the Rx DFE impulse response. Impulse h1 is written in the first column of the impulse matrix. Impulses h2 and h3 are written in the first and second columns after the crosstalk impulse columns in the impulse matrix, respectively. The Rx AMI\_Init function is executed. The Rx AMI\_Init function may modify the impulse responses or choose to leave it unchanged. If the Rx AMI\_Init function modifies the impulse responses, it will write the impulse of Rx’s non-DFE part to h1, modify h2 to include both DFE and non-DFE parts of Rx, and write the impulse of the Rx DFE to h3. Note that the Rx AMI\_Init function must align the cursors in the output h3 with the convolution of the input h1 and the output h1. The output h2 returned by the Rx AMI\_Init function is passed onto Step 4.

Step 4. The EDA tool completes the rest of the simulation/analysis using the impulse response calculated in Step 3 by the Rx executable model file’s AMI\_Init function, which is a complete representation of the behavior of a given [Algorithmic Model] combined with the channel.

10.2.2.3.2 TIME DOMAIN SIMULATION REFERENCE FLOW

Step 1. The EDA tool obtains the impulse response for the analog channel. This represents the combined impulse response of the transmitter’s analog output, the channel and the receiver's analog front end. The transmitter’s output or receiver’s input characteristics must not include any filtering effects, for example equalization, in this impulse response, although it may include any parasitics which are included in the Tx or Rx analog model.

Step 2. The output of Step 1 is presented to the Tx executable model file’s AMI\_Init function and the Tx AMI\_Init function is executed. The Tx AMI\_Init function may modify the impulse response or choose to leave it unchanged.

Step 3. One of the two steps described in Steps 3.1 and 3.2 is executed.

Step 3.1. The output of Step 2 is presented to the Rx executable model file’s AMI\_Init function, and the Rx AMI\_Init function is executed. The Rx AMI\_Init function may modify the impulse response or choose to leave it unchanged.

Under certain circumstances, for example when the Rx AMI\_Init function includes an optimization algorithm, the impulse response presented to the Rx AMI\_Init function must include the Tx equalization effects for the optimization to work correctly. However, when the Tx AMI model contains an AMI\_GetWave function that performs a similar or better equalization than the Tx AMI\_Init function, there is a possibility for “double-counting” the equalization effects in the Tx executable model file. To allow for such models to work correctly, the EDA tool can operate in one of several ways, two of which are documented here:

* not utilize the Tx AMI\_GetWave functionality, by treating the Tx AMI model as if the Tx GetWave\_Exists was False.
* use deconvolution to obtain the impulse response of the Rx filter. Since the AMI\_Init function contains a linear and time invariant algorithm, the Rx equalization can be represented as an impulse response. Since the output of the Rx AMI\_Init function (output of Step 3.1) is an impulse response modified by the Rx equalization (e.g., by convolving the input of the Rx AMI\_Init function with the impulse response of the Rx filter), the impulse response of the Rx filter can be obtained by deconvolving the output of Step 3.1 with the input presented to Step 3.1.

Step 3.2. Three impulse responses, denoted by h1, h2, and h3, are presented to the Rx executable model file’s AMI\_Init function. Impulse h1 is the output of Step 1 if Tx GetWave\_Exists is True or the output of Step 2 if Tx GetWave\_Exists is False. Impulse h2 is the output of Step 2, and h3 an empty placeholder for the Rx DFE impulse response. Impulse h1 is written in the first column of the impulse matrix. Impulses h2 and h3 are written in the first and second columns after the crosstalk impulse columns in the impulse matrix, respectively. The Rx AMI\_Init function is executed. The Rx AMI\_Init function may modify the impulse responses or choose to leave it unchanged. If the Rx AMI\_Init function modifies the impulse responses, it will write the impulse of Rx’s non-DFE part to h1, modify h2 to include both DFE and non-DFE parts of Rx, and write the impulse of the Rx DFE to h3. Note that the Rx AMI\_Init function must align the cursors in the output h3 with the convolution of the input h1 and the output h1.

Note: The Rx executable model file writer should keep in mind that it is not guaranteed that the impulse response that is presented to the Rx AMI\_Init function will always include the effects of the Tx filter. Therefore the Rx AMI\_Init function may not be able to perform accurate optimization under all circumstances. For this reason, the parameters of the Rx AMI\_Init function should always default to valid values or have a mechanism to accept user-defined coefficients and allow the user to turn off any automatic optimization routines to ensure successful simulations.

Step 4. The EDA tool produces a digital stimulus waveform. A digital stimulus waveform is 0.5 when the stimulus is "high", -0.5 when the stimulus is "low", and may have a value between -0.5 and 0.5 such that transitions occur when the stimulus crosses 0.

Step 5. One of the six steps described in Steps 5.1, 5.2.1, 5.2.2, 5.3, 5.4.1 and 5.4.2 is executed.

Step 5.1. If Tx GetWave\_Exists is False and Rx GetWave\_Exists is True, the output of Step 4 is convolved with the output of Step 2 by the EDA tool and the result is presented to the Rx executable model file’s AMI\_GetWave function and the Rx AMI\_GetWave function is executed.

Step 5.2.1. If Tx GetWave\_Exists is False and Rx GetWave\_Exists is False and if Step 3.1 is executed, the output of Step 4 is convolved with the output of Step 3.1 by the EDA tool.

Step 5.2.2. If Tx GetWave\_Exists is False and Rx GetWave\_Exists is False and if Step 3.2 is executed, the output of Step 4 is convolved with the output h2 of Step 3.2 by the EDA tool.

Step 5.3. If Tx GetWave\_Exists is True and Rx GetWave\_Exists is True, the output of Step 4 is presented to the Tx executable model file’s AMI\_GetWave function, and the Tx AMI\_GetWave function is executed. The output of the Tx AMI\_GetWave function is convolved with the output of Step 1 by the EDA tool. The result is presented to the Rx executable model file’s AMI\_GetWave function, and the Rx AMI\_GetWave function is executed.

Step 5.4.1. If Tx GetWave\_Exists is True and Rx GetWave\_Exists is False and if Step 3.1 is executed, the output of Step 4 is presented to the Tx executable model file’s AMI\_GetWave function, and the Tx AMI\_GetWave function is executed. The output of the Tx AMI\_GetWave function is convolved with the output of Step 1 and the Impulse Response of the Rx filter by the EDA tool. (The Impulse Response of the Rx filter may be obtained by deconvolving the output of Step 3.1 by the input of Step 3.1).

Note: For the scenario where the Tx AMI\_Init function does NOT include equalization effects (i.e., does not modify the impulse response of the channel), Step 5.4.1 is functionally equivalent to simply convolving the output of Tx AMI\_GetWave with the output of Step 3.1.

Step 5.4.2. If Tx GetWave\_Exists is True and Rx GetWave\_Exists is False and if Step 3.2 is executed, the following three steps, 5.4.2.1, 5.4.2.2 and 5.4.2.3, are executed.

Step 5.4.2.1. The output of Step 4 is presented to the Tx executable model file’s AMI\_GetWave function, and the Tx AMI\_GetWave function is executed. The output of the Tx AMI\_GetWave function is convolved with the input h1 and the output h1 of Step 3.2 by the EDA tool.

Step 5.4.2.2. The EDA tool shifts the output of Step 4 to align it with the Tx AMI\_GetWave output and convolve the resulting waveform with the output h3 of Step 3.2.

Step 5.4.2.3. The EDA tool sums outputs of Steps 5.4.2.1 and 5.4.2.2.

Step 6. The output of Step 5 becomes the simulation waveform output at the Rx decision point. Rx AMI\_GetWave called in step 5 optionally may also return clock ticks, which may be post-processed by the simulation tool or presented to the user as is.

Steps 4 through 6 can be called once or can be called multiple times to process the full analog waveform. Splitting up the full analog waveform into multiple calls reduces the memory requirements when doing long simulations, and allows AMI\_GetWave to return model status every so many bits. Once all blocks of the input waveform have been processed, Tx AMI\_Close and Rx AMI\_Close are called to perform any final processing and release allocated memory.

*Repeater simulation flows on pages 243 and 244 of the existing IBIS 6.1 specification should be changed from:*

The time domain simulation flow for a Repeater link shown in Figure 40 is defined below.

Here Tx1 denotes the Repeater upstream channel (channel 1) Tx AMI model (including analog and algorithmic models), Rx1 the Repeater Rx AMI model (including analog and algorithmic models), Tx2 the Repeater Tx AMI model (including analog and algorithmic models) and Rx2 the Repeater downstream channel (channel 2) Rx AMI model (including analog and algorithmic models).

Step 1. The EDA tool obtains the impulse response of the upstream analog channel, which represents the combined impulse response of Tx1’s analog model, physical channel 1, and Rx1’s analog model.

Step 2. The output of step 1 is presented to Tx1’s AMI\_Init function and Tx1’s AMI\_Init function is executed.

Step 3. The output of step 2 is presented to Rx1’s AMI\_Init function and Rx1’s AMI\_Init function is executed.

Step 4. The EDA tool obtains the impulse response of the downstream analog channel, which represents the combined impulse response of Tx2’s analog model, physical channel 2, and Rx2’s analog model.

Step 5. The output of step 4 is presented to Tx2’s AMI\_Init function and Tx2’s AMI\_Init function is executed.

Step 6. The output of step 5 is presented to Rx2’s AMI\_Init function and Rx2’s AMI\_Init function is executed.

Step 7. The EDA tool performs simulation on the upstream channel, which consists of Tx1, physical channel 1, and Rx1, according to the AMI flow defined in the specification for channels without Repeaters.

Step 8a. Redriver: The EDA tool uses the signal waveform at the output end of Rx1’s algorithmic model in step 7, regardless whether Rx1’s AMI\_GetWave exists or not, as the stimulus of Tx2’s algorithmic model, regardless whether Tx2’s AMI\_GetWave exists or not, and performs simulation on the downstream channel, which consists of Tx2, physical channel 2 and Rx2, according to the AMI flow defined in the spec for channels without Redrivers.

Step 8b. Retimer: The EDA tool samples the output waveform of Retimer Rx AMI\_GetWave at ½ UI after each clock tick returned by the function, generates a digital stimulus as the input to Tx2’s algorithmic model, regardless whether Tx2’s AMI\_GetWave exists or not, and performs simulation on the downstream channel, which consists of Tx2, physical channel 2 and Rx2, according to the AMI flow defined in the spec for channels without Redriver. The logic level of the digital stimulus is 1 if sampled value >= Rx1’s Rx\_Receiver\_Sensitivity and 0 if sampled value <= −Rx1’s Rx\_Receiver\_Sensitivity. If –Rx1’s Rx\_Receiver\_Sensitivity < sampled value < Rx1’s Rx\_Reciver\_Sensitivity, the logic level is unchanged from the previous bit. The digital stimulus have values of -½ volt for logic 0 and +½ volt for logic 1.

Step 9. The EDA tool calls the AMI\_Close function of each algorithmic model in Tx1, Rx1, Tx2 and Rx2.

Since the Redriver output signal is driven continuously by the input analog signal and does not have a sampling latch, clock times, if returned by a Redriver model, jitter parameters and the Rx\_Noise parameter specified in Redriver .ami files are ignored by the EDA tool. Since the Retimer output signal is driven by a digital stimulus as described above in step 8b, jitter and noise parameters specified in Retimer .ami files are applied according to the specification for channels without Repeaters.

The statistical simulation flow for a Repeater link shown in Fig. 40 is defined below.

Step 1. The EDA tool obtains the impulse response of the upstream analog channel, which represents the combined impulse response of Tx1’s analog model, physical channel 1, and Rx1’s analog model.

Step 2. The output of step 1 is presented to the Tx1’s AMI\_Init function and Tx1’s AMI\_Init function is executed.

Step 3. The output of step 2 is presented to the Rx1’s AMI\_Init function and the Rx1’s AMI\_Init function is executed.

Step 4. The EDA tool obtains the impulse response of the downstream analog channel, which represents the combined impulse response of Tx2’s analog model, physical channel 2, and Rx2’s analog model.

Step 5. The output of step 4 is presented to Tx2’s AMI\_Init function and Tx2’s AMI\_Init function is executed.

Step 6. The output of step 5 is presented to Rx2’s AMI\_Init function and Rx2’s AMI\_Init function is executed.

Step 7a. Redriver: The EDA tool convolves impulse responses returned by Rx1’s AMI\_Init in step 3 and by Rx2’s AMI\_Init in step 6 to obtained the full channel impulse response and uses it to perform statistical simulation.

Step 7b. Retimer: The EDA tool uses the impulse responses returned by Rx1’s AMI\_Init in step 3 to perform a statistical simulation of channel 1. The EDA tool uses the impulse responses returned by Rx2’s AMI\_Init in step 6 to perform a statistical simulation of channel 2.

*… to:*

The time domain simulation flow for a Redriver link shown in Figure 40 is defined below.

Here Tx1 denotes the Redriver upstream channel (channel 1) Tx AMI model (including analog and algorithmic models), Rx1 the Redriver Rx AMI model (including analog and algorithmic models), Tx2 the Redriver Tx AMI model (including analog and algorithmic models) and Rx2 the Redriver downstream channel (channel 2) Rx AMI model (including analog and algorithmic models).

Step 1. The EDA tool obtains the impulse response of the upstream analog channel, which represents the combined impulse response of Tx1’s analog model, physical channel 1, and Rx1’s analog model.

Step 2. The output of step 1 is presented to Tx1’s AMI\_Init function and Tx1’s AMI\_Init function is executed.

Step 3. One of the two steps described in Steps 3.1 and 3.2 is executed.

Step 3.1. The output of Step 2 is presented to the Rx1 executable model file’s AMI\_Init function, and the Rx1 AMI\_Init function is executed.

Step 3.2. Three impulse responses, denoted by hu1, hu2, and hu3, are presented to the Rx1 executable model file’s AMI\_Init function. Impulse hu1 is the output of Step 1 if Tx1 GetWave\_Exists is True or the output of Step 2 if Tx1 GetWave\_Exists is False. Impulse hu2 is the output of Step 2, and hu3 an empty placeholder for the Rx1 DFE impulse response. Impulse hu1 is written in the first column of the impulse matrix. Impulses hu2 and hu3 are written in the first and second columns after the crosstalk impulse columns in the impulse matrix, respectively. The Rx1 AMI\_Init function is executed. If the Rx1 AMI\_Init function modifies the impulse responses, it will write the impulse of the Rx1’s non-DFE part to hu1, modify hu2 to include both DFE and non-DFE parts of Rx1, and write the impulse of the Rx1 DFE to hu3. Note that Rx1 AMI\_Init function must align the cursors in the output hu3 with the convolution of the input hu1 and the output hu1.

Step 4. The EDA tool obtains the impulse response of the downstream analog channel, which represents the combined impulse response of Tx2’s analog model, physical channel 2, and Rx2’s analog model.

Step 5. The output of step 4 is presented to Tx2’s AMI\_Init function and Tx2’s AMI\_Init function is executed.

Step 6. One of the two steps described in Steps 6.1 and 6.2 is executed.

Step 6.1. The output of Step 5 is presented to the Rx2 executable model file’s AMI\_Init function, and the Rx2 AMI\_Init function is executed.

Step 6.2. Three impulse responses, denoted by hd1, hd2 and hd3 respectively, are presented to the Rx2 executable model file’s AMI\_Init function. Impulse hd1 is the output of Step 4 if Tx2\_ GetWave\_Exists is True or the output of Step 5 if Tx2 GetWave\_Exists is False. Impulse hd2 is the convolution of the output of Step 5 with the output of Step 3.1 if Step 3.1 is executed or with the output hu2 of Step 3.2 if Step 3.2 is executed. Impulse hd3 is an empty placeholder for the Rx2 DFE impulse response. Impulse hd1 is written in the first column of the impulse matrix. Impulses hd2 and hd3 are written in the first and second columns after the crosstalk impulse columns in the impulse matrix, respectively. The Rx2 AMI\_Init function is executed. If the Rx2 AMI\_Init function modifies the impulse responses, it will write the impulse of Rx2’s non-DFE part to hd1, modify hd2 to include both DFE and non-DFE parts of Rx2, and write the impulse of the Rx2 DFE to hd3. Note that the Rx2 AMI\_Init function must align the cursors in the output hd3 with the convolution of the input hd1 and the output hd1.

Step 7. The EDA tool performs simulation on the upstream channel, which consists of Tx1, physical channel 1, and Rx1, according to the AMI flow defined in the specification for channels without Repeaters.

Step 8. One of the six steps described in Steps 8.1, 8.2.1, 8.2.2, 8.3, 8.4.1 and 8.4.2 is executed.

Step 8.1. If Tx2 GetWave\_Exists is False and Rx2 GetWave\_Exists is True, the output of Step 7 is convolved with the output of Step 5 by the EDA tool and the result is presented to the Rx2 executable model file’s AMI\_GetWave function and the Rx2 AMI\_GetWave function is executed.

Step 8.2.1. If Tx2 GetWave\_Exists is False and Rx2 GetWave\_Exists is False and if Step 6.1 is executed, the output of Step 7 is convolved with the output of Step 6.1 by the EDA tool.

Step 8.2.2. If Tx2 GetWave\_Exists is False and Rx2 GetWave\_Exists is False and if Step 6.2 is executed, the following three steps, 8.2.2.1, 8.2.2.2 and 8.2.2.3, are executed.

Step 8.2.2.1. The output of Step 7 is convolved with the input hd1 and the output hd1 of Step 6.2 by the EDA tool.

Step 8.2.2.2. The EDA tool shifts the digital stimulus waveform at the Tx1 input generated in Step 7 to align it with the output of step 7 and convolve the resulting waveform with the output hd3 of Step 6.2.

Step 8.2.2.3. The EDA tool sums outputs of Steps 8.2.2.1 and 8.2.2.2.

Step 8.3. If Tx2 GetWave\_Exists is True and Rx2 GetWave\_Exists is True, the output of Step 7 is presented to the Tx2 executable model file’s AMI\_GetWave function, and the Tx2 AMI\_GetWave function is executed. The output of the Tx2 AMI\_GetWave function is convolved with the output of Step 4 by the EDA tool. The result is presented to the Rx2 executable model file’s AMI\_GetWave function, and the Rx2 AMI\_GetWave function is executed.

Step 8.4.1. If Tx2 GetWave\_Exists is True and Rx2 GetWave\_Exists is False and if Step 6.1 is executed, the output of Step 7 is presented to the Tx2 executable model file’s AMI\_GetWave function, and the Tx2 AMI\_GetWave function is executed. The output of the Tx2 AMI\_GetWave function is convolved with the output of Step 4 and the Impulse Response of the Rx2 filter by the EDA tool. (The Impulse Response of the Rx2 filter may be obtained by deconvolving the output of Step 6.1 by the input of Step 6.1).

Note: For the scenario where the Tx2 AMI\_Init function does NOT include equalization effects (i.e., does not modify the impulse response of the channel), Step 8.4.1 is functionally equivalent to simply convolving the output of Tx2 AMI\_GetWave with the output of Step 6.1.

Step 8.4.2. If Tx GetWave\_Exists is True and Rx GetWave\_Exists is False and if Step 6.2 is executed, the following three steps, 8.4.2.1, 8.4.2.2 and 8.4.2.3, are executed.

Step 8.4.2.1. The output of Step 7 is presented to the Tx2 executable model file’s AMI\_GetWave function, and the Tx2 AMI\_GetWave function is executed. The output of the Tx2 AMI\_GetWave function is convolved with the input hd1 and the output hd1 of Step 6.2 by the EDA tool.

Step 8.4.2.2. The EDA tool shifts the digital stimulus waveform at the Tx1 input generated in Step 7 to align it with the Tx2 AMI\_GetWave output and convolve the resulting waveform with the output hd3 of Step 6.2.

Step 8.4.2.3. The EDA tool sums outputs of Steps 8.4.2.1 and 8.4.2.2.

Step 9. The output of Step 8 becomes the simulation waveform output at the Rx2 decision point. Rx2 AMI\_GetWave called in step 8 optionally may also return clock ticks, which may be post-processed by the EDA tool or presented to the user as is.

Steps 7 and 8 can be called once or can be called multiple times to process the full analog waveform. Splitting up the full analog waveform into multiple calls reduces the memory requirements when doing long simulations, and allows AMI\_GetWave to return model status every so many bits. Once all blocks of the input waveform have been processed, the EDA tool calls the AMI\_Close function of each algorithmic model in Tx1, Rx1, Tx2 and Rx2.

Since the Redriver output signal is driven continuously by the input analog signal and does not have a sampling latch, clock times, if returned by a Redriver model, jitter parameters and the Rx\_Noise parameter specified in Redriver .ami files are ignored by the EDA tool.

The statistical simulation flow for a Redriver link shown in Fig. 40 is defined below.

Step 1. The EDA tool obtains the impulse response of the upstream analog channel, which represents the combined impulse response of Tx1’s analog model, physical channel 1, and Rx1’s analog model.

Step 2. The output of step 1 is presented to Tx1’s AMI\_Init function and Tx1’s AMI\_Init function is executed.

Step 3. One of the two steps described in Steps 3.1 and 3.2 is executed.

Step 3.1. The output of Step 2 is presented to the Rx1 executable model file’s AMI\_Init function, and the Rx1 AMI\_Init function is executed.

Step 3.2. Three impulse responses, denoted by hu1, hu2, and hu3, are presented to the Rx1 executable model file’s AMI\_Init function. Both impulses hu1 and hu2 are the output of step 2. Impulse hu3 is an empty placeholder for the Rx1 DFE impulse response. Impulse hu1 is written in the first column of the impulse matrix. Impulses hu2 and hu3 are written in the first and second columns after the crosstalk impulse columns in the impulse matrix, respectively. The Rx1 AMI\_Init function is executed. If the Rx1 AMI\_Init function modifies the impulse responses, it will write the impulse of the Rx1’s non-DFE part to hu1, modify hu2 to include both DFE and non-DFE parts of Rx1, and write the impulse of the Rx1 DFE to hu3. Note that Rx1 AMI\_Init function must align the cursors in the output hu3 with the convolution of the input hu1 and the output hu1.

Step 4. The EDA tool obtains the impulse response of the downstream analog channel, which represents the combined impulse response of Tx2’s analog model, physical channel 2, and Rx2’s analog model.

Step 5. The output of step 4 is presented to Tx2’s AMI\_Init function and Tx2’s AMI\_Init function is executed.

Step 6. One of the two steps described in Steps 6.1 and 6.2 is executed.

Step 6.1. The output of Step 5 is presented to the Rx2 executable model file’s AMI\_Init function, and the Rx2 AMI\_Init function is executed.

Step 6.2. Three impulse responses, denoted by hd1, hd2 and hd3 respectively, are presented to the Rx2 executable model file’s AMI\_Init function. Impulse hd1 is the output of Step 5. Impulse hd2 is the convolution of the output of Step 5 with the output of Step 3.1 if Step 3.1 is executed or with the output hu2 of Step 3.2 if Step 3.2 is executed. Impulse hd3 is an empty placeholder for the Rx2 DFE impulse response. Impulse hd1 is written in the first column of the impulse matrix. Impulses hd2 and hd3 are written in the first and second columns after the crosstalk impulse columns in the impulse matrix, respectively. The Rx2 AMI\_Init function is executed. If the Rx2 AMI\_Init function modifies the impulse responses, it will write the impulse of Rx2’s non-DFE part to hd1, modify hd2 to include both DFE and non-DFE parts of Rx2, and write the impulse of the Rx2 DFE to hd3. Note that the Rx2 AMI\_Init function must align the cursors in the output hd3 with the convolution of the input hd1 and the output hd1.

Step 7.1. If Steps 6.1 and 3.1 are executed, the EDA tool convolves impulse responses returned by Rx1’s AMI\_Init in Step 3.1 and by Rx2’s AMI\_Init in Step 6.1 to obtain the full channel impulse response and uses it to perform statistical simulation.

Step 7.2. If Steps 6.1 and 3.2 are executed, the EDA tool convolves the output hu2 of Rx1’s AMI\_Init in Step 3.2 and the impulse response returned by Rx2’s AMI\_Init in Step 6.1 to obtain the full channel impulse response and uses it to perform statistical simulation.

Step 7.3. If Step 6.2 is executed, the EDA uses the output hd2 of Rx2’s AMI\_Init in Step 6.2 to perform statistical simulation.

The time domain simulation flow for a Retimer link shown in Figure 40 is defined below.

Here Tx1 denotes the Retimer upstream channel (channel 1) Tx AMI model (including analog and algorithmic models), Rx1 the Retimer Rx AMI model (including analog and algorithmic models), Tx2 the Retimer Tx AMI model (including analog and algorithmic models) and Rx2 the Retimer downstream channel (channel 2) Rx AMI model (including analog and algorithmic models).

Step 1. The EDA tool obtains the impulse response of the upstream analog channel, which represents the combined impulse response of Tx1’s analog model, physical channel 1, and Rx1’s analog model.

Step 2. The output of step 1 is presented to Tx1’s AMI\_Init function and Tx1’s AMI\_Init function is executed.

Step 3. One of the two steps described in Steps 3.1 and 3.2 is executed.

Step 3.1. The output of Step 2 is presented to the Rx1 executable model file’s AMI\_Init function, and the Rx1 AMI\_Init function is executed.

Step 3.2. Three impulse responses, denoted by hu1, hu2, and hu3, are presented to the Rx1 executable model file’s AMI\_Init function. Impulse hu1 is the output of Step 1 if Tx1 GetWave\_Exists is True or the output of Step 2 if Tx1 GetWave\_Exists is False. Impulse hu2 is the output of Step 2, and hu3 an empty placeholder for the Rx1 DFE impulse response. Impulse hu1 is written in the first column of the impulse matrix. Impulses hu2 and hu3 are written in the first and second columns after the crosstalk impulse columns in the impulse matrix, respectively. The Rx1 AMI\_Init function is executed. If the Rx1 AMI\_Init function modifies the impulse responses, it will write the impulse of the Rx1’s non-DFE part to hu1, modify hu2 to include both DFE and non-DFE parts of Rx1, and write the impulse of the Rx1 DFE to hu3. Note that Rx1 AMI\_Init function must align the cursors in the output hu3 with the convolution of the input hu1 and the output hu1.

Step 4. The EDA tool obtains the impulse response of the downstream analog channel, which represents the combined impulse response of Tx2’s analog model, physical channel 2, and Rx2’s analog model.

Step 5. The output of step 4 is presented to Tx2’s AMI\_Init function and Tx2’s AMI\_Init function is executed.

Step 6. One of the two steps described in Steps 6.1 and 6.2 is executed.

Step 6.1. The output of Step 5 is presented to the Rx2 executable model file’s AMI\_Init function, and the Rx2 AMI\_Init function is executed.

Step 6.2. Three impulse responses, denoted by hd1, hd2, and hd3, are presented to the Rx2 executable model file’s AMI\_Init function. Impulse hd1 is the output of Step 4 if Tx2 GetWave\_Exists is True or the output of Step 5 if Tx2 GetWave\_Exists is False. Impulse hd2 is the output of Step 5, and hd3 an empty placeholder for the Rx2 DFE impulse response. Impulse hd1 is written in the first column of the impulse matrix. Impulses hd2 and hd3 are written in the first and second columns after the crosstalk impulse columns in the impulse matrix, respectively. The Rx2 AMI\_Init function is executed. If the Rx2 AMI\_Init function modifies the impulse responses, it will write the impulse of the Rx2’s non-DFE part to hd1, modify hd2 to include both DFE and non-DFE parts of Rx2, and write the impulse of the Rx2 DFE to hd3. Note that Rx2 AMI\_Init function must align the cursors in the output hd3 with the convolution of the input hd1 and the output hd1.

Step 7. The EDA tool performs simulation on the upstream channel, which consists of Tx1, physical channel 1, and Rx1, according to the AMI flow defined in the specification for channels without Repeaters.

Step 8. The EDA tool samples the output waveform of Retimer Rx AMI\_GetWave at ½ UI after each clock tick returned by the function, generates a digital stimulus as the input to Tx2’s algorithmic model, regardless whether Tx2’s AMI\_GetWave exists or not, and performs simulation on the downstream channel, which consists of Tx2, physical channel 2 and Rx2, according to the AMI flow defined in the specification for channels without Repeater. The logic level of the digital stimulus is 1 if sampled value >= Rx1’s Rx\_Receiver\_Sensitivity and 0 if sampled value <= -Rx1’s Rx\_Receiver\_Sensitivity. If –Rx1’s Rx\_Receiver\_Sensitivity < sampled value < Rx1’s Rx\_Reciver\_Sensitivity, the logic level is unchanged from the previous bit. The digital stimulus have values of -½ volt for logic 0 and +½ volt for logic 1.

Step 9. The output of Step 8 becomes the simulation waveform output at the Rx2 decision point. Rx2 AMI\_GetWave called in step 8 optionally may also return clock ticks, which may be post-processed by the EDA tool or presented to the user as is.

Steps 7 and 8 can be called once or can be called multiple times to process the full analog waveform. Splitting up the full analog waveform into multiple calls reduces the memory requirements when doing long simulations, and allows AMI\_GetWave to return model status every so many bits. Once all blocks of the input waveform have been processed, the EDA tool calls the AMI\_Close function of each algorithmic model in Tx1, Rx1, Tx2 and Rx2.

Since the Retimer output signal is driven by a digital stimulus as described above in step 8, jitter and noise parameters specified in Retimer .ami files are applied according to the specification for channels without Repeaters.

The statistical simulation flow for a Retimer link shown in Fig. 40 is defined below.

Step 1. The EDA tool obtains the impulse response of the upstream analog channel, which represents the combined impulse response of Tx1’s analog model, physical channel 1, and Rx1’s analog model.

Step 2. The output of step 1 is presented to Tx1’s AMI\_Init function and Tx1’s AMI\_Init function is executed.

Step 3. One of the two steps described in Steps 3.1 and 3.2 is executed.

Step 3.1. The output of Step 2 is presented to the Rx1 executable model file’s AMI\_Init function, and the Rx1 AMI\_Init function is executed.

Step 3.2. Three impulse responses, denoted by hu1, hu2, and hu3, are presented to the Rx1 executable model file’s AMI\_Init function. Both impulses hu1 and hu2 are the output of step 2. Impulse hu3 is an empty placeholder for the Rx1 DFE impulse response. Impulse hu1 is written in the first column of the impulse matrix. Impulses hu2 and hu3 are written in the first and second columns after the crosstalk impulse columns in the impulse matrix, respectively. The Rx1 AMI\_Init function is executed. If the Rx1 AMI\_Init function modifies the impulse responses, it will write the impulse of the Rx1’s non-DFE part to hu1, modify hu2 to include both DFE and non-DFE parts of Rx1, and write the impulse of the Rx1 DFE to hu3. Note that Rx1 AMI\_Init function must align the cursors in the output hu3 with the convolution of the input hu1 and the output hu1.

Step 4. The EDA tool obtains the impulse response of the downstream analog channel, which represents the combined impulse response of Tx2’s analog model, physical channel 2, and Rx2’s analog model.

Step 5. The output of step 4 is presented to Tx2’s AMI\_Init function and Tx2’s AMI\_Init function is executed.

Step 6. One of the two steps described in Steps 6.1 and 6.2 is executed.

Step 6.1. The output of Step 5 is presented to the Rx2 executable model file’s AMI\_Init function, and the Rx2 AMI\_Init function is executed.

Step 6.2. Three impulse responses, denoted by hd1, hd2 and hd3 respectively, are presented to the Rx2 executable model file’s AMI\_Init function. Both impulses hd1 and hd2 are the output of Step 5. Impulse hd3 is an empty placeholder for the Rx2 DFE impulse response. Impulse hd1 is written in the first column of the impulse matrix. Impulses hd2 and hd3 are written in the first and second columns after the crosstalk impulse columns in the impulse matrix, respectively. The Rx2 AMI\_Init function is executed. If the Rx2 AMI\_Init function modifies the impulse responses, it will write the impulse of Rx2’s non-DFE part to hd1, modify hd2 to include both DFE and non-DFE parts of Rx2, and write the impulse of the Rx2 DFE to hd3. Note that the Rx2 AMI\_Init function must align the cursors in the output hd3 with the convolution of the input hd1 and the output hd1.

Step 7.1. If Step 3.1 is executed, the EDA tool uses the impulse responses returned by Rx1’s AMI\_Init in Step 3.1 to perform a statistical simulation of channel 1.

Step 7.2. If Step 3.2 is executed, the EDA tool uses the output hu2 of Rx1’s AMI\_Init in Step 3.2 to perform a statistical simulation of channel 1.

Step 8.1. If Step 6.1 is executed, the EDA tool uses the impulse responses returned by Rx2’s AMI\_Init in Step 6.1 to perform a statistical simulation of channel 2.

Step 8.2. If Step 6.2 is executed, the EDA tool uses the output hd2 of Rx2’s AMI\_Init in Step 6.2 to perform a statistical simulation of channel 2.

*The following paragraph should be added before the last paragraph of the “impulse\_matrix” section on page 183 of the existing IBIS 6.1 specification.*

As described in the simulation flows, the impulse matrix presented to the Rx AMI\_Init function can be extended by two columns. In that case, the first extra column after the crosstalk impulse responses is the impulse response of this Rx’s through channel from the upstream terminal Tx or the nearest upstream retimer Tx to the Rx’s input node. The second extra column after the crosstalk impulse responses is the placeholder for the Rx’s DFE impulse response.

*Two new Reserved Parameter should be added after the Model\_Name parameter definition section on page 207 of the existing IBIS 6.1 specification.*

*Parameter:* **Init\_Supports\_Extended\_Impulse\_Matrix**

*Required:* No

*Direction:* Rx

*Descriptors*:

Usage: Info

Type: Boolean

Format: Value

Default: <Boolean\_literal>

Description:<string>

*Definition:* This parameter is optional and tells the EDA tool whether the Rx AMI\_Init supports the extended impulse response matrix. If not specified in the .ami file, this parameter is defaulted to “False”.

*Usage Rules:* If the value of this parameter is “True”, the Rx AMI\_Init must also support the original unextended impulse response matrix.

*Examples:*

(Init\_Supports\_Extended\_Impulse\_Matrix (Usage Info)(Type Boolean)(Value True)(Description “Rx Init supports both extended and unextended impulse matrices”))

*Parameter:* **Impulse\_Matrix\_Is\_Extended**

*Required:* No

*Direction:* Rx

*Descriptors*:

Usage: In

Type: Boolean

Format: Value

Default: <Boolean\_literal>

Description:<string>

*Definition:* The EDA tool is responsible to set the value of this parameters in the AMI\_parameters\_in string of the Rx AMI\_Init call to tell the Rx DLL whether the impulse response matrix presented is extended. This parameter is optional. If the EDA tool does not specify this parameter in the AMI\_parameters\_in string, the parameter is defaulted to “False”, and the Rx AMI\_Init function must operate accordingly.

*Usage Rules:* If this parameter is defined in the .ami file, its value specified in the file is ignored by the EDA tool.

*The following two rows below the header row should be added to Table 18 on page 207 of the existing IBIS 6.1 specification.*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Reserved Parameter** | **General Rules** | | **Alloweable Usage** | | | | |
| **Required** | **Default** | **Info** | **In** | **Out** | **Def** | **InOut** |
| Init\_Supports\_Extended\_Impulse\_Matrix | No | False | X |  |  |  |  |
| Impulse\_Matrix\_Is\_Extended | No | False |  | X |  |  |  |

*The following two rows below the header row should be added to Table 19 on page 208 of the existing IBIS 6.1 specification.*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Reserved Parameter** | **Data Type** | | | | |
| **Float** | **UI** | **Integer** | **String** | **Boolean** |
| Init\_Supports\_Extended\_Impulse\_Matrix |  |  |  |  | X |
| Impulse\_Matrix\_Is\_Extended |  |  |  |  | X |

*The following two rows below the header row should be added to Table 20 on page 208 of the existing IBIS 6.1 specification.*

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Reserved Parameter** | **Data Format** | | | | | | | | | |
| **Value** | **Range** | **Corner** | **List** | **Increment** | **Steps** | **Gaussian** | **Dual-Dirac** | **DjRj** | **Table** |
| Init\_Supports\_Extended\_Impulse\_Matrix | X |  |  |  |  |  |  |  |  |  |
| Impulse\_Matrix\_Is\_Extended | X |  |  |  |  |  |  |  |  |  |

**BACKGROUND INFORMATION/HISTORY:**