BUFFER ISSUE RESOLUTION DOCUMENT (BIRD)
Rev 1
3/21/15

BIRD NUMBER:	*TBD*
ISSUE TITLE:	*Backchannel BIRD (BCI BIRD)*
REQUESTOR:	*Walter Katz, Signal Integrity Software, Inc.*

DATE SUBMITTED:
DATE REVISED:
DATE ACCEPTED BY IBIS OPEN FORUM:

ANALYSIS PATH/DATA THAT LED TO SPECIFICATION:

Link training communication is required for PCI Express, IEEE 802.3,USB, Fibre Channel and other emerging serial link standards. This communication ‘provides a mechanism through which the receiver can tune the transmitter equalizer to optimize performance’ [1]. Link training capability was initially developed by Sigrity (now Cadence Design Systems) and Snowbush (IP division of Gennum). It was deemed desirable to bring this capability to the IBIS standard in order to encourage other SerDes IP suppliers to enable link training functionality for their IP as well.

This BIRD defines how link training communications are to be handled in the IBIS specification.

ANY OTHER BACKGROUND INFORMATION:

1.1 INTRODUCTION (SECTION 10.1)

(Insert before
‘This section defines how the components of an algorithmic model are specified in an IBIS file.’)

There are scenarios when a receiver and transmitter circuits do not have prior information of the analog channels. Advanced models can perform link training communication to tune the transmitter equalizer parameters for optimized performance and adapt to the signature of any analog channel. This is done when transmitter tap parameters are re-configurable and receivers help them to be configured. Advanced communication specifications such as PCI express, USB, Fibre Channel, and IEEE 802.3 define link training protocols for transmitters and receivers. If both the transmitter and receiver AMI executable models support the same link training protocol, the EDA tool will facilitate the channel for communication between the executable models enabling link training.

A Link Training algorithm can either emulate what the silicon is actually doing, or it can use channel analysis methods to determine the optimal Tx equalization settings. This ability will allow software tools to determine the Tx equalizations settings for channels that do not have automatic link training capabilities.

Communications between the Rx and Tx DLLs are in messages that both the Rx and Tx understand, and the EDA tool does not need to understand. These agreed upon message sets are called a Backchannel Protocol. PIGEONS are BIRDS that define a Backchannel Protocol. Tx and Rx model makers may use a private Backchannel Protocol. Backchannel Protocols can become part of the IBIS standard using the IBIS BIRD process.

The key point is that this BIRD describes the mechanism for the EDA tool to allow information to be transferred from the Tx to the Rx and from the Rx to the Tx without requiring the EDA tool to understand the content of this information.

ADD TO SECTION 10.7 (MOVE SECTION 10.7 to SECTION 10.9?) A NEW SUB-SECTION

AMI RESERVED PARAMETER DEFINITIONS FOR LINK TRAINING COMMUNICATIONS

In this section, the parameters Backchannel_Protocol , BCI_State, BCI_Init_Training, BCI_GetWave_Training, BCI_Init_After_GetWave and BCI_GetWave_Block_Size are documented to enable link training communication. These Reserved Parameters are in the AMI file and positioned under the Reserved_Parameters branch.

Parameter: **Backchannel_Protocol**

Required: No.

Descriptors:

Usage:	In
Type:	String
Format:	Value, List
Default:	<string literal>
Description:	<string>

Definition: This parameter contains the name (or names) of Backchannel Protocol(s) that the model supports. This parameter tells the model which Backchannel Protocol is being used for the training process. Both the transmitter and receiver for a given channel must have identical settings for the Backchannel_Protocol parameter for link training to be enabled. The Backchannel_Protocol defines the contents of the BCI branch that is inserted by the Tx and Rx AMI_Init and AMI_GetWave functions in their AMI_parameters_out or AMI_parameters_inout. The EDA tool is responsible for copying this branch into the AMI_parameters_in or AMI_parameters_inout string that is passed to the next Tx or Rx AMI_Init and AMI_GetWave function calls.

If the name of a Backchannel_Protocol ends in .bci, then it is a name of a file that both the Tx and Rx DLL shall comply with when sending messages between the Tx and Rx DLL in the BCI branch. The Tx and Rx DLL may read the .bci file. These .bci files shall be delivered with the .ami model and be in the same directory as the .ibs file and the .ami file.

Usage Rules:

Other Notes: A Backchannel_Protocol may be private, published, or approved by IBIS. This approval process is explicitly not stated in this BIRD, and left to the IBIS Open Forum to decide.

Example:

```
(Backchannel_Protocol (Usage In)(Type String)(Value "Basic")
 (Description "This Device can support Backchannel Protocol Basic."))
```

Parameter: **BCI_State**

Required: No.

Descriptors:

Usage:	InOut
Type:	String
Format:	List (“Off” ”Training” ”Done” “Abort”)
Default:	<String Literal>
Description:	<string>

Definition: The EDA tool sets the value of BCI_State to either “Off” or ”Training” on each call to the Tx and Rx AMI_Init and AMI_GetWave call in accordance with the flows described below. If BCI_State is “Training” the Rx AMI_Init and AMI_GetWave shall return either the value ”Training”, ”Done”, or “Abort”.

Usage Rules:

Other Notes:

Example:

```
(BCI_State (Usage InOut)(Type String)(List “Off” ”Training” ”Done” “Abort”))
```

Parameter: **BCI_GetWave_Block_Size**

Required: No.

Descriptors:

Usage:	Info
Type:	UI
Format:	Value
Default:	1000
Description:	<string >

Definition: This Rx parameter tells the EDA tool the recommended number of UI in each GetWave call to be used in Time Domain simulations.

Example:

```
(BCI_GetWave_Block_Size(Usage Info) (Type UI) (Value 2000)
      (Description "GetWave blocks should contain 2000 UI”))
```

Parameter: **BCI_Init_Training**

Required: No.

Descriptors:

Usage:	Info
Type:	Boolean
Format:	Value
Default:	True
Description:	<string >

Definition: This Rx parameter tells the EDA tool if this model support statistical training.

Example:

```
(BCI_Init_Training(Usage Info) (Type Boolean) (Value True)
      (Description "This model supports statistical training”))
```

Parameter: **BCI_GetWave_Training**

Required: No.

Descriptors:

Usage:	Info
Type:	Boolean
Format:	Value
Default:	True
Description:	<string >

Definition: This Rx parameter tells the EDA tool if this model supports time domain training.

Example:

```
(BCI_GetWave_Training(Usage Info) (Type Boolean) (Value True)
      (Description "This model support time domain training”))
```

Parameter: **BCI_Init_After_GetWave**

Required: No.

Descriptors:

Usage:	Info
Type:	Boolean
Format:	Value
Default:	True
Description:	<string >

Definition: This Rx parameter tells the EDA tool if this model statistical flow after time domain training.

Example:

```
(BCI_Init_After_GetWave (Usage Info) (Type Boolean) (Value True)
      (Description "This model supports statistical flow after time domain
training”))
```

Notes:

Training and Co-optimization is done by Rx models using one or more Tx equalization exploration algorithms. The Rx model may have Model Specific parameters that allow the user to choose which exploration algorithm to use.

EDA tools already have mechanisms for determining stimulus patters used for analyzing channel performance. During time domain (GetWave) training, the EDA tool shall supply a Link Training Pattern which is general may be different than the pattern used for analyzing channel performance. USB 3.1, IEEE 802.3 and PSIeG3 generally recommend a PRBS 11 pattern for Link Training. The EDA tool may choose to use a PRBS 11 pattern, or may enable the user to choose a different pattern.

The following table indicates which kind of training is supported for all allowed combinations of Tx and Rx Init_Returns_Impulse and GetWave Exists.

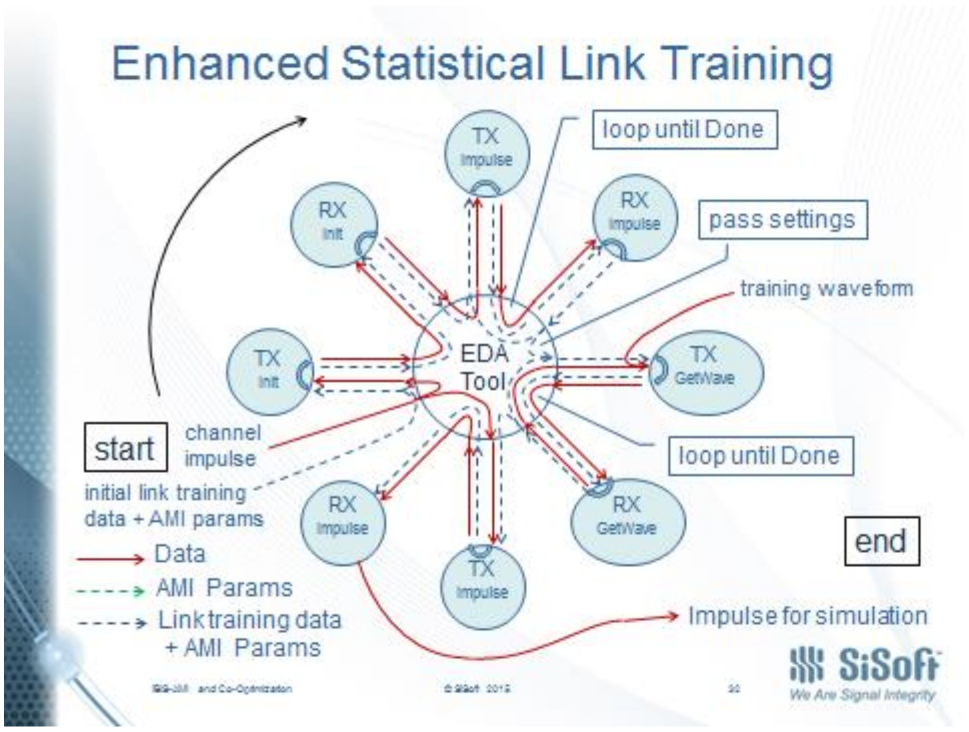
Modes	Training Mode	Tx Init Only	Tx GetWave Only	Tx Dual Mode
Rx Init Only	Init Training	Yes	Disabled	Yes
Rx Init Only	GetWave Training	Disabled	Disabled	Disabled
Rx Init Only	Dual Mode Training	Disabled	Disabled	Disabled
Rx GetWave Only	Init Training	Yes (optimizes at input to Rx)	Disabled	Yes (optimizes at input to Rx)
Rx GetWave Only	GetWave Training	Yes	Yes	Yes
Rx GetWave Only	Dual Mode Training	Yes	Disabled	Yes
Rx Dual Mode	Init Training	Yes	Disabled	Yes
Rx Dual Mode	GetWave Training	Yes	Yes	Yes
Rx Dual Mode	Dual Mode Training	Yes	Disabled	Yes

If a Tx has a Backchannel Protocol that is approved by IBIS (or published by a model maker) there is nothing in this standard that prohibits an EDA tool from configuring the Tx equalization by creating the contents of the BCI branch and therefore allowing the EDA tool to optimize a channel without requiring the use of an Rx model that supports this BIRD.

Not included in this BIRD is a definition of a Tx Meta File that would map Tx AMI parameter values to Tx tap coefficients. Such a file would enable an EDA tool to *optimize a Pre IBIS 7.0 Tx by calling the Tx AMI_Close and then the Tx AMI_Init with the Tx AMI parameters that would configure the Tx to optimized AMI parameters. This would work fine in a Statistical optimization flow, and be more challenging to an EDA tool to implement a Time Domain optimization flow.*

Training/Analysis Flows

Statistical and Time Domain Training Flow



The most complex training and analysis flow can occur when both the Tx and Rx support statistical and time domain training, and statistical analysis after time domain training:

Statistical_Training

Statistical_Analysis

Time_Domain_Training

Statistical_Analysis

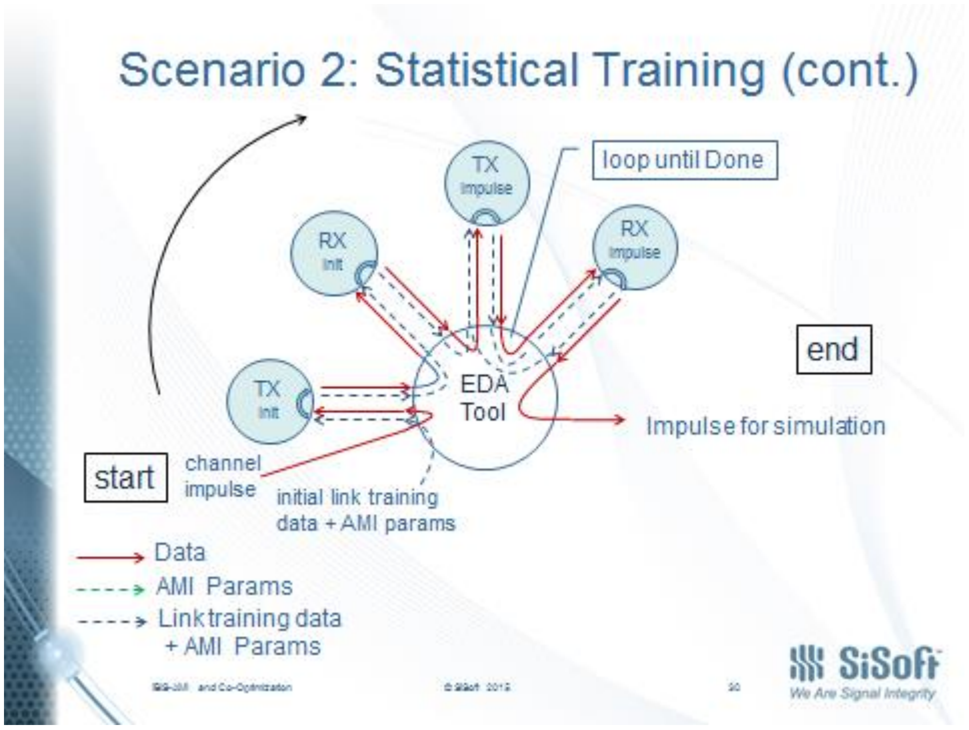
Time_Domain_Analysis

The EDA tool shall make the following calls to the Tx and Rx AMI_Init, AMI_Init and AMI_GetWave functions:

1. Tx AMI_Init is called with (BCI_State “Training”)

   a. Tx AMI_Init outputs a branch (BCI <1>)
2. Rx AMI_Init is called with (BCI_State “Training”) (BCI <1>)
   a. Rx AMI_Init outputs (BCI_State “Training|Done|Abort”) (BCI <2>)
3. Tx AMI_Init is called with (BCI <2>)
   a. Tx AMI_Init outputs a branch (BCI <3>)
4. Rx AMI_Init is called with (BCI_State “Training”) (BCI <3>)
   a. Rx AMI_Init outputs (BCI_State “Training”) (BCI <4>)
5. Steps 3 and 4 are repeated until step 4 returns (BCI_State “Done”)
6. Tx AMI_GetWave is called with (BCI_State “Training”) and a training stimulus pattern
7. Rx AMI_GetWave is called with (BCI_State “Training”)
   a. Rx AMI_GetWave outputs (BCI_State “Training”) (BCI <5>)
8. Tx AMI_GetWave is called with (BCI <5>) and a training stimulus pattern
   a. Tx AMI_GetWave outputs a branch (BCI <6>)
9. Rx AMI_GetWave is called with (BCI_State “Training”)
   a. Rx AMI_GetWave outputs (BCI_State “Training”) (BCI <7>)
10. Steps 8 and 9 are repeated until step 9 returns (BCI_State “Done”)
11. Tx AMI_Init is called with (BCI_State “Off”)
12. Rx AMI_Init is called with (BCI_State “Off”)
13. EDA tool does statistical analysis on the Impulse Response output of Rx AMI_Init
14. Tx AMI_GetWave is called with (BCI_State “Off”) and analysis stimulus pattern
15. Rx AMI_GetWave is called with (BCI_State “Off”)
16. Steps 14 and 15 are repeated, EDA tool analyzes the waveform output of Rx AMI_GetWave

Statistical Training Flow



Statistical_Training

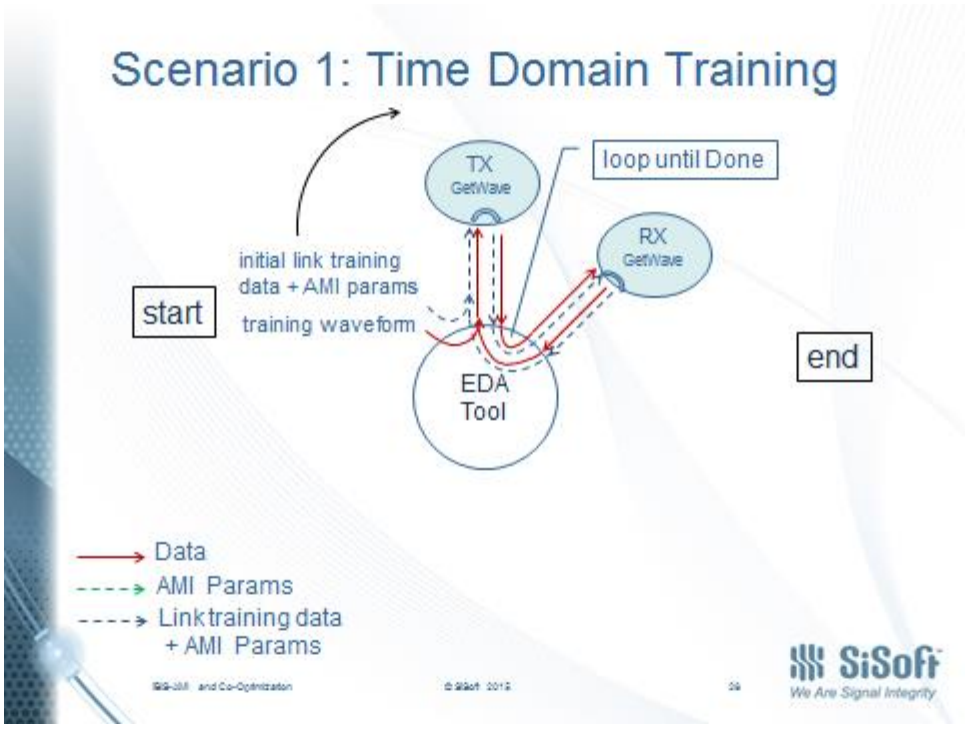
Statistical_Analysis

Time_Domain_Analysis

The EDA tool shall make the following calls to the Tx and Rx AMI_Init, AMI_Init and AMI_GetWave functions:

1. Tx AMI_Init is called with (BCI_State “Training”)
    a. Tx AMI_Init outputs a branch (BCI <1>)
2. Rx AMI_Init is called with (BCI_State “Training”) (BCI <1>)
    a. Rx AMI_Init outputs (BCI_State “Training|Done|Abort”) (BCI <2>)
3. Tx AMI_Init is called with (BCI <2>)
    a. Tx AMI_Init outputs a branch (BCI <3>)
4. Rx AMI_Init is called with (BCI_State “Training”) (BCI <3>)
    a. Rx AMI_Init outputs (BCI_State “Training”) (BCI <4>)
5. Steps 3 and 4 are repeated until step 4 returns (BCI_State “Done”)
6. Tx AMI_GetWave is called with (BCI_State “Off”) and analysis stimulus pattern
7. Rx AMI_GetWave is called with (BCI_State “Off”)
8. Steps 6 and 7 are repeated, EDA tool analyzes the waveform output of Rx AMI_GetWave

Time Domain Training Flow



Statistical_Analysis

Time_Domain_Training

Statistical_Analysis

Time_Domain_Analysis

The EDA tool shall make the following calls to the Tx and Rx AMI_Init, AMI_Init and AMI_GetWave functions:

1. Tx AMI_Init is called with (BCI_State "Off")
    a. Tx AMI_Init outputs a branch
2. Rx AMI_Init is called with (BCI_State "Off")
3. Tx AMI_GetWave is called with (BCI_State "Training") and a training stimulus pattern
4. Rx AMI_GetWave is called with (BCI_State "Training")
    a. Rx AMI_GetWave outputs (BCI_State "Training") (BCI <1>)
5. Tx AMI_GetWave is called with (BCI <1>) and a training stimulus pattern
    a. Tx AMI_GetWave outputs a branch (BCI <2>)
6. Rx AMI_GetWave is called with (BCI_State "Training") (BCI <2>)
    a. Rx AMI_GetWave outputs (BCI_State "Training") (BCI <3>)
7. Steps 5 and 6 are repeated until step 9 returns (BCI_State "Done")
8. Tx AMI_GetWave is called with (BCI_State "Off") and analysis stimulus pattern
9. Rx AMI_GetWave is called with (BCI_State "Off")

10. Steps 8 and 9 are repeated, EDA tool analyzes the waveform output of Rx AMI_GetWave

Notes

For time domain simulations, total number of training bits will limited when Rx indicates BCI_State "Done". When doing time domain link training, the EDA tool shall use BCI_State "Off" to proceed with analyzing the simulation waveforms and shall ignore the first "Ignore_Bits" of waveform data after training is turned off.

The EDA tool can also terminate Init training by calling the Rx AMI_Init function with BCI_State "Off". The EDA tool can terminate GetWave training by calling the Rx AMI_GetWave function with BCI_State "Off".

Rx should output measure of how good eye is

It is recommended that the Rx model have Usage Out parameter(s) that contains a measure of how good the eye is at the decision point.

Model makers should supply silicon programming tools

The Tx and Rx model makers should supply tools to the user that would enable the user to program the Tx and Rx silicon from the outputs supplied by the AMI model.

Example of Rx AMI Parameters Supporting 802.3KR Link Training

```
(Backchannel_Protocol  (Usage In)(Type String)(Value "802.3kr.bci")
   (Description "This Device supports the 802.3kr.ibc protocol."))
(BCI_State (Usage InOut)(Type String)
   (List “Off” “Training” “Done” “Abort”))
   (Description "This model supports only Time Domain training”))
(BCI_GetWave_Block_Size (Usage Info) (Type UI) (Value 1000)
   (Description "GetWave block should contain 1000UI”))
(BCI_Init_After_GetWave (Usage Info) (Type Boolean) (Value False)
   (Description "This model does not support statistical flow after
                time domain training”))
(BCI_Init_Training (Usage Info) (Type Boolean) (Value False)
   (Description "This model does not support statistical training”))
(BCI_GetWave_Training (Usage Info) (Type Boolean) (Value True)
   (Description "This model supports time domain training”))
```

Example of Tx AMI Parameters Supporting 802.3KR Link Training

```
(Backchannel_Protocol  (Usage In)(Type String)(Value "802.3kr.bci")
   (Description "This Device supports the 802.3kr.ibc protocol."))
(BCI_State (Usage In)(Type String)
   (List “Off” “Training” “Done”))
(BCI_Init_After_GetWave (Usage Info) (Type Boolean) (Value False)
   (Description "This model does not support statistical flow after
                time domain training”))
(BCI_Init_Training (Usage Info) (Type Boolean) (Value False)
   (Description "This model does not support statistical training”))
(BCI_GetWave_Training (Usage Info) (Type Boolean) (Value True)
   (Description "This model supports time domain training”))
```

COMMUNICATION PROTOCOL BETWEEN THE TX AND RX FOR LINK TRAINING

THERE IS NOW A NEW USE OF AMI_INIT

Initial call to AMI_Init has three functions (*AMI_memory initialized by EDA tool to Null)

- Returns *AMI_memory pointer to DLL allocated memory.
- Getting model conditions (from In and InOut parameters)
- Determining “Statistical Impulse Response”, or more precisely the effect of the “Statistical Impulse Response on the Channel Impulse Response”

The new AMI_Init functionality shall occur when input *AMI_memory is not Null. This call to AMI_Init will calculate a new impulse response modified by the Tx equalization changes requested by the Rx AMI_Init in Training Mode. AMI_parameters_in string shall contain a BCI branch as documented in the models Backhannel_Protocol.