**BUFFER ISSUE RESOLUTION DOCUMENT (BIRD)**

**BIRD NUMBER:** (for administrative use)

**ISSUE TITLE:** Lightweight Backchannel BIRD

**REQUESTOR:**  Bob Miller, Broadcom, Ltd

 Ambrish Varma, Cadence Design Systems, Inc

 Walter Katz, Signal Integrity Software, Inc

**DATE SUBMITTED:** (for administrative use)

**DATE REVISED:** (for administrative use)

**DATE ACCEPTED:** (for administrative use)

**DEFINITION OF THE ISSUE:**

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]. These mechanisms employ a reliable “backchannel” to support administrative link training communication between the transmitter and receiver SerDes.

Broadcom wants the IBIS standard to be able to define standardized BackChannel Interface (BCI) Protocols so that different IC Vendors can write IBIS-AMI models that can communicate with each other, without requiring any support from EDA Vendor tools.

This BIRD defines how link training communications can be standardized in the IBIS specification. This BIRD expressly does not attempt at this time to define specific standard protocols which utilize these definitions.

[1] Section 5, IEEE Std 802.3.**SOLUTION REQUIREMENTS:**

The IBIS specification must meet these requirements:

|  |  |
| --- | --- |
| Requirement | Notes |
| 1. Enable backchannel link training messages between the Tx and Rx DLL to enable the Rx DLL to control the equalization of the Tx during time domain (AMI\_GetWave) simulations.
 | Backchannel messages are implemented via file I/O in the simulation’s working directory instead of parameter string passing via the EDA tool. |
| 1. Support backchannel messages between the Tx and Rx DLLs in channels that have Redriver(s) to enable the Downstream (primary) Rx DLL to control the equalization of the upstream Tx and Rx DLL’s during time domain (AMI\_GetWave) simulations.
 | A lightweight communication scheme supports multi-hop channel optimization, the details of which may be defined in specific future protocols. |
| 1. Does not require the EDA tool to make any changes to support these communications.
 | There are minor changes the EDA tool can make to improve the user experience, but these changes are not required since they can be accommodated in either the .ami files, or some extra setup required by the user. |
| 1. Allow the user and tool to know when link training has ended and normal operation has begun.
 | EDA support can facilitate user awareness of successful training but is not required. |
| 1. Support both private and published link training protocols.
 | Protocols might be published within IBIS or elsewhere later. |
| 1. Provide all channel DLL instances with a unique file namespace for backchannel communication.
 | If the EDA tool does not directly facilitate the namespace selection, the user may select namespaces which are compatible with a specific EDA tool via the models’ ami files. |

**SUMMARY OF PROPOSED CHANGES:**

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

|  |  |  |
| --- | --- | --- |
| Specification Item | New/Modified/Other | Notes |
| New AMI Reserved\_ParametersBCI\_ProtocolBCI\_StateBCI\_IDBCI\_GetWave\_BitsBCI\_Training\_Bits | All are new AMI Parameters | All affect the operation of the AMI functions AMI\_Init, AMI\_GetWave |

**PROPOSED CHANGES:**

## 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 their analog channel. 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 (Back Channel Protocol), the EDA tool will facilitate the communication between the executable models enabling link training. Another name for Link Training in the industry is Auto-Negotiation.

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 also allow Rx AMI models to determine the Tx equalizations settings for channels that do not have automatic link training capabilities.

Channels with Redrivers will require that the Downstream Rx be able to control all upstream equalization.

Communications between the Rx and Tx DLLs are in messages that both the Rx and Tx DLLs understand, and the EDA tool does not need to understand. These agreed upon messages are called a Backchannel Protocol. This specification does not specify the details of the Backchannel Protocol but only a method to make the communication work.

This specification describes an underlying mechanism for the AMI .ami file and the DLL 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, or even for the EDA tool to know that backchannel communications is occurring.

With the information provided in this specification, IC Vendors can develop models that support Back Channel Training in current IBIS AMI EDA tools.

**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 BCI\_Protocol, BCI\_State, BCI\_ID, BCI\_GetWave\_Bits and BCI\_Training\_Bits are documented to enable link training communication. These Reserved Parameters are in the AMI file and positioned under the Reserved\_Parameters branch.

*Parameter:* **BCI\_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.. The BCI\_Protocol defines the backchannel message files and BCI data contained therein that is read and/or generated by each call to each DLL.

*Usage Rules:* Both the transmitter and receiver for a given channel must have identical settings for the BCI\_Protocol parameter for link training to be enabled. Both the transmitter and receiver for a given channel must have GetWave\_Exists = True for link training to be enabled

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

*Example:*

(BCI\_Protocol (Usage In)(Type String)(Value "Basic")

 (Description "This Device supports Backchannel Protocol Basic."))

*Parameter:*      **BCI\_ID**

*Required:*        No, and illegal before AMI\_Version 7.0?

*Direction:*Rx, Tx

*Descriptors*:

Usage:                   In

Type:                     String

Format:                  Value

Default:                 <string literal>

Description:<string>

*Definition:*           The EDA tool is responsible for recognizing this parameter name and replacing the value declared in the .ami file with a string that contains a unique alphanumeric identifier.  The algorithmic model is responsible for using BCI\_ID as the base name for any data files that the model creates, either for use as temporary storage or for recording output data in accordance with the BCI\_Protocol.  All model instances in a channel between and including the upstream Tx and downstream Rx shall have the same BCI\_ID. The use of BCI\_ID helps guarantee that multiple channels do not mix up data as a result of collisions between temporary or permanent file names.

*Usage Rules:*BCI\_ID needs to be a string that when concatenated with any appropriate namestring and/or extension supplied by the DLL results in a unique valid filename in the system the DLL is called under.

*Example:*

(BCI\_ID (Usage In) (Type String) (Value "placeholder")

 (Description "Unique base name for the AMI models in each channel and run"))

*Other Notes:*  A BCI\_Protocol may define one, two (e.g. one per direction) or any number of bci message files with the same BCI\_ID prefix to be used by the channel Tx and Rx DLLs to support the required backchannel optimization.

If the EDA tool does not modify/uniquify BCI\_ID, the user must select and insert an appropriate matching string in the ami files of each DLL participating in the channel’s BCI\_Protocol to manually provide a functioning file namespace.

*Parameter:* **BCI\_State**

*Required:* No.

*Direction:* Rx, Tx

*Descriptors*:

Usage: InOut

Type: String

Format: List (“Off” ”Training” “Converged” “Failed” “Error”)

Default: <String Literal>

Description:<string>

*Definition:* The user sets the value of BCI\_State to either “Off” or ”Training” on the calls to the Tx and Rx AMI\_Init. The values of BCI\_State sent to the Tx and Rx DLL shall be the same for both the Tx and Rx AMI\_Init.

*Usage Rules:* If the BCI\_State is “Off” on the calls to Tx and Rx AMI\_Init, both the Tx and Rx DLL will not read or generate files in the BCI\_ID namespace. The values of BCI\_Protocol, BCI\_GetWave\_Bits or BCI\_Training\_Bits shall be ignored by the DLL. DLLs receiving BCI\_State “Off” and subsequently returning BCI\_State shall return BCI\_State “Off”.

If the BCI\_State is “Training” on the calls to Tx and Rx AMI\_Init, both the Tx and Rx DLL will read and/or write files in the BCI\_ID namespace per the BCI\_Protocol. The values of BCI\_Protocol, BCI\_ID, BCI\_GetWave\_Bits and BCI\_TRAINING\_Bits are required. The Rx AMI\_GetWave calls shall return a value in BCI\_State of either “Training”, “Converged”, ”Failed” or “Error”. If theTx AMI\_GetWave returns a value in BCI\_State, it shall also be either “Training”, “Converged”, ”Failed” or “Error”; “Training”, “Converged” , and “Failed” should reflect the Rx state per the BCI\_Protocol.

The EDA tool shall consider the value of BCI\_State returned by the terminating Rx DLL to be the definitive BCI\_Protocol training state. However, any DLL in the channel, upon returning a BCI\_State value of “Error”, may thereby signal that a BCI\_Protocol has failed due to a mis-communication under the BCI\_Protocol.

If the returned value is “Training”, then the Tx and Rx AMI\_GetWave will continue to read and/or modify BCI\_ID files per the BCI\_Protocol.

If the returned value is “Converged”, then the Tx and Rx AMI\_GetWave may continue to read and/or modify the BCI\_ID files per the BCI\_Protocol. However, it is implied that no further adaptation is performed under the BCI\_Protocol and the EDA tool may complete the simulation/analysis starting with this waveform.

If the returned value is “Failed” the Rx AMI\_GetWave function indicates a condition that it was not able to converge in its search algorithm. Then the Tx and Rx AMI\_GetWave may continue to read and/or modify the BCI\_ID files per the BCI\_Protocol. However, it is implied that no further adaptation is performed under the BCI\_Protocol and the EDA tool may complete the simulation/analysis starting with this waveform.

If the returned Tx or Rx value is “Error”, the DLL indicating “Error” is unable to understand the messages according to the BCI\_Protocol. The Tx and/or Rx AMI\_GetWave will stop reading and/or modifying the BCI\_ID files. The EDA tool may communicate a protocol error to the user and complete the simulation/analysis starting with this waveform.

*Example:*

(BCI\_State (Usage InOut)(Type String)

 (List“Off” ”Training” “Converged” “Failed” “Error”))

*Other 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.

During “Training”, the EDA tool may supply a “training” stimulus pattern defined by the user. While not required, the Back Channel Protocol will likely specify the pattern that should be used.

*Parameter:* **BCI\_GetWave\_Bits**

*Required:* No.

*Direction:* Rx

*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 AMI\_GetWave call to be used in Time Domain simulations.

*Usage Rules:* The wave\_size passed to AMI\_GetWave would be the value of BCI\_GetWave\_Bits\*bit\_time/sample\_interval.

*Other Notes:* This parameter allows a BCI\_Protocol to define the number of training bits (“dwell time”) between bci messages, which necessarily must occur at most once per GetWave call. This may be necessary in some protocols or rigorous channel simulations to enforce bit-by-bit emulation.

*Example:*

(BCI\_GetWave\_Bits(Usage Info) (Type UI) (Value 2000)

(Description "AMI\_GetWave blocks should contain 2000 UI”))

*Parameter:* **BCI\_Training\_Bits**

*Required:* No

*Direction:* Rx

*Descriptors*:

Usage: In

Type: UI

Format: Value

Default: <numeric\_literal>

Description:<string>

*Definition:* Tells the EDA tool how long the time variant model may take to complete training.

*Usage Rules:* This parameter is meant for Rx models that support BCI Training. The value in this field tells the EDA tool and the Rx AMI\_GetWave function how many bits of the AMI\_GetWave output should be used for training.

BCI\_Training\_Bits must be present if BCI training is enabled.

*Other Notes:* If an EDA tool does not use BCI\_Training\_Bits or BCI\_State to determine when it can start analysis of the optimized waveform generated by the Rx AMI\_GetWave, the user (or .ami file) should set Ignore\_Bits to the same value as BCI\_Training\_Bits.

*Examples:*

(BCI\_Training\_Bits (Usage In) (Type UI) (Value 100000)

 (Description "BCI training may require 100000 UI")

**Training/Analysis Flow for Channels with No Redriver**

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
	1. (BCI\_State “Training”) (BCI\_Protocol “<name>“) (BCI\_ID “<my\_ ID> “)
	2. If the Tx DLL does not implement the BCI\_Protocol, it returns “Error” in BCI\_State.
	3. The Tx may write a message file in the BCI\_ID namespace under BCI\_Protocol.
2. Rx AMI\_Init is called with
	1. (BCI\_State “Training”) (BCI\_Protocol “<name>“) (BCI\_ID “<my\_ID> “) (BCI\_Training\_Bits <# Training Bits>)
	2. If the Rx DLL does not implement BCI\_Protocol, it returns “Error” in BCI\_State.
	3. The Rx may read, write, modify and/or delete message files in the BCI\_ID namespace under BCI\_Protocol.
3. Tx AMI\_GetWave is called with the stimulus pattern. The Tx may read, write, modify and/or delete message files in BCI\_namespace under BCI\_Protocol.
4. Rx AMI\_GetWave is called with the waveform output of Tx AMI\_GetWave convolved with the IR of the channel. The Rx may read, write, modify and/or delete message files under BCI\_Protocol.
5. Steps 3 and 4 are repeated until the EDA tool stops the simulation.
	1. The EDA tool should start processing the output of Rx AMI\_GetWave after Ignore\_Bits and either:

after BCI\_Training\_Bits, or

when the Rx AMI\_GetWave function returns BCI\_State “Converged” or “Failed” or either the Tx or Rx DLL returns “Error”.

Note that the EDA tool does not need to perform any operations specifically assisting the BCI communication between the Tx DLL and the Rx DLL beyond passing the BCI parameters to both DLLs on AMI\_Init.

**Training/Analysis Flow for Channels with One Redriver**

The EDA tool shall make the following calls to the Upstream Tx, Redriver Rx, Redriver Tx, Downstream Rx AMI\_Init, AMI\_Init and AMI\_GetWave functions:

1. Upstream Tx AMI\_Init is called with
	1. (BCI\_State “Training”) (BCI\_Protocol “<name>“) (BCI\_ “<my\_ID>“)
	2. If the DLL does not implement the BCI\_Protocol, it returns “Error” in BCI\_State
	3. The DLL may write a message file in the BCI\_ID namespace under BCI\_Protocol.
2. Redriver Rx AMI\_Init is called with
	1. (BCI\_State “Training”) (BCI\_Protocol “<name>“) (BCI\_ID “<my\_ID>“)
	2. If the DLL does not implement the BCI\_Protocol, it returns “Error” in BCI\_State
	3. The DLL may read, write, modify and/or delete message files in the BCI\_ID namespace under BCI\_Protocol.
3. Redriver Tx AMI\_Init is called with
	1. (BCI\_State “Training”) (BCI\_Protocol “<name>“) (BCI\_ID “<my\_ID>“)
	2. If the DLL does not implement the BCI\_Protocol, it returns “Error” in BCI\_State
	3. The DLL may read, write, modify and/or delete message files in the BCI\_ID namespace under BCI\_Protocol.
4. Downstream Rx AMI\_Init is called with
	1. (BCI\_State “Training”) (BCI\_Protocol “<name>“) (BCI\_ID “<my\_ID>“) (BCI\_Training\_Bits <# Training Bits>)
	2. If the DLL does not implement the BCI\_Protocol, it returns “Error” in BCI\_State
	3. The DLL may read, write, modify and/or delete message files in the BCI\_ID namespace under BCI\_Protocol.
5. Upstream Tx AMI\_GetWave is called with the stimulus pattern. The DLL may read, write, modify and/or delete message files in the BCI\_ID namespace under BCI\_Protocol.
6. Redriver Rx AMI\_GetWave is called with the waveform output of the Upstream Tx AMI\_GetWave modified by the Upstream Channel Impulse Response. The DLL may read, write, modify and/or delete message files in the BCI\_ID namespace under BCI\_Protocol.
7. Redriver Tx AMI\_GetWave is called with the waveform output of the Redriver Rx AMI\_GetWave. The DLL may read, write, modify and/or delete message files in the BCI\_ID namespace under BCI\_Protocol.
8. Downstream Rx AMI\_GetWave is called with the waveform output of the Redriver Tx AMI\_GetWave modified by the Downstream Channel Impulse Response. The DLL may read, write, modify and/or delete message files in the BCI\_ID namespace under BCI\_Protocol
9. Steps 5 through 8 are repeated until the EDA tool stops the simulation.
	1. The EDA tool should start processing the output of Rx AMI\_GetWave after Ignore\_Bits and either:

after BCI\_Training\_Bits, or

when the downstream Rx AMI\_GetWave function returns BCI\_State “Converged” or “Failed” or any DLL in the channel returns “Error”.

Note that it is the responsibility of the BCI \_Protocol to define the bci message files and contents therein so that each DLL in the channel can determine its role/position in the channel optimization.

**BACKGROUND INFORMATION/HISTORY:**

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.