**BUFFER ISSUE RESOLUTION DOCUMENT (BIRD)**

**BIRD NUMBER:** 201

**ISSUE TITLE:** Back-channel Statistical Optimization

**REQUESTOR:**  Walter Katz, Signal Integrity Software

**DATE SUBMITTED:** January 7, 2020

**DATE REVISED:**

**DATE ACCEPTED:**

**DEFINITION OF THE ISSUE:**

IC vendors would like Back-channel to support training in the Statistical flow iterating between the Tx and Rx doing optimization based on impulse responses.

This BIRD defines one new BCI reserved parameter and how it determines what the EDA tool needs to do to support training in the Statistical Flow.

This BIRD also defines one new DLL function AMI\_Impulse. After the initial calls to Tx and Rx AMI\_Init, and if Statistical Training is enabled, the EDA tool shall repeatably call the Tx and Rx AMI\_Impulse functions until the Rx AMI\_Impulse returns BCI\_State Converged.

**SOLUTION REQUIREMENTS:**

The IBIS specification must meet these requirements:

|  |  |
| --- | --- |
| Requirement | Notes |
| 1. Enable back-channel link training messages between the Tx and Rx executable models to enable the Rx or Tx executable model to control the equalization of the Tx or Rx during time statistical simulations.
 |  |
| 1. Allow the user and tool to know when statistical link training has ended, and normal operation has begun.
 |  |

**SUMMARY OF PROPOSED CHANGES:**

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

|  |  |  |
| --- | --- | --- |
| Specification Item | New/Modified/Other | Notes |
| New AMI Reserved ParameterBCI\_Training\_Mode | All are new AMI Parameters | All affect the operation of the AMI function AMI\_Impulse. |
| New DLL function AMI\_Impulse | New |  |

**PROPOSED CHANGES:**

## Introduction (Section 10.1)

This specification describes an underlying mechanism for the AMI .ami file and the executable model to allow iteration of the Tx and Rx AMI\_Impulse function to allow either the Tx and/or the Rx to optimize the equalization of the channel.

With the information provided in this specification, IC Vendors and EDA vendors can develop models that support Statistical Back Channel Training and the enhancements EDA tools will need to support these models.

## New AMI Reserved Parameter DEFINITIONs For Link training Communications in Statistical Flow

*Parameter:* **BCI\_Training\_Mode**

*Required:* No, and illegal before AMI\_Version 7.1

*Direction:* Rx, Tx

*Descriptors*:

Usage: In

Type: String

Format: Value, List

Default: <string literal>

Description:<string>

*Definition:* This parameter tells the EDA tool if the model supports Statistical Optimization Only, Time Domain Optimization Only, or is Dual (Both Statistical and Time Domain) Optimization. The only allowed values of BCI\_Training\_Mode are “Impulse”, “GetWave” or “Dual”.

*Usage Rules:* The User/EDA tool can only choose a BCI\_Training\_Mode value if it is available on both the Tx and the Rx. If BCI\_Training\_Mode is not present, then it shall default to “GetWave”.

In order to be able to run a BCI Statistical simulation, the Tx and Rx model (or the primary Tx, terminal Rx and all repeater Rx and Tx models) must have BCI\_Training\_Mode as either “Impulse” or “Dual”.

In order to be able to run a BCI Time Domain simulation, the Tx and Rx model (or the primary Tx, terminal Rx and all repeater Rx and Tx models) must have BCI\_Training\_Mode as either “GetWave” or “Dual”.

(Note: add this to table 39)

*Example:*

(BCI\_Training\_Mode (Usage In)(Type String)(List “Dual” “Impulse” “GetWave”)

(Description "This Device supports Back-channel Statistical and Time Domain Optimization”)

**Table YY1 – General Rules and Allowable Usage for BCI Reserved Parameters**

| **Reserved Parameter** | **General Rules** | **Allowable Usage** |
| --- | --- | --- |
| **Required** | **Default** | **Info** | **In** | **Out** | **Dep1** | **InOut** |
| BCI\_Message\_Interval\_UI | No, Yes if BCI\_Protocol is present | -- | X |  |  |  |  |
| BCI\_ID | No, Yes if BCI\_Protocol is present | -- |  | X |  |  |  |
| BCI\_Protocol | No, Yes to support BCI protocol | -- |  | X |  |  |  |
| BCI\_State | No, Yes if BCI\_Protocol is present | -- |  |  |  |  | X |
| BCI\_Training\_UI | No, Yes if BCI\_Protocol is present | -- |  | X |  |  |  |
| BCI\_Training\_Mode | No, Yes if BCI\_Protocol support “Init” Optimization |  |  | X |  |  |  |

1. Illegal for AMI\_Version 6.0 and earlier

**Table YY2 – Allowable Data Types for BCI Reserved Parameters**

| **Reserved Parameter** | **Data Type** |
| --- | --- |
| **Float** | **UI** | **Integer** | **String** | **Boolean** |
| BCI\_Message\_Interval\_UI |  |  | X |  |  |
| BCI\_ID |  |  |  | X |  |
| BCI\_Protocol |  |  |  | X |  |
| BCI\_State |  |  |  | X |  |
| BCI\_Training\_UI |  |  | **X** |  |  |
| BCI\_Training\_Mode |  |  |  | X |  |

**Table YY3 – Allowable Data Formats for BCI Reserved Parameters**

| **Reserved Parameter** | **Data Format** |
| --- | --- |
| **Value** | **Range** | **Corner** | **List** | **Increment** | **Steps** | **Gaussian** | **Dual-Dirac** | **DjRj** | **Table** |
| BCI\_Message\_Interval\_UI | X |  |  |  |  |  |  |  |  |  |
| BCI\_ID | X |  |  |  |  |  |  |  |  |  |
| BCI\_Protocol | X |  |  | X |  |  |  |  |  |  |
| BCI\_State |  |  |  | X |  |  |  |  |  |  |
| BCI\_Training\_UI | X |  |  |  |  |  |  |  |  |  |
| BCI\_Training\_Mode | X |  |  | X |  |  |  |  |  |  |

## New DLL Function Signatures Link training Communications in Statistical Flow

*Function:* **AMI\_Impulse**

*Required:* Yes

*Declaration:* long AMI\_Impulse (double \*impulse\_matrix,

 char \*BCI\_parameters\_in,

 char \*\*BCI\_parameters\_out,

 char \*\*AMI\_parameters\_out,

 void \*AMI\_memory)

*Arguments:*

impulse\_matrix

Same impulse\_matrix as defined on page 201 of IBIS 7.0 that is the input to AMI\_Init.

Note that since both AMI\_Init and AMI\_Impulse modify the impulse\_matrix in place, the EDA tool could maintain the original impulse\_matrix and use different memory for the impulse\_matrix input to the AMI\_Init and AMI\_Impulse functions.

Note that the AMI\_Impulse function uses the number\_of\_rows, aggressors, sample\_interval and bit\_time that were passed to the AMI\_Init call.

BCI\_parameters\_in

The BCI\_parameters\_in argument is a pointer to a string. This pointer is returned in the BCI\_parameter\_out argument by a previous call to an AMI\_Impulse function in another DLL in the channel. Memory for the string is allocated and de-allocated by the previous call to an AMI\_Impulse function in another DLL in the channel. The string must be formatted as defined by the BCI\_Protocol. On the first call to the primary Tx AMI\_Impulse function, this pointer shall be the Null pointer (0).

BCI\_parameters\_out

The BCI\_parameters\_out argument is a pointer to a string pointer. Memory for the string is allocated and de-allocated by the algorithmic model. The model returns a pointer to the string as the contents of this argument. The string must be formatted as defined by the BCI\_Protocol.

The EDA tool must initialize the memory content at this address to zero (null pointer) prior to calling the AMI\_Impulse function, so that after the execution of the function it can determine whether or not the function returned a valid string pointer at that address.

AMI\_parameters\_out

See page 205 in IBIS 7.0

**AMI\_memory**

This is the memory which was allocated during the AMI\_Init call.

Return Value

1 for success

0 for failure

Algorithmic models shall return a failure code (0) if and only if the function call fails due to a program execution error. In all other cases the return code shall be "success" (1), even if the function cannot operate properly due to some functional problems. For example, if a function includes a CDR which is unable to get into a stable mode, the function shall still return a success code (1). Examples for returning a failure code (0) may include an invalid data type, a null pointer during run time, or anything that prevents the successful execution of the model’s code.

The authors of Algorithmic Models are encouraged to provide feedback to the EDA tool’s users through the various available messaging options about any difficulties the model encounters during execution, regardless of what the value of the function’s return code is.

**Statistical Training/Analysis Flow for**

**Channels with No Repeater**

The EDA tool shall make the following calls to the Tx and Rx AMI\_Init and AMI\_Impulse functions:

1. Tx AMI\_Init is called with
	1. (BCI\_State “Training”) (BCI\_Protocol “<name>”) (BCI\_ID “<my\_ ID>”) (BCI\_Training\_Mode “Dual|Impulse”)
	2. Impulse Matrix contains Impulse Response of Channel
	3. If the Tx executable model does not implement the BCI\_Protocol and BCI\_Training\_Mode, it returns “Error” in BCI\_State.
2. Rx AMI\_Init is called with
	1. (BCI\_State “Training”) (BCI\_Protocol “<name>”) (BCI\_ID “<my\_ID>”) (BCI\_Training\_Mode “Dual|Impulse”)
	2. Impulse Matrix contains Impulse Response output of Tx AMI\_Init
	3. If the Rx executable model does not implement the BCI\_Protocol and BCI\_Training\_Mode, it returns “Error” in BCI\_State.
	4. The EDA tool may analyze the results of Rx AMI\_Init
3. Tx AMI\_Impulse is called with the same impulse\_matrix used in the call to Tx AMI\_Init. The value of BCI\_parameters\_in shall be Null (0) on the first call to Tx AMI\_Impulse, and the value of BCI\_parameters\_out of the previous call to Rx AMI\_Impulse on subsequent calls to Tx AMI\_Impulse.
4. The Rx AMI\_Impulse is called using the impulse\_matrix output of Tx AMI\_Impulse. The value of BCI\_parameters\_in shall be set to the value of BCI\_parameters\_out of the previous call to Tx AMI\_Impulse.
5. Steps 3. And 4. Are repeated until the Rx AMI\_Impulse returns AMI Reserved Parameter BCI\_State = “Converged|Error|Fail”.
6. The EDA tool may analyze the results of last call to Rx AMI\_Impulse
7. The EDA tool amy proceed with Time Domain Training or Simulation

Note that the EDA tool shall set the value of BCI\_parameters\_in to the value of BCI\_parameters\_out of the previous call to an AMI\_Impulse function in the channel.

**Statistical Training/Analysis Flow for**

**Channels with Repeaters**

1. It is required that all AMI models from the primary Tx to the terminal Rx support BCI\_Training\_Mode “Impulse|Dual”.
2. The AMI\_Init flow is identical to the flow defined on page 264.
3. This same flow is repeated with the calls to AMI\_Init replaced by calls to AMI\_Impulse.
4. The BCI simulation terminates when BCI\_State=” Converged|Error|Fail” in the AMI\_parameters\_out of the terminal Rx AMI\_Impulse function.

Note that the EDA tool shall set the value of BCI\_parameters\_in to the value of BCI\_parameters\_out of the previous call to an AMI\_Impulse function in the channel.

**BACKGROUND INFORMATION/HISTORY:**

**Contributor Michael Mirmak, Intel Corp**