**BUFFER ISSUE RESOLUTION DOCUMENT (BIRD)**

**BIRD NUMBER:** *xxx*

**ISSUE TITLE:** *AMI Reserved Parameters for Buffer Directionality*

**REQUESTOR:**  *Michael Mirmak, Intel Corp.*

**DATE SUBMITTED:** *(draft 2 to IBIS-AMI Task Group on April 6, 2015)*

**DATE REVISED:**

**DATE ACCEPTED BY IBIS OPEN FORUM:**

**STATEMENT OF THE ISSUE:**

The 6.0 specification strongly implies that only input-only and output-only [Model]s may be associated with AMI data using the [Algorithmic Model] keyword pair. However, there is no explicit prohibition on using any model type with [Algorithmic Model] except Terminator, Series, and Series\_Switch.

The ibischk 6.01 parser correctly generates no errors if an [Algorithmic Model] keyword pair is associated with a model of Model\_type I/O. However, the association of an I/O buffer with either a Tx or Rx AMI file creates an ambiguous situation: the model, EDA tool, and user have no way currently to communicate, either in traditional IBIS or using AMI Reserved Parameters, the directionality state of the buffer at any one time.

**ANALYSIS PATH/DATA THAT LED TO SPECIFICATION:**

~~Two Reserved Parameters, AMI\_Model\_Type and AMI\_Model\_Direction, are proposed to define the model type and direction associated with a given Algorithmic Model. These are assumed to be consistent with the Model\_type defined for the associated analog [Model].~~

A new Reserved Parameters, AMI\_Model\_Type, is proposed to define the model type and supported directionality associated with a given Algorithmic Model. This is assumed to be consistent with the Model\_type defined for the associated analog [Model].

Ideally, an error would be generated for buffers, with associated Algorithmic Models, of Model\_type I/O, 3-state, I/O\_open\_drain, I/O\_open\_sink, I/O\_open\_source, I/O\_ECL, or 3-state\_ECL, where this Reserved Parameter is not present.

**ANY OTHER BACKGROUND INFORMATION:**

The intent of this Reserved Parameter is ensure the EDA tool is “aware” of the associated models’ capabilities, to prevent cases where a channel is connected only to Rx endpoints with no Tx in the channel, or only to Tx endpoints with no receiving or 3-state device capable of receiving present or configured to do so.

This parameter set is assumed to be unaffected by the Polarity [Model] subparameter.

Thanks to Walter Katz of Signal Integrity Software (SiSoft) for his suggestions in an earlier verison of this proposal.

Draft 2 incorporates rules for the two restrictions on DLL and .ami file support that were the consensus of the IBIS ATM Task Group:

1) DLLs may be configured to support both TX and RX directions in a single DLL, or separate DLLs may be provided for each direction supported by a model.

2) Each direction shall have a separate associated .ami file.

In addition, a new subparameter, “Direction”, is added for [Algorithmic Model]. The AMI\_Model\_Direction parameter is removed.

Add the following text under the “General Reserved Parameters” section:

…

*Parameter:* **AMI\_Model\_Type**

*Required:* No

*Descriptors*:

Usage: Info

Type: String

Format: Value

Default: <string\_literal>

Description:<string >

*Definition:* Tells the EDA tool whether the associated Algorithmic Model describes a buffer in transmitter (Tx) or receiver (Rx) mode of operation.

*Usage Rules:*

AMI\_Model\_Type accepts string literal values of “Tx” and “Rx”.

AMI\_Model\_Type is optional.

AMI\_Model\_Type is not permitted with all [Model] Model Types. EDA tools are assumed to check the [Model] Model\_Type of an analog buffer against the AMI\_Model\_Type of any associated [Algorithmic Model]. Table 1 below defines the [Model] Model\_Type circumstances in which AMI\_Model\_Type is permitted in the associated [Algorithmic Model], the matching [Model] Model\_type, and where the [Algorithmic Model] Direction subparameter is required.

In the absence of AMI\_Model\_Type, the buffer inherits the [Algorithmic Model] Direction subparameter of the associated [Algorithmic Model] keyword according to Table 2. Note that this does not eliminate potential ambiguities in model treatment during simulation.

AMI\_Model\_Type ensures consistent AMI parameter file and executable file interactions for I/O- or 3-state-capable buffers that can handle both Tx and Rx or Tx and high-impedance functions.

AMI\_Model\_Type is not legal as a Reserved\_Parameter in version 6.0 and earlier.

Only certain Reserved Parameters are consistent with each AMI\_Model\_Type. The rules for AMI\_Model\_Type consistency are shown in Table 1 below.

Table 1 – AMI\_Model\_Type and Reserved Parameter Interaction

| **Reserved Parameter** | **AMI\_Model\_Types Permitted** |
| --- | --- |
| Rx\_Clock\_PDF | Rx |
| Rx\_Clock\_Recovery\_DCD | Rx |
| Rx\_Clock\_Recovery\_Dj | Rx |
| Rx\_Clock\_Recovery\_Mean | Rx |
| Rx\_Clock\_Recovery\_Rj | Rx |
| Rx\_Clock\_Recovery\_Sj | Rx |
| Rx\_DCD | Rx |
| Rx\_Dj | Rx |
| Rx\_Noise | Rx |
| Rx\_Receiver\_Sensitivity | Rx |
| Rx\_Rj | Rx |
| Rx\_Sj | Rx |
| Tx\_DCD | Tx |
| Tx\_Dj | Tx |
| Tx\_Jitter | Tx |
| Tx\_Rj | Tx |
| Tx\_Sj | Tx |
| Tx\_Sj\_Frequency | Tx |
| DLL\_ID | Tx, Rx |
| DLL\_Path | Tx, Rx |
| Supporting Files | Tx, Rx |
| AMI\_Version | Tx, Rx |
| GetWave\_Exists | Tx, Rx |
| Ignore\_Bits | Tx, Rx |
| Init\_Returns\_Impulse | Tx, Rx |
| Max\_Init\_Aggressors | Tx, Rx |
| Use\_Init\_Output | N/A (illegal combination) |

Table 2 – AMI\_Model\_Type, [Algorithmic Model] Direction and [Model] Model\_Type Interaction

| **[Model] Model Type** | **[Algorithmic Model] Direction Permitted, with what Value(s)?** | **AMI\_Model\_Type Permitted, with what Value(s)?** |
| --- | --- | --- |
| Input  Input\_ECL | Optional; “Rx” is assumed if not present | Yes; Value shall be “Rx” |
| I/O  I/O\_open\_drain  I/O\_open\_sink  I/O\_open\_source  I/O\_ECL | Required; at least one “Tx” and “Rx” [Algorithmic Model] required | Yes; Value shall be “Tx” or “Rx” |
| Terminator | N/A (illegal) | N/A (illegal) |
| Output  Output\_ECL | Optional; “Tx” is assumed if not present | Yes; Value shall be “Tx” |
| 3-state  3-state\_ECL | Optional; “Tx” is assumed if not present | Yes; Value shall be “Tx” |
| Open\_sink  Open\_drain  Open\_source | Optional; “Tx” is assumed if not present | Yes; Value shall be “Tx” |
| Series | N/A (illegal) | N/A (illegal) |
| Series\_switch | N/A (illegal) | N/A (illegal) |
| Input\_diff | Optional; “Rx” is assumed if not present | Yes; Value shall be “Rx” |
| Output\_diff | Optional; “Tx” is assumed if not present | Yes; Value shall be “Tx” |
| I/O\_diff | Required; at least one “Tx” and “Rx” [Algorithmic Model] required | Yes; Value shall be “Tx” or “Rx” |
| 3-state\_diff | Optional; “Tx” is assumed if not present | Yes; Value shall be “Tx” |

*Other Notes:* This parameter prevents association of an Algorithmic Model with an incompatible analog model. AMI\_Model\_Type is assumed defined and fixed by the model author.

*Examples:*

(AMI\_Model\_Type (Usage Info) (Type String) (Value “Tx”)

(Description “Valid values are Tx, and Rx”)

)

Add the following updated text under the “Keyword Definitions” section of Chapter 10:

*Keywords:* [Algorithmic Model], [End Algorithmic Model]

*Required:* No

*Description:* Used to reference an executable model file and accompanying parameter definition file. This executable model file encapsulates signal processing functions, while the parameter definition file includes configuration information for the model and EDA tool. In the case of a receiver, the executable model file may additionally include clock and data recovery functions. The executable model file can receive and modify waveforms with the analog channel, where the analog channel consists of the transmitter output stage, the transmission channel itself and the receiver input stage. This data exchange is implemented through a set of software functions. The signature of these functions is elaborated in Section 10.2 of this document. The function interface must comply with the ANSI "C" language.

Note that, while the file is described here as an “executable model file”, the file is a compiled library of functions that may or may not be itself executable.

*Sub-Params:*  Executable, Direction

*Usage Rules:* The [Algorithmic Model] keyword must be positioned within a [Model] section ~~and it may appear only once for each [Model] keyword in a .ibs file~~. It is not permitted under the [Submodel] keyword or in [Model]s which are of Model\_type Terminator, Series or Series\_switch.

The [Algorithmic Model] always processes a single waveform regardless whether the model is single ended or differential. When the model is differential, the waveform passed to the [Algorithmic Model] must be a difference waveform.

[Algorithmic Model], [End Algorithmic Model]:

Begins and ends an algorithmic model section, respectively.

Executable:

Three entries follow the Executable subparameter on each line:

Platform\_Compiler\_Bits File\_Name Parameter\_File

The Platform\_Compiler\_Bits entry provides the name of the operating system, compiler and its version and the number of bits the executable model file is compiled for. It is a string without white spaces, consisting of three fields separated by an underscore (“\_”). The first field consists of the name of the operating system followed optionally by its version. The second field consists of the name of the compiler followed by optionally by its version. The third field is an integer indicating the platform architecture. If the version for either the operating system or the compiler contains an underscore, it must be converted to a hyphen “-”. This is so that an underscore is only present as a separation character in the entry.

The architecture entry can be either “32” or “64”. Examples of Platform\_Compiler\_Bits:

Linux\_gcc3.2.3\_32

Solaris5.10\_gcc4.1.1\_64

Solaris\_cc5.7\_32

Windows\_VisualStudio7.1.3088\_32

HP-UX\_accA.03.52\_32

The EDA tool will check for the compiler information and verify if the executable model file is compatible with the operating system and platform.

Multiple occurrences, without duplication, of Executable are permitted to allow for providing executable model files for as many combinations of operating system platforms and compilers for the same algorithmic model.

The File\_Name provides the name of the executable model file. The executable model file should be in the same directory as the.ibs file.

The Parameter\_File entry provides the name of the parameter definition file, which shall have an extension of .ami. This must be an external file and should reside in the same directory as the .ibs file and the executable model file. See Section 10.3 for details.

Direction:

The Direction subparameter accepts a single string argument, which may be either “Tx” or “Rx”. The subparameter is required if the Model\_type for the associated [Model] is “I/O”, “3-state”, “I/O\_open\_drain”, “I/O\_open\_sink”, “I/O\_open\_source”, “I/O\_ECL”, or “3-state\_ECL”. For any [Model] of types “I/O”, “I/O\_open\_drain”, “I/O\_open\_sink”, “I/O\_open\_source”, or “I/O\_ECL”, at least one [Algorithmic Model] of Direction “Tx” shall be present and at least one [Algorithmic Model] of Direction “Rx” shall be present. For any [Model] of types “3-state” or “3-state\_ECL”, Direction is optional, but only “Tx” as a Direction argument is permitted (no algorithmic model support for non-transmitting mode is provided). For all other Model\_types where [Algorithmic Model] is permitted, the Direction subparameter is optional. If the Direction parameter is optional and omitted, the direction of the associated [Algorithmic Model]s shall be assumed by the EDA tool to follow the [Model] Model\_type declaration.

It is assumed that the [Model] Model\_type, [Algorithmic Model] Direction, and .ami file AMI\_Model\_Type are consistent (e.g., that a [Model] of Model\_type I/O shall have associated [Algorithmic Model]s of Direction “Tx” and “Rx”, each with unique .ami file associations where the .ami files use “Tx” or “Rx” as AMI\_Model\_Types, respectively).

Multiple [Algorithmic Model] declarations may exist under a single [Model]. For any given [Model], each [Algorithmic Model] declaration shall refer to a unique .ami file (Parameter\_Name argument). Identical Executable File\_Name arguments may be used for multiple [Algorithmic Model] declarations under a single [Model], regardless of Direction. In other words, a single executable may be configured to process both transmit and receive waveform information and so may be used for both directions; unique parameter files are required for each direction, however.

The EDA tool is responsible for determining, through interaction with the user, the particular direction and associated files to use for a given simulation.

*Examples:*

Example of Receiver Model in [Algorithmic Model]:

[Algorithmic Model]

|

Executable Windows\_VisualStudio\_32 example\_rx.dll example\_rx\_params.ami

|

[End Algorithmic Model]

Example of Transmitter Model in [Algorithmic Model]:

[Algorithmic Model]

|

Executable Windows\_VisualStudio\_32 tx\_getwave.dll tx\_getwave\_params.ami

Executable Solaris\_cc\_32 libtx\_getwave.so tx\_getwave\_params.ami

|

[End Algorithmic Model]

Example of Bi-directional Model in [Algorithmic Model]:

[Algorithmic Model]  
|  
Direction TX | must be consistent with [Model\_Type]

Executable Windows\_VisualStudio\_32 tx\_getwave.dll tx\_getwave\_params.ami  
Executable Solaris\_cc\_32 libtx\_getwave.so tx\_getwave\_params.ami  
|  
[End Algorithmic Model]

[Algorithmic Model]  
|  
Direction RX | must be consistent with [Model\_Type]

Executable Windows\_VisualStudio\_32 rx\_getwave.dll rx\_getwave\_params.ami  
Executable Solaris\_cc\_32 libtx\_getwave.so rx\_getwave\_params.ami  
|  
[End Algorithmic Model]