10A AMI PARAMETER DEFINITION FILE STRUCTURE

INTRODUCTION

The information provided in this section is applicable to the content of the parameter definition file (.ami). Note that the rules described below deviate from the rules for .ibs files.

PARAMETER DEFINITION FILE ORGANIZATION

The .ami file must contain a distinct section or sub-tree named 'Reserved_Parameters' beginning and ending with parentheses. The file may also contain another section or sub-tree named 'Model_Specific', beginning and ending with parentheses. The sub-trees 'Reserved_Parameters' and 'Model_Specific' are branches of the root of the tree.

The Model Parameter File must be organized in the following way:

```
(my_AMIname                   | Name given to the Parameter file
      (Reserved_Parameters    | Required heading to start the
                              | required Reserve_Parameters
                              | section
            ...

            (Reserved parameter text starting with AMI_Version)
            ...
      )                       | End of Reserved_Parameters
                              | section
      (Model_Specific         | Required heading to start the
                              | optional Model_Specific section
            ...
            (Model specific parameter text)
            ...
      )                       | End of Model_Specific section
      (Description <string>)  | description of the model
                              | (optional)
)                             | End my_AMIname parameter file
```

General Rules and Notes: (THIS IS NOT A NEW SECTION, BUT CONTAINS INFORMATIN RELATED TO THE PARAMETER DEFINITION FILE ORGANIZATION)

The content of the parameter definition file (.ami) is case sensitive.

Only the pipe ("|") character is acceptable as a comment character regardless of what the calling IBIS file uses for the comment character.

The line length of the parameter definition file (.ami) is not limited to a specific number of characters.

The root name in the file may contain an arbitrary string and does not need to match the file name.

A white space in the parameter definition file (.ami) may be one or more space, tab, and/or line termination characters.

The ‘Reserved_Parameter’ section is required while the 'Model_Specific' section is optional. For AMI_Version “5.1” and above, the Reserved_Parameter sub-tree shall appear before the Model_Specific sub-tree. Sub-trees of these sub-trees can be in any order in the parameter file. The '|' character is the comment character. Any text after the '|' character until the end of the line will be ignored by the parser.

Notes:

1. Throughout the section, text strings inside the symbols “<” and “>” should be considered to be supplied or substituted by the model maker. Text strings inside “<” and “>” are not reserved and can be replaced.
2. Throughout the document, terms “long”, “double” etc. are used to indicate the data types in the C programming language as published in ISO/IEC 9899-1999.

PARAMETER NAME RULES AND SUMMARY

All leaves of the .ami file must begin with one of the following reserved words:

Type
Usage
Description
Default
<data_format> or Format <data_format>

A branch in the .ami file is an "AMI Parameter" if it contains the leaves Type, Usage, and any of the following leaves:

Default
<data_format> or Format <data_format>

and does not contain another branch. Multiple leaves containing the same reserved word are not allowed within an AMI Parameter branch. A branch which contains one or more sub-branches may only contain the (Description <string>) leaf/value pair in addition to the sub-branches. Each sub-branch of a branch must have a unique name.

All parameters must be in the following format:

(parameter_name
(Usage <usage>)
(Type <data_type>)
({Format} <data_format> <data>)

(Default <value>)
(Description <string>))

Notes:

1) The order of the entries is not important.
2) The word Format is optional as indicated by the curly braces "{" and "}" and may be ignored by EDA tools (the examples do not show the word Format).
3) Certain reserved parameter names allow only certain <data_format> selections, as described below.
4) The <data_format> selection of Value and Default are always mutually exclusive. Certain parameters may require Value or Default, but Value and Default are not allowed to be present together for the same parameter.
5) <data_format> is always required for selections other than Value.
6) Default is optional for <data_format> Range, List, Corner, Increment and Steps.
7) Default is not allowed for Usage Out parameters.
8) Default is not allowed for <data_format> Table, Gaussian, Dual-Dirac and DjRj.
9) Additional rules apply when <data_format> is Table. The format for <data> describes a set of rows containing data values. Each row has its set of column data values enclosed by parentheses '(' and ')'. Each row contains the same number of column values. Any or all of these columns may have different data types. For this case the <data_type> argument is either a list of data types (one for each column), or a single data type. If it is a single data type then this type shall be applied to all of the columns in each row.
10) <data_format> Corner is not allowed for Usage Out.
11) Description is optional.

RESERVED WORD RULES:

Note: Usage, Type, Format and Default and their allowed values are reserved names in the parameter definition file (.ami) discussed in the "KEYWORD DEFINITION" section.

Usage <usage>:

Required for model-specific parameters, where <usage> must be substituted by one of the following:

In

Parameter value is a required input to the AMI model

Out

Parameter value is coming from the AMI model

Info

Information for user or EDA platform

InOut

Parameter value is a required input to the AMI model The AMI model may return a different value.

Note that the purpose of Usage Out or InOut is to provide a mechanism for the Algorithmic Model to return values to the EDA tool to be used as described by the IBIS-AMI specification.

Type <data_type>:

Required, where <data_type> must be substituted by one of the following:

Float

Float numbers are in general represented by a floating point number that may be scaled using a decimal exponent. A floating point number is represented by the significant digits, and optionally a sign and decimal point. For example, -1.23e-3, 123e-3, 1.23, 1 are all of type float

Scaling factors or suffixes, such as p, n, etc., are not permitted in the parameter definition file (.ami). Scientific and floating point notation is permitted.

Integer

Integers are numbers which are written without a fractional or decimal component, and fall within -2147483648 and 2147483647. If scientific notation is used then the exponent must be positive. For example, 65, 7, and -756, 123e3 are integers, but 1.6, 123e99 or 123e-2 are not integers.

String

String is a sequence of ASCII characters enclosed in double quotes ("). As defined in ANSI Standard X3.4-1986, the allowable ASCII characters consist of hexadecimal 20, 21, 23 to 7E, and the ASCII control characters 09 (HT), 0A (LF), and 0D (CR) for defining tabs and line termination sequences. The double quote character 22 (") is not allowed inside string literals.

Boolean

(True/False)

Tap

(For use by TX and RX equalizers)

A tapped delay line can be described by creating a separate parameter for each tap weight and grouping all the tap weights for a given tapped delay line in a single parameter group which is given the name of the tapped delay line. If in addition the individual tap weights are each given a name which is their tap number (i.e., "-1" is the name of the first precursor tap, "0" is the name of the main tap, "1" is the name of the first postcursor tap, etc.) and the tap weights are declared to be of type Tap, then the EDA platform can assume that the individual parameters are tap weights in a tapped delay line, and use that assumption to perform tasks such as optimization. The model developer is responsible for choosing whether or not to follow this convention.

The type Tap implies that the parameter takes on floating point values. Note that if the type Tap is used and the parameter name is not a number, this is an error condition for which EDA platform behavior is not specified.

UI

Unit Interval, 1 UI is the inverse of the data rate frequency, for example 1 UI of a channel operating at 10 Gb/s is 100 ps. . UI values are unitless. The parameter may take on either floating point or integer values.

Format <data_format> <data> or **<data_format>** <data>:

Where Format is optional and <data_format> and <data> are required. <data_format> and <data> must be substituted with one of the following::

Value <value>

Single value data.

The user may assign any value without any restrictions within the constraints of the Type of the variable. Note that Value and Default (defined below) are mutually exclusive, and must not be used together for the same parameter.

Range <typ value> <min value> <max value>

This defines a continuous range for which the user may select any value greater than or equal to <min value> and less than or equal to <max value> within the constraints of the Type of the variable

List <default value> <value> <value> <value> ... <value>

This defines a discrete set of values from which the user may select one value

Corner <typ value> <slow value> <fast value>

Corner is not allowed with Usage Out parameters. The selection of one value is automatically carried out by the EDA tool based on its internal simulation corner setting

Increment <typ> <min> <max> <delta>

where min <= typ <= max and delta is always positive. After expansion, the expanded values of the parameter are typ + N*delta where N is any positive or negative integer value provided by the EDA tool during the expansion process so that: min <= expanded values <= max

Steps <typ> <min> <max> <# steps>

Treat exactly like Increment with <delta> == (<max>-<min>)/<# steps>

Table and optional leaf **Labels**

The Format Table states that this parameter consists of one or more columns of data, with each row delimited by parentheses '(' and ')'. All rows must contain the same number of entries (columns). At least one row must be included. Default is illegal when Format Table is used.

The column entries must be of Type Float, UI, Integer, String or Boolean.

Type Tap is illegal. If only one Type is provided, then all Table entries shall be of the specified type.

(Type <type>)

For Table only, Type can also be used to designate the entries for each column. In this case, type entries shall be given for each column in the Table:

(Type <type1> <type2> <type3> ...)

Labels is an optional leaf within Table and it is followed by a String entry for each column in the Table. Quoted null entries are permitted. Labels shall be positioned immediately before the first row in a Table and are of the for

(Labels <"label1"> <"label2"> <"label3"> ...)

If Table is used for a reserved parameter, the rules for the number of columns and their meaning are described in the Reserved_Parameters section.

The EDA tool and the algorithmic model must always transmit the entire contents of a table through the AMI_parameters_in or AMI_parameters_out string (defined in Section 10 and illustrated in the examples below). Only the parameter_name and values in the table are included in the parameter string. The values in each row of the table are flattened into a single row of values without the parentheses surrounding each row when producing the parameter string.

For Usage Out and InOut, the number of rows returned by the executable model may differ from the number of rows documented in the .ami file, but a minimum of one row must be returned. Multiple GetWave calls are not required to return the same number of rows. For Usage Out, a one-row Table is required in the .ami file to serve as a template for single and multi-row tables. This can be used by the EDA tool to reconstruct a sequence of data values returned by the executable model into a table with as many rows as needed, and optionally for parameter initialization before being replaced by the actual Table data returned by the executable model.

Examples:

Single Row Table where all numbers are Float (note that '1' is a legal float entry):

```
(fwd (Usage In) (Type Float)
      (Table
            (1 -0.169324  1.40308  0.33024)
      )
      (Description "Application Description")
)
```

The EDA tool sends to the executable model in the parameter string:

```
(fwd  1 -0.169324  1.40308  0.33024)
```

Single Row, all numbers would be encoded as integers by the EDA tool:

```
(bit_pattern (Usage In) (Type Integer)
      (Table
```

```
            (1 1 1 1 0 0 0 1 0 0 1)
      )
      (Description "Bit Pattern Sequence")
)
```

The EDA tool sends to the executable model in the parameter string:

```
(bit_pattern  1 1 1 1 0 0 0 1 0 0 1)
```

Multiple row Table example with Labels:

The optional Labels line is added above the first row. It is not sent or returned to/from the executable model, but is available to the EDA tool for information.

```
(poles (Usage InOut) (Type Float)
      (Table
      (Labels "complex_conj_flag" "real_part" "imag_part")
            (1    -5e8        0)
            (2    -9.4e8      8.3e8)
            (1    -7.3e8      0)
      )
      (Description "Two real and two complex poles")
)
```

The EDA tool sends to the executable model in the parameter string:

```
(poles  1  -5e8  0  2  -9.4e8  8.3e8  1  -7.3e8  0)
```

An updated set with a different number of pole and row entries can be returned with a similar sequence to be converted back into the same or a different number of rows.

Type used to specify the type entry for each column (the example above is modified with Type entries for each column):

```
(poles (Usage InOut) (Type Integer Float Float)
      (Table
      (Labels "complex_conj_flag" "real_part" "imag_part")
            (1    -5e8        0)
            (2    -9.4e8      8.3e8)
            (1    -7.3e8      0)
      )
      (Description "Two real and two complex poles")
)
```

The encoding in the previous example is sent to the EDA tool and returned to the executable model.

Example of two rows with Type entries for each column (the fourth column numbers are interpreted as UI values):

```
(pdf (Usage In) (Type Integer Integer Float UI Float)
      (Table
      (Labels "Row" "Bin number" "Time" "UI" "Probability")
            (1   -5   -5e-9   -1     1e-5)
            (2   -4   -4e-9   -0.8   1e-4)
      )
      (Description "Probability Distribution Function Table")
)
```

The EDA tool sends to the executable model in the parameter string:

```
 (pdf  1  -5  -5e-9  -1  1e-5  2  -4  -4e-9  -0.8  1e-4  ...)
```

Example above, but with Usage Out (only one row is necessary in the .ami file):

```
(pdf (Usage Out) (Type Integer Integer Float UI Float)
      Table
      (Labels "Row" "Bin number" "Time" "UI" "Probability")
           (1   -5   -5e-9    -1     1e-5)
      )
      (Description "Probability Distribution Function Table")
)
```

One row is provided as a template, but the executable model can return, in the parameter string, different data and more than one row such as shown.

```
(pdf 1 -6 -6e-9 -1.2 3e-6 2 -5 -5e-9 -1 9e-6 ...)
```

Gaussian <mean> <sigma>

Dual-Dirac <mean> <mean> <sigma>

Composite of two Gaussian

DjRj <minDj> <maxDj> <sigma>

Convolve Gaussian (sigma) with uniform Modulation PDF

Default <value>:

When used with single value data, Default and Value are mutually exclusive, and must not be used together for the same parameter. In these situations, Default is a synonym of Value and does not imply any additional meaning or actions. Default is not allowed for any Usage Out parameter types, and Table, Gaussian, Dual-Dirac and DjRj. Default is optional for Range, List, Corner, Increment and Steps. When Default is specified for any of these parameter types, it shall be used by the EDA tool to pick one value from all the possibilities for that parameter if the user does not make such a selection.

If a Default <value> is specified, its value must have the same Type as the parameter. For example, if Type is Boolean, <value> must be either True or False, if Type is Integer, <value> must be an integer. Also, if Default is specified, <value> must be a member of the set of allowed values of the parameter. If Default is not specified, the default value of the parameters will be the <typ> value.

Description <string>:

The string following Description may describe a reserved parameter, a model specific parameter, or the Algorithmic model itself. This string is used by the EDA platform to convey information to the end-user. Description <string> is optional, but its usage is highly recommended for describing the Algorithmic model and the model specific parameters of the Algorithmic model.

The Description string may span multiple lines, but it is recommended that the text contained in the Description string should not exceed 120 characters per line.

The location of Description will determine what the parameter or model is being described.

COMBINATION AND CORNER RULES

For Usage Out parameters, ({Format} <data_format> <data>) may be ignored by the EDA tool, except when <data_format> is Table where at least a one-row Table is required in <data> to serve as a template for single and multi-row tables.

Formats Value, Corner and List can be of any defined Types whereas Formats Range, Increment and Steps can be of Types Float, UI, Integer and Tap only. Formats Gaussian, Dual-Dirac and DjRj can only be of Types Float and UI. For Format Table, the column entries must be of Type Float, UI, Integer, String or Boolean. Type Tap is illegal for Format Table. If only one Type is provided, then all Table entries shall be of the specified type. Type can also be used to designate the entries for each column in the table. More information is provided in the definition of the Table format.

Note that modeling and simulating different corner cases is a fundamental concept in IBIS. For each model instance, the EDA tool will make use of either the "Typ", "Min" or "Max" data provided in the IBIS file, according to the user's simulation setup.

As described in the "Notes on Data Derivation Method" section of this document, the "Min" and "Max" data for the I-V tables and their corresponding voltage reference keywords, [Ramp] and V-T tables represent the slow and fast behavior of the device, respectively. Following the conservative approach, the "Max" value of C_comp represents the slow, and the "Min" value of C_comp represents the fast behavior of the device.

For IBIS-AMI parameters defined as Format Corner, the EDA tool will pick one of the three supplied values (<typ value>, <slow value>, <fast value>) in the .ami file for any given model instance. This selection is governed by the same internal corner variable in the EDA tool that controls the selection of the "Typ", "Min", "Max" model data. <typ value> corresponds to "Typ", <slow value> corresponds to "Min" (slow or weak performance) and <fast value> corresponds to "Max" (fast or strong performance). For AMI parameters, <slow value> does not have to be less than <fast value>.

For AMI parameter type 'Range', 'Increment' and 'Steps' <min value>, <max value> does not imply slow and fast corners, and the user may select any value provided by these parameters regardless of what corner is used for the simulation. If the user does not make a selection for parameter types 'Range', 'List', 'Increment' and 'Steps', the EDA tool shall automatically use the value defined by Default, if it exists, or the <typ value> otherwise (regardless of what corner is used for the simulation).

When a [Model] that is associated with any of the pins listed under the [Diff Pin] keyword contains the [Algorithmic Model] keyword, the tdelay_*** parameters in the fourth, fifth and sixth columns of the [Diff Pin] keyword are ignored in AMI channel characterization simulations, i.e., they are treated as if their value would be zero.

PROCESSING AND PASSING PARAMETER STRING RULES

The parameter string passed in and out of the AMI executable model (described in the Sections **Error! Reference source not found.**, **Error! Reference source not found.** and **Error!**

Reference source not found.) is formatted the same way as the tree data structure in the .ami file with the following exceptions.

The EDA tool must process the content of the .ami file so that

1) the 'Reserved_Parameters' and 'Model_Specific' branch names and their associated open and close parentheses "()" are not included in the AMI_parameters_in string, and

2) the AMI Parameter branches with Usage In or Usage InOut are converted to leaf/value pairs for the AMI_parameters_in string, possibly incorporating user selections. In this conversion each AMI parameter branch name becomes a leaf name in the AMI_parameters_in string and each leaf name is followed by a white space, a value and a closing parenthesis ")".

The AMI executable model must generate a parameter string that is consistent with the content of the .ami file so that

1) the 'Reserved_Parameters' and 'Model_Specific' branch names and their associated open and close parentheses "()" are not included in the AMI_parameters_out string, and

2) the AMI Parameter branches Usage Out or Usage InOut are returned as leaf/value pairs in the AMI_parameters_out string.

The EDA tool must pass a string to the AMI executable model through the AMI_parameters_in argument. This string must contain all of the leaf/value pair formatted Usage In and Usage InOut AMI parameters if there are any defined in the .ami file. No other information may be included in this string. The string must always include the root name of the parameter tree, even if there are no parameters to pass to the algorithmic model.

The AMI executable model must return a string to the EDA tool through the AMI_parameters_out argument. This string must contain all of the leaf/value pair formatted Usage InOut and Usage Out AMI parameters if there are any defined in the .ami file. No other information may be included in this string. The string must always include the root name of the parameter tree, even if there are no parameters to return to the EDA tool.

For Usage In, the value in the "AMI Parameter" leaves are determined by the EDA tool based on the "AMI Parameter" branches in the .ami file. For Usage Out, the value in the "AMI Parameter" leaves are determined by the Algorithmic Model. For Usage InOut, the value in the "AMI Parameter leaves are first determined by the EDA tool based on the "AMI Parameter" branches in the .ami file and passed into the Algorithmic Model which may return a new value in the "AMI Parameter" leaves after some processing.

RESERVED PARAMETERS REFERENCE

The Model parameter file must have a sub-tree with the heading ‘Reserved_Parameters’. This sub-tree shall contain all the reserved parameters for the model.

The following reserved parameters are used by the EDA tool and are required if the [Algorithmic Model] keyword is present. The entries following the reserved parameter names determine their usage, type and default value. Their value may be defined using either Default or Value but not both. Description is optional.

Parameter: **AMI_Version**

Required: Yes for AMI_Version “5.1” and above, illegal before AMI_Version “5.1”

Descriptors:

Usage:	Info
Type:	String
Format:	Value
Default:	Value
Description:	<string_literal>

Definition: Tells EDA platform what version of the AMI modeling language is supported.

Usage Rules: AMI_Version is required in the .ami parameter files of AMI models which are written in compliance with the IBIS Version 5.1 or later specification(s), but it is not allowed in the .ami parameter files of AMI models which are written in compliance with the IBIS Version 5.0 specification. When required, this parameter must be the first parameter defined in the Reserved_Parameters branch of the .ami file.

The value of this parameter shall be “5.1” or greater for AMI models written in compliance with the IBIS Version 5.1 or later specifications. The absence of AMI_Version indicates that the AMI model was written in compliance with the IBIS Version 5.0 specification.

The version numbers of .ibs files and AMI models do not have to match. The EDA tool is expected to execute the AMI model according to the rules of the specification which corresponds to AMI file version number.

Other Notes: For AMI_Version “5.1” or later.

Throughout this document, the shorthand, AMI_Version <version_number>, is used to indicate the minimum AMI_Version level that is supported. If the AMI_Version is not used, then the AMI model is processed at the level defined in [IBIS Ver] 5.0. In some cases, it will be noted that a rule has changed, has become more restrictive or more relaxed for a specified AMI_Version level.

Examples:

```
(AMI_Version (Usage Info)(Type String)(Value “5.1”)
      (Description “Valid for AMI_Version 5.1 and above”)
)

(AMI_Version (Usage Info)(Type String)(Default “5.1”)
      (Description “Valid for AMI_Version 5.1 and above”)
)
```

Parameter: **Init_Returns_Impulse**

Required: Yes

Descriptors:

Usage: Info

Type: Boolean

Format: Value

Default: <Boolean_literal>

Description: <string_literal>

Definition: Tells EDA platform whether the AMI_Init function returns a modified impulse response.

Usage Rules: When the Boolean_literal value is set to ‘True’, the model returns the convolution of the impulse response with the impulse response of the equalization.

Other Notes:

Examples:

```
(Init_Returns_Impulse (Usage Info)(Type Boolean)(Default True)
      (Description “Valid for all AMI_Version levels”)
)

(Init_Returns_Impulse (Usage Info)(Type Boolean)(Value True)
      (Description “Valid for all AMI_Version levels”)
)
```

Parameter: **GetWave_Exists**

Required: Yes

Descriptors:

Usage: Info

Type: Boolean

Format: Value

Default: <Boolean_literal>

Description: <string_literal>

Definition: Tells EDA platform whether the AMI_GetWave is implemented in this model

Usage Rules: Note that if Init_Returns_Impulse is set to ‘False’, then GetWave_Exists MUST be set to ‘True’.

Other Notes: Either Value or Default (but not both) are required.

Examples:

```
(GetWave_Exists (Usage Info)(Type Boolean)(Default True)
      (Description “Valid for all AMI_Version levels”)
)

(GetWave_Exists (Usage Info)(Type Boolean)(Value True)
      (Description “Valid for all AMI_Version levels”)
```

)

Parameter: **Use_Init_Output**
Required: No, and legal only before AMI_Version “5.1”

Descriptors:

Usage: Info

Type: Boolean

Format: Value

Default: <Boolean_literal>

Description: <string_literal>

Definition: Tells EDA platform whether the AMI_GetWave is implemented in this model

Usage Rules: When Use_Init_Output is set to ‘True’, the EDA tool is instructed to use the output impulse response from the AMI_Init function when creating the input waveform presented to the AMI_GetWave function.

If the Reserved Parameter, Use_Init_Output, is set to ‘False’, EDA tools will use the original (unfiltered) impulse response of the channel when creating the input waveform presented to the AMI_GetWave function.

The algorithmic model is expected to modify the waveform in place.

Use_Init_Output is optional. The default value for this parameter is ‘True’.

If Use_Init_Output is ‘False’, GetWave_Exists must be ‘True’.

Other Notes:

Examples:

```
(Use_Init_Output (Usage Info)(Type Boolean)(Default True)
      (Description “Use_Init_Output if AMI_Version is omitted”)
)
```

The following reserved parameters are optional. If the following parameters are not present, the values are assumed as ‘0’.

Parameter: **Max_Init_Aggressors**
Required: No

Descriptors:

Usage: Info

Type: Integer

Format: Value

Default: <numeric_literal>

Description: <string_literal>

Definition: Tells the EDA platform how many aggressor Impulse Responses the AMI_Init function is capable of processing.

Usage Rules: Its value is assumed ‘0’ if Max_Init_Aggressors is not present.

Other Notes:

Examples:

```
(Max_Init_Aggressors (Usage Info)(Type Integer)(Default 5)
      (Description “Valid for all AMI_Version levels”)
)

(Max_Init_Aggressors (Usage Info)(Type Integer)(Value 5)
      (Description “Valid for all AMI_Version levels”)
)
```

Parameter: **Ignore_Bits**
Required: No

Descriptors:

Usage:	Info
Type:	Integer
Format:	Value
Default:	<numeric_literal>
Description:	<string_literal>

Definition: Tells the EDA platform how long the time variant model takes to complete initialization.

Usage Rules: This parameter is meant for AMI_GetWave functions that model how equalization adapts to the input stream. The value in this field tells the EDA platform how many bits of the AMI_GetWave output should be ignored.

Its value is assumed ‘0’ if Ignore_Bits is not present.

Other Notes:

Examples:

```
(Ignore_Bits (Usage Info)(Type Integer)(Default 1000)
      (Description “Valid for all AMI_Version levels”)
)

(Ignore_Bits (Usage Info)(Type Integer)(Value 1000)
      (Description “Valid for all AMI_Version levels”)
)
```

Jitter and Noise Reserved Parameters:

Tx-only reserved parameters: Tx_Jitter and Tx_DCD

These reserved parameters only apply to Tx models. There parameters are optional. If these parameters are not specified the values default to no jitter specified in the model (“0” jitter).

Parameter: **Tx_Jitter**
Required: No

Descriptors:

Usage:	Info, Out
Type:	Float, UI
Format:	Gaussian, Dual-Dirac, DjRj, Table
Default:	(Illegal)
Description:	<string_literal>

Definition: Tells EDA platform how much jitter exists at the input to the transmitter's analog output buffer.

Usage Rules:

Other Notes:

Examples:

```
(Tx_Jitter (Usage Info)(Type Float)
      (Gaussian <mean> <sigma>))

(Tx_Jitter (Usage Info)(Type Float) (Dual-Dirac <mean> <mean> <sigma>)
)

(Tx_Jitter (Usage Info)(Type Float)
      (DjRj <minDj> <maxDj> <sigma>)
)

(Tx_Jitter (Usage Info)(Type Integer Float Float)
      (Table
      (Labels  “Row_No” Time” “Probability”)
            (-5  -5e-12  1e-10)
            (-4  -4e-12  3e-7)
            (-3  -3e-12  1e-4)
            (-2  -2e-12  1e-2)
            (-1  -1e-12  0.29)
            (0    0      0.4)
            (1    1e-12  0.29)
            (2    2e-12  1e-2)
            (3    3e-12  1e-4)
            (4    4e-12  3e-7)
            (5    5e-12  1e-10)
      )
)
```

Parameter: **Tx_DCD**
Required: No

Descriptors:

Usage:	Info, Out
Type:	Float, UI
Format:	Value, Range, Corner, List, Increment, Steps
Default:	<numeric_literal>

Description: <string_literal>

Definition: Tx_DCD (Transmit Duty Cycle Distortion) tells the EDA platform the maximum percentage deviation of the duration of a transmitted pulse from the nominal pulse width.

Usage Rules:

Other Notes:

Examples:

```
(Tx_DCD (Usage Info)(Type Float)
      (Range <typ> <min> <max>))
```

Rx-only reserved parameters: Rx_Clock_PDF and Rx_Receiver_Sensitivity

These reserved parameters only apply to Rx model. These parameters are optional. If the parameters are not specified, the values default to '0'.

Parameter: **Rx_Clock_PDF**
Required: No

Descriptors:

Usage: Info, Out

Type: Float, UI

Format: Gaussian, Dual-Dirac, DjRj, Table

Default: (Illegal)

Description: <string_literal>

Definition: Tells EDA platform the probability density function of the recovered clock.

Usage Rules:

Other Notes:

Examples:

```
(Rx_Clock_PDF (Usage Info)(Type Float)
      (Gaussian <mean> <sigma>))

(Rx_Clock_PDF (Usage Info)(Type Float)
      (Dual-Dirac <mean> <mean> <sigma>”))

(Rx_Clock_PDF (Usage Info)(Type Float)
      (DjRj <MinDj> <MaxDj> <sigma>))

(Rx_Clock_PDF (Usage Info)(Type Integer Float Float)
      (Table
      (Labels  “Row_No” Time” “Probability”)
            (-5  -5e-12  1e-10)
            (-4  -4e-12  3e-7)
            (-3  -3e-12  1e-4)
            (-2  -2e-12  1e-2)
            (-1  -1e-12  0.29)
            (0    0      0.4)
```

```
            (1    1e-12  0.29)
            (2    2e-12  1e-2)
            (3    3e-12  1e-4)
            (4    4e-12  3e-7)
            (5    5e-12  1e-10)
      )
)
```

Parameter: **Rx_Receiver_Sensitivity**
Required: No

Descriptors:

Usage:	Info, Out
Type:	Float
Format:	Value, Range, Corner, List, Increment, Steps
Default:	<numeric_literal>
Description:	<string_literal>

Description: Tells the EDA platform the voltage needed at the receiver data decision point to ensure proper sampling of the equalized signal.

Usage Rules:

Other Notes:

Examples:

In the example below, 100 mV (above +100 mV or below -100 mV is needed to ensure the signal is sampled correctly).

(make the example for above description and make other examples for other Formats)

```
(Rx_Receiver_Sensitivity (Usage Info)(Type Float)
      (Value <value>))

(Rx_Receiver_Sensitivity (Usage Info)(Type Float)
      (List <default value> <value> <value> … <value>)

(Rx_Receiver_Sensitivity (Usage Info)(Type Float)
      (Range <typ> <min> <max>))
(Rx_Receiver_Sensitivity (Usage Info)(Type Float)(Corner 0.0 0.1 -0.1)
      (Corner <typ> <min> <max>))
```

(Is 0.1 for slow corner and -0.1 for fast corner?)

TABLES SUMMARIZING THE RESERVED PARAMETER AND DATA TYPE RULES

The tables below summarize the valid combinations of Reserved Parameters, defaults, Data Types and Data Formats.

Table 1 - General Rules and Allowed Usage for Reserved Parameters

Reserved Parameter	General Rules		Allowed Usage			
	Required	Default	Info	In	Out	InOut
AMI_Version[1]	Yes	--	X			
Init_Returns_Impulse	Yes	--	X			
GetWave_Exists	Yes	--	X			
Use_Init_Output[2]	No	True	X			
Ignore_Bits	No	0	X			
Max_Init_Aggressors	No	0	X			
Tx_Jitter	No	No Jitter	X		X	
Tx_DCD	No	0	X		X	
Rx_Receiver_Sensitivity	No	0	X		X	
Rx_Clock_PDF	No	Clock Centered	X		X	

1) Required for IBIS Version 5.1 and above as the first reserved parameter, and illegal for IBIS Version 5.0
2) Illegal for Version 5.1 and above

Table 2 - Allowed Data Types for Reserved Parameters

Reserved Parameter	Data Type				
	Float	UI	Integer	String	Boolean
AMI_Version[1]				X	
Init_Returns_Impulse					X
GetWave_Exists					X
Use_Init_Output[2]					X
Ignore_Bits			X		
Max_Init_Aggressors			X		
Tx_Jitter	X	X			
Tx_DCD	X	X			
Rx_Receiver_Sensitivity	X				
Rx_Clock_PDF	X	X			

1) Required for IBIS Version 5.1 and above as the first reserved parameter, and illegal for IBIS Version 5.0
2) Illegal for Version 5.1 and above

Table 3 - Allowed Data Formats for Reserved Parameters

Reserved Parameter	Data Format									
	Value	Range	Corner	List	Increment	Steps	Gaussian	Dual-Dirac	DjRj	Table
AMI_Version[1]	X									
Init_Returns_Impulse	X									
GetWave_Exists	X									

Use_Init_Output[2]	X									
Ignore_Bits	X									
Max_Init_Aggressors	X									
Tx_Jitter							X	X	X	X
Tx_DCD	X	X	X	X	X	X				
Rx_Receiver_Sensitivity	X	X	X	X	X	X				
Rx_Clock_PDF							X	X	X	X

1) Required for IBIS Version 5.1 and above as the first reserved parameter, and illegal for IBIS Version 5.0
2) Illegal for Version 5.1 and above

Table 4 summarizes the relationships between the different Format and Data Types for Reserved or Model Specific Parameters.

Table 4 - Allowed Data Types for Format Values

Format	Data Type					
	Float	**UI**	**Integer**	**String**	**Boolean**	**Tap**
Value	X	X	X	X	X	X
Range	X	X	X			X
Corner	X	X	X	X	X	X
List	X	X	X	X	X	X
Increment	X	X	X			X
Steps	X	X	X			X
Gaussian	X	X				
Dual-Dirac	X	X				
DjRj	X	X				
Table	X	X	X	X	X	

MODEL SPECIFIC PARAMETERS

The Following section describes the user defined parameters. The algorithmic model expects these parameters and their values to function appropriately. The model maker can specify any number of user defined parameters for their model. The user defined parameter section subtree must begin with the reserved parameter 'Model_Specific'.

TAPPED DELAY LINE FULL EXAMPLE

(This is repeated from the Tap definition, but the placement and text is to be resolved later)

A tapped delay line can be described by creating a separate parameter for each tap weight and grouping all the tap weights for a given tapped delay line in a single parameter group which is

given the name of the tapped delay line. If in addition the individual tap weights are each given a name which is their tap number (i.e., "-1" is the name of the first precursor tap, "0" is the name of the main tap, "1" is the name of the first postcursor tap, etc.) and the tap weights are declared to be of type Tap, then the EDA platform can assume that the individual parameters are tap weights in a tapped delay line, and use that assumption to perform tasks such as optimization. The model developer is responsible for choosing whether or not to follow this convention.

The type Tap implies that the parameter takes on floating point values. Note that if the type Tap is used and the parameter name is not a number, this is an error condition for which EDA platform behavior is not specified.

Example:

```
(mySampleAMI                            | Name given to the Parameter file
  (Description "Sample AMI File")
  (Reserved_Parameters                  | Required heading
    (AMI_Version (Usage Info) (Type String) (Value “5.1”)
      (Description “Valid here only for AMI_Version ‘5.1’ and above”)
    (Ignore_Bits (Usage Info) (Type Integer) (Default 21)
      (Description "Ignore 21 Bits"))
    (Max_Init_Aggressors (Usage Info) (Type Integer)(Default 25))
    (Init_Returns_Impulse (Usage Info) (Type Boolean)(Default True))
    (GetWave_Exists (Usage Info) (Type Boolean) (Default True))
  )                                     | End Reserved_Parameters

  (Model_Specific                       | Required heading
    (txtaps
      (-2 (Usage InOut)(Type Tap) (Range 0.1 -0.1 0.2)(Default 0.1)
          (Description "Second Precursor Tap"))
      (-1 (Usage InOut)(Type Tap) (Range 0.2 -0.4 0.4)(Default 0.2)
          (Description "First Precursor Tap"))
      (0  (Usage InOut)(Type Tap) (Range 1 -1 2)(Default 1)
          (Description "Main Tap"))
      (1  (Usage InOut)(Type Tap) (Range 0.2 -0.4 0.4)(Default2 0.2)
          (Description "First Post cursor Tap"))
      (2  (Usage InOut)(Type Tap) (Range 0.1 -0.1 0.2)(Default 0.1)
          (Description "Second Post cursor Tap"))
    )                                   | End txtaps
    (tx_freq_offset (Range 1 0 150) (Type UI) (Default 0))
  )                                     | End Model_Specific
)                                       | End SampleAMI
```