****
****

BIRD ID#: 119
ISSUE TITLE: IBIS-AMI New Reserved Parameters
AUTHOR: Walter Katz, Mike Steinberger, Todd Westerhoff, SiSoft
DATE SUBMITTED: October 1, 2010
DATE REVISED:
DATE ACCEPTED BY IBIS OPEN FORUM:

****
****

STATEMENT OF THE ISSUE:

Model developers and EDA vendors building IBIS-AMI models using the IBIS 5.0 specification have come across a number of modeling issues that are not addressed in IBIS 5.0. In order to deliver models and EDA tools that meet end-user demands for model accuracy and functionality, EDA vendors have defined "extensions" to add new capabilities to IBIS-AMI models. Unfortunately, EDA vendors have had to use proprietary (and different) syntax to add these capabilities to models, limiting model portability between different EDA tools.

This BIRD proposes new syntax for the .ami control file that improves model functionality and accuracy. Including this syntax in the IBIS standard will allow creation of accurate, compliant IBIS-AMI models that are readily portable between commercial EDA simulators.

The parameters defined in this document are to be added in Section 6c of the IBIS 5.0 specification as new Reserved_Parameters. The new parameters are defined in the following different categories:

Data Management & Simulation Control
Supporting_Files, DLLPath, DLLid, Samples_Per_Bit

Broadband Analog Model
Tstonefile, Nodemap

Equivalent-Circuit Analog Model
Voh, Vol, Rt, Rs, Cc, Vt, Tr, Tf, Trf, Rd, Cd,
Voh_L, Vol_L, Rt_L, Rs_L, Cc_L, Tr_L, Tf_L,
Voh_H, Vol_H, Rt_H, Rs_H, Cc_H, Tr_H, Tf_H

Jitter, Noise and Clock Modeling
Tx_Rj, Tx_Sj, Tx_Sj_frequency, Rx_Clock_Recovery_Mean,
Rx_Clock_Recovery_Rj, Rx_Clock_Recovery_Sj,
Rx_Clock_Recovery_DCD, Rx_Rj, Rx_Sj, Rx_DCD, Rx_Noise

The following parameter exists in the IBIS 5.0 specification but its definition is replaced using the text in this BIRD:

Tx_DCD

The following new keywords allow the construction of Dependency Tables in IBIS-AMI models. Dependency Tables define a relationship between one or more input parameters and one or more output parameters. Note that in this context, "input parameter" and "output parameter" refer to inputs and outputs of the relationship being defined, not to whether the parameters in question are used as inputs or outputs by the algorithmic model itself. Dependency Tables make IBIS-AMI models easier for the systems designer by allowing complex relationships between related parameters to be automatically defined as part of the model.

Dependency, Parameter, Default_Row

Note that all of the parameters defined in this BIRD **may** be declared in the Model_Specific section of the .ami file to allow the use of some legacy models. However, using these parameters in the Model_Specific section of the .ami file is considered legacy use and will likely be deprecated in IBIS versions beyond 5.1.

****

On page 146 replace:

```
|               Tx_DCD:
|
|               Tx_DCD (Transmit Duty Cycle Distortion) can be of Usage Info
|               and Out.  It can be of Type Float and UI and can have Data
|               Format of Value, Range and Corner.  It tells the EDA platform
|               the maximum percentage deviation of the duration of a
|               transmitted pulse from the nominal pulse width.  Example of
|               TX_DCD declaration is:
|
|               (Tx_DCD (Usage Info)(Type Float)
|                       (Format Range <typ> <min> <max>))
```

with:

```
|               Tx_DCD:
|
|               Tx_DCD (Transmit Duty Cycle Distortion) can be of Usage Info
|               or Out.  It can be of Type Float and UI and can have Data
|               Format of Value, Range and Corner.  It defines half the peak
|               to peak clock duty cycle distortion, in seconds or UI, to be
|               added to the behavior implemented directly by the transmitter
|               model.
|
|               Example of TX_DCD declaration is:
|
|       (Tx_DCD (Usage Info)(Corner 0.008 0.016 0.005)(Type UI)
|          (Description "TX Duty Cycle Distortion in UI.")
|       )
```

The following text is added immediately before Table 1 on page 148:

Data Management & Simulation Control Parameters

"Supporting_Files" is an AMI parameter of Type String, Usage Info, Format List that contains a list of the files the model requires in addition to the DLL or shared object file. In the specified List, each String is the relative path from the .ibs file directory to one supporting file or directory.

Example:

```
(Supporting_Files (Usage Info)(Type String)
   (List "my_stuff_dir" "m1.s4p" "m2.s4p" "m3.s4p")
   (Description "Additional files that support this model")
)
```

"DLLPath" is an AMI parameter of Type String, Usage In and format Value that gives the model the path to the directory it is being executed from.

Example:

```
(DLLPath (Usage In)(Type String)(Value "NA")
   (Description "Path to where the DLL is running")
)
```

The EDA tool is responsible for recognizing this parameter name and replacing the value declared in the .ami file with a string that contains the correct path information for the algorithmic model. In this string, the path separator is the forward slash ("/"), and the model is responsible for making any OS-specific adjustments (for example, replacing forward slashes "/" with backslashes "\" if necessary).

"DLLid" is an AMI parameter of Type String, Usage In and format Value that is guaranteed to have a unique name for each instance of an IBIS-AMI model and simulation run in a single results directory.

Example:

```
(DLLid (Usage In)(Type String)(Value "NA")
   (Description "Unique base name for each AMI model instance and run")
)
```

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 **DLLid** as the base name for any data files that the model creates, either for use as temporary storage or for recording output data. The use of **DLLid** helps guarantee that multiple instances of the same model (or different models from the same vendor) do not mix up data as a result collisions between temporary file names.

DLLid can optionally be used to create a standardized ASCII report file that EDA tools can parse and display. This report file has the base name of **DLLid** and extension ".report". For example, if the value of **DLLid** was set to "base_sim.Rx1.dll", then the name report file created would be "base_sim.Rx1.dll.report". The report file is ASCII and may contain lines that begin with the keyword "Result", followed by two fields, a <parameter name> and a <value>. There is no restriction that the <parameter name> be the name of a declared AMI parameter. The <value> may either be a string or a number. No two Result records in a DLL Report file may have the same <parameter name>. If either <parameter name> or <value> is a string that has embedded white space or ",", then the <parameter name> or <value> must be surrounded by two double quotes ("). Double quotes (") are not permitted within <parameter name> or <value>.

The DLL Report file is not required to exist, may be empty or may have any number of Result records. The format of the .report file is defined to make it easier for EDA tools and/or user scripts to parse and report on specific contents of a DLLid.report file. The model may convey additional information in this file, but these ASCII records should not begin with the keyword "Result".

"Samples_Per_Bit" is an AMI parameter of Type Integer, Usage Info, and format Value that states the number of samples per bit which should be used with the model to assure correctness, obtain greatest accuracy, or achieve greatest efficiency. This parameter should only be declared for models where it is desirable to use a specific number of samples per bit. If the model is designed to be independent of the Samples_Per_Bit setting, this parameter should not be declared.

Example:

```
(Samples_Per_Bit (Usage Info)(Type Integer)(Value 16)
   (Description "This model requires 16 samples per bit")
)
```

The EDA tool is expected to take the model's requested Samples_Per_Bit setting into account when preparing the input data for the algorithmic model. This includes the values of **bit_time** and **sample_interval** passed to the model and how waveform data is prepared.

Analog Model Parameters

The analog portion of an IBIS-AMI model is used by the EDA tool to derive the impulse response of the analog channel. Nominally, the simulation uses the traditional IBIS [Model] data from the .ibs file. This provides the highest level of model portability between EDA tools but does not always provide the highest level of simulation accuracy. These parameters define two additional levels of information that can be used to model the analog buffer.

The EDA tool is expected to use the following precedence for determining the data to be used to model the analog buffer:

1. Broadband Analog Model
2. Equivalent-circuit Analog Model
3. [Model] declaration in .ibs file

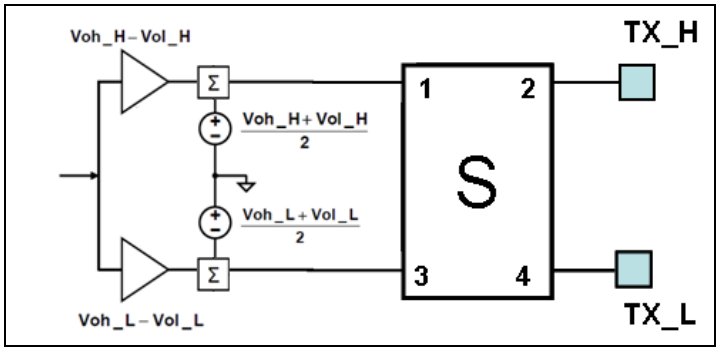
The lowest precedence data (#3) is REQUIRED and therefore should always be present in the .ibs file. Either method #1 or #2 (or both) may also be specified in the model's .ami file, in which case the EDA tool is expected to use the model data according to the defined order of precedence, which reflects the relative accuracy of each of these three methods.

Broadband Analog Model Parameters

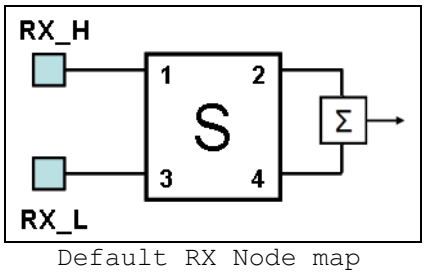
"Tstonefile" is an AMI parameter of Type String and Usage Info that specifies the relative path from the directory containing the .ibs and .ami files to a Touchstone file that is to be used as the analog model for the device when determining the impulse response of the channel. This is used in place of the analog model described in the IBIS (.ibs) file for the buffer. The S-parameter file represents the analog buffer model only and should not contain any package model data.

When **Tstonefile** is used to describe a TX analog buffer, the S-parameter file is connected between an input voltage source and the TX die pad. The TX die pad ports are connected to the same nodes in the circuit netlist as a traditional IBIS output model would be. The value of the voltage source is expected to be defined using the Voh/Vol and Trf parameters. The Voh/Vol and Trf parameters are defined in the Equivalent Analog Circuit Model Parameter section and are shared between the Equivalent Circuit Analog Model and the Broadband Analog Model.

When **Tstonefile** is used to describe an RX analog buffer, the S-parameter file is connected between the RX die pad and the output nodes used by the simulator to derive the differential impulse response for the network.



Default TX node map



Default RX Node map

TX Example:

```
(Tstonefile (Usage Info)(Type String)
   (Corner "NC.s4p" "WC.s4p" "BC.s4p")
   (Description "Driver on-die S-parameter file")
)
```

Note that **Tstonefile** can be declared using different formats. The example above declares Tstonefile using format Corner so that different analog models are used for typ, slow and fast cases.

The default TX and RX node maps assume that the high-speed serial channel flows from left to right, and that analog model S parameters flow from left to right. Transmitter input ports are assumed to be the left hand (Near) ports 1 and 3, while the pad ports are assumed to be the right hand (Far) ports 2 and 4. Receiver pad ports are assumed to be the left hand (Near) ports 1 and 3, while the output ports are assumed to be the right hand (Far) ports 2 and 4.

Transmitter Default Port Map

Port	Description
1	Input true port
2	Pad true port
3	Input complement port
4	Pad complement port

Receiver Default Port Map

1	Pad true port
2	Output true port
3	Pad complement port
4	Output complement port

For S parameter files that follow this convention, the first and third rows of the four-port matrix [S11, S12, S13, S14, S31, S32, S33, S34] for a transmitter are not used, and may be set to all zeros. Similarly, the second and fourth columns [S12, S22, S32, S42, S14, S24, S34, S44] of the four-port matrix for a receiver are not used, and may be set to all zeros. Other port mappings are supported through the use of the optional parameter **Nodemap**.

"Nodemap" is an AMI parameter of Type String, Usage Info and format Value that is used to override the default node mapping for on-die S parameters. The defined string consists of a sequence of letter/number pairs. The letter refers to the side the port is on and the number indicates the port number. The letter N refers to the left hand or Near side and the letter F refers to the right hand or Far side. The specific format for each case is

```
TX four port: N<Input true>N<Input comp>F<Pad true>F<Pad comp>
RX four port: N<Pad true>N<Pad comp>F<Die true>F<Die comp>
```

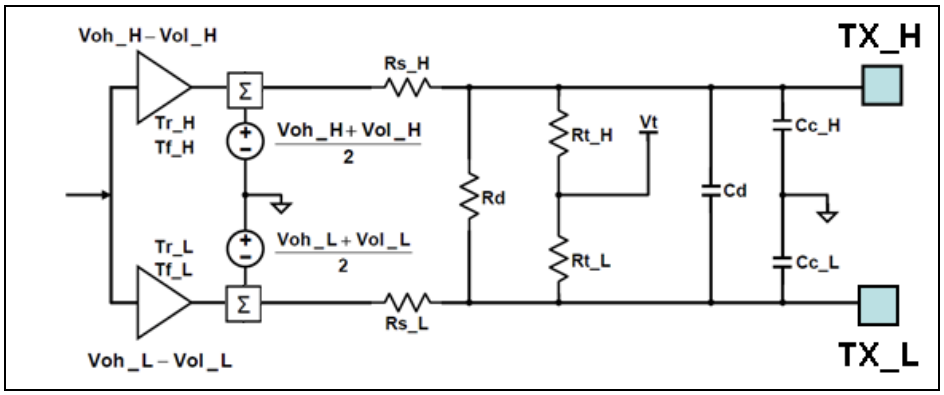
Default TX node map example:

```
(Nodemap (Usage Info)(Type String)(Value "N1N3F2F4")
   (Description "Default nodemap for on-die TX")
)
```

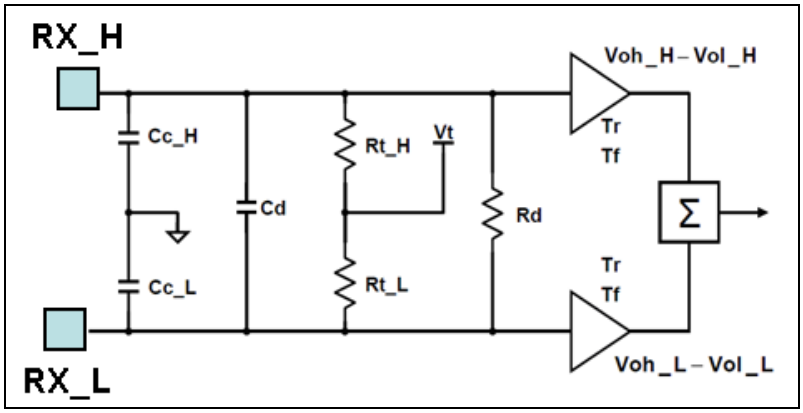
The above example defines the default node mapping the EDA tool would assume for a **Tstonefile** if **Nodemap** were not defined.

Equivalent-Circuit Analog Model Parameters

The AMI parameters listed below are of Type Float and Usage Info. They correspond to equivalent circuits that should be used in place of the TX or RX analog model described in the IBIS (.ibs) file.



TX Analog Buffer Equivalent Circuit



RX Analog Buffer Equivalent Circuit

All voltages are in volts, all resistances in ohms, all capacitances in farads, and all rise/fall times in seconds.

"Voh": Output voltage for logic 1 (Default 1.0)
"Vol": Output voltage for logic 0 (Default 0.0)
"Vt": Termination voltage (Default 0.0)
"Tr": 20%-80% rise time (Default 0.0)
"Tf": 80%-20% fall time (Default 0.0)
"Trf": 20%-80% rise/fall time) (Implies Tr = Tf = Trf) (Default 0.0)
"Rt": Termination resistance (Default 1e+6)
"Rd": Differential termination resistance (Default 100)
"Rs": Series output resistance (Default 50.0)
"Cc": Single ended common mode termination capacitance (Default 0.0)
"Cd": Differential termination capacitance (Default 0.0)

For each of the parameters Voh, Vol, Rt, Rs, and Cc, there are two additional parameters defined: one with "**_H**" appended to the parameter name and one with "**_L**" appended to the parameter name. When these suffixes are attached to the name, they refer to "high" side and "low" side, respectively. When these suffixes are omitted, the same element value applies to both the "high" side and the "low" side.

Example:

```
(Voh (Usage Info)(Value 0.9)(Type Float)
   (Description "Output open circuit high voltage")
)
(Vol (Usage Info)(Value 0.0)(Type Float)
   (Description "Output open circuit low voltage")
)
(Trf (Usage Info)(Value 40e-12)(Type Float)
   (Description "20%-80% output rise time")
)
(Rs (Usage Info)(Value 47.75)(Type Float)
   (Description "Single-ended output resistance")
)
(Cc (Usage Info)(Value 0.5e-12)(Type Float)(Default 0.5e-12)
   (Description "Output Capacitance")
)
```

This defines an output model with a 900mV supply voltage, a 20-80% rise time of 40ps, an output impedance of 47.75 Ohms and a output capacitance of 0.5pF.

Jitter, Noise and Clock Parameters

The following optional Reserved Parameters are used to specify impairments for the transmitter output. These budgets specify the impairment as measured at the TX output (i.e. the transmitter output is expected to be directly modulated by these amounts). This data is used by the simulator to either modify the input presented to the algorithmic model or when post-processing the results from the model; the budget values specified by these parameters are not passed directly to the model itself.

"Tx_Rj" is an AMI parameter of Type either Float or UI and Usage either Info or Out which defines the standard deviation, in seconds or UI, of a Gaussian phase noise process at the transmitter which is to be added to the behavior implemented directly by the transmitter model.

Example:

```
(Tx_Rj (Usage Info)(Corner 0.005 0.006 0.004)(Type UI)
   (Description "TX Random Jitter in UI.")
)
```

"Tx_Sj" is an AMI parameter of Type either Float or UI and Usage either Info or Out which defines half the peak to peak amplitude, in seconds or UI, of a sinusoidal jitter which is to be added to the behavior implemented directly by the transmitter model.

Example:

```
(Tx_Sj (Usage Info)(Corner 0.05 0.07 0.4)(Type UI)
   (Description "TX Sinusoidal Jitter in UI.")
)
```

"Tx_Sj_frequency" is an AMI parameter of Type Float and Usage either Info or Out which defines the frequency, in Hertz, of the sinusoidal jitter at the transmitter.

Example:

```
(Tx_Sj_Frequency (Usage Info)(Corner 6.5E7 6.5E7 6.5E7)(Type UI)
   (Description "TX Sinusoidal Jitter Frequency in Hz.")
)
```

"Tx_DCD" is an AMI parameter of Type either Float or UI and Usage either Info or Out which defines half the peak to peak clock duty cycle distortion, in seconds or UI, to be added to the behavior implemented directly by the transmitter model.

Example:

```
 (Tx_DCD (Usage Info)(Corner 0.008 0.016 0.005)(Type UI)
    (Description "TX Duty Cycle Distortion in UI.")
)
```

The following optional Reserved Parameters are used to specify characteristics of the receiver's recovered clock when the model does not return clock_ticks information from an AMI_Getwave call. This data is used by the simulator when post-processing the results from the model; the budget values specified by these parameters are not passed directly to the model itself.

"Rx_Clock_Recovery_Mean" is an AMI parameter of Type either Float or UI and Usage either Info or Out which defines a static offset, in seconds or UI, between the recovered clock and the median threshold crossing time in the eye diagram plus one half bit period.

Example:

```
(Rx_Clock_Recovery_Mean (Usage Info)(Value 0.05)
   (Type UI)(Description "Recovered Clock offset in UI.")
)
```

"Rx_Clock_Recovery_Rj" is an AMI parameter of Type either Float or UI and Usage either Info or Out which defines the standard deviation, in seconds or UI, of a Gaussian phase noise exhibited by the recovered clock.

Example:

```
(Rx_Clock_Recovery_Rj (Usage Info)(Corner 0.005 0.006 0.004)
   (Type UI)(Description "RX Random Clock Jitter in UI.")
)
```

"Rx_Clock_Recovery_Sj" is an AMI parameter of Type either Float or UI and Usage either Info or Out which defines half the peak to peak variation, in seconds or UI, of a sinusoidal phase noise exhibited by the recovered clock.

Example:

```
(Rx_Clock_Recovery_Sj (Usage Info)(Corner 0.05 0.07 0.4)(Type UI)
   (Description "RX Sinusoidal Jitter in UI.")
)
```

"Rx_Clock_Recovery_DCD" is an AMI parameter of Type either Float or UI and Usage either Info or Out which defines half the peak to peak variation, in seconds or UI, of a clock duty cycle distortion exhibited by the recovered clock.

Example:

```
(Rx_Clock_Recovery_DCD (Usage Info)(Corner 0.008 0.016 0.005)
   (Type UI)(Description "RX Duty Cycle Distortion in UI.")
)
```

The following optional Reserved Parameters are used to modify the statistics associated with receiver's recovered clock when the model returns clock ticks information from an AMI_Getwave call. This data is used by the simulator when post-processing the results from the model; the budget values specified by these parameters are not passed directly to the model itself.

"Rx_Rj" is an AMI parameter of Type either Float or UI and Usage either Info or Out which defines the standard deviation, in seconds or UI, of a Gaussian phase noise driven by impairments external to the receiver. This phase noise is to be accounted for in both Statistical and Time-Domain simulations.

Example:

```
(Rx_Rj (Usage Info)(Corner 0.005 0.006 0.004)(Type UI)
   (Description "RX Random Jitter in UI.")
)
```

"Rx_Sj" is an AMI parameter of Type either Float or UI and Usage either Info or Out which defines half the peak to peak variation, in seconds or UI, of a sinusoidal phase noise driven by impairments external to the receiver. This phase noise is to be accounted for in both Statistical and Time-Domain simulations.

```
(Rx_Sj (Usage Info)(Corner 0.05 0.07 0.4)(Type UI)
   (Description "RX Sinusoidal Jitter in UI.")
)
```

"Rx_DCD" is an AMI parameter of Type either Float or UI and Usage either Info or Out which defines half the peak to peak variation, in seconds or UI, of a clock duty cycle distortion driven by impairments external to the receiver. This phase noise is to be accounted for in both Statistical and Time-Domain simulations.

Example:

```
(Rx_DCD (Usage Info)(Corner 0.008 0.016 0.005)(Type UI)
   (Description "RX Duty Cycle Distortion in UI.")
)
```

The following optional Reserved Parameter is used to modify the statistics associated with the data input to the receiver's sampling latch. This data is used by the simulator when post-processing the results from the model; the budget values specified by this parameter are not passed directly to the model itself.

"Rx_Noise" is an AMI parameter of Type Float and Usage either Info or Out which defines the standard deviation, in volts into a 100 ohm differential load, of a set of independent samples of a Gaussian noise process measured at the input of a receiver.

Example:

```
(Rx_Noise (Usage Info)(Format Corner 0.0030 0.0035 0.0025)
   (Type Float)(Description "RX amplitude noise in V.")
)
```

Dependency Tables

Dependency Tables are used to specify relationships between different AMI parameters. This is desirable because sometimes the value to be used for an AMI parameter is dependent on the selected value of another AMI parameter. For example, consider the case of an output buffer with a user-selectable output strength setting. The strength selected by the user affects both the output's voltage swing and impedance.

Let's assume that the output strength is controlled by a Model_Specific parameter "Tx_Strength" that has 8 discrete settings from 0 to 7, with a nominal value of 4:

```
(Tx_Strength (Usage In)(Type Integer)(Range 4 0 7)
   (Description "Output buffer strength setting")
)
```

The TX analog buffer's impedance is specified using the equivalent circuit analog model parameters. The output impedance values for each of the 8 output strength settings are:

```
(Rs (Usage Info)(Type Float)
   (List 45.0 46.0 48.0 50.0 52.0 50.0 48.0 45.0)
   (Description "TX output impedance")
)
```

The TX analog buffer's output swing is also specified using the equivalent circuit analog model parameters. The output voltage values for each of the 8 output strength settings are:

```
(Voh (Usage Info)(Type Float)
   (List 0.40 0.42 0.44 0.46 0.48 0.50 0.52 0.54)
   (Description "TX output voltage")
)
```

The model user is to be presented with a single control parameter, Tx_Strength, with the values of Rs and Voh automatically selected by the EDA tool based on the value the user specifies for Tx_Strength. Dependency Tables specify relationships between parameters to make this possible.

Input parameters are placed in the left-most columns and output parameters are placed in the right-most columns of the table. The table header ("Parameter") row lists the Parameter names used in the table and explicitly defines whether each Parameter is used as an Input ("In") to the table, or is an output ("Out_") whose value is derived from the table.

For the example just described, one possible Dependency Table declaration is:

```
(Tx_Strength_Table
   (Dependency
      (Parameter (Usage Info)(Type String)
           (List "Tx_Strength In" "Rs Out_Match" "Voh Out_Match"))
      (Row1 (List 0  45.0  0.40)(Usage Info)(Type Float))
      (Row2 (List 1  46.0  0.42)(Usage Info)(Type Float))
      (Row3 (List 2  47.0  0.44)(Usage Info)(Type Float))
      (Row4 (List 3  50.0  0.46)(Usage Info)(Type Float))
      (Row5 (List 4  52.0  0.48)(Usage Info)(Type Float))
      (Row6 (List 5  50.0  0.50)(Usage Info)(Type Float))
      (Row7 (List 6  48.0  0.52)(Usage Info)(Type Float))
      (Row8 (List 7  45.0  0.54)(Usage Info)(Type Float))
   )
)
```

This tells the EDA tool that a Tx_Strength setting of 0 corresponds to Rs = 45.0 ohms, Voh = 0.40V, a setting of 1 corresponds to Rs = 46.0 ohms, Voh = 0.42V, and so on.

The following rules apply to the use of Dependency Tables:

1. The Parameter names used in the Dependency Table must already be declared in the .ami file, either in the Reserved_Parameter or Model_Specific sections
2. The values listed in the table for each Parameter must be legal values for that Parameter based on its definition in the .ami file.
3. Dependency tables can only be declared in the Model_Specific section of the .ami file
4. A Dependency Table defines the relationship between one or more input parameters and one or more output parameters.
5. Two or more Dependency Tables can be linked together (i.e. output variables in one table can be used as input variables in another table) to specify complex relationships
6. Dependency Tables are evaluated in the order they are declared in the .ami file.

The following elements define a Dependency table:

- Table declaration
- Header row (Parameter names & Input/Output declarations)
- Data rows

A Dependency Table declaration begins with a table name followed by the keyword **"Dependency"**. The name of the Dependency Table must be unique, and by convention, ends with "_Table":

Example:

```
(My_Table
   (Dependency
      (
```

The header row begins with the keyword **"Parameter"** and declares the parameters defined in the table and whether they are inputs or outputs to the relationship being defined. A parameter is defined as an input to the table by adding the keyword "In" after the parameter name. A parameter is defined as an output from the table by adding **one** of the keywords **"Out_Match", "Out_Closest", "Out_Range"** or **"Out_PWL"** after the parameter name.

Example:

```
(Parameter (Usage Info)(Type String)
  (List "Tx_Strength In" "Rs Out_Match" "Voh Out_Match"))
```

The header row is always of Usage Info, Type String and Format List. The data rows that follow must contains lists must have the same number of items as the header row.

The data rows contain the input parameter values that, when matched according to the defined criteria, provide the values to be used for the output parameters. Data rows are always specified as Usage Info and Format List. Note that in order to comply with the definition of the List format, all entries in the list must have the same Type. This poses a bit of a challenge since different columns can have parameters of different types (remember that the parameters used in each column are defined separately in the .ami file). For example, the parameter in one column may be of Type Integer, the next column of Type Float, and the next column of Type String. The Type chosen for the List must therefore be one that can express all of the entries in the row, and EDA tool is be responsible for converting each parameter value to the correct Type as determined by the respective parameter definition.

In practice, the most general solution is to always declare data rows to be of Type String and surround each entry in the table with double quotes "". The EDA tool is responsible for performing any necessary Type conversion.

Example:

```
(Tx_Strength (Usage In)(Type String)
   (List "0" "1" "2" "3" "4" "5" "6" "7")
   (Description "Output buffer strength setting")
)

(Rs (Usage Info)(Type Float)
   (List 45.0 46.0 48.0 50.0 52.0 50.0 48.0 45.0)
   (Description "TX output impedance")
)

(Voh (Usage Info)(Type Float)
   (List 0.40 0.42 0.44 0.46 0.48 0.50 0.52 0.54)
   (Description "TX output voltage")
)

.
.
.

(Tx_Strength_Table
   (Dependency
      (Parameter (Usage Info)(Type String)
           (List "Tx_Strength In" "Rs Out_Match" "Voh Out_Match"))
      (Row1 (List "0"  "45.0"  "0.40")(Usage Info)(Type String))
      (Row2 (List "1"  "46.0"  "0.42")(Usage Info)(Type String))
      (Row3 (List "2"  "47.0"  "0.44")(Usage Info)(Type String))
      (Row4 (List "3"  "50.0"  "0.46")(Usage Info)(Type String))
      (Row5 (List "4"  "52.0"  "0.48")(Usage Info)(Type String))
      (Row6 (List "5"  "50.0"  "0.50")(Usage Info)(Type String))
      (Row7 (List "6"  "48.0"  "0.52")(Usage Info)(Type String))
      (Row8 (List "7"  ‘45.0"  "0.54")(Usage Info)(Type String))
   )
)
```

If all of the parameters listed in the data row are a numeric Type (Float, Tap, UI, or Integer), the row can be declared as Type Float.

Example:

```
      (Tx_Strength (Usage In)(Type Integer)(Range 4 0 7)
         (Description "Output buffer strength setting")
      )

      (Rs (Usage Info)(Type Float)
         (List 45.0 46.0 48.0 50.0 52.0 50.0 48.0 45.0)
         (Description "TX output impedance")
      )

      (Voh (Usage Info)(Type Float)
         (List 0.40 0.42 0.44 0.46 0.48 0.50 0.52 0.54)
         (Description "TX output voltage")
      )

      .
      .
      .

      (Tx_Strength_Table
         (Dependency
            (Parameter (Usage Info)(Type String)
                 (List "Tx_Strength In" "Rs Out_Match" "Voh Out_Match"))
            (Row1 (List 0  45.0  0.40)(Usage Info)(Type Float))
            (Row2 (List 1  46.0  0.42)(Usage Info)(Type Float))
            (Row3 (List 2  47.0  0.44)(Usage Info)(Type Float))
            (Row4 (List 3  50.0  0.46)(Usage Info)(Type Float))
            (Row5 (List 4  52.0  0.48)(Usage Info)(Type Float))
            (Row6 (List 5  50.0  0.50)(Usage Info)(Type Float))
            (Row7 (List 6  48.0  0.52)(Usage Info)(Type Float))
            (Row8 (List 7  45.0  0.54)(Usage Info)(Type Float))
         )
      )
```

If all of the parameters listed in the data row are of the same Type, then that Type can be used, but that’s usually not the case.

The Dependency Table contains as many data rows as needed to specify all the combinations of input parameters to be tested (matched) and their corresponding output parameter values. In order to comply with the rules of the .ami file syntax, each data row must start with a name (Row1 ... Row8 in the above example). There are no constraints on row names, and the EDA tool, except for the reserved row name "Default_Row", ignores the row names.

A row with the name "Default_Row" may be used to define the output parameter values to be used when values of the input parameters do not match any row in the Dependency Table. The values of the input parameters specified in "Default Row" are ignored by the EDA tool. If no Default_Row is defined in the table and there is no match, then an output parameter will be set to its default value from its declaration.

Example:

```
(Default_Row (List "NA"  "45.0"  "0.54")(Usage Info)(Type String))
```

Note that the Type used for the **Default_Row** must match the Type declared for the other data rows in the table.

Dependency Tables support one or more columns of input parameter data and one or more columns of output parameter data. Generally speaking, the values of the Input Parameters must exactly match the values listed in a data row to select the Output Parameter values for that row. The last input column in a Dependency Table (or the only Input column in the single Input case) is treated differently, however, in that interpolation and non-exact value matches are supported on the Input values listed in that column. Limiting special treatment to a single input column limits the complexity of the Dependency Table.

The type of special treatment used when matching the last column of input parameters is specified with the Output parameter, and as such, the method of matching used can be different for each output parameter.

The types of parameter matching supporting on the last input parameter column are:

Out_Match – Output parameter value is set to the row where the last input parameter column is an exact string match or it is a numeric match to within floating Point resolution. If no matching criteria are met, the output parameter is set to the default value as defined above.

Out_Closest – Output parameter value is set to the row where the last input parameter column matches closest numerically to the value of that parameter. This requires the last input parameter column to contain numeric data. If there are two equal choices, the larger value will used for selection. If it does not contain numeric data, Out_Match will be used instead. If no matching criteria are met, the output parameter is set to the default value as defined above.

Out_Range – Output parameter value is set to the row where the last input parameter column is less than or equal to the value of that parameter and the next numerically larger row is greater than that value. If there is not a next numerically larger row, then the output parameter value is set to the row where the last input parameter column is less than or equal to the value of that parameter. This requires the last input parameter column to contain numeric data. If it does not contain numeric data, Out_Match will be used instead. If no matching criteria are met, the output parameter is set to the default value as defined above.

Out_PWL – Output parameter value is set to the linear interpolated value between the row where the last input parameter column is less

than or equal to the value of that parameter and the next numerically larger row is greater than that value. If there is not a next numerically larger row, then the output parameter value is set to the extrapolated value using the next numerically smaller row. If this row does not exist, then the output parameter value is set to the value where the last input parameter column is less than or equal to the value of that parameter. This requires the last input parameter column to contain numeric data. If it does not contain numeric data, Out_Match will be used instead. If no matching criteria are met, the output parameter is set to the default value as defined above.

As an example, let's return to the example where the TX output buffer has a variable strength setting, but instead of 8 values, we'll increase the number of strength selections to 71:

```
(Tx_Strength (Usage In) (Type Integer) (Range 35 0 70)
       (Description "Output buffer strength setting")
)
```

Note that we could have defined Tx_Strength as a list with 71 values, but that's impractical – it's more convenient to define it as a range of Integers.

Let's assume that Rs and Voh have the same minimum and maximum values as before, such that it makes sense to specify them as Format Range as well:

```
(Rs (Usage Info) (Type Float) (Range 48.0 45.0 52.0)
   (Description "TX output impedance")
)

(Voh (Usage Info) (Type Float) (List 0.46 0.40 0.54)
   (Description "TX output voltage")
)
```

It would be possible to define a Dependency Table for all 71 possible settings of Tx_Strength, but let's assume that, in this case, it's just as accurate to specify the values of Rs and Voh as piecewise-linear data spaced in increments 10 units of Tx_Strength apart:

```
(Tx_Strength_Table
   (Dependency
      (Parameter (Usage Info) (Type String))
         (List "Tx_Strength In" "Rs Out_PWL" "Voh Out_PWL")
      )
      (Row1 (List  0 45.0 0.40)(Usage Info)(Type Float))
      (Row2 (List 10 46.0 0.42)(Usage Info)(Type Float))
      (Row3 (List 20 47.0 0.44)(Usage Info)(Type Float))
      (Row4 (List 30 50.0 0.46)(Usage Info)(Type Float))
      (Row5 (List 40 52.0 0.48)(Usage Info)(Type Float))
      (Row6 (List 50 50.0 0.50)(Usage Info)(Type Float))
      (Row7 (List 60 48.0 0.52)(Usage Info)(Type Float))
      (Row8 (List 70 45.0 0.54)(Usage Info)(Type Float))
   )
)
```

This tells the EDA tool that a Tx_Strength setting of 15 corresponds to Rs = 46.5 ohms, Voh = 0.43V, a setting of 27 corresponds to Rs = 49.1 ohms, Voh = 0.434V, and so on.

Note that the use of **Out_PWL** in the above table means a **Default_Row** declaration is not needed, since the only legal values for Tx_Strength are between 0 and 70 inclusive, and there is no value for which the EDA tool could not interpolate the values for Rs and Voh.

The syntax definition for a Dependency Table is:

```
dependency table:
            (<table name>
                  (Dependency
                        <column header row>
                        <table body>
                  )
            )

column header row:
            (Parameter
                  (Usage Info) (Type String)
                  (List <column header list>)
            )
```

column header list: <input column header list> <output column header list>

input column header list: <input column header> <input column header> ...

output column header list: <output column header> <output column header> ...

input column header: "<parameter name> In"

output column header: "<parameter name> <output column type>"

output column type: Out_Match | Out_Closest | Out_Range | Out_PWL

table body: <table row> <table row> <table row> ...

table row: (<row name> (List <row list>) (Usage Info) (Type <row type>))

row list: <parameter value> <parameter value> <parameter value> ...

row type: String | Float | Integer | Boolean | Tap | UI

The column header list and all of the row lists in a dependency table shall have the same number of list entries.

The row type supplied for a table row shall be a type that can be converted to the type of any parameter value in the row.

"[Corner]" is a predefined AMI parameter of Type String and Usage Info that can be used to create an input column in a dependency table. It has a List format with the allowed values "Typ", "Slow" and "Fast".

"[bit_time]" is a predefined AMI parameter of Type Float and Usage Info that can be used to create an input column for data rate in a dependency table. It has a Value format and is equal to the length of one data bit in seconds.

"[BAUD]" is a predefined AMI parameter of Type Float and Usage Info that can be used to create an input column for data rate in a dependency table. It has a Value format and is equal to 1/[bit_time].

"[GBAUD]" is a predefined AMI parameter of Type Float and Usage Info that can be used to create an input column for data rate in a dependency table. It has a Value format and is equal to 1/([bit_time] * 1e9).

"[Model]" is a predefined AMI parameter of Type String and Usage Info that can be used to create an input column to declare which IBIS [Model] each row refers to in a dependency table.

****

ANALYSIS PATH/DATA THAT LED TO SPECIFICATION

The parameters defined in this BIRD came from commercial IBIS-AMI model development efforts where new functionality was needed to meet customer expectations for model functionality, accuracy and performance. The parameters in this BIRD were defined by SiSoft and its semiconductor partners. These parameters are being contributed to IBIS to ensure IBIS-AMI model accuracy and portability.

****

ANY OTHER BACKGROUND INFORMATION:

This BIRD is being requested by the following IBIS users and model developers, in conjunction with the authors:

Cisco Systems: Upen Reddy, Doug White
Ericsson: Anders Ekholm
Broadcom: Yunong Gan
IBM: Adge Hawes
TI: Alfred Chong, Srikanth Sundaram

****