****
****

BIRD ID#: ???
ISSUE TITLE: Clarify sample_interval for IBIS-AMI
REQUESTOR: Arpad Muranyi, Mentor Graphics
DATE SUBMITTED: October 18, 2011
DATE REVISED:
DATE ACCEPTED BY IBIS OPEN FORUM:

****
****

STATEMENT OF THE ISSUE:

The IBIS 5.0 specification does not spell out the rules clearly regarding sample_interval. As a result, numerous AMI models have been released which can only operate at certain sampling rates, while EDA tools pass data into the AMI models at a sample_interval of their own choice. Consequently, simulation results are often incorrect, or in more severe cases the AMI model and/or the EDA platform might crash as well.

****

STATEMENT OF THE RESOLVED SPECIFICATIONS:

In BIRD 130, replace the following paragraph:

| 3.1.2.1 impulse_matrix
| ======================
|
|* 'impulse_matrix' points to a memory location where the collection of
|* channel voltage impulse responses, called the "impulse response matrix",
|* is stored in the form of a single dimensional array of floating point
|* numbers. The impulse response values are uniformly spaced in time.
|* The sample spacing is given by the parameter ‘sample_interval’.
|

with:

| 3.1.2.1 impulse_matrix
| ======================
|
|* 'impulse_matrix' points to a memory location where the collection of
|* channel voltage impulse responses, called the "impulse response matrix",
|* is stored in the form of a single dimensional array of floating point
|** numbers. The impulse responses pointed to by the 'impulse_matrix'
|** argument are both input and output. The EDA platform provides the
|** impulse responses. The algorithmic model is expected to modify the
|** impulse responses in place by applying a filtering behavior, for
|** example, an equalization function, if modeled in the AMI_Init
|** function. The impulse response values are uniformly spaced in time.
|** The sample spacing is determined by the EDA tool and passed to the
|** algorithmic model trough the AMI_Init function's ‘sample_interval’
|** argument.

On pg. 186, replace the following lines:

```
| 3.1.2.4 sample_interval
|
| This is the sampling interval of the impulse_matrix. Sample_interval is
| usually a fraction of the highest data rate (lowest bit_time) of the
| device.  Example:
|
|   Sample_interval = (lowest_bit_time/64)
```

with:

```
| 3.1.2.4 sample_interval
|
|* This is the sampling interval of the 'impulse_matrix' passed into the
|* AMI_Init function and the 'wave' passed into the AMI_GetWave function.
|* The sample_interval is determined by the EDA tool and it is usually a
|* fraction of the bit_time.  The 'impulse_matrix' and 'wave' returned by
|* the algorithmic model must have the same 'sample_interval' as the
|* original 'impulse_matrix' and 'wave' that was passed into the
|* algorithmic model.
|*
|* Impulse responses in 'impuse_matrix' and waveforms in 'wave' should be
|* treated as continuous analog waveforms by the algorithmic models.  For
|* this reason, algorithmic models must be able to produce valid results
|* at any sample_interval.  Any internal analog to digital conversion or
|* resampling is the responsibility of the algorithmic model.  In case the
|* algorithmic model is unable to operate at a given sample_interval, it
|* should abort gracefully with an exit code 0 (failure) and appropriate
|* messaging.
|*
|* Example:
|
|   Sample_interval = (bit_time/64)
|
```

On pg. 188 replace these lines:

```
| 3.2.2.1 wave
|
| A vector of a time domain waveform, sampled uniformly at an interval
| specified by the ‘sample_interval’ specified during the init call. The
| wave is both input and output. The EDA platform provides the wave. The
| algorithmic model is expected to modify the waveform in place by applying
| a filtering behavior, for example, an equalization function, being
| modeled in the AMI_Getwave call.
```

with:

```
| 3.2.2.1 wave
|
|* 'wave' points to a memory location where a uniformly sampled vector of
|* a time domain waveform is stored.  The waveform pointed to by the
|* 'wave' argument is both input and output.  The EDA platform provides
| the wave.  The algorithmic model is expected to modify the waveform in
| place by applying a filtering behavior, for example, an equalization
|* function, if modeled in the AMI_Getwave function.  The sample spacing
|* is determined by the EDA tool and passed to the algorithmic model
```

```
|* trough the AMI_Init function's ‘sample_interval’ argument.
|
```

On pg. 188 replace these lines:

```
| 3.2.2.2 wave_size
|
| Number of samples in the waveform vector.
```

with:

```
| 3.2.2.2 wave_size
|
|* This is the number of samples in the waveform vector that is in the
|* AMI_GetWave function argument 'wave'.  The length of this waveform is
|* determined by the EDA tool.  The 'wave' returned by the algorithmic
|* model must have the same number of samples as the original 'wave' that
|* was passed into the algorithmic model.  Algorithmic models must be able
|* to produce valid results with any wave_size.  In case the algorithmic
|* model is unable to operate with a given wave_size, it should abort
|* gracefully with an exit code 0 (failure) and appropriate messaging.
```

****

ANALYSIS PATH/DATA THAT LED TO SPECIFICATION:

This topic was discussed at length in IBIS-ATM teleconferences as well as on the IBIS-ATM email reflector. Suggestions were made to introduce a new required parameter called Samples_per_Bit to provide a mechanism for algorithmic models to communicate to the EDA tool how many samples per bit in the impulse response and waveform they were designed to operate with. These suggestions were turned down in favor of clarifying the original intent of the specification that it is the EDA tool's responsibility to determine the sampling rate of the impulse responses and waveforms, and all algorithmic models must be able to produce correct results with any samples per bit values.

The question whether there should be a mechanism to allow algorithmic models to impose a limit for the EDA tool on how it determines the value for samples per bit was tabled due to insufficient experience in this area.

During the review period of this BIRD, it was noted that some models do not work with certain 'wave_size' values, yet the length of the 'wave' is also determined by the EDA tool. For this reason, similar rules were added to the 'wave_size' argument of the AMI_GetWave function in section 3.2.2.2.

****

ANY OTHER BACKGROUND INFORMATION:

****