**BUFFER ISSUE RESOLUTION DOCUMENT (BIRD)**

**BIRD NUMBER:** 214

**ISSUE TITLE:** Change “bit\_time” to “symbol\_time”

**REQUESTOR:**  Arpad Muranyi, Siemens EDA

**DATE SUBMITTED:** September 15, 2021

**DATE REVISED:**

**DATE ACCEPTED:** October 8, 2021

**DEFINITION OF THE ISSUE:**

Discussions on a “PAMn” proposal in the Advanced Technology Modeling Task Group revealed that the terminology “bit\_time” used in the AMI portion of the IBIS v7.0 specification is technically incorrect for signaling types other than NRZ. While the IBIS v7.0 specification does state that “For PAM4 models, bit\_time shall be the symbol time.” (pg. 203), the “PAMn” proposal to extend the IBIS specification with additional non-NRZ signaling types raised the question whether the terminology “bit\_time” should be replaced with the more general term “symbol\_time” which is correct for all signaling types.

The difficulty in doing so is rooted in the fact that the IBIS v7.0 specification contains 24 occurrences of “bit\_time”, 4 occurrences of “bit time”, two occurrences of “symbol\_time”, two occurrences of “symbol time”, zero occurrences of “unit\_interval”, 4 occurrences of “unit interval”, and 145 occurrences of “UI”, which are interrelated. For this reason, the changes proposed in this BIRD must be made with careful consideration of each of these occurrences.

**SUMMARY OF PROPOSED CHANGES:**

Replace “bit\_time” with “symbol\_time” and related text where appropriate without making any technical changes to the specification.

**PROPOSED CHANGES:**

On pg. 199, replace:

“6. The executable model computes dependent parameter values according to independent parameter values in AMI\_parameters\_in, bit\_time, corner and model\_name.”

With:

“6. The executable model computes dependent parameter values according to independent parameter values in AMI\_parameters\_in, symbol\_time, corner and model\_name.”

On pg. 200, replace:

“In this example, the Rx analog model is represented with a 4-port Touchstone file specified by parameter my\_file. Both Rx\_Receiver\_Sensitivity and my\_file depend on the legacy IBIS model name, parameter my\_corner, and parameter OP\_mode, which specifies the device operation mode. Rx\_Receiver\_Sensitivity also depends on bit\_time. Parameters Model\_Name, my\_corner and OP\_mode, having usage type In, are included in both input parameter strings to AMI\_Resolve and AMI\_Init. my\_file is of usage type Dep, and its dependency on Model\_Name, my\_corner and OP\_mode is resolved in AMI\_Resolve, which returns the value of my\_file. Rx\_Receiver\_Sensitivity is of usage type Out, and its dependency on Model\_Name, my\_corner, OP\_mode and bit\_time is resolved in AMI\_Init, which returns the value of Rx\_Receiver\_Sensitivity.”

With:

“In this example, the Rx analog model is represented with a 4-port Touchstone file specified by parameter my\_file. Both Rx\_Receiver\_Sensitivity and my\_file depend on the legacy IBIS model name, parameter my\_corner, and parameter OP\_mode, which specifies the device operation mode. Rx\_Receiver\_Sensitivity also depends on symbol\_time. Parameters Model\_Name, my\_corner and OP\_mode, having usage type In, are included in both input parameter strings to AMI\_Resolve and AMI\_Init. my\_file is of usage type Dep, and its dependency on Model\_Name, my\_corner and OP\_mode is resolved in AMI\_Resolve, which returns the value of my\_file. Rx\_Receiver\_Sensitivity is of usage type Out, and its dependency on Model\_Name, my\_corner, OP\_mode and symbol\_time is resolved in AMI\_Init, which returns the value of Rx\_Receiver\_Sensitivity.”

On pg. 201, replace:

“*Declaration:* long AMI\_Init (double \*impulse\_matrix,

long number\_of\_rows,

long aggressors,

double sample\_interval,

double bit\_time,

char \*AMI\_parameters\_in,

char \*\*AMI\_parameters\_out,

void \*\*AMI\_memory\_handle,

char \*\*msg)”

With:

“*Declaration:* long AMI\_Init (double \*impulse\_matrix,

long number\_of\_rows,

long aggressors,

double sample\_interval,

double symbol\_time,

char \*AMI\_parameters\_in,

char \*\*AMI\_parameters\_out,

void \*\*AMI\_memory\_handle,

char \*\*msg)”

On pg. 203, replace:

“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. The unit for sample\_interval is the second.

Impulse responses in “impulse\_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 = (lowest\_bit\_time / 64)

bit\_time

bit\_time is the bit time or unit interval (UI) of the current data, e.g., 100 ps, 200 ps etc. The executable model file may use this information along with the impulse\_matrix to initialize the filter coefficients. The unit for bit\_time is the second. For PAM4 models, bit\_time shall be the symbol time.”

With:

“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 symbol\_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. The unit for sample\_interval is the second.

Impulse responses in “impulse\_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 = (lowest\_symbol\_time / 64)

symbol\_time

symbol\_time is the unit interval (UI) of the current data, e.g., 100 ps, 200 ps etc. For NRZ signaling, it is equivalent to bit time. The executable model file may use this information along with the impulse\_matrix to initialize the filter coefficients. The unit for symbol\_time is the second.”

On pg. 208, replace:

“**clock\_times**

Vector to return clock times. The clock times are referenced to the start of the simulation (the first AMI\_GetWave call). The clock\_times vector is allocated by the EDA tool and is guaranteed to be greater than the number of clocks expected during the AMI\_GetWave call. The clock times are exactly bit\_time/2 before the input data signal is sampled. The algorithmic model will return non-negative clock\_times values, and place -1 after the last valid clock tick in the clock\_times vector during each AMI\_GetWave call. If there are no valid clock ticks for the duration of an AMI\_GetWave call, a single entry of -1 will be returned in the clock\_times vector. The units of clock\_times are seconds.

The clock ticks represented by clock times should be strictly monotonic, both within the clock\_times vector returned from a single call to AMI\_GetWave and between successive calls to AMI\_GetWave. That is, within a given clock\_times vector each successive valid value is greater than the value that preceded it, and the first valid value from a given call to AMI\_GetWave must be greater than the last valid value from the preceding call to AMI\_GetWave. Any non-strictly-monotonic behavior of clock times (including two identical values) should be considered by the EDA tool as an algorithmic model failure.

Each valid value in the clock\_times vector shall be used to sample the output waveform by adding to it bit\_time/2 (symbol time/2 for PAM4), regardless of whether that waveform sample occurs in the waveform segment being returned by the current call to AMI\_GetWave, or in the waveform segment to be returned by the next AMI\_GetWave call. Care should be taken in implementation of clock\_times to insure that the calculations used always maintain full double-precision floating point accuracy across multi-million bit simulations.”

With:

“**clock\_times**

Vector to return clock times. The clock times are referenced to the start of the simulation (the first AMI\_GetWave call). The clock\_times vector is allocated by the EDA tool and is guaranteed to be greater than the number of clocks expected during the AMI\_GetWave call. The clock times are exactly symbol\_time/2 before the input data signal is sampled. The algorithmic model will return non-negative clock\_times values, and place -1 after the last valid clock tick in the clock\_times vector during each AMI\_GetWave call. If there are no valid clock ticks for the duration of an AMI\_GetWave call, a single entry of -1 will be returned in the clock\_times vector. The units of clock\_times are seconds.

The clock ticks represented by clock times should be strictly monotonic, both within the clock\_times vector returned from a single call to AMI\_GetWave and between successive calls to AMI\_GetWave. That is, within a given clock\_times vector each successive valid value is greater than the value that preceded it, and the first valid value from a given call to AMI\_GetWave must be greater than the last valid value from the preceding call to AMI\_GetWave. Any non-strictly-monotonic behavior of clock times (including two identical values) should be considered by the EDA tool as an algorithmic model failure.

Each valid value in the clock\_times vector shall be used to sample the output waveform by adding symbol\_time/2 to it, regardless of whether that waveform sample occurs in the waveform segment being returned by the current call to AMI\_GetWave, or in the waveform segment to be returned by the next AMI\_GetWave call. Care should be taken in implementation of clock\_times to ensure that the calculations used always maintain full double-precision floating point accuracy across multi-million bit simulations.”

On pg. 210, replace:

“*Function:* **AMI\_Resolve**

*Required:* No, and illegal before AMI\_Version 6.1

*Declaration:* AMI\_Resolve (double bit\_time,

char \* AMI\_parameters\_in,

char \*\* AMI\_parameters\_out)

*Arguments:*

**bit\_time**

Input argument, in seconds, equals 1/data rate.”

With:

“*Function:* **AMI\_Resolve**

*Required:* No, and illegal before AMI\_Version 6.1

*Declaration:* AMI\_Resolve (double symbol\_time,

char \* AMI\_parameters\_in,

char \*\* AMI\_parameters\_out)

*Arguments:*

**symbol\_time**

symbol\_time is the unit interval (UI) of the current data, e.g., 100 ps, 200 ps etc. For NRZ signaling, it is equivalent to bit time. The unit for symbol\_time is the second.”

On pg. 216, replace:

“**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 parameter values are in units of UI (bit time). The parameter may take on either floating point or integer values.”

With:

“**UI**

Unit Interval. 1 UI is the inverse of the symbol rate. For example, 1 UI of 100 ps for an NRZ channel transmits 10 Gbit/s, while the same 100 ps UI for a PAM4 channel transmits 20 Gbit/s (or 10 Gsymbols/s). UI parameter values are in units of UI (symbol time). The parameter may take on either floating point or integer values.”

On pg. 236, replace:

“*Usage Rules:* For formats Gaussian, Dual-Dirac and DjRj, entries are assumed to be in units of UI when declared as Type UI and in units of seconds when Type Float.

For the Table format, only three table columns are permitted, which shall be entered in the following order:

Row\_number Time Probability, or

Row\_number UI Probability

where each Row\_number is an integer (positive or negative), each Time value is a floating point number in seconds or a bit time in units of UI, and each Probability is a unitless floating point number. The Type for each column must be specified when Format Table is used, as in:

(Type Integer Float Float)

(Type Integer UI Float)”

With:

“*Usage Rules:* For formats Gaussian, Dual-Dirac and DjRj, entries are assumed to be in units of UI when declared as Type UI and in units of seconds when Type Float.

For the Table format, only three table columns are permitted, which shall be entered in the following order:

Row\_number Time Probability, or

Row\_number UI Probability

where each Row\_number is an integer (positive or negative), each Time value is a floating point number in seconds or a symbol time in units of UI, and each Probability is a unitless floating point number. The Type for each column must be specified when Format Table is used, as in:

(Type Integer Float Float)

(Type Integer UI Float)”

On pg. 237, replace:

“*Definition:* Tx\_DCD (Transmit Duty Cycle Distortion) defines half the peak to peak clock duty cycle distortion to be added to the behavior implemented by the EDA tool by modifying the stimulus input or by post processing the simulation.

Time(n) = n \* bit\_time + Tx\_DCD \* (-1.0)n

where:

* n\*bit\_time is the ideal time of the nth clock.
* Time(n) is the time of the nth clock modified when creating input waveforms for the Tx.”

With:

“*Definition:* Tx\_DCD (Transmit Duty Cycle Distortion) defines half the peak-to-peak clock duty cycle distortion to be added to the behavior implemented by the EDA tool by modifying the stimulus input or by post processing the simulation.

Time(n) = n \* symbol\_time + Tx\_DCD \* (-1.0)n

where:

* n\*symbol\_time is the ideal time of the nth clock.
* Time(n) is the time of the nth clock modified when creating input waveforms for the Tx.”

On pg. 238, replace:

“Time(n) = n \* bit\_time + Tx\_Rj \* gaussian\_rand()”

With:

“Time(n) = n \* symbol\_time + Tx\_Rj \* gaussian\_rand()”

And also replace:

“Time(n) = n \* bit\_time + 2.0 \* Tx\_Dj \* rand()”

With:

“Time(n) = n \* symbol\_time + 2.0 \* Tx\_Dj \* rand()”

On pg. 239, replace:

“Time(n) = n \* bit\_time + Tx\_Sj \* sin((n \* bit\_time \* 2.0 \* Pi) \* Tx\_Sj\_Frequency)”

With:

“Time(n) = n \* symbol\_time + Tx\_Sj \* sin((n \* symbol\_time \* 2.0 \* Pi) \* Tx\_Sj\_Frequency)”

And also replace:

“Time(n) = n \* bit\_time + Tx\_Sj \* sin((n \* bit\_time \* 2.0 \* Pi) \* Tx\_Sj\_Frequency)”

With:

“Time(n) = n \* symbol\_time + Tx\_Sj \* sin((n \* symbol\_time \* 2.0 \* Pi) \* Tx\_Sj\_Frequency)”

On pg. 243, replace:

“*Usage Rules:* For formats Gaussian, Dual-Dirac and DjRj, entries are assumed to be in units of UI when declared as Type UI and in units of seconds when Type Float.

For the Table format, only three table columns are permitted, which shall be entered in the following order:

Row\_number Time Probability, or

Row\_number UI Probability

where each Row\_number is an integer (positive or negative), each Time value is a floating point number in seconds or a bit time in units of UI, and each Probability is a unitless floating point number. The Type for each column must be specified when Format Table is used, as in:

(Type Integer Float Float)

(Type Integer UI Float)”

With:

“*Usage Rules:* For formats Gaussian, Dual-Dirac and DjRj, entries are assumed to be in units of UI when declared as Type UI and in units of seconds when Type Float.

For the Table format, only three table columns are permitted, which shall be entered in the following order:

Row\_number Time Probability, or

Row\_number UI Probability

where each Row\_number is an integer (positive or negative), each Time value is a floating point number in seconds or a symbol time in units of UI, and each Probability is a unitless floating point number. The Type for each column must be specified when Format Table is used, as in:

(Type Integer Float Float)

(Type Integer UI Float)”

On pg. 253, replace:

“Prior to AMI\_Version 6.1, AMI modeling supported only NRZ SerDes signaling. AMI\_Version 6.1 introduces support for PAM4 SerDes signaling. A SerDes waveform is periodically sampled to determine the value of the waveform between transitions. The time interval between these samples is the Unit Interval (UI), also referred to as bit\_time (the value passed into the AMI\_Init function), and . Symbol\_time is a more generic name since a single UI can either represent a bit in NRZ or two bits in PAM4. The clock\_times returned by AMI\_GetWave are edge transition times, and are ½ UI before the nominal sample times. For NRZ, the mean edge transition time is close to the mean zero crossing time. For PAM4, the zero crossing time is only meaningful for transitions between symbols 0 and 3 and between symbols 1 and 2. In summary, UI, bit\_time and symbol\_time are the same and correspond to the time between the waveform edges sampled at the receiver latch. For clock\_times, zero crossing time and edge transition time are the same and are defined as ½ UI before the times that the Rx latch is sampled.”

With:

“Prior to AMI\_Version 6.1, AMI modeling supported only NRZ SerDes signaling. AMI\_Version 6.1 introduces support for PAM4 SerDes signaling. A SerDes waveform is periodically sampled to determine the value of the waveform between transitions. The time interval between these samples is the Unit Interval (UI), also referred to as symbol\_time (the value passed into the AMI\_Init function). Symbol\_time is a generic name since a single symbol (or UI) can either represent a bit in NRZ signaling or two bits in PAM4 signaling. The clock\_times returned by AMI\_GetWave are edge transition times and are ½ UI before the nominal sample times. For NRZ, the mean edge transition time is close to the mean zero-crossing time. For PAM4, the zero-crossing time is only meaningful for transitions between symbols 0 and 3 and between symbols 1 and 2. In summary, symbol\_time and UI have the same meaning and correspond to the time between the waveform edges sampled at the receiver latch. For clock\_times, zero-crossing time and edge transition time are the same and are defined as ½ UI before the times that the Rx latch is sampled.”

**BACKGROUND INFORMATION/HISTORY:**

The Editorial Task Group discussed this topic on September 1, 2021.