**BUFFER ISSUE RESOLUTION DOCUMENT (BIRD)**

**BIRD NUMBER:** 204

**ISSUE TITLE:** DQ\_DQS GetWave Flow for Clock Forwarding Modeling

**REQUESTOR:**  Walter Katz, The MathWorks

Fangyi Rao, Keysight

Wendem Beyene, Intel

Ambrish Varma, Cadence

**DATE SUBMITTED:** April 22, 2020

**DATE REVISED:**

**DATE ACCEPTED:** June 26, 2020 (Superseded by BIRD209)

**DEFINITION OF THE ISSUE:**

In DDR systems, DQ Rx uses DQS Rx output signal as a forwarded clock to clock the slicer and the DFE. To model such clock forwarding functionality, the DQ Rx model needs as input the DQS clock times. This BIRD proposes a new AMI Reserved Parameter to enable the EDA tool to input clock times from the DQS Rx AMI\_GetWave function to the DQ Rx AMI\_GetWave function.

DQ and DQS AMI models will be delivered as a matched pair. The model maker can put all the DQS path delays in the DQS AMI\_GetWave, all the DQS path delays in the DQ AMI\_GetWave or split the DQS path delays between the DQ and DQS AMI\_GetWave functions. There are several scenarios that demonstrate this:

1. The path delay from the DQS pad to all DQ latches in the chip are the same in a x4 lane and the same in a x8 lane, but the x4 path delay is different than a x8 path delay. In this case the x4 DQ and the x8 DQ would have the same DQ executable model. There would be different DQS executable models for x4 and x8 chips.
2. The path delay from the DQS pad to all DQ latches in the chip are different. In this case there would be one DQS executable model which had a common path delay (either 0 or some common path delay for all bits), and each bit would have a different DQ executable model (or other modeling configurations such as a different delay parameter value with a common DQ DLL) that would accommodate the different DQS path delay for each bit.

Since clock forwarding can be used for other applications, the words Clock and Data will be used in the BIRD in lieu of DQS and DQ.

**SOLUTION REQUIREMENTS:**

The IBIS specification must meet these requirements:

Table 1: Solution Requirements

|  |  |
| --- | --- |
| Requirement | Notes |
| 1. Allow the Rx AMI\_GetWave function to model clock forwarding. |  |

**SUMMARY OF PROPOSED CHANGES:**

Add new Reserved Parameter Rx\_Use\_Clock\_Input

**PROPOSED CHANGES:**

*Parameter:* **Rx\_Use\_Clock\_Input**

*Required:* No, and illegal before AMI\_Version 7.1

*Direction:* Rx

*Descriptors:*

Usage: In

Type:                     String

Format: List, Value

Default:                 *<*string\_literal>

Description:*<*string>

*Definition:* Specifies the content of the Data Rx AMI\_GetWave clock\_times input supported by the model. The three possible content types are: (1) to be ignored, (2) the clock\_times and (3) the wave output of the Clock Rx AMI\_GetWave. If this parameter is present in the .ami file, the EDA tool is responsible to pass the selected value to the AMI\_Init function.

*Usage Rules:* Allowed values are “None”, “Times” and “Waves”. If omitted, the default is “None”. If “None” is selected, then the content of clock\_times will be ignored by the model. If “Times” is selected, then the EDA tool will use the clock\_times values that were output by the Clock Rx AMI\_GetWave call as the clock\_times values in the call to the Data Rx AMI\_GetWave. If “Wave” is selected, then the EDA tool will use the wave values that were output by the Clock Rx AMI\_GetWave call as the clock\_times values in the call to the Data Rx AMI\_GetWave.

*Other Notes:* The wave input to both Data and Clock shall have the same block size and sample\_interval. For “Times” and “Wave” options, if the Clock does not have a DLL or has a DLL without an AMI\_GetWave, then the EDA tool should effectively insert a passthrough Clock AMI\_GetWave function to create the wave or clock\_times output vector required by the Data AMI\_GetWave clock\_times input.

*Example:*

(Rx\_Use\_Clock\_Input (Usage In) (Type String) (List “None” “Times”)

(Description "The model can use the Clock AMI\_GetWave output clock\_times

to sample the Data waveform at the Rx Data latch"))

+

-

…

…

DQ0

DQ Tx DLL

DQS Tx DLL

DQ Tx DLL

DQ7

DQS0

+

-

…

…

DQ8

DQ Tx DLL

DQS Tx DLL

DQ Tx DLL

DQ15

DQS1

DQ0

DQ7

+

-

+

-

DQS0

DQ8

DQS1

DQ15

…

…

…

…

DQS Rx DLL

DQ Rx

clock\_times

DQ Rx

clock\_times

DQS Rx DLL

DQ Rx

clock\_times

DQ Rx

clock\_times

1

1

1

1

1

1

2

2

3

3

3

3

Step 1: compute analog channel output according to IBIS 5.1-7.0 (crosstalk taken into account)

Step 2: compute output of all DQS Rx DLLs according to IBIS 5.1-7.0

Use either DQS Rx clock\_times or wave output values as DQ Rx clock\_times input values

Step 3: compute output of all DQ Rx DLLs

Channel

**BACKGROUND INFORMATION/HISTORY:**