**BUFFER ISSUE RESOLUTION DOCUMENT (BIRD)**

**BIRD ID#:**

**ISSUE TITLE:** *New AMI API to Resolve Dependent Model Parameter*

**REQUESTER:** *Fangyi Rao, Agilent Technologies, Inc.*

**DATE SUBMITTED:**

**ANALYSIS PATH/DATA THAT LED TO SPECIFICATION:**

AMI model parameters that are used by EDA tools can depend on other model parameters and simulation parameters including data rate, IBIS corner and IBIS model name. The form of such dependency relation varies from IC vendor to IC vendor and from device to device. The number of possible variations among vendors and devices is infinite. Model vendors need a flexible mechanism to implement parameter dependency according to their proprietary formula and pass the dependent parameter values to EDA tools. It’s foreseeable that certain vendors need to conceal the dependency formula.

A new API is added to AMI and a new reserved parameter is introduced. The API declaration is

*long AMI\_ResolveDependentParam(double bit\_time,*

*char \* corner,*

*char \* model\_name,*

*char \* AMI\_parameters\_in,*

*char \*\* AMI\_parameters\_out);*

Argument definitions are

**bit\_time:** input argument, in second, equals 1/data rate.

**corner:** input argument, ibis model corner, allowed values are “typ”, “min” and “max”.

**model\_name:** input argument, ibis model name.

**AMI\_parameters\_in:** input argument, a string that contains name-value pairs of all parameters of Usage Type In. The format of this string is the same as that of the AMI\_parameters\_in argument in AMI\_Init.

**AMI\_parameters\_out:** output argument, pointer to a string that contains name-value pairs of dependent parameters. The format of this string is the same as that of the AMI\_parameters\_out argument in AMI\_Init.

The new reserved parameter, ResolveDependentParam\_Exists, indicates whether the model implements the AMI\_ResolveDependentParam function and is defined as

(ResolveDependentParam\_Exists (Usage Info) (Type Boolean) (Default False)

(Description “Indicates whether DLL implements ResolveDependentParam.”) )

Independent parameters must be of Usage type In. Because their values are used to determine dependent parameters, they must not be updated by AMI\_Init and therefore must not be of type Out or InOut. Independent parameters must not be of type Info either as they are used by DLL.

Dependent parameters must be of Usage Type Info or In. Since their values are already determined by dependency relations, they must not be updated by AMI\_Init and therefore must not be of type Out or InOut.

The usage of the new API is described below.

1. User selects ibis model and specifies corner and data rate.
2. EDA tool initializes AMI\_parameters\_out to NULL.
3. If ResolveDependentParam\_Exists is False, go to step 9.
4. If ResolveDependentParam\_Exists is True, EDA tool allocates memory for the AMI\_parameters\_in string and writes to it name-value pairs of all parameters of Usage type In.
5. EDA tool calls AMI\_ResolveDependentParam before analog channel impulse characterization.
6. DLL computes dependent parameter values according to independent parameter values in AMI\_parameters\_in, bit\_time, corner and model\_name.
7. DLL allocates memory for the AMI\_parameters\_out string and writes to it name-value pairs of dependent parameters.
8. EDA tool sets/adjusts analog model parameters if their values are returned by DLL in AMI\_parameters\_out.
9. EDA tool characterizes analog channel impulse responses.
10. EDA tool calls AMI\_Init and passes the AMI\_parameters\_out pointer to DLL.
11. DLL free the memory of AMI\_parameters\_out. If AMI\_Init needs to return any parameter value, DLL must reallocate memory for AMI\_parameters\_out.
12. EDA tool finishes the rest of the simulation.

The new API provides model vendors infinite scalability, extensibility and flexibility to implement dependency relations. It also conceals the dependency formula. It allows any complex dependency relation. A few examples are listed below.

Example 1: multi-dimensional functions such as y = f(x1, x2, x3)

Example 2: various interpolation methods

Example 3: various extrapolation methods

Example 4: expression in condition statement such as



Example 5: advanced functions such as

y(tap1, tap2, tap3) = FIR(tap1, tap2, tap3) spectrum at data rate

The proposed approach does not require any ad hoc syntax or rule to be added for new dependency forms. Bit\_time, corner and model\_name are formal arguments of AMI\_ResolveDependentParam, therefore there is no need to introduce “simulation reserved parameters”. The same DLL can resolve dependent parameters for different ibis models according to the model\_name input argument. The API is a sensible partition between EDA tool and model, allowing model vendors to have full control on dependency definition as well as implementation.