New AMI API to Resolve Model Parameter Dependencies

Fangyi Rao and Radek Biernacki
Agilent Technologies, Inc.

IBIS Summit at DesignCon
Santa Clara, California
January 31, 2013

© Copyright Agilent Technologies 2013

Requirements

Model parameters used by EDA tools, including AMI and analog model parameters, could depend on other model parameters.

Model parameters could also depend on simulation parameters such as data rate, corner and IBIS [Model] name.

The form of dependency relation varies from IC vendor to IC vendor and from process to process – the number of possible variations 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.

© Copyright Agilent Technologies 2013

Requirements (cont'd)

Some vendors may need to conceal the dependency formula.

Resolving dependent parameters is a task that belongs to model.

Model developers should have full control on implementation – .ami files while convenient for updating are also susceptible to both intentional and unintentional modifications.

© Copyright Agilent Technologies 2013

Solution

BIRD 150 – Dependency Tables (in the .ami files) – it was the first step towards addressing the requirements.

BIRD 155 – New AMI API (in the DLLs).

The rest of this presentation is about BIRD 155 – AMI API to resolve dependent parameter values.

© Copyright Agilent Technologies 2013

Dependency Resolution API

long AMI_ResolveDependentParam(double bit_time,
*char * corner,*
*char * model_name,*
*char * AMI_parameters_in,*
*char ** AMI_parameters_out)*

bit_time: input argument, in *sec*, 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 *In* parameters. 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*

© Copyright Agilent Technologies 2013

Proposed New Reserved Parameter: ResolveDependentParam_Exists

(ResolveDependentParam_Exists (Usage Info) (Type Boolean) (Value True)

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

© Copyright Agilent Technologies 2013

Splitting *Out* Parameters

The function *AMI_ResolveDependentParam* belongs to the configuration process that takes place before the start of simulation and before TX/RX instantiation.

The function *AMI_Init* belongs to the instantiation process.

Therefore, the integrity of the user selected values must be preserved and is achieved by

- splitting the entire set of ***Out*** parameters into two complementary subsets of parameters, one handled by the *Resolve* function and the other by the *Init* function
- using separate memory allocations for the strings *AMI_Parameters_out* to return data to the EDA platform from the *Resolve* function and from the *Init* function

© Copyright Agilent Technologies 2013

Usage Type of Independent and Dependent Parameters

Independent parameters can only be of type ***In*** because

- their values are used to resolve dependency; they cannot be updated by *AMI_Init* and therefore cannot be of type ***Out*** or ***InOut***
- they are used by *DLL* so they cannot be of type ***Info***

Dependent parameter allowed type is ***Dep***

their values are already determined by dependency relations; they cannot be updated by *AMI_Init* and therefore cannot be of type ***InOut*** or ***Out***

A new Usage Type "***Dep***" is proposed. It is similar to type ***Out***, but is exclusively intended for those parameters that are to be determined by the dependency relations. Parameters of this type are not intended to be included in *AMI_Parameters_in* passed to *AMI_Init.*

© Copyright Agilent Technologies 2013

Usage

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 ***In*** parameters.
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*.

© Copyright Agilent Technologies 2013

Usage (cont'd)

7. DLL allocates memory for the *AMI_parameters_out* string and writes to it name-value pairs of dependent parameters that are subject to resolution by DLL.
8. EDA tool sets/adjusts analog model parameters to the values that 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 frees 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.

© Copyright Agilent Technologies 2013

Advantages

- Infinite scalability, extensibility and flexibility for model developers
- Conceal dependency formula
- Allow any complex dependency relations
  - multi-dimensional function (e.g. *y* = *f*(*x1, x2, x3*))
  - different interpolation methods
  - different extrapolation methods
  - expression in condition statement, e.g.,

$$y(x1, x2, x3) = \begin{cases} f(x3) & if \quad x1 + x2 < 0 \\ g(x3) & if \quad x1 + x2 \geq 0 \end{cases}$$

  - advanced functions, e.g.

    y(*tap*1, *tap*2, *tap*3) = *FIR*(*tap*1, *tap*2, *tap*3) spectrum at data rate

© Copyright Agilent Technologies 2013

Advantages (cont'd)

- No need to add ad hoc syntax or rules for new dependent forms.
- *bit_time*, *corner* and *model_name* are formal arguments of. *AMI_ResolveDependentParam – there is* no need to introduce the awkward "simulation reserved parameters".
- The same DLL can resolve dependent parameters for different IBIS models according to *model_name.*
- Sensible partition between EDA tool and model: the party that defines dependency has full control on implementation.
- It doesn't mean DLL has to be regenerated each time dependency is updated: there are many ways to reuse DLL and update dependency using auxiliary files or regular expressions specified in .ami files – model makers can utilize all appropriate means.

© Copyright Agilent Technologies 2013

Summary

A new BIRD 155 has been proposed.

It is considered as a replacement to BIRD 150.

It provides more flexibility.

It guarantees better consistency between EDA vendors.

© Copyright Agilent Technologies 2013