The IBIS-X Specification

Stephen Peters,
Intel Corp.

Agenda

- What is IBIS-X? How is it different from IBIS 3.2?
- Backwards Compatibility Concerns
- An Introduction to Defining New Models
- Walk thru of the IBIS-X document itself

What is IBIS-X?

- More than just a ‘template tweak’ of IBIS 3.2
  - Formalizes fundamental idea behind IBIS: separation of model data from the EDA tool algorithm(s) that processes data
  - ***IBIS-X introduces the mechanism whereby a file creator can tell the EDA tool how to process the data*** – i.e. the user can now “create a model” by using a new [Define] keyword and a nodal-based ‘macro language’
  - IBIS-X will eventually include a nodal-based package description
- IBIS-X spec itself still focuses on the model data transfer standard
  - IBIS Macro Language (IBIS-ML) is addressed in a separate Language Reference Manual (LRM)

Backwards Compatibility

- Existing IBIS keywords are unchanged
  - IBIS-X capable EDA tool can still parse a 3.2 file with a 3.2 parser (or IBIS-X parser in ‘compatibility mode’)
    - *All existing models still work with IBIS-X (see below)*
  - Syntax changes to data template format are transparent to operation of simulation
- Transition between IBIS -> IBIS-X?
  - IBIS Open Forum will (must!) provide
    - *IBIS-X parser and (by default) IBIS-X to “SPICE” converter*
    - *Library of all existing IBIS 3.2 model types written in IBIS-ML*
    - *Existing IBIS 3.2 models documented in Library guide*

Introduction to Creating New Models

- An I/O buffer model is a specific example of a simulation *object*
  - Simulations objects are grouped into “*classes*”, different objects within a class can be of different “*type*”
- [Model] is a class of object, while “input”, “output”, “3-state”, etc. are of different types within the class
  - Notice: objects are not limited to I/O buffer models. [Test Load] is a class of simulation object, as is [Submodel].
- User can create a new type within the ‘Model’ class, or create a whole new class

Creating New Models (cont.)

- Object (model) creation is a two step process
  - The structure and behavior of a simulation object is defined using the [Define “class name”] keyword and described using IBIS-ML constructs – I.e. the model creator builds an object “*template*”
  - The IBIS-X file then supplies a specific set of simulation data for that object – I.e. the file creator “*customizes*” an object template
- To be used, an object has to be both ‘defined’ and ‘customized’

The [Define “class name”] keyword

- Syntax:

```
[Define “class_name”] “type name” (<optional port list>)
|
| IBIS-ML Constructs
|
[End “class name”] “type name’
```

- [Define] keyword can be used in IBIS-X file itself – or – [Defines] are collected into a *library file* (‘included’ into an IBIS-X file)
- Once an object is defined, it is customized as shown:

```
[Begin “class name”] “object name”
“class name”_type  “type name”
|
| optional data specific to this instance
|
[End “class name”]
```

- Object customization takes place in the IBIS-X file

Specific Example (from section 8)

- **Create a template for a differential receiver (new class and type)**

```
[Define Receiver] differential (in_pos, in_neg, pwr, gnd)
|
| IBIS-ML constructs
|
[End Reciever] differential
```

- **To create object named ‘data_receiver’, customize this template**

```
[Begin Receiver] data_receiver
Receiver_type differential
|
| data set for this model
|
[End receiver]
```

Specific Example (cont.)

- **To create a second object ‘data_receiver_alt’ from same template…**

```
[Begin Receiver] data_receiver_alt
Receiver_type differential
|
| data set specific to data_receiver_alt
|
[End receiver]
```

- **User now has two models (objects): ‘data_receiver’ and ‘data_receiver_alt’**

Overview of the IBIS-X Document

- 7 major technical sections
  - General Syntax Rules and Guidelines
  - IBIS File Header & Global Keywords (2 sections)
  - Component Information (unchanged from current IBIS)
  - New Model Definition (replaces [Model] documentation)
  - Package Modeling (placeholder for new .pkg model)
  - EBD (unchanged from current IBIS)
- IBIS-X spec does not address macro language, but will include a section on “data structures”

General Syntax Rules & Guidelines

- Like connector spec, seek to remove limits
  - 120 character lines (rule 7)
- Clarify usage, resolve ambiguities
  - Added ‘arguments’ to keywords and explicitly defined ‘text blocks’ & their termination
  - Explicit definition of ‘sections’
  - Allow text before [Begin Header] and after [End] (rule 13)
- Also, added a keyword tree diagram

IBIS File Header & Global Keywords

- Same intent, with a few additions
  - Added begin/end blocking and IBIS-X specific version marker
    - *[Begin Header], [End Header], [IBIS-X Ver]*
  - Added Support for model library include function
    - *[Include Library]*
- “steal with pride” connector spec ideas
  - Added support and redistribution information keywords
    - *[Support], [Redistribution], [Redistribution Text]*
- Added a keyword for “include text” function
  - [Include] includes a text file, is global is scope

Component & “Model” Section

- Component section basically unchanged from current IBIS 3.2
  - Clarified that [Model Selector] is global in scope
  - Added [End Component] to match [Begin Component]
- New “Models” section documents the model definition process
  - Introduces concept of object *classes* and *types* within a class
    - *[Define “class name”] mechanism for creating a new object (model)*
  - Discusses object template and *customizing* an object
    - *[Begin “class name”] keyword for supplying data to object (model)*
  - Need to add discussion on model data ‘types and structures’

Package Modeling & EBD sections

- Package modeling is part of IBIS-X spec
  - IBIS-X will incorporate, by reference, current .pkg file syntax
  - Section is placeholder for nodal-based package modeling
- Electrical Board Description (EBD) is incorporated unchanged from IBIS 3.2
  - EBDs still serves a useful purpose