IQC – IBIS QUALITY CHECKER

Manfred Maurer  manfred maurer@siemens.com
Dr. Christian Sporrer  christian sporrer@infineon.com
Overview

- **Motivation**
  - SIG → „Siemens“ IBIS Group
  - Improve the quality of the IBIS files

- **Characteristics of the Quality Checker**
  - begin with the most important tests
  - modular structure
  - improvements – step-by-step

- **Example of an IQC**

- **Summary**
SIG – SIEMENS IBIS GROUP

SIG ➔ „Siemens“ IBIS Group

➔ The goal we had ➔

➢ On the SIEMENS WEB-PAGE for ALL to use
  http://ibis.siemens.com/home

➔ „…define the desired quality level but also explain in detail what we need, and why we need it.”

➔ “...we expect from our IC vendors high IBIS model availability and quality …”

➔ “…hints and examples for IBIS modeling…”

➢ All the results of the SI / EMC – tools are ➔ as good as the models are!

➢ SIG-Members: ➔ SIEMENS ➔ Continental
  ➔ Nokia Siemens Networks
  ➔ Fujitsu-Siemens
  ➔ INFINEON

IBIS Summit Meeting Munich 2008
Overview

Motivation

Characteristics of IQC

Example of IQC

Summary

IBIS Quality Task Group

– Chairman: Mike LaBonte

- EXCEL-sheet with quality issues listed (each model ca. 80 lines)
- Very extensive and time consuming checks
- EXCEL-sheet should be part of the model
- Low acceptance, if made by hand
- Feasible, only if at least in major parts automated
- 20-FEB-2008: The IBIS Quality Specification 1.1 is currently under review

Excerpt from the EXCEL-sheet:

| IQ | In column B below mark each item PASS, FAIL, or N/A. The IQ level is the highest level number for which all checks PASS. The IQ level is FAIL if no level completely passes. See IQ_specification.txt for complete descriptions of the checks. |
| IQ | 4.1.1 | LEVEL 0 | [Model] parameters have correct typ/min/max order |
| IQ | 4.1.2 | LEVEL 0 | [Model] Model_type |
| IQ | 4.1.3 | LEVEL 0 | [Model] C_comp is reasonable |
| IQ | 4.1.4 | LEVEL 1 | [Model] C_comp is correct |
| IQ | 4.1.5 | LEVEL 2a | [Model] C_comp SPICE correlation |
| IQ | 4.1.6 | LEVEL 2b | [Model] C_comp laboratory correlation |
| IQ | 4.1.7 | LEVEL 1 | [Temperature Range] is reasonable |
| IQ | 4.1.8 | LEVEL 1 | [Voltage Range] or [* Reference] is complete |
Quality inside a IBIS model

- Advantages for all:
  - increased confidence of the model
  - better comparability between different models

- Disadvantages for the:
  - VENDOR ➔ very time consuming
  - USER ➔ restricted consistency checks only
    ➔ NO possibility to prove the quality info
    ➔ vendor „specific“ doing ➔ time consuming

Conclusion: ➔ uncertain cost-value ratio
Remedy: Automation

Overview

Motivation

Characteristics of IQC

Example of IQC

Summary

Old idea

- every company has its own scripts,
  - for testing of different kind of quality issues

Better idea:

- to have the same quality-check for all available
- to improve the quality only once, at the origin (vendor)
- to use all the possible synergy effects
- if the checker is for free,
  - more acceptance and dissemination will be gained
Thoughts about an „Automatic“ Quality Check

Based on

- EXCEL-sheet summary of the Quality Group Initiative
- Needs & requests from the SIG-members

One possible Proceeding

- Concept of the Quality Checker
  - Priorities should be set by Pareto-principle (80/20)
  - A lot of constructive discussions needed

- Modular structure
  - new checks should be included as „plugins“
  - implementation can be distributed on different places/companies

- Transfer of the Quality Checker to the Quality Group
  - maximum usage for all model VENDORS & USERS
  - concentrating all activities at a central point of access

IBIS Summit Meeting Munich 2008
What should the IQC do?

- **Consistency checks for parameter and curves**
  - inside the IBIS model
  - against Data Sheet
  - EBD / PKG / Pinning

- **Curves evaluation and checks**
  - Compare and find identical tables
  - Extract and compare the driver strength

- **Statistics**
  - Number of models / #pins w/o models
  - Parameter of interest
First step of an IQC was implemented by Dr. Christian Sporrer

- Presentation
  - Input files
  - Concept
  - Implementation of initial checks
  - Output processing
IQC Implementation Overview

Overview
Motivation
Characteristics of IQC
Example of IQC
Summary

IQC Implementation Overview

Library
- IBIS File
  - IBIS Model File
- Parameter File
  - processParameterFile
- Configuration File
  - processConfigurationFile

Encapsulation
- Results
  - processResults
- executeIBISChecks
- executeDatasheetChecks
- statistics

IQC
- check 1
- check 2
- check n
- executeIBISChecks
- executeDatasheetChecks
- statistics

Utilities
IQC Input Files  (Syntax)

- Same syntax for parameter and configuration files
- Grouped into sections: (@ section delimiter)
  
  @A Package
  @B Pin
  @C Model (general)
  @D Model specific information
  @D1 Model class 1
  @D2 Model class 2
  @E Datasheet

- Parameter entry:
  
  <keyword> :no of parameters: <parameter 1> <parameter 2> ..... # comment

  <keyword> is every IBIS keyword (e.g. RAMP)
  Non IBIS keywords are indicated with the prefix NI_ (e.g. NI_technology)
  <parameter> is a number or string (e.g. cmos for NI_technology)

- Example:
  
  @A Package
  #==========
  #
  R_pkg : 3 : 0.5 1 0.1 # resistor range
  L_pkg : 3 : 2e-9 5e-9 0.1 # inductance range
  C_pkg : 3 : 2e-12 5e-12 0.1 # capacitance range

IBIS Summit Meeting Munich 2008
IQC Input Files  (Configuration File)

Configuration File:

- Contains a set of 'more general' parameters useful for the check of certain classes of IBIS models
- Defines default values to enable a first basic model check

Example:

@C Model

#======
C_comp : 2 : 10.0e-15 30.0e-12 # min_value max_value
Voltage Range : 2 : 1.0 6.0 # min_value max_value
Temperature Range : 2 : 0 150 # min_value max_value
D_overshoot_high : 1 : 9.0 # max_value
D_overshoot_low : 1 : -3.0 # min_value
D_overshoot_time : 1 : 100e-9 # min_value
NI_short_current : 2 : 1.0e-6 1.0 # min_value max_value

# Ramp
dV/dt_r : 2 : 3.0/0.2e-9 8.0/0.2e-9 # min_value max_value
dV/dt_f : 2 : 3.0/0.2e-9 8.0/0.2e-9 # min_value max_value
IQC Input Files (Parameter File)

Parameter File:

- Contains all parameters needed to check a specific IBIS model
- Refines most of the values of the configuration file
  (the parameter file overrules the configuration file)
- Can contain specific parameters not defined in the configuration file

Example:

```plaintext
[C Model]

# C_comp : 2 : 0.1e-12 5.0e-12  # min_value max_value
Voltage Range : 2 : 1.4 1.6  # min_value max_value
Temperature Range : 2 : 0 90  # min_value max_value
D_overshoot_high : 1 : 4.6  # max_value
D_overshoot_low : 1 : -1.0  # min_value
D_overshoot_time : 1 : 3.0e-9  # min_value
NI_technology : 1 : cmos  # (cmos oder bip)
NI_short_current : 2 : 1.0e-3 25.0e-3  # min_value max_value

# Ramp
dV/dt_r : 2 : 0.3/0.2e-9 0.8/0.2e-9  # min_value max_value
dV/dt_f : 2 : 0.3/0.2e-9 0.8/0.2e-9  # min_value max_value
```

IBIS Summit Meeting Munich 2008
The IBIS model file contains the IBIS model to be checked

**Example:**

<table>
<thead>
<tr>
<th>[Model]</th>
<th>MODEL_GPIO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model_type</td>
<td>3-state</td>
</tr>
<tr>
<td>Polarity</td>
<td>Inverting</td>
</tr>
<tr>
<td>C_comp</td>
<td>typ min max</td>
</tr>
<tr>
<td>Vmeas = 0.750000</td>
<td>2.00e-12 NA NA</td>
</tr>
<tr>
<td>Cref = 2.000000e-11</td>
<td></td>
</tr>
<tr>
<td>[Temperature range]</td>
<td>typ min max</td>
</tr>
<tr>
<td>8.00e+01</td>
<td>1.25e+02 0.00e+00</td>
</tr>
<tr>
<td>[Voltage range]</td>
<td>typ min max</td>
</tr>
<tr>
<td>1.50e+00</td>
<td>1.40e+00 1.60e+00</td>
</tr>
</tbody>
</table>
File IO Methods:

- **processConfigurationFile:**
  - readConfigurationFile()
  - getConfigurationVoltageTolerance()
  - getConfigurationTypVoltageRange()
  - getConfigurationTemperatureRange()
  - getConfigurationStepGradient()
  - getConfigurationNumberOfPins()

- **processParameterFile:**
  - readParameterFile()
  - getParameterPackageLimits()
  - getParameterDatasheetModels()
  - getParameterDatasheetPins()
  - getParameterTypVoltage()

- **processIBISFile:**
  - readIBISFile()
  - getIBISPackages()
  - getIBISPinListPackageAvgMinMax()
  - getIBISPinList()
  - getIBISPinMapping()
  - getIBISModelList()
  - getIBISModel()
  - getIBISModelType()
  - getIBISModelVoltageRange()
  - getIBISModelTemperatureRange()
  - getIBISModelRamp()
  - getIBISModelTestLoad()
  - getIBISModelIVTable()
  - getIBISModelCurrentSum()
Computing Methods:

- **executeIBISChecks:**
  - compareValues()
  - valueSequence()
  - valueSequenceArray()
  - checkPinForModel()
  - checkIVTableOrder()
  - checkRampOrder()
  - checkStairStepping()
  - checkTestLoadCurrent()

- **executeDatasheetChecks:**
  - checkDatasheetModels()
  - checkDatasheetPins()

- **statistics:**
  - countPins()
  - countModels()
  - globalPackage()
  - filePackage()

- **utilities:**
  - file2string()
  - file2array()
  - interpolate()
  - mergeXAxis()
IQC Simple Checks for Demonstrator

Checks implemented for the IQC Demonstrator:

- **Section A (Package):**
  - Range check for global package values
  - Compliance check for pinlist RLC values and global package

- **Section B (Pinning):**
  - Number of pins
  - Pins connected to model

- **Section D (Model):**
  - Voltage range
  - Temperature range
  - Ordering of ramp values
  - Ordering of IV tables

- **Section E (Datasheet):**
  - Comparison of model names
  - Comparison of pins

IBIS Summit Meeting Munich 2008
How to create a new check?

- More general functions which can be used for several checks should be implemented as method into the IQC library. (High level of reuse, high level of modularity).

- The check itself is implemented as a method call or a combination of method calls from the IQC library in the IQC main file.

- The processing of the results is done using methods from the output processor (not yet implemented).
### Voltage Check Plug In ###

# read typ supply voltage range and voltage tolerance from configuration file
my @configFile = $processConfigurationFile->readConfigurationFile($options{cfg}, $path, 'array');
my %vcc_typ_range = $processConfigurationFile->getConfigurationTypVoltageRange(@configFile);
my %vcc_tolerance = $processConfigurationFile->getConfigurationVoltageTolerance(@configFile);

# read IBIS voltage range
my @model = $processIBISFile->getIBISModel(@ibisFile, $modelname);
my %voltage = $processIBISFile->getIBISModelVoltageRange(@model);

# supply check
if($executeIBISChecks->compareValues($voltage{typ}, $vcc_typ, "=")){
    processResult->print("supply check:  ok    
    
");
} else{
    processResult->print("supply check:  error 
    
");
}

# sequence check
if($executeIBISChecks->valueSequence($voltage{min}, $voltage{typ}, $voltage{max})){
    processResult->print("sequence check:  ok    
    
");
} else{
    processResult->print("sequence check:  error 
    
");
}

# range check
if($executeIBISChecks->valueSequence($vcc_typ_range{min}, $voltage{typ}, $vcc_typ_range{max})){
    processResult->print("range check:  ok    
    
");
} else{
    processResult->print("range check:  error 
    
");
}
IQC Output Processor

**Output processor:**
- Basic print functionality
- Sophisticated output processing (not yet implemented)
  - Sorted by kind of test, error type, quality level, etc.
  - Configurable output formats

**Example:**

```
========== Model Checks ============
+++++ Name: MODEL_3549_2411_7  Type: 3-state ++++

D.1 Voltage Checks:
  Correct ordering of IBIS voltage values: [x]
  Typical IBIS voltage is compliant to parameter file: [x]

D.2 Temperature Checks:
  Correct ordering of IBIS temperature values (CMOS): [x]
  Typical IBIS temperature is in range of configuration file (CMOS): [x]

D.4 Ramp Checks:
  Correct ordering of IBIS rising ramp values: [x]
  Correct ordering of IBIS falling ramp values: [x]

D.5 IV Table Checks:
  Correct ordering of IBIS I-V table values:
    Pulldown: [x]
    GND_clamp : [ ]
    Pullup: [x]
    POWER_clamp: [ ]
```

IBIS Summit Meeting Munich 2008
The success of the IBIS QUALITY TASK requires a commonly accepted procedure of automation.

The benefits of an IQC are indisputable, but no one will start, yet.

Large amount of work, to implement the tool, with “all” of the quality checks included at once.

Existing local solutions can be integrated into a powerful centralized solution to the benefit of the whole IBIS community.

If modular concept is used, a reasonable IBIS QUALITY CHECKER should be reachable, which is “self-improving” by use.