

Achieving Interoperability through iParams

Enabled by OpenAccess

Neel Gopalan

21st EDA Interoperability Forum

November 6, 2008

Agenda

- Properties and Parameters available in OA today
- Need to extend
- Discuss iParam semantics
- iParams as stored in OA
- Show interoperability of iParams
- Summary

Proprietary Database

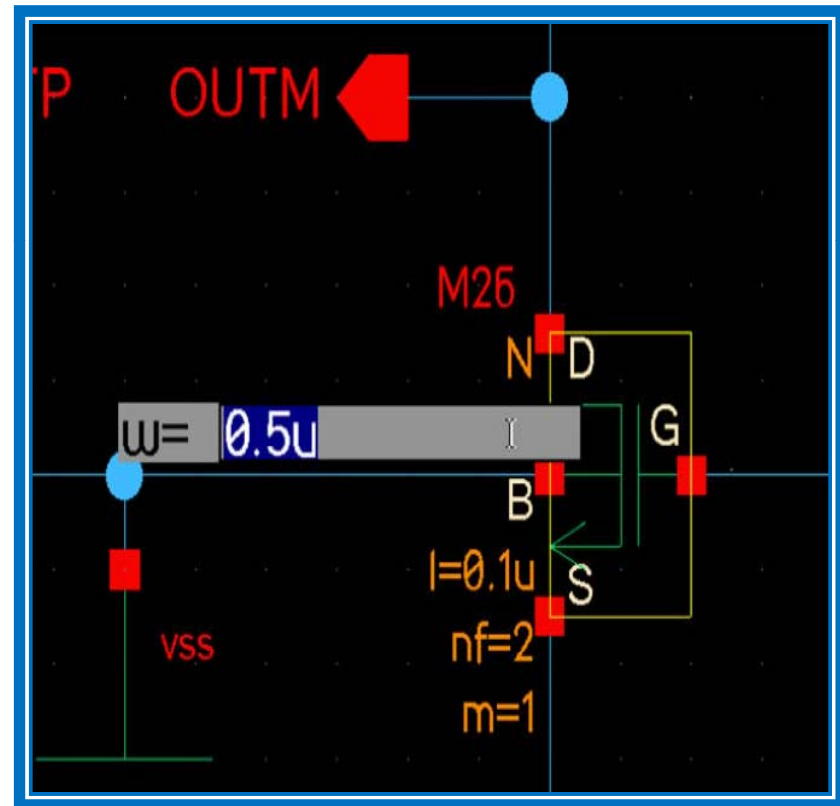
- OpenAccess is an enabler and set the stage
 - Provides access to core design data
 - Shapes, instances, connectivity, some technology
 - Being adopted by key vendors in several areas
 - Synopsys, Mentor, Silicon Canvas, Cadence & Magma
- BUT:
 - Leaves a lot to semantic interpretation
 - Doesn't include guidance for how to interpret and interoperate certain key data
 - Parameters
 - Definition
 - Expression languages
 - Parameter passing
 - Interpreted labels



OpenAccess

Zoomed in schematic:

- Pins / terminals
- Interconnect
 - Paths
 - Lines
 - Dots (ellipses)
- Instances
 - Parameters
 - Interpreted labels



OpenAccess

- Notice the number of parameters and interpreted labels?
 - More parameters and labels on a device than shapes on the symbol
 - 100+ parameters on current technology node transistor
 - Without the parameters, a schematic is useless
- Through IPL we have achieved
 - Common Pcells
 - Common interpreted labels
 - Common parameter definitions
 - Common parameter expression language
- The result, at DAC 2008:
 - Synopsys Custom Designer
 - SpringSoft Laker
 - Mentor Calibre



Interoperable Parameters (iParam)

- Goal
 - Create standards for interoperable parameters to describe schematics
 - Use OA Parameter / Property System
 - Describe semantics
 - Expression language
 - Ability to reference local and passed parameters
 - Customize parameter definitions using scripts



iParam Attributes

- *name* – iParam Name
- *type* – Parameter type
 - String, Boolean, Int, Float, Radio, Cyclic, Button, netSet
- *prompt* – The name that will appear in property editor GUI
- *units* – lengthMetric, capacitance, ...
- *defVaule* – default value of the parameter
- *callback* – procedure that gets triggered by changing a parameter
- *display* – should the parameter be displayed in property editor
- *editable* – should the parameter be editable in property editor
- *parseAsPEL* – parameter expression language, stored in OA as
 parseAsCEL
- *parseAsNumber* – should the parameter be interpreted as a number
- *storeDefault* – should the parameter be stored on the instance



iParam in OA

- All iParams are stored as a string in OA
- It is an oaAppProp of type “ILList” named “cdfData”
 - Format is structured Lisp list
- iParams can be stored at Library or Cell level
 - oaLibDMDData or oaCellDMDData
 - Effective definition comes from cell, for clashes
 - Support for legacy tools also included
 - Example : promptWidth, fieldHeight,...



Parameter Value Resolution

- Concept of default value
- Local overrides
 - Stored on oaRef's by creating an oaProp / oaParam
 - Same name as iParam definition
- Precedence for Pcells
 - oaParam stored on oaInstHeader
 - oaParam stored in SuperMaster
 - oaProp stored on the instance
 - defValue at cell level
 - defValue at library level



Callbacks

- Scripting language required
 - To describe certain attributes -- Visibility, Editability
 - Calculate other iParams based on new value
- Syntax is a subset of common Lisp
 - To support Infix operators & C-style function calls
 - Used as a compatibility layer
 - Extensive usage of IL in parameter definitions in existing design data
 - <Identifier>(<args>)
- Advanced Support
 - Extensive use of *cdfgData*
 - Support for *cdfgData->param->value*



Parameter Expression Language

- Support for
 - arithmetic
 - hierarchical parameter passing
 - design variables
 - engineering notation
- Parameter Access Functions
 - inst, parent, lineage
 - Support for legacy data
 - iPar, pPar, atPar, dotPar



Substitution Expression Language

- Required for netlisting and interpreted label support
- A simple text substitution format
 - [`@cellName`], [`@instanceName`]
- SEL evaluation done when local values is an `oaAppProp`, type “`nlpExpr`”



Summary

- Interoperability is becoming a reality
 - Design data via OpenAccess
 - Consistent usage within OpenAccess
 - Interoperable PDKs via IPL
 - Not only Pcells
 - But for parameters to describe schematic

