**Link: “**[**Introduction to SKILL | Multifunctional Integrated Circuits and Systems Group (MICS) | Virginia Tech (vt.edu)**](https://www.mics.ece.vt.edu/ICDesign/Tutorials/Cadence/layout_pg5.html)**”.**

Introduction to SKILL

*Authors: Jeannette Djigbenou and Meenatchi Jagasivamani*

The SKILL language has been developed by Cadence to be used with their tool suites. It allows the user to write a "script" to perform any command in Cadence. SKILL is an interpretive language like *LISP* and *Perl*. SKILL was designed to work on repetitive tasks and several of its functions are based on lists.

For more information about the structure of SKILL code, you can go to the following sources:

* Finder: Allows the user to search its database by entering a keyword. Finder will display all functions containing the keyword. For each function, it gives the usage and a brief description about the function.

To invoke: From the CIW, type startFinder()

* cdsdoc: Detailed manual about Cadence that includes function reference and user guides to SKILL. It contains the functions for the other cadence tool suites as well.

To invoke: From the terminal, type cdsdoc & --> click on the SKILL and SKILL PI menu

* Look at the following tutorial: [**skill.pdf**](http://www.vtvt.ece.vt.edu/tutorial/skill.pdf)
* [**See section below for information on how to create pins & symbols**](http://www.vtvt.ece.vt.edu/tutorial/tutorialCadence_skill.php#pins)

A simple SKILL procedure is given as an example below. The objective of this procedure is to modify all nfet3 and pfet3 objects' bulk node value to be vss! and vdd!. This function is library specific and is given here as an example only.

**To execute this procedure:**

1. Copy the following procedure and save it as "ChangeProperty.il"
2. From the CIW, type: load "ChangeProperty.il"
3. To execute the procedure, type: cb(libraryname cellview)

Example: cb("vlsi\_proj" "tspc\_rs")

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;This procedure will change the bulk node values of nfet3 and pfet3

;;to be vss! and vdd!, respectively.

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

procedure(cb(lib cell)

;let will allow you to access the cv and newlabel variables outside

;of this procedure

let((cv newLabel)

;open the cellview as a database object and assign it to cv

cv = dbOpenCellViewByType(lib cell "schematic" "" "a")

;for loop to go through all instances in this cellview

foreach(instancecv~>instances

;open the property window -- this will display the window!

geSelectObjectNoFilter(instance)

schHiObjectProperty()

;check the cellname and change the bulk node (bn) property

if(schObjPropForm->cellName->value == "nfet3"

then

schObjPropForm->bn->value = "vss!"

)

if(schObjPropForm->cellName->value == "pfet3"

then

schObjPropForm->bn->value = "vdd!"

)

;close the form

hiFormDone(schObjPropForm)

geDeselectAll()

);end foreach

);end of let

);end procedure

Building Nets, Terminals, and Pins

The following steps show how to define pins in a layout.

1. Create the shape that will serve as the pin. The shape is usually a rectangle. Note: The shape cannot be a polygon. fig = dbCreateRect( pcCellView layer list( x1:y1 x2:y2))

2. Create the net to which the pin attaches. In this example, the pin name n1 matches the name of the corresponding pin in the schematic symbol for this cell: net = dbCreateNet( pcCellView "n1")

3. Create a terminal on the same net. The terminal must have the same name as the net and match the terminal type. In this example, the terminal type is inputOutput, the same terminal type as the corresponding pin in the schematic symbol: trm = dbCreateTerm( net "n1" "inputOutput")

4. Create a pin: pin = dbCreatePin( net fig "n1")

The pin database object connects the pin figure with the net. The pin name can match the terminal name, but does not have to. In the example, the pin name n1 matches the terminal name. Within the pcell, you can have multiple shapes that all belong to the same electrical terminal. Each shape would have a pin to associate it to the same net. In such cases, each pin is created on the same net and must have a unique pin name.

5. If your tool requires pins to have an access direction, define the access direction: pin~>accessDir = '( "top" "left")

The access direction is a list identifying the correct sides of the pin shape for connection.