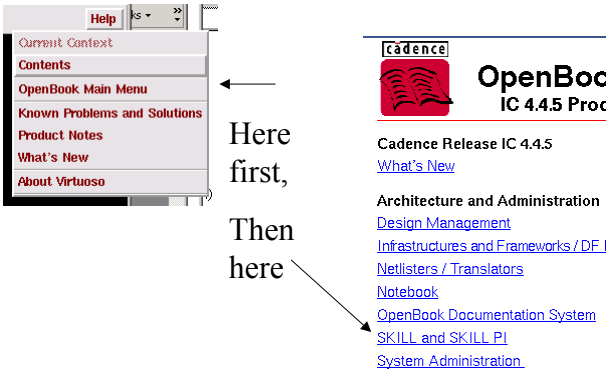
SKILL – Cadence Extension Language

- *SKILL* is powerful extension language that can be used to add new capabilities to Cadence tools
- SKILL is based upon LISP, will look very strange if you are not already familiar with LISP
  - LISP is a interpreted language that is popular among the AI community
  - LISP has a built-in *eval* function that can be used to execute LISP code that is dynamically generated
  - The basic data structure in LISP is the list, with many built-in functions for manipulaing list data structures
  - SKILL also supports a syntax form that is more 'Pascal'-like
- The key to SKILL's power is a large set of library functions that allow you to manipulate data structures such as cells, nets, mask information, etc.

To Get Help On Skill

- To get help on SKILL, click on the Help menu from within the Cadence layout editor, then on "*Openbook Main Menu*". Choose the 'SKILL and SKILL PI to open the Skill documentation.



A Sample SKILL Function

This SKILL function will create a padframe with X number of pads per side:

```
procedure( placePadFrame( @optional no_pads )
 (if (null no_pads) then
   (setq no_pads (enterNumber ?prompts '("Please enter the number of
pads on a side"))))
   (setq xpointh 291)
   (setq ypointh 201)
   (setq xpointv 201)
   (setq ypointv 291)
   (for i 1 no_pads
        (dbCreateInst (getEditRep) (dbOpenCellViewByType "tutorial"
"PADNC" "layout") nil (list xpointh ypointh) "R180")
        (dbCreateInst (getEditRep) (dbOpenCellViewByType "tutorial"
"PADNC" "layout") nil (list xpointh-90 ypointh+no_pads*90) "R0")
        (setq xpointh xpointh+90)
        )
```


SKILL function (cont)

```
 (for i 1 no_pads
   (dbCreateInst (getEditRep) (dbOpenCellViewByType "tutorial"
"PADNC" "layout") nil (list xpointv ypointv-90) "R90")
  (dbCreateInst (getEditRep) (dbOpenCellViewByType "tutorial"
"PADNC" "layout") nil (list xpointv+no_pads*90 ypointv) "R270")
   (setq ypointv ypointv+90)
        )
 (dbCreateInst (getEditRep) (dbOpenCellViewByType "tutorial" "PADFC"
"layout") nil (list xpointv-300 ypointv-90) "R0")
 (dbCreateInst (getEditRep) (dbOpenCellViewByType "tutorial" "PADFC"
"layout") nil (list xpointh-90 ypointv+210) "R270")
(dbCreateInst (getEditRep) (dbOpenCellViewByType "tutorial" "PADFC"
"layout") nil (list xpointh+210 201) "R180")
 (dbCreateInst (getEditRep) (dbOpenCellViewByType "tutorial" "PADFC"
"layout") nil (list 201 -99) "R90")
)
```

Comments on SKILL function

- Contained in file called 'pads.il'
  - To load function, in icfb command line type "load pads.il"
  - To execute function, have a layout view open and type "placePadFrame 10" if you want 10 pads per side
- Uses the *dbCreateInst* function for instance creation
  - Documented in *DFII SKILL Functions Reference*
- Function parameters are:
  - d_cellview - cellview where instance is placed
  - d_master – master cell view of the instance to be created
  - t_name – instance name. If 'nil' is used, then generate an instance name
  - l_point – origin of new instance as 2-element list
  - orientation of new instance as a string, some possible strings are "R0", "R90", "R180", "R270"

dbCreateInst

- The function *getEditRep* was used to return the currently open cell view
- The function *dbOpenCellViewByType* was used to specify the master view of the instance to be placed.
  - The minimum set of parameters to *dbOpenCellViewByType* are library_name, cell_name, view_name
  - See docs for other optional parameters
- The *list* function used to create a list required to pass instance origin
  - *(list first_elem second_elem .. N_elem)* returns a N-element list

Creating an Rows x Cols Array of Instances

```
procedure( placeArray( @optional cols rows cellname x_width y_height)
  (setq ypnt 0)
  (for i 1 rows
    (setq xpnt 0)
    (for j 1 cols
       (dbCreateInst (getEditRep) (dbOpenCellViewByType "tutorial"
cellname "layout") nil (list xpnt ypnt) "R0")
       (setq xpnt xpnt+x_width)
     )
     (setq ypnt ypnt+y_height)
  )
)
```

Tested with standard cell instance via:
placeArray 20 4 “INVX1” 4.8 21.6

placeArray 20 4 “INVX1” 4.8 21.6