Application Note for Customized Device in TSMC PDK

4/21/2004 v0.1

Introduction

This document is an application note for TSMC PDK users who want to add customized devices or 3rd IP into TSMC PDK. There are three parts for users from data preparation, adding devices flow, to design flow.

1. **Data preparation**
    - Layout
    - Netlist or model
        i. Simulation netlist for IP or model for device
        ii. LVS netlist for IP or device
        iii. Pin order, name and position
2. **Add customized devices into TSMC PDK**
    - Layout view
    - Symbol view
    - Simulation views (hpsiceS, spectre, ads, eldo ….)
    - LVS views (auCdl, auLvs ..)
3. **Design flow for customized devices**
    - Schematic entry
    - Artist simulation
    - VirtuosoXL layout driven flow
    - Assura LVS flow
    - Assura RCX flow
    - Assura Extract view
    - Calibre LVS flow
    - Calibre RCX flow

The limitations of this method are

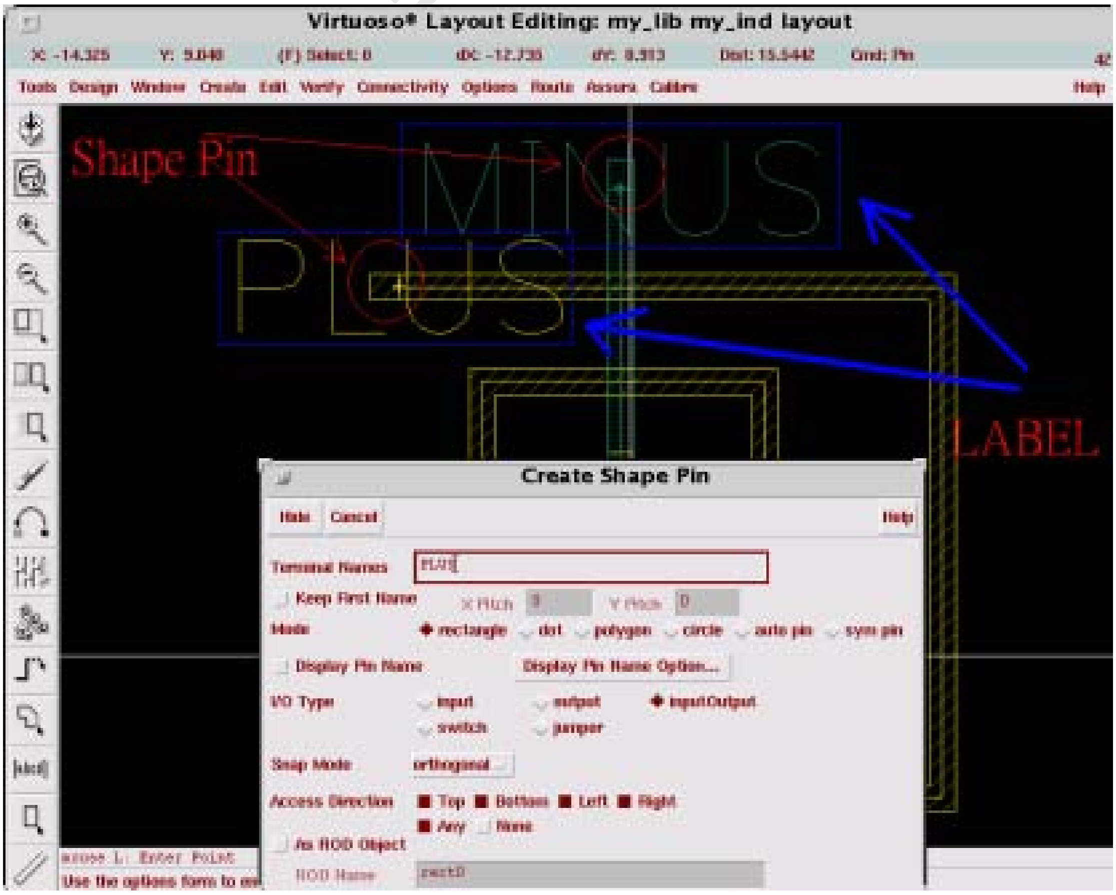
1. Parameterized device is not includes in the document.
2. The black box flow will skip the parasitic RC which over or near the customized cells.

1. Data Preparation

- Layout

After the layouts ready, user needs to stream them into TSMC PDK library by using TSMC virtuoso tech file. And the cell name must be assigned to a corresponding name. There are two important steps need to be added in the layout.

1. Create shape pin, the pin layer number must be the same as input/output layer. To add the pin layer is essential for Virtuoso XL and Assura LVS.
2. Create label pin, the labels are used for Hercules/Calibre/Assura LVS recognized. User needs to follow TSMC layer definition to add corresponding label layer. For example, metal1's label layer is ("METAL1" "pin"). Those names must be all the same as symbol.



Virtuoso® Layout Editing: my_lib my_ind layout

X: -14.325 Y: 5.040 (F) Select: 0 dX: -12.735 dY: 8.313 Dist: 15.5442 Cmd: Pin

mouse L: Enter Point

Use the options form to en

Create Shape Pin

Terminal Names: PLUS

□ Keep First Name X Pitch 0 Y Pitch 0

Mode: ● rectangle ○ dot ○ polygon ○ circle ○ auto pin ○ sync pin

□ Display Pin Name

I/O Type: ○ input ○ output ● inputOutput ○ switch ○ jumper

Snap Mode: orthogonal

Access Direction: ■ Top ■ Bottom ■ Left ■ Right ■ Any □ None

□ As ROD Object

ROD Name: rect0

- Netlist or model

  If the customized cell is a primitive device, it is necessary to prepare corresponding simulation model and LVS sub-circuit. The following is an example for a customized device

  a. model for my_ind cell

  ```
  subckt my_ind PLUS MINUS
  R1 (PLUS MINUS) resistor r=1000
  ends my_ind
  ```

  b. LVS sub-circuit

  ```
  .SUBCKT my_ind PLUS MINUS
  .ENDS
  ```

  NOTE : The cell name, pin names and order of pin name must be all the same at any place.

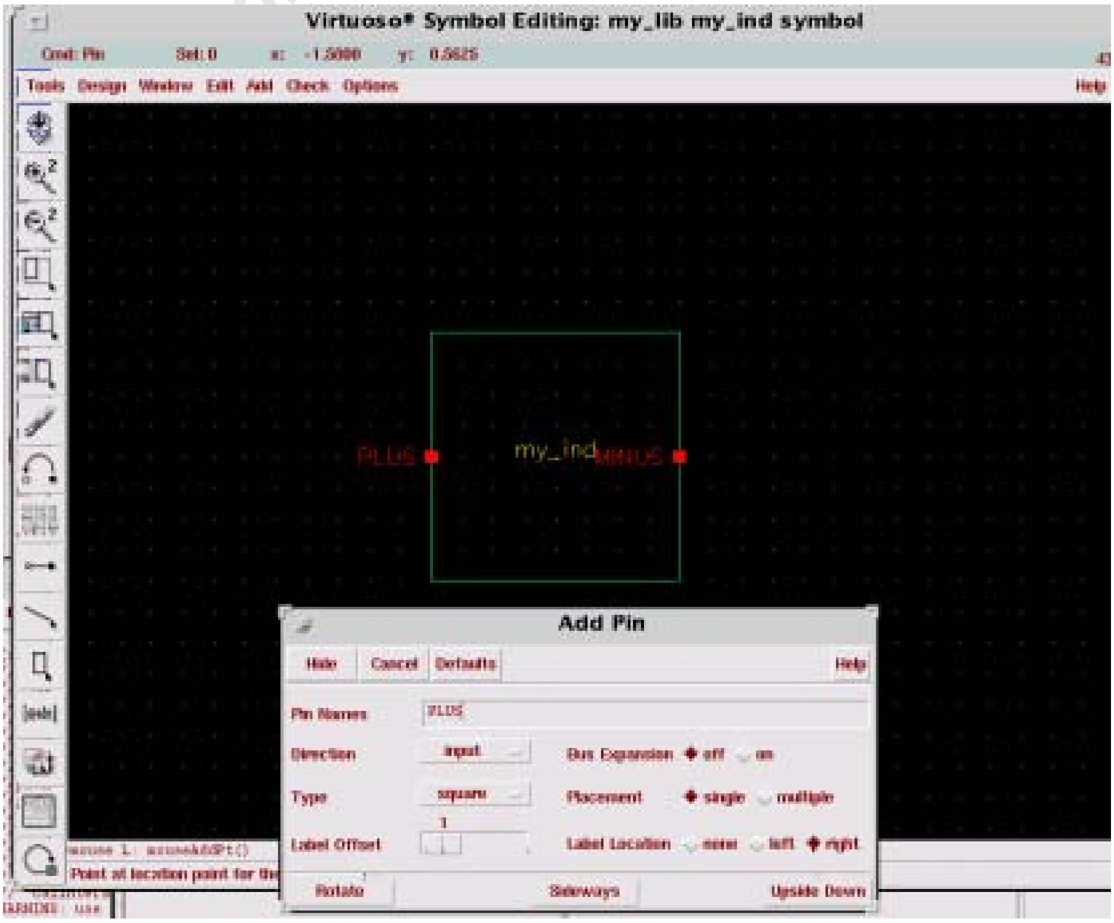
2. Add customized devices into TSMC PDK

- Layout view

  Stream into TSMC PDK library as a cell, then and add shape pin and label. Please refer to layout of data preparation section.

- Symbol view

  Use Virtuoso Symbol Editor to edit a symbol view, the pins must be correctly assigned. And the pin name must be the same with others.



Virtuoso® Symbol Editing: my_lib my_ind symbol

Cmd: Pin Sel: 0 x: -1.5000 y: 0.5625

Add Pin

Pin Names: PLUS

Direction: input

Type: square

Label Offset: 1

Bus Expansion: ● off ○ on

Placement: ● single ○ multiple

Label Location: ○ none ○ left ● right

- Other views (spectre, hspiceS, auLVS, auCdl ….)

  User needs to copy symbol view to other views, then loading a corresponding CDF (component description file). The sample CDF file is attached below. The red ink means they need to be modified case by case.

```
/*****************************************************/
  LIBRARY = "my_lib"
  CELL        = "my_ind"
```

```
/**************************************************/

let( ( libId cellId cdfId )
    unless( cellId = ddGetObj( LIBRARY CELL )
        error( "Could not get cell %s." CELL )
    )
    when( cdfId = cdfGetBaseCellCDF( cellId )
        cdfDeleteCDF( cdfId )
    )
    cdfId    = cdfCreateBaseCellCDF( cellId )

    ;;; Parameters
    cdfCreateParam( cdfId
        ?name               "model"
        ?prompt             "Model name"
        ?defValue           "my_ind"
        ?type               "string"
        ?display            "t"
        ?editable           "nil"
        ?parseAsCEL         "yes"
    )
    cdfCreateParam( cdfId
        ?name               "macro"
        ?prompt             "Hspice S model name"
        ?defValue           "my_ind"
        ?type               "string"
        ?display            "nil"
        ?editable           "nil"
        ?parseAsCEL         "yes"
    )
    cdfCreateParam( cdfId
        ?name               "macroArgumentStyle"
        ?prompt             "macroStyles"
        ?defValue           "hspiceS"
        ?type               "string"
        ?display            "nil"
        ?editable           "nil"
        ?parseAsCEL         "yes"
```

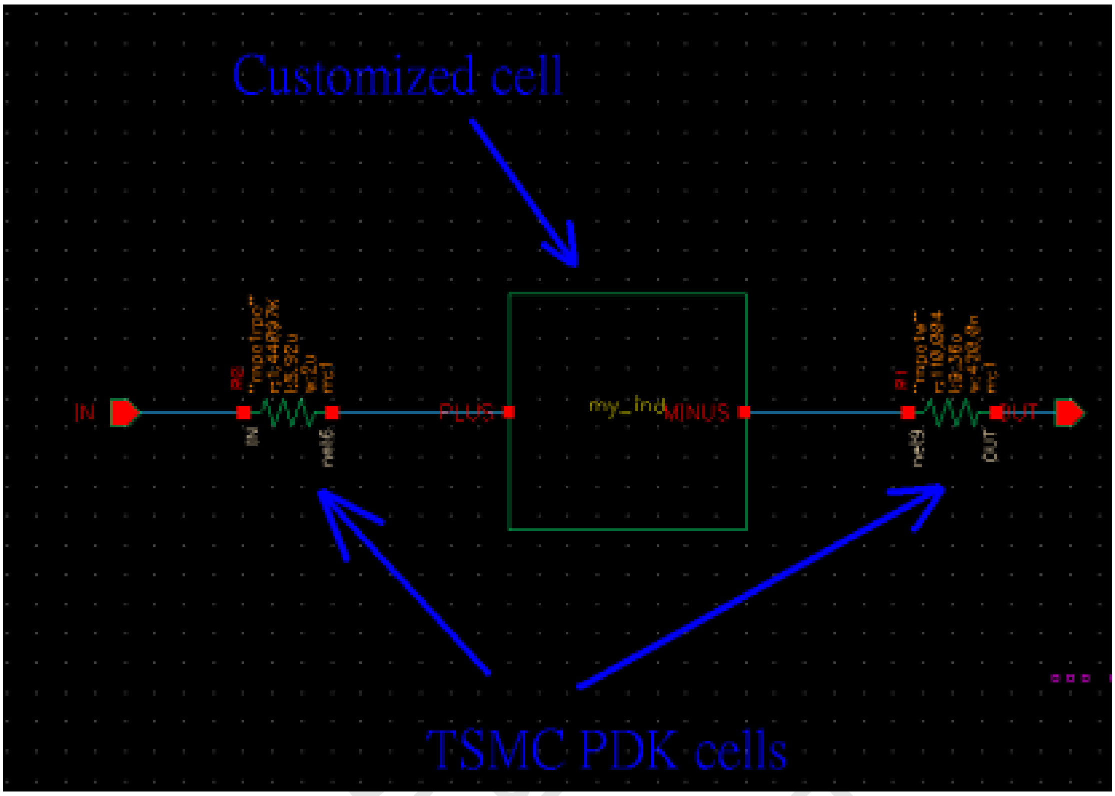
```
  )
;;; Simulator Information
  cdfId->simInfo = list( nil )
  cdfId->simInfo->Cdl = '( nil
       netlistProcedure   tsmcCdlSubcktCall
       componentName          my_ind
       termOrder             (PLUS MINUS)
       namePrefix            "X"
       modelName               "model"
  )
  cdfId->simInfo->auCdl = '( nil
       netlistProcedure   tsmcCdlSubcktCall
       componentName          my_ind
       termOrder             (PLUS MINUS)
       namePrefix            "X"
       modelName               "model"
  )
  cdfId->simInfo->auLvs = '( nil
       netlistProcedure   tsmcCdlSubcktCall
       componentName          my_ind
       termOrder             (PLUS MINUS)
       namePrefix            "X"
       modelName               "model"
  )
  cdfId->simInfo->hspiceS = '( nil
       otherParameters     (macro)
       termOrder             (PLUS MINUS)
       netlistProcedure   ansHspiceSsubcktCall
       componentName          subcircuit
       macroArguments       nil
       namePrefix            "X"
  )
  cdfId->simInfo->spectre = '( nil
       otherParametes       (model)
       termOrder             (PLUS MINUS)
       termMapping            (nil PLUS ":1" MINUS ":2")
  )
```

```
    ;;; Properties
    cdfId->formInitProc                 = ""
    cdfId->doneProc                       = ""
    cdfId->buttonFieldWidth            = 340
    cdfId->fieldHeight                    = 35
    cdfId->fieldWidth                     = 350
    cdfId->promptWidth                    = 175
    cdfId->paramLabelSet                  = "-model n r widthW    "
    cdfId->opPointLabelSet              = "cap    "
    cdfId->modelLabelSet                 = "c    "
    cdfSaveCDF( cdfId )
)
```

3. Design flow

- Schematic Entry

  Just use the customized cells as normal device. Instance customized cells then connect them with other cells in a schematic. The following schematic is a simple example. It will be used as a demo case for explaining design flow.



- Artist simulation (analog design environment)

  Include the customized model when doing simulation. The following spectre and spice netlists are generated from Cadence Analog Enviroment, it shows us the connectivity are all correct.

<u>Spectre netlist</u>

```
….
include "/export/home/jwchen/PDK/project/IP_solution/data/my_ind.scs"

// Library name: test
// Cell name: test_my_ind
// View name: schematic
I1 (net6 net9) my_ind
```

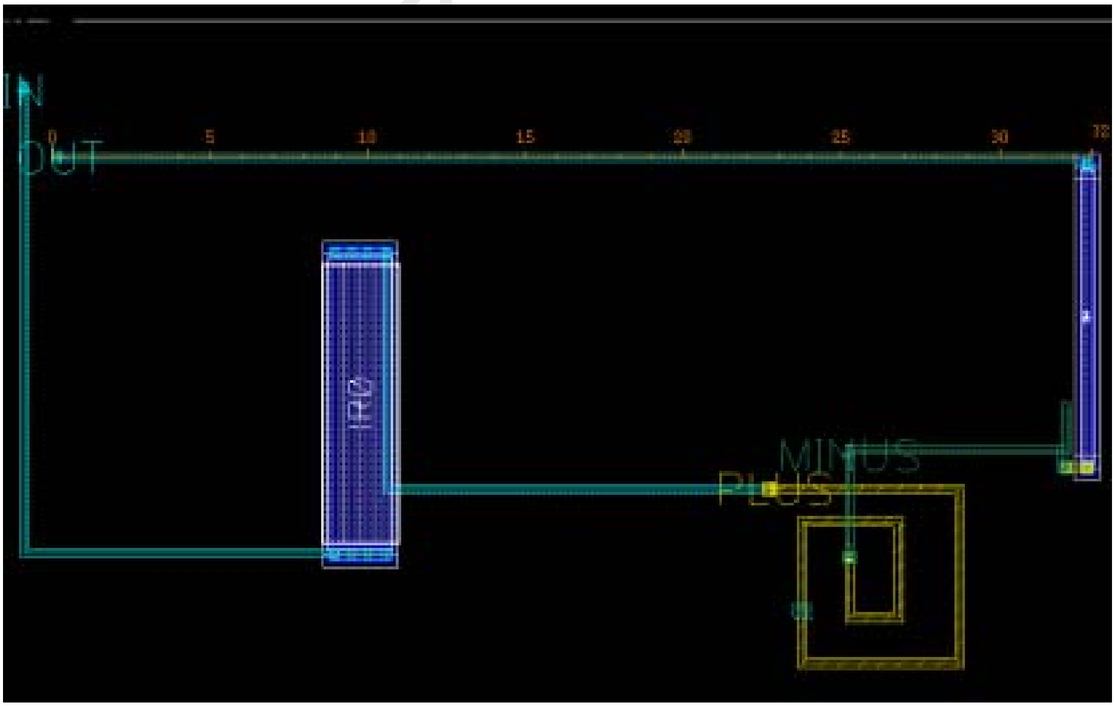
```
R1 (net9 OUT) rnpo1w l=9.36u w=420.0n mf=(1)
R0 (IN net6) rnpo1rpo l=8.92u w=2u mf=(1)
…
```

Spice netlist

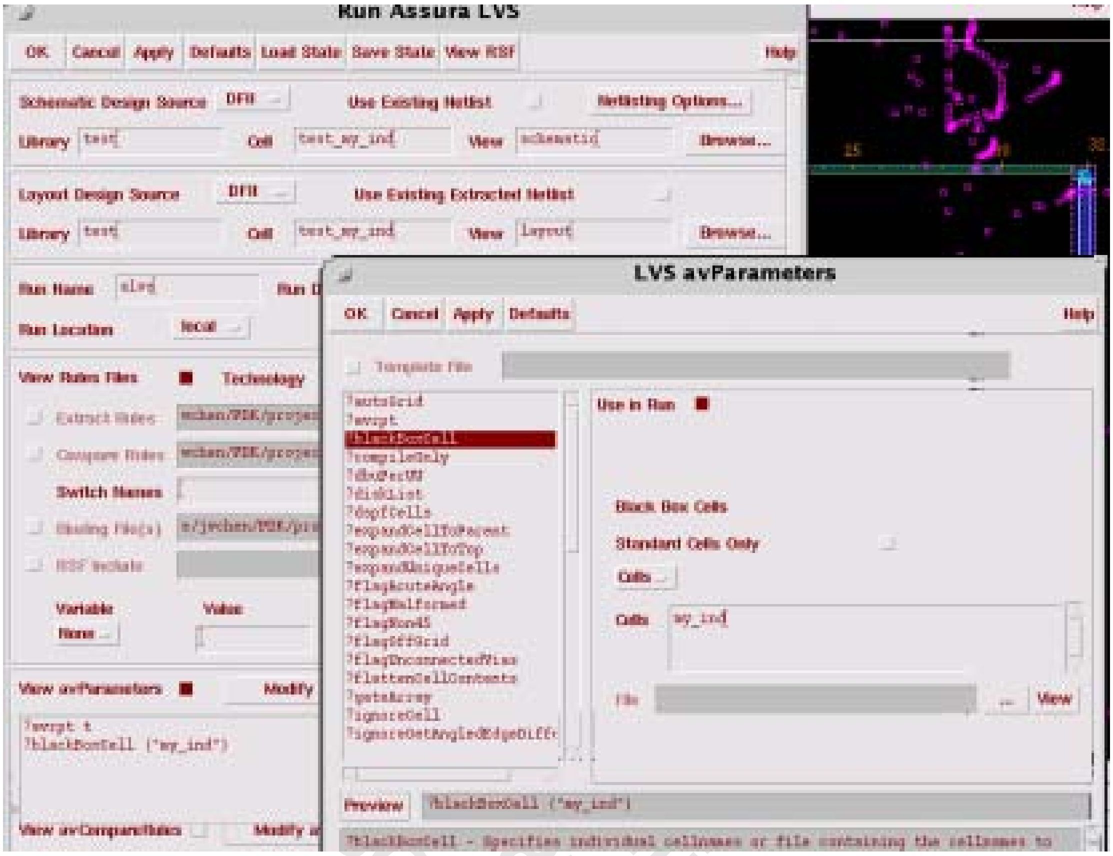
```
XI1 NET6 NET9 MY_IND
XR1 NET9 OUT RNPO1W W=420E-9 L=9.36E-6 MF=1
XR0 IN NET6 RNPO1RPO W=2E-6 L=8.92E-6 MF=1
```

- VirtuosoXL layout driven flow

  The layout driven flow can work well. One important thing needs to take care is, the interconnect must connect to pin shape. The following layout is modified to let interconnect connect to pin shape after auto-router.



- Assura LVS flow for DFII flow and batch mode
  1. Need to assign “blackBoxCell” to let LVS deck know it should not be extracted inside of the customized cell.
  2. If the LVS deck has not defined pinLayer before, users have to modify it. The pinLayer is essential for black box. (Please refer Cadence manual for the detail)



Schematic Design Source DFII
Use Existing Netlist □
Library test Cell test_my_ind View schematic
Layout Design Source DFII
Use Existing Extracted Netlist □
Library test Cell test_my_ind View layout
Run Name alvs
Run Location local
View Rules Files ■ Technology
□ Extract Rules wchen/PDK/project
□ Compare Rules wchen/PDK/project
Switch Names
□ Binding File(s) s/jwchen/PDK/pro
□ RSF Include
Variable None Value
View avParameters ■

```
?avrpt t
?blackBoxCell ("my_ind")
```

View avCompareRules □

□ Template File

```
?autoGrid
?avrpt
?blackBoxCell
?compileOnly
?dbuPerUU
?dirList
?dspfCells
?expandCellToParent
?expandCellToTop
?expandUniqueCells
?flagAcuteAngle
?flagMalformed
?flagNon45
?flagOffGrid
?flagUnconnectedVias
?flattenCellContents
?getsArray
?ignoreCell
?ignoreOutAngledEdgeDiff
```

Use in Run ■
Black Box Cells
Standard Cells Only □
Cells my_ind
File

Preview ?blackBoxCell ("my_ind")

?blackBoxCell - Specifies individual cellnames or file containing the cellnames to

extract.rul

```
layerDefs( "df2"
…
metal2_player    = pinLayer( "METAL2")
metal3_player    = pinLayer( "METAL3")
…
)
layerDefs( "gds2"
…
metal2_player    = pinLayer( 18)
metal3_player    = pinLayer( 28)
…
)
M2=geomOr(M2 metal2_player)
M3=geomOr(M3 metal3_player)
…
```

In batch model, the LVS.rsf need to be modified as following

LVS.rsf

avParameters(

..

?blackBoxCell ("my_ind")

..

?textPriOnly nil

..

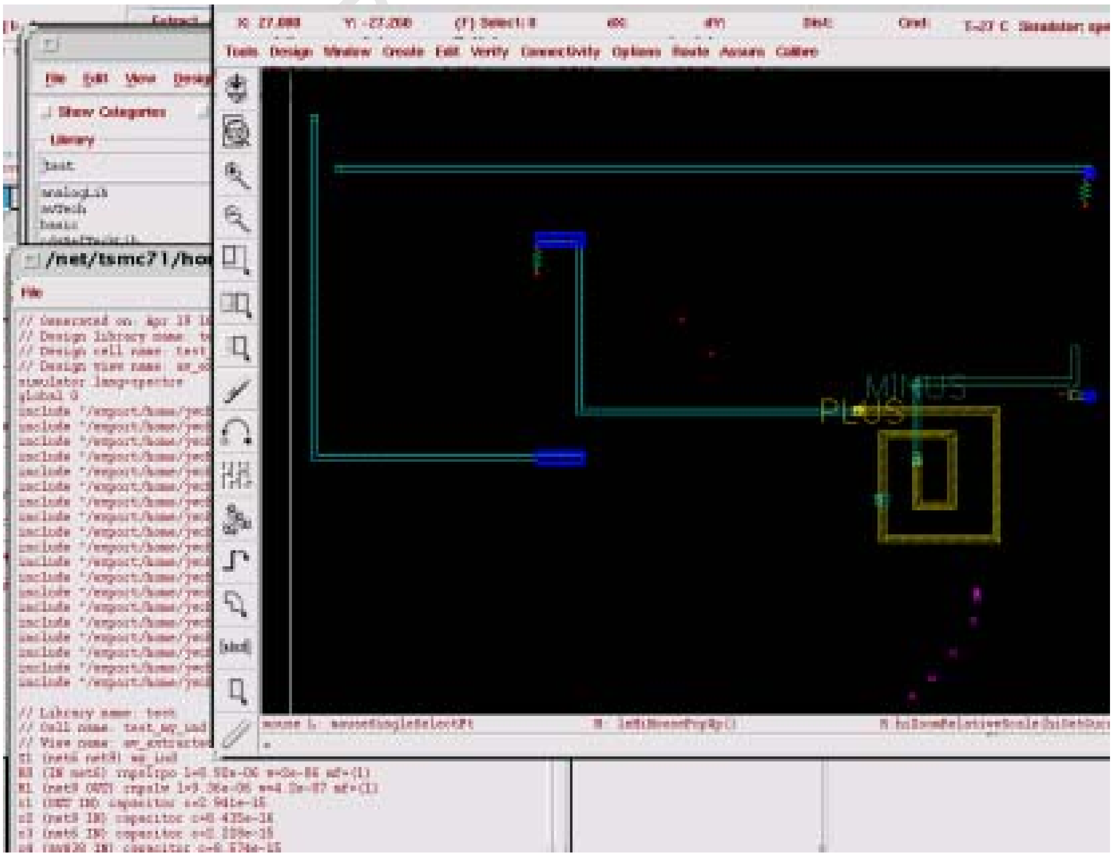
)

- Assura RCX flow

  Use the same setup method as Assura LVS

- Extracted view

  The Assura extracted view will use layout view as symbol. The following picture is extracted view and extracted simulation netlist.



```
// Generated on: Apr 18 1
// Design library name: t
// Design cell name: test
// Design view name: av_e
simulator lang=spectre
global 0
include "/export/home/ye
include "/export/home/ye
include "/export/home/ye
include "/export/home/ye
include "/export/home/ye
include "/export/home/ye
include "/export/home/ye
include "/export/home/ye
include "/export/home/ye
include "/export/home/ye
include "/export/home/ye
include "/export/home/ye
include "/export/home/ye
include "/export/home/ye
include "/export/home/ye
include "/export/home/ye
include "/export/home/ye
include "/export/home/ye
include "/export/home/ye

// Library name: test
// Cell name: test_my_ind
// View name: av_extracted
I1 (net6 net9) my_ind
R3 (IN net6) rnpolypo l=6.92e-06 w=2e-06 mf=(1)
R1 (net9 OUT) rnpolw l=9.36e-06 w=4.2e-07 mf=(1)
c1 (OUT IN) capacitor c=2.981e-15
c2 (net9 IN) capacitor c=0.435e-18
c3 (net6 IN) capacitor c=1.226e-15
c4 (net20 IN) capacitor c=6.574e-15
```

- Calibre LVS flow

  i. Output a CDL netlist then include LVS sub-circuit. The complete LVS is as below “test_my_ind.cdl”.

  ii. Insert a line “LVS BOX cell_name” into LVS deck

  iii. Different pins should not be shorted inside of blackbox, user can use

RMDUMMY to let LVS avoid the pin short problem.

Test_my_ind.cdl

```
.SUBCKT test_my_ind IN OUT
*.PININFO IN:I OUT:I
XI1 net6 net9 my_ind
RR1 net9 OUT 164.914 $[NS]
RR0 IN net6 1.30232K $[NR]
.ENDS

.SUBCKT my_ind PLUS MINUS
.ENDS
```

LVS deck

```
…
LVS BOX my_ind
…
```

- Calibre RCX flow
  - i. Insert a line “LVS BOX cell_name” into RCX deck
  - ii. Prepare PEX X-Cells file
  - iii. Change extract type to gate level extraction

The Calire RCX extracted netlist is as below, customized cell has been extracted as a sub-circuit.

```
.subckt test_my_ind    IN OUT
*
xRR0 IN net6 rnpo1rpo $w=2e-06 $l=8.92e-06
xRR1 net9 OUT rnpo1w $w=4.2e-07 $l=9.36e-06
xI1 net6 net9 MY_IND
c_3 IN 0 2.46362f
c_6 net6 0 2.04346f
c_7 net9 0 0.674213f
c_10 OUT 0 2.94089f
*
```

```
.include test_my_ind.pex.netlist.TEST_MY_IND.pxi
*
```