Laker CDPR Tutorial for Timing-driven Flow

1 Introduction

The Laker Custom Digital solution is for cell based designs in an environment of custom layout automation system. The recommended flow described in this documentation will guide you through the typical flow starting from data preparation, floor planning, power planning, cell placement, and finishing digital routing.

Data Preparation is covered by the following topics:

2 Environment Setup
3 Technology File Preparation
4 Library Preparation
5 Design Preparation

Recommended Flow is covered by the following topics:

6 Floorplan
7 Pre-Placement
8 Power Plan
9 In-Placement
10 Post-Placement
11 In-Route

Recommended Timing Driven Flow is covered by the following topics:

12 Timing Driven Flow

2 Environment Setup

2.1 Tool Installation Version

Laker 2011.06p1 for PG router, digital router, and timing driven Automatic Place and Route (APR).
LakerOA 2011.08p1 for PG router, digital router, and timing driven APR.

2.2 Open Cell Library

The Si2 Nangate Open Cell Library is a generic open-source, standard-cell library provided for the purposes of research, testing, and exploring EDA flows. Therefore, Si2 Nangate [1] PDKv1.3_2009_07 release of the Open Cell Library was selected for the common library preparation.

For more information, refer to the following website page:
http://www.si2.org/openeda.si2.org/projects/nangatelib

Both Laker DB and Laker OA utilize the basic Open Cell Library in data preparation.

The following steps are optional if you only want to install the original package for reference.

1. Download the Open Cell Library and unzip the tar file.
```
> tar zxvf NangateOpenCellLibrary_PDKv1_3_v2009_07.tgz
```

2. You can also create a soft link for central installation.
```
> ln –s <your_install_path>/NangateOpenCellLibrary_PDKv1_3_v2009_07 .
```

2.3 Custom Digital Tutorial

On top of previous work of Open Cell Library database, the following derivatives are created for the Custom Digital tutorial.

source/technology	
OpenCellLibrary.tf	Laker technology file with advanced routing rules
ref_oa.tf	Laker OpenAccess technology file for reference library creation
laker.dsp	Laker display file
OpenCellLibrary.itf	Laker OpenCellLibrary Interconnect Technology File
OpenCellLibrary_x2d.captbl	Laker OpenCellLibrary reference RC Table file
source/library	
OpenCellLibrary.gds	Laker revised GDS file with new cells.
OpenCellLibrary.sp	Laker revised SPICE file with new cells and PG ports.
OpenCellLibrary_RC_ANT.lef	Laker revised LEF file with modified metal1 fat metal rules, antenna rule, antenna data and RC values.
OpenCellLibrary.cp	Laker cell property file for updating Cell Property
OpenCellLibrary_ANT.cp	Laker cell property file of antenna data for updating Cell Property
OpenCellLibrary.idx	Laker generic cell content index file for Row Placement
TAPCELL_X1.gds	Laker generic TAP cell, necessary for tap-less standard cell flow
FILLCELL_X3.gds	Laker generic FILL3 cell, it is necessary for nofiller1 flow
PGMUX2_X1.gds	Laker generic pass-gate MUX2 design for PG ESD spacing flow

[1] The example information was obtained from the Si2 website (http://www.si2.org/openeda.si2.org/projects/nangatelib).

PGMUX2_X1.sp	Laker genetic pass-gate MUX2 design for PG ESD spacing flow

source/openaccess

NCSU_FreePDK_45nm	OpenAccess library for custom via
NangateOpenCellLibrary	OpenAccess library for standard cell layout

source/liberty Laker liberty timing libraries with modified foot print and scan attributes.

NangateOpenCellLibrary_slow_conditional_nldm.lib	worst case timing library
NangateOpenCellLibrary_typical_conditional_nldm.lib	typical case timing library
NangateOpenCellLibrary_fast_conditional_nldm.lib	best case timing library

source/design

Divide.sp	CDL netlist of Divide example
Divide_pl.def	DEF floorplan file
Divide_pl.gds	GDS floorplan file
Divide.v	gate level Verilog netlist of Divide example
CPU.sp	CDL netlist of CPU example
CPU.v	gate level Verilog netlist of CPU example
CPU_eco.v	gate level Verilog netlist of CPU ECO example

source/constraint

pin_bus.const	Laker pin constraint for Auto Pin Assignment in bus format
pin_opt.const	Laker pin optimization constraint for Auto Pin Assignment
placement.const	Laker placement constraint
matrix_rp.tcl	Laker hierarchical matrix constraint Tcl file
CPU_init_pin.const	Initial pin placement constraint
CCU.sdc	Timing constraint file for CCU block
CCU_timing.tcl	Timing Configuration file for CCU block

source/map

lef_layer.map	Laker layer map file for Import LEF
lef_layer_out.map	Laker layer map file for Export LEF
lef2oa.map	Laker OA lef2oa layer map file for Import LEF
gds_layer.map	Laker layer map file for Import Stream
gds_font.map	Laker font size map file for Import Stream
SDL_def.map	Laker model map file for SDL flow
SDL_def_abs.map	Laker model map file for SDL flow with abs view
SDL_def_oa.map	Laker OA model map file for SDL flow
SDL_ref_nangate.map	Laker OA model map file for reference library flow

source/script

route.tcl	sample script for batch procedure of digital routing

The following steps install tutorial source files and a working directory.

1. Unzip the Custom Digital Tutorial source files in your tutorial directory.
   ```
   > tar zxvf Custom_Digital_Tutorial_xxxxyyzz.tgz
   ```

2. Set up a new working directory environment.
   ```
   > copy work_clean work
   >  cd work
   ```

2.4 Enhanced CustomDigital Toolbox

The original CustomDigital Toolbox version installed in the Laker released package provides essential features. An enhanced toolbox of power route and digital route is prepared for a quick start tutorial. It includes a customized menu and frequently used features.

1. Set search path for LakerAPR window because it needs several commands.

   ```
   > set path = ( <laker_install_path>/bin $path)
   > which laker
   > which april
   ```

2. Unzip the enhanced Digital Flow Toolbox in your working directory.

   ```
   > tar zxvf CustomDigital_Enhanced_Toolbox_xxxxyyzz.tgz
   ```

   You can also create a soft link for central installation.

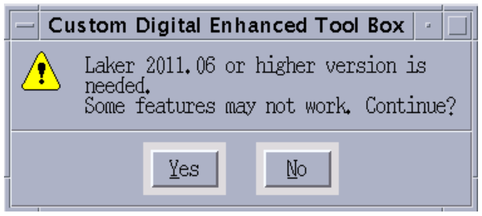
   ```
   > ln –s <your_install_path>/CustomDigital .
   ```

3. Set up a few environment variables.

   ```
   > source CustomDigital/integ.cshrc
   ```

   ```
   # Environment variable settings for integration
   setenv DIGITALROUTER_INTEG_PATH .
   setenv DIGITALROUTER_INTEG_VERSION_CHECK 1

   unsetenv LAKER_TCL_DIGITAL_ROUTEALL
   ```

   The `DIGITALROUTER_INTEG_VERSION_CHECK` environment variable is enabled to check the Enhanced CustomDigital ToolBox compatibility with the Laker tool version. This check can be disabled by removing this environment variable.

   
   Custom Digital Enhanced Tool Box

   Laker 2011.06 or higher version is needed.
   Some features may not work. Continue?

   The `LAKER_TCL_DIGITAL_ROUTEALL` environment variable is used to invoke the basic CustomDigital menu of **PG Route, Route** and **Delete Route** packed in the released package. It is disabled by removing the environment variable.

   The **PG Route, Route** and **Delete Route** forms from release package are reused in the enhanced Digital Flow.

4. Add the following lines in your *laker.rc* resource file to automatically enable this tool box in the Laker Design Window.

   ```
   [SourceTcl]
   Source1 = $env(DIGITALROUTER_INTEG_PATH)/CustomDigital/integ.tcl
   ```

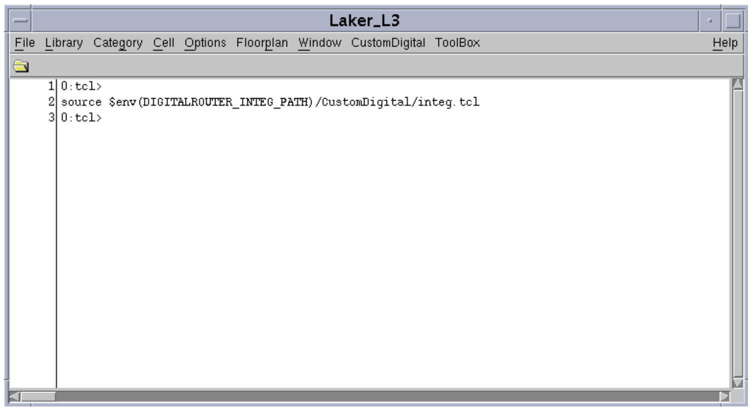
5. Create an *april.rc* resource file is used for timing-driven window.

   ```
     [SourceTcl]
     Source1 = $env(DIGITALROUTER_INTEG_PATH)/CustomDigital/apr/flow.tcl
   ```

```
Source2 = $env(DIGITALROUTER_INTEG_PATH)/CustomDigital/apr/integ.tcl
```

Contact your support AE for feature feedback and suggestions about this enhanced toolbox. Useful scripts will be integrated in later versions.

6. Invoke the Laker system and check for a successful message in the *Main* window.


Laker_L3
File Library Category Cell Options Floorplan Window CustomDigital ToolBox Help

```
0:tcl>
source $env(DIGITALROUTER_INTEG_PATH)/CustomDigital/integ.tcl
0:tcl>
```

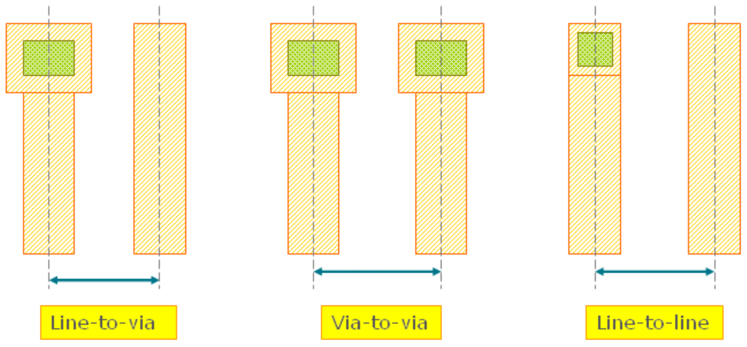
3 Technology File Preparation

There are two main parts of routing technology. One is routing rules, including layer, pitch, offset, default width, default space, preferred track direction and cost. The other is foundry DRC rules for metal and cut layers in a higher abstract description.

How does a library developer define a routing rule before creation of a standard cell library?

- Preferred direction depends on a standard layout style. Both HVH and VHV style are popular in advanced process nodes.
- Minimum routing width might be larger than minimum width defined in a Design Rule Manual (DRM). For example, larger routing width is used to cover the whole via closure.
- Pitch value is decided by minimum wire-wire spacing or minimum wire-via spacing. For example:
  - Horizontal metal1 pitch is wire-wire spacing style.
    Pitch value = 0.5 * routing width + routing spacing + 0.5 * routing width

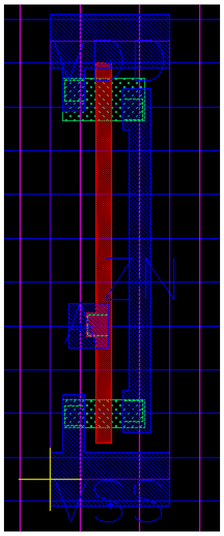
  - Vertical metal2 pitch is wire-via spacing style.
    Pitch value = 0.5 * routing width + routing spacing + 0.5 * via enclosure width



- A larger value of via enclosure width can be used to guarantee easy routing because no via rotation is necessary.

- Vertical pitch offset value is usually half of vertical pitch. This offset value matches pin offset of a standard cell layout. Otherwise, a short problem occurs when two standard cells are abutted.

- A common standard cell is 7-track, 9-track or 12-track of horizontal track (wire-wire spacing). The smallest driving inverter is 2-track of vertical track (wire-spacing).



The figure illustrates OpenCellLibrary INV_X1 routing pitch and offset. It is a 10-track height cell.

You can also get reference information from OA technology database, a LEF file or a third party technology file if it is available.

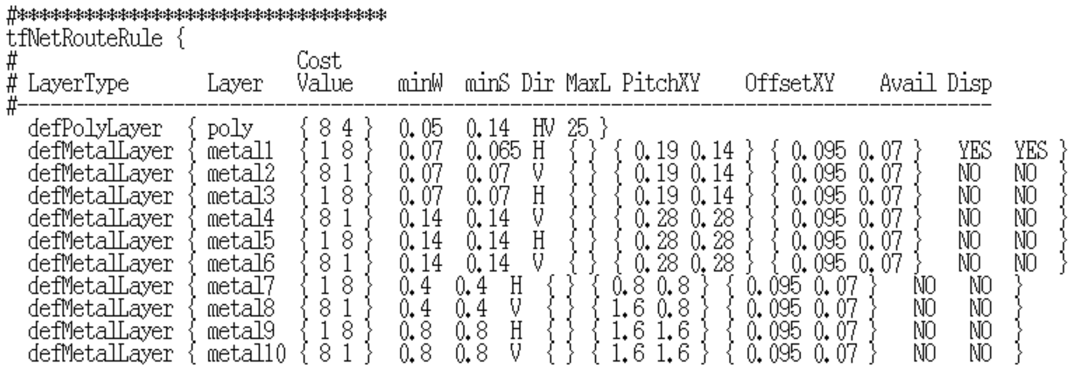
3.1 Lab-1A: Routing rules

Name:	Specify a valid routing layer. Poly layer can be defined if poly routing.
Cost XY:	Specify horizontal cost and vertical cost for a routing layer.
Width:	Specify the minimum width for a routing layer.
Spacing:	Specify the minimum spacing for a routing layer.
Pitch XY:	Specify the pitch or track value.
Track Direction	Specify available track direction.
Offset XY:	Specify the offset value of a pitch.

How does a library developer define a routing rule?

- Cost XY values need to be matched with track direction. For example, ratio of {1 8} for metal1 layer means horizontal cost is smaller. It results in X-preferred direction. A ratio of {8 1} for metal2 layer results in Y-preferred direction.

The *tfNetRouteRule* section defines routing rules with the conditions of routing tracks, and also defines other routing constraints and spacing rules. For detailed information, please refer to the *Laker Command Reference* document.



```
#*********************************
tfNetRouteRule {
#                          Cost
# LayerType        Layer   Value    minW  minS Dir MaxL PitchXY     OffsetXY     Avail Disp
#-----------------------------------------------------------------------------------------
  defPolyLayer   { poly    { 8 4 }  0.05  0.14  HV 25 }
  defMetalLayer  { metal1  { 1 8 }  0.07  0.065 H  { } { 0.19 0.14 } { 0.095 0.07 }  YES  YES }
  defMetalLayer  { metal2  { 8 1 }  0.07  0.07  V  { } { 0.19 0.14 } { 0.095 0.07 }  NO   NO  }
  defMetalLayer  { metal3  { 1 8 }  0.07  0.07  H  { } { 0.19 0.14 } { 0.095 0.07 }  NO   NO  }
  defMetalLayer  { metal4  { 8 1 }  0.14  0.14  V  { } { 0.28 0.28 } { 0.095 0.07 }  NO   NO  }
  defMetalLayer  { metal5  { 1 8 }  0.14  0.14  H  { } { 0.28 0.28 } { 0.095 0.07 }  NO   NO  }
  defMetalLayer  { metal6  { 8 1 }  0.14  0.14  V  { } { 0.28 0.28 } { 0.095 0.07 }  NO   NO  }
  defMetalLayer  { metal7  { 1 8 }  0.4  0.4  H  { } { 0.8 0.8 } { 0.095 0.07 }  NO   NO  }
  defMetalLayer  { metal8  { 8 1 }  0.4  0.4  V  { } { 1.6 0.8 } { 0.095 0.07 }  NO   NO  }
  defMetalLayer  { metal9  { 1 8 }  0.8  0.8  H  { } { 1.6 1.6 } { 0.095 0.07 }  NO   NO  }
  defMetalLayer  { metal10 { 8 1 }  0.8  0.8  V  { } { 1.6 1.6 } { 0.095 0.07 }  NO   NO  }
```

3.2 Lab-1B: Foundry DRC rules

Advanced process has new rules for width, space, fat metal space, end of line, enclosure edge, min area, min enclosure, min edge, min cut, max stack via, etc.

A technology LEF file is another high level description for necessary DRC information for routing. The following table is a brief example for one-to-one syntax format mapping between tfNetRouteRule and LEF.

tfNetRouteRule syntax format	***LEF syntax format***

`defRouteRule{`
`    layerName { M1 }`
`    endofLine {`
`        { Spacing 0.1 }`
`        { Threshold 0.1 }`
`        { WithIn 0.04 }`
`        { ParallelEdge 0.1 }`
`        { ParallelWithIn 0.1 }`
`        { MinLength 0.07 }`
`    }`
`    endofLine {`
`        { Spacing 0.12 }`
`        { Threshold 0.1 }`
`        { WithIn 0.04 }`
`        { ParallelEdge 0.1 }`
`        { ParallelWithIn 0.1 }`
`        { MinLength 0.07 }`
`        { BelowEncloseCut 0.05 }`
`        { CutSpacing 0.15 }`
`    }`
`}`
	`SPACING 0.10 ENDOFLINE 0.10 WITHIN 0.04`
	`PARALLELEDGE 0.10 WITHIN 0.10 MINLENGTH 0.07`

	`SPACING 0.12 ENDOFLINE 0.10`
	`WITHIN 0.04 PARALLELEDGE 0.1 WITHIN 0.10 MINLENGTH`
	`0.07 ENCLOSECUT BELOW 0.05 CUTSPACING 0.15 ;`

`defViaRouteRule{`
`    viaName { VIA1 }`
`    minCutRule {`
`        { MetalWidth 0.3 }`
`        { {CutNum 2} {WithIn 0.2 } }`
`        { {CutNum 4} {WithIn 0.25} }`
`    }`
`    minCutRule {`
`        { MetalWidth 0.3 }`
`        { { MetalLength 0.3 }`
	`MINIMUMCUT 2 WIDTH 0.3 WITHIN 0.2`
	`MINIMUMCUT 4 WIDTH 0.3 WITHIN 0.25`

tfNetRouteRule syntax format	***LEF syntax format***

{ MetalWithIn 0.8 } }
{ { CutNum 2 } }
}
}
	MINIMUMCUT 2 WIDTH 0.3 LENGTH 0.3 WITHIN 0.8

defMaxViaStackRule {
{maxViaStack 4}
{range { M1 M6 }}
}
	MAXVIASTACK 4 RANGE M1 M6;

defRouteRule {
layerName {M1}
minArea {
{ Area 0.027 }
}
minArea {
{ Area 0.06 }
{ ExceptEdgeLength 0.21 }
{ ExceptMinSize 0.07 0.21 }
}
minEnclosedArea { 0.2 }
}
	AREA 0.027
	
	AREA 0.06 EXCEPTEDGELENGTH 0.21 EXCEPTMINSIZE 0.07 0.21;
	
	MINENCLOSEDAREA 0.20

defRouteRule {
minEdges {
{ MinEdgeLength 0.1 }
{ MaxEdges 1 }
}
	MINSTEP 0.1 MAXEDGES 1

defViaRouteRule{
viaName { VIA1 }
cutSpacing { 0.1 }
cutSpacing { 0.13 ParallelOverlap}
adjCutSpacing {
{ Spacing 0.13 }
{ AdjacentCuts 3 }
{ WithIn 0.14 }
}
}
	SPACING 0.1
	SPACING 0.13 PARALLELOVERLAP
	
	SPACING 0.13 ADJCENTCUTS 3 WITHIN 0.14 [EXCEPTSAMEPGNET]

spacingRule { VIA1 VIA1 0.1 { SameNet Stack } }	LEF5.6: SPACING SAMENET VIA1 VIA2 0 STACK ; SAMENET VIA1 VIA1 0.1 ; END SPACING LEF5.7: SPACING 0 LAYER VIA1 STACK ;

tfNetRouteRule syntax format	*LEF syntax format*

`defViaRouteRule{`
`    viaName { VIA1 }`
`    enclosure {`
`        {Below}`
`        {EnclosureEdge 0.015}`
`        {MetalWidth 0.11}`
`        {ParallelLength 0.27}`
`        {ParallelWithIn 0.08}`
`        {ExceptExtraCut}`
`    }`
`}`
	`ENCLOSUREEDGE BELOW 0.015 WIDTH 0.11 PARALLEL 0.27`
	`WITHIN 0.08 EXCEPTEXTRACUT`

`space   { M1      0.07    {`
`        { 0.08  { parallel > 0.30 }`
`{ width > 0.17 } }`
`        { 0.12  { parallel > 0.30 }`
`{ width > 0.24 } }`
`        { 0.14  { parallel > 0.40 }`
`{ width > 0.31 } }`
`        { 0.21  { parallel > 0.62 }`
`{ width > 0.62 } }`
`        { 0.5   { parallel > 1.50 }`
`{ width > 1.50 } }`
`      } }`
`antennaRule {`
`  { AntennaCumSideDiffAreaRatio`
`    { { 0 200 } { 0.010 200 }`
`    { 0.015 500 } { 0.110 550 }`
`    { 1.010 1000 } } }`
`  { Thickness 0.13 }`
`}`
`antennaRule {`
`  { AntennaDiffAreaRatio {`
`   { 0 20 } { 0.010 20 }`
`   { 0.015 99999 }`
`   { 0.110 99999 } } }`
`  { AntennaCumDiffAreaRatio {`
`   { 0 40 } { 0.010 40 }`
`   { 0.015 211 } { 0.110 215 }`
`   { 1.010 250 } } }`
`}`
	`PARALLELRUNLENGTH   0.00   0.30   0.40   0.62  1.50`
	`WIDTH    0.00     0.07   0.07   0.07   0.07   0.07`
	`WIDTH    0.17     0.07   0.08   0.08   0.08   0.08`
	`WIDTH    0.24     0.07   0.12   0.12   0.12   0.12`
	`WIDTH    0.31     0.07   0.12   0.14   0.14   0.14`
	`WIDTH    0.62     0.07   0.12   0.14   0.21   0.21`
	`WIDTH    1.50     0.07   0.12   0.14   0.21   0.50 ;`
	
	`ANTENNACUMDIFFSIDEAREARATIO PWL ( ( 0 200 )`
	`  ( 0.01 200 ) ( 0.015 500 )( 0.11 550 )`
	`  ( 1.01 1000 ) ) ;`
	`THICKNESS 0.13 ;`
	
	`ANTENNADIFFAREARATIO PWL ( ( 0 20 ) ( 0.01 20 )`
	`  ( 0.015 99999 )( 0.11 99999 ) ) ;`
	` ANTENNACUMDIFFAREARATIO PWL ( ( 0 40 ) ( 0.01 40 )`
	` ( 0.015 211 ) ( 0.11 215 )( 1.01 250 ) ) ;`

Regarding technology file preparation for poly routing, please refer to the *Laker Command Reference* document.

4 Library Preparation

4.1 Standard Cell Design Kit

The standard cell design kit is a set of deliverables for the timing driven design flow. Usually, it is well defined and prepared by library vendors or foundry design service teams.

The following is a list of common deliverables for reference.

Name	*Purpose*	*Comments*
Technology LEF (.lef)	1. Define routing pitch, direction, DRC/Antenna check rules for layer and via 2. Define standard via and custom via	

Macro LEF (.lef)	1. Define cell size, pin, blockage, etc. 2. Define antenna gate and diffusion area info	
Timing Liberty (.lib)	1. Timing library for logic synthesis and timing analysis.	Supports Non-Linear Delay Model (NLDM)
Verilog library (.v)	1. Simulation model for post-layout SDF back-annotation simulation.	
RC Technology file (.itf)	1. Define process profile with layer thickness, height, width, spacing, dielectric, etc. 2. Define resistance by layer and via	Supports Interconnect Technology File (ITF)

Usually, the standard cell layout view is fixed with multiple modes of timing characterization in each official release. To get better performance and interoperability, it is recommended to use technology LEF and macro LEF files for a quick start of timing driven CDPR.

The abstract layout view can be replaced with the full layer layout view in the Laker system before stream out for final layout merge.

4.2 Lab-2A: Library Preparation by GDS

In this lab, you will learn how to create a GDS library, assign net/port information, define a site, and update cell property files.

4.2.1 Import Stream files

1. Open the *Import Stream* form by invoking **File → Import → Stream**.
2. Select Open Cell Library GDS file named OpenCellLibrary.gds.
3. Assign Library name to OpenCellLibrary.
   Import Stream → Library Name: OpenCellLibrary

4. Assign Technology file to OpenCellLibrary.tf
   Import Stream → Technology → ASCII File: OpenCellLibrary.tf

5. Assign a Layer Map File to gds_layer.map
   Import Stream → Basic → Layer Map File: gds_layer.map

   In general, a standard cell library has one dedicated PR boundary layer for placement abutment. Usually, this layer is smaller than the data extension bounding box (BBOX) to share power and ground rails. The special layer 235, data type 0 is defined as PR boundary of the Open Cell library. Therefore, add one layer mapping rule in the layer mapping file.

```
[gds_layer.map]
CellBdry boundary 235 0
```

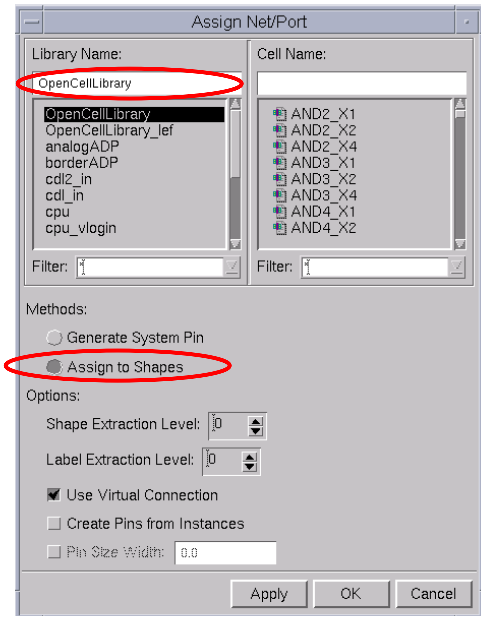
6. Assign a Font Map File to gds_font.map
   Import Stream → Basic → Font Map File: gds_font.map

7. Click **OK** and finish importing geometry data of the library GDS file.
8. Open a cell of NAND2_X1 to display imported layout, text and CellBdry layer.

4.2.2 Net/Port Extraction

Based on layout geometry and annotated text, connectivity information can be added by assigning net and port information on cell layout view (not abstract view). The related rules are defined in the *tfAbstractCell* section.

1. Invoke the **Library → Assign Net/Port** command for library level net/port assignment.
2. Select a library named as OpenCellLibrary. Do not select any cell for library level net/port assignment.
3. In the *Assign Net/Port* form, change **Methods** to **Assign to Shapes** and click **OK**.


Assign Net/Port

Library Name: OpenCellLibrary

OpenCellLibrary
OpenCellLibrary_lef
analogADP
borderADP
cdl2_in
cdl_in
cpu
cpu_vlogin

Cell Name:

AND2_X1
AND2_X2
AND2_X4
AND3_X1
AND3_X2
AND3_X4
AND4_X1
AND4_X2

Filter:

Methods:
Generate System Pin
[✓ Assign to Shapes]

Options:
Shape Extraction Level: 0
Label Extraction Level: 0
[✓] Use Virtual Connection
[ ] Create Pins from Instances
[ ] Pin Size Width: 0.0

The following section is an example of *tfAbstractCell* for OpenCellLibrary.

```
tfAbstractCell {
    #-----------------------------------------------------------------
    # define cell pin extraction rule
    #-----------------------------------------------------------------
    #mapText2Pin { { textLayerName } { ExtractedLayer1 ...} $ExtractAll }
    mapText2Pin { { poly drawing } { poly drawing } }
    mapText2Pin { { metal1 drawing } { metal1 drawing } }
    mapText2Pin { { metal2 drawing } { metal2 drawing } }
    mapText2Pin { { metal3 drawing } { metal3 drawing } }
    mapText2Pin { { metal4 drawing } { metal4 drawing } }
    mapText2Pin { { metal5 drawing } { metal5 drawing } }
    mapText2Pin { { metal6 drawing } { metal6 drawing } }
    mapText2Pin { { metal7 drawing } { metal7 drawing } }
    mapText2Pin { { metal8 drawing } { metal8 drawing } }
    mapText2Pin { { metal9 drawing } { metal9 drawing } }
    mapText2Pin { { metal10 drawing } { metal10 drawing } }

    defPower { vdd VDD pwr PWR vcc VCC }
    defGround { vss VSS gnd GND }

    #-----------------------------------------------------------------
    # define rules for extracting blockage for routing layers
    #-----------------------------------------------------------------
    # genBlockage { { SourceLayerName1 ...} $fill-value { BlockageLayer } }
    #      SourceLayerName1 ... only assign LayerName, because the
    #      blockage generation # need not only "drawing" but also "pin"
    # genBlockage { { MT2VDD  MT2VSS }    0.505    { MT2 blockage } }
    # please use 0 for full shape model
    genBlockage { metal1 0 { metal1 blockage } }
    genBlockage { metal2 0 { metal2 blockage } }
    genBlockage { metal3 0 { metal3 blockage } }
    genBlockage { metal4 0 { metal4 blockage } }
    genBlockage { metal5 0 { metal5 blockage } }
    genBlockage { metal6 0 { metal6 blockage } }
    genBlockage { metal7 0 { metal7 blockage } }
    genBlockage { metal8 0 { metal8 blockage } }
    genBlockage { metal9 0 { metal9 blockage } }
    genBlockage { metal10 0 { metal9 blockage } }

    #-----------------------------------------------------------------
    # define rules for extracting boundary for routing layers
    #-----------------------------------------------------------------
    # genBoundary { cellBBox offset };# 1.cellBBox
    #                                   2.cellBoundary
    #                                   3.pinBBox
    #                                   4.pinBoundary
    genBoundary { cellBoundary 0 };# the same size as routing layers
}
```

4.2.3 Define Global Power/Ground Net

The Global Power/Ground (PG) net has an impact on several features like Flight Line, PG Router, etc. There are several ways to define the global PG net.

1. Add *defPower* and *defGound* definitions in the *tfAbstractCell* section of a technology file.

   ```
   defPower { vdd VDD pwr PWR vcc VCC }
   defGround { vss VSS gnd GND }
   ```

   Invoke **Library → Technology File → Replace** to update a new technology file for an existing library.

2. Add the *GLOBAL_NET* section in a model map file used in a SDL flow.

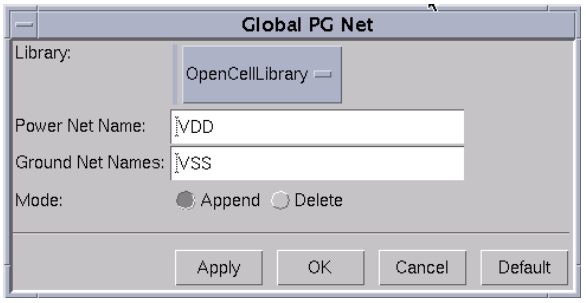
   ```
   [GLOBAL_NET]
     VDD  P
     VSS  G
   ```

   Invoke **Library → Replace Model Map File** to update a new model map file for an existing library.

3. Use the `lakerDefGlobalNet` Tcl command to append, query and remove global nets.

   ```
   lakerDefGlobalNet -lib myLib -power VDD -ground VSS
   lakerDefGlobalNet -lib myLib -remove -power VDD -ground VSS
   ```

   You can invoke a customized script **CustomDigital → PG** in the *Main* window to define simple global power and ground nets as shown above for a library.



Library: OpenCellLibrary

Power Net Name: VDD

Ground Net Names: VSS

Mode: [✓ Append] Delete

4.2.4 Antenna Pin Information

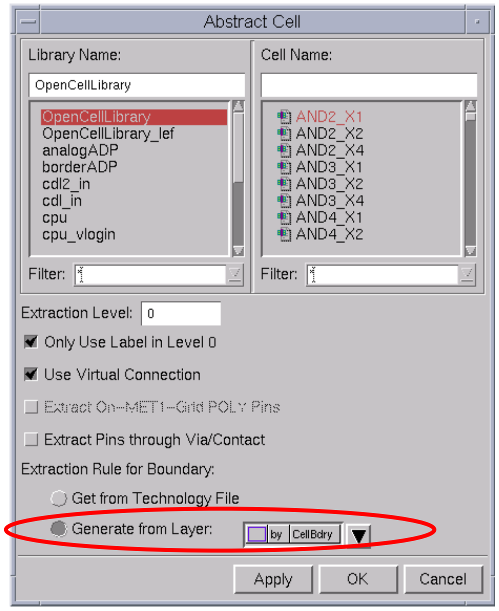
Library preparation for antenna rule check has two parts. One part is defining the antenna rule in the *tfNetRouteRule* section of a Laker technology file. The other part is extracting the antenna gate area and diffusion area from the physical layout view. This step is also optional for third party tool by Export LEF if necessary.

The following section is an example of the *tfAbstractCell* for OpenCellLibrary.

```
tfAbstractCell {

    # define pin/port/blockage extraction over here

    #------------------------------------------------------------------
    # define rules for extracting boundary for routing layers
    #------------------------------------------------------------------
    # genBoundary { cellBBox offset };# 1.cellBBox
    #                                   2.cellBoundary
    #                                   3.pinBBox
    #                                   4.pinBoundary
    genBoundary { cellBoundary 0 };# the same size as routing layers

    genPinAntennaInfo {
    defPolyLayer { poly }
    defODLayer { active }
    defMET1Layer { metal1 }
    defODContLayer { contact }
    }
}
```

Perform abstract view generation to extract antenna gate area, antenna diffusion area, boundary, and pin information.

1. Invoke the **Library → Abstract Cell** command.
2. In the *Abstract Cell* form, enable the **Generate from Layer** option and select [v] **CellBdry** under the **Extract Rule for Boundary** section.


Abstract Cell

Library Name: OpenCellLibrary
OpenCellLibrary
OpenCellLibrary_lef
analogADP
borderADP
cdl2_in
cdl_in
cpu
cpu_vlogin
Filter: *

Cell Name:
AND2_X1
AND2_X2
AND2_X4
AND3_X1
AND3_X2
AND3_X4
AND4_X1
AND4_X2
Filter: *

Extraction Level: 0
☑ Only Use Label in Level 0
☑ Use Virtual Connection
☐ Extract On-MET1-Grid POLY Pins
☐ Extract Pins through Via/Contact
Extraction Rule for Boundary:
○ Get from Technology File
● Generate from Layer: by CellBdry

Apply OK Cancel

4.2.5 Define Site information

Library level SITE information is strongly recommended for SITE related applications. For example, row placer, row snapping, cell legalization, filter insertion, etc.

How does a library developer to define a SITE for the Laker system in GDS flow?

- Refer to a SITE definition in a library technology LEF file if applicable. For example:

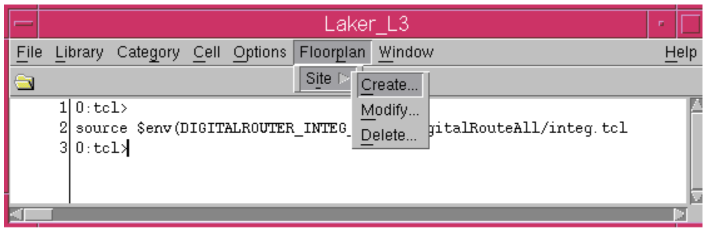
```
SITE NCSU_FreePDK_45nm
  SYMMETRY y ;
  CLASS core ;
  SIZE 0.19 BY 1.4 ;
END NCSU_FreePDK_45nm
```

- Refer to the smallest inverter, e.g. "INV_X1", of a vendor-provided standard library.
  Width of (SITE) = 0.5 * Width of (INV_X1)
  Height of (SITE) = Height of (INV_X1)

```
MACRO INV_X1
  CLASS core ;
  FOREIGN INV_X1 0.0 0.0 ;
  ORIGIN 0 0 ;
  SYMMETRY X Y ;
  SITE NCSU_FreePDK_45nm ;
  SIZE 0.38 BY 1.4 ;
```

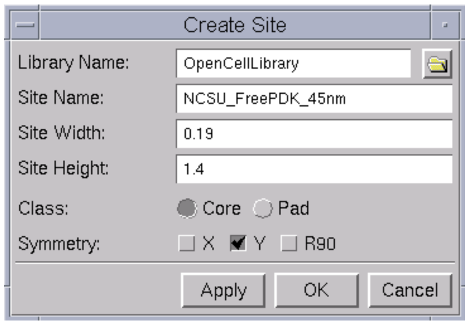
- Use minimum dbResolution for custom cells which have no strict vertical track planning.

1. Invoke the **Floorplan → Site → Create** command in the *Main* window.



```
0:tcl>
source $env(DIGITALROUTER_INTEG_...gitalRouteAll/integ.tcl
0:tcl>
```

2. Define site information according to SITE declared in the OpenCellLibrary.lef
   Site symmetry (usually Y only) is different from cell symmetry. Check Y symmetry carefully.

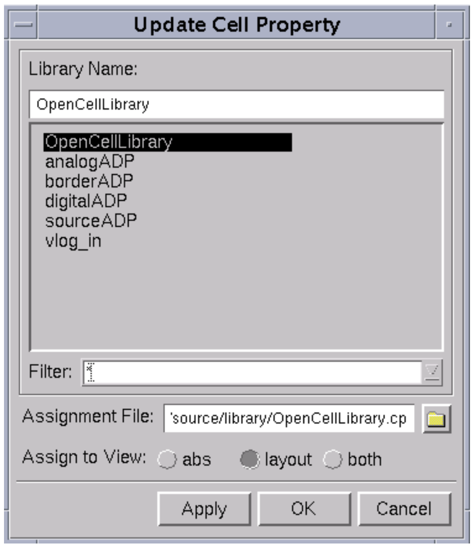


Library Name:	OpenCellLibrary
Site Name:	NCSU_FreePDK_45nm
Site Width:	0.19
Site Height:	1.4
Class:	[✓ Core] Pad
Symmetry:	X [✓ Y] R90

4.2.6 Update Macro Cell Property

An additional Macro cell property defined in a LEF file can be equivalently set by a macro cell property file for GDS In flow.

1. Invoke the **Library → Update Cell Property** command in the *Main* window for library level update.



Library Name: OpenCellLibrary

OpenCellLibrary
analogADP
borderADP
digitalADP
sourceADP
vlog_in

Filter: *

Assignment File: 'source/library/OpenCellLibrary.cp

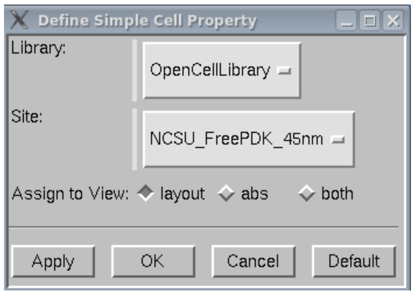
Assign to View: abs [✓ layout] both

2. In the *Update Cell Property* form, do the following:
   a. Select **Library Name** as OpenCellLibrary
   b. Set **Assignment File** to OpenCellLibrary.cp
   c. Check **Assign to View**: to **layout** for GDS flow only
   d. Click **OK** to finish library level cell property update.

Here is an example of all supported properties for reference. Refer to reference manuals for more detailed information.

```
defCell AND2_X1 {
  defCellSite NCSU_FreePDK_45nm
  defClass "CORE"
  defCellSymmetry "X Y"
}
```

You can invoke a customized script **CustomDigital → Simple Cell Property** in the *Main* window to define simple default site, X and Y symmetry shown as above for a standard cell library.


Define Simple Cell Property

Library: OpenCellLibrary

Site: NCSU_FreePDK_45nm

Assign to View: layout abs both

4.3 Lab-2B: Library Preparation by LEF

LEF library preparation is only necessary for pure LEF/DEF flows. In this lab, you will learn how to create a LEF library in abs (abstract) view.

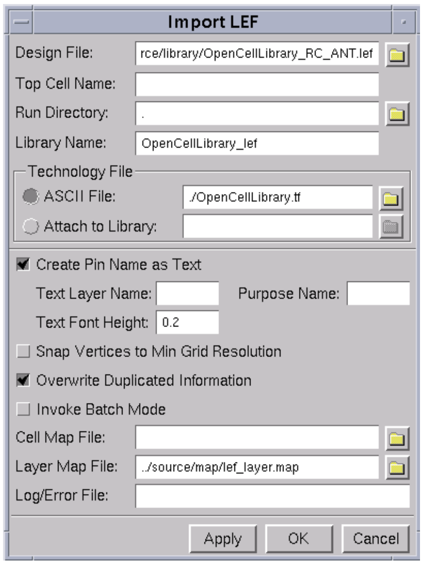
4.3.1 Import LEF Files

Usually, the library vendor provides a LEF file as a phantom/abstract view for DEF/LEF design flows. This standard file can be read into the Laker system to create physical shapes with Pin/Port information.

1. Invoke **File → Import → LEF** to input a LEF file.
2. In the *Import LEF* form, do the following:
    a. Select **Design File** as *OpenCellLibrary_RC_ANT.lef*
    b. Assign **Library Name** to OpenCellLibrary_lef
    c. Assign **Technology → ASCII File** to OpenCellLibrary.tf
    d. Assign **Layer Map File** to lef_layer.map.

After a LEF file is imported with a Laker technology file, macro cell information is from a LEF file and router rule information is from the *tfNetRouteRule* section defined in a Laker technology file.

Route Via has higher priority than MCell via if both of them exist.



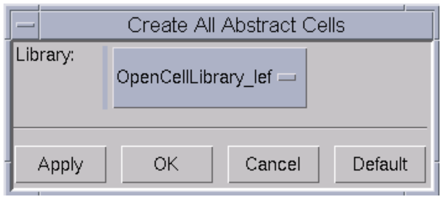
Import LEF

Design File: rce/library/OpenCellLibrary_RC_ANT.lef
Top Cell Name:
Run Directory: .
Library Name: OpenCellLibrary_lef
Technology File
● ASCII File: ./OpenCellLibrary.tf
○ Attach to Library:
☑ Create Pin Name as Text
Text Layer Name: Purpose Name:
Text Font Height: 0.2
☐ Snap Vertices to Min Grid Resolution
☑ Overwrite Duplicated Information
☐ Invoke Batch Mode
Cell Map File:
Layer Map File: ../source/map/lef_layer.map
Log/Error File:

4.3.2 Export Stream of Abstract Views

If you want to reuse LEF macro cell information in an annotated GDS file, refer to the following steps. Because an abstract view cannot be streamed out from the Laker system, all abstract views have to be instantiated in a layout cell in advance.

1. Use the script **CustomDigital → Create All Abstract Cells** in the *Main* window to create a layout view named "CreateAll".

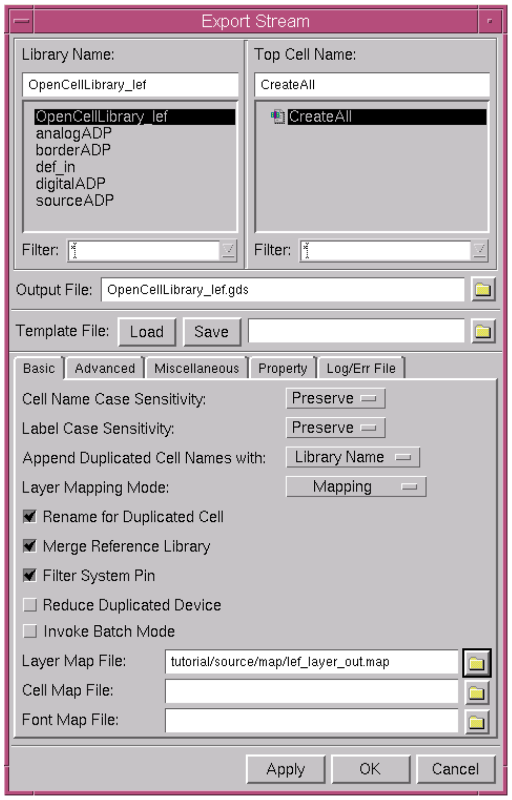


Create All Abstract Cells

Library: OpenCellLibrary_lef

2. Invoke **File → Export → Stream** to open the *Export Stream* form and stream out the cell of “CreateAll”.
3. In the *Export Stream* form, do the following:
   a. On the **Basic** tab, set **Layer Map File** to lef_layer_out.map
   Usually, contact, metal1, via1 and metal2 blockage are also mapped to normal drawing layers without net information.
   b. On the **Advanced** tab, set **Export Abstract Cell View**: All
   This is necessary because the Laker system does not stream out the Abstract view by default.
   c. On the **Property** tab, enable the **Generate Logic Info to Prop No.:** option and set the value to 103.

An annotated GDS can easily recover pin/port information by **Import Stream**.



Output File: OpenCellLibrary_lef.gds

Cell Name Case Sensitivity: Preserve
Label Case Sensitivity: Preserve
Append Duplicated Cell Names with: Library Name
Layer Mapping Mode: Mapping
☑ Rename for Duplicated Cell
☑ Merge Reference Library
☑ Filter System Pin
☐ Reduce Duplicated Device
☐ Invoke Batch Mode
Layer Map File: tutorial/source/map/lef_layer_out.map
Cell Map File:
Font Map File:

4.4 Lab-2C: Verilog and SPICE library files

Dummy Verilog and SPICE library files are useful for design import in Laker CDL-In and Laker-ADP Verilog In.

In general, a Verilog library file provided by a library vendor is a behavior model of module and primitive for post-layout simulation. These behavior sections are not suitable for several backend tools. Only a dummy Verilog library file is needed to define module port name and direction.

A pre-layout SPICE netlist is provided for transistor level cell creation and LVS purposes. For cell level applications, only a dummy SPICE library file is needed for sub-circuit port name and direction.

How to create a dummy Verilog file based on a Verilog library behavior model?

- It can be easily done by Unix “egrep” and “vi” commands.
  a. Sort out module and interface definition
     > egrep "module|input|output|inout" OpenCellLibrary.v > OpenCellLibrary_dummy.v
     > Open “OpenCellLibrary_dummy.v” and remove some extra lines from a primitive section.

  b. Sort out module and interface definition
     > egrep –i “subckt|ends|pininfo" OpenCellLibrary.sp > OpenCellLibrary_dummy.sp
     > Open “OpenCellLibrary_dummy.sp” and remove some extra lines if necessary.

- A simple script in Perl, Tcl or Python can be written for customized result.

- Use Mentor Calibre **v2lvs** or Synopsys Hercules “**nettran**” command

In genral, a verilog netlist of a standard cell does not include power and ground informaiton. Howerwe, a layout view of a standard cell has power and ground shapes and results in port mismatch in Laker SDL flow.

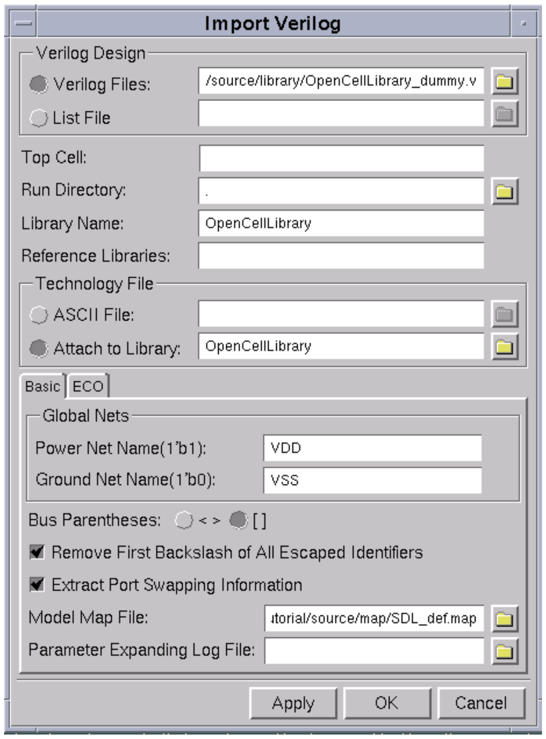
One solution of this port discprepancy issue is adding supply1 and supply0 nets in a verilog netlist.

```
module AND2_X1 (A1, A2, ZN);
  supply1 VDD;
  supply0 VSS;
  input A1;
  input A2;
  output ZN;
endmodule
```

4.4.1 Import Verilog library Files

This dummy Verilog library file can be read into the Laker system to create symbol views.

1. Invoke **File → Import → Verilog** to input a Verilog library file.
2. In the *Import Verilog* form, do the following:
   a. Select **Verilog Files** as OpenCellLibrary_dummy.v
   b. Assign **Library Name** to OpenCellLibrary
   c. Assign **Technology → Attach to Library** to OpenCellLibrary
   d. Assign **Global Nets → Power Net Name(1'b1)** to VDD
   e. Assign **Global Nets → Ground Net Name(1'b0)** to VSS

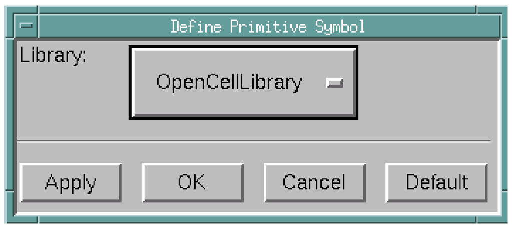

Import Verilog

Verilog Design
● Verilog Files: /source/library/OpenCellLibrary_dummy.v
○ List File:
Top Cell:
Run Directory: .
Library Name: OpenCellLibrary
Reference Libraries:
Technology File
○ ASCII File:
● Attach to Library: OpenCellLibrary
Basic | ECO
Global Nets
Power Net Name(1'b1): VDD
Ground Net Name(1'b0): VSS
Bus Parentheses: ○ < > ● [ ]
✓ Remove First Backslash of All Escaped Identifiers
✓ Extract Port Swapping Information
Model Map File: ıtorial/source/map/SDL_def.map
Parameter Expanding Log File:

4.4.2 Define Primitive Symbol

We can define the *verilogStop* attribute on a symbol view to suppress the primitive cells exported by **File → Export → Verilog** command.

1. Use the script **CustomDigital → Define Primitive Symbol** in the *Main* window to generate a *verilogStop* attribute on all symbol views of the selected standard cell library.

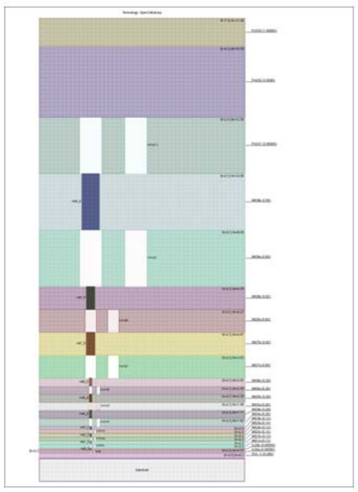


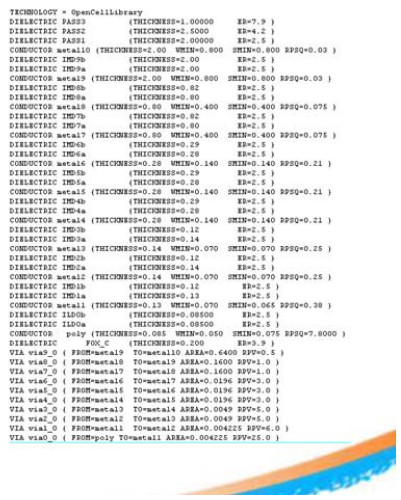
Library: OpenCellLibrary

4.5 RC Table Preparation

Interconnect Technology File format defines the process profile of layer and via, including thickness, height, width, spacing, dielectric, etc. It also defines the resistance of layer and via.

- Based on a RC Technology file in ITF format, the Laker system can create large sets of look-up.
- Tables for pre-defined patterns can be created by built in or third party field solver.
- A LEF file is for reference of routing pitch instead of minimum spacing.
- For third party correlation flow, a correlation SPEF is used.



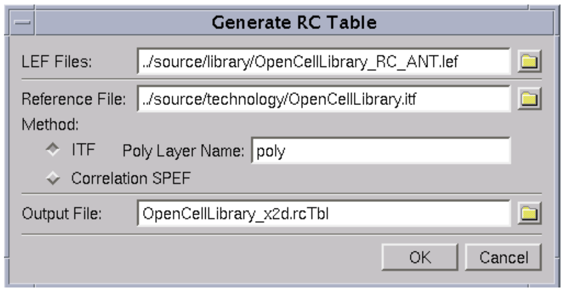


```
TECHNOLOGY = OpenCellLibrary
DIELECTRIC PASS3         (THICKNESS=1.00000     ER=7.9 )
DIELECTRIC PASS2         (THICKNESS=2.5000      ER=4.2 )
DIELECTRIC PASS1         (THICKNESS=2.00000     ER=2.5 )
CONDUCTOR metal10 (THICKNESS=2.00  WMIN=0.800  SMIN=0.800 RPSQ=0.03 )
DIELECTRIC IMD9b         (THICKNESS=2.00        ER=2.5 )
DIELECTRIC IMD9a         (THICKNESS=2.00        ER=2.5 )
CONDUCTOR metal9 (THICKNESS=2.00  WMIN=0.800  SMIN=0.800 RPSQ=0.03 )
DIELECTRIC IMD8b         (THICKNESS=0.82        ER=2.5 )
DIELECTRIC IMD8a         (THICKNESS=0.80        ER=2.5 )
CONDUCTOR metal8 (THICKNESS=0.80  WMIN=0.400  SMIN=0.400 RPSQ=0.075 )
DIELECTRIC IMD7b         (THICKNESS=0.82        ER=2.5 )
DIELECTRIC IMD7a         (THICKNESS=0.80        ER=2.5 )
CONDUCTOR metal7 (THICKNESS=0.80  WMIN=0.400  SMIN=0.400 RPSQ=0.075 )
DIELECTRIC IMD6b         (THICKNESS=0.29        ER=2.5 )
DIELECTRIC IMD6a         (THICKNESS=0.28        ER=2.5 )
CONDUCTOR metal6 (THICKNESS=0.28  WMIN=0.140  SMIN=0.140 RPSQ=0.21 )
DIELECTRIC IMD5b         (THICKNESS=0.29        ER=2.5 )
DIELECTRIC IMD5a         (THICKNESS=0.28        ER=2.5 )
CONDUCTOR metal5 (THICKNESS=0.28  WMIN=0.140  SMIN=0.140 RPSQ=0.21 )
DIELECTRIC IMD4b         (THICKNESS=0.29        ER=2.5 )
DIELECTRIC IMD4a         (THICKNESS=0.28        ER=2.5 )
CONDUCTOR metal4 (THICKNESS=0.28  WMIN=0.140  SMIN=0.140 RPSQ=0.21 )
DIELECTRIC IMD3b         (THICKNESS=0.12        ER=2.5 )
DIELECTRIC IMD3a         (THICKNESS=0.14        ER=2.5 )
CONDUCTOR metal3 (THICKNESS=0.14  WMIN=0.070  SMIN=0.070 RPSQ=0.25 )
DIELECTRIC IMD2b         (THICKNESS=0.12        ER=2.5 )
DIELECTRIC IMD2a         (THICKNESS=0.14        ER=2.5 )
CONDUCTOR metal2 (THICKNESS=0.14  WMIN=0.070  SMIN=0.070 RPSQ=0.25 )
DIELECTRIC IMD1b         (THICKNESS=0.12         ER=2.5 )
DIELECTRIC IMD1a         (THICKNESS=0.13         ER=2.5 )
CONDUCTOR metal1 (THICKNESS=0.13  WMIN=0.070  SMIN=0.065 RPSQ=0.38 )
DIELECTRIC ILD0b         (THICKNESS=0.08500      ER=2.5 )
DIELECTRIC ILD0a         (THICKNESS=0.08500      ER=2.5 )
CONDUCTOR   poly (THICKNESS=0.085  WMIN=0.050  SMIN=0.075 RPSQ=7.8000 )
DIELECTRIC      FOX_C    (THICKNESS=0.200       ER=3.9 )
VIA via9_0 ( FROM=metal9  TO=metal10 AREA=0.6400 RPV=0.5 )
VIA via8_0 ( FROM=metal8  TO=metal9 AREA=0.1600 RPV=1.0 )
VIA via7_0 ( FROM=metal7  TO=metal8 AREA=0.1600 RPV=1.0 )
VIA via6_0 ( FROM=metal6  TO=metal7 AREA=0.0196 RPV=3.0 )
VIA via5_0 ( FROM=metal5  TO=metal6 AREA=0.0196 RPV=3.0 )
VIA via4_0 ( FROM=metal4  TO=metal5 AREA=0.0196 RPV=3.0 )
VIA via3_0 ( FROM=metal3  TO=metal4 AREA=0.0049 RPV=5.0 )
VIA via2_0 ( FROM=metal2  TO=metal3 AREA=0.0049 RPV=5.0 )
VIA via1_0 ( FROM=metal1  TO=metal2 AREA=0.004225 RPV=6.0 )
VIA via0_0 ( FROM=poly TO=metal1 AREA=0.004225 RPV=25.0 )
```

4.5.1 ITF Flow

Given a pre-defined ITF file, an RC table can be created by the Laker built-in solver.

1. Invoke **Library → Prepare RC Table → Generate RC Table** from the *Main* window.
2. In the *Generate RC Table* form, do the following:
   a. Select **LEF Files** as *OpenCellLibrary_RC_ANT.lef*
   b. Assign **Reference File** to *OpenCellLibrary.itf*
   c. Select **Method → ITF**
      Assign **Poly Layer Name** to *poly*
   d. Assign **Output File** to *OpenCellLibrary_x2d.rcTbl*



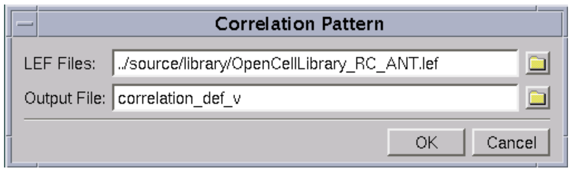
LEF Files: ../source/library/OpenCellLibrary_RC_ANT.lef
Reference File: ../source/technology/OpenCellLibrary.itf
Method:
ITF Poly Layer Name: poly
Correlation SPEF
Output File: OpenCellLibrary_x2d.rcTbl

4.5.2 Third Party Correlation Flow

4.5.2.1 Create Correlation Pattern Files

For those who want to use a third-party sign off tool to extract created correlation patterns, the Laker Custom Digital product also provides a smooth correlation flow.

1. Invoke **Library → Prepare RC Table → Correlation Pattern** from *Main* window.
2. In the *Correlation Pattern* form, do the following:
   a. Select **LEF Files** as *OpenCellLibrary_RC_ANT.lef*
   b. Assign **Output File** to *correlation_def_v*



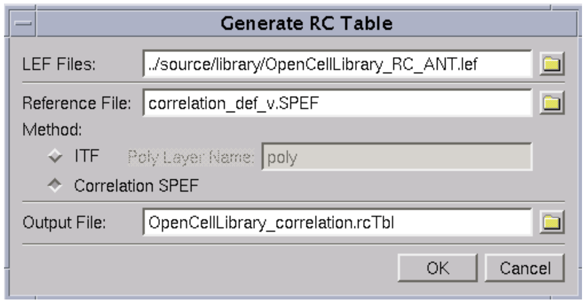
LEF Files: ../source/library/OpenCellLibrary_RC_ANT.lef
Output File: correlation_def_v

Two files named correlation_def_special.def and correlation_def_special.v are created for third party sign off extraction tool.

4.5.2.2 Create correlation RC Table

A correlation RC table can be generated after a correlation SPEF file is generated by third party tools.

1. Invoke **Library → Prepare RC Table → Generate RC Table** from the *Main* window.
2. In the *Generate RC Table* form, do the following:
   a. Select **LEF Files** as *OpenCellLibrary_RC_ANT.lef*
   b. Assign **Reference File** to *correlaton_dev_v.SPEF*
   c. Select **Method → Correlation SPEF**
   d. Assign **Output File** to *OpenCellLibrary_correlation.rcTbl*



LEF Files: ../source/library/OpenCellLibrary_RC_ANT.lef
Reference File: correlation_def_v.SPEF
Method:
ITF Poly Layer Name: poly
Correlation SPEF
Output File: OpenCellLibrary_correlation.rcTbl

4.6 Dual View Library Preparation

Dual library preparation is a common method to utilize layout views for custom layout and abstract views for timing driven related environment. The following steps will guide you to preparing a dual view library in the Laker Custom Digital environment.

1. Import a complete LEF file with a Laker technology file.
   A technology database comes from a Laker technology file followed by a LEF file. For example, both MCell and RouteVia are available. Site and antenna information are available from a LEF file as well.

   Only abstract view is available at current stage.

2. Import a complete stream file with annotated pin/port information if applicable.
   Both layout view and abstract view are available.

3. Assign pin/port information on the layout view.
4. Update the cell property if necessary.
5. Import a library Verilog netlist with port interface and supply1/supply0 global nets.
   A symbol view is created for import Verilog flow.

6. Define the verilogStop attribute on a symbol view.

Layout, abstract, and symbol views are all ready for Laker timing driven flow information after this stage.

5 Design Preparation

The Laker SDL flow needs both logic and layout views. The logic view represents the logic connection of a design with full parameters. The layout view represents the layout connection.

- Logic view: Laker ADP schematic + Expand Schematic, Laker ADP Import Verilog + Expand Schematic, Laker CDL In

- Layout view: Laker Import DEF with Laker TF/LEF library

The benefits the SDL flow provides are incremental implementation and cross-probing among design hierarchy browser window, schematic window and layout window.

The support of LEF/DEF flow provides the same essential kernel features for third party tools.

5.1 Lab-3A: Design Preparation by CDL

In this lab, you will learn how to create a CDL In design library with a Laker library in layout view.

5.1.1 Import CDL files

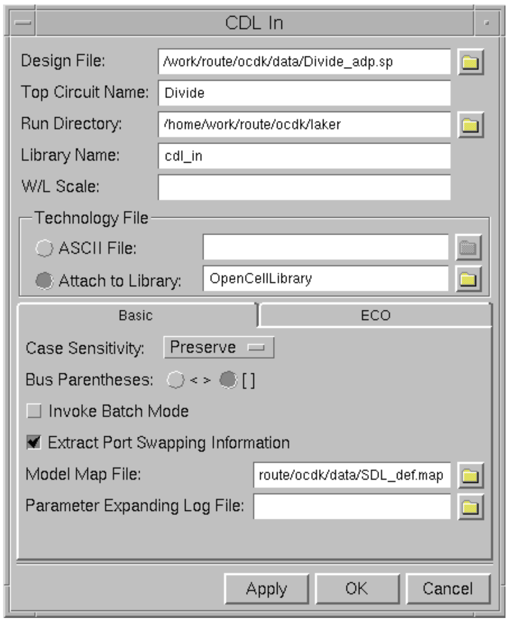
1. Define a cell library in a library mapping path by **Library → Mapping Path**
   Make sure *OpenCellLibrary* is listed in the mapping path.

2. Invoke **File → Import → CDL In** to import a CDL design.
   A CDL dummy library file is included in the design file to create the necessary logic view.

   CDL *PININFO information is highly recommended to provide necessary port direction for automatic smart schematic generation. It is not only for a more readable schematic for manual selection but also for meaningful topology relationship for some topology driven features. For example;

```
.subckt XNOR2_X2 A B ZN
*.pininfo A:I B:I ZN:O

.ends XNOR2_X2
```

3. In the *CDL In* form, do the following:
   a. Assign **Design File** to *Divide.sp*
   b. Assign **Top Circuit Name** to *Divide*
   c. Assign **Library Name** to *cdl_in*
   d. Assign **Technology File → Attach to Library** to *OpenCellLibrary*
   e. Assign **Model Map File** to *SDL_def.map*.
   f. Click **OK** to finish importing a CDL file.



Design File: /work/route/ocdk/data/Divide_adp.sp
Top Circuit Name: Divide
Run Directory: /home/work/route/ocdk/laker
Library Name: cdl_in
W/L Scale:
Technology File
ASCII File:
● Attach to Library: OpenCellLibrary
Case Sensitivity: Preserve
Bus Parentheses: < > [✓ [ ]]
☐ Invoke Batch Mode
☑ Extract Port Swapping Information
Model Map File: route/ocdk/data/SDL_def.map
Parameter Expanding Log File:

Here is an example of legacy model map file to enable cell mapping and removal of prefix “X” from CDL file format.

```
[MAP]
X   *  OpenCellLibrary   *

[ELEMENT_PREFIX]
X *         X

[GLOBAL_NET]
VDD           p
VSS           g
```

Here is another example of new model map file to enable cell mapping with view names and removal of prefix “X” from CDL file format.

```
[MAP]
X   *  OpenCellLibrary  { * abs }
[ELEMENT_PREFIX]
X *         X
```

```
[GLOBAL_NET]
VDD          p
VSS          g
```

NOTE: LakerOA has a new model map file section to support MCell, Tcl PCell, PyCell in a general SDL flow. Please refer to Limitations and Known Problems section for details.

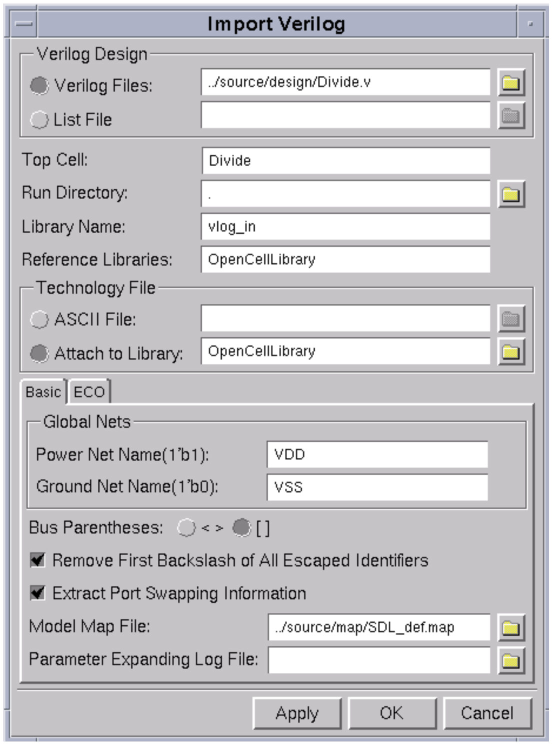
5.2 Lab-3B: Design Preparation by Verilog

In this lab, you will learn how to create a Verilog In design library with a Laker library in layout view.

5.2.1 Import Verilog Files

The Laker system supports structural Verilog netlist with simple "assign" statements for SDL flow.

1. Define a cell library in a library mapping path by invoking the **Library → Mapping Path** command. Make sure OpenCellLibrary is listed in the mapping path.
2. Invoke **File → Import → Verilog** in the *Main* window to import a Verilog netlist design.



Import Verilog

Verilog Design
- ● Verilog Files: ../source/design/Divide.v
- ○ List File:

Top Cell: Divide
Run Directory: .
Library Name: vlog_in
Reference Libraries: OpenCellLibrary

Technology File
- ○ ASCII File:
- ● Attach to Library: OpenCellLibrary

Basic | ECO

Global Nets
- Power Net Name(1'b1): VDD
- Ground Net Name(1'b0): VSS

Bus Parentheses: ○ < > ● [ ]
☑ Remove First Backslash of All Escaped Identifiers
☑ Extract Port Swapping Information
Model Map File: ../source/map/SDL_def.map
Parameter Expanding Log File:

3. In the *Import Verilog* form, do the following:
   a. Set **Verilog Design → Verilog Files** to ../source/design/*Divide.v*
   It is recommended to create a symbol view and attach the primitive attribute by importing library Verilog files during the library preparation.
   b. Set **Library Name** to *vlog_in*
   c. Set **Reference Library** to *OpenCellLibrary*
   d. Set **Technology File → Attach to Library** to *OpenCellLibrary*
   e. Set **Global Nets → Power Net Name(1'b1)** to *VDD*
   f. Set **Global Nets → Ground Net Name(1'b0)** to *VSS*
   g. Set **Model Map File** to ../source/map/*SDL_def.map*

4. Click **OK** to finish importing the Verilog files.

NOTE: LakerOA and LakerDB 2011.06 versions have a new model map file section to support MCell, Tcl PCell, PyCell in a general SDL flow. Please refer Limitation and Known Problems for details. A new example is also available in *source/SDL_oa_def.map* for Laker OA flow.

5.3 Lab-3C: Design Preparation by Verilog to CDL

5.3.1 Verilog to CDL Conversion by LVS Utility

Signoff LVS tool provides a utility for LVS schematic netlist preparation converted from Verilog, EDIF, SPICE, CDL netlist, etc.

For example, Mentor Calibre **v2lvs** or Synopsys Hercules “**nettran**” commands can be used for Verilog to CDL conversion.

1. Use Mentor Calibre v2lvs command:
```
v2lvs -i -s0 VSS -s1 VDD -v Divide.v –l OpenCellLibrary_dummy.v
-o Divide.sp
```

2. Use Synopsys Hercules nettran command:
```
nettran -verilog-b0 VSS -verilog-b1 VDD -verilog Divide.v
OpenCellLibrary_dummy.v –outName Divide.sp -outType spice
```

After a design CDL file is successfully converted, follow the standard flow of Laker design preparation by CDL as described above.

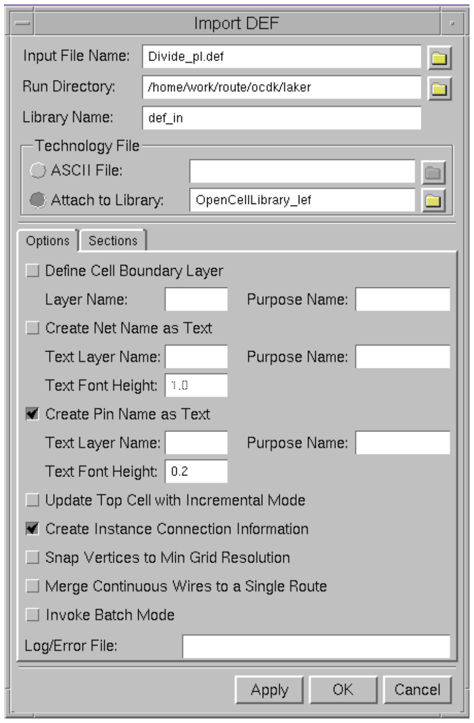
5.4 Lab-3D: Design Preparation by DEF

In this lab, you will learn how to create a DEF In design library with a Laker library in abs (abstract) view.

5.4.1 Import DEF files

1. Define a cell library in a library mapping path by the **Library → Mapping Path** command. Make sure *OpenCellLibrary_lef* is listed in the mapping path.

2. Invoke the **File → Import → DEF** command to import a DEF design.
Standard via, custom via, site information used in a DEF file are pre-defined in a corresponding LEF file. A correct Laker TF/LEF library is necessary for a successful DEF import task.



Input File Name: Divide_pl.def
Run Directory: /home/work/route/ocdk/laker
Library Name: def_in
Technology File
ASCII File:
Attach to Library: OpenCellLibrary_lef
Define Cell Boundary Layer
Layer Name: Purpose Name:
Create Net Name as Text
Text Layer Name: Purpose Name:
Text Font Height: 1.0
[✓] Create Pin Name as Text
Text Layer Name: Purpose Name:
Text Font Height: 0.2
Update Top Cell with Incremental Mode
[✓] Create Instance Connection Information
Snap Vertices to Min Grid Resolution
Merge Continuous Wires to a Single Route
Invoke Batch Mode
Log/Error File:

3. In the *Import DEF* form, do the following:
   a. Set **Input File Name** to *Divide_pl.def*
   b. Set **Library Name** to *def_in*
   c. Set **Technology File → Attach to Library** to *OpenCellLibrary_lef*
   d. Set **Create Pin Name as Text → Text Font Height** to 0.2
   e. Enable **Create Instance Connection Information**
4. Click **OK** to finish importing a DEF file.

Remember that logic views are not created for pure DEF/DEF flow. Only a layout view is created after Import DEF.

Row area created by DEF/LEF flow cannot be recognized by the Laker SDL flow. You can only use **Placer → Place All** from the Layout Design Window.

5.5 Lab-3E: Design Preparation by GDS

In this lab, you will learn how to create a GDS In design library with a Laker library in layout view.

The Laker generated annotated GDS file is well supported to restore instance name, net, port and instance port by Import Stream. If you have an annotated GDS file created by a third party tool, the instance name and net name can be restored with correct property.

The following are known limitations for this flow:

- Standard via and custom via will be mapped to normal instances for a GDS file created from a DEF/DEF database.
- Row information is not stored and recovered.

5.5.1 Import Stream Files

1. Open the *Import Stream* form by **File → Import → Stream**.
2. Select **Open Cell Library GDS** file named *Divide_pl.gds.*

3. In the *Import Stream* form, do the following:
   a. Assign **Library Name**: OpenCellLibrary
   b. Assign Technology file to *OpenCellLibrary.tf*
      Technology → Attach to Library to *OpenCellLibrary*
   c. Assign **Basic → Layer Map File**: gds_layer.map
   d. Set **Basic → Reference Libraries**: to *OpenCellLibrary*
   e. Enable **Advance → Reference Cell from Other Library First**
      Cell mapping to central library OpenCellLibrary is enforced by this option.
   f. Enable **Property → Generate Instance Name as Prop No.:** *102*
   g. Enable **Property → Generate Net as Prop No.:** *102*
   h. Enable **Property → Generate Logic Info as Prop No.:** *103*
4. Click **OK** and finish importing a stream file.

5.5.2 Assign Net Names

If only instances names are available to be restored, a net name may be recovered by **Design Hierarchy Browser → Fix Discrepancy** or **Net Propagation**.

1. Create Port/Pin by *Query* attribute form for top metal, cut and bottom metal layers of via instances streamed in from a third party tool. This procedure is necessary for Net Propagation.
2. In the design hierarchy browser pane, invoke **Fix Discrepancy** after selecting all instances by **Select All**.
3. Use toolbox utility by **L3TCL → Rebuild Connection**.

6 Floorplan

6.1 Lab-4A: Floorplan Initialization

In this lab, you will learn how to create a row area and initial pin assignment in a layout view after design preparation.

The following procedures apply to general Laker SDL database either by CDL-In or by Expand Schematic. A CDL-In case will be used for demonstration.

6.1.1 Routing Resource Plan

In general practice, area utilization rate is commonly used to decide the size of Row Area for row placement. The area utilization rate is defined as the ratio of total size of cells over row area in analog design style. The magic utilization rate number differs due to design style and available routing layers. The average magic number for general design is listed below for reference.

Of course, you can use higher utilization for a compact design in local net connection.

Available Routing Layer	***Average Utilization***	***Comments***
2m	0.5 ~ 0.6 (*)	Channel or pseudo blockage (row abutted)
3m	0.7 ~ 0.75	Channel-less (row abutted)
4m	0.8 ~ 0.9	Channel-less (row abutted)
5m	0.8 ~ 0.9	Channel-less (row abutted)

* Note: site utilization rate will be higher than area utilization rate if row space is reserved for limited route layer in channel floor plan.

The following conditions will be used for this demo case.

- 4 metal routing layers
- The area multiplier rate used for Design Hierarchy Browser → Re-Estimate Area is 1.2.
- The area utilization rate used for **Placer → Create Row** is 0.8.

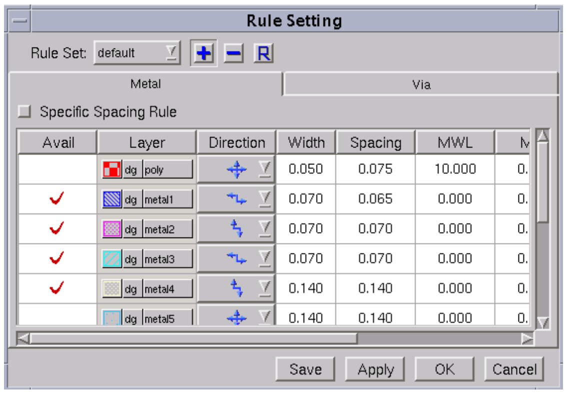
The initial utilization rate does not consider placement blockage, physical only cells and so on. Therefore, final utilization rate will be higher than the initial utilization rate.

6.1.2 Set Routing Layers

The Custom Digital features honor the Laker routing resource files for available routing layer planning.

If 4 metal routing layers are used, non-available routing layers in 10 metal layers have to be disabled in the routing rule setting.

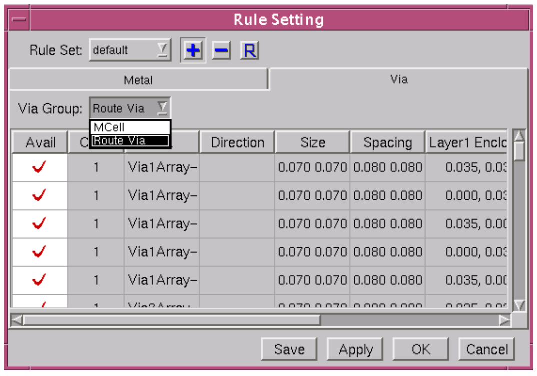
1. Invoke the **Router → Digital Route → Rule Setting** command.
2. Disable availability of non-available layers and vias by clicking the **Avail** field of **Rule Setting → Metal** tab.
3. Click **Rule Set → Save** to save the 4 metal routing conditions to *default* for future use.



Avail	Layer	Direction	Width	Spacing	MWL	M
	dg poly		0.050	0.075	10.000	0.
✓	dg metal1		0.070	0.065	0.000	0.
✓	dg metal2		0.070	0.070	0.000	0.
✓	dg metal3		0.070	0.070	0.000	0.
✓	dg metal4		0.140	0.140	0.000	0.
	dg metal5		0.140	0.140	0.000	0.

4. Select Via group of *MCell* or *Route Via* by **Via Group** field of **Rule Setting → Via** tab.
5. Select availability of vias by clicking the **Avail** field of the **Rule Setting → Metal** tab.

Route Via of **Via Group** is recommended for timing driven flow because most APR tools supports standard LEF/DEF interface.



Avail	C		Direction	Size	Spacing	Layer1 Enclo
✓	1	Via1Array–		0.070 0.070	0.080 0.080	0.035, 0.03
✓	1	Via1Array–		0.070 0.070	0.080 0.080	0.000, 0.03
✓	1	Via1Array–		0.070 0.070	0.080 0.080	0.035, 0.00
✓	1	Via1Array–		0.070 0.070	0.080 0.080	0.000, 0.03
✓	1	Via1Array–		0.070 0.070	0.080 0.080	0.035, 0.00

6. Click **OK** to close the *Rule Setting* form.

6.1.3 Initialize Area Estimation

After the CDL In task is done, both schematic view and logic view are created. An initial layout view can be created now.

1. Invoke the **File → Open** command.
2. In the *Open Cell* form, do the following:
   a. Set **Library** to *cdl_in*
   b. Set **Cell** to *Divide*
   c. Set **View** to *logic*
3. Click **Okay** to create an initial layout view.
4. Estimate design area
5. Manually remove power and ground Soft Pins

Soft Boundary (the *softBdry* layer in cyan color) is a boundary layer of soft macro used in top down floor planning. It provides flexibility of manual or automatic macro shaping while keeping a reserved area size. You can increase view level and stretch *softBdry* layer of lower level block. Stretching the *softBdry* layer of cell level (or Edit-In-Place) will not keep the reserved area size.

If it is a hierarchical layout design implementation, the **L3TCL → Hierarchical Re-Area** Tcl script can be used to bottom-up estimate area of the whole hierarchy.

The **LakerAPR → Estimate Area** feature was developed for area re-estimation of realized hierarchical instances with *softBdry* layer. The layout area of the referenced cell view is retrieved from the model map file.

If it is a flat layout design implementation, the hierarchy of logic view can be flattened by **Design Hierarchy Browser → Flatten → All Levels** first followed by **Design Hierarchy Browser → Re-Estimate Area.**

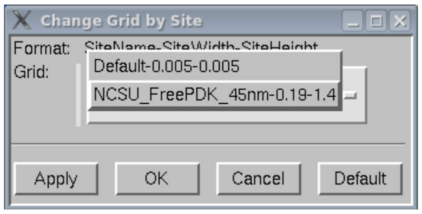
Soft Pin (the softPin layer in pink color) is a concept of pseudo pin without dedicated layer assignment. It is created for manual or automatic pin planning.

6.1.4 Create Row

6.1.4.1 Create Row by Row Area

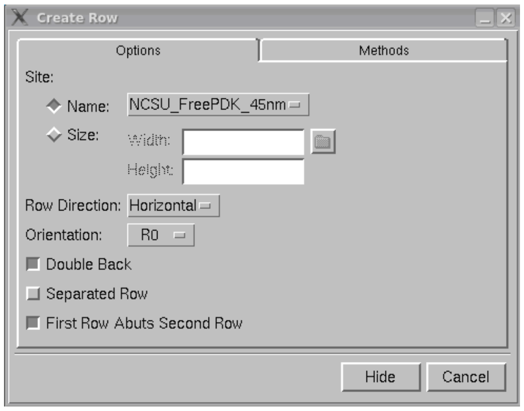
Given an estimated SoftBdry, you can draw a polygon shape with rough equivalent area size. This is good for polygon shape row area creation.

1. Define grid size by site width and height by a customized script **CustomDigital → Change Grid by Site**. Select *NCSU_FreePDK_45nm-0.19-1.4* to snap cursor to site grid points. It is an optional step to draw a polygon shape row area.


Change Grid by Site
Format: SiteName-SiteWidth-SiteHeight
Grid: NCSU_FreePDK_45nm-0.19-1.4
Default-0.005-0.005
NCSU_FreePDK_45nm-0.19-1.4

2. Invoke the **Placer → Create Row** command.
3. In the *Create Row* form, do the following:

a. On the **Options** tab, specify an existing pre-defined site by selecting **Name** as *NCSU_FreePDK_45nm*.
b. Enable row abutted style by enabling the **Double Back** option.
c. On the **Methods** tab, select the **Row Area** method and drawing a rough equivalent area size in polygon shapes. For example, a rectilinear polygon is created referring to rectangle SoftBdry.
d. Set **Row Spacing** to 0.0 for channel-less floor plan.



Create Row

Options | Methods

Site:

◆ Name: NCSU_FreePDK_45nm

◇ Size: Width: Height:

Row Direction: Horizontal

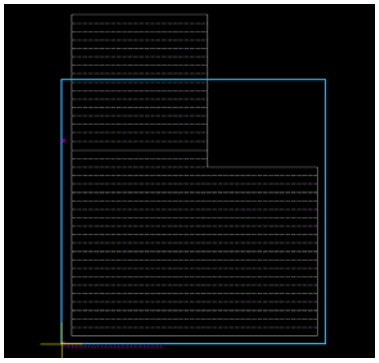
Orientation: R0

☐ Double Back

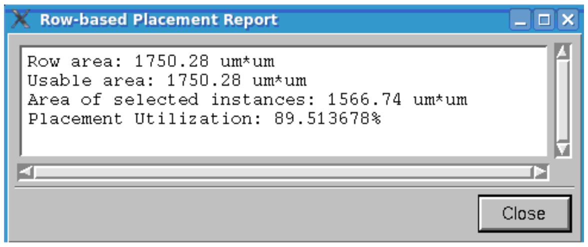
☐ Separated Row

☐ First Row Abuts Second Row

Hide | Cancel



4. Click **Select All** icon in the design hierarchy browser pane to select all objects because the Laker system can support partial selection for implementation.
5. Invoke the **Design Brower → Placement → Row Placer** command.
6. Click the created polygon shape row area and apply **Design Brower → Placement → Row Placer → Estimate** to get the final utilization rate calculated by placer engine.



Row-based Placement Report

```
Row area: 1750.28 um*um
Usable area: 1750.28 um*um
Area of selected instances: 1566.74 um*um
Placement Utilization: 89.513678%
```

Close

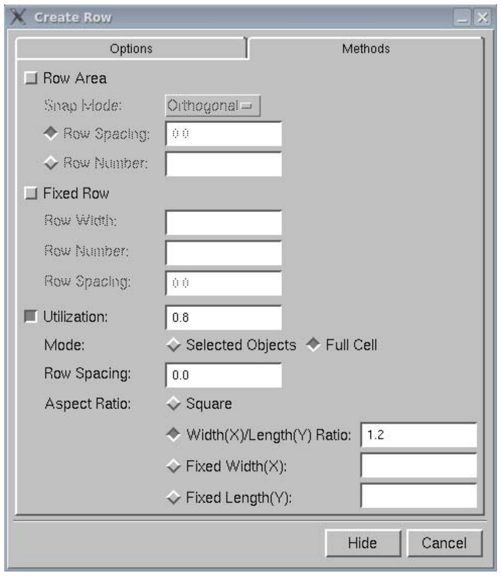
7. Enlarge the created row area by invoking **Placer → Stretch Row** or **Placer → Configure Row** to adjust high utilization to a reasonable rate.
8. Reset grid size by a customized script **CustomDigital → Change Grid by Site**. Select *Default-0.005-0.005* to reset default minimum grid.

You can delete a SofrBdry layer after you have finished the initial floor plan. A CellBdry layer will be created later.

6.1.4.2 Create Row by Utilization

Creating a rectangle row area by utilization is an easier method.

1. Invoke the **Placer → Create Row** command.
2. In the *Create Row* form, do the following:
   a. Select the **Options** tab and specify an existing pre-defined site by selecting **Name** as*NCSU_FreePDK_45nm.*
   b. Enable row abutted style by enabling the **Double Back** option.
   c. Select the **Methods** tab and enable the **Utilization** option and set the associated utilization rate to 0.8 for 4 metal layer routing.
   d. Enable the **Full Cell** option under **Mode**.
   e. Set **Row Spacing** to 0.0 for a channel-less floor plan.



Row Area	☐
Snap Mode:	Orthogonal
Row Spacing:	● 0.0
Row Number:	◇
Fixed Row	☐
Row Width:	
Row Number:	
Row Spacing:	0.0
Utilization:	■ 0.8
Mode:	◇ Selected Objects [✓ Full Cell]
Row Spacing:	0.0
Aspect Ratio:	◇ Square
	[✓ Width(X)/Length(Y) Ratio: 1.2]
	◇ Fixed Width(X):
	◇ Fixed Length(Y):

Now, you need to define the origin of the row area using one of the following methods:

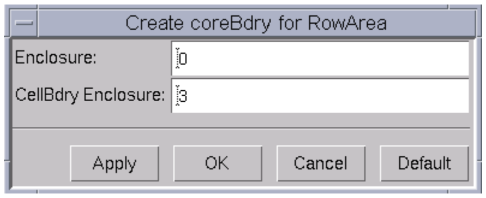
- By mouse left-click, or
- Press **TAB** to enter (X,Y) value at the top of the Layout window, or
- Press **H** to enter (X,Y) value from *User Input Coordinate* window.

1. Click **Select All** icon in the design hierarchy browser pane to select all objects because the Laker system can support partial selection for implementation.
2. Invoke the **Design Brower → Placement → Row Placer** command.
3. Click the created rectangle row area and apply **Design Brower → Placement → Row Placer → Estimate** to get the final utilization rate calculated by placer engine.

You can delete the SofrBdry layer after you have finished the initial floor plan. A CellBdry layer will be created later.

6.1.5 Create Cell Boundary

Create a CellBdry based on planned core area by a customized script **CustomDigital → Create Core Bdry**.



Enclosure: 0
CellBdry Enclosure: 3

CoreBdry is a layer created for legacy supported feature only if necessary.

CellBdry Enclosure is the distance reserved for ring structure around the whole Row Area.

6.1.6 Initial Pin Assignment

At the very beginning stage, pin constraint can be assigned from top-down or bottom-up floor plan tasks. In this case of block level design, how to handle pin assignment with initial pin constraints will be demonstrated.

The common basic pin constraint includes edge, order, layer and size, etc. The following pin constraint file is an example to simply arrange bus signals on edge and evenly distribute before cell placement. Because top down pin assignment is supported, pin constraint is library and design related.

```
pinConstraint {
   cellName {Divide}
#  refLibrary {cdl_in}
   group {
   }
   boundary {
      {Edge1 {

       {CLK  {layer metal3 drawing} {size 0.07 0.4}}
       {X[0:11] {layer metal3 drawing} {size 0.07 0.4}}
       {Y[0:11] {layer metal3 drawing} {size 0.07 0.4}}

      }}
      {Edge2 {

      }}
      {Edge3 {

       {Q[11:3] {layer metal3 drawing} {size 0.07 0.4}}
       {R[11:0] {layer metal3 drawing} {size 0.07 0.4}}
       {Q[0:2] {layer metal3 drawing} {size 0.07 0.4}}

      }}
      {Edge4 {

      }}
   }
   distribution { Edge1 Edge3 }
}
```

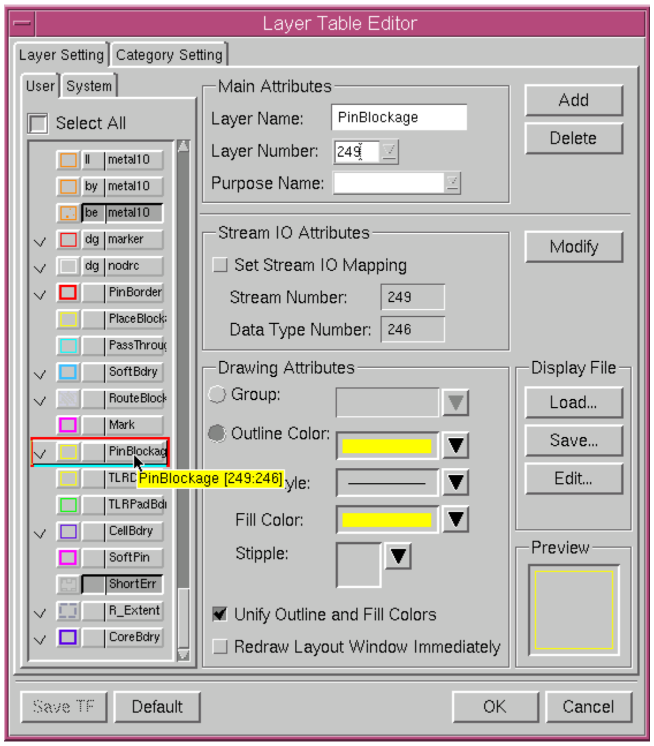
6.1.6.1 Create Soft Pin

If you removed the pink SoftPin in previous steps, you can re-generate all of them by **CustomDigital → Create Soft Pin** or **SDL → Soft Pin → Create Soft Pin**.

Auto pin assignment will place all pins with Float pin status. If you want to manually place a pin, you can change its status to *Fix* by **Query → Connection → Pin Status**.

6.1.6.2 Create Pin Blockage

If you want to protect some regions to avoid pin placement in these regions, you can add PinBlockage (249:246) layer in the floor plan layout. For example, you can add PinBlockage around four corners to prevent any pin placed far away from Row Area.



Main Attributes
Layer Name: PinBlockage
Layer Number: 249
Purpose Name:

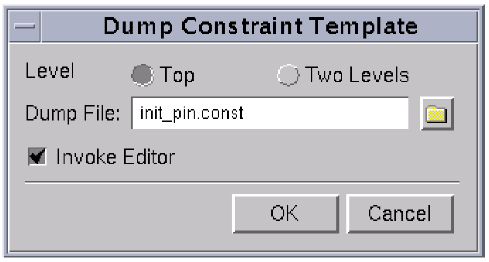
Stream IO Attributes
Set Stream IO Mapping
Stream Number: 249
Data Type Number: 246

Drawing Attributes
Group:
Outline Color:
Style:
Fill Color:
Stipple:
Unify Outline and Fill Colors [✓]
Redraw Layout Window Immediately

6.1.6.3 Dump Template of Pin Constraint

Dumping a template of pin constraint file is a good start to understanding constraint format and knowing how to create an initial pin constraint with few modifications.

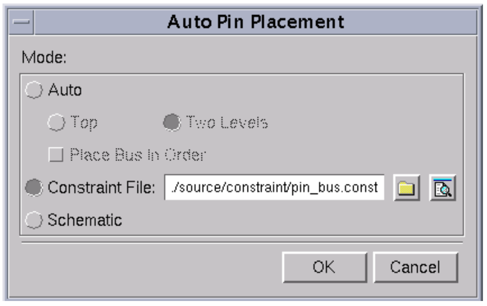
The **Placer → Pin Placer → Dump Constraint Template** command can dump a template file.



Level [✓ Top] Two Levels
Dump File: init_pin.const
Invoke Editor [✓]

6.1.6.4 Auto Pin Assignment

After a pin constraint is ready, the **Placer → Pin Placer → Auto Pin Placement** command can be used to realize initial pin assignment.


Auto Pin Placement

Mode:

( ) Auto

( ) Top [✓ Two Levels]

☐ Place Bus In Order

[✓ Constraint File:] ./source/constraint/pin_bus.const

( ) Schematic

OK Cancel

After this step, the task of initial pin placement is finished.

1. **Auto** mode is for automatic pin assignment without pin constraints. It is often used for top down floor plan or after row placement is done.
   Enabling the **Place Bus in Order** option can automatically recognize the bus structure of pin names in square brackets [] and angle brackets <>.
2. **Constraint File** mode follows user defined pin constraints for **Top** or **Two Levels** assignment.
3. **Schematic** mode recognizes pin locations placed in the top schematic and honor them for initial pin assignment.

The purpose of initial pin placement before a row placement is to guide a net weight of I/O connection. Otherwise, the first stage instance connected to I/O pins will be biased by the net weight of the second stage instances.

After a row placement, the optional pin optimization step will be introduced later.

7 Pre-Placement

7.1 Lab-5A: Add Physical Only Cells (Optional)

In this lab, you will learn how to add well tap or end cap cells, assign PG logic connection, create ring, stripes and follow pin before row placement.

7.1.1 Introduction of Physical Only Cells

Physical only cells are layout only cells which are not included in the imported design netlist. For example, well tap, end cap, core filler, I/O filler, I/O corner, antenna diode, decoupling-cap, GA spare cells, etc. Standard spare cells are usually planned and included in the original netlist.

- Well tap cell: a well-tie pick-up to prevent latch up issues for tap-less standard cells without built-in well-tie pick up. These cells are placed regular within defined distance to cover all place-able sites for tap-less cells. You do not need to insert well tap cells for normal standard cells.

- End cap cell: They are placed at the edge of place-able row area to keep the regularity of layer patterns to minimize process effect.

- Core filler cell: generic filler cells can be inserted in empty sites to prevent DRC violations.

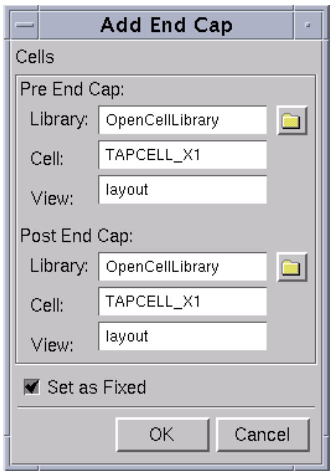
- I/O filler cell: generic I/O filler is added to connect I/O ring power distribution and prevent DRC violations.

- I/O corner cell: generic cells to connect I/O ring power distribution for different I/O ring.
- Antenna cell: antenna diode to provide additional diffusion protection for antenna violation fix.
- Decoupling-cap cell: power decoupling cells to reduce current surge and voltage drop.
- GA spare cell: programmable gate array spare cells for flexible post-silicon ECO.

7.1.2 Add End Cap

TAPCELL_X1 is used for both well tap cell and end cap cells in this tutorial.

1. Invoke the **Placer → Add End Cap** command.
2. In the *Add End Cap* form, do the following:
   a. Select an existing pre-defined TAPCELL_X1 by specifying **Pre End Cap:**
      Library: OpenCellLibrary
      Cell: TAPCELL_X1
      View: layout
   b. Select an existing pre-defined TAPCELL_X1 by specifying **Post End Cap:**
      Library: OpenCellLibrary
      Cell: TAPCELL_X1
      View: layout
   c. Set placement status of fixed after placement enabling the **Set as Fixed** option.
3. Click **OK** to finish end cap placement.



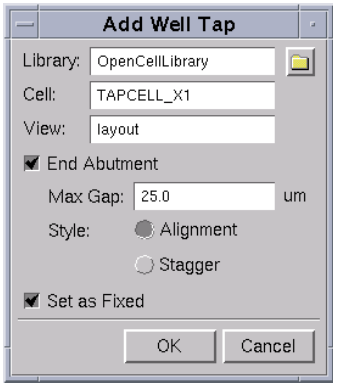
Pre End Cap:
Library: OpenCellLibrary
Cell: TAPCELL_X1
View: layout
Post End Cap:
Library: OpenCellLibrary
Cell: TAPCELL_X1
View: layout
✓ Set as Fixed

7.1.3 Add Well Tap

TAPCELL_X1 is used for both well tap cell and end cap cells in this tutorial.

1. Invoke the **Placer → Add Well Tap** command.
2. In the *Add Well Tap* form, do the following:
   a. Select an existing pre-defined TAPCELL_X1 by specifying:
      Library: OpenCellLibrary
      Cell: TAPCELL_X1
      View: layout
   b. Select **End Abutment** to add abutted well tap cells.
   c. Set maximum distance gap to 25.0 um by specifying:
      Max Gap: 25.0 um
      Style: Alignment
   d. Set placement status of fixed after placement enabling the **Set as Fixed** option.

3. Click **OK** to finish well tap and end cap placement.



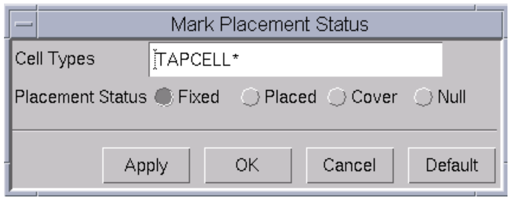
Library: OpenCellLibrary
Cell: TAPCELL_X1
View: layout
[✓] End Abutment
Max Gap: 25.0 um
Style: [✓ Alignment] Stagger
[✓] Set as Fixed

7.1.4 Mark Placement Status of Physical Cells

You can mark placement status of physical cells to prevent change due to automatic or manual operation.

- A cell with **Fixed** placement status cannot be touched by automatic tools like Row Placer, Legalization, etc. but it can be moved by manual editing features.
- A cell with **Cover** placement status cannot be touched by either automatic or manual editing features. You can only change its placement status.
- A cell with **Placed** or **Null** placement status can be touched by automatic or manual editing features.

1. Invoke the **CustomDigital → Mark Placement Status** command.
2. Fill in the **Cell Types** field and select one **Placement Status**.
3. Click **OK** to assign placement status to specified cell types.



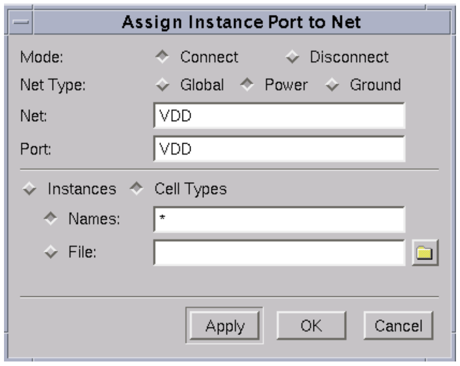
Cell Types: TAPCELL*
Placement Status: [✓ Fixed] Placed Cover Null

You can use **Query → Attribute** to see the placement status is successfully assigned.

7.1.5 PG Connection of Physical Cells

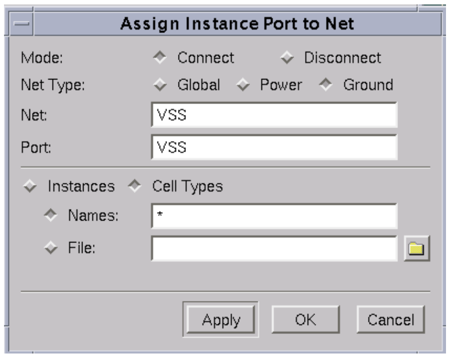
Power and ground ports of a physical only cell are floating because they are not defined during design import. An explicit procedure is needed to connect the power and ground ports to global PG nets.

1. Invoke the **Router → Digital Router → Assign Instance Port to Net** command.
2. Set **Net Type** to *Power*
3. Fill in both **Net** and **Port** fields as VDD
4. Select the **Cell Types** and **Names** options.
5. Fill in wildcard * for all cell types.
6. Click **Apply** to assign power nets to a physical cell by cell type.
7. Set **Net Type** to *Ground.*
8. Fill in both **Net** and **Port** fields as VSS.
9. Click **OK** to assign ground nets to a physical cell by cell type.


Assign Instance Port to Net

Mode: [✓ Connect] Disconnect
Net Type: Global [✓ Power] Ground
Net: VDD
Port: VDD
Instances [✓ Cell Types]
[✓ Names:] *
File:

Apply OK Cancel


Assign Instance Port to Net

Mode: [✓ Connect] Disconnect
Net Type: Global Power [✓ Ground]
Net: VSS
Port: VSS
Instances [✓ Cell Types]
[✓ Names:] *
File:

Apply OK Cancel

You can use **Query → Connectivity** to see if power and ground names are successfully assigned to power and ground ports.

8 Power Plan

8.1 Lab-6A: Create Core Ring and Stripe

In this lab, you will learn how to add core ring and stripe around a core area using the Custom Digital enhanced toolbox.

8.1.1 Introduction of PG Route Type

A SHAPE definition is used to specify a wire with special connection requirements because of its shape. This applies to vias as well as wires.

RING	Used as ring, target for connection
PADRING	Connects padrings
BLOCKRING	Connects rings around the blocks
STRIPE	Used as stripe
FOLLOWPIN	Connects standard cells to power structures
IOWIRE	Connects I/O to target
COREWIRE	Connects endpoints of follow pin to target
BLOCKWIRE	Connects block pin to target
BLOCKAGEWIRE	Connects blockages
FILLWIRE	Represents a fill shape that does not require OPC. It is normally connected to a power or ground net. Floating fill shapes should be in the FILL section.
FILLWIREOPC	Represents a fill shape that requires OPC. It is normally connected to a power or ground net. Floating fill shapes should be in the FILL section.
DRCFILL	Used as a fill shape to correct DRC errors, such as SPACING, MINENCLOSEDAREA, or MINSTEP violations on wires and pins of the same net.

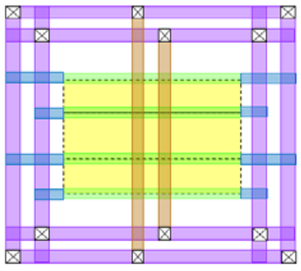
PG routing for RING, STRIPE, FOLLOWPIN and COREWIRES are supported in Laker 2010.03 and later versions.

PG route type in DEF definition

- RING
- STRIPE
- FOLLOWPIN
- COREWIRE
- BLOCKWIRE
- IOWIRE

PG Route supports

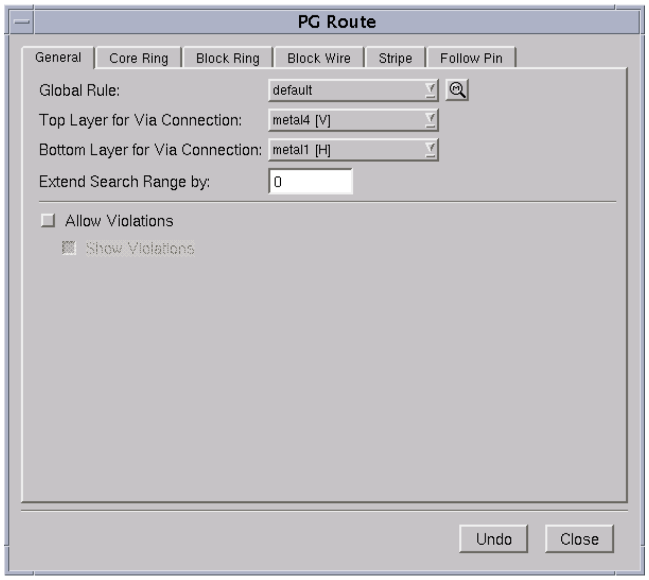
- RING
- STRIPE
- FOLLOWPIN (including COREWIRE)



8.1.2 Create Core Rings

Before creation of core rings, the ring structure, routing layer, width and offset spacing to core area should be planned. In addition, the total reserved distance between CellBdry and Row Area needs to be checked. If its value is not large enough to create the whole ring structure, the floor plan needs to be adjusted in advance.

1. Invoke the **Router → Digital Router → PG Route** command.



PG Route

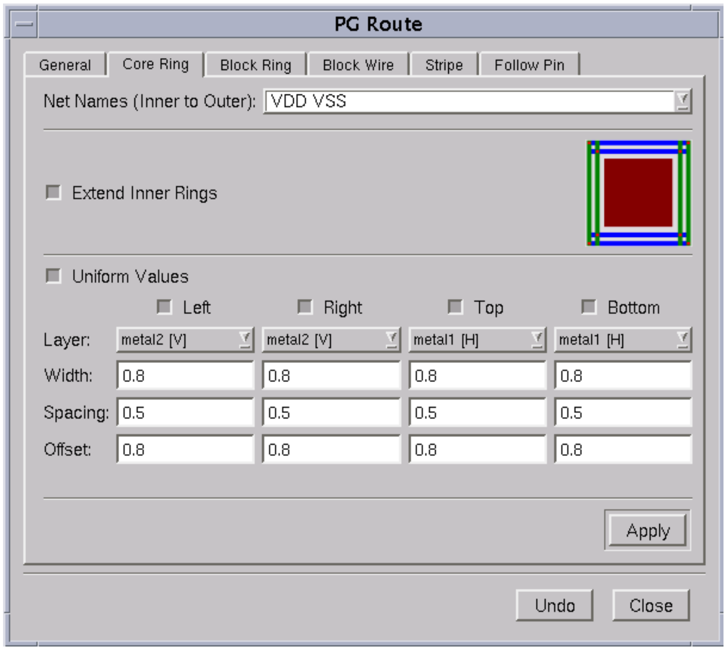
Global Rule:	default
Top Layer for Via Connection:	metal4 [V]
Bottom Layer for Via Connection:	metal1 [H]
Extend Search Range by:	0

☐ Allow Violations
☐ Show Violations

The PG route is designed in non-blocking mode which allows interactive Laker operations.

- The **Apply** button of an action tab executes its dedicated parameters with global options in the **General** tab.
- The **Undo** button reverses the previous action.
- The **Close** button closes the form.

2. Select the **Core Ring** tab.


PG Route

General | Core Ring | Block Ring | Block Wire | Stripe | Follow Pin

Net Names (Inner to Outer): VDD VSS

Extend Inner Rings

Uniform Values

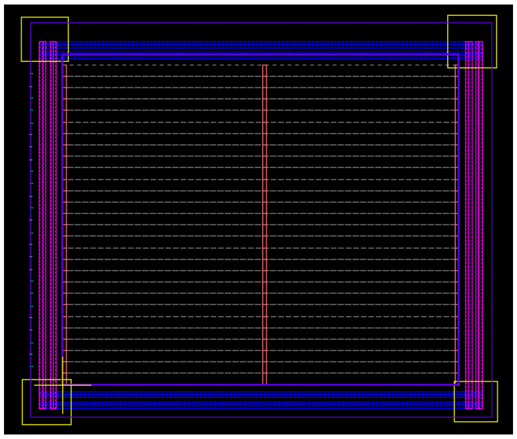
	Left	Right	Top	Bottom
Layer:	metal2 [V]	metal2 [V]	metal1 [H]	metal1 [H]
Width:	0.8	0.8	0.8	0.8
Spacing:	0.5	0.5	0.5	0.5
Offset:	0.8	0.8	0.8	0.8

3. Fill in power and ground net names in the **Core Ring** tab by entering *VDD VSS* under **PG Route → Core Ring → Net Names.**
4. Click **Uniform Values** for rapid input of the same width, spacing and offset values.
5. Default setting of sides is all enabled for **Left**, **Right**, **Top** and **Bottom**.
6. Fill in your planned **Layer**, **Width**, **Spacing** and **Offset** for each Left, Right, Top, Bottom edges around Row Area.

   Minimum spacing of fat metal is automatically checked and updated when a width is changed. After a width becomes smaller, it will not be restored back to the minimum value if the corresponding DRC spacing rule is satisfied.

   The definition of four sides also applies to the Row Area in a polygon shape.

7. Click **Apply** to realize core ring creation.
8. Click **Close** to close the form or keep it for the next stripe creation.



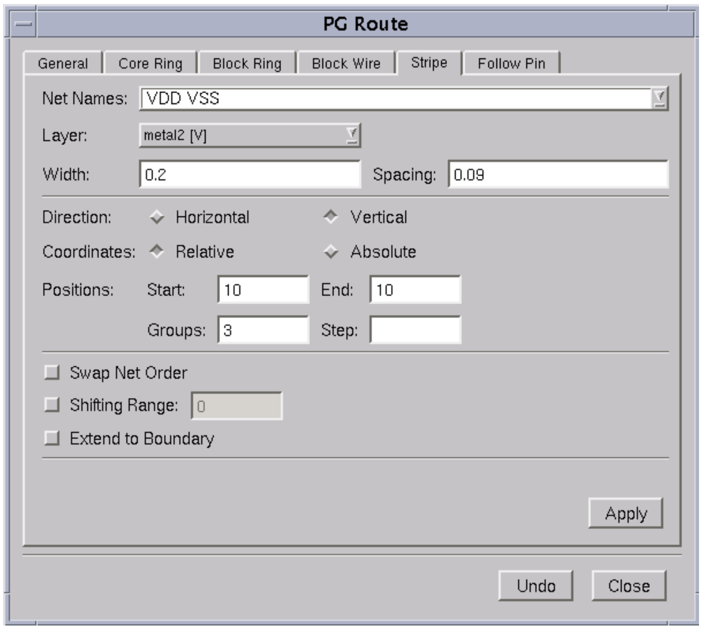
The current function of core ring creation does not honor previously created ring structure. Repeated core creation might result in DRC violations and be removed.

8.1.3 Create Stripes

Before creation of stripes, you have to plan the stripe structure, routing layer, width and offset spacing within the core area.

- Routing layers usually follow routing preferred direction to save routing resources. Horizontal stripes use horizontal routing layers. Vertical stripes use vertical routing layers.
- Decide **Net Names** by bundled group members or single member. Define width and spacing between members of **Net Names**.
- Define a **Coordinates** by Start and End forms valid region to create stripes. It can be **Relative** or **Absolute** mode.
- Define repeat patterns by combination of Start, End, Groups and Step.
  - (Start, End, Groups) mode automatically calculates desired Step value between two consecutive net groups.
  - (Start, End, Groups, Step) mode creates repeated patterns from left to right (bottom-to-top) by Step values. End value is the most right (top) boundary to filter out non-valid stripes.
  - Default End value is the most right (top) edge of Row Area.

1. Invoke the **Router → Digital Router → PG Route** command again if it is closed.
2. Select the **Stripe** tab.
3. Fill in your planned **Net Names**, **Direction**, **Coordinates**, **Start**, **End** and **Groups** of **Positions**.

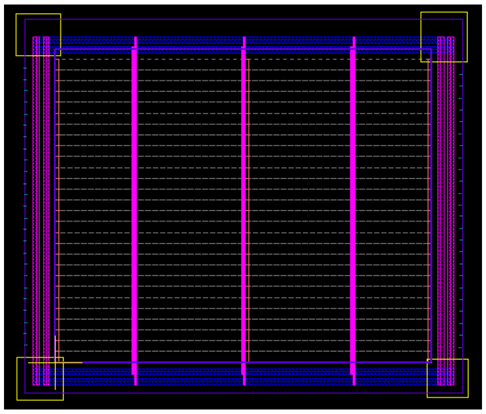

PG Route

General | Core Ring | Block Ring | Block Wire | Stripe | Follow Pin

Net Names: VDD VSS
Layer: metal2 [V]
Width: 0.2 Spacing: 0.09
Direction: Horizontal [✓ Vertical]
Coordinates: [✓ Relative] Absolute
Positions: Start: 10 End: 10
Groups: 3 Step:
☐ Swap Net Order
☐ Shifting Range: 0
☐ Extend to Boundary

Apply
Undo Close

4. Click **Apply** to realize stripes creation.



5. Click **Close** to close the form.

9 In-Placement

9.1 Lab-7A: Row Placement

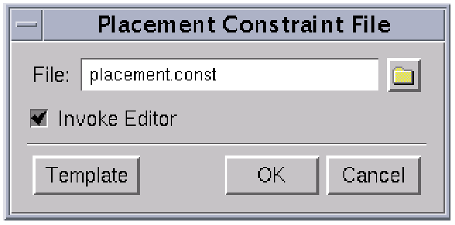
In this lab, you will learn how to create placement constraints and invoke row placement in a specified row area.

9.1.1 Preparation of Placement Constraint File

Row Placer provides rich placement constraints to control the placement result. It can support spare cell even distribution, net weight, group, placement effort, etc.

An initial placement constraint is provided and is a good start of study.

1. Invoke **Placer → Placement Constraint** command, type in constraint file name.



Placement Constraint File

File: placement.const

[✓] Invoke Editor

2. Click **Template** to generate a template of placement constraint.

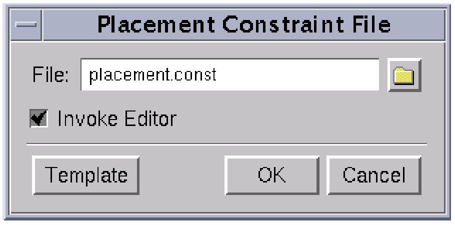
In this demo case, spare cells, NoFiller1 and Cell Index Aware flow will be enabled. A simple placement constraint is illustrated as follows.

```
##### Assign spare instances
.BeginSection Spare
*spare_*
.EndSection Spare

##### Cell-Index-aware placement
##### Use 'lakerCellIndexPlacementScore' to show the score of current
placement. #####
##### Cell index file format: #####
##### CellMasterName   SingleScore   AbutScore #####
```

```
.CellIndexFile OpenCellLibrary.cell_index

# No Filler1 flow
# FILL3 Cell is necessary for library preparation to avoid consecutive
filler1
.NoFiller1

# Ignore via instances from gds import flow
.IgnoreMaster VIAGEN*

#.PlacementEffort high
```

9.1.2 Assign Placement Constraint File

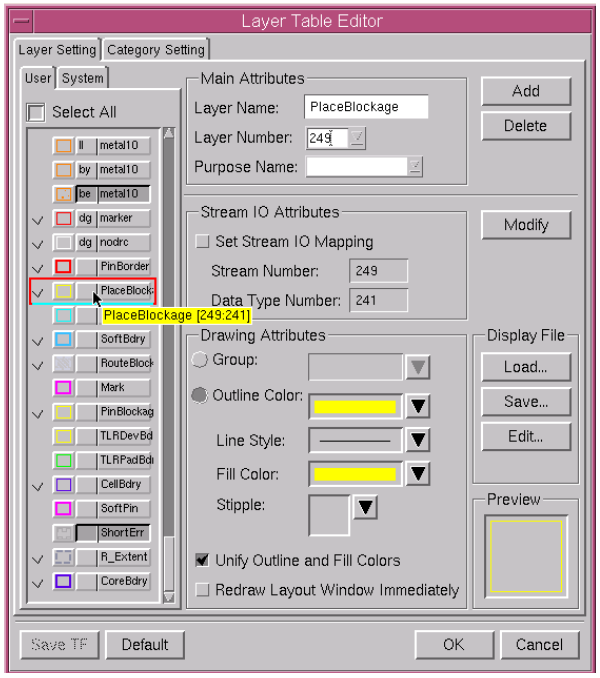
1. Invoke the **Placer → Placement Constraint** command, type in constraint file name.


Placement Constraint File
File: placement.const
[✓] Invoke Editor

2. Click **OK** to assign a placement constraint for placer related features.

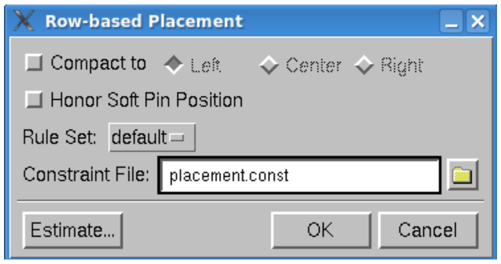
9.1.3 Create Placement Blockage

If you want to protect some regions to avoid cell placement in these regions, you can add the PlacementBlockage (249:241) layer in the floor plan layout.


Layer Table Editor
Main Attributes
Layer Name: PlaceBlockage
Layer Number: 249
Purpose Name:
Stream IO Attributes
[ ] Set Stream IO Mapping
Stream Number: 249
Data Type Number: 241
Drawing Attributes
[✓] Unify Outline and Fill Colors
[ ] Redraw Layout Window Immediately
PlaceBlockage [249:241]

9.1.4 Row-based Placement in SDL

1. Click the **Select All** icon in the design hierarchy browser pane to select all objects because the Laker system can support partial selection for implementation.
2. Invoke the **Design Brower → Placement → Row Placer** command.
3. Select the **Rule Set** according to the original routing layer plan.
   Available routing layers have an impact on congestion driven placement results. You can confirm in the **Router → Net Router → Rule** tab if necessary.
4. Fill in the **Constraint File** field with the file name of a placement constraint.
5. Click one Row area and apply **Design Brower → Placement → Row Placer → Estimate** to get the utilization rate calculated by placer engine.


Row-based Placement

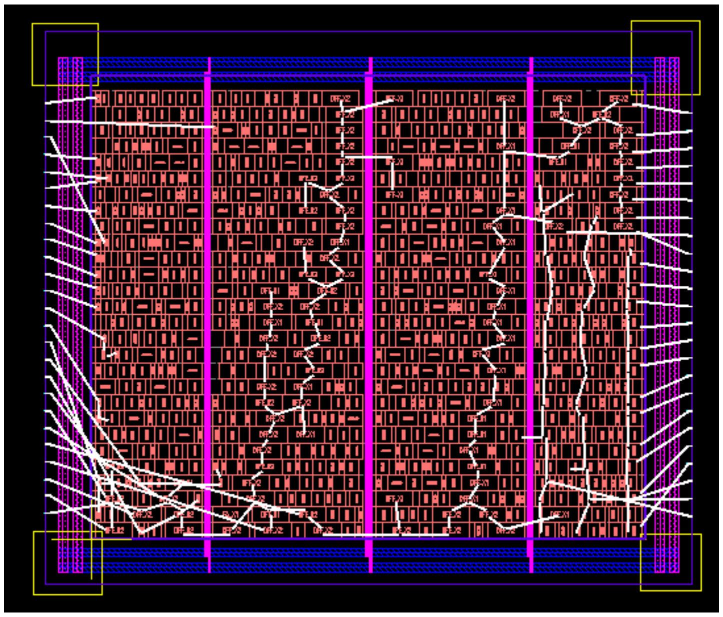
☐ Compact to ◆ Left ◇ Center ◇ Right

☐ Honor Soft Pin Position

Rule Set: default

Constraint File: placement.const

6. Click one Row area and apply **Design Brower → Placement → Row Placer → OK** to finish row placement task.



It will take a longer time to explore a possible smaller size by enabling the option of **Compact to**. This option is useful for getting a compact design in a sparse initial floor plan and adjusting the floor plan by editing features.

10 Post-Placement

10.1 Lab-8A: Post-Placement

In this lab, you will learn how to perform pin optimization, placement check, PG follow pin, core filler insertion.

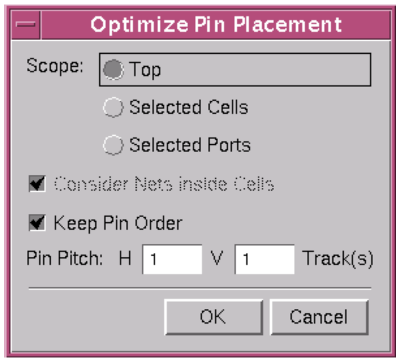
10.1.1 Pin Optimization

This is an optional step and can be used if you want to optimize evenly distributed pins based on the new cell placement.

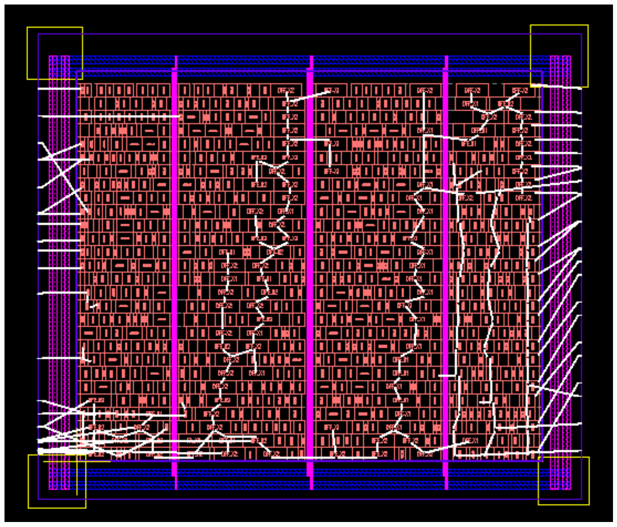
- Redo auto pin assignment again by removing edge distribution. The pin order and edge constraint are kept while pin placement is not evenly distributed.
- Redo auto pin assignment again without any constraints. The pin order and edge are decided by wire length minimization.
- Redo row placement by increasing net weight cost of I/O nets in placement constraint file.

```
#.NetWeight netName netWeight_int_1_to_50
```

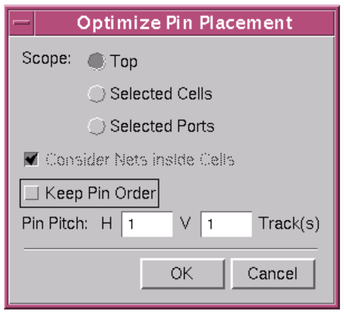
1. Use the **Placer → Pin Placer → Optimize Pin Placement** command for pin optimization.
   a. Set **Scope** as **Top** for top pins.
   b. Enable **Keep Pin Order** without changing current pin order


Optimize Pin Placement
Scope: [● Top] ○ Selected Cells ○ Selected Ports
[✓] Consider Nets inside Cells
[✓] Keep Pin Order
Pin Pitch: H 1 V 1 Track(s)

2. After this step, finish the task of pin optimization while keeping edge and order.



3. Click the **Undo** button to check the difference of results without keeping pin order.
4. Use the **Placer → Pin Placer → Optimize Pin Placement** command for pin optimization.
    a. Set **Scope** as **Top** for top pins.
    b. Disable **Keep Pin Order** without changing the current pin order.



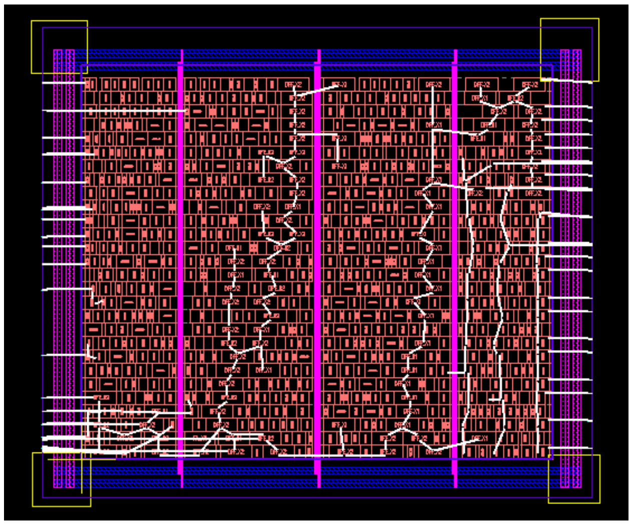
Optimize Pin Placement

Scope: [● Top] ○ Selected Cells ○ Selected Ports

☑ Consider Nets inside Cells

☐ Keep Pin Order

Pin Pitch: H 1 V 1 Track(s)

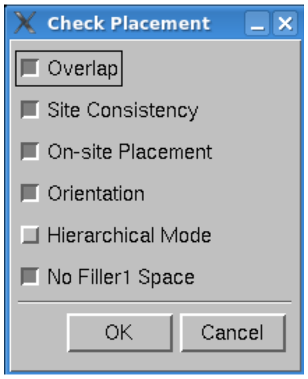


5. Click **OK** to perform pin optimization by only keeping edge constraint.

10.1.2 Check Placement

A placement check utility is to check cell placement in a row area, including cell overlap, legalization, filler1 gap, etc. It is useful especially for manual cell placement.

1. Invoke the **Placer → Check Placement** command.
2. In the *Check Placement* form, enable the **NO Filler1 Space** option.
3. Click **OK** to check cell placement in a row area.

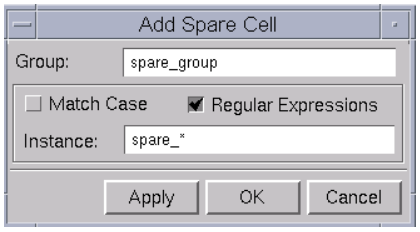


You can invoke **Verify → View Error** to bring up an error view for review if a placement error is found.

10.1.3 Check Spare Cell Placement

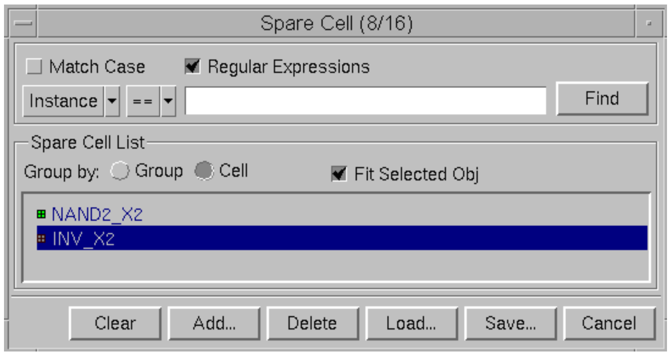
Silicon ECO spare cells are automatic even spread during Row Placement procedure. These spare cells can be highlighted to make sure correct spare cell constraint is set.

1. Invoke **Query → Spare Cell → Add** to define spare cell groups.
2. In the *Add Spare Cell* form, do the following:
   a. Set **Group** to *spare_group*
   b. Set **Instance** to *spare_**



Group: spare_group
Instance: spare_*

3. In the *Spare Cell* form, click the spare cell name to highlight cell placement in a row area.


Spare Cell (8/16)

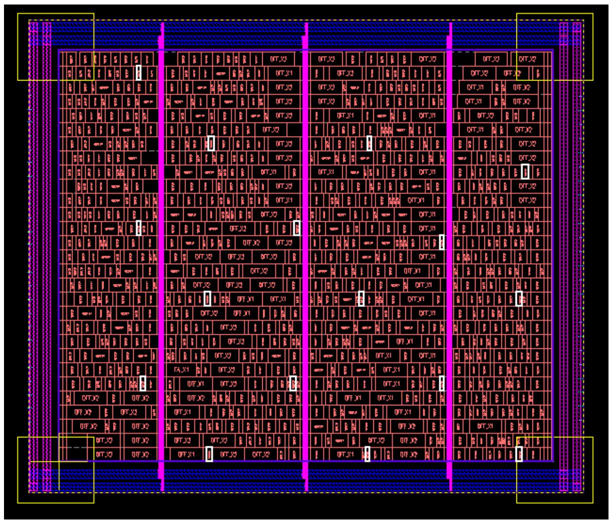
☐ Match Case ☑ Regular Expressions

Instance ▾ == ▾ Find

Spare Cell List

Group by: ○ Group ● Cell ☑ Fit Selected Obj

NAND2_X2
INV_X2

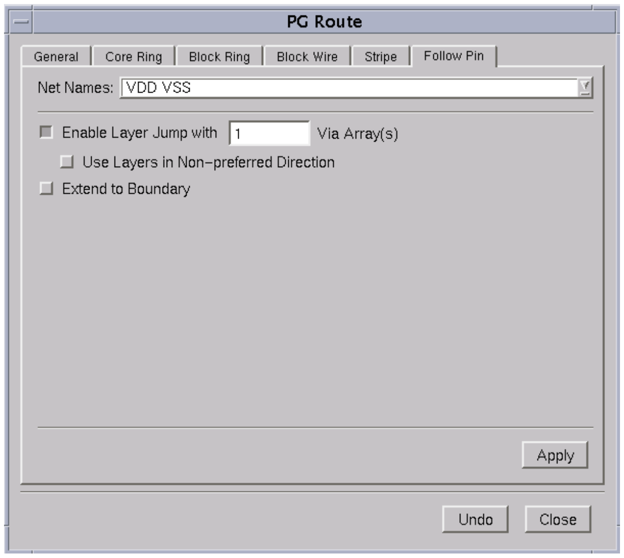


4. Click **Query → Spare Cell → Clear** or **F8** bind key to clear highlight of selected spare cells.
5. Click **Query → Spare Cell → Cancel** to close the form.

10.1.4 Connect PG Rails of Standard Cells

PG rails routing has an impact on routing resources and DRC checking in a limited layer design. It is recommended to finish PG rails before starting signal routing.

1. Invoke **Router → Digital Router → PG Route** command.
2. Select the **Follow Pin** tab.
3. Fill in **Net Names** as *VDD VSS*.



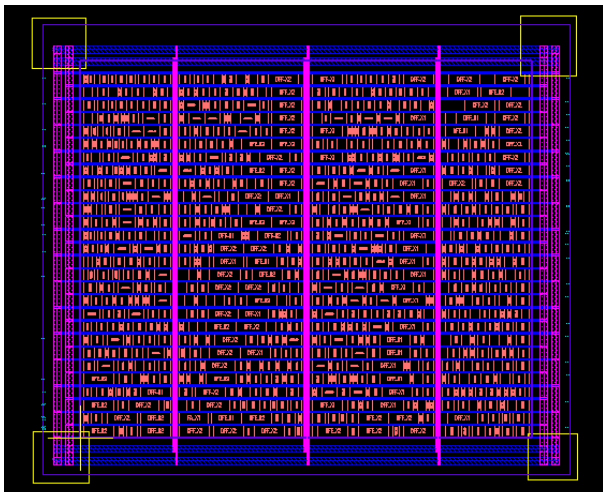
Net Names: VDD VSS

☐ Enable Layer Jump with 1 Via Array(s)

☐ Use Layers in Non-preferred Direction

☐ Extend to Boundary

4. Click **Apply** to realize stripes creation.



5. Click **Close** to close the form.

If wire extension is blocked by end cap cells or tap cells, connect power and ground ports to power and ground nets with the **Router → Digital Router → Assign Instance Port to Net** command.

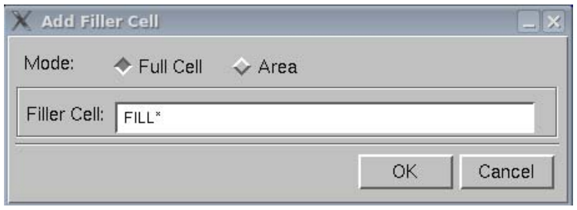
10.1.5 Add Core Fillers

Core filler insertion can be done before routing or after routing. The benefit of post-route insertion is to have better performance and database size without lots of core filler cells in an ASIC chip design. If the metal structure of core filler design is not simple, pre-route insertion is recommended to avoid potential DRC violations in advance.

A FILL3 cell is necessary in library preparation to avoid consecutive filler1 cells.
One FILLCELL_X3 will be created for this tutorial.

1. Invoke the **Placer → Add Filler Cell** command.

2. In the *Add Filler Cell* form, set filler cell names by specifying **Filler Cell → *FILLER***.*



Filler Cell: FILL*

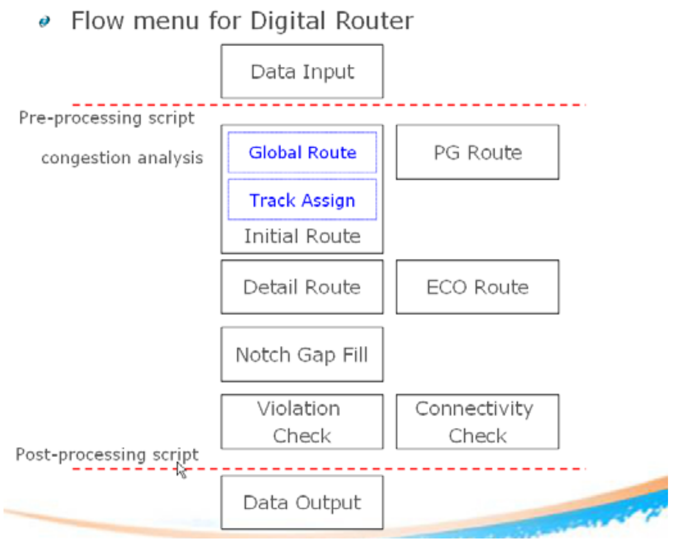
3. Click **OK** to finish core fillers insertion in a row area.

11 In-Route

11.1 Introduction

The Custom Digital router flow is a series of routing kernels executed in a pre-defined sequence for global router, track assignment, detail route, violation check, violation fix, and notch gap filling.

- Auto Route flow is recommended to serve most of typical designs.
- Primitive commands provide flexibility of routing procedure customization to fulfill different routing requirements. Usually, they are written in a script and executed in a batch.

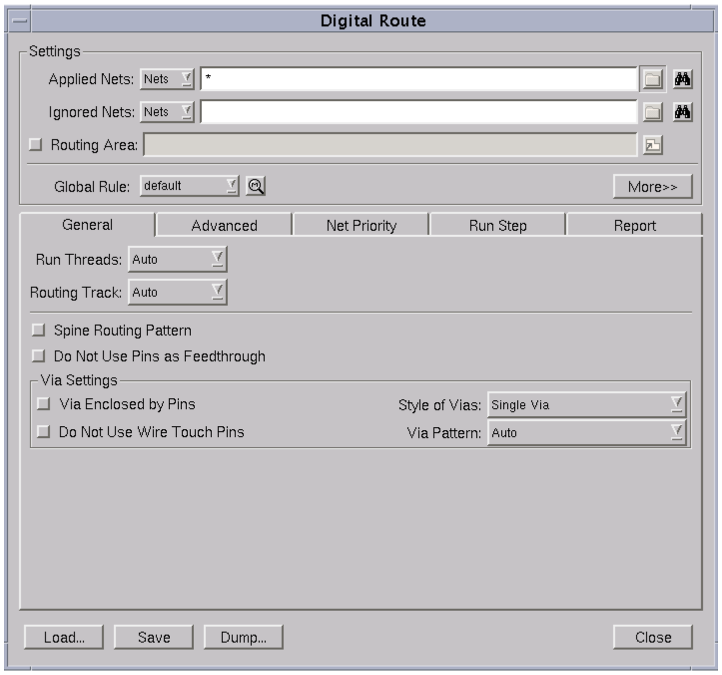


11.2 Lab-9A: Auto Route

In this lab, you will learn how to easily finish routing most typical designs, analyze quality of results.

11.2.1 Auto Route

1. Invoke the **Router→ Digital Router →Digital Route** command.
2. In the *Digital Router* form, set up global options under the **General** tab.
3. Click the **Global Rule** icon to confirm available routing layers from metal1 to metal4



Digital Route

Settings

Applied Nets: Nets *

Ignored Nets: Nets

☐ Routing Area:

Global Rule: default

Run Threads: Auto

Routing Track: Auto

☐ Spine Routing Pattern

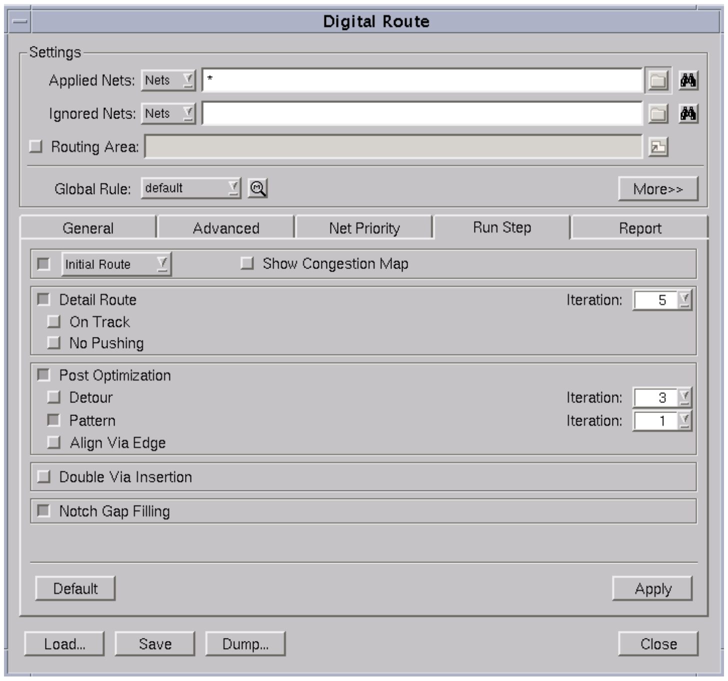
☐ Do Not Use Pins as Feedthrough

Via Settings

☐ Via Enclosed by Pins — Style of Vias: Single Via

☐ Do Not Use Wire Touch Pins — Via Pattern: Auto

4. On the *Digital Router* form, select the **Run Step** tab.
   a. Make sure **Initial Route**, **Detail Route**, **Post Optimization** and **Notch Gap Fill** procedures are enabled.
   b. Enable **Pattern** option of **Post Optimization**. Set **iteration** to 1.



Digital Route

Settings

Applied Nets: Nets *

Ignored Nets: Nets

☐ Routing Area:

Global Rule: default

☑ Initial Route — ☐ Show Congestion Map

☑ Detail Route — Iteration: 5

☐ On Track

☐ No Pushing

☑ Post Optimization

☐ Detour — Iteration: 3

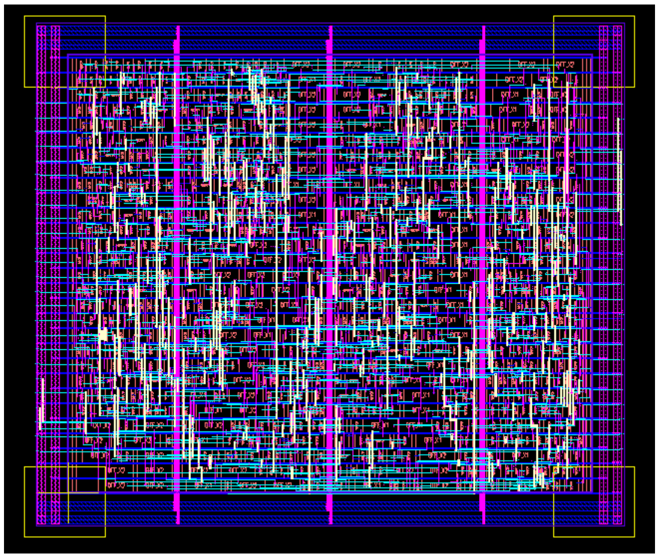
☑ Pattern — Iteration: 1

☐ Align Via Edge

☐ Double Via Insertion

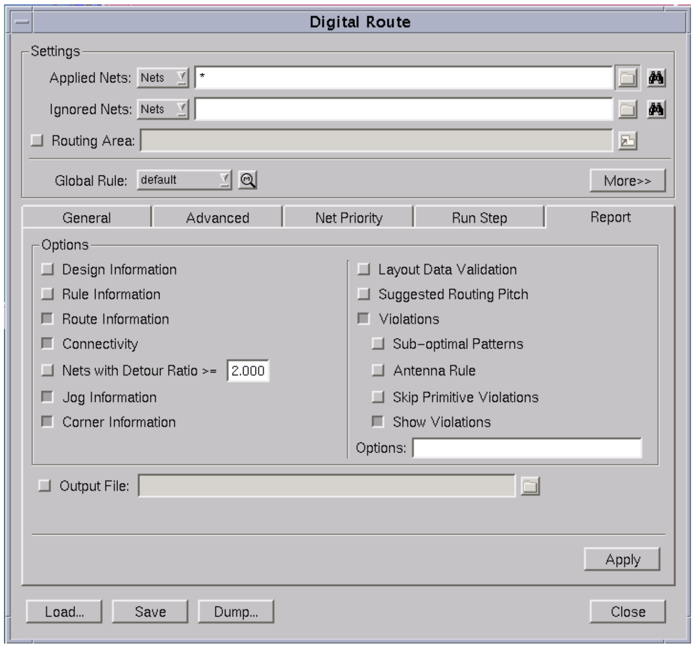
☑ Notch Gap Filling

5. Click **Apply** to complete auto routing tasks.



11.2.2 Routing Quality of Result

1. Switch to the **Router → Digital Router → Digital Route → Report** tab for reporting several routing quality results.
    a. Enable **Violations** and **Show Violations** options to bring up the Error Viewer if any DRC violations are found.
    b. Enable **Connectivity** option to check routing connectivity.
    c. Enable **Route Information** to report wire and via statistics. Double cut rate per layer can be reported this way.
    d. Enable **Jog information** to get jog counts.
    e. Enable **Corner information** to get checksum values per layer. It is a useful index to verify frozen layer checksum before and after automatic or manual ECO routing.



Digital Route

Settings
Applied Nets: Nets *
Ignored Nets: Nets
☐ Routing Area:
Global Rule: default
More>>

General | Advanced | Net Priority | Run Step | Report

Options
☐ Design Information
☐ Rule Information
☑ Route Information
☑ Connectivity
☐ Nets with Detour Ratio >= 2.000
☑ Jog Information
☑ Corner Information
☐ Layout Data Validation
☐ Suggested Routing Pitch
☑ Violations
☐ Sub−optimal Patterns
☐ Antenna Rule
☐ Skip Primitive Violations
☑ Show Violations
Options:

☐ Output File:

Apply

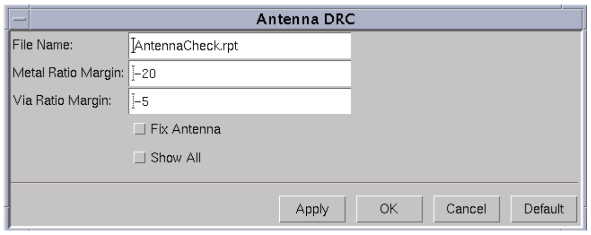
Load... Save Dump... Close

2. Click **Apply** to complete router verification tasks.
3. Select the **Router → Digital Router → Digital Route → Close** command to close the Digital Route form.

11.2.3 Antenna Check and Fix

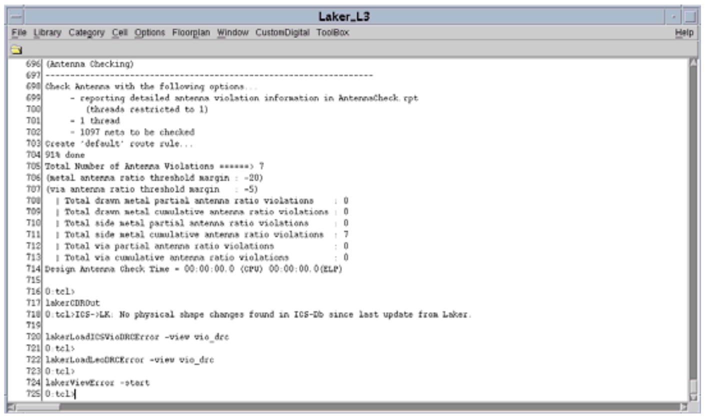
Antenna rules and pin information are necessary to check and fix antenna violations. Refer to sections 3.2and chapter 4.2.4for more detailed information.

1. Invoke **CustomDigital → Antenna DRC** script for reporting antenna violations.
   a. Set **File Name** as *AntennaCheck.rpt*
   b. Set **Metal Ratio Margin** to *-20*
   c. Set **VIA Ratio Margin** to *-5*
   Usually, a small margin of antenna rules is set to eliminate iterations due to minor mismatch of calculations between routers and signoff tools.



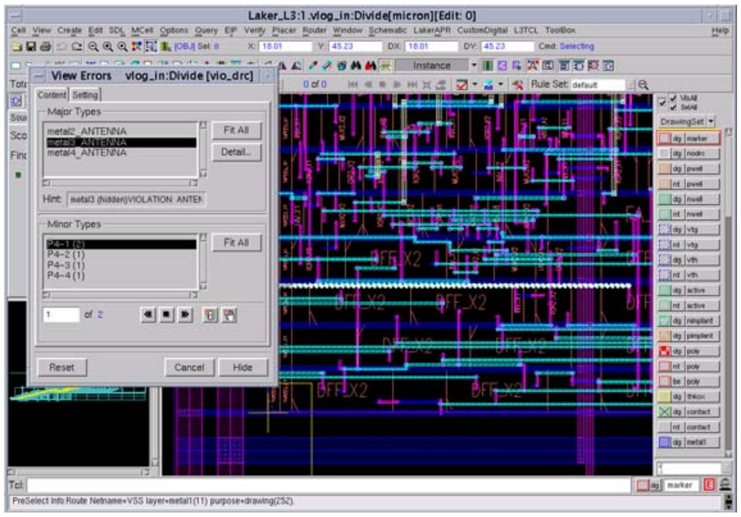
File Name:	AntennaCheck.rpt
Metal Ratio Margin:	-20
Via Ratio Margin:	-5

   d. Click **Apply** to check antenna rules.

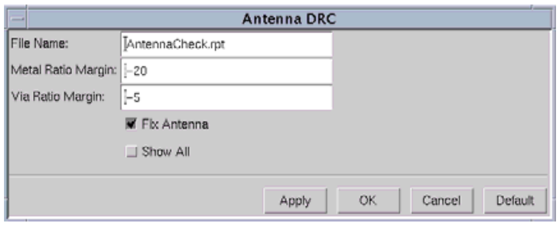

Laker_L3
File Library Category Cell Options Floorplan Window CustomDigital ToolBox Help

```
(Antenna Checking)
-----------------------------------------------------------------
Check Antenna with the following options...
      - reporting detailed antenna violation information in AntennaCheck.rpt
         (threads restricted to 1)
      - 1 thread
      - 1097 nets to be checked
Create 'default' route rule...
91% done
Total Number of Antenna Violations ======> 7
(metal antenna ratio threshold margin : -20)
(via antenna ratio threshold margin   : -5)
  | Total drawn metal partial antenna ratio violations    : 0
  | Total drawn metal cumulative antenna ratio violations : 0
  | Total side metal partial antenna ratio violations     : 0
  | Total side metal cumulative antenna ratio violations  : 7
  | Total via partial antenna ratio violations            : 0
  | Total via cumulative antenna ratio violations         : 0
Design Antenna Check Time = 00:00:00.0 (CPU) 00:00:00.0(ELP)

0:tcl>
lakerCDROut
0:tcl>ICS->LK: No physical shape changes found in ICS-Db since last update from Laker.

lakerLoadICSVioDRCError -view vio_drc
0:tcl>
lakerLoadLeoDRCError -view vio_drc
0:tcl>
lakerViewError -start
0:tcl>
```

Several side metal cumulative antenna violations are found. An error browser is automatically invoked to review these violations.



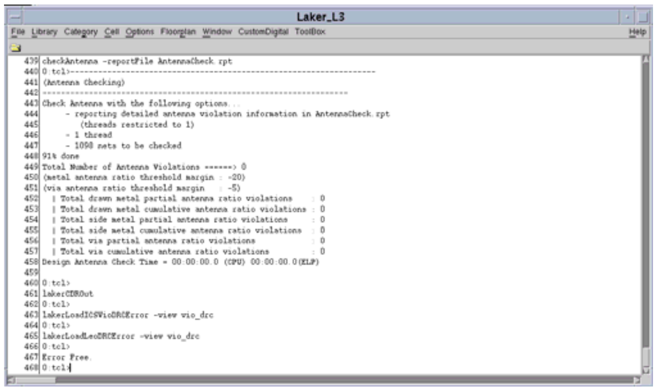
e. Enable the **Fix Antenna** option to fix existing antenna violations.



Antenna DRC

File Name: AntennaCheck.rpt
Metal Ratio Margin: -20
Via Ratio Margin: -5
☑ Fix Antenna
☐ Show All

f. Click **OK** to fix antenna rules.


Laker_L3
File Library Category Cell Options Floorplan Window CustomDigital ToolBox Help

```
checkAntenna -reportFile AntennaCheck.rpt
0:tcl>-----------------------------------------------------------------
(Antenna Checking)
=================================================================
Check Antenna with the following options...
      - reporting detailed antenna violation information in AntennaCheck.rpt
         (threads restricted to 1)
      - 1 thread
      - 1098 nets to be checked
91% done
Total Number of Antenna Violations ======> 0
(metal antenna ratio threshold margin : -20)
(via antenna ratio threshold margin   : -5)
   | Total drawn metal partial antenna ratio violations    : 0
   | Total drawn metal cumulative antenna ratio violations : 0
   | Total side metal partial antenna ratio violations     : 0
   | Total side metal cumulative antenna ratio violations  : 0
   | Total via partial antenna ratio violations            : 0
   | Total via cumulative antenna ratio violations         : 0
Design Antenna Check Time = 00:00:00.0 (CPU) 00:00:00.0(ELP)

0:tcl>
lakerCDBOut
0:tcl>
lakerLoadICSVioDRCError -view vio_drc
0:tcl>
lakerLoadLeoDRCError -view vio_drc
0:tcl>
Error Free.
0:tcl>
```

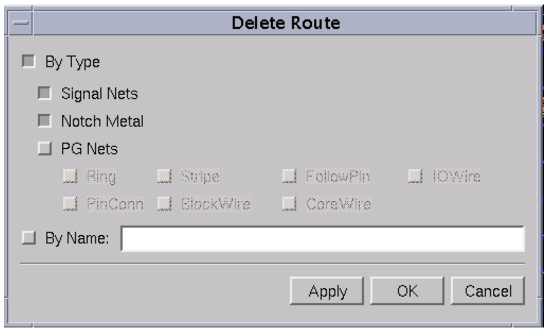
Antenna violations are fixed now after antenna fixing.

11.3 Lab-9B: Congestion Analysis

In this lab, you will learn how to perform congestion analysis and review congestion map.

11.3.1 Remove Signal Routing (optional)

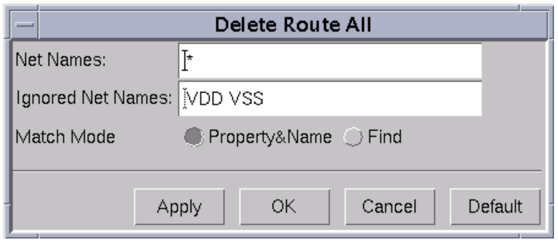
Delete existing signal routing by invoking the **Router → Digital Router → Delete Route** command if necessary.


Delete Route

- By Type
  - Signal Nets
  - Notch Metal
  - PG Nets
    - Ring Stripe FollowPin IOWire
    - PinConn BlockWire CoreWire
- By Name:

Apply OK Cancel

The **CustomDigital → Delete Route All By → Name** script can be used to delete existing nets which are not created by the router. It is a fast script to remove all routing signals by traversing property or names.


Delete Route All

Net Names: *

Ignored Net Names: VDD VSS

Match Mode [✓ Property&Name] Find

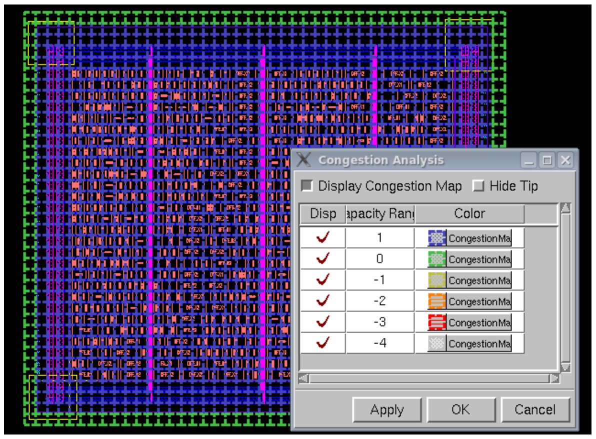
Apply OK Cancel Default

11.3.2 Congestion Analysis and Map

Post-placement congestion map can be rapidly analyzed by global route kernel. This feature is useful for manual placement of high performance designs. You can also review the congestion map whenever a highly congested design is reported after automatic row placement.

1. Invoke **Router → Congestion Analysis** command.
2. Check the available routing layer settings of **Rule Set** with the **Router → Digital Router → Rule Setting** command.
3. Enable **Display Congestion Map** option.
4. Click **Apply** to analyze global route congestion and display the color map on layout window.

   The congestion map is displayed in different color by capacity range definition. Negative values mean the estimated routing is overflowed based on the given floor plan and routing setting.


Congestion Analysis

Display Congestion Map Hide Tip

Disp	apacity Ran	Color
✓	1	CongestionMa
✓	0	CongestionMa
✓	-1	CongestionMa
✓	-2	CongestionMa
✓	-3	CongestionMa
✓	-4	CongestionMa

Apply OK Cancel

5. After reviewing the congestion region, disable **Congestion Analysis → Display Congestion Map**.
6. Select the **Congestion Analysis → OK** button to close the congestion map.

Here are a few methods to solve the issue of routing congestion.

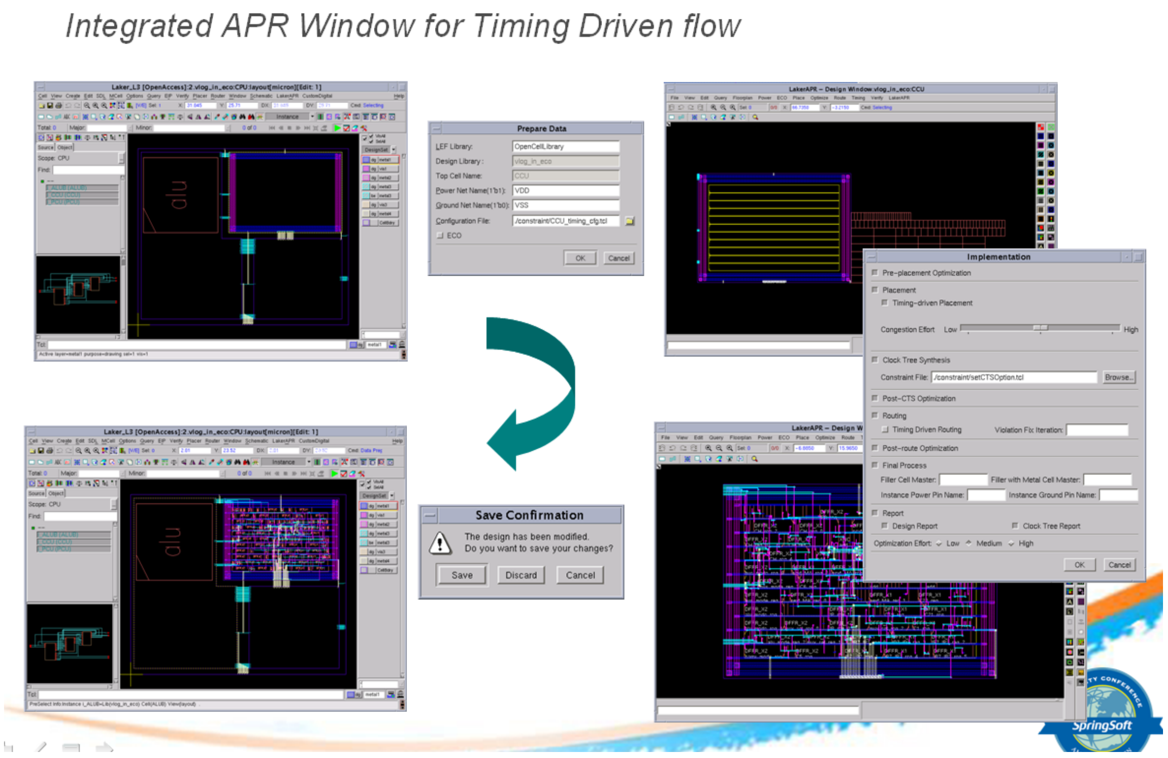
- Add placement blockage to relax hot spot of local congestion.
- Adjust floor plan aspect ratio and size
- Adopt fine power stripes or hierarchical mesh in power plan

12 Timing Driven Flow

12.1 Introduction

The Laker timing driven CDPR window is a dedicated working window for timing driven related features, including parasitic extraction, delay calculation, static timing analysis, clock tree synthesis, timing optimization, and interactive ECO.

- Add a configuration file to define RC table, timing library, timing constraints and timing analysis views.
- A recommended implementation flow is provided to serve most of typical designs.
- Timing driven CDPR In-memory database can be synchronized back to Laker database for milestone check points.
- Individual features provide flexibility of procedure customization to fulfill different requirements.


Integrated APR Window for Timing Driven flow

Prepare Data

LEF Library: OpenCellLibrary
Design Library : vlog_in_eco
Top Cell Name: CCU
Power Net Name(1'b1): VDD
Ground Net Name(1'b0): VSS
Configuration File: /constraint/CCU_timing_cfg.tcl
ECO

Implementation

Pre-placement Optimization
Placement
Timing-driven Placement
Congestion Effort Low – High
Clock Tree Synthesis
Constraint File: /constraint/setCTSOption.tcl
Post-CTS Optimization
Routing
Timing Driven Routing Violation Fix Iteration:
Post-route Optimization
Final Process
Filler Cell Master: Filler with Metal Cell Master:
Instance Power Pin Name: Instance Ground Pin Name:
Report
Design Report Clock Tree Report
Optimization Effort: Low [✓ Medium] High

Save Confirmation

The design has been modified.
Do you want to save your changes?

12.2 Lab-10A: Configuration Setup

In this lab, you will learn how to set up timing configuration for timing driven CDPR window.

12.2.1 Prepare Data

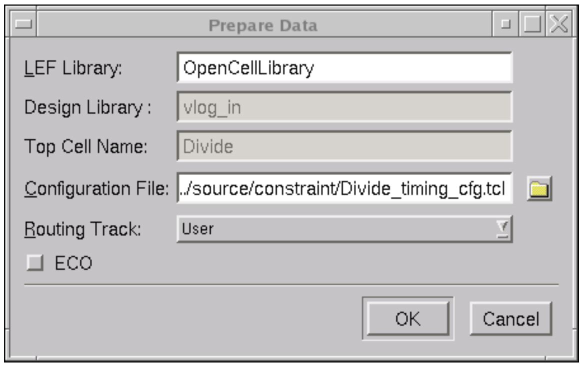
A configuration file is used to define the RC table, timing library, timing constraint and timing analysis views.

If you have only need net based optimization without RC table, timing library and constraint files, you can create an empty configuration file.

Here is an example of configuration file named *Divide_timing_cfg.tcl.* Two analysis views are defined for slow and fast timing libraries.

```
set rcTable ../source/technology/OpenCellLibrary_x2d.rcTbl
set normalSdc ../source/constraint/Divide.sdc
set fastLib { ../source/liberty/NangateOpenCellLibrary_fast_conditional_nldm.lib }
set slowLib { ../source/liberty/NangateOpenCellLibrary_slow_conditional_nldm.lib }

readRCTable -rcTable "$rcTable $rcTable" -rcGroup { slowRC fastRC }
initializeRc -rcGroup { slowRC fastRC }

readLib $fastLib
readLib $slowLib

createLibGroup -libGroup worstGroup $slowLib
createLibGroup -libGroup bestGroup $fastLib

readSdc -sdcGroup normalGrp  $normalSdc
addView normalSlowView -rcGroup slowRC -libGroup worstGroup -sdc normalGrp
addView normalFastView -rcGroup fastRC -libGroup bestGroup -sdc normalGrp

# Check Active Timing View
setActiveView -enable -all
listViews
```

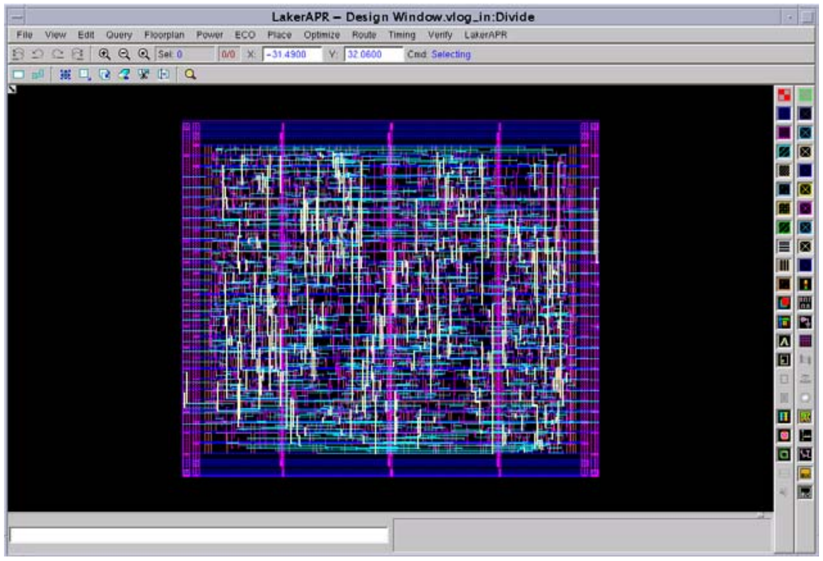
1. Invoke the **LakerAPR → Prepare Data** command.
2. Set **Configuration File** as ../source/constraint/*Divide_timing_cfg.tcl*
3. Set **Routing Track** to *User*
4. Disable **ECO** option.



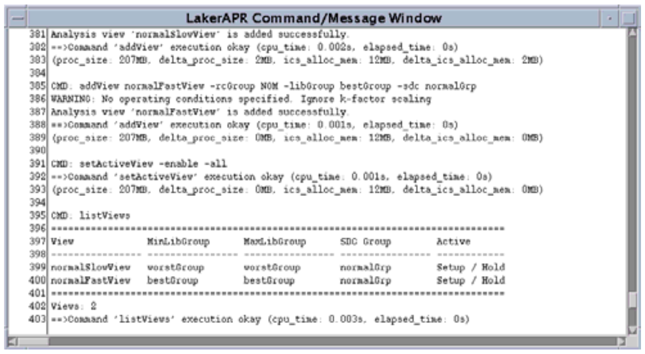
LEF Library: OpenCellLibrary
Design Library : vlog_in
Top Cell Name: Divide
Configuration File: ../source/constraint/Divide_timing_cfg.tcl
Routing Track: User
☐ ECO

5. Click **OK** to start data preparation from schematic and layout views.

A new timing driven CDPR window is opened for timing driven related features. Laker windows are locked to prevent database inconsistence issue.


LakerAPR – Design Window.vlog_in:Divide
File View Edit Query Floorplan Power ECO Place Optimize Route Timing Verify LakerAPR
Sel: 0
0/0
X: -31.4900
Y: 32.0600
Cmd: Selecting

A new timing driven CDPR Command and Message window is also opened for timing driven related messages and commands.



```
Analysis view 'normalSlowView' is added successfully.
==>Command 'addView' execution okay (cpu_time: 0.002s, elapsed_time: 0s)
(proc_size: 207MB, delta_proc_size: 2MB, ics_alloc_mem: 12MB, delta_ics_alloc_mem: 2MB)

CMD: addView normalFastView -rcGroup NOM -libGroup bestGroup -sdc normalGrp
WARNING: No operating conditions specified. Ignore k-factor scaling
Analysis view 'normalFastView' is added successfully.
==>Command 'addView' execution okay (cpu_time: 0.001s, elapsed_time: 0s)
(proc_size: 207MB, delta_proc_size: 0MB, ics_alloc_mem: 12MB, delta_ics_alloc_mem: 0MB)

CMD: setActiveView -enable -all
==>Command 'setActiveView' execution okay (cpu_time: 0.001s, elapsed_time: 0s)
(proc_size: 207MB, delta_proc_size: 0MB, ics_alloc_mem: 12MB, delta_ics_alloc_mem: 0MB)

CMD: listViews
==========================================================================================
View             MinLibGroup      MaxLibGroup      SDC Group        Active
---------------- ---------------- ---------------- ---------------- ------------
normalSlowView   worstGroup       worstGroup       normalGrp        Setup / Hold
normalFastView   bestGroup        bestGroup        normalGrp        Setup / Hold
==========================================================================================
Views: 2
==>Command 'listViews' execution okay (cpu_time: 0.003s, elapsed_time: 0s)
```

12.3 Lab-10B: Net Optimization

High fan-out nets and long distances are common in top-down RTL logic synthesis flow or schematic entry from scratch. For those designs without an RC table, a liberty timing library, or SDC timing constraints, a net optimization flow is provided based on logic information only in the Timing Driven-CDPR Window.

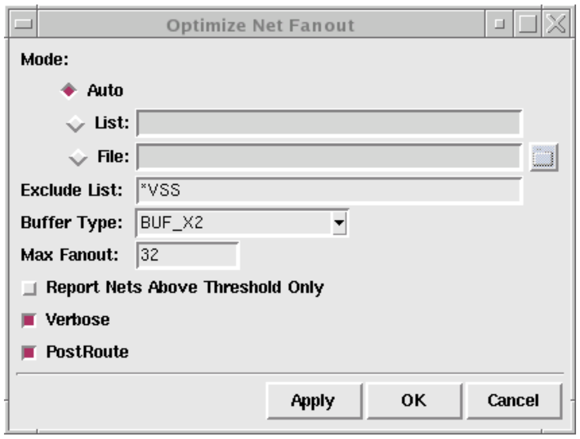
If you have a full set of timing libraries and constraints, the timing driven flow in the next section is recommended.

12.3.1 Fan out Based Optimization

1. Invoke the customized script from the CustomDigital → Optimize Net Fanout command and set up the form as follows:
   a. Select the **Mode** as **Auto**
   b. Set the **Exclude List** field to **VSS*
   c. Set the **Buffer Type** field to *BUF_X2*
   d. Set the **Max Fanout** field to *32*
   e. Disable the **Report Nets Above Threshold Only** option.

You can enable this option to report a netlist first and set an active netlist with **Mode** of **List** or **File**.

f. Enable the **Verbose** option for more messages.
g. Enable the **PostRoute** option.



Mode:
[✓ Auto]
List:
File:
Exclude List: *VSS
Buffer Type: BUF_X2
Max Fanout: 32
Report Nets Above Threshold Only
[✓ Verbose]
[✓ PostRoute]

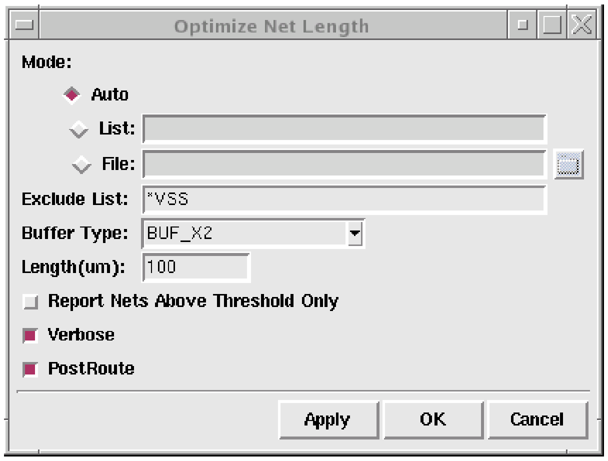
2. Click the **OK** button.

   You can find *HFN* instances for inserted buffers.

12.3.2 Length Based Optimization

1. Invoke the customized script from the **CustomDigital → Optimize Net Length** command and set up the form as follows.
   a. Select the **Mode** as *Auto*
   b. Set the **Exclude List** field to **VSS*
   c. Set the **Buffer Type** field to BUFX2
   d. Set the **Length(um)** field to *100*
   e. Disable the **Report Nets Above Threshold Only** option.
   You can enable this option to report a net list first and set an active net list with **Mode** of **List** or **File**.

   f. Enable the **Verbose** option for more messages.
   g. Enable the **PostRoute** option.



Mode:
[✓ Auto]
List:
File:
Exclude List: *VSS
Buffer Type: BUF_X2
Length(um): 100
Report Nets Above Threshold Only
[✓ Verbose]
[✓ PostRoute]

2. Click the **OK** button.

   You can find *LEN* instances for inserted buffers.

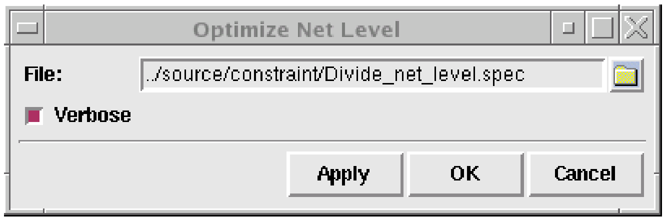
The naming convention of inserted buffers is **HFN** for *optimizeNetFanout* and **LEN** for *optimizeNetLength.*

12.3.3 Level Based Optimization

Level based net optimization is a bottom up approach of high fan-out net synthesis. The level specification format is defined as follows.

```
.NetName <net_name>
.Level <number> MaxFanout <number> BufType <buffer_name>

# Level based optimization
.NetName CLK_dly
.Level 1 MaxFanout 8  BufType BUF_X4
.Level 2 MaxFanout 8  BufType BUF_X4
.Level 3 MaxFanout 16 BufType BUF_X2
```

1. Invoke the customized script from the **CustomDigital → Optimize Net Level** command and set up the form as follows:
   a. Enter ../source/constraint/Divide_net_level.spec in the **File** text field.
   b. Enable the **Verbose** option.



File: ../source/constraint/Divide_net_level.spec
Verbose ✓

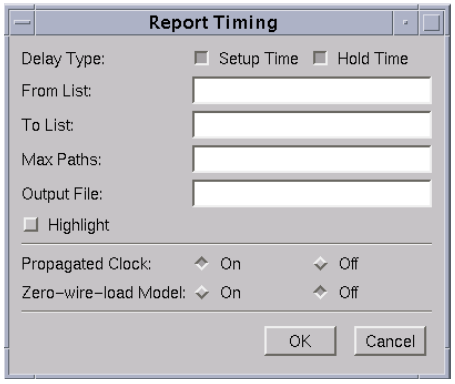
2. Click the **OK** button.

12.4 Lab-10C: Timing Driven CDPR Window

12.4.1 Report Timing Path and Constraint

Multiple mode multiple view (MMMV) timing analysis is supported. Timing violations can be analyzed to clarify false paths or multiple cycle paths before timing optimization.

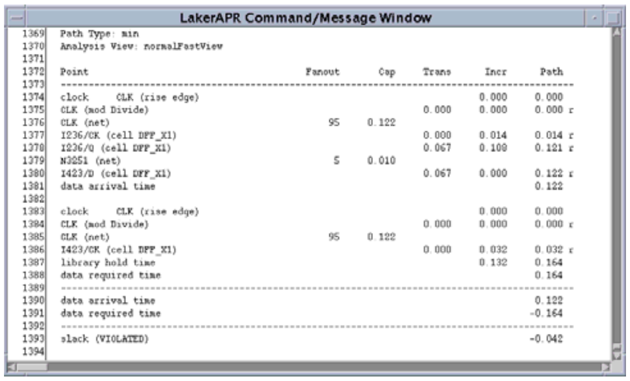
1. Invoke the **Timing → Report Timing** command.
2. Enable the **Setup time** and **Hold time** of **Delay Type** options.



Delay Type: [✓ Setup Time] [✓ Hold Time]
From List:
To List:
Max Paths:
Output File:
Highlight
Propagated Clock: [✓ On] Off
Zero-wire-load Model: On [✓ Off]

Delete existing signal routing by invoking the **Router → Digital Router → Delete Route** command if necessary.

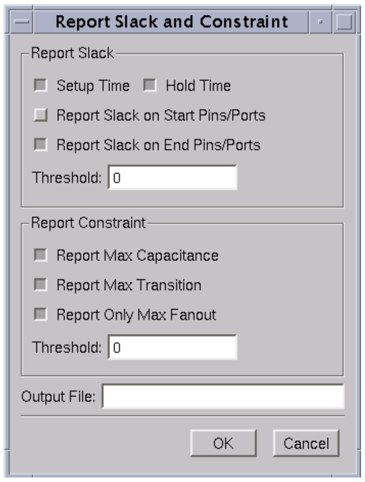
3. Click **OK** to analyze all active timing views.
   Both timing paths of setup time and hold time has violations by previous non-timing driven flow.



```
Path Type: min
Analysis View: normalFastView

Point                                   Fanout    Cap     Trans    Incr     Path
--------------------------------------------------------------------------------
clock     CLK (rise edge)                                          0.000    0.000
CLK (mod Divide)                                          0.000    0.000    0.000 r
CLK (net)                                 95     0.122
I236/CK (cell DFF_X1)                                     0.000    0.014    0.014 r
I236/Q (cell DFF_X1)                                      0.067    0.108    0.121 r
N3251 (net)                                5     0.010
I423/D (cell DFF_X1)                                      0.067    0.000    0.122 r
data arrival time                                                           0.122

clock     CLK (rise edge)                                          0.000    0.000
CLK (mod Divide)                                          0.000    0.000    0.000 r
CLK (net)                                 95     0.122
I423/CK (cell DFF_X1)                                     0.000    0.032    0.032 r
library hold time                                                  0.132    0.164
data required time                                                          0.164
--------------------------------------------------------------------------------
data arrival time                                                           0.122
data required time                                                         -0.164
--------------------------------------------------------------------------------
slack (VIOLATED)                                                           -0.042
```

To get an overview of timing design slack and design rule, the slack of all end points, max capacitance, max transition time and max fanout load can be reported.

4. Invoke the **Timing → Report Slack and Constraint** command.
5. Enable the **Setup time**, **Hold time** and **Slack on End Pin/Ports** of **Report Slack** options.
6. Enable the **Max Capacitance, Max Transition** and **Max Fanout** of **Report Constraint** options.



Report Slack and Constraint

Report Slack

☐ Setup Time ☐ Hold Time

☐ Report Slack on Start Pins/Ports

☐ Report Slack on End Pins/Ports

Threshold: 0

Report Constraint

☐ Report Max Capacitance

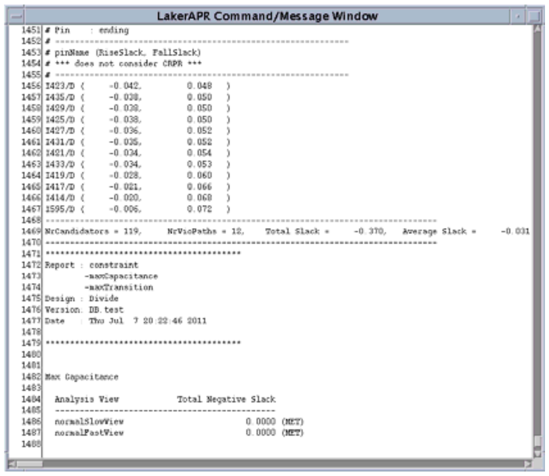
☐ Report Max Transition

☐ Report Only Max Fanout

Threshold: 0

Output File:

7. Click **OK** to analyze all active timing views.



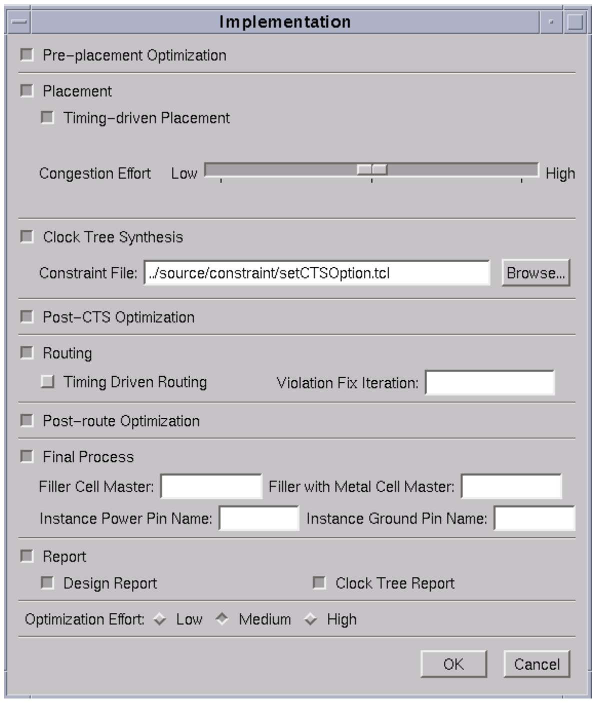
LakerAPR Command/Message Window

```
# Pin    : ending
# -------------------------------------------------------------
# pinName (RiseSlack, FallSlack)
# *** does not consider CRPR ***
# -------------------------------------------------------------
I423/D (     -0.042,          0.048    )
I435/D (     -0.038,          0.050    )
I429/D (     -0.038,          0.050    )
I425/D (     -0.038,          0.050    )
I427/D (     -0.036,          0.052    )
I431/D (     -0.035,          0.052    )
I421/D (     -0.034,          0.054    )
I433/D (     -0.034,          0.053    )
I419/D (     -0.028,          0.060    )
I417/D (     -0.021,          0.066    )
I414/D (     -0.020,          0.068    )
I595/D (     -0.006,          0.072    )
-----------------------------------------------------------------------------------
NrCandidators = 119,     NrVioPaths = 12,     Total Slack =     -0.370,   Average Slack =     -0.031
-----------------------------------------------------------------------------------
****************************************
Report : constraint
        -maxCapacitance
        -maxTransition
Design : Divide
Version: DB.test
Date   : Thu Jul  7 20:22:46 2011

****************************************


Max Capacitance

  Analysis View            Total Negative Slack
  ---------------------------------------------
  normalSlowView                          0.0000 (MET)
  normalFastView                          0.0000 (MET)

```

12.4.2 Recommend Implementation Flow

A recommended implementation flow is ready to use for most typical designs. Major standard steps of timing driven procedures can be enabled or disabled. The benefit is an ease of use environment to handle timing driven blocks by mixed layout designers.

1. Invoke the **LakerAPR → Implementation** command.
2. Assign **Constraint File** of **Clock Tree Synthesis** to ../source/constraint/setCTSOption.tcl



Implementation

Pre–placement Optimization

Placement

Timing–driven Placement

Congestion Effort Low High

Clock Tree Synthesis

Constraint File: ../source/constraint/setCTSOption.tcl

Post–CTS Optimization

Routing

Timing Driven Routing Violation Fix Iteration:

Post–route Optimization

Final Process

Filler Cell Master: Filler with Metal Cell Master:

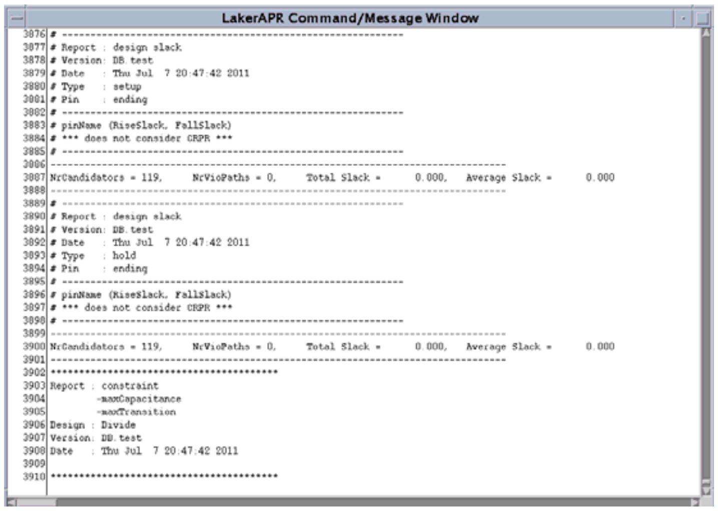
Instance Power Pin Name: Instance Ground Pin Name:

Report

Design Report Clock Tree Report

Optimization Effort: Low [✓ Medium] High

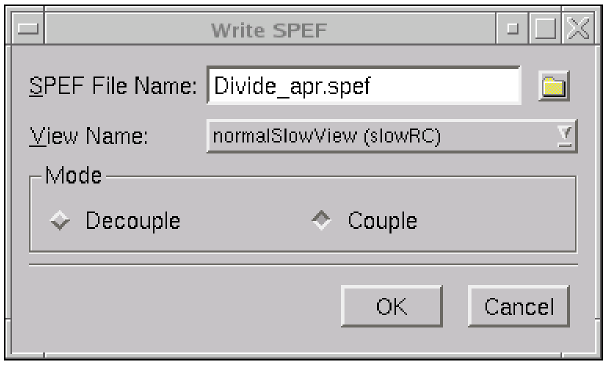
3. Click **OK** to finish all selected steps.
4. Analyze slack and constraint again by Invoking **Timing → Report Slack and Constraint** command.



LakerAPR Command/Message Window

```
# ------------------------------------------------------------
# Report  : design slack
# Version: DB.test
# Date    : Thu Jul  7 20:47:42 2011
# Type    : setup
# Pin     : ending
# ------------------------------------------------------------
# pinName (RiseSlack, FallSlack)
# *** does not consider CRPR ***
# ------------------------------------------------------------
------------------------------------------------------------------------------
NrCandidators = 119,     NrVioPaths = 0,     Total Slack =      0.000,   Average Slack =      0.000
------------------------------------------------------------------------------
# ------------------------------------------------------------
# Report  : design slack
# Version: DB.test
# Date    : Thu Jul  7 20:47:42 2011
# Type    : hold
# Pin     : ending
# ------------------------------------------------------------
# pinName (RiseSlack, FallSlack)
# *** does not consider CRPR ***
# ------------------------------------------------------------
------------------------------------------------------------------------------
NrCandidators = 119,     NrVioPaths = 0,     Total Slack =      0.000,   Average Slack =      0.000
------------------------------------------------------------------------------
****************************************
Report : constraint
        -maxCapacitance
        -maxTransition
Design : Divide
Version: DB.test
Date   : Thu Jul  7 20:47:42 2011

****************************************
```

5. Invoke the **Timing → Write SPEF** command to create a SPEF file for post-layout timing analysis.



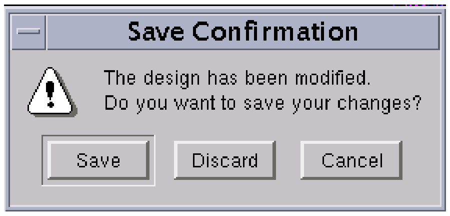
SPEF File Name: Divide_apr.spef
View Name: normalSlowView (slowRC)
Mode: Decouple Couple

12.5 Lab-10D Realize Timing Driven CDPR Implementation

12.5.1 Synchronize Database

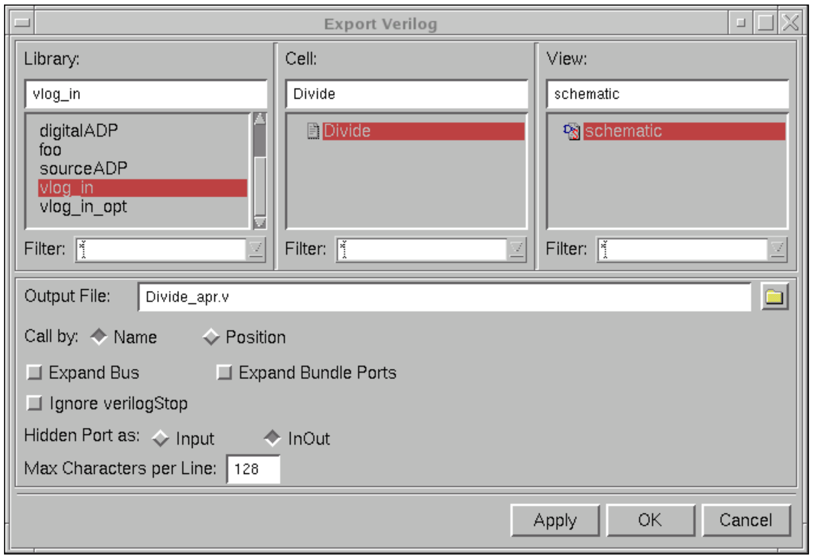
After analyzing the timing slack and constraints, you can update the timing driven CDPR implementation database back to the Laker database.

1. Invoke the **File → Close** command to finish the APR tasks and close the timing driven window.
   The new optimized netlist (sizing, clock tree buffer, buffer insertion) and new layout will be updated to the current cell view.



The design has been modified.
Do you want to save your changes?

2. Export the post-layout hierarchical Verilog file CCU_apr.v by invoking the **File → Export → Verilog** command from the main window.

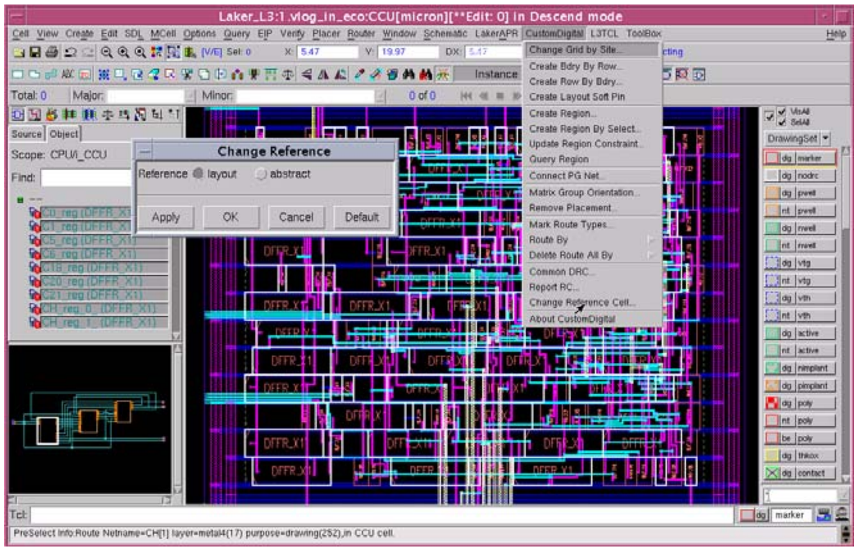


Library: vlog_in (digitalADP, foo, sourceADP, vlog_in, vlog_in_opt)
Cell: Divide
View: schematic
Output File: Divide_apr.v
Call by: Name Position
Expand Bus Expand Bundle Ports
Ignore verilogStop
Hidden Port as: Input InOut
Max Characters per Line: 128

3. Use the **CustomDigital → Change Reference View** command to solve master mismatch issues if necessary.

LakerOA DEF In has a new keyword to define the priority of the reference view list in the *laker.rc* file. The master mismatch issue can be automatically solved in LakerOA.

```
[ImportExport]
ImportDefRefViews = layout abstract abs mdevice
```



PreSelect Info:Route Netname=CH[1] layer=metal4(17) purpose=drawing(252),in CCU cell.

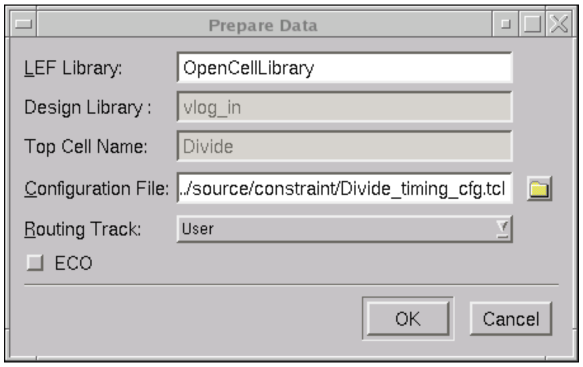
12.6 Lab-10E Post Layout Timing ECO

The Laker system can export hierarchical Verilog files and SPEF files for post-layout timing analysis. Third party tools can conduct RC extraction based on the physical database using LEF/DEF or stream files.

PrimeTime provides what-if analysis and timing ECO change files. For example:

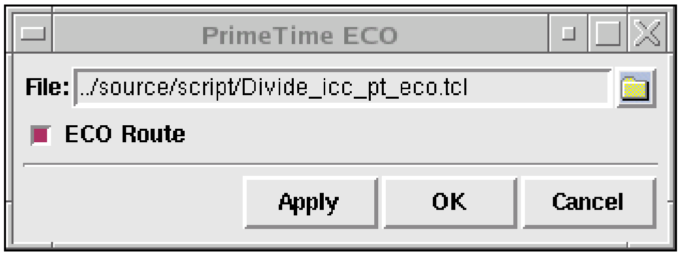
```
pt_shell > write_changes –format icctcl Divide_icc_pt_eco.tcl

########################################################################
# Change list, formatted for IC Compiler
########################################################################
current_instance
current_instance
insert_buffer  [get_pins  {I3/ZN}]  OpenCellLibrary/BUF_X2  -new_net_names
{Y0_BI} -new_cell_names {I3_BI}
current_instance
current_instance
size_cell {I607} {OpenCellLibrary/INV_X2}
```

1. Invoke the **LakerAPR → Prepare Data** command.



LEF Library: OpenCellLibrary
Design Library : vlog_in
Top Cell Name: Divide
Configuration File: ../source/constraint/Divide_timing_cfg.tcl
Routing Track: User
ECO

2. Click the **OK** button to start data preparation from the schematic and layout views.
3. Finish the post-layout timing ECO by invoking the **CustomDigital** → **PrimeTime ECO** command.



File: ../source/script/Divide_icc_pt_eco.tcl
ECO Route

13 Other Topics

13.1 Lab-A1: LEF/DEF Flow in Laker DB

In this lab, you will learn how to go through a LEF/DEF flow in Laker DB custom digital environment.

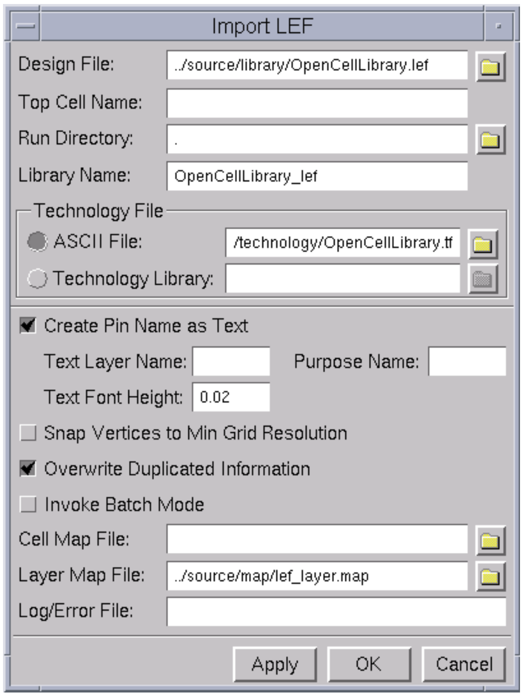
13.1.1 Introduction

LEF/DEF flow from a third party tool is also supported in Laker DB. However, there are differences from the Laker SDL flow. The differences are as listed below.

- MACRO cell layout geometry is from a LEF file by Import LEF. It is stored in an abs (abstract) view.
- Advanced routing rules are not supported by Import LEF yet. They are defined in the same tfNetRouteRule section as Laker SDL flow in DB.
- A TF/LEF library of *OpenCellLibrary_lef* is created by Import LEF.
- A floor plan DEF generated from a third party tool is imported by Import DEF with referring to *OpenCellLibrary_lef*.
- No logic view is created after Import DEF. Therefore, partial selection and realization from the *design hierarchy browser* pane is not supported.
- Row placement of existing cells in Layout Window can be done by **Placer → Place All**.
- Route Vias have higher priority than MCells for routing if both of them are available.
- Both Edit → Others → Change Via and Trim Wire are supported.
- Export Stream will not export the abstract cell view. You can turn on **Export Stream → Advance → Export Abstract Cell View**: **All** or **Via Only** to export them.

13.1.2 TF/LEF Library Creation

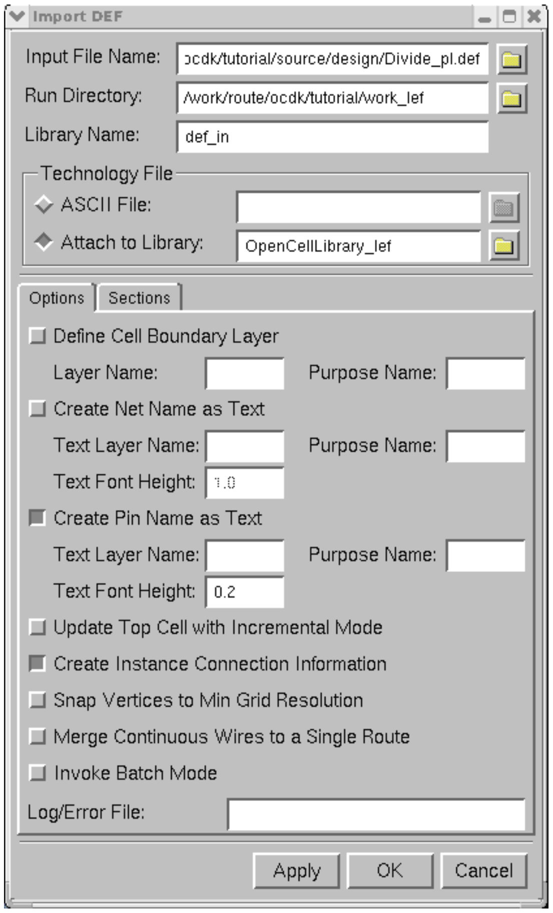
1. Create a TF/LEF library by invoking the **File → Import → LEF** command.
2. In the *Import LEF* form, do the following:
   a. Assign **Design File** to *OpenCellLibrary.lef* for importing MACRO cell layout geometry.
   b. Assign **Library Name** to *OpenCellLibrary_lef*
   c. Assign **Technology File** to *OpenCellLibrary.tf*
   d. Enable the **Create Pin Name as Text** option and assign the associated **Text Font Height** to 0.02.
   e. Assign **Layer Map File** to *lef_layer.map*.
3. Click **OK** button to finish importing a LEF design.


Import LEF

Design File: ../source/library/OpenCellLibrary.lef
Top Cell Name:
Run Directory: .
Library Name: OpenCellLibrary_lef
Technology File
[✓ ASCII File]: /technology/OpenCellLibrary.tf
Technology Library:
[✓] Create Pin Name as Text
Text Layer Name:
Purpose Name:
Text Font Height: 0.02
[ ] Snap Vertices to Min Grid Resolution
[✓] Overwrite Duplicated Information
[ ] Invoke Batch Mode
Cell Map File:
Layer Map File: ../source/map/lef_layer.map
Log/Error File:

13.1.3 DEF Library Creation

1. Create a DEF design library by **File → Import → DEF**.
2. On the *Import DEF* form, do the following:
   a. Assign **Input File Name** to *Divide_pl.def* for importing third party floor plan.
   b. Assign **Library Name** to *def_in*
   c. Assign **Technology File** to *OpenCellLibrary_lef*
   d. Enable the **Create Pin Name as Text** option and assign the associated **Text Font Height** to 0.2.



Input File Name: ocdk/tutorial/source/design/Divide_pl.def

Run Directory: /work/route/ocdk/tutorial/work_lef

Library Name: def_in

Technology File

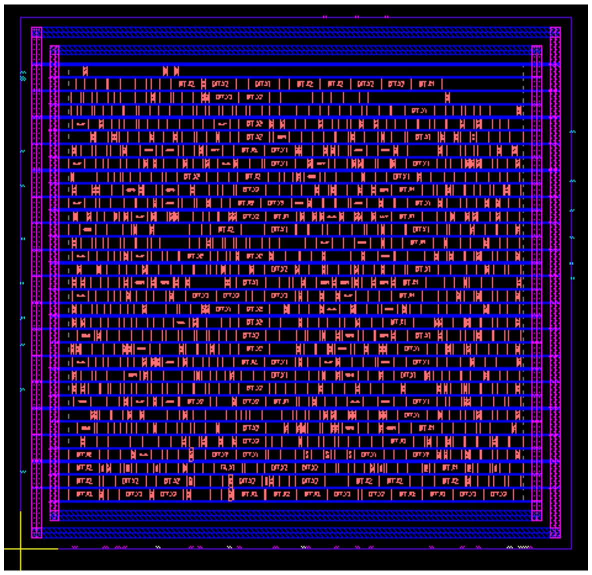
- ASCII File:
- [✓ Attach to Library:] OpenCellLibrary_lef

Options | Sections

- [ ] Define Cell Boundary Layer
  - Layer Name: Purpose Name:
- [ ] Create Net Name as Text
  - Text Layer Name: Purpose Name:
  - Text Font Height: 1.0
- [x] Create Pin Name as Text
  - Text Layer Name: Purpose Name:
  - Text Font Height: 0.2
- [ ] Update Top Cell with Incremental Mode
- [x] Create Instance Connection Information
- [ ] Snap Vertices to Min Grid Resolution
- [ ] Merge Continuous Wires to a Single Route
- [ ] Invoke Batch Mode

Log/Error File:

3. Click the **OK** button to finish importing a DEF design.



13.2 Lab-A2: LEF/DEF Flow in Laker OA

In this lab, you will learn how to go through a pure LEF/DEF flow in the Laker OA custom digital environment.

13.2.1 Introduction

LEF/DEF flow from a third party tool is supported in Laker OA. However, there are differences from the Laker SDL flow. The differences are as listed below.

- Advanced routing rules and MACRO cell layout geometry are supported from a LEF file by Import LEF.
- A LEF library *OpenCellLibrary_lef* is created by Import LEF.
- A floor plan DEF generated from a third party tool is imported by Import DEF with referring to *OpenCellLibrary_lef*.
- No logic view is created after Import DEF. Therefore, partial selection and realization from the *design hierarchy browser* pane is not supported.
- Row placement of existing cells in Layout Window can be done by **Placer → Place All**.
- Route Vias have higher priority than MCells for routing if both of them are available.
- Both Edit → Others → Change Via and Trim Wire are supported.
- Export Stream will not export the abstract cell view. You can turn on **Export Stream → Advance → Export Abstract Cell View**: **All** or **Via Only** to export them.
-

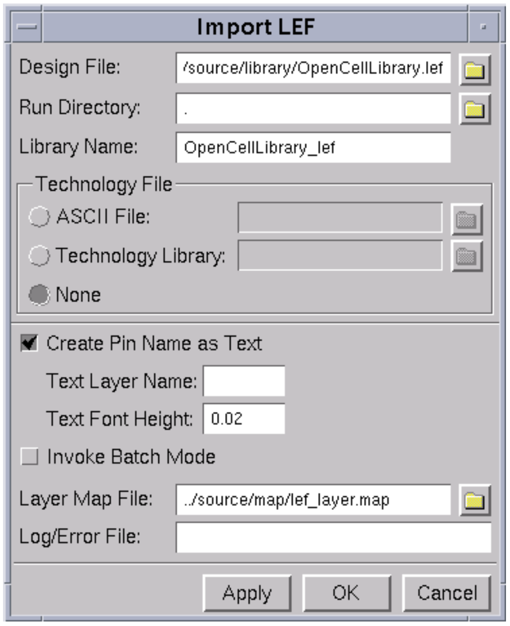
Other Laker technology file sections can be updated by using the **Library → Technology File Import → Replace** command. Database unit must be exactly the same in Laker technology file and LEF files. For example:

```
# database unit definition in OpenCellLibrary.lef
UNITS
  DATABASE MICRONS 2000 ;
END UNITS

MANUFACTURINGGRID 0.005 ;

# database unit definition in OpenCellLibrary.tf
tfLayoutSystemUnit {
  userUnit                micron
  dbScale               0.0005
  dbResolution          0.0050
  xGridSpacing          0.0050
  yGridSpacing          0.0050
  majorGridRatio        5
  gridType                Dot
}
```

13.2.2 LEF Library Creation

1. Create a LEF library by invoking the **File → Import → LEF** command.
2. In the *Import LEF* form, do the following:
   a. Assign **Design File** to *OpenCellLibrary.lef* Assign **Library Name** to *OpenCellLibrary_lef*.
   b. Assign **Technology File** to *None.*
   c. Enable the **Create Pin Name as Text** option and assign the associated **Text Font Height** to 0.02.
   d. Assign **Layer Map File** to *lef_layer.map*.

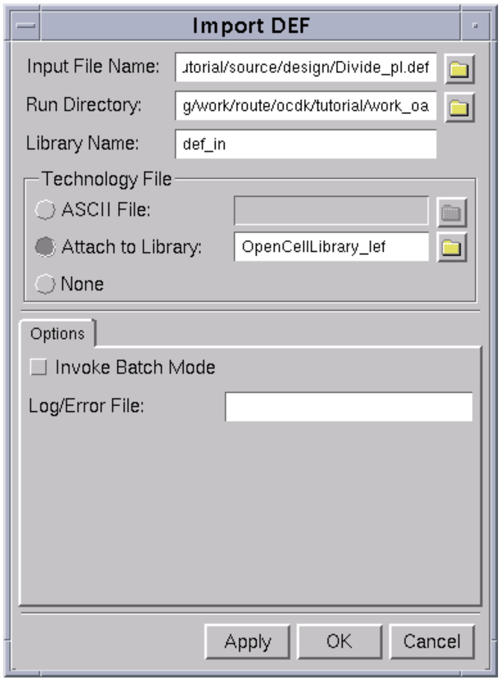
3. Click the **OK** button to finish importing a LEF design.



Design File: /source/library/OpenCellLibrary.lef
Run Directory: .
Library Name: OpenCellLibrary_lef
Technology File
ASCII File:
Technology Library:
[✓ None]
[✓] Create Pin Name as Text
Text Layer Name:
Text Font Height: 0.02
Invoke Batch Mode
Layer Map File: ../source/map/lef_layer.map
Log/Error File:

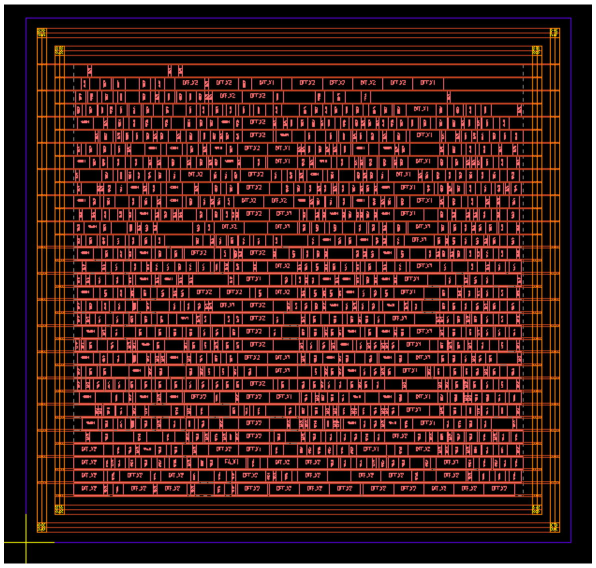
13.2.3 DEF Library Creation

1. Create a DEF design library by **File → Import → DEF**.
2. On the *Import DEF* form, do the following:
   a. Assign **Input File Name** to *Divide_pl.def* for importing a third party floor plan.
   b. Assign **Library Name** to *def_in*.
   c. Assign **Technology File** to *OpenCellLibrary_lef*.



Input File Name: .torial/source/design/Divide_pl.def
Run Directory: g/work/route/ocdk/tutorial/work_oa
Library Name: def_in
Technology File
ASCII File:
[✓ Attach to Library:] OpenCellLibrary_lef
None
Options
Invoke Batch Mode
Log/Error File:

3. Click the **OK** button to finish importing a DEF design.

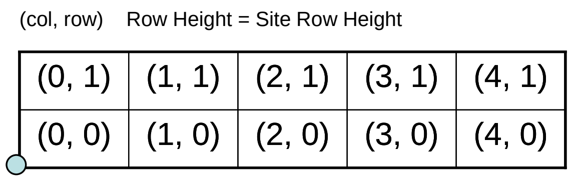


13.3 Lab-B1: Matrix Constraint

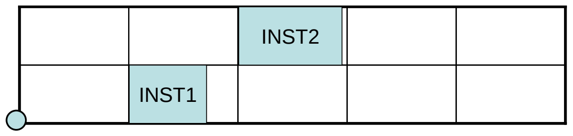
In this lab, you will learn how to use the matrix constraint and invoke the row placement in a specified row area.

13.3.1 Introduction of Matrix Constraint

Matrix constraint is a high level planning tool for relative placement in data path style design. Group members or hierarchical groups can be assigned to valid column and row positions of a defined matrix structure. It is also called relative placement or structure placement.



```
lakerCreateRPGroup –group rpg1
                   –columns 5 –rows 2
                   –xOffset x –yOffset y
```



```
lakerAddToRPGroup –group rpg1  –column 1 –rows 0  –leaf INST1
lakerAddToRPGroup –group rpg1  –column 2 –rows 1  –leaf INST2
```

13.3.2 Define Matrix Constraint

Usually, data path synthesizer and compiler generate a physical constraint in an ASCII format in a tabular form or a line-by-line format.

Essential Tcl commands are provided for handling matrix constraint, including:

lakerCreateRPGroup, *lakerAddToRPGroup*, *lakerDeleteRPGroup*.

Similar relative placement constraints can be converted to Laker generic formats mentioned above by a customized automation script. Refer to the *Laker Tcl Reference* document for complete details on these Tcl commands.

13.3.3 Row Placement in SDL

13.3.3.1 Define Minimum Height of Row Area

The minimum row number should be larger than the number of rows defined in a matrix constraint. After deciding utilization rate and fixing row height, **Placer → Create Row** can be used to create a valid row area.

Referring to matrix constraint file names as matrix_rp.tcl, row number is 13.
Given a site height of 1.4 um, minimum height of a row area is 13 x 1.4 um = 18.2 um.

13.3.3.2 Create Row by Utilization

Create a rectangle row area by utilization for matrix constraint.

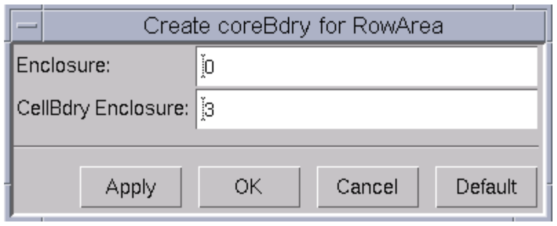
1. Invoke the **Placer → Create Row** command.
2. In the *Create Row* form, do the following:
    a. On the **Options** tab, select an existing pre-defined site by selecting **Name** as *NCSU_FreePDK_45nm.*
    b. Enable row abutted style by enabling the **Double Back** option.
    c. On the **Methods** tab, select the **Utilization** option and set the associated utilization rate to 0.75 first for sparse structure placement first and enable the **Full Cell** option for **Mode**.
    d. Set **Row Spacing** to 0.0 for channel-less floor plan.
    e. Under **Aspect Ratio**, set **Fixed Length** (Y) to 18.5 as previously calculated for channel-less floor plan.

Now, you need to define the origin of the row area using one of the following methods:

- By mouse left-click, or
- Press **TAB** to enter (X,Y) value at the top of Layout window, or
- Press **H** to enter (X,Y) value from **User Input Coordinate** window.

13.3.3.3 Create Cell Boundary

A CellBdry can be created based on a planned core area by a customized script **CustomDigital → Create Bdry by Row**.

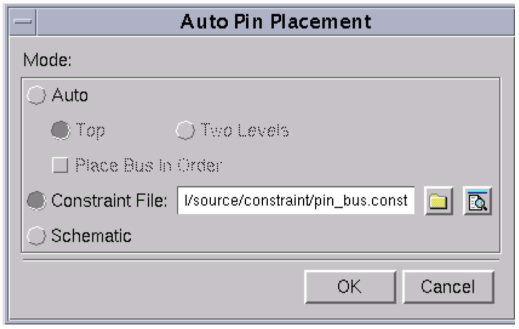

Create coreBdry for RowArea
Enclosure: 0
CellBdry Enclosure: 3

CoreBdry is a layer created for legacy supported feature only if necessary.

CellBdry Enclosure is the distance reserved for ring structure around the whole Row Area.

13.3.3.4 Auto Pin Assignment

After a pin constraint is ready, the **Placer → Pin Placer → Auto Pin Assignment** command can be used to realize the initial pin assignment.

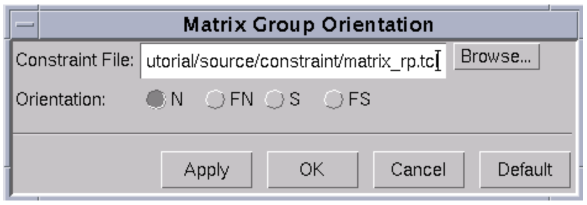


Constraint File: l/source/constraint/pin_bus.const

After this step, the task of initial pin placement is finished.

13.3.3.5 Row-based Placement in SDL

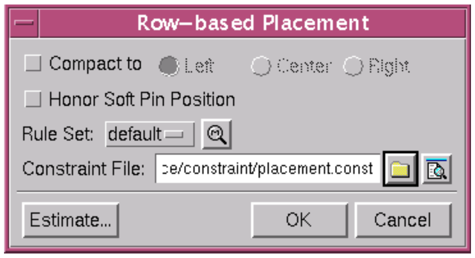
1. Load matrix constraint by the customized **CustomDigital → Matrix Group Orientation** script.



Constraint File: utorial/source/constraint/matrix_rp.tcl
Orientation: N

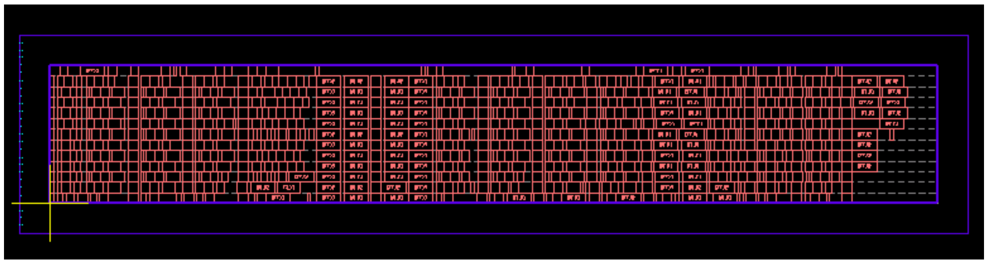
   a. Set **Constraint File** to *../source/constraint/matrix_rp.tcl*
   b. Click **OK** to load the matrix constraint.
2. Click **Select All** icon in the design hierarchy browser panel to select all objects because the Laker system can support partial selection for implementation.
3. Invoke the **Design Brower → Placement → Row Placer** command.
4. Select **Rule Set** according to original routing layer planning
   Available routing layers have an impact on congestion driven placement results. You can confirm in the **Router → Digital Router → Rule Setting** tab if necessary.

5. Fill in **Constraint File** field with the file name of a placement constraint.

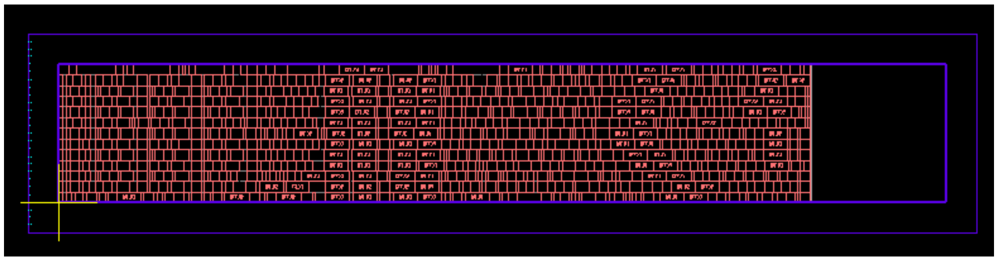


Rule Set: default
Constraint File: ce/constraint/placement.const

6. Click one Row area and apply **Design Brower → Placement → Row Placer → OK** to finish row placement task.



7. The width of row area can be reduced by invoking **Placer → Stretch** to check the placement result of the matrix constraint.
8. Invoke **Placer → Placer All → OK** to finish a faster placement without cell realization because all cells are available in layout window.



13.4 Lab-C1: Limited Layer Routing

13.4.1 Introduction

Two or three limited layer routing needs careful resource arrangement by several approaches. Assuming a floor plan with horizontal row area and HV or HVH routing preferred direction, the following methods can be applied to limit layer routing:

- Manually define the row spacing to reserve more metal1 routing resource in the blank channel.
- Row spacing is usually an integral multiplier of metal1 routing pitch to avoid unnecessary waste of metal1 routing resources in the channel. This method is useful for small Y/X aspect ratio floor plan. More metal1 resources are available.
- Manually define the placement blockage tile array in a local congested area. This is a heuristic approach.
- Use SPINE style routing to keep an effective routing channel usage.
- Manual SPINE pre-wiring followed by automatic placement. This method reduces high fan out routing resources and gets better routing patterns for critical nets.

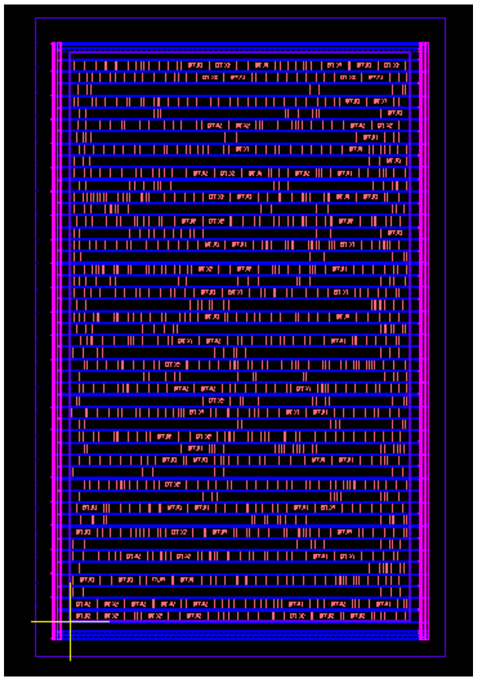
A good floor plan solution is very design dependent for limited layer routing. The Laker Custom Digital solution covers these using several approaches.

13.4.2 Channel Floorplan

1. Invoke the **Placer → Create Row → Methods → Row Spacing** command,
2. Disable the **Placer → Create Row → Options → Double Back** command if necessary.

13.4.3 Automatic Pseudo blockage

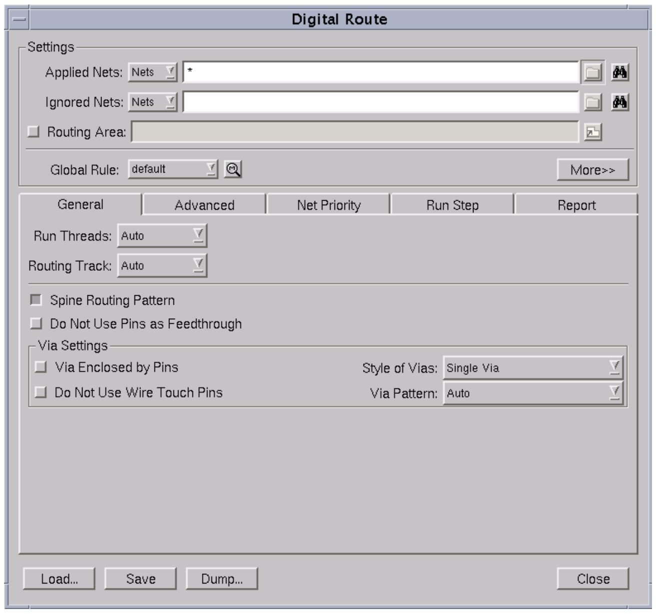
Pseudo blockage is automatically triggered during detailed placement optimization.



As illustrated, several continuous empty channels are revered by row placement for SPINE style routing if applicable.

13.4.4 SPINE Style Routing

Enable SPINE routing during Initial Route or fix them as much as possible during Detail Route in the **General** tab of the *Digital Route* form (invoked with **Router→ Digital Router → Digital Route**).



Digital Route

Settings

Applied Nets: Nets *

Ignored Nets: Nets

☐ Routing Area:

Global Rule: default

Run Threads: Auto

Routing Track: Auto

☐ Spine Routing Pattern

☐ Do Not Use Pins as Feedthrough

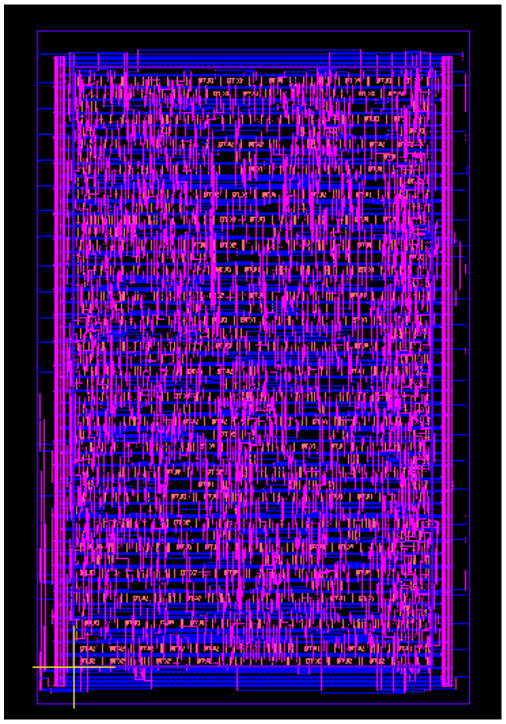
Via Settings

☐ Via Enclosed by Pins

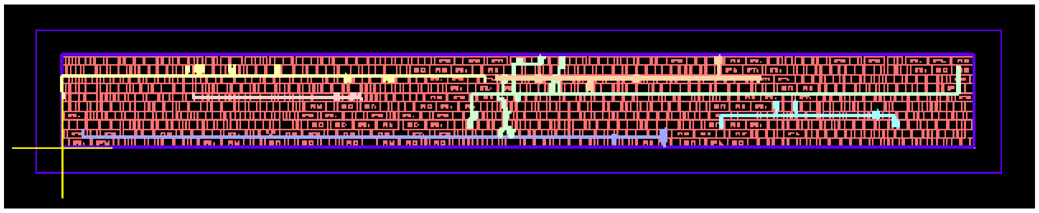
☐ Do Not Use Wire Touch Pins

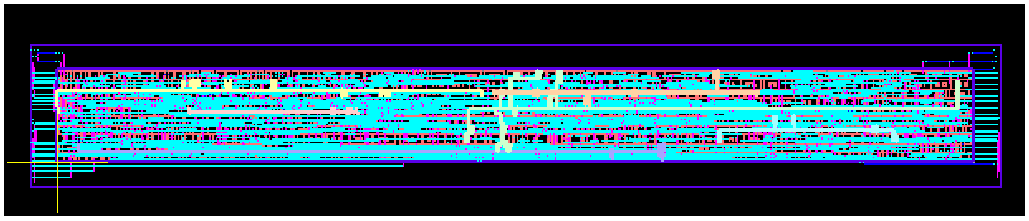
Style of Vias: Single Via

Via Pattern: Auto



Initial Route will automatically create main SPINE and secondary SPINE simultaneously without manual setting. The following example is a large aspect ratio floor plan and a few highlighted long wires after routing.



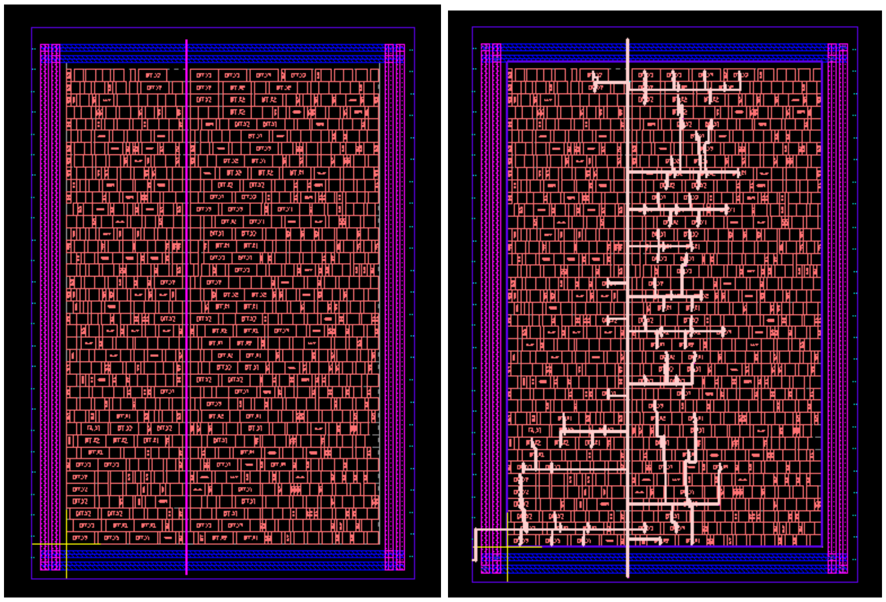


13.4.5 Manual SPINE Pre-wiring

You can set the placement constraint to honor long pre-wire style for critical signals.

If the pin width or depth is larger than a row height, it is defined as a long pre-wire pin. For example, the following statement identifies that the net weight connects to long pre-wire pins.

```
.NetWeightPreWire netWeight_1_to_255
```



The left figure illustrates a main SPINE that is defined by manual wiring. The secondary SPINE structure based on the main SPINE is automatically created by Initial Route.

13.5 Lab-D1: Cell Level OD Sharing

13.5.1 Introduction

Custom cells designed with common power and ground portions can be overlapped with each other to reduce chip area if applicable.

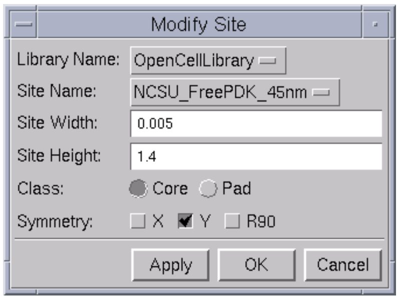
- The definition of potential overlapped width and cell type is defined by a generic constraint file.
- The site width must be defined as a new fine width to keep on-site placement results after cell level overlap.

13.5.2 Cell Level OD Sharing

13.5.2.1 Define Fine Site Width

Because the overlapped width is smaller than a normal site width for a standard cell, a fine site width needs to be redefined to ensure on-site legalization.

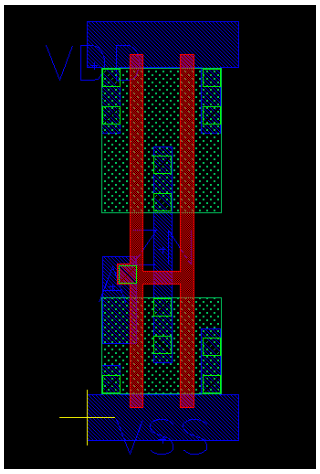
1. Use minimum dbResolution for cell level OD sharing scheme.
2. Invoke the **Floorplan → Site → Modify** command in the *Main* window.
3. In the *Modify Site* form, define **Site Width** as minimum resolution 0.005.



Library Name: OpenCellLibrary
Site Name: NCSU_FreePDK_45nm
Site Width: 0.005
Site Height: 1.4
Class: [✓ Core] Pad
Symmetry: X [✓ Y] R90

13.5.2.2 Define Cell Overlap Region

The library developer has to provide necessary information of legal cell overlap regions in order to prepare for a placement constraint file. According to layout style of Open Cell Library, INV_X8 has the capability for cell level OD sharing on the right side.



The legal overlap value can be calculated based on the design rule manual in a definition file *OpenCellLibrary.od*.

```
# Overlapped region: 0.205
INV_X8 R
```

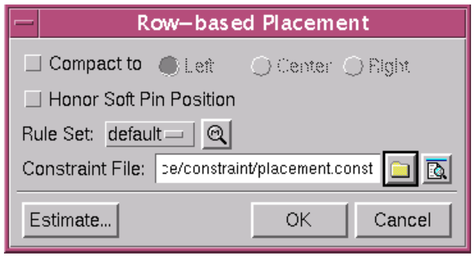
13.5.2.3 Preparation of Placement Constraint File

A new section of placement constraint is added for cell level OD sharing scheme.

```
# You need to redefine fine site width of 0.005 for valid cell
level OD sharing
.ODSharingLibraryFile ../source/library/OpenCellLibrary.od
.ODSharingCompactEffort medium
```

13.5.2.4 Row-based Placement in SDL

1. Import design and create a floorplan by redefined site.
2. Click the **Select All** icon in the design hierarchy browser pane to select all objects because the Laker system can support partial selection for implementation.
3. Invoke the **Design Brower → Placement → Row Placer** command.
4. Select the **Rule Set** according to the original routing layer plan.
5. Available routing layers have an impact on congestion driven placement results. You can confirm in the **Router → Net Router → Rule** tab if necessary.
6. Fill in the **Constraint File** field with the file name of a placement constraint.
7. Click one Row area and apply **Design Brower → Placement → Row Placer → Estimate** to get the utilization rate calculated by placer engine.


Row–based Placement

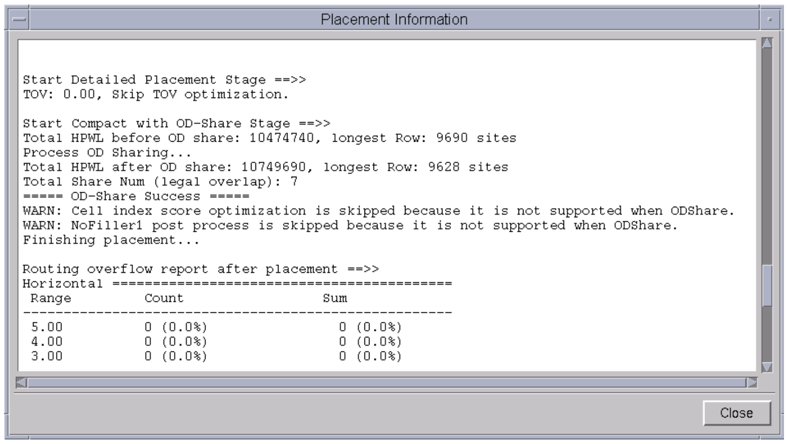
☐ Compact to ● Left ○ Center ○ Right

☐ Honor Soft Pin Position

Rule Set: default

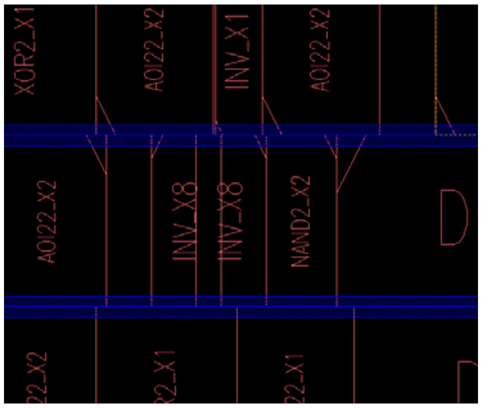
Constraint File: :e/constraint/placement.const

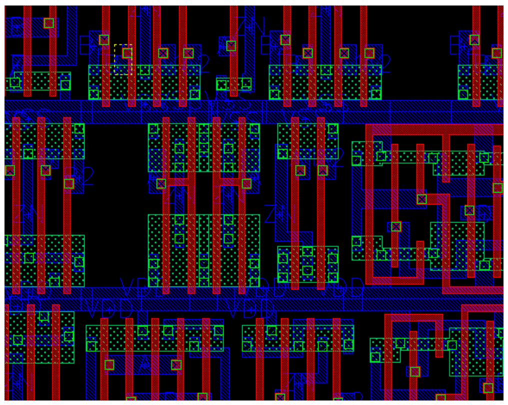
8. Click one Row area and apply **Design Brower → Placement → Row Placer → OK** to finish row placement task.
9. Check related information about cell level OD sharing process in the log file.



```
Start Detailed Placement Stage ==>>
TOV: 0.00, Skip TOV optimization.

Start Compact with OD-Share Stage ==>>
Total HPWL before OD share: 10474740, longest Row: 9690 sites
Process OD Sharing...
Total HPWL after OD share: 10749690, longest Row: 9628 sites
Total Share Num (legal overlap): 7
===== OD-Share Success =====
WARN: Cell index score optimization is skipped because it is not supported when ODShare.
WARN: NoFiller1 post process is skipped because it is not supported when ODShare.
Finishing placement...

Routing overflow report after placement ==>>
Horizontal ===========================================
 Range          Count                     Sum
-----------------------------------------------------
 5.00           0 (0.0%)                  0 (0.0%)
 4.00           0 (0.0%)                  0 (0.0%)
 3.00           0 (0.0%)                  0 (0.0%)
```

10. Use **Query → Find** to search for INV_X8 and check the overlapped placement result.





13.6 Lab-E1: Cell Level ESD Spacing

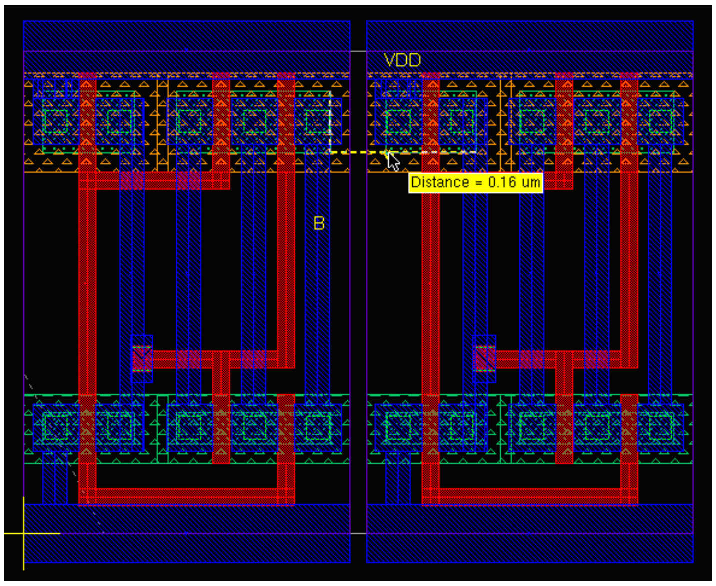
13.6.1 Introduction

ESD issues within a core area are usually solved by using a tie-high and tie-low cell without directly connecting to power and ground rails in advanced process nodes. For mainstream nodes, direct connection of power and ground rails are still used for traditional flow and compact size in a high utilization design.

When an input in of a pass gate design is tied-high or tied-low, PG connected OD regions will be analyzed and follows a larger spacing rule to avoid direct power ground path due to small resistance. ESD OD spacing checks between PG and normal signals is excluded.

A DRC based rule checker cannot be verified unless tie-high and tie-low signals are completely routed in advanced. The longer iteration cycle is also an issue for high utilization designs.

The Laker system supports this special requirement by a customized rule checker with logic information during the placement stage. It provides a seamless flow and shorter iteration cycle.



The figure above illustrates an example of PG ESD distance check. An input pin B of the left PGMUX2_X1 instance is connected to ground. An OD region connected to power of the right PGMUX2_X1 instance is placed nearby.

13.6.2 Cell Level ESD Spacing

13.6.2.1 Define ESD spacing rule

The library developer has to provide necessary information of PG ESD spacing rules in order to prepare for a placement constraint file. According to design rule manual of Open Cell Library, a placement constraint file can be prepared as follows.

```
.ESDRuleSpacing 0.2
.PowerNetName VDD
.GroundNetName VSS
.PolyLayerPurpose 9 252
.ODNTypeLayerPurpose 1 252
.ODPTypeLayerPurpose 1 252
```

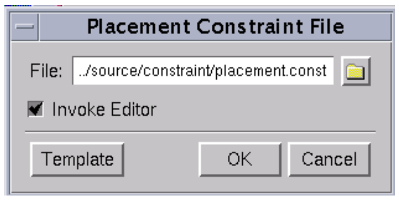
Open Cell Library uses single diffusion (active area) layout. However, Open Cell Library has no pass gate design style.

Note that the Laker layer number and purpose number can be identified by the yellow tips of the layer panel. It is layer number and data type defined in a GDS stream file.

13.6.2.2 Row-based Placement in SDL

1. Import the reference design CPU.sp and create a row area with 0.85 utilization rate as usual.
2. Click the **Select All** icon in the design hierarchy browser pane to select all objects because the Laker system can support partial selection for implementation.
3. Invoke the **Design Brower → Placement → Row Placer** command.
4. Select the **Rule Set** according to the original routing layer plan.
5. Available routing layers have an impact on congestion driven placement results. You can confirm in the **Router → Net Router → Rule** tab if necessary.
6. Leave the **Constraint File** field blank to catch potential PG ESD spacing violations if any.
7. Click one Row area and apply the **Design Brower → Placement → Row Placer → Estimate** command to get the utilization rate calculated by placer engine.

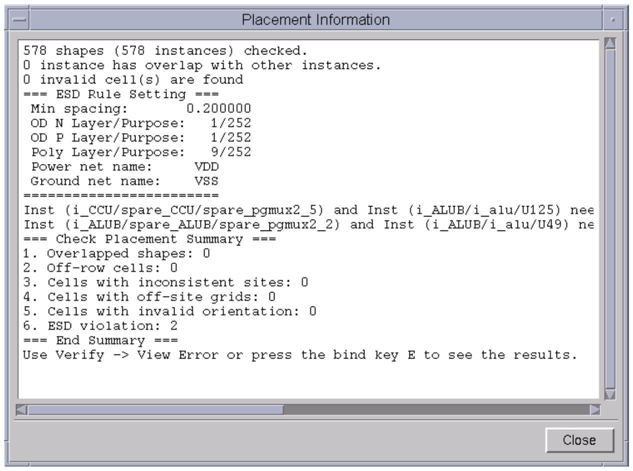
8. Click one Row area and apply the **Design Brower → Placement → Row Placer → OK** command to finish the row placement task.
9. Invoke the **Placer → Placement Constraint** command.


Placement Constraint File
File: ../source/constraint/placement.const
Invoke Editor
Template
OK
Cancel

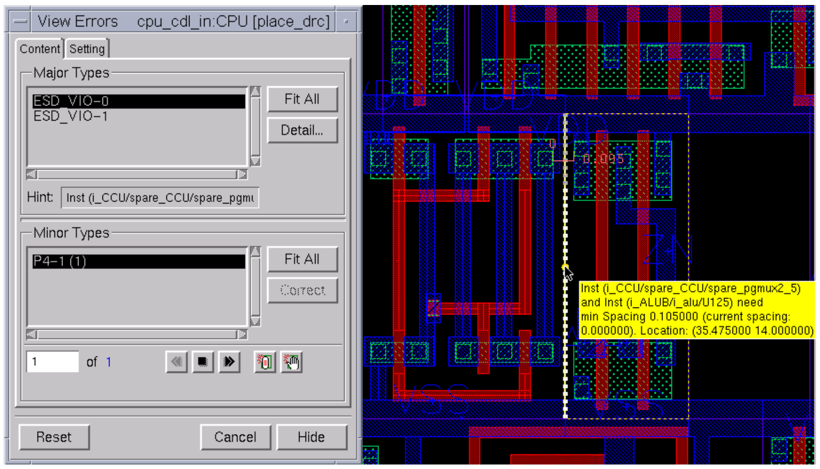
10. Click **OK** to assign a placement constraint for placer related features.
11. Invoke the **Placer → Check Placement** command to check if a PG ESD rule exists after placement. Press "E" or **Verify → View Error** to display a violation if PG ESD rule violations are found.

    Two violations are found for this demo case. For example, measured OD spacing is 0.095 which is smaller than min PG ESD spacing 0.2. The solution is to have a minimum spacing of 0.105 between these two instances.

    If you cannot any catch PG ESD rule violations, try to increase the utilization rate and place it again.

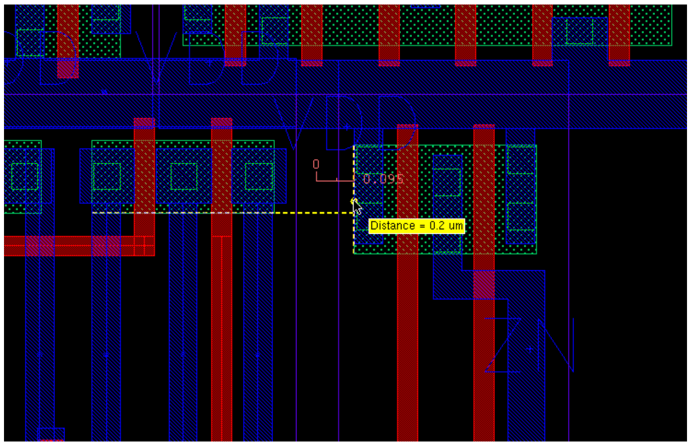


```
578 shapes (578 instances) checked.
0 instance has overlap with other instances.
0 invalid cell(s) are found
=== ESD Rule Setting ===
 Min spacing:        0.200000
 OD N Layer/Purpose:    1/252
 OD P Layer/Purpose:    1/252
 Poly Layer/Purpose:    9/252
 Power net name:      VDD
 Ground net name:     VSS
========================
Inst (i_CCU/spare_CCU/spare_pgmux2_5) and Inst (i_ALUB/i_alu/U125) nee
Inst (i_ALUB/spare_ALUB/spare_pgmux2_2) and Inst (i_ALUB/i_alu/U49) ne
=== Check Placement Summary ===
1. Overlapped shapes: 0
2. Off-row cells: 0
3. Cells with inconsistent sites: 0
4. Cells with off-site grids: 0
5. Cells with invalid orientation: 0
6. ESD violation: 2
=== End Summary ===
Use Verify -> View Error or press the bind key E to see the results.
```



Inst (i_CCU/spare_CCU/spare_pgmux2_5) and Inst (i_ALUB/i_alu/U125) need min Spacing 0.105000 (current spacing: 0.000000). Location: (35.475000 14.000000)

12. Use the **Placer → Legalize Cell Placement** command to solve found PG ESD rule violations.
13. Invoke the **Placer → Check Placement** command to check if PG ESD rule still exists after legalization.



13.7 Lab-F1: Laker OpenAccess Incremental Technology

13.7.1 Introduction

An OA Technology may have any number of references to OA Technologies in other OA Libs. This can result in a hierarchy of references across OA Libs, thereby forming a graph of technology data. An OA Technology that has a reference to another Technology

1. is derived from the referenced Technology.
2. inherits information from the referenced Technology.

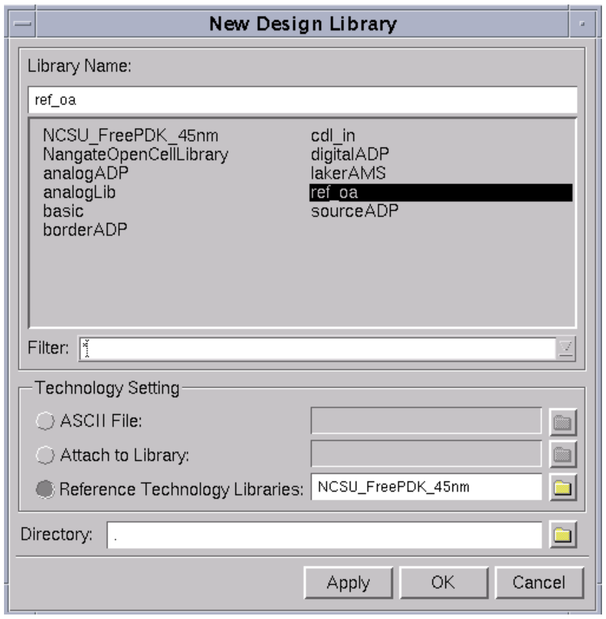
13.7.2 Create an Incremental Technology Library

OpenCellLibrary has two OpenAccess libraries, including *NangateOpenCellLibrary* and *NCSU_FreePDK_45nm*. The goal is to create a new technology library referred to *NangateOpenCellLibrary* and inheriting information from it.

The original OpenCellLibrary library release has two libraries named NCSU_FreePDK_45nm and NangateOpenCellLibrary. You can add these two libraries in the library mapping path in the *lib.defs* file.

```
DEFINE NCSU_FreePDK_45nm ../source/openaccess/NCSU_FreePDK_45nm
ASSIGN NCSU_FreePDK_45nm writePath ../source/openaccess/NCSU_FreePDK_45nm
ASSIGN NCSU_FreePDK_45nm libMode shared
DEFINE NangateOpenCellLibrary ../source/openaccess/NangateOpenCellLibrary
ASSIGN NangateOpenCellLibrary
writePath ../source/openaccess/NangateOpenCellLibrary
ASSIGN NangateOpenCellLibrary libMode shared
```

1. Create a new library by invoking the **Library → New** command.
2. In the *New Design Library* form, do the following:
   a. Assign **Library Name** to *ref_oa*
   b. Enable **Technology Setting → Technology Libraries Setting** and assign it to *NCSU_FreePDK_45nm.*



Library Name: ref_oa

NCSU_FreePDK_45nm	cdl_in
NangateOpenCellLibrary	digitalADP
analogADP	lakerAMS
analogLib	ref_oa
basic	sourceADP
borderADP	

Filter: *

Technology Setting
ASCII File:
Attach to Library:
[✓ Reference Technology Libraries:] NCSU_FreePDK_45nm

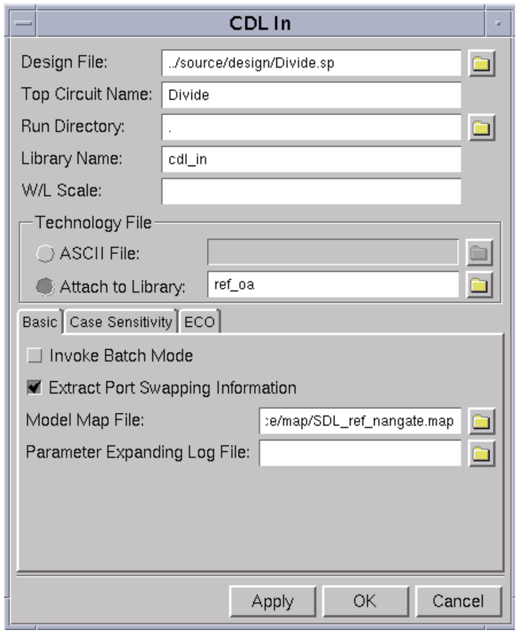
Directory: .

3. Click the **OK** button to create a new library.

13.7.3 Create a Design Library

You can import a design with a technology library attached to *ref_oa*. A modified model map file is necessary to map original *NangateOpenCellLibrary* OpenAccess library.

1. Define a cell library in a library mapping path by selecting the **Library → Mapping Path** command.
   Make sure ref_oa, NangateOpenCellLibrary and NCSU_FreePDK_45nm are listed in the mapping path.
2. Invoke **File → Import → CDL In** to import a CDL design.
3. In the *CDL In* form, do the following:
   a. Assign **Design File** to *Divide.sp*
   b. Assign **Top Circuit Name** to *Divide*
   c. Assign **Library Name** to *cdl_in*
   d. Assign **Technology File → Attach to Library** to *ref_oa*
   e. Assign **Model Map File** to *SDL_ref_nangate.map*.


CDL In

Design File: ../source/design/Divide.sp
Top Circuit Name: Divide
Run Directory: .
Library Name: cdl_in
W/L Scale:

Technology File
( ) ASCII File:
(●) Attach to Library: ref_oa

Basic | Case Sensitivity | ECO

☐ Invoke Batch Mode
☑ Extract Port Swapping Information
Model Map File: :e/map/SDL_ref_nangate.map
Parameter Expanding Log File:

f. Click **OK** to finish importing a CDL file.

14 Limitations and Known Problems

14.1 Laker OA Flow

- Default name space of bus representation is angle bracket in Laker OA. For example, A[7:0] will become A<7:0>.
- The format of layer map file for Import LEF in Laker OA supports both Si2 lef2oa style and Laker layer map style after OA2010.05 release.

14.2 Laker Import Verilog Flow

Import Verilog support in Laker DB main window is supported.

- Default name space of bus representation is angle bracket. For example, A[7:0] will become A<7:0>.

14.3 Laker Cell Level ESD Spacing and OD Sharing

Cell level PG ESD spacing only checks the same row spacing at the current stage. Cell level OD sharing only supports one value for a library layout at current stage.

Contact Laker Technical Marketing for early access support of this new Custom Digital features.

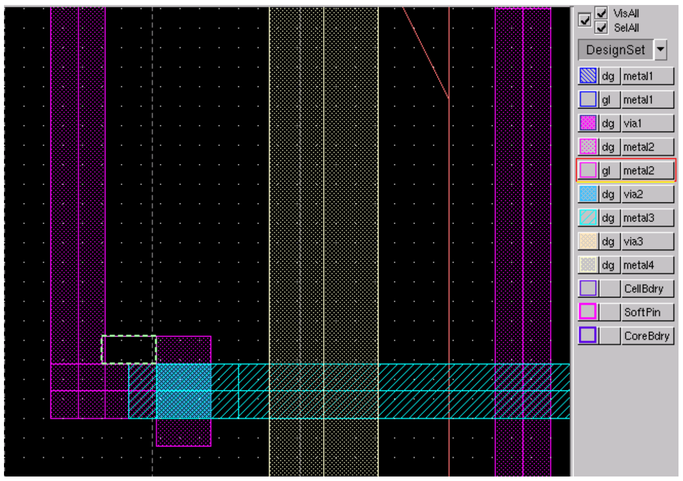
14.4 Layer Mapping for Notch Gap Fill

Starting from the Laker 2010.07 version, notch gap fill shapes generated by routers are created with gapFill purpose. The main reason is to have consistent behavior with the OpenAccess standard.

- Major Laker features automatically handle gapFill purpose. For example, Flight line, Net Tracer, Short Detector, etc.

- Data import and export features need additional layer mapping descriptions for a layer map file. For example:

```
metal1 gapFill 11 0
metal2 gapFill 13 0
metal3 gapFill 15 0

poly  drawing  9  0
contact  drawing  10  0
metal1  drawing  11  0
via1  drawing  12  0
metal2  drawing  13  0
via2  drawing  14  0
metal3  drawing  15  0
```



14.5 New SDL Model Map File in OA

Starting from the Laker OA2010.05p1 version, new model map file sections are introduced to support MCells, Tcl PCells, and PyCells in a general SDL flow.

- New sections in OA
  [MODEL_MAP]
    Specify the mapping between logic and realized layout.
  [DEVICE_MOS]
    Specify the mapping between the *Stick Diagram Compiler* or *Matching Device Creator* windows and the realized layout.

  Example:

  ```
  [MODEL_MAP]
  X AND2_X1     OpenCellLibrary { AND2_X1  layout }
  X AND2_X2     OpenCellLibrary { AND2_X2  layout }
  ```

- Obsoleted sections in OA
  [MAP]

```
[PARAMETER]
[PORTMAP]

Example:

[MAP]
X AND2_X1     OpenCellLibrary AND2_X1
X AND2_X2     OpenCellLibrary AND2_X2
```

Invoke **Library → Replace Model Map File** to update a new model map file for an existing Laker OA library.

14.6 Preferred Routing Direction

A correct preferred routing direction is important for a clean digital route with good performance. The *tfNetRouteRule* section defines the preferred direction as H, V, or HV. The direction value in a *laker.ro* file copies the direction from the technology database during its initial creation. Rules defined by a *laker.ro* file have higher priority than ones defined in a technology file for routing constraints.

- If H or V is defined in a laker.ro file or a technology file, preferred routing direction is also explicitly defined. X/Y cost has no effect for final preferred direction.
- If HV is defined in a laker.ro or a technology file, lakerICSInput converter decides the final preferred routing direction based on X/Y cost of each layr. For example,
- Preferred routing direction is H when X/Y cost is 1/8.
- Preferred routing direction is V when X/Y cost is 8/1.

By setting the routing direction as HV in the *tfNetRouteRule* section, it is possible to dynamically adjust the routing preferred direction after changing the X/Y cost in the **Rule Tab** of current GUI form.

You can also define explicit preferred routing in form opened with the **Router → Digital Route → Rule Setting** command.

Revision History

Revision	Date	Description	
6.2	02/05/13	Updated the footer information.	MY
6.1	12/26/12	Updated the header and footer information.	MY
6.0	06/14/12	Update net optimization Add PT ECO flow	HSW
5.11	11/23/11	Update dual view library preparation Update dummy Verilog library preparation	HSW
5.1	08/01/11	Update release schedule to 2011.06p1 and OA2011.08p1	HSW
5.0	07/26/11	Add timing driven flow Add antenna check and fix Add antenna rule example Update symbol view creation of Verilog library file Update Import Verilog flow with standard SDL flow Add RC table preparation Update new PG Route and Digital Route	HSW
4.1	12/03/10	Add description of preferred routing direction	HSW
4.0	09/09/10	Update Model Map file section for Laker OA2010.08 Add LEF/DEF flow for Laker OA	HSW
3.0	08/05/10	Update tutorial materials. Remove Tcl scripts which are supported by Laker2010.07 features. For example, Pin Placer, etc. Update Laker OA incremental Tech Update notch gap fill	HSW
2.0	06/11/10	Change DigitalRouteAll to CustomDigital Change routing track definition of OpenCellLibrary.tf for better hierarchical layout implementation	HSW
1.6	04/16/10	Add a limitation of PG routing with read only LEF master library.	HSW
1.5	04/13/10	Added details to the Introduction, modified Verilog Import, added SPINE. Based on Laker 2010.03.	HSW
1.4	3/31/10	Update OpenCellLibrary source file, change menu name to CustomDigital, add Main window menu. Based on Laker 2010.03.	HSW
1.3	3/23/10	Added cell level ESD spacing. Based on Laker 2010.03.	HSW
1.2	3/18/10	Added Cell level OD spacing and overlap. Based on Laker 2010.03.	HSW
1.1	3/11/10	Added technology file preparation section. Based on Laker 2010.03.	HSW
1.0	3/1/10	Initial release. Based on Laker 2010.03.	HSW