Laker CDPR Data Preparation and SDL Tutorial

1 Overview 2
2 Technology Overview 2
2.1 The Laker™ Custom Digital Placer 2
2.2 The Laker™ Custom Digital Router 3
3 Environment Setup 4
3.1 Tool Installation Version 4
3.2 Open Cell Library 4
3.3 Laker CDPR Data Preparation Tutorial 5
4 Technology File Preparation 6
4.1 Lab-4A: Routing Rules 7
4.2 Lab-4B: Foundry DRC Rules 8
5 Library Preparation 10
5.1 Lab-5A: Library Preparation by GDS 10
5.1.1 Import Stream Files 10
5.1.2 Net/Port Extraction 11
5.1.3 Define Global Power/Ground Net 12
5.1.4 Abstract Creation (Optional) 13
5.1.5 Define Site Information 13
5.1.6 Update Macro Cell Property 14
5.2 Lab-5B: Verilog and SPICE Library Files 15
6 Design Preparation 16
6.1 Lab-6A: Design Preparation by CDL 16
6.1.1 Import CDL Files 16
6.2 Lab-6B: Design Preparation by Verilog 17
6.2.1 Import Verilog Files 17
7 CDPR Tutorials 18
8 Floorplan 19
8.1 Lab-8A: Floorplan Initialization 19
8.1.1 Routing Resource Plan 19
8.1.2 Set Routing Layers 20
8.1.3 Initialize Area Estimation 20
8.1.4 Create Row 21
8.1.5 Create Cell Boundary 23
8.1.6 Initial Pin Assignment 24
9 Pre-Placement 25
9.1 Lab-9A: Add Physical Only Cells (Optional) 25
9.1.1 Introduction of Physical Only Cells 26
9.1.2 Add End Cap 26
9.1.3 Add Well Tap 27
9.1.4 Mark Placement Status of Physical Cells 28
10 Power Plan 28
10.1 Lab-10A: Create Core Ring and Stripes 28
10.1.1 Introduction of PG Route Type 28
10.1.2 Create Core Rings 29
10.1.3 Create Stripes 31
11 In-Placement 32
11.1 Lab-11A: Row Placement 32
11.1.1 Preparation of Placement Constraint File 32
11.1.2 Assign Placement Constraint File 33
11.1.3 Create Placement Blockage 33

11.1.4 Row-based Placement in SDL 34
12 Post-Placement 35
12.1 Lab-12A: Post-Placement 35
12.1.1 Pin Optimization 35
12.1.2 Check Placement 37
12.1.3 Check Spare Cell Placement 37
12.1.4 Add Core Fillers 39
12.1.5 PG Connection of Physical Cells 39
12.1.6 Connect PG Rails of Standard Cells 39
13 In-Route 40
13.1 Introduction 40
13.2 Lab-13A: Digital Route 41
13.2.1 Digital Route 41
13.2.2 Routing Quality of Result 43
14 Limitations and Known Problems 44
14.1 Laker OA Flow 44
14.2 Laker Import Verilog Flow 44
14.3 New SDL Model Map File in OA 44

1 Overview

The Laker™ Custom Digital Placer and Router (CDPR) provide unique automation for placement and routing of custom and standard cells within the Laker Custom Layout environment. It allows precise custom design of the digital blocks often used in mixed signal and custom digital designs in order to meet the critical performance requirements that often times cannot be achieved with a standalone digital automatic place and route (P&R) tool. Its proprietary technology allows you to:

- Save time with automated creation of digital blocks without leaving the Laker Custom Layout environment
- Achieve the performance of full-custom layout with the speed of P&R
- Enjoy the confidence of using proven standard cells while maintaining hand-crafted quality
- Leverage all the features of the Laker Custom Layout Automation System for things like hand-routing of critical nets, or hand-drawing of routing spines.
- Save time using proven standard cells for high performance digital applications that previously had to be done by hand
- Avoid time-consuming switching between digital automatic place and route (P&R) and custom layout environments and the associated data preparation and translation
- Improve yield with post-route optimization that includes double via insertion, antenna fixing, and jog removal

2 Technology Overview

2.1 The Laker™ Custom Digital Placer

The Laker™ Custom Digital Placer can obtain optimum placement results with ease by allowing the user to:

- Quickly and automatically place standard cells, optimized for minimum wire length
- Perform incremental selection and placement with drag-and-drop simplicity

- Pack the placement area or selected regions automatically
- Manually optimize placement, layout, or routing with any of the features of the Laker Custom Layout Automation System
- Automatically place pins
- Avoid the set-up, data translation, and time penalty of using a standalone P&R tool
- Work seamlessly with the Laker Custom Digital Router

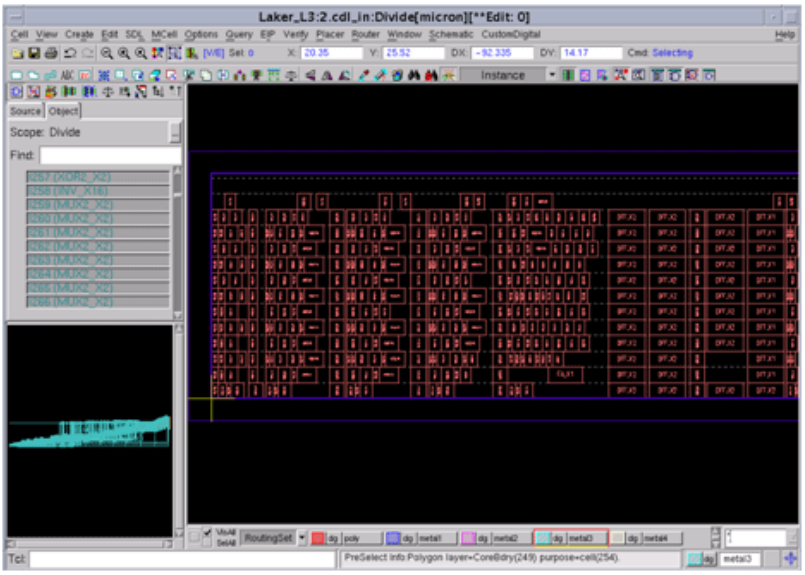

Figure 1: The Laker Custom Digital Placer

2.2 The Laker™ Custom Digital Router

The Laker™ Custom Digital Router saves time with unique automation technology for routing of digital blocks within the Laker Custom Layout Automation System in the following ways:

- Unique hybrid routing technology combines grid-and shape-based routing for very high route completion rates. Global routing enables congestion analysis, mapping, and display during the floor-planning and placement stage (see Figure 2).

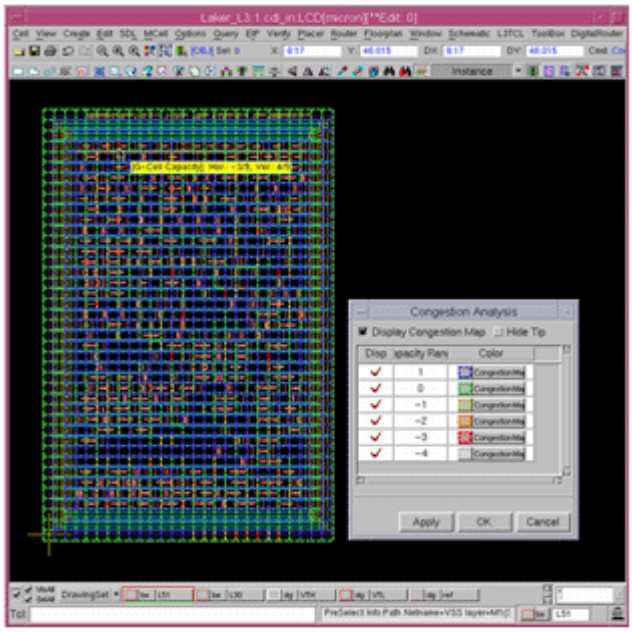

Figure 2: **Optimize Placement and Routing Channels Using Congestion Maps**

- Next, shape-based routing is used to go off the digital routing grid to connect to off-grid pins, complete routes, and avoid DRC violations.
- Spine-style routing is available that is ideally suited for the routing of memory blocks.

3 Environment Setup

3.1 Tool Installation Version

Laker OA2011.03 for PG router and digital router is required.

3.2 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, the Si2 Nangate[1] PDKv1.3_2009_07 release of the Open Cell Library was selected for 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 .
   ```

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

3.3 Laker CDPR Data Preparation Tutorial

On top of the previous work of the Open Cell Library database, the following derivatives are created for the Laker CDPR Data Preparation Tutorial.

source/technology	
OpenCellLibrary.tf	Laker technology file with advanced routing rules
laker.dsp	Laker display file
OpenCellLibrary.captbl	OpenCellLibrary reference CapTable 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.lef	Laker revised LEF file with modified metal1 fat metal rules and new cells
OpenCellLibrary.cpf	Laker cell property file 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 (necessary for nofiller1 flow)
PGMUX2_X1.gds	Laker generic pass-gate MUX2 design for PG ESD spacing flow
PGMUX2_X1.sp	Laker genetic pass-gate MUX2 design for PG ESD spacing flow
source/liberty	OpenCellLibrary Liberty 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
Divide_vlog.f	List file for Import Verilog
CPU.sp	CDL netlist of CPU example
CPU.v	Gate level Verilog netlist of CPU example
CPU_vlog.f	List file for Import Verilog
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
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_oa_def.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

Taking the following steps will 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
   ```

4 Technology File Preparation

The two main parts of routing technology should be defined for Laker CDPR. 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 styles 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

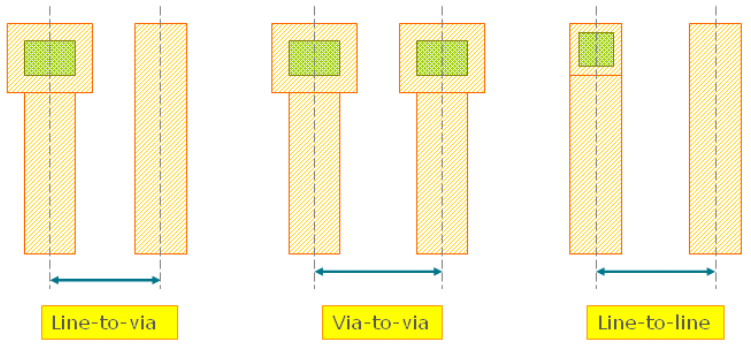

Figure 3: **Pitch Value**

- The pitch offset value is usually half of 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 height is 10 ~ 12 tracks of horizontal track (wire-wire spacing). The width of the smallest driving inverter is 2-track of vertical track (wire-spacing).

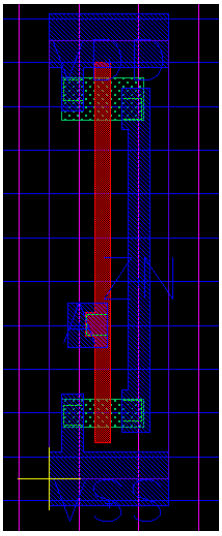

Figure 4: Routing Pitch and Offset

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

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

4.1 Lab-4A: Routing Rules

Name:	Specify a valid routing layer. Poly layer can be defined for 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 the 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, a ratio of {1 8} for metal1 layer means the horizontal cost is smaller. It results in an X-preferred direction. A ratio of {8 1} for metal2 layer results in a 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  }
```

4.2 Lab-4B: Foundry DRC Rules

Advanced process requires new rules for width, spacing, fat metal spacing, end of line, enclosure edge, minimum area, minimum enclosure, minimum edge, minimum cut, and maximum stack via.

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 } } } }	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 ;

Refer to the *Laker Command Reference* manual for more details on the technology file preparation of poly routing.

5 Library Preparation

Physical Layout, port information, cell boundary (PR boundary for PR tools) and SITE (minimum placement unit) should be available for custom cell or standard cells for running CDPR. The following Laboratories show you how to build this data.

5.1 Lab-5A: 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.

5.1.1 Import Stream Files

1. Open the *Import Stream* form by invoking **File → Import → Stream**.
2. Select an Open Cell Library GDS file named OpenCellLibrary.gds.
3. Assign a Library name to the 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 the imported layout, text, and CellBdry layer.

5.1.2 Net/Port Extraction

Based on layout geometry and annotated text, connectivity information can be added by assigning net and port information on the 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**.

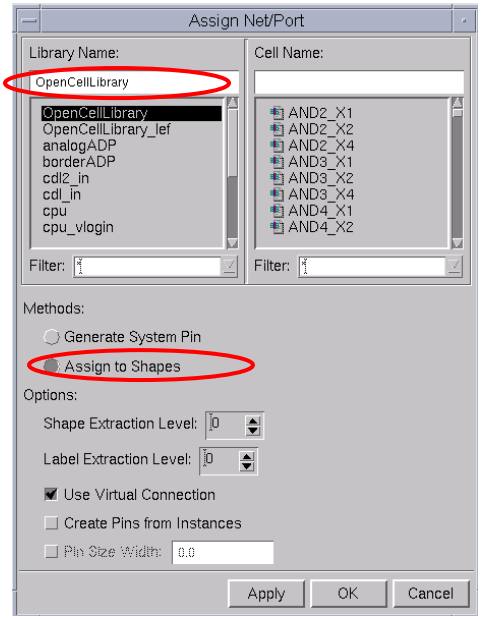

Figure 5: Assign Net/Port Form

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
}
```

5.1.3 Define Global Power/Ground Net

The Global Power/Ground (PG) net has an impact on several features such Flight Line and PG Router. 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
```

5.1.4 Abstract Creation (Optional)

This step is optional for third party tool by Export LEF if necessary. The Laker Custom Digital flow will generate hierarchical pin and blockage representation on-the-fly. It does not need abstraction preparation.

1. Perform abstract view generation to extract boundary and/or pin information (optional)
2. Invoke the **Library → Abstract Cell** command.
3. In the *Abstract Cell* form, enable the **Generate from Layer** option and select [v] **CellBdry** under the **Extract Rule for Boundary** section.

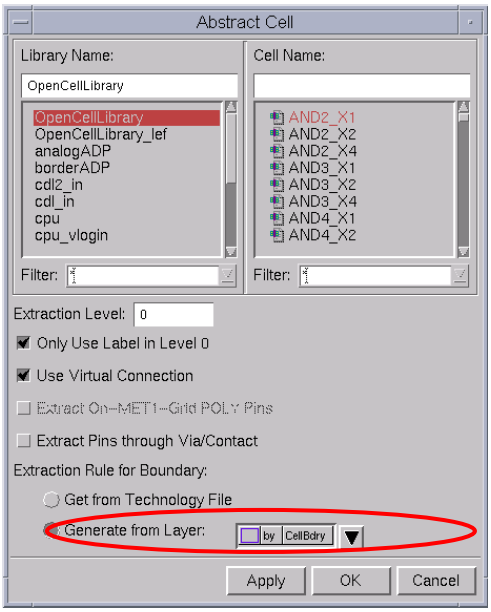

Figure 6: **Abstract Cell Form**

5.1.5 Define Site Information

Library level SITE information is strongly recommended for SITE related applications (e.g. row placer, row snapping, cell legalization, and filter insertion).

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

- Refer to the 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 the minimum dbResolution for custom cells which have no strict vertical track planning.

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

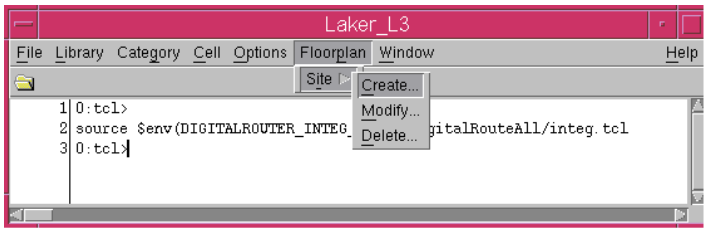

Figure 7: Select Floorplan → Site → Create

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

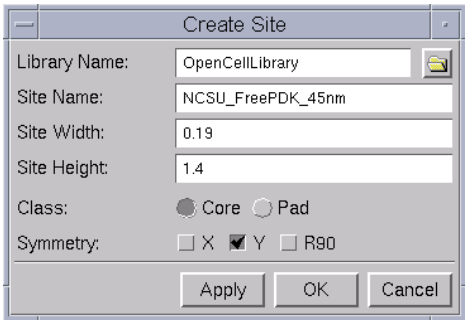

Figure 8: Create Site Form

5.1.6 Update Macro Cell Property

An additional Macro cell property defined in an 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 a library level update.

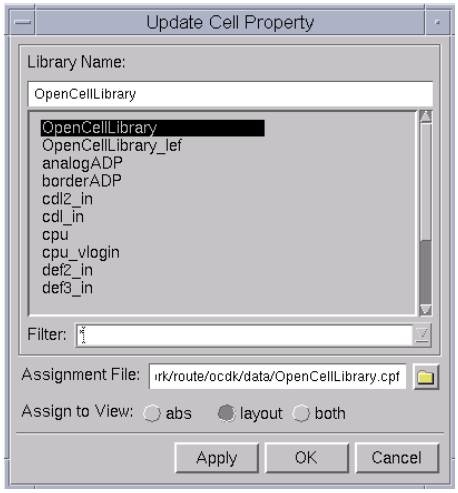

Figure 9: Update Cell Property Form

2. In the *Update Cell Property* form, do the following:
    a. Select **Library Name** as OpenCellLibrary
    b. Set **Assignment File** to OpenCellLibrary.cpf
    c. Check **Assign to View** to **layout** for GDS flow only
    d. Click **OK** to finish the 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"
}
```

5.2 Lab-5B: Verilog and SPICE Library Files

Verilog and SPICE library netlists including cells and their port information are useful in building a logic view of custom or standard cells.

In general, a Verilog library netlist file provided by a library vendor is a behavior model of modulea and primitives for post-layout simulation. These behavior sections are not suitable for several backend tools. For Cell level applications, a Verilog library netlist defines the module port name and direction.

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

How do you create a Verilog library netlist including the module port name and direction based on a Verilog library behavior model?

- It can be easily done by utilizing 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.

6 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.

There are two ways to build a logic view:

- Design is saved in the schematic view of Laker ADP:
  Logic view: Laker ADP schematic + Laker Expand Schematic (**File→Expand Schematic**)
- Design is in CDL or Verilog netlist:
  Logic view: **Import→ CDL In**, **Import→Verilog**

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

Refer to reference manuals for more detailed information for import CDL and Verilog.

The following tutorials will guild you the design preparation flow for CDL netlist and Verilog netlist.

6.1 Lab-6A: 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.

6.1.1 Import CDL Files

1. Define a cell library in a library mapping path by invoking **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 provides a more readable schematic for manual selection, and also provides a more meaningful topology relationship for certain 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.

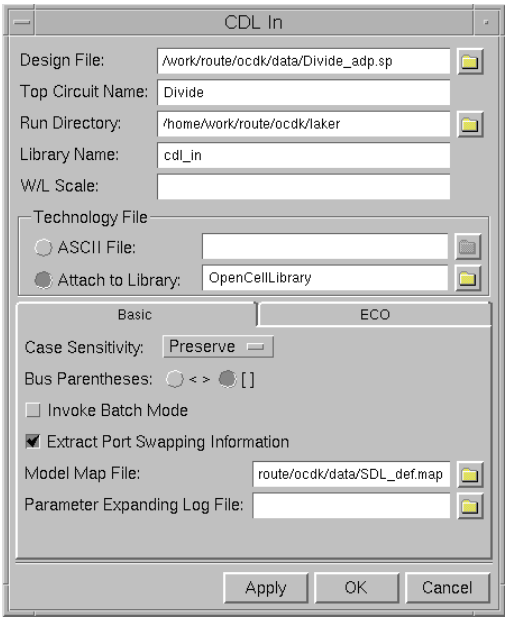

Figure 10: CDL In Form

Here is an example of a model map file to enable cell mapping and removal of the prefix "X" for a CDL file format.

```
[MAP]
X  AND2_X1  OpenCellLibrary  AND2_X1
X  AND2_X2  OpenCellLibrary  AND2_X2
…
…
X  XOR2_X1  OpenCellLibrary  XOR2_X1
X  XOR2_X2  OpenCellLibrary  XOR2_X2
X  TAPCELL_X1    OpenCellLibrary  TAP_X1

[ELEMENT_PREFIX]
X *        X

[GLOBAL_NET]
VDD          p
VSS          g
```

6.2 Lab-6B: 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.

6.2.1 Import Verilog Files

The Laker system supports a 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.

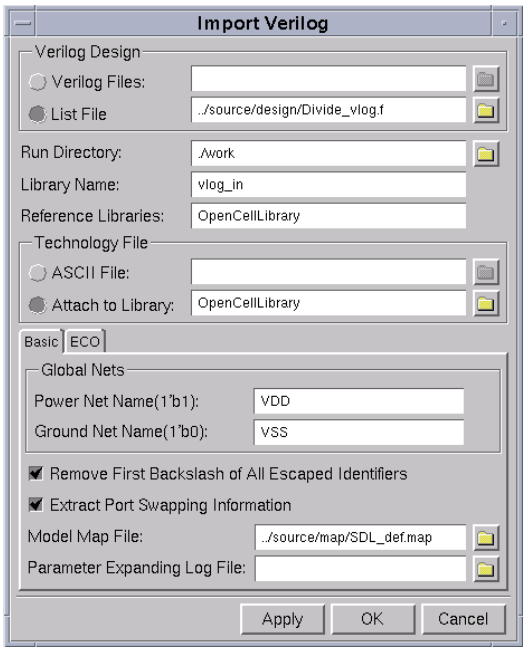

Figure 11: Import Verilog Form

3. In the *Import Verilog* form, do the following:
   a. Set **Verilog Design → List File** to ../source/design/*Divide_vlog.f*

      The following is an example of a Verilog list file:
      ```
      [vlog.f]
      ../source/Divide.v
      ../source/OpenCellLibrary_dummy.v
      ```
   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]: The Laker OA version has a new model map file section to support MCells, Tcl PCells, and PyCells in a general SDL flow. Please refer to the ***Limitations and Known Problems*** section for details. A new example is also available in *source/SDL_oa_def.map* for Laker OA flow.

7 CDPR Tutorials

In the following sections, we will guide you through the typical Laker™ CDPR flow starting from floor planning, power planning, cell placement, and finishing digital routing on top of the SDL flow. For LEF/DEF flow, refer to the *Laker CDPR LEF/DEF Tutorial.*

Recommended Flow is covered by the following topics:

- ***錯誤! 找不到參照來源。***
- ***錯誤! 找不到參照來源。***
- ***錯誤! 找不到參照來源。***
- ***錯誤! 找不到參照來源。***
- ***錯誤! 找不到參照來源。***
- ***錯誤! 找不到參照來源。***

8 Floorplan

8.1 Lab-8A: 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 the general Laker SDL database either by CDL-In or by Expand Schematic. A CDL-In case will be used to demonstrate..

8.1.1 Routing Resource Plan

In general practice, the 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 the 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.

Alternatively, you can use higher utilization for a compact design in a 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]: The site utilization rate will be higher than area utilization rate if row space is reserved for limited route layer in a 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 or physical only cells.. Therefore, the final utilization rate will be higher than the initial utilization rate.

8.1.2 Set Routing Layers

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

When 4 metal routing layers (metal1 ~ metal4) are used, the 6 non-available metal routing layers (metal5 ~ metal10) have to be disabled in the Laker technology file.

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

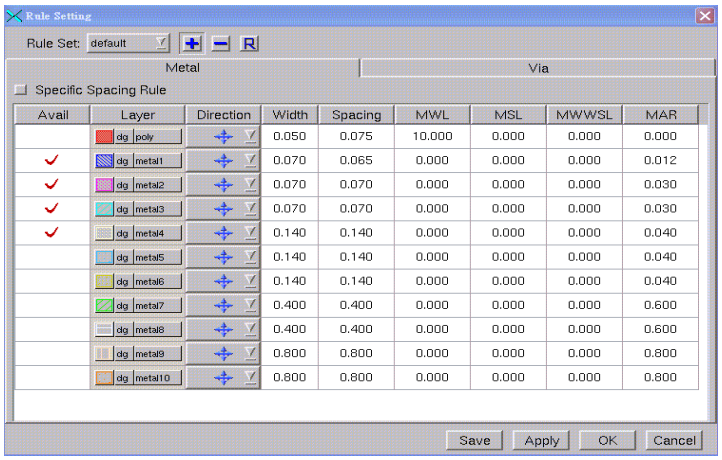

Figure 12: Rule Setting Form

8.1.3 Initialize Area Estimation

After the CDL is imported, both a 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

The 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 the view level and stretch the softBdry layer of the lower level block.

Stretching the softBdry layer of the cell level (or Edit-In-Place) will not keep the reserved area size.

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 pins without a dedicated layer assignment. It is created for manual or automatic pin planning.

8.1.4 Create Row

8.1.4.1 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. Invoke the **Placer → Create Row** command.
2. 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 draw a rough equivalent area size in polygon shapes. For example, a rectilinear polygon is created by referring to the SoftBdry.
    d. Set **Row Spacing** to 0.0 for a channel-less floor plan.

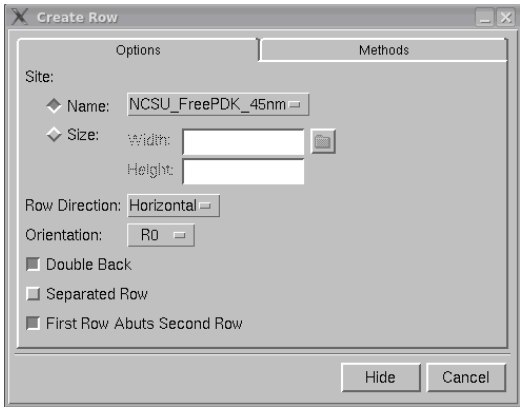

Figure 13: Create Row Form

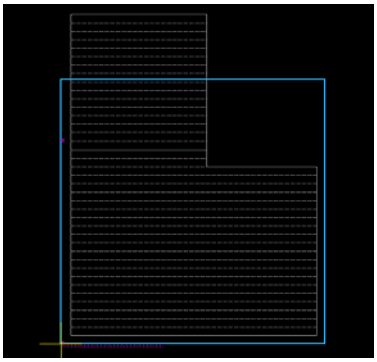

Figure 14: Created Row Area

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

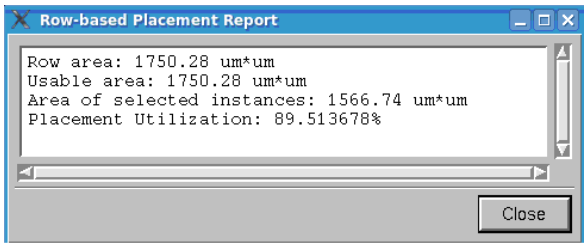

Figure 15: Row-based Placement Report Form

6. Enlarge the created row area by invoking **Placer → Stretch Row** or **Placer → Configure Row** to adjust high utilization to a reasonable rate.

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

8.1.4.2 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.

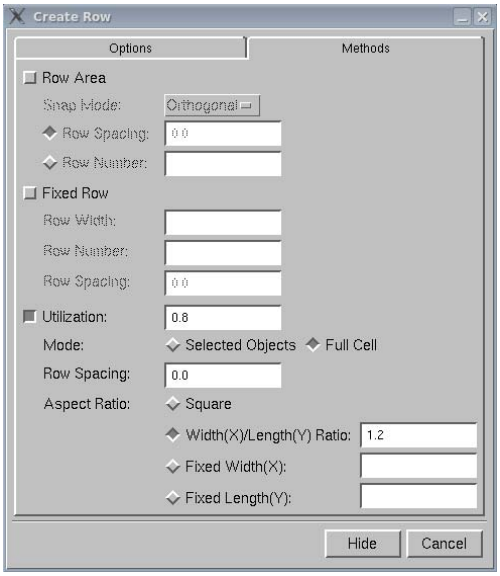

Figure 16: Create Row Form

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

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

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

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

8.1.5 Create Cell Boundary

You can create CellBdry manually by setting CellBrdy layer as the active layer and invoke **Create → Rectangle** to create it. A customized Tcltk script, **createCoreBdry4RowArea.tcl**, is provided to create a CellBdry based on the planned core area, In the *Main* window, source the Tcltk script.

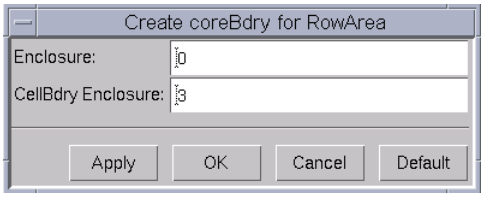

Figure 17: Create coreBdry Form

CoreBdry is a layer created for legacy supported features (if necessary).

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

8.1.6 Initial Pin Assignment

At the very beginning stage, pin constraint can be assigned from top-down or bottom-up floor plan tasks. In the 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. The following pin constraint file is an example on how to simply arrange bus signals on edge and evenly distribute before cell placement. As 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 }
}
```

8.1.6.1 Create Soft Pin

If you removed the pink SoftPin in previous steps, you can re-generate all of them by invoking **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 **Fixed** by invoking **Query → Connection → Pin Status**.

8.1.6.2 Dump Template of Pin Constraint

Dumping the template from a 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.

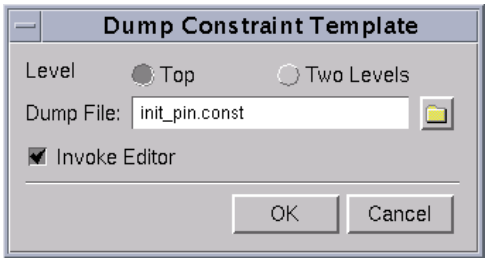

Figure 18: Dump Constraint Template Form

8.1.6.3 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.

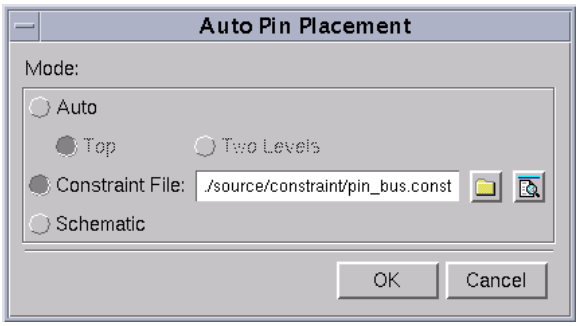

Figure 19: Auto Pin Placement Form

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 a top down floor plan or after row placement is done.
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 honors 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 at a later time.

9 Pre-Placement

9.1 Lab-9A: Add Physical Only Cells (Optional)

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

9.1.1 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, and GA spare cells. 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 in a regular format within a defined distance to cover all placeable 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 a placeable row area to keep the regularity of layer patterns to minimize the 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 rings.
- Antenna cell: Antenna diode to provide additional diffusion protection for an antenna violation fix.
- Decoupling-cap cell: Power decoupling cells to reduce the current surge and voltage drop.
- GA spare cell: Programmable gate array spare cells for flexible post-silicon ECO.

9.1.2 Add End Cap

TAPCELL_X1 is used for both well tap cells 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.

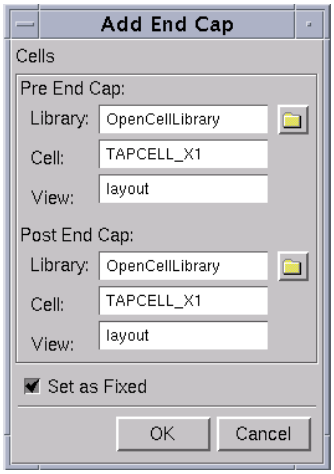

Figure 20: Add End Cap Form

9.1.3 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.

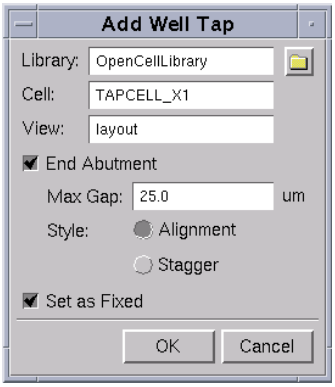

Figure 21: Add Well Tap Form

9.1.4 Mark Placement Status of Physical Cells

You can mark the 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 or Legalization, 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.

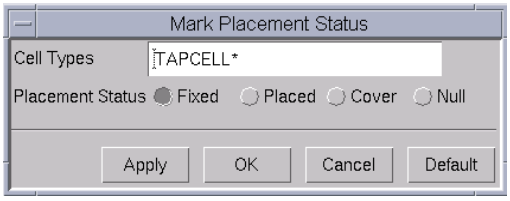

Figure 22: Mark Placement Status Form

10 Power Plan

10.1 Lab-10A: Create Core Ring and Stripes

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

10.1.1 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 pad rings
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 OA2010.08 and later versions.

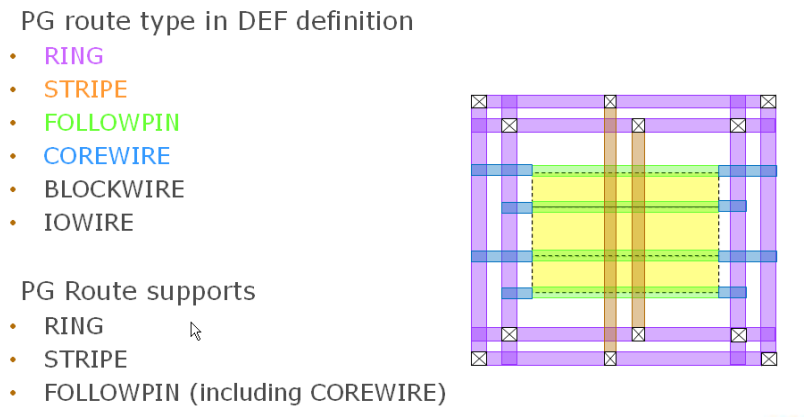

Figure 23: PG Routing for RING, STRIPE, FOLLOWPIN, and COREWIRES

10.1.2 Create Core Rings

Before creation of core rings, you have to plan ring structure, routing layer, width and offset spacing to the core area. In the meantime, you also have to check the total reserved distance between CellBdry and Row Area. If its value is not large enough to create the whole ring structure, you need to adjust the floor plan in advance.

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

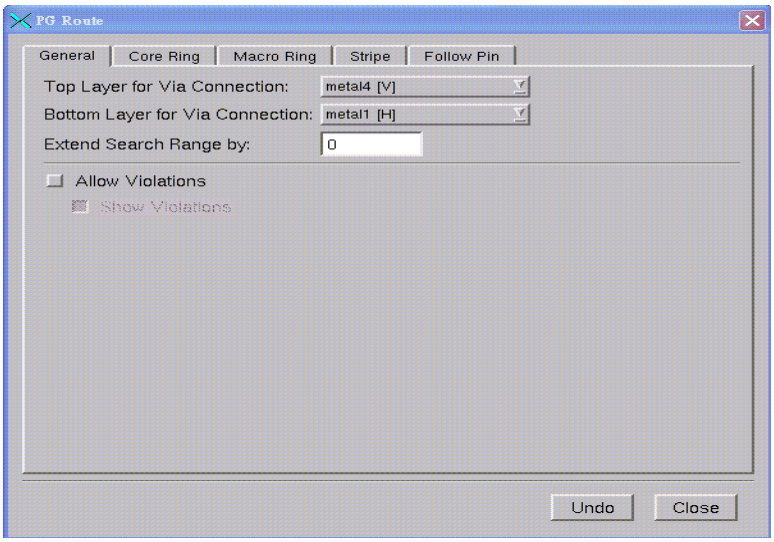

Figure 24: PG Route Form, General Tab

In the **General** tab, you can set the top and bottom routing metal layer as follows:

- Top Layer for Via Connection: metal4
- Bottom Layer for Via Connection: metal1

2. Switch to the **Core Ring** tab.

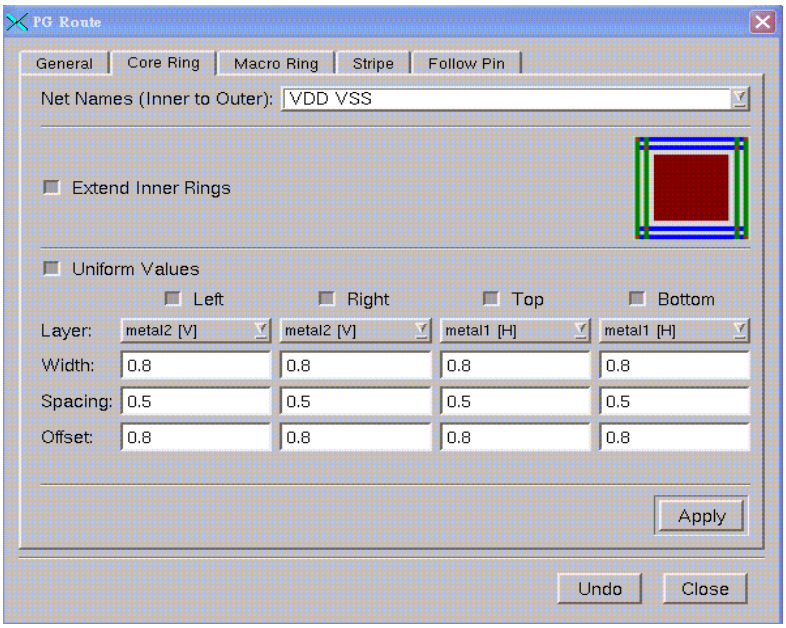

Figure 25: PG Route Form, Core Ring Tab

3. Fill in power and ground net names in the **Core Ring** tab: *VDD VSS*
4. Fill in your planned **Layer**, **Width**, **Spacing** and **Offset** for each Left, Right, Top, and Bottom edges around Row Area. Turning on the **Uniform Values** button can set the same Width, Spaing and Offset values for all sides.

   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 Row Area in a polygon shape.
5. Click **Apply** to realize core ring creation.

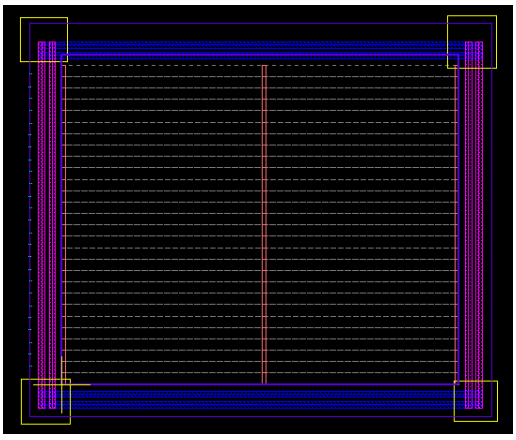

Figure 26: Created Core Ring

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

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

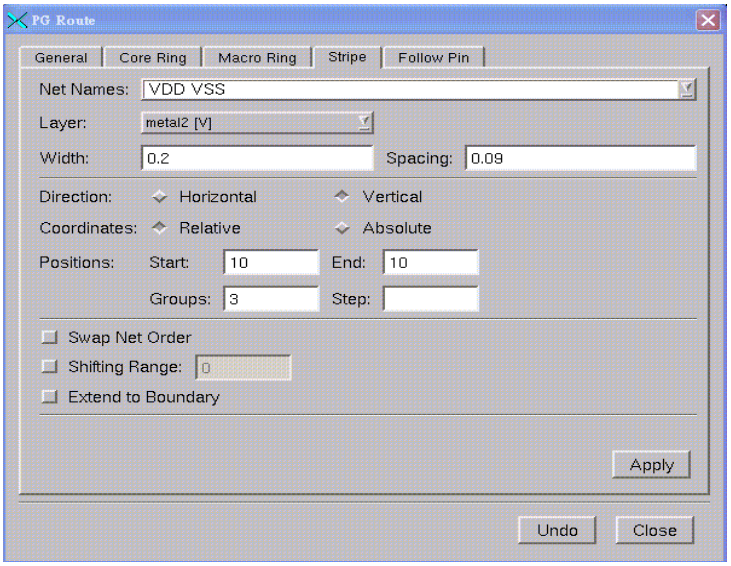

Figure 27: PG Route Form, Stripe Tab

1. Switch to the **Stripe** tab again if it is closed.
2. Fill in your planned **Net Names**, **Layer**, **Direction**, **Shifting Range**, **Start**, **End**, and **Groups**.
3. Click **Apply** to realize the creation of stripes.

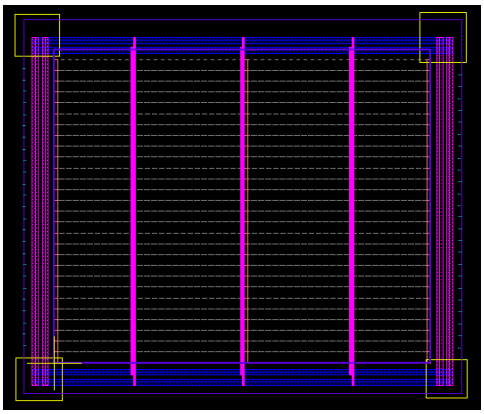

Figure 28: Created Stripe

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

11 In-Placement

11.1 Lab-11A: Row Placement

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

11.1.1 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, and placement effort.

An initial placement constraint is provided.

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

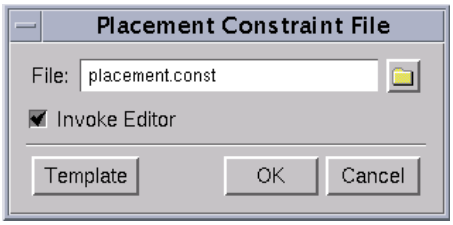

Figure 29: Placement Constraint File Form

2. Click **Template** to generate a template for 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
```

11.1.2 Assign Placement Constraint File

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

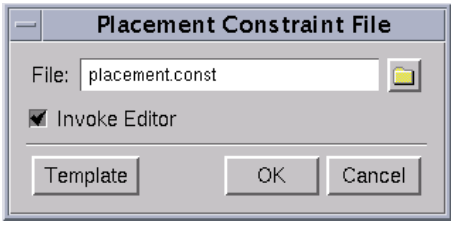

Figure 30: Placement Constraint File Form

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

11.1.3 Create Placement Blockage

To protect some regions from cell placement, add the PlacementBlockage (249:241) layer in the floor plan layout.

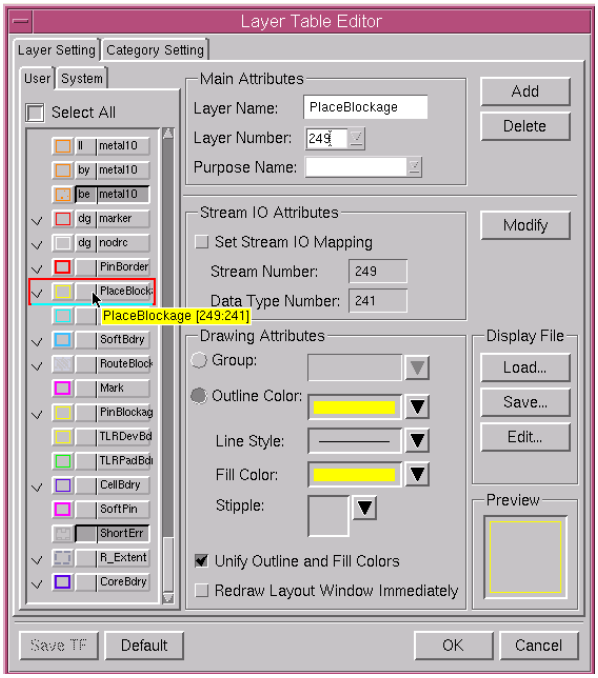

Figure 31: Layer Table Editor Form

11.1.4 Row-based Placement in SDL

1. Click the **Select All** icon in the design hierarchy browser pane to select all objects as 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. 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 the placer engine.

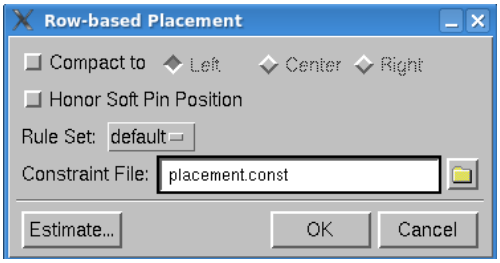

Figure 32: Row-based Placement Form

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

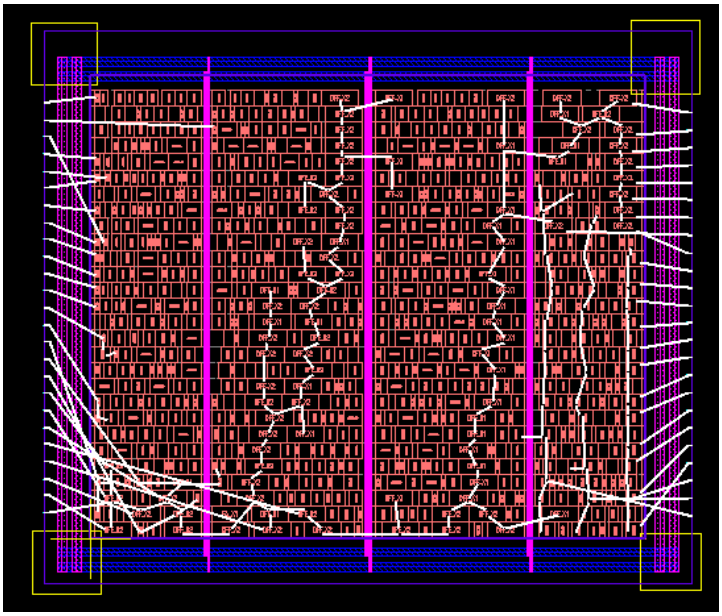

Figure 33: Row-based Placement

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

12 Post-Placement

12.1 Lab-12A: Post-Placement

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

12.1.1 Pin Optimization

This is an optional step and can be used 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 the net weight cost of I/O nets in the 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 the current pin order

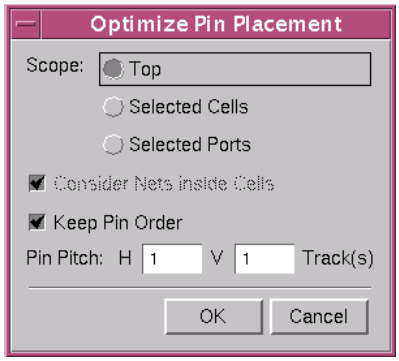

Figure 34: Optimize Pin Placement Form

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

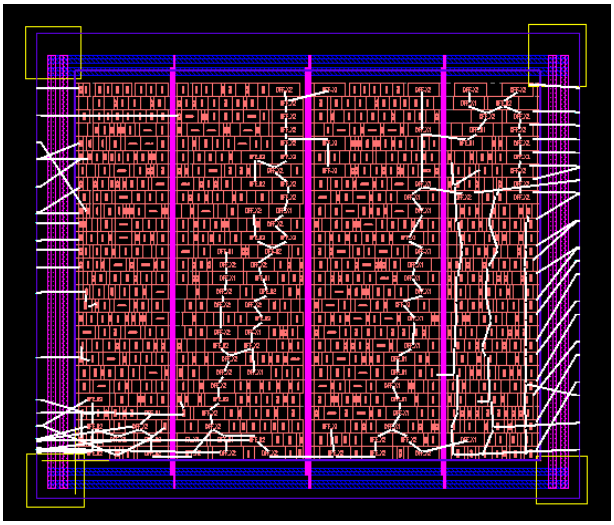

Figure 35: Optimized Pin Placement

3. Click the **Undo** button to check the difference between results without keeping the 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.

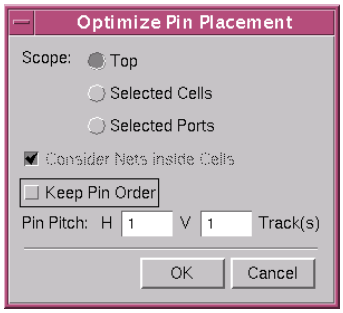

Figure 36: Optimize Pin Placement Form

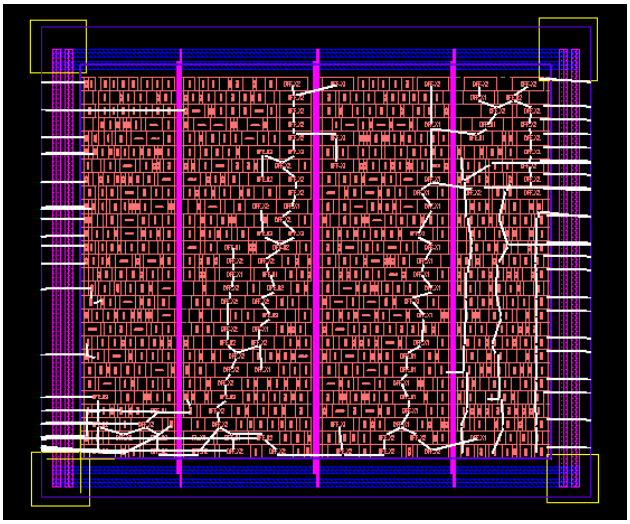

Figure 37: Optimized Pin Placement

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

12.1.2 Check Placement

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

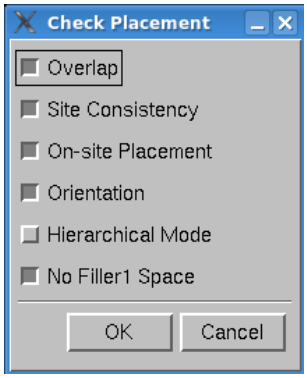

Figure 38: Check Placement Form

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

12.1.3 Check Spare Cell Placement

Silicon ECO spare cells are evenly spread during row placement. These spare cells can be highlighted to make sure the 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_**

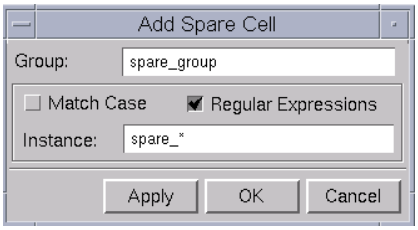

Figure 39: Add Spare Cell Form

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

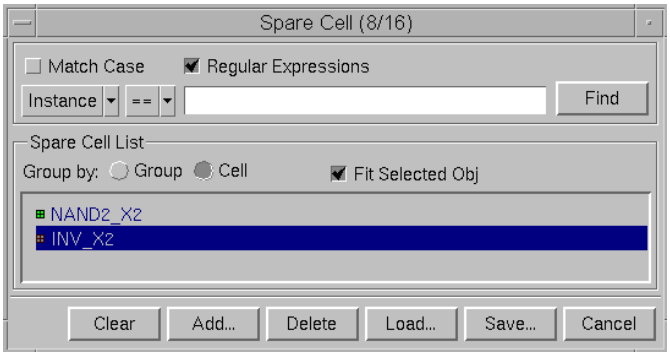

Figure 40: Spare Cell Form

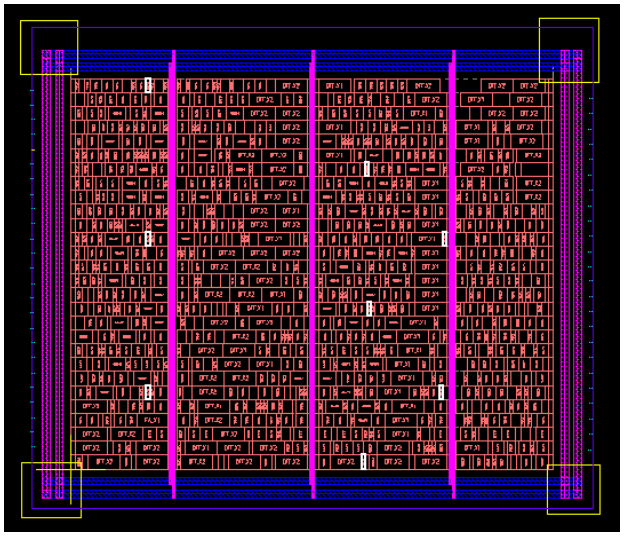

Figure 41: Spare Cells

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

12.1.4 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 the 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*.*

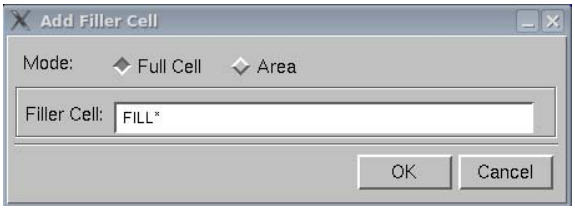

Figure 42: Add Filler Cell Form

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

12.1.5 PG Connection of Physical Cells

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

1. Invoke the **Router → Digital Router → Assign Instance Port to Net** command.
2. Fill in instance names, power, and ground net/port names.
3. Click **OK** to assign global power and ground nets to a physical cell by cell type.

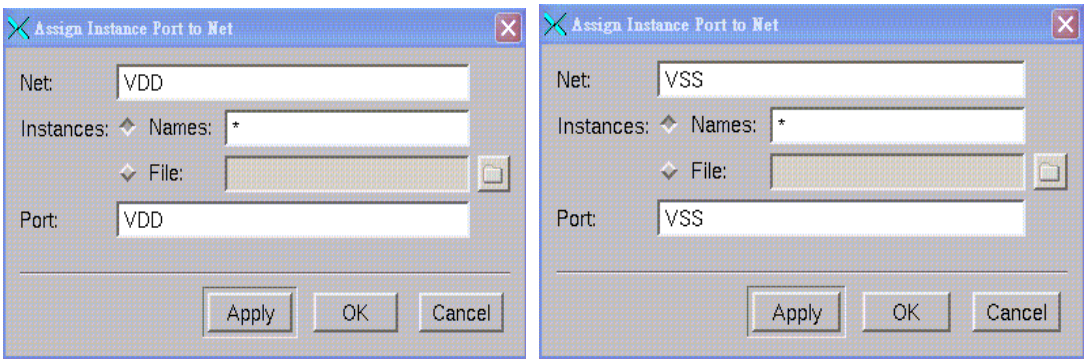

Figure 43: Assign Instance Port to Net Form

12.1.6 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 signal routing is started.

1. Invoke the customized **Router → Digital Router → PG Route** command.
2. Switch to the **Follow Pin** tab.

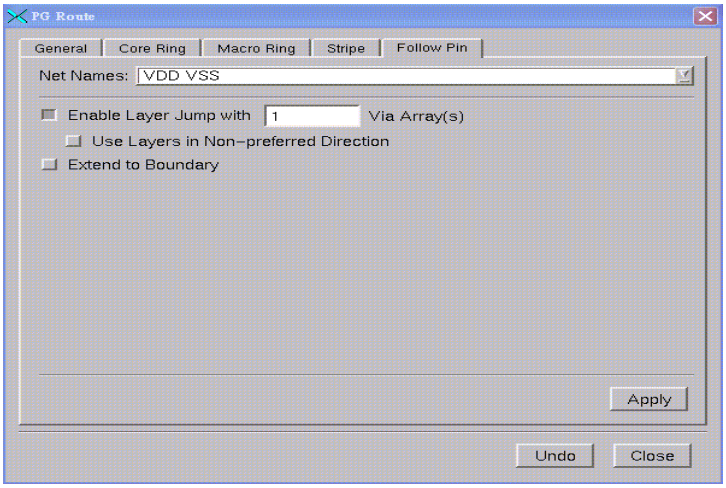

Figure 44: PG Route Form, Follow Pin Tab

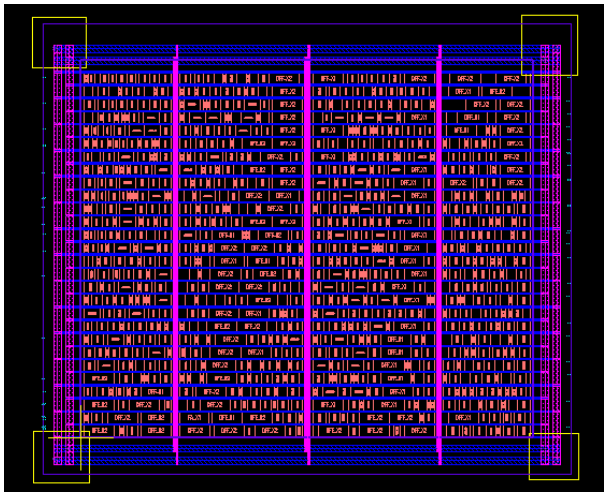

Figure 45: Inserted Follow Pins

13 In-Route

13.1 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 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.

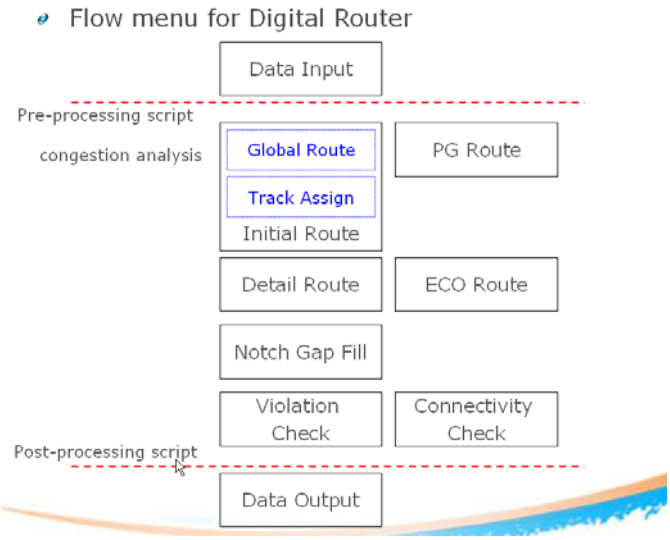

Figure 46: Introduction of In-Route

13.2 Lab-13A: Digital Route

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

13.2.1 Digital Route

1. Invoke the **Router → Digital Router → Digital Route** command.
2. In the *Digital Route* form, set up global options as default under the **General** tab.

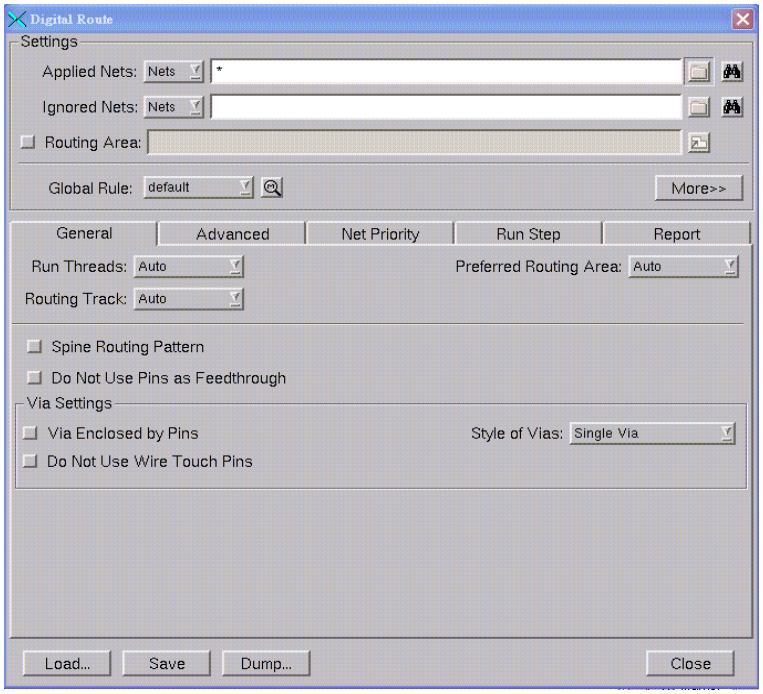

Figure 47: Digital Route Form, General Tab

3. On the *Digital Router* form, switch to the **Run Step** tab.
    a. Enable **Initial Route** and **Detail Route** procedures.
    b. Enable the **Post Optimization** procedure as the demo case is small.
    c. Enable the **Double Via Insertion** procedure.
    d. Enable **Notch Gap Filling** procedure.

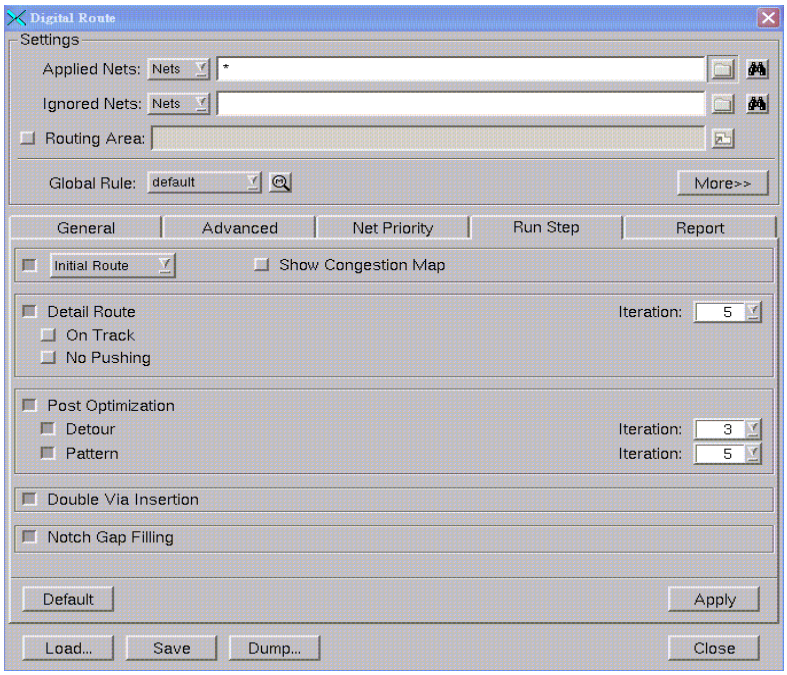

Figure 48: Digital Route Form, Run Step Tab

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

    The **Post Route** option on the **Auto Route** tab is disabled by default to enable a fast routing result for debugging once a routing problem is detected.

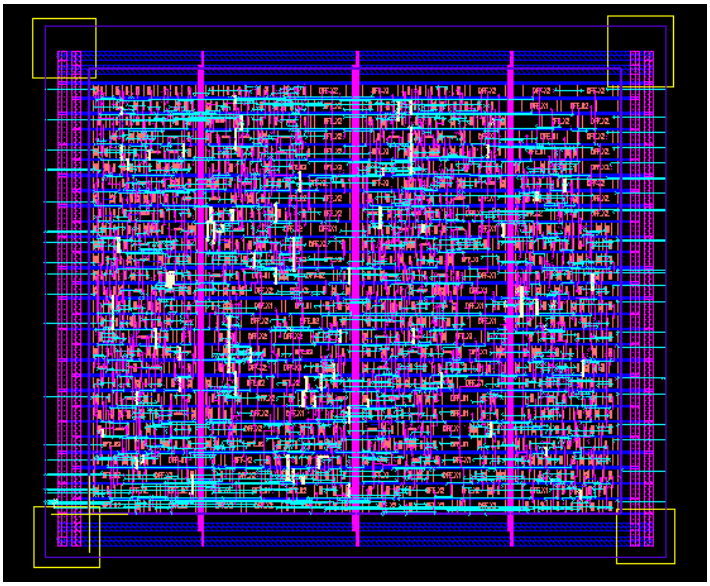

Figure 49: Routing Results

13.2.2 Routing Quality of Result

1. Switch to the **Report** tab for reporting several routing quality results.

    a. Enable **Route Information** to report wire and via statistics. Double cut rate per layer can be reported this way.
    b. Enable the **Check Connectivity** option to check routing connectivity.
    c. Enable the **Jog Information** option to get jog counts.
    d. Enable **Violations** and its sub-option **Show Violations** to bring up the *Error Viewer* form if any DRC violations are found.

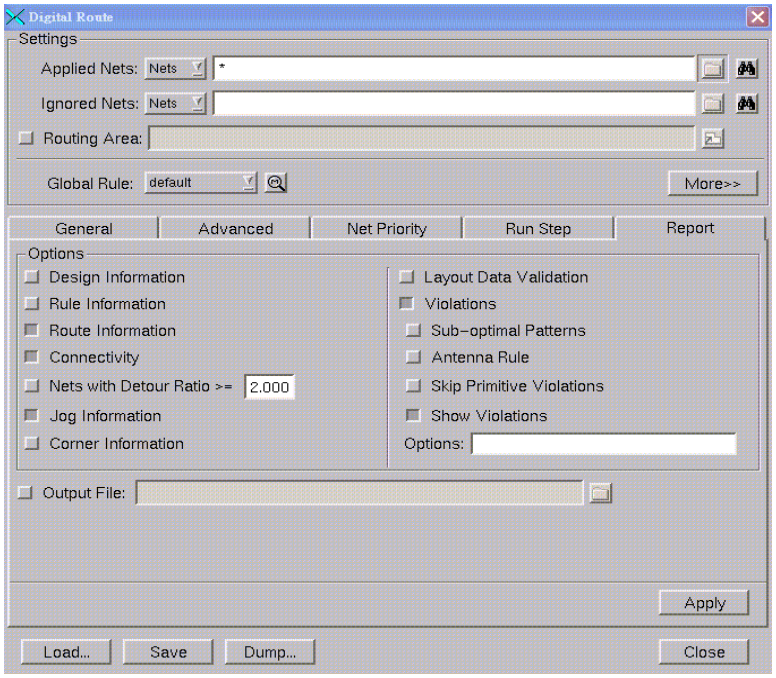

Figure 50: Digital Route Form, Report Tab

2. Click **Apply** to complete router verification tasks.

3. Select the **CustomDigital → Route → Close** command to close the tool box.

14 Limitations and Known Problems

14.1 Laker OA Flow

Default name space of bus representation is an angle bracket in Laker OA. For example, `A[7:0]` will become `A<7:0>`.

14.2 Laker Import Verilog Flow

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

14.3 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 }
  ```

- Obsolete 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.

Revision History

Revision	Date	Description	
7.2	02/05/13	Updated the footer information.	MY
7.1	12/26/12	Updated the header and footer information.	MY
7.0	03/28/11	Split the Custom Digital Tutorial as CDPR LEF/DEF Tutorial and CDPR_SDL Tutorial, and updated the tutorial contents. Based on Laker OA2011.03.	HLH
6.0	12/16/10	Added an Overview section.	Rich Morse
5.0	11/02/10	Updated “Lab-1B: Foundry DRC rules”.	HSW
4.0	09/09/10	Updated Model Map file section for Laker OA2010.08. Add LEF/DEF flow for Laker OA.	HSW
3.0	08/05/10	Updated tutorial materials. Removed Tcl scripts which are supported by Laker2010.07 features. For example, Pin Placer, etc. Updated Laker OA incremental Tech. Updated notch gap fill.	HSW
2.0	06/11/10	Changed DigitalRouteAll to CustomDigital. Changed routing track definition of OpenCellLibrary.tf for better hierarchical layout implementation.	HSW
1.6	04/16/10	Added a limitation of PG routing with read only LEF master library.	HSW
1.5	04/13/10	Added details to Introduction, modified Verilog Import, and 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 a technology file preparation section. Based on Laker 2010.03.	HSW
1.0	3/1/10	Initial release. Based on Laker 2010.03.	HSW