SiGe Mixed Signal BiCMOS7HP Design Reference Flow for Digital Macros

Cadence Synthesis Place & Route and Physical Verification

Prepared By
Cadence Design Systems, Inc.

TABLE OF CONTENTS

1.0 Introduction

This SP&R flow was developed to address the unique features of the IBM SiGe process.There are numerous ways a design may be conveyed from RTL through GDSII, so we have described a typical base flow – you can make modifications to this flow as required. For each step in this flow, we detail the required inputs and resulting outputs, command files, setup requirements, and recommended practices. We have also included descriptions of any utilities available to automate the steps in the SPnR flow, how and where they are used, and information on how to get them.

This flow was automated, in large portion, by scripts. Each step in the design flow can be manually entered by individual instructions. This is obviously error prone and does not lend itself to easy repeatability. In this document the scripts that are used will be disected and explained in terms of the individual command actions.

2.0 Infrastructure

Setting up the design database and following a canonized procedure will help in creating reproducible results. It is important to control the inputs and feedback results to achieve timing correlation. This section outlines a suggested method of controlling the complete SP&R environment.

2.1 General Setup Files

You need to have certain files and libraries available before starting the SPnR flow. . The categories of these files and libraries along with their supported formats are illustrated in the figure below.

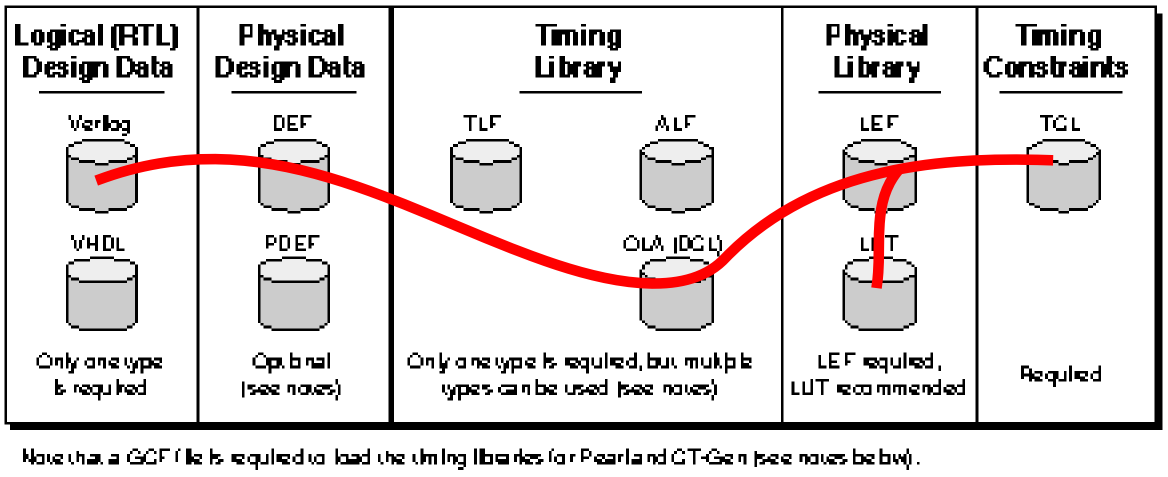

Logical (RTL) Design Data
Physical Design Data
Timing Library
Physical Library
Timing Constraints
Verilog
DEF
TLF
ALF
LEF
TCL
VHDL
PDEF
OLA (DCL)
LUT
Only one type is required
Optional (see notes)
Only one type is required, but multiple types can be used (see notes)
LEF required, LUT recommended
Required
Note that a GCF file is required to load the timing libraries for Pearl and CT-Gen (see notes below).

Figure 1: Initial files and libraries required by the SPnR flow

Each of these categories is discussed below for the general application. For this flow we will utilize the item connected by the line in the above figures. Other data will also be needed and will be discussed in the appropriate sections to follow.

2.1.1 Logical (RTL) Design Data

The original logical design data is presented in register transfer language (RTL) format and can comprise a mixture of high-level statements, gate-level netlists, and macro instantiations.

Supported Formats:	Verilog and VHDL
Source:	Generated manually or from design capture tools, including block diagram, schematic, state diagram, flow chart, truth table, and textual HDL editors.

2.1.2 Physical Design Data

Physical design data can be represented using the de facto standard DEF or industry standard PDEF formats. (Note that PDEF can only describe placement and bounding box data, while DEF can include additional information such as routing grids, power grids, pre-routes, and rows.)

Supported Formats:	DEF 5.1+, PDEF 2.0
Source:	Generated by a physical design tool such as a floor planner or a placement engine.

The floorplan for this flow is generated in Silicon Ensemble during the Cold Start flow. The details are covered in a susequent section.

2.1.3 Timing Library

A timing library includes all of the timing information associated with a particular manufacturing process. This flow uses the OLA library format, but it is possible to use a mixture of timing libraries if you wish. For example, OLA for standard cells and TLF for macro libraries.

Supported Formats:	ALF 3.0, OLA (DCL), TLF 4.3
Source:	Timing libraries are supplied by IBM.
Note:	If you do use multiple timing library formats within the flow, you must ensure that they are well correlated.

2.1.4 Timing Constraints

The timing constraints are presented to the SPnR flow in the form of a TCL file (this file is assumed to be named constraints.tcl for the purposes of this document).

Supported Format:	TCL
Source:	The initial constraints are generated manually or programmatically and are derived from the design's timing specification. (Note that a conversion utility that can convert Synopsys write-script into TCL is available).
Note:	Be sure to run check_timing to validate that your constraints are correct.

2.1.5 Physical Library

The physical library describes the physical characteristic for a particular standard cell library. The LEF abstracts provide information for placement and routng. This includes information like physical footprints, pin locations, etc.

Supported Formats:	LEF 5.2 or LEF 5.3
Source:	As for the timing libraries, physical libraries are generated by IBM.

Note:	There should be a one to one correspondence between the abstract views and the timing views.

2.1.6 Technology File

The technology file provides the process information necessary for routing. The technology file contains layer information supplying data about spacing and parasitic characteristics. Also contained in the technology file is information about vias, blockages and sites.

Supported Formats:	LEF 5.2 or LEF 5.3
Source:	
Note:	In order to achieve the most accurate results possible, it is important that the R and C values in the LEF be well correlated with the results from HyperExtract (HE).
Note:	Be sure to run check_library to verify that your physical and timing libraries match.

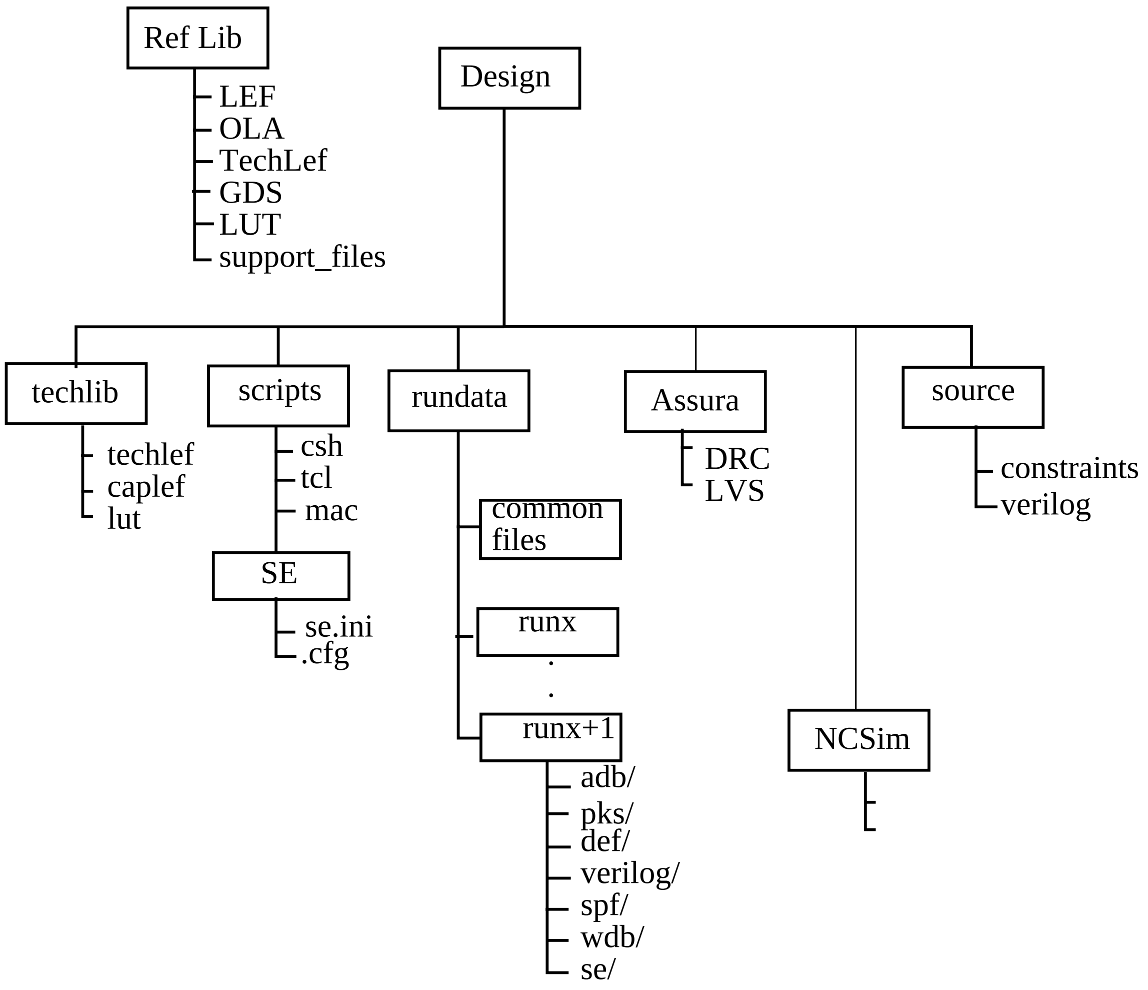
2.1.7 Layer Utilization Table

The Layer Utilization table is an ASCII text file used to provide information to the fast router in PKS. This information guides PKS in its initial estimation phase to use the specified percentages of each metalization layer for horizontal and vertical routes. For example: horizontal tracks = 35% on metal layer 1, 40% on metal layer 3, and 25% on metal layer 6; vertical tracks = 45% on metal layer 2, 45% on metal layer 4, and 10% on metal layer 6 (the actual percentages used will be based on empirical data gathered from previous designs).

Supported Format:	ASCII text (see notes below)
Source:	Based on empirical data gathered from previous designs.
Note:	The Layer Utilization Table is considered to be part of the physical library data and is required every time a LEF physical library is used.

3.0 Directory Structure

A standard directory structure is not required but is strongly recommended. Many different configurations are possible. In the example below the design specific files are separate from the reference libraries which are independent from the design. This section provides details about the directory structure and the general application of the files and utilities contained therein.



3.1 Reference Libraries

The reference files consist of all the files that define a technology and are supplied by IBM. They include timing and physical libraries and other technology files. Other essential files for the flow, but are technology independent, are also included in this area.

3.1.1 Timing Library

The timing library includes all of the timing information associated with a particular manufacturing process. This flow uses the OLA library format, but it is possible to use a mixture of timing libraries if you wish. For example, OLA for standard cells and TLF for macro libraries.

3.1.2 Physical Technology File

The technology file provides the process information necessary for routing. The technology file contains layer information supplying data about spacing and parasitic characteristics. Also contained in the technology file is information about vias, blockages and sites. The technology file in this area should not be modified.

3.1.3 Physical Cell Library

The physical library describes the physical characteristic for a particular standard cell library. The LEF abstracts provide information for placement and routng. This includes information like physical footprints, pin locations, etc. These abstracts can be modified to include antenna information.

3.1.4 Layer Utilization Table

The default Layer Utilization table is an ASCII text file used to provide information to the fast router in PKS. An updated LUT will be derived in the Cold-Start flow and then used in the general Sp&R flow.

An example LUT for a four-layer device technology is shown below. Further details can be found in the associated PKS application note Creating a Layer Utilization Table.

```
Layer_Usages () {
 Length_Range (0) {
 Utilization_Horizontal: "0.229 0.005 0.765 0.001" ;
 Utilization_Vertical: "0.002 0.585 0.001 0.412" ;
 Contact_Spacing_Vertical: "20" ;
 Contact_Spacing_Horizontal: "20" ;
 }
```

3.1.5 Support Files

Script files are used to manipulate the data files to add properties, components or create new tables for PKS. In this area the Perl script to add filler cells, generate the layer utilazation table and update the lef capacitances in the techfile are located. All of these are technology independent and can be used as standalone utilities.

Also included in this is the map file used when reading gds data into SE. This file is specific to the technology used.

The following is a listing of the files contained.

```
add_fillercells.pl*
gds2_se.map*
he2lef.awk*
lef_update.pl*
lutgen.pl*
```

3.2 Design Files

Design files constist of files that are specific to each design. They include design specific technology files, initialization and run scripts and files, the source data, and the run data.

3.2.1 Design technology files (techlib)

The design technology files are specific to each design and are used to capture detailed parasitic numbers tailored to each design. The files are generated by taking the design through the "cold start" flow, explained later in this document.

3.2.1.1 Physical Technology File

The physical technology file is a refined version of the reference library technology file. It contains parasitic capacitance numbers based on the results of a preliminary run of the design (see cold start flow).

3.2.1.2 Layer Utilization Table

The layer utilization table is a refined version of the reference layer utilization table. It is based on the results of a preliminary run of the design (see cold start flow).

3.2.2 Scripts

Initialization files take the form of scripts or other text files which setup each tool prior to or upon invocation. Run scripts are used to drive the tools throughout the design process. They are tool specific and allow the tools to be run in batch mode.

3.2.2.1 PKS tcl

PKS can be controlled through tcl scripts therefore the most common method of initialization is to run a tcl setup script imediately following invocation of the tool. If an OLA library is to be used however, a shell setup script is required.

3.2.2.1.1 OLA Setup

When using an OLA library certain shell environment variables must be set **prior** to invoking pks. The following example shows the csh commands used to set the variables:

```
#!/bin/csh
setoladir =            $LIB_DIR/ndr/SOL
setenvDCMRULESPATH     ${oladir}:${oladir}/%RULENAME
setenvDCMRULEPATH      ${oladir}/DCMinterface_SOL
setenvDCMTABLEPATH     ${oladir}/tables:${oladir}/%RULENAME
setenvDCMBOMPATH       ${oladir}
setenvDCMDoRangeCheck  0
setenvDCMWireLoadLevel 6mtwb
```

3.2.2.1.2 PKS Setup

PKS is usually initialized with a setup tcl script. This script can define global variables, load the appropriate libraries, and even define procedures that can be later called by a run tcl script. Some examples of tcl commands used to setup pks are shown below:

```
set_globalecho_commandstrue
set_globalfix_multiport_netstrue
set_globalline_length166

read_ola
set_dcl_calculation_mode -mode worst_case
set_operating_parameter -process 1.00
set_operating_parameter -temperature 100
set_operating_parameter -voltage 2.3
set lib_dir<path_to_lib_data>

read_lef $lib_dir/tech7hp.6mt.lef
read_lef_update $lib_dir/abstract7hp.lef
read_layer_usages $lib_dir/lut
```

3.2.2.1.3 PKS Run Scripts

PKS run scripts consist entirely of tcl scripts. These scripts can include specific pks command which accomplish the desired tasks or call procedures previously defined in a setup script.
Examples of the commands typically used are shown later in the sections which describe the flow.Silicon Ensemble

3.2.2.2 Silicon Ensemble Setup

Silicon Ensemble is initialized by a file named se.ini. This file must be present in the run directory from which se is invoked. This file contains everything from color tables for layers to special variables for the router. The complete se.ini used for the SiGe flow is shown in Appendix A. Amongst the color and stipple patterns in this setup file are some variables that are crucial for the IBM process. The important variable are shown in a segment of the se.ini below.

```
set v WROUTE.TOPLAYER.LIMIT 5 ;
set var output.def.spnet.wildcard true ;
SET VAR PLAN.LOWERLEFT.ORIGIN TRUE           ;# Sets the origin of the
                                         design to be
                                             # located on the Lower left
                                         corner.
SET VAR INPUT.VERILOG.POWER.NET ’VDD!’        ;# Set verilog power net
                                         name to vdd
SET VAR INPUT.VERILOG.GROUND.NET ’GND!’       ;# Set verilog ground net
                                         name to vss
SET VAR INPUT.VERILOG.LOGIC.1.NET ’VDD!’      ;# Set verilog tie ups to
                                         net name to vdd
SET VAR INPUT.VERILOG.LOGIC.0.NET ’GND!’     ;# Set verilog tie downs to
                                         net name to vss
SET VAR INPUT.VERILOG.SPECIAL.NETS ’VDD! GND!’;# Set verilog special net
                                         names
                                              # to vdd and vss
SET VAR FROUTE.MANUFACTURING.YGRID 2         ;# Set the Y manufacturing
                                         grid for Wroute
SET VAR FROUTE.MANUFACTURING.XGRID 2         ;# Set the X manufacturing
                                         grid for Wroute
SET VAR QPLACE.LLC.PREWIRE.KEEPOUT FALSE      ;# Prevents Qplace from
                                         placing cells
                                              # under the prewires.
SET VAR SROUTE.VIARULE.OVERHANG.COMMON TRUE   ;# Forces the power router
                                         to consider overhangs
                                          # against the layer direction.
                                         Used to prevent
                                           # problems with M1 followpins
                                         when the gap
                                             # between M2 VDD! and GND!
                                         stripes is 0.4
SET VAR SROUTE.VIA.SNAPMANUFACTURINGGRID TRUE ;# Snap cuts on via rule
                                         generate to the
                                              # manufacturing grid.
SET VAR SROUTE.STRIPE.SNAP.RGRID "GRID"      ;# Snap the stripes to the
                                         routing grid.
SET VAR PLACE.LLC.PREROUTEDWIRECHECK FALSE    ;# Allows filler cells to be
                                         inserted
                                              # under the stripes.
SET VAR WROUTE.ALLOW.PORT.SHORTS FALSE      ;# Provides more flexibility
                                         to Wroute used
                                             # due to number of offgrid
                                         pins.
SET VAR INPUT.DEF.ECO.TRYNETRENAME TRUE     ;# Used during ECO to try to
                                         preserve clock tree cells.


# Silicon Ensemble environment variables ;
set v  db.design.dir      "dbs" ;
# ;
# Set relative sizes to multiple of metal2 rGrid pitch ;
# NOTE: This values need to be adjusted depending on design sizes.
#
set v draw.size.small           9600 ; #  120 tracks ;
set v draw.size.medium          3200 ; #  400 tracks ;
set v draw.size.big           160000 ; #  2000 tracks ;
set v snapRadius.small            20 ; #  1/4 track ;
set v snapRadius.medium           80 ; #  1 track ;
set v snapRadius.large           400 ; #  5 tracks ;
# ;
# Silicon Ensemble floorplan variables - recommened  ;
set v  plan.rgrid.M1offset           40 ;
set v  plan.rgrid.M2offset           40 ;
set v  plan.rgrid.M3offset           40 ;
set v  plan.rgrid.M4offset           40 ;
set v  plan.rgrid.M5offset           40 ;
set v  plan.rgrid.M6offset          256 ;
# ;
# Silicon Ensemble route variables - recommened  ;
# NOTE: With autoAbgen models, none or very little performance loss in
                                          offgrid mode. ;


set v  groute.Allow.OffGrid.PinAccess true ;
set v  froute.Allow.OffGrid.PinAccess true ;
set v  froute.Avoid.OffGrid.Blockage  true ;
set v  froute.Build.OffGrid.SPins     true ;


#   set v  froute.allow.pinaccess.atblockageedge true ;
# ;
# Silicon Ensemble verification variables - recommened ;
set v  verify.geometry.allow.same.cell.violations   true ;
set v  verify.geometry.report.samenet     false ;
```

3.2.2.2.1 SE Run Macros

Silicon Ensemble is driven by macro scripts. Each design must have it’s own set of customized scripts which accoplish the desired tasks. An example of a SE macro is shown below.

Note that these macros may be nested so that a top level macro can be developed to control the flow of subsequent macros. This can be seen in the example in the Appendix, cold_start_flow.mac.

```
######################################################################
#
#       Place ios
#

IOPLACE         STYLE EVEN AUTOMATIC TOPBOTTOMLAYER M4 RIGHTLEFTLAYER
                                          M3 ;
SET VARIABLE    DRAW.PIN.AT     HERE;
REFRESH ;
IOPLACE         FILENAME ioplace.ioc WRITE ;
SAVE DESIGN     ios ;
FLOAD DESIGN    ios ;
```

3.2.3 Source Files

Source files comprise all the design data available at the entry point into the flow. They consist of logical data, and constraints.

3.2.3.1 Logical data

The logical data usually consists of verilog rtl but it can also include gate level verilog as well as VHDL.

The original logical design data is presented in register transfer language (RTL) format and can comprise a mixture of high-level statements, gate-level netlists, and macro instantiations.

3.2.3.2 Constraints

The constraints are spedified as tcl commands and are assembled together into a constraints tcl script. First the verilog rtl is read into pks and then the generic database is built. Once the generic database is built the constraints tcl script is executed and the design is then ready for mapping, optimization, etc. Some example constraint tcl commands are shown below:

```
set_clock clk -waveform {0 12.5} -period 25
set_clock_root -clock clk -pos {clk}
set_input_delay -clock clk -late 2 {in1}
set_input_delay -clock clk -early 1.5 {in1}
set_clock_uncertainty -clock_to {clk} -edge_to leading 0.8
set_clock_uncertainty -clock_to {clk} -edge_to trailing 0.8
set_clock_insertion_delay -pvt max 3.3 {clk}
set_port_capacitance 20 out1
```

3.2.4 Run Data Files

The run data files constist of all the data, reports, logs, and other files which are created by the tools used in this flow. Together these files document each step in the design process and include the intermediate and final data used to generate the device or block.

- adbAmbit database binary files
- def Physical design data
- rpt Report files
- pks PKS run files (cmd, log, etc.)
- gds GDSII files
- verilog Verilog files
- se data files
- wdb Wroute database binary files
- spf Standard Parasitic Files

4.0 Cold-Start Flow

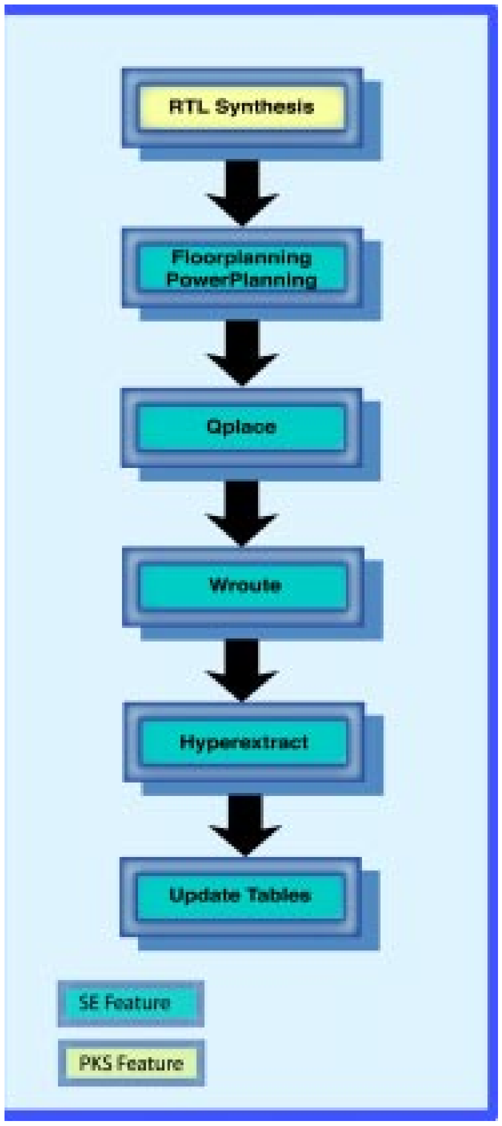


Figure 1 Cold Start Flow

The Cold-Start flow is a fast pass from RTL through layout. The purpose of this is to generate more accurate LEF capacitance values and a realistic layer utilization table. These values are directly related to the aspect ratio and the congestion of the design. To obtain numbers that are as accurate as possible it is necessary to generate a complete floorplan including macro/block placement, power mesh and any filler cells required.

Some steps are not as exacting as necessary for the normal flow For instance, the design does not have to be routed to perfection and the design is only placed for congestion. There are, however, basic steps that are necessary to create the information needed for the layer utilization tables and the update the LEF capacitance tables. Figure 1 illustrates the basic steps that need to be accomplished for the Cold-Start flow.

It is important to note that this procedure may not be necessary for every design that uses the SP&R flow. Designs that are similar in size and complexity would not necessitate going through this process. If, however, it becomes difficult to achieve timing closure this process should be utilized.

The intent of this exercise is quickly place and route the design while utilizing the floorplanning that will be necessary for the design. For the Cold-Start flow it will be assumed that all the tools, Qplace, Wroute and Hyperextract will be run from the SE environment to minimize data transfer.

Scripts were created to transport this design through the cold start flow. Separate scripts drive the PKS tasks and the SE related tasks. The complete scripts appear in Appendix. These scripts consist of a group of lower level command files that contain actual directives for PKS or SE respectively. Each of the following sections will describe the actual commands used in the lower level procedure. These commands could actually be entered manually to acheive the same results as running the scripts in the Appendix. Note that to use the commands as shown in the following the proper setup as described in Section 3 must be performed.

4.1 RTL Synthesis

PKS synthesis is used to construct a gate level netlist from an RTL level design. PKS can generate an initial area estimation values and perform Timing Driven Block Placement. Although PKS can also be used to create an initial floorplan, the physical data will be imported in the form of a DEF file. Silicon Ensemble will be used to generate the floorplan and subsequent DEF file. During this step in the SPnR flow, the HDL source file is read in and a generic netlist is created. This generic netlist is optimized and mapped to generic logic resources. The generic netlist is then mapped to the target library to be used for the design. This synthesis is done with low effort in order to expedite the design through this process.

4.1.1 The main actions in the RTL synthesis step

- Input the RTL design.
- Generate generic netlist
- Perform generic optimization
- Map design to target library
- Perform pre-placement optimizations
- Output a mapped gate-level Verilog netlist

4.1.2 RTL Synthesis Inputs and Outputs

Inputs:
- Timing library data files (OLA).
- Physical library data files (LEF).
- Layer utilization table, if final flow.
- The original RTL source file.
- The timing constraints TCL file.
- dont_utilize list for synthesis

Outputs:
- An Ambit database.
- The hierarchical verilog gate/macro-level netlist for the entire design.

4.1.3 RTL Synthesis Procedure

A script was created to automate the creation of a mapped netlist for use in the Cold-Start flow. The script encapsulates all of the steps outlined below. It sets up the PKS environment, sets the environment variables for OLA and runs the tcl scripts to generate the verilog netlist of the design. The necessary setup tcl scripts were described in section 2 of this document. The complete script is shown in Appendix.

4.1.3.1 Load libraries

The target libraries need to be loaded into PKS prior to any synthesis. The OLA libraries must be setup prior to envoking PKS (see section 3.2.2.1.1). The libraries are usually loaded via a setup script (see section 3.2.2.1.2, read_lef commands).

To start source the OLA setup:

```
prompt> source ola setup

Where ola setup is
#!/bin/csh
 set    oladir =                $LIB_DIR/ndr/SOL
        setenv DCMRULESPATH             ${oladir}:${oladir}/%RULENAME
        setenv DCMRULEPATH              ${oladir}/DCMinterface_SOL
        setenv DCMTABLEPATH             ${oladir}/tables:${oladir}/
                                      %RULENAME
        setenv DCMBOMPATH               ${oladir}
        setenv DCMDoRangeCheck          0
        setenv DCMWireLoadLevels        6mtwb
```

Then start PKS load setup conditions and read libraries:

```
prompt> pks_shell

set_globalecho_commandstrue
set_globalfix_multiport_netstrue
set_globalline_length166

read_ola
set_dcl_calculation_mode -mode worst_case
set_operating_parameter -process 1.00
set_operating_parameter -temperature 100
set_operating_parameter -voltage 2.3
set lib_dir<path_to_lib_data>

read_lef $lib_dir/tech7hp.6mt.lef
read_lef_update $lib_dir/abstract7hp.lef
read_layer_usages $lib_dir/lut

source sige.dont_use_cells.tcl
```

The above files are examples. The actual setup and run files are given in the appendix.

Many other variables and synthesis parameters are specified in the setup file. An important one to notice is the invocation of the sige.dont_use_cells.tcl. This marks cells in the library so they will not be used during synthesis.

4.1.3.2 Input the RTL design into PKS.

The following command sample loads all the verilog files into PKS. There are a number of ways to get the RTL source files into PKS. For example they can be loaded individually or put in a list that a tcl script can read as in the script in the Appendix.

```
read_verilog $SOURCE_DIR/dft_chip.v \
$SOURCE_DIR/adjust.v \
...
```

4.1.3.3 Generate generic netlist

Once all the verilog files have been read into PKS the internal database is built with the do_build_generic command. The design is also synthesized here using generic cells but it is not yet mapped to the target library. Following this synthesis the generic database and verilog are written out. It’s a good idea to check the netlist at this point for things like undriven nets, multiply driven nets and undriven ports. The check_netlist does this and more.

```
do_build_generic
write_adb -all $ADB_GENERIC
write_verilog -hier $VERILOG_GENERIC
check_netlist -verbose
```

4.1.3.4 Read Constraints

The constraints are read in at this time for use by subsequent optimization efforts. The constraints can either be written in Ambit format or translated from Synopsys constraints. When using constraints in Synopsys constraint format Cadence provides a mechanism for translation. The constraints are in the form of a tcl file and is sourced into PKS with the following command.

```
source $CONSTRAINTS_AMBIT
```

It is important to note here that certain cells are constrained from being used during the synthesis and mapping process. This list of cells is kept in a tcl file that is sourced from the setup script. The contents of the **dont_utilize** list is in the Appendix.

4.1.3.5 Optimize generic netlist and map design to target library

The design is mapped to the target library and then optimized with the given constraints.. A timing reportis generated and the design is checked. At this time the clock tree has not been created. Depending on how large the clock network is, it may become necessary to either buffer the clock net or build a clock tree to reduce congestion in the cold-start flow.
Depending on the design, further optimization may need to be performed to more accurately size the design. The simplest method is to let PKS run in a default mode and use the resulting netlist. If this is anticipated to be too time consuming, the utilization can be lowered (10-15%) to compensate for any growth in size of the design due to optimization to meet timing. Then optimization can be run in the non PKS mode.

```
do_optimize -pks -effort low
```

4.1.3.6 Output a mapped gate-level Verilog netlist

The database and verilog of the mapped design is written out. This netlist is the one that is supplied to SE for the remainder of the cold start flow. This verilog netlist will be used by SE to size the design according to the utilization goals.

```
write_adb -all $ADB_MAPPED
write_verilog -hier $VERILOG_MAPPED # (dft_chip.mapped.v)
```

4.2 Floorplanning/ Powerplanning

This is the point in the SPnR flow where the macro blocks and I/O pins get fixed into their final position. The power grid and n-well fillers are also added in this step. Note that you can perform many floorplanning tasks using tools like PKS or DP, but this SPnR flow is based in using SE to perform the detailed floorplanning activities.

4.2.1 The main actions performed during floorplanning

- Loading the design: This entails reading the LEF, and the verilog netlist into SE. Since timing operations won’t be performed there is no need to read in timing libraries or other timing constraints into the database.
- I/O’s are placed. Placement can be predetermined or randomly assigned.
- Fixing block placements: Using the move cell functions, the final location of the macro blocks will be set. One of the main things to watch here is that the power pins of the macros may dictate their orientation.
- Adding cells: Power pads and any other pre-placement filler cells such as N-well ties or EndCaps may be added.
- Power planning: This step entails working with the power planner in SE to create a grid of the main power buses on the design. If macros exist, power rings can be created around the macros as required. There is no power analysis in this version of the SPnR flow. The expectation is that “rules of thumb” or apriori knowledge will be used when creating the power grid.
- Placement. A full design placement can be done in either the SE environment or read in from PKS.
- Outputting a new DEF: When the floorplan is completed a new DEF containing the final placement of the I/O pins, macro blocks, power routes, and any additional pre-placement filler cells will be output.

4.2.2 Floorplanning Inputs and Outputs:

Inputs

- Physical library data files.
- The mapped gate level netlist (verilog) generated in the RTL synthesis step.

Outputs:

- A new DEF floorplan file containing
  - Die Area
  - Row definitions
  - I/O pins placed
  - Pre-placed blocks and macros
  - N-well ties placed
  - Power mesh in place

4.2.3 Floorplanning Procedure

All of the floorplanning is done in SE for this flow. Other floorplan tools may also be used as long as they provide a def placement file as an output. An SE macro script was created that calls a series of additional macro scripts to execute activities that lead to a floorplan. The macro files are described below. The complete macro file for the SE part of the Cold-Start flow is shown in the Appendix.

4.2.3.1 Load Libraries

The libraries are loaded and the database is immediately saved for future use

Silicon Ensemble GUI command : File -> Import -> LEF

```
FINPUT LEF FILENAME "./lef/tech7hp.6mt.lef" ;
INPUT LEF FILENAME "./lef/abstract7hp.ibm.lef" ;
save lib ;
fload lib ;
```

4.2.3.2 Input verilog

The verilog imported here is the output of the previous PKS run in section 4.1.

```
# SE Gui command : File -> Import -> Verilog

#************************************
# READ VERILOG
#************************************

SET VAR INPUT.VERILOG.POWER.NET "VDD!";
SET VAR INPUT.VERILOG.GROUND.NET "GND!";
SET VAR INPUT.VERILOG.LOGIC.1.NET "VDD!";
SET VAR INPUT.VERILOG.LOGIC.0.NET "GND!";
SET VAR INPUT.VERILOG.SPECIAL.NETS "VDD! GND!";
INPUT VERILOG FILE "ibm.stub.v" LIB "lib_vbin" REFLIB "lib_vbin" ;
SAVE  DESIGN ’v_lib’ ;
FLOAD DESIGN ’v_lib’ ;
INPUT VERILOG FILE "./dft_chip_struct.v" LIB "design_vbin" REFLIB
     "lib_vbin" DESIGN "design_vbin.dft_chip:hdl" ;
REPORT SUMMARY FILENAME "./REPORTS/design.summary" ;
SAVE DESIGN ’design’ ;
FLOAD DESIGN ’design’ ;
```

4.2.3.3 Initialize Floorplan

The basic floorplan is established at this point. Row utilization is set and will determine the congestion and ultimately the size of the design. The row spacing is set to 0 and a halo is placed around any blocks in the design. The rows are flipped and abutted. The rows start at the coordinate 6000, 6000. The row utilization is also set at this point. It is set to 80% but can be increased or decreased depending on design considerations such as size and congestion.

SE Gui command : Floorplan -> Initialize Floorplan

```
#************************************
# Initialize FLOORPLAN

FLOAD DESIGN    design;
SET VARIABLE    USERLEVEL           EXPERT;
SET VARIABLE    PLAN.REPORT.STAT     "   ";
SET VARIABLE    UPDATECOREROW.BLOCKHALO.GLOBAL  2000;
FINITIALIZE FLOORPLAN
                ASPECTRATIO             1
                ROWUTILIZATION          0.80
                XIOTOCOREDIST           6000
                YIOTOCOREDIST           6000
                ABUTPAIRSOFROWS
                FLIPALTERNATEROWS
                ROWSPACING              0
                BLOCKHALO               2000 ;
REPORT SUMMARY  FILENAME "rpt/init_fp.summary.rpt" ;
SAVE DESIGN     init_fp ;
```

4.2.3.4 IO Placement

There are several methods of placing the IO’s in a design. In this case a random placement was selected . See Floorplaning strategies in section 4.2.3.5 for alternate IO placement options.

Random IO pin placement : Place -> IOs

```
#************************************
# PLACE IOS
#************************************
#

IOPLACE AUTOMATIC STYLE EVEN TOPBOTTOMLAYER M4 RIGHTLEFTLAYER M3 ;
SET VARIABLE DRAW.PIN.AT HERE;
REFRESH ;
IOPLACE FileName ./ioplace.ioc WRITE;
SAVE DESIGN "ios" ;
```

4.2.3.5 IO Placement Strategies

Five possible IO pin placement techniques are presented here. Usage is entirely dependent on design requirements. In the sample testcase used for this flow the automatic IO placement was used. The following gives a brief introduction to each of the techniques.

4.2.3.5.1 Float all pins:

This can be accomplished by initializing the floorplan in SE.

```
Floorplan -> Initialize floorplan in SE.
```

All the pins will have a status of floating (No placement STATUS).
The IO placement file can be written out:

```
Place -> IOs -> I/O Constraint file ->Write
```

Notice that all the I/O pins will have a status of IGNORE.

A random placement is also easily generated.

```
Place -> IOs -> Random
```

This in turn, could be used as a starting point for I/O optimization.

4.2.3.5.2 Group pins/side:

This can be done by specifying the order of the I/O pins using the ioplace.ioc file. This file can be generated via the command:

```
Place -> IOs -> I/O Constraint file -> Write
```

Sample of the file:

```
###################################################################
# In each of TOP()/BOTTOM()/LEFT()/RIGHT() section, there are #
# placed IOs. In the IGNORE() section, the IOs are ignored        #
# by the IOPlacer. In every section, the IO syntax could be: #
#         for pin:        (IOPIN iopinName ); #
#         for pad:        iopadName orientation ; #
#         for space:      SPACE  value; #
# The capital words are keywords. orientation is notrequired.     #
# The value is the space between the IO above and the IO below it.#
###################################################################
IOPLACEHEADER (
(VERSION 5.3 )
(DIVIDERCHAR "/" )
(BUSBITCHARS "[]" )
)
TOP ( # IOs are ordered from left to right)
BOTTOM ( # IOs are ordered from left to right)
LEFT ( # IOs are ordered from bottom to top)
RIGHT ( # IOs are ordered from bottom to top)
```

To read the above file back into SE after modifying, use the following :

```
Place -> IOs -> I/O Constraint file and
              specify the I/O constraint file.
```

4.2.3.5.3 Fix All Pins:

In order to fix the pins a def file must first be generated that contains all the pins. Output def from SE via:

```
File -> Export -> DEF
```

w/ following options turned on: All and External Pins

Change the PINS status from PLACED to FIXED in the output DEF using your favorite editor.

Then read the def back into SE via:

```
File -> Import -> DEF and sepcify the filename
```

4.2.3.5.4 Fix some Pins/Float the rest:

First run random placement via:

```
Place -> IOs -> Random
```

Use actions described in section 4.2.4.2 to specify sides for the pins that need to be FIXED

To specify specific coordinates for some pins, invoke:

```
Edit -> Modify -> Pin
```

*Specify Pin Name
*Specify new coordinates under "To New Point"
*Click OK

Output DEF via:

```
File -> Export -> DEF
```

Make sure the following options turned on: All and External Pins

Change the PINS status from PLACED to FIXED in the outputted DEF on the specific pins that need to be FIXED
Read the def back into SE via:

```
File -> Import -> DEF and sepcify the filename.
```

Note that when in PKS, do_place will have to be run w/ the option [-pin {concurrent|refine}] to either run concurrent IO’s and std cell placement or in the refine mode to optimize the IO’s after std cell placement. To keep the pins on the same side, you will have to also add the option [-pin_same_side true].

For example:

```
do_place -timing_driven -pin concurrent -pin_same_side true
```

4.2.3.5.5 Automatic Pin placement

Automatically places ios with specific layer on specific side.

```
Place ios -> automatic   : specify options for side and layer
```

4.2.3.6 Power Structure

The power stripes are added in all of the required layers. Power rings are placed around the block. Note that the step size determines the spacing between the stripe placement.

The stripes can be manually added by using the following commands in the SE GUI.
Route -> *Plan Power* -> *Add Stripes*

Once the power stripes are added the row power can be inserted and connections can be made to the surrounding power ring. The following script performs this action.

```
#************************************
#       Generate Power
#
FLOAD DESIGN    ios ;
SET VARIABLE    SROUTE.VIA.MERGEONSAMELAYERS    TRUE ;
SET VARIABLE    SROUTE.STACKVIASATCROSSOVER     TRUE ;
SET VARIABLE    DRAW.SWIRE.AT                   ON ;
SET VARIABLE    DRAW.CHANNEL.AT                 ON ;
BUILD CHANNEL ;
CONSTRUCT RING  NET "GND!" NET "VDD!"
```

```
                LAYER M1 CORERINGWIDTH 1500 SPACING CENTER BLOCKRING-
                                      WIDTH 0
                LAYER M2 CORERINGWIDTH 1500 SPACING CENTER BLOCKRING-
                                      WIDTH 0 ;
ADD STRIPE NET "GND!" DIRECTION Horizontal LAYER MT WIDTH 120 LOWERMAR-
                                      GIN
1320 STEP 1920 ALL ;
ADD STRIPE NET "VDD!" DIRECTION Horizontal LAYER MT WIDTH 120 LOWERMAR-
                                      GIN
             2280 STEP 1920 ALL ;
ADD STRIPE NET "GND!" DIRECTION Vertical   LAYER M4 WIDTH 120 LOWERMAR-
                                      GIN
             1320 STEP 1920 ALL ;
ADD STRIPE NET "VDD!" DIRECTION Vertical   LAYER M4 WIDTH 120 LOWERMAR-
                                      GIN
             2280 STEP 1920 ALL ;
ADD STRIPE NET "GND!" DIRECTION Horizontal LAYER M3 WIDTH 120 LOWERMAR-
                                      GIN
             1320 STEP 1920 ALL ;
ADD STRIPE NET "VDD!" DIRECTION Horizontal LAYER M3 WIDTH 120 LOWERMAR-
                                      GIN
             2280 STEP 1920 ALL ;
ADD STRIPE NET "GND!" DIRECTION Vertical   LAYER M2 WIDTH 120 LOWERMAR-
                                      GIN
             1320 STEP 1920 ALL ;
ADD STRIPE NET "VDD!" DIRECTION Vertical   LAYER M2 WIDTH 120 LOWERMAR-
                                      GIN
             2280 STEP 1920 ALL ;
DISPOSE CHANNEL ;
SET VARIABLE    DRAW.SWIRE.GEOM "On";
SET VARIABLE    DRAW.CHANNEL.AT OFF ;
REFRESH ;
SET VARIABLE    OUTPUT.DEF.SPNET.WILDCARD       TRUE ;
SET VARIABLE    SROUTE.LPR.STACKVIASATCROSSOVER TRUE;
SET VARIABLE    SROUTE.LPR.VIASATCROSSOVER      TRUE ;
SET VARIABLE    SROUTE.FOLLOWPINS.FILL          TRUE ;
SET VARIABLE    SROUTE.VIA.MERGEONSAMELAYERS    TRUE ;
SET VARIABLE    SROUTE.STACKVIASATCROSSOVER     TRUE ;
SET VARIABLE    SROUTE.FOLLOWPINS.MAXROWS       500 ;

CONNECT RING    NET "GND!" NET "VDD!" STRIPE FOLLOWPIN ;
OUTPUT DEF      FILENAME def/pwr.def EXTPIN SPECIALNETS VIAS ;
SAVE DESIGN     pwr ;
```

The def written out at this point does not contain any of the cells in the design. It is this def that will be modified by adding the nwell ties.

4.2.3.7 Insert N-Well Cells

The n-well ties are inserted to a def file via a *Perl* program. The def file is written out of SE in the previous step. The Perl progam is invoked from the SE environment (system command) and adds the ties as specified in the calling parameters. By invoking the Perl commandwith the -h option the usage is described. This program can also be used to add any additional filler cells in a pattern.

SE GUI Command : Place -> Filler Cells -> Add cells

```
#*************************************
# ADD N-WELL TIES
#*************************************
#
system ’reflib/cds_scripts/add_fillercells.pl -def def/pwr.def -
       cell_lef reflib/lef/abstract7hp.cds.lef -tech_lef reflib/lef/
       tech7hp.6mt.lef -out def/fp.def -cell NWSX -prefix fil -inc
       72.00 -FS_offset 36.00 -N_offset 36.00 -vss GND! -vdd VDD! ’ ;
FLOAD DESIGN    design ;
FINPUT DEF      FILENAME        def/fp.def
                REPORTFILE      rpt/fp.def_in.rpt
                NOALLOWEEQSUBSTITUTION ;
REPORT SUMMARY  FILENAME        rpt/fp.summary.rpt ;
SAVE DESIGN     fp ;
```

The def description of the floorplan is output to a file, fp.def. This floorplan will be used by PKS directly. The placement and congestion analysis in PKS considers the obstructions placed in the def during SE floorplanning.

4.3 Qplace

Qplace is the placement tool embedded in Silicon Ensemble (SE). It can be run stand-alone or within the SE environment. The placer places I/O pins, blocks and core components of a design based on information contained in the LEF and DEF files.

For the first pass of a new design a full placement and routing needs to be performed. If there are any blocks in the design that were not placed in the previous Floorplanning step, they will be placed by the Block Placer. Once this is done all the standard cells will be placed. This placement will be congestion based.

This effort is an extension of the Floorplanning and Power planning task in the main SP&R flow. The placement performed during this phase are not utilized outside of the Cold-Start flow.

4.3.1 The main actions performed during Qplace

- Loading the design: This entails reading the LEF and the design verilog netlist into SE. Since timing operations won’t be performed there is no need to read in timing libraries or other timing constraints into the database.
- Fixing block placements: Using the automatic block placer, the final location of the macro blocks will be set.
- I/O’s are placed. Placement can be predetermined or randomly assigned.
- Placement. A full design placement can be done in either the SE environment or read in from PKS.
- Outputting a DEF: When the floorplan is completed a new DEF containing the placement of the macro blocks, standard cells, and any additional pre-placement filler cells will be output.

4.3.2 Qplace Inputs and Outputs:

Inputs

- Physical library data files (LEF).
- DEF file from floorplanning step.
- The mapped gate level netlist (verilog) generated in the RTL synthesis step.

Outputs:

- The new flat physical data containing pre-placed blocks, and standard cells in addition to the power mesh and n-well filler cells.

4.3.3 Placing the Cells

The placement of the standard cells performed here is only a congestion driven placement. The goal is to create a routable design. This placement is not the same as that to be used during PKS operations. No attempt is made to place cells for timing reasons.Note here that very few restrictions are applied to this placement. Cells are allowed to be placed under the power stripes.

SE GUI Command : Place -> Cells

```
#************************************
# PLACE CELLS
#************************************

FLOAD DESIGN    fp ;
SET VARIABLE    QPLACE.PLACE.PIN "CONCURRENT" ;
SET VARIABLE    QPLACE.FREE.TRACK.PCT.1 20 ;
SET VARIABLE    QPLACE.FREE.TRACK.PCT.2 88 ;
SET VARIABLE    QPLACE.FREE.TRACK.PCT.3 88 ;
SET VARIABLE    QPLACE.FREE.TRACK.PCT.4 88 ;
SET VARIABLE    QPLACE.FREE.TRACK.PCT.5 88 ;
SET VARIABLE    QPLACE.FREE.TRACK.PCT.6 0 ;
QPLACE          NOCONFIG ;
REPORT SUMMARY  FILENAME rpt/qp.summary.rpt ;
SAVE DESIGN     qp ;
```

The variables in the script QPLACE.FREE.TRACK.PCT sets the estimated percentage of free routing tracks provided for the given layer. Free routing tracks are defined as tracks that are not occupied by power strips or blocked by obstructions in the cells.

4.4 WRoute

Wroute is the routing tool which can be run stand-alone or within the SE environment. It is used to perform global routing, track assignment, final or detail routing, and search-and-repair routing. Wroute is a highly capable router and is normally used in a timing driven manner. This routing is a quick congestion based routing. The only requirement here is that the routing completes enough to get good parasitic information. If there are a few routes that cannot be easily completed, they can be ignored for purposes of parasitic extraction.

4.4.1 The main actions in the routing step

- Input the Verilog netlist and DEF files created by the Qplace step (this will be contained in the database created during the Qplace step).
- Perform global routing.
- Output a new gate-level Verilog netlist along with a new DEF
- Create a Wroute database for hand-off to detail router.

4.4.2 Routing Inputs and Outputs

Inputs:
- LEF physical library data.
- A DEF from the previous floorplaning step
- The gatelevel verilog netlist used to floorplan
- Configuration file instructing Wroute what to do.

Outputs:
- A DEF containing global route info.
- A binary Wroute database file (.wdb).
- Log file (containing utilization information)

4.4.3 Global and Detail Routing

The router is invoked with a configuration file. The configuration file is the method of providing parameters to the router. Input and output file names, lef abstacts and other parameters are specified.

It is in this configuration file that the type of routing to be performed can be specified, global or final routing. In the cold start flow both global and final routing are done at this time.

An important parameter to note is the “routeTopLayerLimit 5”. This, as implied limits the number of routing layers to 5. The wr_cold.cfg file has been tailored for this process. Great care should be taken when modifying this file, it can drastically change the results of routing.

The configuration file used in this flow is shown below.

```
inputDBName             ""
inputLefName            "reflib/lef/tech7hp.6mt.lef reflib/lef/ab
stract7hp.cds.lef"
inputDefName            def/placed_cold.def
outputDbName            drouted_cold.wdb
inputLdefCommentChar    "#"
routeTopLayerLimit      5
routeSelectNet          "@clock"
routeSelectNetFirst     true
routeFinal              true
routeGlobal             true
timingDrivenRouting     FALSE
frouteOnGridViaOnly     FALSE
frouteManufacturingGrid "2x2"
frouteAllowPortShort    FALSE
froutePinAccessMode     "normal"
froutePreferOnGrid      TRUE
frouteSearchRepair      TRUE
outputFullDef           TRUE
outputDefName           def/drouted_cold.def
# Option below is used for Poly pin connection only.
routePolyPinEnclosure   "TRUE"
```

The following executes the global and detail routing.

SE GUI Command : Route -> Wroute

```
#*************************************
# ROUTE THE DESIGN
#*************************************
#
SYSTEM          ’cp ../../../scripts/se/wr_cold.cfg .’ ;
SYSTEM          ’rm -f drouted_cold.log*’
SYSTEM          ’wroute -l drouted_cold.log -q wr_cold.cfg’ ;
FLOAD DESIGN    lib ;
INPUT DEF       FILENAME        def/drouted_cold.def
                REPORTFILE      rpt/drouted.def_in.rpt
                NOALLOWEEQSUBSTITUTION ;
SAVE DESIGN     drouted_cold ;
```

4.5 HyperExtract

HyperExtract extracts resistance and capacitance (RC) parasitics for all nets of the a design that has been placed and routed. HyperExtract extracts parasitics for overlapping area, interlayer fringe capacitance, and coupling capacitance.

HyperExtract uses capacitance extraction rules, also called Capacitance Coefficients, to determine fringe, coupling and other values. Coefficient Generator is used to generate the Capacitance Coefficient values and are generated only once per technology. The Coefficient Generator uses foundry process data to generate the coefficients.

4.5.1 The main actions in the extraction step

- Generate Capacitance Coefficients
- Create an RC RSPF file

4.5.2 HyperExtract Inputs and Outputs

Inputs
- Capacitance Coefficients
- Library data (LEF), Design data (DEF)
- Running from the SE environment the capacitance extraction rules file is the only input necessary.

Outputs:
- A design specific DSPF file.
- hyperextract.log

4.5.3 Coeffgen and HyperRules

The HyperExtract rules are generated as an output of the Coeffgen tool. The data for coeffgen comes from the process foundry documents. For the purposes of this flow it is assumed that this data is supplied by IBM. It is not expected that the user will generate this data.

4.5.4 Parasitic Extraction

HyperExtract is run on the currently loaded database. This is an advantage to running in the SE environment. All of the design files and rules do not have to be entered they are picked up from the database. HyperExtract requires some variables to be set prior invocation. These variables have been set to work with the process used for this flow and may change based on other process requirements.

This exact procedure will be used in the general flow for the actual design. The DSPF output from the cold-start will not be used at this time. For the cold start flow we are interested in the log file produced from running HyperExtract.

SE GUI Command : Report -> RC -> HyperExtract

```
###########################################################################
#
#       Extract cold start the parasitics
#

FLOAD DESIGN    drouted_cold ;
SET VARIABLE    HYPEREXTRACT.PINS.CAPACITANCE           "NONE" ;
SET VARIABLE    HYPEREXTRACT.TOPPINS.CAPACITANCE        "NONE" ;
SET VARIABLE    HYPEREXTRACT.WRITE.DSPF.INSTANCE        "TRUE";
SET VARIABLE    TIMING.REPORTRC.FORMAT                  ’DSPF’ ;
SET VARIABLE    HYPEREXTRACT.SHRINK.FACTOR              ’1.00’;
SET VARIABLE    HYPEREXTRACT.WIRELOAD.MEDIAN            0.0;
SET VARIABLE    HYPEREXTRACT.WIRELOAD.MIN.FANOUT        0;
#SET VARIABLE   HYPEREXTRACT.OTC.GDSII.FILE ’reflib/gds2/cmos7hpv12.gds’;
#SET VARIABLE   HYPEREXTRACT.OTC.GDSII.MAP.FILE ’reflib/hypext/gds_he.map’;
SET VARIABLE    TIMING.CAPACITANCE.MODEL                ’HYPEREXTRACT’;
SET VARIABLE    HYPEREXTRACT.RULES.FILE ’reflib/hypext/HYPER.rules’;
SET VARIABLE    HYPEREXTRACT.WIRELOAD.FILE              ’’;
SET VARIABLE    HYPEREXTRACT.SYNOPSYSLOAD.FILE          ’’;
REPORT RC       FILENAME drouted_cold.dspf ;
```

4.6 Update Tables

The extracted parasitics from the previous step are used to create a new capacitance LEF file and create an updated layer utilization table based on the actual design. The capacitance values that are output in the HyperExtract log files are used to create a new technology LEF file.This new technology lef will then be used in the general design flow to generate more accurate timing. Hyperextract estimates the capacitive effects of each layer in the design based on the routing. These estimates are, in general, much more accurate than the default values supplied in the library defaults.

Similarly figures for layer utilization can be extracted from the WRoute log file. Without a layer utilization file the default would be to allow a uniform distribution of routing on all layers. This could have severe impact on the timing due to differing layer capacitance values.

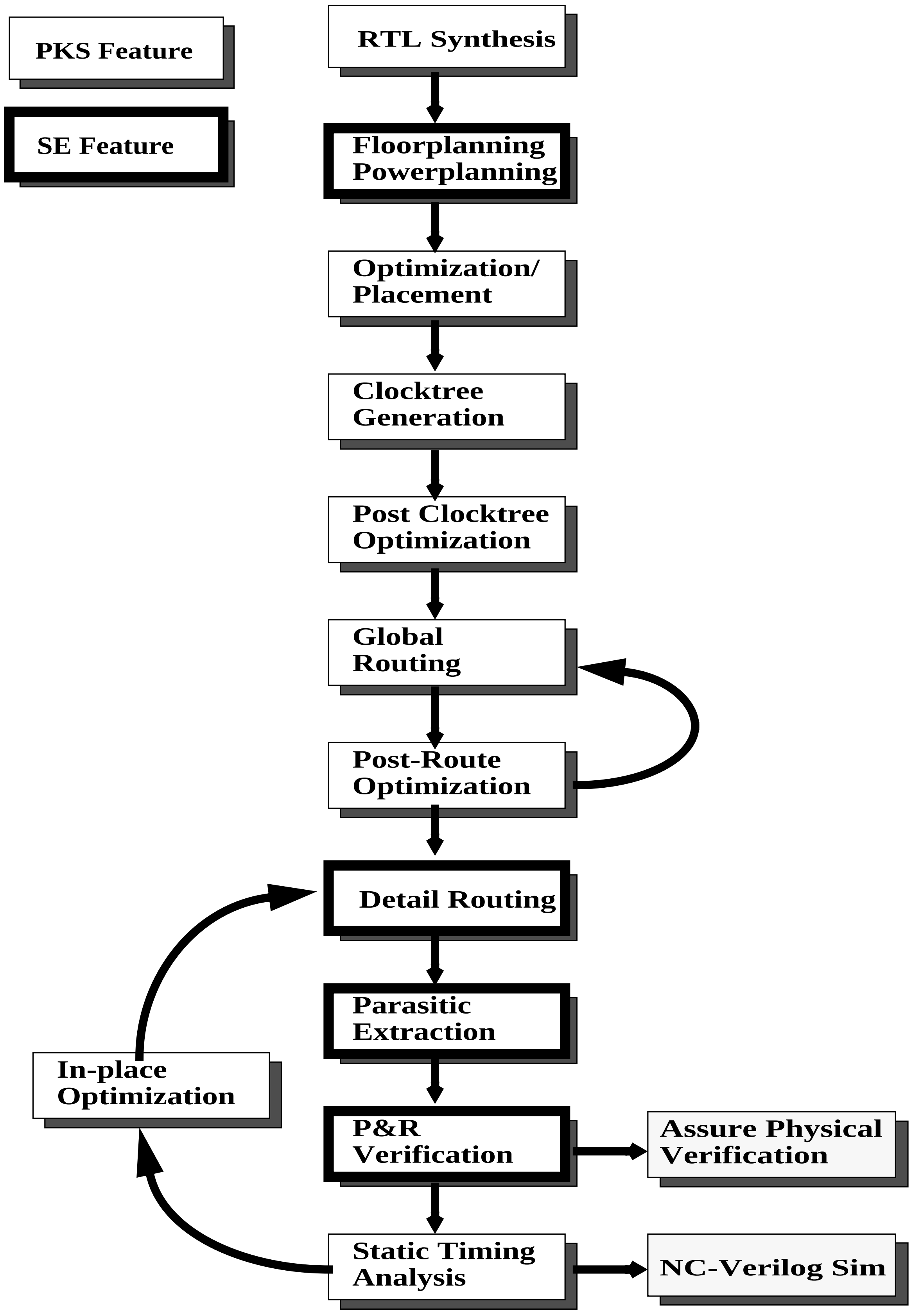
4.6.1 Table Generation Scripts

Scripts have been created to automatically generate the LUT and a new design specific technology file with updated layer capacitances.

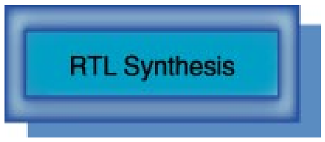
```
########################################################################
#
#       Generate Updated Lut and LEF Capacitances
#

SYSTEM ’reflib/support_files/lutgen.pl ./drouted_cold.log updated.lut’
                                        ;
SYSTEM ’reflib/support_files/lef_update.pl HyperExtract.log reflib/
       lef/tech7hp.6mt.lef updated.lef’ ;
```

5.0 SP&R Flow

The following discussion details each of the major steps in the proposed SP&R flow excluding the Cold Start flow. This flow, shown in Figure 2, is designed to show a succession of major steps needed to complete a design. Much like the Cold-Start flow scripts were written to automate this entire process. The scripts are documented in the appendix. The essential commands that are executed in the scripts are presented here. Manual execution of the commands should lead to the same outcome as the scripts.


PKS Feature
SE Feature
RTL Synthesis
Floorplanning Powerplanning
Optimization/ Placement
Clocktree Generation
Post Clocktree Optimization
Global Routing
Post-Route Optimization
Detail Routing
Parasitic Extraction
P&R Verification
Assure Physical Verification
Static Timing Analysis
NC-Verilog Sim
In-place Optimization

5.1 RTL Synthesis



PKS synthesis is used to construct a gate level netlist from an RTL level design. PKS can generate an initial area estimation values and perform Timing Driven Block Placement. This flow assumes the cold-start procedure was followed, therefore this task is reduced to reading the generic.adb. It is unnecessary to go back to the original RTL unless some change has been made to the original RTL code. If extensive changes have occurred it may be necessary to re-floorplan the design, since this may change the utilization.

During this step in the SPnR flow, the generic netlist created during the cold-start flow is the starting point. This generic netlist is optimized and mapped to generic logic resources. The generic netlist is then mapped to the target library to be used for the design.

5.1.1 The main actions in the RTL synthesis step

- Read in previously generated adb.
- Read constraints
- Read updated technology LEF
- Read Layer Utilization table

5.1.2 RTL Synthesis Inputs and Outputs

Inputs:
- Timing library data files (OLA).
- Physical library data files (LEF).
- Updated Technology LEF.
- Layer utilization table
- The original RTL source file.
- The timing constraints TCL file.

Outputs:
- An Ambit database (ADB).
- The hierarchical verilog gate/macro-level netlist for the entire design.
- The flat physical data (def) containing initial floorplanning and power design information.
- Timing report file.

5.1.3 Design Synthesis

This step is identical to that in the Cold-Start flow. It is re-iterated here for convenience only.

It is not required or necessary to execute this step again if the Cold-Start flow has been implemented.

5.1.3.1 Input the RTL design

The following command loads all the verilog files into pks.

```
read_verilog $SOURCE_DIR/dft_chip.v ...
```

5.1.3.2 Generate generic netlist

Once all the verilog files have been read into pks the internal database is built with the do_build_generic command. The design is also synthesized here using generic cells but it is not yet mapped to the target library. Following this synthesis the generic database and verilog are written out.

```
do_build_generic
write_adb -all $ADB_GENERIC
write_verilog -hier $VERILOG_GENERIC
```

5.2 Floorplanning/ Powerplanning

This Floorplanning/Powerplanning step utilizes the results of activities performed in the cold-start flow. The DEF file created in the cold-start flow is used directly. This establishes the basic floorplan including IO pins, power grid and nwell ties.

5.2.1 The main actions performed during floorplanning

- Loading the design: This entails reading the LEF, DEF's and optionally the verilog netlist into SE. Since timing operations won't be performed there is no need to read in timing libraries or other timing constraints into the database.
- I/O's are placed. Placement can be predetermined or randomly assigned.
- Fixing block placements: Using the QPlace BlockPlace and move cell functions, the final location of the macro blocks will be set. One of the main things to watch here is that the power pins of the macros may dictate their orientation.
- Adding cells: Power pads and any other pre-placement filler cells, such as N-well ties or EndCaps may be added.
- Power planning: This step entails working with the power planner in SE to create a grid of the main power buses on the design. If macros exist, power rings can be created around the macros as required. There is no power analysis in this version of the SPnR flow. The expectation is that "rules of thumb" or apriori knowledge will be used when creating the power grid.
- Placement. A full design placement can be done in either the SE environment or read in from PKS.
- Outputting a new DEF: When the floorplan is completed a new DEF containing the final placement of the I/O pins, macro blocks, power routes, and any additional pre-placement filler cells will be output.

5.2.2 Floorplanning Inputs and Outputs

Inputs

- DEF file from SE floorplanning .

Outputs:

- A new .adb containing
  - Die Area
  - Row definitions
  - I/O pins placed
  - Pre-placed blocks and macros
  - N-well ties placed
  - Power mesh in place

5.2.3 Import Floorplan

The floorplan for this design was determined in the Cold-Start flow. It is imported here by reading in the def file output by SE. This floorplan will contain all of the power rails, n-well ties and rows.

The following command reads in the def file:

```
read_def $def
```

The following information will be read or calculated from the DEF. Some of this information may be overwritten with PKS commands (using set_floorplan_parameters command).

- DieSize: This is the bounding box.
- Rows: These rows are used for the bins in PKS, and will be passed to Qplace, and written back into the DEF at the end of PKS.
- Tracks: The tracks are used to calculate congestion. They are also used by Qplace and the global router called by do_route.
- IO to Core distance (This is from the die edge to core rows)
- Row orientation: Horizontal or Vertical
- Row Flipping
- Row Spacing
- Initial utilization is calculated.
- Any instance placed in the DEF that can be found in the netlist will get placed.
- Any placed instance with + FIXED will get the dont_move attribute set on it.
- Cover Macros found in DEF will be placed.
- Filler Cells from DEF can be included at your request.

Note that PKS has an extensive floorplanning capability and is constantly being advanced. Most of the features utilized during SE floorplanning could be done in PKS. However, since a floorplan is required for the Cold-Start flow it would be redundant to recreate one here.

5.3 Optimization/Placement

Once the power plan is completed and the macro block placement finalized in the floorplanning step, the netlist is now ready to be placed and optimized. An initial timing analysis is performed to determine if solution at this stage is acceptable. The timing calculation is based on the Steiner Minimum Spanning Tree (MST) at this point.

The updated Layer Utilization file and the technology lef containing the updated capacitance values from the Cold-Start flow are now included in the setup tcl script

Some of the optimizations available at this point include: Buffer insertion, repeater insertion
resizing, restructuring and cloning

5.3.1 The main actions in the placement, and optimization step

- Read Verilog RTL level netlist or gatelevel netlist generated in RTL synthesis step.
- Input the DEF file with the updated floorplan and power plan The information that will be read from this DEF is: Rows, Tracks, Pre-routes in the SPECIALNETS section, block and filler Placements.
- Set the appropriate PKS controls.
- Perform Timing Driven Placement.
- Perform optimization.
- Output a new gate-level Verilog netlist along with a new DEF for use with the clock tree generation step.

5.3.2 Optimization Placement Inputs and Outputs

Inputs:

-
- The following should be loaded in the database from the previous Floorplanning step.
  - Timing library data files.
  - Updated Layer Utilization Tables and technology files.
  - Physical library data files.
  - Layer utilization table.

Outputs:

- An Ambit database (adb) containing design, floorplan, and constraint information.

- The new hierarchical gate/macro-level netlist for the entire design.
- The new flat physical data containing floorplanning, placement, and power design information.
- Timing report file.

5.3.3 Design Placement and/or Optimization

With the floorplan complete the database created during the previous step is is mapped to the target library, placed and then optimized with respect to the constraints. In general if clock uncertainty is set up in the constraints then the timing estimates should be reasonable. After placement slack could again be optimized based on the ideal clock.

5.3.3.1 Perform timing driven placement

The design is placed using timing information and a timing report is generated.

```
do_optimize -pks
report_timing > $REPORT_TIMING_PLACED
```

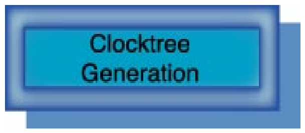
There are many xforms and possible sequences available. Their use is entirely design dependent. The do_optimize -pks is a default mode that performs a set of optimizations in a specific order. Refer to the Envisia and Ambit Synthesis Command Reference Manual for a complete detailed description of the commands.

5.3.3.2 Output a placed adb

The resultant .adb should be saved at this time. It will be the starting point for subsequent operations such as clock insertion.

```
write_adb -all $ADB_PLACED
write_verilog -hier $VERILOG_PLACED
```

5.4 Clocktree Generation



In this step CTPKS is used to generate clock trees that satisfy the clock constraints specified in the constraint file. The CTPKS tool can be invoked from within PKS. At this point the RC estimations used in timing analysis is still based on the Steiner Minimum Spanning Tree.

5.4.1 The main action in the Clock Tree Generation step

- Invoke CTPKS with associated command and constraint files

5.4.2 CTPKS Inputs and Outputs:

Inputs:
These input files should already be loaded into the PKS database but are repeated here for completeness.

- Timing library data files.
- Physical library data files.
- Layer utilization table.
- CTPKS constraints (a tcl file containing clock definitions)
- PKS database generated by the previous step.

Outputs:
These outputs are automatically written to the PKS database.

- The new hierarchical gate/macro-level netlist (including the clock tree information) for the entire design.
- The new flat physical data containing floorplanning, power design, and clock tree information.
- Many different timing report files can be written out to analyze the clock network to meet performance criteria.

5.4.3 Clocktree Generation

Invoke CTPKS with the associated command and constraint files.

5.4.3.1 Load clock constraints

The clock constraints are kept in a tcl file that is identified in the setup.tcl. In this testcase the constraints are relatively simple and could easily be executed directly. It is however advisable to maintain a constraint file for reference and repeatability. The clock constraints look like the following example:

```
set_clock_tree_constraints -min_delay 2.3 -max_delay 3.3 -max_skew 0.5
                         -pin [find -inputs clk]
```

The constraints are loaded with the following command.

```
source $CONSTRAINTS_CTPKS
```

The clock tree requires the use of special buffers and inverters designed specifically for use in the clock and physical trees. These cells are enabled for use and all other buffers and inverters are disabled during clock tree generation.

```
set buffers [get_names [get_clock_tree_objects -buffer]]
foreach cell_id [find -cell $buffers] {
set_attribute $cell_id ct_dont_utilize true
}
set inverters [get_names [get_clock_tree_objects -inverter]]
foreach cell_id [find -cell $inverters] {
set_attribute $cell_id ct_dont_utilize true
}
set_cell_property dont_utilize false $CTPKS_USABLE_CELLS
foreach cell_id [find -cell $CTPKS_USABLE_CELLS] {
set_attribute $cell_id ct_dont_utilize false
}
set ctpks_buffers [get_names [get_clock_tree_objects -buffer]]
set ctpks_inverters [get_names [get_clock_tree_objects -inverter]]
issue_message "CTPKS usable buffers:   $ctpks_buffers"
issue_message "CTPKS usable inverters: $ctpks_inverters"
```

5.4.3.2 Set clock mode to propagated

Here we set the clock propagation to propagated. This instructs ctpks to use computed delays for the clock propagation instead of the ideal propagation.

```
set_clock_propagation propagated
```

5.4.3.3 Generate Clock Report

Here we generate a clock tree report. At this stage we would have one driver cell driving all the leaf pins.

```
report_clock_tree > $REPORT_CLOCK
```

5.4.3.4 Build the Clock Tree

At this point we actually build the clock tree and then we generate another clock tree report which should show the synthesized tree. We also report any clock tree violations.

```
do_build_clock_tree -pin {clk}
report_clock_tree >> $REPORT_CLOCK
report_clock_tree_violations
```

Next we build all the physical tree, in this case just the reset. This satisfies all design rule requirements such as fanout without any consideration to timing.

```
do_build_physical_tree reset
```

5.4.3.5 Final Clock Tree Report

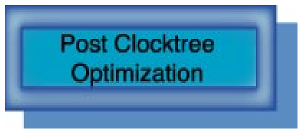
Finally we generate a timing report with the new clock tree and then we write out the updated database, verilog and def files.

```
report_timing > $REPORT_TIMING_CLOCKED
write_adb -all $ADB_CLOCKED
write_verilog -hier $VERILOG_CLOCKED
write_def -placement all -netlist $DEF_CLOCKED
```

After clock and physical tree generation we disable again any cells that were meant to be used only in the clock tree.

```
source $env(SCRIPT_DIR)/tcl/sige.dont_use_cells.tcl
```

5.5 Post Clocktree Optimization



This step entails any post-CTPKS optimizations of the design for setup violations. This is also the first point at which hold time violations will be fixed. During the optimization in the propagated clock mode, all elements (e.g. flip-flops) of the clock network are fixed in both placement (dont_move) and timing (dont_modify).

5.5.1 The main actions in the post clock tree optimizations

- Set the clock mode to propagated.
- Perform the optimizations (with the clock network fixed, the optimizations cannot change any of the clock tree elements, nor any of the sequential elements connected to the clock tree).
- Fix hold times.
- Output a new gate-level Ambit database along with a Verilog netlist and a new DEF for use with the global routing step.

5.5.2 Post Clock Tree Optimizations Inputs and Outputs

The input and output data mentioned below are internalized in the PKS database and do not have to be explicitly loaded

Inputs:

- Timing library data file(s).
- LEF physical library data file(s).
- An LUT Layer utilization table.
- The hierarchical gate/macro-level netlist.
- The original timing constraints TCL file.

Outputs:

- An Ambit database containing design, floorplan, and constraint information.
- The new hierarchical gate/macro-level netlist for the entire design.
- The new flat physical data (post_ctgen.def) containing floorplanning, power design, and clock tree information.

5.5.3 Optimize the Design

The object of this stage is to remove any negative slack in the design. Actual clock delays are realized. The net delays are still Steiner based at this point.

5.5.3.1 Set the clock mode to propagated.

In this step we make sure that pks is using computed delays for the propagation of the clock instead of ideal guesstimated delays with no skew.

```
set_clock_propagation propagated
```

5.5.3.2 Perform Synthesis Transformations

The design is optimized with full knowledge of the clock tree and clock tree delays. At this stage the design is placed but not routed so all the delays are based on steiner routes. Following optimization we report timing.

```
do_xform_ipo
do_place -eco
report_timing > $REPORT_TIMING_OPTIMIZED
```

5.5.3.3 Fix hold times

The fixing of hold time violations can be performed at this point. If the timing library contains both best and worst case conditions, you will simply change modes for the library. The OLA library is capable of giving both best and worst case figures. The best and worst case parameters are contained in the setup.tcl file. The case desired can be loaded at any time and new timing reports can be generated.

The commands used to perform this operation are as follows.

```
set_design_ola_best_case
set_clock_propagation propagated
set_clock_uncertainty  -pin clk 0.0
do_xform_fix_hold -incremental -dont_reclaim_area
do_place -eco
```

This will fix the hold time violations with minimum impact to creating setup violations. It is important to remember to re-place the design after any hold fixes as components may have moved.

5.5.3.4 Output a new gate-level database

After each step we continue to write out the current database, verilog and def. This allows for a quick and easy recovery in the event of an error or power outage.

```
write_adb -all $ADB_OPTIMIZED
write_verilog -hier $VERILOG_OPTIMIZED
write_def -placement all -netlist $DEF_OPTIMIZED
```

5.6 Global Routing

This step entails running the global route portion of Ultra Router Wroute inside the PKS environment. The initial timing for the global route optimization is based on the fast route done inside the PKS tool. For subsequent optimization passes of the global route, the timing is based on RC information derived from the global route estimations.

The global routing step creates a plan for how the final router completes its routing. The area of the chip to be routed is divided into cells called GCells. Each GCell contains a number of vertical and horizontal routing tracks. When the global router is run tracks from GCells are allocated to nets that are determined to pass through them.

During global routing, the router interconnects the regular (signal) nets defined in the NETS section of the DEF file for the design, based on the timing constraints and available routing tracks. It finds generalized pathways, without laying down actual wires, and makes iterative passes to optimize the global routing, shorten wire length, and minimize the use of vias. During global routing, the router also updates the congestion map.

5.6.1 The main actions in the global routing step

- Input the Verilog netlist and DEF files created by the post clock tree optimizations step or a previously saved .adb file.
- Perform global routing (do_route).
- Output a new gate-level Verilog netlist along with a new DEF
- Create a Wroute database for hand-off to detail router.

5.6.2 Global Routing Inputs and Outputs

Inputs:
- Timing library data.
- LEF physical library data.
- LUT Layer utilization table.
- An Ambit database containing design, floorplan, and constraint information.

Outputs:
- An Ambit database containing design, floorplan, and constraint information.
- A binary Wroute database file (.wdb). This database is all that is necessary to run the detail router.

5.6.3 Perform global routing

Wroute runs using the timing engine of PKS for the timing of nets to determine their criticality and order. Timing reports of the worst case slack are generated during the route. These will be printed periodically as the run progresses. The first timing report is based on the steiner estimates before the global route begins, the rest of the timing reports during the route are based on RC information extracted from the global route. As the run progresses the timing may get worse, due to the fact that trade-offs will be made between congestion and timing.

A global variable was set in the setup.tcl file to pass the antenna lef to wroute. A configuration file is created in the setup.tcl. When wroute is invoked the global indicates which lef to use. The script below is from the setup.tcl and shows how the global is set and created.

```
set_global _wr_config $env(PKS_DIR)/wr_lef.cfg
set wr_lef_cfg [open [get_global _wr_config] w]
puts $wr_lef_cfg "inputLefName '$LEF_DESIGN_FILES $LEF_ANTENNA'"
close $wr_lef_cfg
```

At this point check to make sure the clock propagation mode is set to propagated, report on the clocks and timing and generate the global routing. The timing is reported after the global routing.

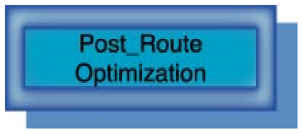
```
set_clock_propagation propagated
```

It is a good idea to route the clocks first. This allows greater freedom to route without obstructions and would more easily accomodate shielding if desired.

```
report_clocks
report_timing
do_route \
          -timing_driven \
          -route_top_layer_limit 5 \
          -route_select_nets_first true \
          -route_select_net “@clock” \
          -output_db_name $WDB \
          -def_out_name   $DEF_GROUTED
report_timing > $REPORT_TIMING_GROUTED
```

After the routing is completed a WDB (WROUTE database) is created along with a new DEF file that corresponds exactly to the database inside WROUTE. Also, the RC information is back annotated into the PKS database.

5.7 Post Route Optimization



Once the route is complete, the global route RC information is automatically back-annotated into PKS as predictors on each net. A predictor is the difference between the steiner estimate of the net RC value and the actual RC value from the global route. Any change that effects the net will calculate the RC by doing the Steiner MSP route estimation of the net, and then reapplying the predictor (delta).

Following post-route optimization, the global router must be re-run. The global router is by nature non incremental. Placement optimization may change routing cell utilization estimation and thereby force rerouting. Timing corrections, however, should have minimal impact on placement and should only be making small adjustments to the design. The most common optimization will be buffer sizing while not reclaiming area.

5.7.1 Main actions in Post Route Optimization

- Selectively execute PKS optimizations
- resizing
- buffer insertion
- cloning
- reclaim area
- Reroute the design
- do_route -timing_driven

5.7.2 Post Route Inputs and Outputs

This optimization is executed on the internal PKS database. No additional input is necessary to perform this operation.
Outputs:
- New Groute database (.wdb)
- New Verilog netlist and DEF (optional)
- Timing Report
- Congestion Map

5.7.3 Re-optimize Design

Once the route is complete, it may be necessary to run post groute optimizations. At this level minimal changes to the netlist should be made to ensure timing closure. Unless the design has undue congestion, there should be a limited number of issues to fix. At this point, it is recommended to do only resizing and/or buffering and to not allow area reclamation

5.7.3.1 Selectively execute PKS optimizations

This is done only if the timing report indicated that timing has not yet been met. The clock propagation is set to propagated and the design is optimized. An incremental placement is then performed to legalize any touched cells. The usual reports and updated files are generated following the optimization.

```
do_xform_ipo
do_place -eco
report_timing > $REPORT_TIMING_IPO
write_adb -all $ADB_IPO
write_verilog -hier $VERILOG_IPO
write_def -placement all -netlist $DEF_IPO
```

5.7.3.2 Reroute the design.

The design is global routed again in the same manner as before. Global routing is always the last step performed prior to leaving the PKS environment.

```
set_clock_propagation propagated
report_clocks
report_timing
do_route \
          -timing_driven \
          -route_select_nets_first true \
          -route_select_net “@clk \
          -output_db_name $WDB_IPO \
          -def_out_name   $DEF_IPO_GROUTED
report_timing > $REPORT_TIMING_IPO_GROUTED
write_adb -all $ADB_IPO_GROUTED
```

5.8 Detail Routing

In this step the WROUTE router, that is part of SE Ultra, routes the placed and globally routed design using information described in the warp-route database. The router, during operation, can create shorts or spacing violations in order to complete 100% of the routing. A search and repair mode is automatically invoked to attempt to fix any violations. Later on, during In Place Optimization, WRoute may be called in an ECO mode.

5.8.1 The main actions in Detail Routing:

- Invoke WROUTE with an associated configuration file. The wroute tool can be called from the SE GUI, or run as a standalone utility. For the purposes of these discussions we shall assume that wroute is being called as a stand alone utility.
- During final routing, the router follows the global routing plan and lays down actual wires to connect the pins to their corresponding nets. It also removes routing violations and minimizes wire length and the number of vias. Final routing automatically runs search-and-repair routing unless you specify otherwise. Final routing stops automatically if it exceeds the time limit you set for routing or if it cannot make further progress on routing the design.
- The search-and-repair routing corrects violations. It delineates areas that are centered on design violations and reroutes each area to eliminate the violations.

5.8.2 Final Routing Inputs and Outputs

Inputs:
- DEF physical library data file.
- A global routed WROUTE database from PKS (.wdb).
- Wroute configuration file.
- If ECO, the modified DEF.

Outputs:
- The new flat physical data containing floorplanning, power design, clock tree, and final routing information.
- A binary warp route database file

5.8.3 Invoke Final Route

The final route is invoked with the following command

```
wroute -l wr_droute.log -q wr_droute.cfg
```

A sample wroute config file :

```
inputDBName              wdb/grouted.wdb
outputDbName             wdb/drouted.wdb
outputDefName            "def/drouted.def"
outputFullDef            true
routeGlobal              false
routeFinal               true
frouteSearchRepair       true
routeSelectNet           @clock
routeSelectNetFirst      true
inputLdefCommentChar     "#"
routeTopLayerLimit       5
timingDrivenRouting      false
frouteOnGridViaOnly      false
frouteManufacturingGrid 2x2
frouteAllowPortShort     false
froutePinAccessMode      normal
froutePreferOnGrid       true
antennaFullReport        true
antennaReportName        rpt/antenna_droute.rpt
frouteFixAntennaPass     2
frouteAntennaCellPass    2
frouteAntennaMethod      LAYERONLY
# Option below is used for Poly pin connection only.
routePolyPinEnclosure    true
#frouteRuntimeLimit      300
```

It is important to note that in the configuration file passed to wroute the global routing option is set to false. You want to use the global routing that was done in PKS. That routing is timing driven and has been optimized to meet constraints.

5.9 Parasitic Extraction

HyperExtract is used to calculate the resistance and capacitance (RC) parasitics of design interconnections for all of the nets (or a subset) at the block or chip level.

HyperExtract extracts parasitics for overlapping area, interlayer fringe, and coupling capacitance, including fringe capacitance in the presence of an adjacent conductor. These parasitics include effects for nets routing over the cell (OTC) and over the block (OTB). These OTC and OTB effects include coupling interactions to conductors within the cell or block.

HyperExtract runs on a connectivity-based layout and outputs extracted RC values in a variety of SPF formats. HyperExtract can generate either a capacitance file for each net or an RC file with a SPICE like description of each net.

5.9.1 The main action in the Parasitic Extraction

- Invoke HyperExtract from within the PKS interface with an associated rules file.

5.9.2 Hyperextract Inputs and Outputs

Inputs:
- A LEF physical library data file.
- The flat physical data containing floorplanning, power design, clock tree, and final routing information.
- Hyperextract.rules file.
- GDS2 file of standard cells and blocks.

Outputs:
- A standard parasitic format (DSPF) file.

5.9.3 Extract Parasitics

HyperExtract is run on the currently loaded database. This is an advantage to running in the SE environment. All of the design files and rules do not have to be entered they are picked up from the database. HyperExtract requires some variables to be set prior invocation. These variables have been set to work with the process used for this flow and may change based on other process requirements.

```
SET VARIABLE HYPEREXTRACT.PINS.CAPACITANCE      NONE ;
SET VARIABLE HYPEREXTRACT.TOPPINS.CAPACITANCE   NONE ;
SET VARIABLE HYPEREXTRACT.WRITE.DSPF.INSTANCE   TRUE ;
SET VARIABLE TIMING.REPORTRC.FORMAT             DSPF ;
SET VARIABLE HYPEREXTRACT.SHRINK.FACTOR         1.00 ;
SET VARIABLE HYPEREXTRACT.WIRELOAD.MEDIAN       0.0 ;
SET VARIABLE HYPEREXTRACT.WIRELOAD.MIN.FANOUT   0 ;
SET VARIABLE TIMING.CAPACITANCE.MODEL           HYPEREXTRACT ;
SET VARIABLE HYPEREXTRACT.RULES.FILE reflib/hypext/hyper7hp_wc.rules ;
SET VARIABLE HYPEREXTRACT.WIRELOAD.FILE         '';
SET VARIABLE HYPEREXTRACT.SYNOPSYSLOAD.FILE     '';
```

The dspf can be generated either from the GUI or command line .

SE GUI Command : Report -> RC -> HyperExtract

```
REPORT RC       FILENAME final.dspf
```

5.10 P&R Verification

After the design routing is complete it will need to be verified. A quick check is first performed to see that all nets were routed, a preliminary DRC check of the physical routing can also be done. Both of these checks are performed inside of SE.

5.10.1 The main actions performed during verification

- Verify Geometry: This will verify that the routing does not violate spacing and width rules. The user should turn off the same cell checks. Turning these off will prevent false violations that are often times created inside of the abstracts used for LEF from being flagged. This is performed inside of SE.
- Verify Connectivity: This will verify that all the nets in the DEF are completely connected. This check will report opens, antennas, loops, partial routing and other connectivity problems. If the design being run is a sub-block, it is recommended that only the regular nets be checked with this command.

5.10.2 Inputs and Outputs

Inputs:
- .
- No special inputs are necessary from within the SE environment.

Outputs:
- DEF of completed design
- Verilog netlist
- GDS output at this time

5.10.3 Executing verification in SE

Geometry and connectivity verification can be done from either the GUI or batch commands. Here we use batch commands.

```
verify geometry ;
report info filename verify_rpt all;
verify connectivity ;
```

5.11 Static Timing Analysis

The parasitic DSPF file created from Hyperextract can be directly read into PKS.

Once the DSPF for a routed design has been backannotated, you can run static timing analysis and, if necessary, perform additional In-Place Optimizations. To do this you will read the SPF for the design back into PKS to create predictors for the nets. These predictors are used to keep track of the difference between the PKS prediction for a net and the actual delay of the net.

5.11.1 The main actions in Static Timing Analysis

- Verify all timing constraints are being met, including setup/hold test mode timing.
- Write SDF file for Verilog post-layout timing simulation

5.11.2 STA Inputs and Outputs

Inputs:
- The PKS database should contain all the necessary physical placement and connectivity information.
- SPF from Hyperextract

Outputs:
- Timing Report
- Clock report
- SDF file
- Current verilog design file

5.11.3 Generating Timing Reports

The latest global routed database is reloaded into PKS. The DSPF generated in the parasitic extraction step is back annotated into the PKS database. The timing now will be based on the delays calculated with actual parasitic values.

The tcl file final_sta.tcl loads in the libraries, the design, the DSPF and then reports the timing. The following commands are all that's needed to generate the timing report.

```
read_lef_files $LEF_DESIGN_FILES
read_lut_file $LUT_DESIGN
read_design_adb $ADB_GROUTED
read_spf -verbose $DSPF_FINAL
report_timing > $RPT_TIM_FINAL
```

Once this report is generated it must be examined to determine if timing is met. If not then further optimizations must be done.

5.12 InPlace Optimization

In-place optimization attempts to fix timing violations that exist after final routing. Changes to the design are limited to those actions that have minimum impact on the final routed design. The design is iterated through PKS and back into SE for incremental placement and rerouting.

An IPO can be considered a limited Engineering Change Order (ECO). ECO is a process that reads a new logical netlist, compares it to the existing layout, and changes the layout where there are differences. During ECO routing, the router tries to reroute nets that have been modified.

During global ECO routing, the router removes the routes for nets that have been modified and reroutes them. Also, during final ECO routing, the router runs final routing on a routed design that has already had at least one final routing pass.

An ECO input file has the same syntax as a netlist DEF file. The ECO system automatically determines the differences between the new netlist and the database in memory. All such differences cause ECO actions. For example, ECO deletes nets that appear in the database but not in the new netlist. ECO also adds nets or cells that occur in the new netlist but not in the design. Therefore, the new netlist must be complete, even if you want to make successive changes to the same database.

When the design is passed to WRoute (shown as Detail Route in the flow) it is done so in an ECO mode. The router will then only reroute affected nets, not the complete design.

5.12.1 The main actions in the In-Place Optimizations

- Load the DSPF file into PKS.
- Check timing.
- Perform a post route optimization.
- Send the modified DEF to WRoute in ECO mode.
- Route only those parts of the netlist that have been modified.

5.12.2 Inputs and Outputs

Inputs:
- Timing library data file(s).
- LEF physical library data file(s).
- The hierarchical gate/macro-level netlist for the entire design.

- The DEF, flat physical data containing floorplanning, power design, clock tree, and global routing.

Output:

- Timing report.
- A modified DEF containing changed elements.

5.13 Assura Physical Verification

Assura tools provide a physical verification suite. These tools distinguish themselves from the SE verification utilities in that they provide verification down to the device level. Assura checks the layout against the geometric spacing rules that you define in your process rules files. Assura also compares the Layout Versus Schematic (LVS).

For the LVS flow, the Verilog netlist must be brought into Composer using verilogIn.

5.13.1 Inputs and Outputs:

Inputs:

- GDSII from SE of the P&R verified design
- GDSII or DFII layouts for all the devices and standard cells
- Verilog structural netlist for the top level design
- DRC, device extraction and device comparison (LVS) rules

Outputs:

- ASCII and binary files containing Assura created data and viewable results for DRC and LVS debugging.

5.13.2 Setting up Assura DRC environment

You must have the variable AMSPATH defined to point to the location where the bicmos7hp library kit is installed.

Copy the ASCII technology file from the design kit to the current directory.

```
cp $AMSPATH/bicmos7hp/rel/bicmos7hp/techfile.asc   .
```

Copy the GDS2 to DFII layer map file from the design kit to the current directory.

```
cp $AMSPATH/bicmos7hp/rel/bicmos7hp/gds2cds.map .
```

5.13.2.1 Streaming in the Data

The cds.lib must have definitions of bicmos7hp technology library containing the layer definitions and primitives for this technology containing the standard cells.

From unix prompt type ***icca*** to invoke the DFII interface
From the icca main window select **File -> Inport -> Stream**.

5.13.2.2 Filling in the forms

The forms need to be filled in once. Using the **Save** button will save a template file that can later be edited with new gds name. Using the edited template file will allow the DRC checking to be run in a more automated fashion.

The following describes entries as they should be made to the forms.

Input file The design GDSII file generated from Silicon Ensemble.
Top Cell Name This is the name specified as the “Top structure name” in the Silicon Ensemble “Export GDSII” interface.
Output Select “Opus DB” to specify a target library.
Library Name A new library name.
ASCII Technology
File Name The name of the technology file copied to the current directory.
Scale UU/DBU This should be the same database resolution as the number specified in the Silicon Ensemble “Export GDSII” interface.

5.13.2.3 Sample Template File

As stated earlier once this template is created it is easy to modify the “inFile” with the name of your new GDS file. Below is a sample of the template file created.

```
streamInKeys = list(nil
        'runDir   "."
        'inFile   "LEF_TEST.new.gds2"
        'primaryCell            ""
        'libName  "tony2"
        'techfileName           "/reflibs/tmp/AMSPATH/bicmos7hp/
                                      V2.1.0.0/bicmos7hp/tech-
                                      file.asc"
        'scale    0.001000
        'units    "micron"
        'errFile  "tony.LOG"
        'refLib   t
        'hierDepth32
        'maxVertices            1024
        'checkPolygon           nil
        'snapToGrid             nil
        'arrayToSimMosaic       t
        'caseSensitivity        "preserve"
        'zeroPathToLine         "lines"
        'skipUndefinedLPP       nil
        'ignoreBoxnil
        'reportPrecision        nil
        'runQuiet nil
        'saveAtTheEnd           nil
        'noWriteExistCell nil
        'NOUnmappingLayerWarning  nil
        'cellMapTable           ""
        'layerTable             "/reflibs/tmp/AMSPATH/bicmos7hp/
```

```
                                        V2.1.0.0/bicmos7hp/gds2cds.map"
           'textFontTable          ""
           'restorePin             0
           'propMapTable           ""
           'propSeparator          ","
           'userSkillFile          ""
         'refLibOrder        "cmos7hpv12"
           'rodDir   ""
```

5.13.2.4 Running the DRC

- Edit the template file previously created (e.g. pipoin.template) and add your gds2 filename and the name of the library you wish to create.
- Stream in the data
  ```
  pipo strmin  filename.template  >& pipoin.log
  ```
- Check the log, pipoin.log, file created above for errors.
- Start Assura (icca, layoutplus, icfb depending on version you own).
- Select Run DRC
- Check that Technology name is set correctly
- All other info should be filled in automatically
- Run
- To rerun in batch mode modify the .rsf file then

assura lef_test.rsf >& assura.log &

- Check .err files for errors.

5.13.3LVS Setup

The LVS setup is slightly more technical due to database formatting issues. The following is a simplified procedure. Refer to the Assura documentation for more detailed descriptions of the setup involved.

5.13.3.1 Modifying layer info

The Silicon Ensemble output GDSII utility writes all the information to a layer purpose 0. The Assura LVS deck is setup to expect the text in purpose label only. Therefore it will fail. In addition when the GL1 is created the text will be converted into polygons and it could create DRC rule violations.

A SKILL code utility is provided to make the necessary layer purpose modifications.

```
change_text_purpose.il
```

To use the SKILL code
In the CIW type:

```
load("<script_location>/change_text_purpose.il")
```

Bring up the design layout window by using the library browser. When the layout is up, in the CIW enter:

```
sige_change_text_purpose()
```

5.13.3.2 Running LVS

- Start **icca**
- Open the cellview dft_final layout from the library design.
- Bring up the Assura Menu
- Bring up the Assura Run LVS form
- In the “Assura Run LVS form, select **Technology: av7hpLVS6**. All the necessary file fields will automatically be filled in from the Technology directory.
- Click OK to start the LVS. A Progress form appears.
- If you press Watch Log File, a log window comes up with the current run status information.
- After the run completes a window pops up asking if you want to see the run results. Press Yes.
- Another small window pops up to say there are no DRC errors. Click OK in this window.
- The DRC errors here refer to softconnect errors, which are considered DRC errors by Assura but are often checked during LVS.

5.14 NC-Verilog Simulation

Functional simulation is an essential step in determining if the routed design meets the functional and timing requirements. The NC verilog is an extremely fast *Native Compiled* simulator capable of mixed language simulation.

After place and route the simulation is run using the gate-level netlist with an SDF file for back annotation. The SDF file contains the actual timing delay information resulting from the layout. This follow-up simulation of the design wil verify that the functionality is maintained after delays are taken into account. If the delays are greater than expected , results of the simulation will indicate where actual outputs did not meet expected results.

5.14.1 Inputs and Outputs:

Inputs:

- Verilog netlist
- An SDF generated from the completed design
- Testbench
- Expected output file
- Verilog simulation library.

Outputs:

- Feedback to the console: pass/fail
- Comparison file, showing results of expected outputs and actuals
-

5.14.2 Running a simulation

A test bench must be provided for the simulation. The test bench provides all of the IO communication with the unit (design) under test and the stimulus vectors for the test. The test bench for this design is dft_test.v.

The verilog file for simulation is written out of PKS after the static timing analysis is complete. The verilog file must be modified to add the sdf annotation to the design when being simulated. The following lines must be added after the wire definitions as in the example below.

```
// Generated by ac_shell v4.0-s007 on Mon Jun 18 12:18:42 PST 2001.

// Restrictions concerning the use of Ambit BuildGates are covered in
                                      the
// license agreement.  Distribution to third party EDA vendors is
// strictly prohibited.

module align_samples(wdata_AS, waddr_AS, write_AS, allSamplesReceived,
                                      FPnum,
                FP_valid, reset_inv, clk, clk_p467, clk_p4857,
                                      clk_p2611,
                clk_p31, clk_p16, clk_p481, clk_p3281, clk_p606,
                                      clk_p22,
                reset_p2721, reset_p3502, reset_p395);

        output [63:0] wdata_AS;
        output [6:0] waddr_AS;
        output write_AS;
        output allSamplesReceived;
        input [31:0] FPnum;
        input FP_valid;
        input reset_inv;
        input clk;
        input clk_p467;
        input clk_p4857;
        input clk_p2611;
        input clk_p31;
        input clk_p16;
        input clk_p481;
        input clk_p3281;
        input clk_p606;
        input clk_p22;
        input reset_p2721;
        input reset_p3502;
        input reset_p395;

        wire [6:0] sampleCnt;

        initial
         begin
          $sdf_annotate("dft_chip.final.sdf", align_samples);
         end

        BUFFER_C i_48515(.A(n_87233), .Z(n_139686));
        BUFFER_C i_48510(.A(n_87221), .Z(n_139678));
```

The simulation is executed by running a script file, sim_struct_sdf, as shown below. Following the simulation run the output is compared to an expected results file. If the comparison is coherent then the simulation is successful.

```
#!/bin/ksh
#
ncverilog -v ibm.sige.library.v \
  dft_test.v dft_chip.final_sim.v constant_rom.v \
  +ncelabargs+-neg_tchk +define+SINE +define+SINGLE $*

cmp peaks.out peaks.exp
if [ "$?" -eq 0 ]
  then
  echo "Simulation successful!"
else
  echo "Simulation failed"
```

Appendix A se.ini

SE.INI file

```
set v WROUTE.TOPLAYER.LIMIT 5 ;
set var output.def.spnet.wildcard true ;
SET VAR PLAN.LOWERLEFT.ORIGIN TRUE           ;# Sets the origin of the
                                       design to be
                                                # located on the Lower
                                       left corner.
SET VAR INPUT.VERILOG.POWER.NET ’VDD!’        ;# Set verilog power net
                                       name to vdd
SET VAR INPUT.VERILOG.GROUND.NET ’GND!’        ;# Set verilog ground
                                       net name to vss
SET VAR INPUT.VERILOG.LOGIC.1.NET ’VDD!’       ;# Set verilog tie ups
                                       to net name to vdd
SET VAR INPUT.VERILOG.LOGIC.0.NET ’GND!’      ;# Set verilog tie downs
                                       to net name to vss
SET VAR INPUT.VERILOG.SPECIAL.NETS ’VDD! GND!’;# Set verilog special
                                       net names
                                                # to vdd and vss
SET VAR FROUTE.MANUFACTURING.YGRID 2          ;# Set the Y manufactur-
                                       ing grid for Wroute
SET VAR FROUTE.MANUFACTURING.XGRID 2          ;# Set the X manufactur-
                                       ing grid for Wroute
SET VAR QPLACE.LLC.PREWIRE.KEEPOUT FALSE      ;# Prevents Qplace from
                                       placing cells
                                                # under the prewires.
SET VAR SROUTE.VIARULE.OVERHANG.COMMON TRUE   ;# Forces the power
                                       router to consider overhangs
                                                # against the layer
                                       direction. Used to prevent
                                                # problems with M1 fol-
                                       lowpins when the gap
                                                # between M2 VDD! and
                                       GND! stripes is 0.4
SET VAR SROUTE.VIA.SNAPMANUFACTURINGGRID TRUE ;# Snap cuts on via rule
                                       generate to the
                                                # manufacturing grid.
SET VAR SROUTE.STRIPE.SNAP.RGRID "GRID"       ;# Snap the stripes to
                                       the routing grid.
SET VAR PLACE.LLC.PREROUTEDWIRECHECK FALSE    ;# Allows filler cells to
                                       be inserted
                                                # under the stripes.
SET VAR WROUTE.ALLOW.PORT.SHORTS FALSE        ;# Provides more flexi-
                                       bility to Wroute used
                                            # due to number of offgrid
                                       pins.
SET VAR INPUT.DEF.ECO.TRYNETRENAME TRUE      ;# Used during ECO to try
                                       to preserve clock tree cells.


# ;
# Silicon Ensemble environment variables ;
set v  db.design.dir      "dbs" ;
# ;
# Set relative sizes to multiple of metal2 rGrid pitch ;
# NOTE: This values need to be adjusted depending on design sizes.
```

```
#
set v draw.size.small           9600 ; #  120 tracks ;
set v draw.size.medium          3200 ; #  400 tracks ;
set v draw.size.big           160000 ; #  2000 tracks ;
set v snapRadius.small            20 ; #  1/4 track ;
set v snapRadius.medium           80 ; #  1 track ;
set v snapRadius.large           400 ; #  5 tracks ;
# ;
# Silicon Ensemble floorplan variables - recommened  ;
set v  plan.rgrid.M1offset            40 ;
set v  plan.rgrid.M2offset            40 ;
set v  plan.rgrid.M3offset            40 ;
set v  plan.rgrid.M4offset            40 ;
set v  plan.rgrid.M5offset            40 ;
set v  plan.rgrid.M6offset           256 ;
# ;
# Silicon Ensemble route variables - recommened  ;
# NOTE: With autoAbgen models, none or very little performance loss in
                                      offgrid mode. ;


set v  groute.Allow.OffGrid.PinAccess true ;
set v  froute.Allow.OffGrid.PinAccess true ;
set v  froute.Avoid.OffGrid.Blockage  true ;
set v  froute.Build.OffGrid.SPins     true ;


#   set v  froute.allow.pinaccess.atblockageedge true ;
# ;
# Silicon Ensemble verification variables - recommened ;
set v  verify.geometry.allow.same.cell.violations   true ;
set v  verify.geometry.report.samenet     false ;
#
#Routing Layers
#
SET V DRAW.BLOCKAGE.1.COLOR     2;
SET V DRAW.BLOCKAGE.1.FILL      12;
SET V DRAW.BLOCKAGE.1.LINE      0;
SET V DRAW.GROUTE.1.COLOR       2;
SET V DRAW.GROUTE.1.FILL        0;
SET V DRAW.GROUTE.1.LINE        0;
SET V DRAW.LPORT.1.COLOR        2;
SET V DRAW.LPORT.1.FILL         0;
SET V DRAW.LPORT.1.LINE         0;
SET V DRAW.PIN.1.COLOR  2;
SET V DRAW.PIN.1.FILL   0;
SET V DRAW.PIN.1.LINE   0;
SET V DRAW.RGRID.1.COLOR        5;
SET V DRAW.RGRID.1.FILL         0;
SET V DRAW.RGRID.1.LINE         0;
SET V DRAW.SWIRE.1.COLOR        2;
SET V DRAW.SWIRE.1.FILL         14;
SET V DRAW.SWIRE.1.LINE         0;
SET V DRAW.WIRE.1.COLOR         2;
SET V DRAW.WIRE.1.FILL  14;
SET V DRAW.WIRE.1.LINE  0;
SET V DRAW.BLOCKAGE.2.COLOR     1;
SET V DRAW.BLOCKAGE.2.FILL      12;
SET V DRAW.BLOCKAGE.2.LINE      0;
SET V DRAW.GROUTE.2.COLOR       1;
SET V DRAW.GROUTE.2.FILL        0;
SET V DRAW.GROUTE.2.LINE        0;
SET V DRAW.LPORT.2.COLOR        1;
SET V DRAW.LPORT.2.FILL         0;
SET V DRAW.LPORT.2.LINE         0;
SET V DRAW.PIN.2.COLOR  1;
SET V DRAW.PIN.2.FILL   0;
SET V DRAW.PIN.2.LINE   0;
SET V DRAW.RGRID.2.COLOR        5;
SET V DRAW.RGRID.2.FILL         0;
SET V DRAW.RGRID.2.LINE         0;
SET V DRAW.SWIRE.2.COLOR        1;
SET V DRAW.SWIRE.2.FILL         15;
SET V DRAW.SWIRE.2.LINE         0;
SET V DRAW.WIRE.2.COLOR         1;
SET V DRAW.WIRE.2.FILL  15;
SET V DRAW.WIRE.2.LINE  0;


SET V DRAW.BLOCKAGE.3.COLOR     3;
SET V DRAW.BLOCKAGE.3.FILL      12;
SET V DRAW.BLOCKAGE.3.LINE      0;
SET V DRAW.GROUTE.3.COLOR       3;
SET V DRAW.GROUTE.3.FILL        0;
SET V DRAW.GROUTE.3.LINE        0;
SET V DRAW.LPORT.3.COLOR        3;
SET V DRAW.LPORT.3.FILL         0;
SET V DRAW.LPORT.3.LINE         0;
SET V DRAW.PIN.3.COLOR  3;
SET V DRAW.PIN.3.FILL   0;
SET V DRAW.PIN.3.LINE   0;
SET V DRAW.RGRID.3.COLOR        5;
SET V DRAW.RGRID.3.FILL         0;
SET V DRAW.RGRID.3.LINE         0;
SET V DRAW.SWIRE.3.COLOR        3;
SET V DRAW.SWIRE.3.FILL         16;
SET V DRAW.SWIRE.3.LINE         0;
SET V DRAW.WIRE.3.COLOR         3;
SET V DRAW.WIRE.3.FILL  16;
SET V DRAW.WIRE.3.LINE  0;


SET V DRAW.BLOCKAGE.4.COLOR     1;
SET V DRAW.BLOCKAGE.4.FILL      12;
SET V DRAW.BLOCKAGE.4.LINE      0;
SET V DRAW.GROUTE.4.COLOR       1;
SET V DRAW.GROUTE.4.FILL        0;
SET V DRAW.GROUTE.4.LINE        0;
SET V DRAW.LPORT.4.COLOR        1;
SET V DRAW.LPORT.4.FILL         0;
SET V DRAW.LPORT.4.LINE         0;
SET V DRAW.PIN.4.COLOR  1;
SET V DRAW.PIN.4.FILL   0;
SET V DRAW.PIN.4.LINE   0;
SET V DRAW.RGRID.4.COLOR        5;
SET V DRAW.RGRID.4.FILL         0;
SET V DRAW.RGRID.4.LINE         0;
SET V DRAW.SWIRE.4.COLOR        1;
SET V DRAW.SWIRE.4.FILL         17;
SET V DRAW.SWIRE.4.LINE         0;
SET V DRAW.WIRE.4.COLOR         1;
SET V DRAW.WIRE.4.FILL  17;
SET V DRAW.WIRE.4.LINE  0;


SET V DRAW.BLOCKAGE.5.COLOR     2;
SET V DRAW.BLOCKAGE.5.FILL      12;
SET V DRAW.BLOCKAGE.5.LINE      0;
SET V DRAW.GROUTE.5.COLOR       2;
SET V DRAW.GROUTE.5.FILL        0;
SET V DRAW.GROUTE.5.LINE        0;
SET V DRAW.LPORT.5.COLOR        2;
SET V DRAW.LPORT.5.FILL         0;
SET V DRAW.LPORT.5.LINE         0;
SET V DRAW.PIN.5.COLOR  2;
SET V DRAW.PIN.5.FILL   0;
SET V DRAW.PIN.5.LINE   0;
SET V DRAW.RGRID.5.COLOR        5;
SET V DRAW.RGRID.5.FILL         0;
SET V DRAW.RGRID.5.LINE         0;
SET V DRAW.SWIRE.5.COLOR        2;
SET V DRAW.SWIRE.5.FILL         14;
SET V DRAW.SWIRE.5.LINE         0;
SET V DRAW.WIRE.5.COLOR         2;
SET V DRAW.WIRE.5.FILL  14;
SET V DRAW.WIRE.5.LINE  0;


SET V DRAW.BLOCKAGE.6.COLOR     3;
SET V DRAW.BLOCKAGE.6.FILL      12;
SET V DRAW.BLOCKAGE.6.LINE      0;
SET V DRAW.GROUTE.6.COLOR       3;
SET V DRAW.GROUTE.6.FILL        0;
SET V DRAW.GROUTE.6.LINE        0;
SET V DRAW.LPORT.6.COLOR        3;
SET V DRAW.LPORT.6.FILL         0;
SET V DRAW.LPORT.6.LINE         0;
SET V DRAW.PIN.6.COLOR  3;
SET V DRAW.PIN.6.FILL   0;
SET V DRAW.PIN.6.LINE   0;
SET V DRAW.RGRID.6.COLOR        5;
SET V DRAW.RGRID.6.FILL         0;
SET V DRAW.RGRID.6.LINE         0;
SET V DRAW.SWIRE.6.COLOR        3;
SET V DRAW.SWIRE.6.FILL         15;
SET V DRAW.SWIRE.6.LINE         0;
SET V DRAW.WIRE.6.COLOR         3;
SET V DRAW.WIRE.6.FILL  15;
SET V DRAW.WIRE.6.LINE  0;


#
```

```
#Master Slice Layers
#
SET V DRAW.BLOCKAGE.7.COLOR     0;
SET V DRAW.BLOCKAGE.7.FILL      12;
SET V DRAW.BLOCKAGE.7.LINE      0;
SET V DRAW.GROUTE.7.COLOR       0;
SET V DRAW.GROUTE.7.FILL        0;
SET V DRAW.GROUTE.7.LINE        0;
SET V DRAW.LPORT.7.COLOR        0;
SET V DRAW.LPORT.7.FILL         0;
SET V DRAW.LPORT.7.LINE         0;
SET V DRAW.PIN.7.COLOR  0;
SET V DRAW.PIN.7.FILL   0;
SET V DRAW.PIN.7.LINE   0;
SET V DRAW.RGRID.7.COLOR        5;
SET V DRAW.RGRID.7.FILL         0;
SET V DRAW.RGRID.7.LINE         0;
SET V DRAW.SWIRE.7.COLOR        0;
SET V DRAW.SWIRE.7.FILL         0;
SET V DRAW.SWIRE.7.LINE         0;
SET V DRAW.WIRE.7.COLOR         0;
SET V DRAW.WIRE.7.FILL  0;
SET V DRAW.WIRE.7.LINE  0;


#
#Cut Layers
#
SET V DRAW.BLOCKAGE.8.COLOR     7;
SET V DRAW.BLOCKAGE.8.FILL      12;
SET V DRAW.BLOCKAGE.8.LINE      0;
SET V DRAW.GROUTE.8.COLOR       7;
SET V DRAW.GROUTE.8.FILL        0;
SET V DRAW.GROUTE.8.LINE        0;
SET V DRAW.LPORT.8.COLOR        7;
SET V DRAW.LPORT.8.FILL         0;
SET V DRAW.LPORT.8.LINE         0;
SET V DRAW.PIN.8.COLOR  7;
SET V DRAW.PIN.8.FILL   0;
SET V DRAW.PIN.8.LINE   0;
SET V DRAW.RGRID.8.COLOR        5;
SET V DRAW.RGRID.8.FILL         0;
SET V DRAW.RGRID.8.LINE         0;
SET V DRAW.SWIRE.8.COLOR        7;
SET V DRAW.SWIRE.8.FILL         0;
SET V DRAW.SWIRE.8.LINE         0;
SET V DRAW.WIRE.8.COLOR         7;
SET V DRAW.WIRE.8.FILL  0;
SET V DRAW.WIRE.8.LINE  0;


SET V DRAW.BLOCKAGE.9.COLOR     4;
SET V DRAW.BLOCKAGE.9.FILL      12;
SET V DRAW.BLOCKAGE.9.LINE      0;
SET V DRAW.GROUTE.9.COLOR       4;
SET V DRAW.GROUTE.9.FILL        0;
SET V DRAW.GROUTE.9.LINE        0;
SET V DRAW.LPORT.9.COLOR        4;
SET V DRAW.LPORT.9.FILL        0;
SET V DRAW.LPORT.9.LINE        0;
SET V DRAW.PIN.9.COLOR  4;
SET V DRAW.PIN.9.FILL   0;
SET V DRAW.PIN.9.LINE   0;
SET V DRAW.RGRID.9.COLOR       5;
SET V DRAW.RGRID.9.FILL        0;
SET V DRAW.RGRID.9.LINE        0;
SET V DRAW.SWIRE.9.COLOR       4;
SET V DRAW.SWIRE.9.FILL        0;
SET V DRAW.SWIRE.9.LINE        0;
SET V DRAW.WIRE.9.COLOR        4;
SET V DRAW.WIRE.9.FILL  0;
SET V DRAW.WIRE.9.LINE  0;


SET V DRAW.BLOCKAGE.10.COLOR   5;
SET V DRAW.BLOCKAGE.10.FILL    12;
SET V DRAW.BLOCKAGE.10.LINE    0;
SET V DRAW.GROUTE.10.COLOR     5;
SET V DRAW.GROUTE.10.FILL      0;
SET V DRAW.GROUTE.10.LINE      0;
SET V DRAW.LPORT.10.COLOR      5;
SET V DRAW.LPORT.10.FILL       0;
SET V DRAW.LPORT.10.LINE       0;
SET V DRAW.PIN.10.COLOR        5;
SET V DRAW.PIN.10.FILL  0;
SET V DRAW.PIN.10.LINE  0;
SET V DRAW.RGRID.10.COLOR      5;
SET V DRAW.RGRID.10.FILL       0;
SET V DRAW.RGRID.10.LINE       0;
SET V DRAW.SWIRE.10.COLOR      5;
SET V DRAW.SWIRE.10.FILL       0;
SET V DRAW.SWIRE.10.LINE       0;
SET V DRAW.WIRE.10.COLOR       5;
SET V DRAW.WIRE.10.FILL        0;
SET V DRAW.WIRE.10.LINE        0;


SET V DRAW.BLOCKAGE.11.COLOR   7;
SET V DRAW.BLOCKAGE.11.FILL    12;
SET V DRAW.BLOCKAGE.11.LINE    0;
SET V DRAW.GROUTE.11.COLOR     7;
SET V DRAW.GROUTE.11.FILL      0;
SET V DRAW.GROUTE.11.LINE      0;
SET V DRAW.LPORT.11.COLOR      7;
SET V DRAW.LPORT.11.FILL       0;
SET V DRAW.LPORT.11.LINE       0;
SET V DRAW.PIN.11.COLOR        7;
SET V DRAW.PIN.11.FILL  0;
SET V DRAW.PIN.11.LINE  0;
SET V DRAW.RGRID.11.COLOR      5;
SET V DRAW.RGRID.11.FILL       0;
SET V DRAW.RGRID.11.LINE       0;
SET V DRAW.SWIRE.11.COLOR      7;
SET V DRAW.SWIRE.11.FILL       0;
SET V DRAW.SWIRE.11.LINE       0;
SET V DRAW.WIRE.11.COLOR       7;
SET V DRAW.WIRE.11.FILL         0;
SET V DRAW.WIRE.11.LINE         0;


SET V DRAW.BLOCKAGE.12.COLOR    4;
SET V DRAW.BLOCKAGE.12.FILL     12;
SET V DRAW.BLOCKAGE.12.LINE     0;
SET V DRAW.GROUTE.12.COLOR      4;
SET V DRAW.GROUTE.12.FILL       0;
SET V DRAW.GROUTE.12.LINE       0;
SET V DRAW.LPORT.12.COLOR       4;
SET V DRAW.LPORT.12.FILL        0;
SET V DRAW.LPORT.12.LINE        0;
SET V DRAW.PIN.12.COLOR         4;
SET V DRAW.PIN.12.FILL  0;
SET V DRAW.PIN.12.LINE  0;
SET V DRAW.RGRID.12.COLOR       5;
SET V DRAW.RGRID.12.FILL        0;
SET V DRAW.RGRID.12.LINE        0;
SET V DRAW.SWIRE.12.COLOR       4;
SET V DRAW.SWIRE.12.FILL        0;
SET V DRAW.SWIRE.12.LINE        0;
SET V DRAW.WIRE.12.COLOR        4;
SET V DRAW.WIRE.12.FILL         0;
SET V DRAW.WIRE.12.LINE         0;


SET V DRAW.BLOCKAGE.13.COLOR    5;
SET V DRAW.BLOCKAGE.13.FILL     12;
SET V DRAW.BLOCKAGE.13.LINE     0;
SET V DRAW.GROUTE.13.COLOR      5;
SET V DRAW.GROUTE.13.FILL       0;
SET V DRAW.GROUTE.13.LINE       0;
SET V DRAW.LPORT.13.COLOR       5;
SET V DRAW.LPORT.13.FILL        0;
SET V DRAW.LPORT.13.LINE        0;
SET V DRAW.PIN.13.COLOR         5;
SET V DRAW.PIN.13.FILL  0;
SET V DRAW.PIN.13.LINE  0;
SET V DRAW.RGRID.13.COLOR       5;
SET V DRAW.RGRID.13.FILL        0;
SET V DRAW.RGRID.13.LINE        0;
SET V DRAW.SWIRE.13.COLOR       5;
SET V DRAW.SWIRE.13.FILL        0;
SET V DRAW.SWIRE.13.LINE        0;
SET V DRAW.WIRE.13.COLOR        5;
SET V DRAW.WIRE.13.FILL         0;
SET V DRAW.WIRE.13.LINE         0;


#
#Overlap and Extra Layers
#
SET V DRAW.BLOCKAGE.14.COLOR    0;
SET V DRAW.BLOCKAGE.14.FILL     12;
SET V DRAW.BLOCKAGE.14.LINE     0;
SET V DRAW.GROUTE.14.COLOR      0;
SET V DRAW.GROUTE.14.FILL       0;
SET V DRAW.GROUTE.14.LINE       0;
SET V DRAW.LPORT.14.COLOR       0;
SET V DRAW.LPORT.14.FILL        0;
SET V DRAW.LPORT.14.LINE        0;
SET V DRAW.PIN.14.COLOR         0;
SET V DRAW.PIN.14.FILL  0;
SET V DRAW.PIN.14.LINE  0;
SET V DRAW.RGRID.14.COLOR       5;
SET V DRAW.RGRID.14.FILL        0;
SET V DRAW.RGRID.14.LINE        0;
SET V DRAW.SWIRE.14.COLOR       0;
SET V DRAW.SWIRE.14.FILL        0;
SET V DRAW.SWIRE.14.LINE        0;
SET V DRAW.WIRE.14.COLOR        0;
SET V DRAW.WIRE.14.FILL         0;
SET V DRAW.WIRE.14.LINE         0;


SET V DRAW.LAYER.ORDER  "7 1 2 3 4 5 6 8 9 10 11 12 13 ";
SET V DRAW.GROUTE.LAYERSET      "1 2 3 4 5 6 ";
SET V DRAW.BLOCKAGE.LAYERSET    "1 2 3 4 5 6 8 9 10 11 12 13 ";
SET V DRAW.LPORT.LAYERSET       "1 2 3 4 5 6 8 9 10 11 12 13 ";
SET V DRAW.PIN.LAYERSET         "1 2 3 4 5 6 8 9 10 11 12 13 ";
SET V DRAW.RGRID.LAYERSET       "1 2 3 4 5 6 8 9 10 11 12 13 ";
SET V DRAW.SWIRE.LAYERSET       "1 2 3 4 5 6 8 9 10 11 12 13 ";
SET V DRAW.WIRE.LAYERSET        "1 2 3 4 5 6 8 9 10 11 12 13 ";
```

Appendix B PKS Shell Script

run_pks_job

```
#!/bin/csh

if ($#argv < 2) then
        echo "Usage:"
        echo "run_pks_job <run_name> <tcl_script>"
        exit 1
else
        echo "run_name: $1:t"
        echo "tcl_script:       $2:t"
endif

#####   Set environmental variables   #####
setenv DESIGN_NAME      dft_chip
setenv ROOT_DIR         /ambit/sipg/ibm/Sige/digital/flowkit/dft_chip
setenv LIB_DIR          /ambit/sipg/ibm/Sige/digital/digitalkit
setenv LIB_TYPE         ola

setenv RUN_NAME         $1:t
setenv RUN_DIR          $ROOT_DIR/rundata/$RUN_NAME
setenv SCRIPT_DIR       $ROOT_DIR/scripts

#####   Set up for ola run   #####
if ($LIB_TYPE == "ola") then
        set    oladir =                $LIB_DIR/ndr/SOL
        setenv DCMRULESPATH            ${oladir}:${oladir}/%RULENAME
        setenv DCMRULEPATH             ${oladir}/DCMinterface_SOL
        setenv DCMTABLEPATH            ${oladir}/tables:${oladir}/
                                         %RULENAME
        setenv DCMBOMPATH              ${oladir}
        setenv DCMDoRangeCheck         0
        setenv DCMWireLoadLevels       6mtwb

        if ($?LD_LIBRARY_PATH) then
                setenv LD_LIBRARY_PATH  $oladir/lib:$LD_LIBRARY_PATH
        else
                setenv LD_LIBRARY_PATH  $oladir/lib
        endif
endif

#####   Create run directories if necessary   #####

setenv CMN_DIR  $ROOT_DIR/rundata/common; if (! -e $CMN_DIR) mkdir -p
                                          $CMN_DIR
setenv ADB_DIR  $RUN_DIR/adb;           if (! -e $ADB_DIR)   mkdir -p
                                          $ADB_DIR
setenv DEF_DIR  $RUN_DIR/def;          if (! -e $DEF_DIR)   mkdir $DEF_DIR
setenv PKS_DIR  $RUN_DIR/pks;          if (! -e $PKS_DIR)   mkdir $PKS_DIR
setenv RPT_DIR  $RUN_DIR/rpt;          if (! -e $RPT_DIR)   mkdir $RPT_DIR
setenv HDL_DIR  $RUN_DIR/verilog;      if (! -e $HDL_DIR)   mkdir $HDL_DIR
setenv WDB_DIR  $RUN_DIR/wdb;          if (! -e $WDB_DIR)   mkdir $WDB_DIR
setenv SPF_DIR  $RUN_DIR/spf;          if (! -e $SPF_DIR)   mkdir $SPF_DIR
setenv SDF_DIR  $RUN_DIR/sdf;          if (! -e $SDF_DIR)   mkdir $SDF_DIR
```

```
#####   Define the tcl, cmd, and log files   #####
set tcl_script = $SCRIPT_DIR/tcl/$2:t
if (! -f $tcl_script) then
        echo "run_pks_job: ERROR: $tcl_script not found\!"
        exit 1
endif

set fhead = ‘echo $tcl_script:t | sed -e ’s/.tcl$//’‘
set cmd_file =          $PKS_DIR/${fhead}.cmd
set log_file =          $PKS_DIR/${fhead}.log

#####   Set license variable and put ambit bin at the head of path   #####
setenv LM_LICENSE_FILE  11536@ra:5280@mammoth
set ambit_bin = /ambit/cm/prod/v4.0-s007/release/BuildGates/version/bin
set path = ($ambit_bin ‘echo $path | /usr/bin/sed -e {s@$ambit_bin@@}‘)

#####   Run PKS   #####
cd $RUN_DIR
pks_shell -continue -cmdfile $cmd_file -logfile $log_file $tcl_script
```

Appendix C PKS setup.tcl

```
uname -a

############################################################################
                                             ####
#
#       Set ambit globals
#
set_global echo_commands        true
set_global line_length          166
set_global fix_multiport_nets   true
set_global slew_time_limit      0.5
set_global report_timing_format \
        {instance cell arc net delay arrival load fanout slew}
set_table_style \
        -name report_timing \
        -max_widths {30,30,30,30,5,5,5,3,4} \
        -indent 0
set_global pks_do_place_option [concat \
        -timing_driven \
        -free_track_percent_1 0.20 \
        -free_track_percent_2 0.88 \
        -free_track_percent_3 0.88 \
        -free_track_percent_4 0.88 \
        -free_track_percent_5 0.88 \
        -free_track_percent_6 0.00 \
        ]
set_global hdl_verilog_out_unconnected_style full

############################################################################
                                             ####
#
#       Define variables
#
set DESIGN_NAME         dft_chip
set SOURCE_DIR          $env(ROOT_DIR)/source
set TECH_DIR            $env(ROOT_DIR)/techlib

set ALF_FILES           "$env(LIB_DIR)/synthesis/ambit/IBM_SIGE12E.alf"
set DONT_USE_CELLS_FILE $env(LIB_DIR)/synthesis/sige.dont_use_cells.tcl

set CTPKS_USABLE_CELLS  "CLK_I  CLK_K   CLK_M   CLK_O   CLK_Q \
                        CLKI_I  CLKI_K  CLKI_M  CLKI_O  CLKI_Q"

set HDL_SOURCE_FILES    [concat         dft_chip.v      adjust.v        \
                        align_samples.v barrell.v       barrelr.v       \
                        bit_reverse.v   bit_reverse_1.v cmult.v         \
                        dft_controller.v                expcomp.v       \
                        fadd_clocked.v  fmult.v         fneg.v          \
                        imult.v         int2fp.v        leadsign.v      \
                        manadd.v        manmux.v        mul_add.v       \
                        mux_readport.v  mux_writeport.v regfile_128_8.v]

#       Reference library lef files
set LEF_REF_FILES       [concat \
                        $env(LIB_DIR)/lef/tech7hp.6mt.lef \
                        $env(LIB_DIR)/lef/abstract7hp.cds.lef]
```

```
#       Updated design lef files
set LEF_DESIGN_FILES    [concat \
                        $TECH_DIR/${DESIGN_NAME}_tech.lef \
                        $env(LIB_DIR)/lef/abstract7hp.cds.lef]
#       Reference antenna lef file
set LEF_ANTENNA         $env(LIB_DIR)/lef/antenna7hp.6mt.lef
#       Reference layer utilization table
set LUT_REF             $env(LIB_DIR)/layer.utilization
#       Updated design layer utilization table
set LUT_DESIGN          $TECH_DIR/${DESIGN_NAME}_lut


############################################################################
                                             ####
#
#       Create the wroute lef configuration file which is passed to wroute
#       through the global _wr_config
#

set_global _wr_config $env(PKS_DIR)/wr_lef.cfg
set wr_lef_cfg [open [get_global _wr_config] w]
puts $wr_lef_cfg "inputLefName '$LEF_DESIGN_FILES $LEF_ANTENNA'"
close $wr_lef_cfg

############################################################################
                                             ####
#
#       Define default file names
#


############################################################################
                                             ####
#
#       Define default file names
#

set CONSTRAINTS_DIR     $SOURCE_DIR/constraints
set CONSTRAINTS_DC      $CONSTRAINTS_DIR/${DESIGN_NAME}_constraints.dc
set CONSTRAINTS_AMBIT   $CONSTRAINTS_DIR/${DESIGN_NAME}_constraints.tcl
set CONSTRAINTS_CTPKS   $CONSTRAINTS_DIR/${DESIGN_NAME}_ctpks_constraints.tcl

set ADB_GENERIC         $env(CMN_DIR)/${DESIGN_NAME}.generic.adb
set ADB_MAPPED          $env(ADB_DIR)/${DESIGN_NAME}.mapped.adb
set ADB_PLACED          $env(ADB_DIR)/${DESIGN_NAME}.placed.adb
set ADB_OPTIMIZED       $env(ADB_DIR)/${DESIGN_NAME}.optimized.adb
set ADB_CLOCKED         $env(ADB_DIR)/${DESIGN_NAME}.clocked.adb
set ADB_CLKOPT          $env(ADB_DIR)/${DESIGN_NAME}.clkopt.adb
set ADB_HOLDOPT         $env(ADB_DIR)/${DESIGN_NAME}.holdopt.adb
set ADB_GROUTED         $env(ADB_DIR)/${DESIGN_NAME}.grouted.adb
set ADB_IPO             $env(ADB_DIR)/${DESIGN_NAME}.ipo.adb
set ADB_IPO_GROUTED     $env(ADB_DIR)/${DESIGN_NAME}.ipo_grouted.adb

set HDL_GENERIC         $env(HDL_DIR)/${DESIGN_NAME}.generic.v
set HDL_MAPPED          $env(HDL_DIR)/${DESIGN_NAME}.mapped.v
set HDL_PLACED          $env(HDL_DIR)/${DESIGN_NAME}.placed.v
set HDL_OPTIMIZED       $env(HDL_DIR)/${DESIGN_NAME}.optimized.v
set HDL_CLOCKED         $env(HDL_DIR)/${DESIGN_NAME}.clocked.v
set HDL_CLKOPT         $env(HDL_DIR)/${DESIGN_NAME}.clkopt.v
set HDL_HOLDOPT        $env(HDL_DIR)/${DESIGN_NAME}.holdopt.v
set HDL_GROUTED        $env(HDL_DIR)/${DESIGN_NAME}.grouted.v
set HDL_IPO            $env(HDL_DIR)/${DESIGN_NAME}.ipo.v
set HDL_IPO_GROUTED    $env(HDL_DIR)/${DESIGN_NAME}.ipo_grouted.v
set HDL_FINAL_SIM      $env(HDL_DIR)/${DESIGN_NAME}.final_sim.v

set DEF_FLOORPLAN      $env(DEF_DIR)/${DESIGN_NAME}.floorplan.def
set DEF_PLACED         $env(DEF_DIR)/${DESIGN_NAME}.placed.def
set DEF_OPTIMIZED      $env(DEF_DIR)/${DESIGN_NAME}.optimized.def
set DEF_CLOCKED        $env(DEF_DIR)/${DESIGN_NAME}.clocked.def
set DEF_CLKOPT         $env(DEF_DIR)/${DESIGN_NAME}.clkopt.def
set DEF_HOLDOPT        $env(DEF_DIR)/${DESIGN_NAME}.holdopt.def
set DEF_GROUTED        $env(DEF_DIR)/${DESIGN_NAME}.grouted.def
set DEF_IPO            $env(DEF_DIR)/${DESIGN_NAME}.ipo.def
set DEF_IPO_GROUTED    $env(DEF_DIR)/${DESIGN_NAME}.ipo_grouted.def

set WDB                $env(WDB_DIR)/${DESIGN_NAME}.grouted.wdb
set WDB_IPO            $env(WDB_DIR)/${DESIGN_NAME}.ipo_grouted.wdb

set DSPF_FINAL         $env(SPF_DIR)/${DESIGN_NAME}.final.dspf
set SDF_FINAL          $env(SDF_DIR)/${DESIGN_NAME}.final.sdf

set RPT_LIBRARY        $env(RPT_DIR)/${DESIGN_NAME}_chk_lib.rpt
set RPT_CHK_TIMING     $env(RPT_DIR)/${DESIGN_NAME}_chk_timing.rpt
set RPT_CHK_NETLST_GEN $env(RPT_DIR)/${DESIGN_NAME}_chk_netlist.generic.rpt
set RPT_CHK_NETLST_MAP $env(RPT_DIR)/${DESIGN_NAME}_chk_netlist.mapped.rpt
set RPT_CHK_NETLST_OPT $env(RPT_DIR)/${DESIGN_NAME}_chk_netlist.opti-
                                           mized.rpt
set RPT_CHK_DESIGN     $env(RPT_DIR)/${DESIGN_NAME}_chk_design.rpt
set RPT_CLOCKS         $env(RPT_DIR)/${DESIGN_NAME}_clocks.rpt
set RPT_CLOCK_TREE     $env(RPT_DIR)/${DESIGN_NAME}_clock_tree.rpt
set RPT_CLK_VIOL       $env(RPT_DIR)/${DESIGN_NAME}_clk_viol.clocked.rpt
set RPT_CLK_VIOL_GROUTD $env(RPT_DIR)/${DESIGN_NAME}_clk_viol.grouted.rpt
set RPT_CLK_VIOL_IPO_GR $env(RPT_DIR)/
                                           ${DESIGN_NAME}_clk_viol.ipo_gro
                                           uted.rpt
set RPT_CLK_VIOL_FINAL $env(RPT_DIR)/${DESIGN_NAME}_clk_viol.drouted.rpt
set RPT_TIM_MAPPED     $env(RPT_DIR)/${DESIGN_NAME}_timing.mapped.rpt
set RPT_TIM_OPTIMIZED  $env(RPT_DIR)/${DESIGN_NAME}_timing.optimized.rpt
set RPT_TIM_PLACED     $env(RPT_DIR)/${DESIGN_NAME}_timing.placed.rpt
set RPT_TIM_CLOCKED    $env(RPT_DIR)/${DESIGN_NAME}_timing.clocked.rpt
set RPT_TIM_CLKOPT     $env(RPT_DIR)/${DESIGN_NAME}_timing.clkopt.rpt
set RPT_TIM_HOLDOPT    $env(RPT_DIR)/${DESIGN_NAME}_timing.hldopt.rpt
set RPT_TIM_GROUTED    $env(RPT_DIR)/${DESIGN_NAME}_timing.grouted.rpt
set RPT_TIM_IPO        $env(RPT_DIR)/${DESIGN_NAME}_timing.ipo.rpt
set RPT_TIM_IPO_GROUTED $env(RPT_DIR)/${DESIGN_NAME}_timing.ipo_grouted.rpt
set RPT_TIM_FINAL      $env(RPT_DIR)/${DESIGN_NAME}_timing.drouted.rpt

#       Set verilog source path
set_global hdl_verilog_vpp_arg -I$SOURCE_DIR/verilog

############################################################################
                                           ####
#
#       Check that we are using the expected version of software
#
#set thisversion [lindex [get_global version] 0]
```

```
#if { $thisversion != $reqversion } {
#       issue_message -type error \
#               "Current verion: $thisversion. Expected: $reqversion"
#       exit 1
#}

############################################################################
                                             ####
#
#       Read synthesis libraries
#

proc set_design_ola_best_case {} {
        set_dcl_calculation_mode -mode best_case
        set_operating_parameter -process 1.00
        set_operating_parameter -temperature 0
        set_operating_parameter -voltage 3.3
        }

proc set_design_ola_worst_case {} {
        set_dcl_calculation_mode -mode worst_case
        set_operating_parameter -process 1.00
        set_operating_parameter -temperature 100
        set_operating_parameter -voltage 2.3
        }

#####   Read libs
if {$env(LIB_TYPE) == "ola"} {
        read_ola
        set_design_ola_worst_case
} else {
        issue_message -type error "Invalid LIB_TYPE: $env(LIB_TYPE)"
        exit 1
}
source $DONT_USE_CELLS_FILE

############################################################################
                                             ####
#
#       Define job procedures
#

proc read_lef_files {lef_files} {
        global RPT_LIBRARY
        #
        if {[llength $lef_files] > 0} {
                read_lef [lindex $lef_files 0]
                if {[llength $lef_files] > 1} {
                        foreach lef [lrange $lef_files 1 end] {
                                read_lef_update $lef
                        }
                }
                check_library $RPT_LIBRARY
        } else {
                issue_message -type error "No lef files specified!"
                exit 1
        }
        }
```

```
proc read_lut_file {lut} {
        if {[llength $lut] > 0} {
                read_layer_usages $lut
        } else {
                issue_message -warning "No layer utilization table specified"
                exit 1
        }
        }

proc build_generic {} {
        global HDL_SOURCE_FILES RPT_CHK_NETLST_GEN ADB_GENERIC HDL_GENERIC
        #
        read_verilog $HDL_SOURCE_FILES
        do_build_generic
        write_adb -all $ADB_GENERIC
        check_netlist -verbose $RPT_CHK_NETLST_GEN
        cat $RPT_CHK_NETLST_GEN
        write_verilog -hier $HDL_GENERIC
        }

proc read_design_adb adb {
        set_message_verbosity TCLNL-307 off
        read_adb $adb
        set_message_verbosity TCLNL-307 on
        check_netlist -verbose
        check_libraries_and_design_compatibility
        }

proc translate_constraints {} {
        global CONSTRAINTS_DC CONSTRAINTS_AMBIT RPT_CHK_TIMING
        #
        read_dc_script -ambit $CONSTRAINTS_AMBIT $CONSTRAINTS_DC
        check_timing > $RPT_CHK_TIMING
        cat $RPT_CHK_TIMING
        }

proc read_constraints {} {
        global CONSTRAINTS_AMBIT RPT_CHK_TIMING
        #
        source $CONSTRAINTS_AMBIT
        check_timing > $RPT_CHK_TIMING
        cat $RPT_CHK_TIMING
        }

proc map_design {} {
        global RPT_CHK_NETLST_MAP RPT_TIM_MAPPED ADB_MAPPED HDL_MAPPED
        #
        do_optimize -pks -effort low
        write_adb -all $ADB_MAPPED
        check_netlist -verbose $RPT_CHK_NETLST_MAP
        cat $RPT_CHK_NETLST_MAP
        report_timing > $RPT_TIM_MAPPED
        cat $RPT_TIM_MAPPED
        write_verilog -hier $HDL_MAPPED
        }

proc read_design_def {def} {
        read_def $def
        report_floorplan_parameters
        }

proc map_place_and_optimize_design {} {
        global ADB_OPTIMIZED HDL_OPTIMIZED DEF_OPTIMIZED
        global RPT_CHK_NETLST_OPT RPT_TIM_OPTIMIZED
        #
        do_optimize -pks
        write_adb -all $ADB_OPTIMIZED
        write_verilog -hier $HDL_OPTIMIZED
        write_def -placement all -netlist $DEF_OPTIMIZED
        check_netlist -verbose $RPT_CHK_NETLST_OPT
        cat $RPT_CHK_NETLST_OPT
        report_timing > $RPT_TIM_OPTIMIZED
        cat $RPT_TIM_OPTIMIZED
        }

proc place_design {} {
        global RPT_CHK_DESIGN DEF_PLACED RPT_TIM_PLACED ADB_PLACED HDL_PLACED
        #
        check_design $RPT_CHK_DESIGN
        do_place -def_out_name  $DEF_PLACED
        write_adb -all $ADB_PLACED
        write_verilog -hier $HDL_PLACED
        report_timing > $RPT_TIM_PLACED
        cat $RPT_TIM_PLACED
        }

proc optimize_design {} {
        global RPT_TIM_OPTIMIZED ADB_OPTIMIZED HDL_OPTIMIZED DEF_OPTIMIZED
        #
        report_timing
        do_xform_optimize_slack -pks
        write_adb -all $ADB_OPTIMIZED
        write_verilog -hier $HDL_OPTIMIZED
        write_def -placement all -netlist $DEF_OPTIMIZED
        report_timing > $RPT_TIM_OPTIMIZED
        cat $RPT_TIM_OPTIMIZED
        }

proc generate_clock_trees {} {
        global CONSTRAINTS_CTPKS RPT_CLOCKS CTPKS_USABLE_CELLS RPT_CLOCK_TREE
        global RPT_CLK_VIOL RPT_TIM_CLOCKED DONT_USE_CELLS_FILE ADB_CLOCKED
        global HDL_CLOCKED DEF_CLOCKED
        #
        source $CONSTRAINTS_CTPKS
        report_clocks > $RPT_CLOCKS
        cat $RPT_CLOCKS
        #       Set the ctpks usable cells
        if {[llength $CTPKS_USABLE_CELLS] > 0} {
                set buffers [get_names [get_clock_tree_objects -buffer]]
                foreach cell_id [find -cell $buffers] {
                        set_attribute $cell_id ct_dont_utilize true
                }
                set inverters [get_names [get_clock_tree_objects -inverter]]
                foreach cell_id [find -cell $inverters] {
                        set_attribute $cell_id ct_dont_utilize true
                }
                set_tech_info -cell [find -cell $CTPKS_USABLE_CELLS] \
                              -dont_utilize false
                foreach cell_id [find -cell $CTPKS_USABLE_CELLS] {
                        set_attribute $cell_id ct_dont_utilize false
                }
        }
        set ctpks_buffers [get_names [get_clock_tree_objects -buffer]]
        set ctpks_inverters [get_names [get_clock_tree_objects -inverter]]
        issue_message "CTPKS usable buffers:   $ctpks_buffers"
        issue_message "CTPKS usable inverters: $ctpks_inverters"
        #
        set_clock_propagation propagated
        report_clock_tree > $RPT_CLOCK_TREE
        #       Build clock trees
        do_build_clock_tree -pin {clk}
        report_clock_tree >> $RPT_CLOCK_TREE
        report_clock_tree_violations > $RPT_CLK_VIOL
        cat $RPT_CLK_VIOL
        #       Build physical trees
        do_build_physical_tree reset
        #
        report_timing > $RPT_TIM_CLOCKED
        cat $RPT_TIM_CLOCKED
        source $DONT_USE_CELLS_FILE
        write_adb -all $ADB_CLOCKED
        write_verilog -hier $HDL_CLOCKED
        write_def -placement all -netlist $DEF_CLOCKED
        }

proc optimize_post_clock_insertion_design {} {
        global RPT_TIM_CLKOPT ADB_CLKOPT HDL_CLKOPT DEF_CLKOPT
        #
        set_clock_propagation propagated
        set_floorplan_parameters -max_row_utilization 95
        do_xform_ipo
        do_place -eco
        write_adb -all $ADB_CLKOPT
        write_verilog -hier $HDL_CLKOPT
        write_def -placement all -netlist $DEF_CLKOPT
        report_timing > $RPT_TIM_CLKOPT
        cat $RPT_TIM_CLKOPT
        }

proc optimize_design_hold_times {} {
        global ADB_HOLDOPT HDL_HOLDOPT DEF_HOLDOPT RPT_TIM_HOLDOPT
        #
       set_design_ola_best_case
       set_clock_propagation propagated
       set_clock_uncertainty  -pin clk 0.0
       report_timing -early -summary > $RPT_TIM_prehold
       if { [ get_module_worst_slack -early ] < 0 } {
       do_xform_fix_hold -incremental -dont_reclaim_area
       do_place -eco
       }
       set_design_ola_worst_case
       write_adb -all $ADB_HOLDOPT
       write_verilog -hier $HDL_HOLDOPT
```

```
        write_def -placement all -netlist $DEF_HOLDOPT
        report_timing > $RPT_TIM_HOLDOPT
        report_timing -early >> $RPT_TIM_HOLDOPT
        cat $RPT_TIM_HOLDOPT
        }

proc global_route_design {} {
        global WDB DEF_GROUTED RPT_TIM_GROUTED RPT_CLK_VIOL_GROUTD ADB_GROUTED
         global CONSTRAINTS_CTPKS
         set_clock_propagation propagated
         report_clocks
         report_timing
         do_route \
                 -timing_driven \
                 -route_top_layer_limit  5 \
                 -route_select_net_first true \
                 -route_select_net       "@clock" \
                 -output_db_name         $WDB \
                 -def_out_name           $DEF_GROUTED
         report_timing > $RPT_TIM_GROUTED
         cat $RPT_TIM_GROUTED
         source $CONSTRAINTS_CTPKS
         report_clock_tree_violations > $RPT_CLK_VIOL_GROUTD
         cat $RPT_CLK_VIOL_GROUTD
         write_adb -all $ADB_GROUTED
         }

proc optimize_design_in_place {} {
         global RPT_TIM_IPO ADB_IPO HDL_IPO DEF_IPO
         #
         set_clock_propagation propagated
         do_xform_ipo
         do_place -eco
         write_adb -all $ADB_IPO
         write_verilog -hier $HDL_IPO
         write_def -placement all -netlist $DEF_IPO
         report_timing > $RPT_TIM_IPO
         cat $RPT_TIM_IPO
         }

proc global_route_ipo_design {} {
         global WDB_IPO DEF_IPO_GROUTED RPT_TIM_IPO_GROUTED
         global RPT_CLK_VIOL_IPO_GR ADB_IPO_GROUTED
         #
         set_clock_propagation propagated
         report_clocks
         report_timing
         do_route \
                 -timing_driven \
                 -route_top_layer_limit  5 \
                 -route_select_net_first true \
                 -route_select_net       "@clock" \
                 -output_db_name         $WDB_IPO \
                 -def_out_name           $DEF_IPO_GROUTED
         report_timing > $RPT_TIM_IPO_GROUTED
         cat $RPT_TIM_IPO_GROUTED
         source $CONSTRAINTS_CTPKS
         report_clock_tree_violations > $RPT_CLK_VIOL_IPO_GR
         cat $RPT_CLK_VIOL_IPO_GR
         write_adb -all $ADB_IPO_GROUTED
         }



 proc final_sta {} {
         global RPT_TIM_FINAL CONSTRAINTS_CTPKS RPT_CLK_VIOL_FINAL
         report_timing > $RPT_TIM_FINAL
         cat $RPT_TIM_FINAL
         source $CONSTRAINTS_CTPKS
         report_clock_tree_violations > $RPT_CLK_VIOL_FINAL
         cat $RPT_CLK_VIOL_FINAL
         }
 return
```

Appendix D PKS Control Scripts

rtl2generic.tcl

```
source ${env(SCRIPT_DIR)}/tcl/setup.tcl
build_generic
exit
```

cold_start.tcl

```
source ${env(SCRIPT_DIR)}/tcl/setup.tcl
read_lef_files $LEF_REF_FILES
read_lut_file $LUT_REF
read_design_adb $ADB_GENERIC
read_constraints
set_global pre_placement_max_sinks_per_net 10
map_design
```

general_flow.tcl

```
source ${env(SCRIPT_DIR)}/tcl/setup.tcl
read_lef_files $LEF_DESIGN_FILES
read_lut_file $LUT_DESIGN
read_design_adb $ADB_GENERIC
read_constraints
read_design_def $DEF_FLOORPLAN
map_place_and_optimize_design
generate_clock_trees
optimize_post_clock_insertion_design
optimize_design_hold_times
global_route_design
if { [ get_module_worst_slack ] < 0 } {
        optimize_design_in_place
        global_route_ipo_design
}
exit
```

final_sta.tcl

```
source ${env(SCRIPT_DIR)}/tcl/setup.tcl
read_lef_files $LEF_DESIGN_FILES
read_lut_file $LUT_DESIGN
read_design_adb $ADB_GROUTED
read_spf -verbose $DSPF_FINAL
final_sta
exit
```

gen_final_sdf.tcl

```
source ${env(SCRIPT_DIR)}/tcl/setup.tcl
read_lef_files $LEF_DESIGN_FILES
read_lut_file $LUT_DESIGN
read_design_adb $ADB_GROUTED
#write_sdf $SDF_FINAL
set_global hdl_verilog_out_unconnected_style full
write_verilog -hier $HDL_FINAL_SIM
exit
```

Appendix E run_se_job

```
#!/bin/csh

if ($#argv != 2) then
        echo "Usage:"
        echo "run_pks_job <run_name> <mac_file>"
        exit 1
else
        echo "run_name: $1:t"
        echo "mac_file: $2:t"
endif

#####   Set environment variables   #####
setenv DESIGN_NAME      dft_chip
setenv ROOT_DIR         /ambit/sipg/ibm/Sige/digital/flowkit/dft_chip
setenv LIB_DIR          /ambit/sipg/ibm/Sige/digital/digitalkit

setenv RUN_NAME $1:t
setenv RUN_DIR          $ROOT_DIR/rundata/$RUN_NAME
setenv SCRIPT_DIR       $ROOT_DIR/scripts

#####   Create run directories if necessary   #####
setenv SE_DIR   $RUN_DIR/se; if (! -e $SE_DIR)    mkdir $SE_DIR
setenv DBS_DIR  $SE_DIR/dbs; if (! -e $DBS_DIR)   mkdir $DBS_DIR
setenv RPT_DIR  $SE_DIR/rpt; if (! -e $RPT_DIR)   mkdir $RPT_DIR
setenv DEF_DIR  $SE_DIR/def; if (! -e $DEF_DIR)   mkdir $DEF_DIR
setenv WDB_DIR  $SE_DIR/wdb; if (! -e $WDB_DIR)   mkdir $WDB_DIR
setenv SPF_DIR  $RUN_DIR/spf; if (! -e $SPF_DIR)   mkdir $SPF_DIR

#####   Set Library environmental variable      #####
if (! $?LD_LIBRARY_PATH) then
        setenv LD_LIBRARY_PATH  $OPENWINHOME/lib
endif

#####   Create links to relocate some SE files  #####
set liblnk = $SE_DIR/reflib
set verlnk = $SE_DIR/mapped.v
set deflnk = $SE_DIR/def/fp.def
set spflnk = $SE_DIR/final.dspf
set grlnk  = $SE_DIR/wdb/grouted.wdb
set leflnk = $SE_DIR/updated.lef
set lutlnk = $SE_DIR/updated.lut

cd $SE_DIR
test -L $liblnk; if ($status) ln -s $LIB_DIR $liblnk
test -L $verlnk; if ($status) ln -s ../verilog/${DESIGN_NAME}.mapped.v
                                      $verlnk
test -L $deflnk; if ($status) ln -s ../../def/${DESIGN_NAME}.floorp-
                                      lan.def $deflnk
test -L $spflnk; if ($status) ln -s ../spf/${DESIGN_NAME}.final.dspf
                                      $spflnk
test -L $grlnk;  if ($status) ln -s ../../wdb/
                                      ${DESIGN_NAME}.grouted.wdb
                                      $grlnk
test -L $leflnk; if ($status) ln -s ../../../techlib/
                                      ${DESIGN_NAME}_tech.lef
                                      $leflnk
```

```
test -L $lutlnk; if ($status) ln -s ../../../techlib/
                                      ${DESIGN_NAME}_lut   $lutlnk
#####   Define the mac and jnl files
set mac_file = $SCRIPT_DIR/mac/$2:t
if (! -f $mac_file) then
        echo "run_se_cold_job: ERROR: $mac_file not found\!"
        exit 1
endif
set fhead = ‘echo $mac_file:t | sed -e ’s/.mac$//’‘
set jnl_file = $SE_DIR/${fhead}.jnl

#####   Set license variable and put dsm bin at the head of path
                                      #####
setenv LM_LICENSE_FILE  11536@ra:5280@mammoth
set dsm_bin = /net/cds11410/usr6/alanza/ibm/sige/SE_5.3.125/
                                      DSMSE53ISR_sun4v/tools/dsm/bin
#set dsm_bin = /net/hotfix/cds/DSMSE53ISR_sun4v/tools/dsm/bin
#set dsm_bin = /net/dsm_cm1/usr2/dsm/POMPEII/sun4v/tools/dsm/bin
set path = ($dsm_bin ‘echo $path | /usr/bin/sed -e {s@$dsm_bin@@}‘)

#####   Run SE   #####
(cd $SCRIPT_DIR/se; cp se.ini *.cfg $SE_DIR)
seultra -gd=ansi -j=$jnl_file -m=800 "execute $mac_file ;"
```

Appendix F SE Cold Start Scripts

The following is the main SE macro that sequentially calls each of the listed macros

```
EXECUTE ../../../scripts/mac/load_libraries.mac ;
EXECUTE ../../../scripts/mac/read_ver_cold.mac ;
EXECUTE ../../../scripts/mac/init_fp_cold.mac ;
EXECUTE ../../../scripts/mac/place_ios_cold.mac ;
EXECUTE ../../../scripts/mac/gen_pwr_cold.mac ;
EXECUTE ../../../scripts/mac/add_nwties_cold.mac ;
EXECUTE ../../../scripts/mac/place_cells_cold.mac ;
EXECUTE ../../../scripts/mac/add_filler_cold.mac ;
EXECUTE ../../../scripts/mac/route_cold.mac ;
EXECUTE ../../../scripts/mac/gen_dspf_cold.mac ;
EXECUTE ../../../scripts/mac/update_tech_cold.mac ;
FQUIT ;
```

Each of the named macro's above is shown in its entirety below.

```
############################################################################
#
#       Input LEF library files
#

FINPUT LEF FILENAME "reflib/lef/tech7hp.6mt.lef" ;
INPUT  LEF FILENAME "reflib/lef/abstract7hp.cds.lef" ;
SAVE DESIGN     lib ;

############################################################################
#
#       Read mapped verilog file
#

FLOAD DESIGN    lib ;
SET VARIABLE    INPUT.VERILOG.POWER.NET         "VDD!";
SET VARIABLE    INPUT.VERILOG.GROUND.NET        "GND!";
SET VARIABLE    INPUT.VERILOG.LOGIC.1.NET       "VDD!";
SET VARIABLE    INPUT.VERILOG.LOGIC.0.NET       "GND!";
SET VARIABLE    INPUT.VERILOG.SPECIAL.NETS      "VDD! GND!";
INPUT VERILOG   FILE    "reflib/verilog/ibm.stub.v"
                LIB     "lib_vbin"
                REFLIB "lib_vbin" ;
#SAVE  DESIGN   v_lib ;
#FLOAD DESIGN   v_lib ;
INPUT VERILOG   FILE    "mapped.v"
                LIB     "design_vbin"
                REFLIB  "lib_vbin"
                DESIGN  "design_vbin.dft_chip:hdl" ;
REPORT SUMMARY  FILENAME "rpt/design.summary.rpt" ;
SAVE DESIGN     design ;
```

```
############################################################################
                                             ####
#
#       Initialize Floorplan
#

FLOAD DESIGN    design;
SET VARIABLE    USERLEVEL                     EXPERT;
SET VARIABLE    PLAN.REPORT.STAT              "   ";
SET VARIABLE    UPDATECOREROW.BLOCKHALO.GLOBAL  2000;
FINITIALIZE FLOORPLAN;
FINITIALIZE FLOORPLAN
                ASPECTRATIO             1
                ROWUTILIZATION          0.80
                XIOTOCOREDIST           6000
                YIOTOCOREDIST           6000
                ABUTPAIRSOFROWS
                FLIPALTERNATEROWS
                ROWSPACING              0
                BLOCKHALO               2000 ;
REPORT SUMMARY  FILENAME "rpt/init_fp.summary.rpt" ;
SAVE DESIGN     init_fp ;


############################################################################
#
#       Place ios
#

FLOAD DESIGN    init_fp ;
IOPLACE         STYLE EVEN AUTOMATIC TOPBOTTOMLAYER M4 RIGHTLEFTLAYER M3 ;
SET VARIABLE    DRAW.PIN.AT     HERE;
REFRESH ;
IOPLACE         FILENAME ./ioplace.ioc WRITE ;
SAVE DESIGN     ios ;



############################################################################
                                             ####
#
#       Generate Power
#

FLOAD DESIGN    ios ;
SET VARIABLE    SROUTE.VIA.MERGEONSAMELAYERS    TRUE ;
SET VARIABLE    SROUTE.STACKVIASATCROSSOVER     TRUE ;
SET VARIABLE    DRAW.SWIRE.AT                   ON ;
SET VARIABLE    DRAW.CHANNEL.AT                 ON ;
BUILD CHANNEL ;
CONSTRUCT RING  NET "GND!" NET "VDD!"
                LAYER M1 CORERINGWIDTH 1500 SPACING CENTER BLOCKRINGWIDTH 0
                LAYER M2 CORERINGWIDTH 1500 SPACING CENTER BLOCKRINGWIDTH 0 ;
ADD STRIPE     NET "GND!" DIRECTION Horizontal LAYER MT WIDTH 120 LOWERMARGIN
                                             1320 STEP 1920 ALL ;
ADD STRIPE     NET "VDD!" DIRECTION Horizontal LAYER MT WIDTH 120 LOWERMARGIN
                                             2280 STEP 1920 ALL ;
ADD STRIPE     NET "GND!" DIRECTION Vertical   LAYER M4 WIDTH 120 LOWERMARGIN
```

```
                                              1320 STEP 1920 ALL ;
ADD STRIPE     NET "VDD!" DIRECTION Vertical   LAYER M4 WIDTH 120 LOWERMARGIN
                                              2280 STEP 1920 ALL ;
ADD STRIPE     NET "GND!" DIRECTION Horizontal LAYER M3 WIDTH 120 LOWERMARGIN
                                              1320 STEP 1920 ALL ;
ADD STRIPE     NET "VDD!" DIRECTION Horizontal LAYER M3 WIDTH 120 LOWERMARGIN
                                              2280 STEP 1920 ALL ;
ADD STRIPE     NET "GND!" DIRECTION Vertical   LAYER M2 WIDTH 120 LOWERMARGIN
                                              1320 STEP 1920 ALL ;
ADD STRIPE     NET "VDD!" DIRECTION Vertical   LAYER M2 WIDTH 120 LOWERMARGIN
                                              2280 STEP 1920 ALL ;
DISPOSE CHANNEL ;
SET VARIABLE    DRAW.SWIRE.GEOM "On";
SET VARIABLE    DRAW.CHANNEL.AT OFF ;
REFRESH ;
SET VARIABLE    OUTPUT.DEF.SPNET.WILDCARD       TRUE ;
SET VARIABLE    SROUTE.LPR.STACKVIASATCROSSOVER TRUE;
SET VARIABLE    SROUTE.LPR.VIASATCROSSOVER      TRUE ;
SET VARIABLE    SROUTE.FOLLOWPINS.FILL          TRUE ;
SET VARIABLE    SROUTE.VIA.MERGEONSAMELAYERS    TRUE ;
SET VARIABLE    SROUTE.STACKVIASATCROSSOVER     TRUE ;
SET VARIABLE    SROUTE.FOLLOWPINS.MAXROWS       500 ;
CONNECT RING    NET "GND!" NET "VDD!" STRIPE FOLLOWPIN ;
OUTPUT DEF      FILENAME def/pwr.def EXTPIN SPECIALNETS VIAS ;
SAVE DESIGN     pwr ;


############################################################################
#
#       Add N-well Tie Cells
#

system ’reflib/support_files/add_fillercells.pl -def def/pwr.def -cell_lef
                                              reflib/lef/abstract7hp.ibm.lef
                                              -tech_lef reflib/lef/
                                              tech7hp.6mt.lef -out def/fp.def
                                              -cell NWSX -prefix fil -inc
                                              72.00 -FS_offset 36.00 -N_offset
                                              36.00 -vss GND! -vdd VDD! ’ ;
FLOAD DESIGN    design ;
FINPUT DEF      FILENAME        def/fp.def
                REPORTFILE      rpt/fp.def_in.rpt
                NOALLOWEEQSUBSTITUTION ;
REPORT SUMMARY  FILENAME        rpt/fp.summary.rpt ;
SAVE DESIGN     fp ;



############################################################################
#
#       Place Cells
#
#

FLOAD DESIGN    fp ;
SET VARIABLE    QPLACE.PLACE.PIN "CONCURRENT" ;
SET VARIABLE    QPLACE.FREE.TRACK.PCT.1 20 ;
SET VARIABLE    QPLACE.FREE.TRACK.PCT.2 88 ;
SET VARIABLE    QPLACE.FREE.TRACK.PCT.3 88 ;
SET VARIABLE    QPLACE.FREE.TRACK.PCT.4 88 ;
SET VARIABLE    QPLACE.FREE.TRACK.PCT.5 88 ;
SET VARIABLE    QPLACE.FREE.TRACK.PCT.6 0 ;
QPLACE          NOCONFIG ;
REPORT SUMMARY  FILENAME rpt/qp.summary.rpt ;
SAVE DESIGN     qp ;


############################################################################
#
#       Add Filler Cells
#

FLOAD DESIGN    qp ;
EXECUTE         ../../../scripts/mac/add_filler_cells.mac ;
SET VARIABLE    OUTPUT.DEF.SPNET.WILDCARD       TRUE ;
OUTPUT DEF      FILENAME def/placed_cold.def ;
SAVE DESIGN     placed_cold ;



############################################################################
#
#       Route the cold start design
#

SYSTEM          ’cp ../../../scripts/se/wr_cold.cfg .’ ;
SYSTEM          ’rm -f drouted_cold.log*’
SYSTEM          ’wroute -l drouted_cold.log -q wr_cold.cfg’ ;
FLOAD DESIGN    lib ;
INPUT DEF       FILENAME        def/drouted_cold.def
                REPORTFILE      rpt/drouted.def_in.rpt
                NOALLOWEEQSUBSTITUTION ;
SAVE DESIGN     drouted_cold ;

############################################################################
#
#       Extract cold start the parasitics
#

FLOAD DESIGN    drouted_cold ;
EXECUTE         ../../../scripts/mac/setup_hyper.mac ;
REPORT RC       FILENAME drouted_cold.dspf ;

############################################################################
#
#       Generate Updated Lut and LEF Capacitances
#

SYSTEM ’reflib/support_files/lutgen.pl ./drouted_cold.log updated.lut’ ;
SYSTEM ’reflib/support_files/lef_update.pl HyperExtract.log reflib/lef/
                                              tech7hp.6mt.lef updated.lef’ ;
```

Appendix G SE General Flow Scripts

```
EXECUTE ../../../scripts/mac/load_libraries.mac ;
EXECUTE ../../../scripts/mac/droute.mac ;
EXECUTE ../../../scripts/mac/add_filler.mac ;
EXECUTE ../../../scripts/mac/gen_dspf.mac ;
EXECUTE ../../../scripts/mac/verify.mac ;


######################################################################
#
#       Input LEF library files
#

FINPUT LEF FILENAME "reflib/lef/tech7hp.6mt.lef" ;
INPUT  LEF FILENAME "reflib/lef/abstract7hp.cds.lef" ;
SAVE DESIGN     lib ;

######################################################################
#
#       Route the design
#

SYSTEM          'cp ../../../scripts/se/wr_droute.cfg .' ;
SYSTEM          'rm -f wr_droute.log* '
SYSTEM          'wroute -l wr_droute.log -q wr_droute.cfg' ;
FLOAD DESIGN    lib ;
INPUT DEF       FILENAME        def/drouted.def
                REPORTFILE      rpt/drouted.def_in.rpt
                NOALLOWEEQSUBSTITUTION ;
SAVE DESIGN     drouted ;



#####################################################################
#
#       Add Filler Cells
#

FLOAD DESIGN    drouted ;
EXECUTE         ../../../scripts/mac/add_filler_cells.mac ;
SET VARIABLE    OUTPUT.DEF.SPNET.WILDCARD       TRUE ;
OUTPUT DEF      FILENAME        def/final.def ;
OUTPUT GDSII    MAPFILE         reflib/support_files/gds2.map
                FILE            final.gds
                UNITS           Thousands  ;
SAVE DESIGN     final ;
```

```
######################################################################
#
#       Add Filler Cells
#

SROUTE ADDCELL  MODEL GAUNUSED096 PREFIX filler NORTH FNORTH SOUTH
                                      FSOUTH
                SPIN GND! NET GND! SPIN VDD! NET VDD!
                AREA ( -10000000 -10000000) ( 10000000 10000000 ) ;
SROUTE ADDCELL  MODEL GAUNUSED048 PREFIX filler NORTH FNORTH SOUTH
                                      FSOUTH
                SPIN GND! NET GND! SPIN VDD! NET VDD!
                AREA ( -10000000 -10000000) ( 10000000 10000000 ) ;
SROUTE ADDCELL  MODEL GAUNUSED024 PREFIX filler NORTH FNORTH SOUTH
                                      FSOUTH
                SPIN GND! NET GND! SPIN VDD! NET VDD!
                AREA ( -10000000 -10000000) ( 10000000 10000000 ) ;
SROUTE ADDCELL  MODEL GAUNUSED012 PREFIX filler NORTH FNORTH SOUTH
                                      FSOUTH
                SPIN GND! NET GND! SPIN VDD! NET VDD!
                AREA ( -10000000 -10000000) ( 10000000 10000000 ) ;
SROUTE ADDCELL  MODEL GAUNUSED006 PREFIX filler NORTH FNORTH SOUTH
                                      FSOUTH
                SPIN GND! NET GND! SPIN VDD! NET VDD!
                AREA ( -10000000 -10000000) ( 10000000 10000000 ) ;
SROUTE ADDCELL  MODEL GAUNUSED003 PREFIX filler NORTH FNORTH SOUTH
                                      FSOUTH
                SPIN GND! NET GND! SPIN VDD! NET VDD!
                AREA ( -10000000 -10000000) ( 10000000 10000000 ) ;
SROUTE ADDCELL  MODEL FILL2 PREFIX filler NORTH FNORTH SOUTH FSOUTH
                SPIN GND! NET GND! SPIN VDD! NET VDD!
                AREA ( -10000000 -10000000 ) ( 10000000 10000000 ) ;
SROUTE ADDCELL  MODEL FILL1 PREFIX filler NORTH FNORTH SOUTH FSOUTH
                SPIN GND! NET GND! SPIN VDD! NET VDD!
                AREA ( -10000000 -10000000 ) ( 10000000 10000000 ) ;


######################################################################
#
#       Extract final routed parasitics
#

FLOAD DESIGN    final ;
EXECUTE         ../../../scripts/mac/setup_hyper.mac ;
REPORT RC       FILENAME final.dspf ;
######################################################################
                                      ##########
#
#       Setup hyperextract to extract the parasitics
#

SET VARIABLE HYPEREXTRACT.PINS.CAPACITANCE      NONE ;
SET VARIABLE HYPEREXTRACT.TOPPINS.CAPACITANCE   NONE ;
SET VARIABLE HYPEREXTRACT.WRITE.DSPF.INSTANCE   TRUE ;
SET VARIABLE TIMING.REPORTRC.FORMAT             DSPF ;
SET VARIABLE HYPEREXTRACT.SHRINK.FACTOR         1.00 ;
SET VARIABLE HYPEREXTRACT.WIRELOAD.MEDIAN       0.0 ;
SET VARIABLE HYPEREXTRACT.WIRELOAD.MIN.FANOUT   0 ;
SET VARIABLE TIMING.CAPACITANCE.MODEL           HYPEREXTRACT ;
SET VARIABLE HYPEREXTRACT.RULES.FILE            "reflib/hypext/
                                      hyper7hp_wc.rules" ;
SET VARIABLE HYPEREXTRACT.WIRELOAD.FILE         ’’;
SET VARIABLE HYPEREXTRACT.SYNOPSYSLOAD.FILE



################################
# Verifying
################################
verify geometry ;
verify connectivity ;
report info filename verify_rpt all;
```

Appendix H Sige.dont_use_cells.tcl

```
# DONT UTILIZE BASED ON THE DIFFERENCE OF CELLTYPE BETWEEN DOT LIB AND
                                       OLA
# IN OLA LIBRARY THESE FOLLOWING CELLS ARE RECOGNIZED AS LATCH
# IN DOT LIB THEY ARE RECOGNIZED AS FF
#---------------------------------------------------------------------
                                       ------
#set_cell_property dont_utilize true [find -cell  DFF0003_E]
#set_cell_property dont_utilize true [find -cell  DFF0003_H]
#set_cell_property dont_utilize true [find -cell  DFF0003_K]
#set_cell_property dont_utilize true [find -cell  DFF1013_E]
#set_cell_property dont_utilize true [find -cell  DFF1013_H]
#set_cell_property dont_utilize true [find -cell  DFF1013_K]
#set_cell_property dont_utilize true [find -cell  DFF4043_E]
#set_cell_property dont_utilize true [find -cell  DFF4043_H]
#set_cell_property dont_utilize true [find -cell  DFF4043_K]


# DONT UTILIZE BECAUSE THESE CELLS ARE DECLARED TO HAVE NEGATIVE
                                       RESISTANCE BY LIBCHECK UTILITY
# A CELL IS DECLARED TO HAVE NEGATIVE RESISTANCE IF THERE IS DECREASING
                                       DELAY IN INCREASING LOAD
#---------------------------------------------------------------------
                                       ------
set_cell_property dont_utilize true [find -cell  BUFFER_N]
set_cell_property dont_utilize true [find -cell  BUFFER_O]
set_cell_property dont_utilize true [find -cell  CLKGATE_B]
set_cell_property dont_utilize true [find -cell  CLKGATE_C]
set_cell_property dont_utilize true [find -cell  CLKGATE_D]
set_cell_property dont_utilize true [find -cell  CLKGATE_E]
set_cell_property dont_utilize true [find -cell  CLKGATE_F]
set_cell_property dont_utilize true [find -cell  CLKGATE_I]
set_cell_property dont_utilize true [find -cell  CLKGATE_K]
set_cell_property dont_utilize true [find -cell  CLKGATE_M]
set_cell_property dont_utilize true [find -cell  CLKGATE_O]
set_cell_property dont_utilize true [find -cell  CLKGATE_Q]
set_cell_property dont_utilize true [find -cell  CLKG_A]
set_cell_property dont_utilize true [find -cell  CLKI_M]
set_cell_property dont_utilize true [find -cell  CLKI_O]
set_cell_property dont_utilize true [find -cell  CLKI_Q]
set_cell_property dont_utilize true [find -cell  CLK_M]
set_cell_property dont_utilize true [find -cell  CLK_Q]
#set_cell_property dont_utilize true [find -cell  DFF6063_K]
set_cell_property dont_utilize true [find -cell  MUX21BAL_J]
set_cell_property dont_utilize true [find -cell  MUX21BAL_L]
set_cell_property dont_utilize true [find -cell  NOR3_B]
set_cell_property dont_utilize true [find -cell  NOR3_C]
set_cell_property dont_utilize true [find -cell  TTMUX_M]

# DONT UTILIZE BASED ON IBM's DONT-USE LIST
#---------------------------------------------------------------------
                                       ------
set_cell_property dont_utilize true [find -cell  CG_AND*]        ;#
                                       Clock gating usage only
set_cell_property dont_utilize true [find -cell  CG_OR*]         ;#
                                       Clock gating usage only
set_cell_property dont_utilize true [find -cell  CLK*]           ;#
```

```
                                       Standard clock tree usage only
set_cell_property dont_utilize true [find -cell  CLKG*]         ;#
                                       Structure Clock Buffer clock
                                       tree usage only
set_cell_property dont_utilize true [find -cell  CLKGATE*]      ;#
                                       Clock gating usage only
set_cell_property dont_utilize true [find -cell  CLKI*]         ;#
                                       Standard clock tree usage only
set_cell_property dont_utilize true [find -cell  DECAP*]        ;#
                                       Decoupling capacitor (PD only)
set_cell_property dont_utilize true [find -cell  DELAY*]        ;#
                                       Must be instantiated
set_cell_property dont_utilize true [find -cell  DELAY4*]       ;#
                                       Must be instantiated
set_cell_property dont_utilize true [find -cell  DELAY6*]       ;#
                                       Must be instantiated
set_cell_property dont_utilize true [find -cell  FGTIE*]        ;#
                                       Floating gate contact
set_cell_property dont_utilize true [find -cell  FILL1*]       ;# 1
                                       cell filler cell (post-fill)
set_cell_property dont_utilize true [find -cell  FILL2*]       ;# 2
                                       cell filler cell (post-fill)
set_cell_property dont_utilize true [find -cell  GAUNUSED003*]  ;# 3
                                       cell filler cell (post-fill)
set_cell_property dont_utilize true [find -cell  GAUNUSED006*]  ;# 6
                                       cell filler cell (post-fill)
set_cell_property dont_utilize true [find -cell  GAUNUSED012*]  ;# 12
                                       cell filler cell (post-fill)
set_cell_property dont_utilize true [find -cell  GAUNUSED024*]  ;# 24
                                       cell filler cell (post-fill)
set_cell_property dont_utilize true [find -cell  GAUNUSED048*]  ;# 48
                                       cell filler cell (post-fill)
set_cell_property dont_utilize true [find -cell  GAUNUSED096*]  ;# 96
                                       cell filler cell (post-fill)
set_cell_property dont_utilize true [find -cell  INVERTBAL*]    ;#
                                       Special balanced function
set_cell_property dont_utilize true [find -cell  MCMUX*]        ;#
                                       Datapath/bitstack use only
set_cell_property dont_utilize true [find -cell  MUX21BAL*]     ;#
                                       Special balanced function
set_cell_property dont_utilize true [find -cell  NAND2BAL*]     ;#
                                       Special balanced function
set_cell_property dont_utilize true [find -cell  NWSX*]        ;# N-
                                       well/Substrate contact cell
                                       (pre-fill)
set_cell_property dont_utilize true [find -cell  TERM*]         ;#
                                       Must be instantiated
set_cell_property dont_utilize true [find -cell  TTMUX*]        ;#
                                       Datapath/bitstack use only



# DONT UTILIZE BASED ON THE DIFFERENCE BETWEEN OLA AND LEF CELLS
# CELLS IN THE OLA LIBRARY BUT NOT IN THE DOT LIB
#-------------------------------------------------------------------
                                       ------
set_cell_property dont_utilize true [find -cell  CLKCHP_M]
set_cell_property dont_utilize true [find -cell  CLKCHP_Q]
set_cell_property dont_utilize true [find -cell  CLKCHP_U]
set_cell_property dont_utilize true [find -cell  CLKSPC2R_D]
set_cell_property dont_utilize true [find -cell  CLKSPC2R_H]
set_cell_property dont_utilize true [find -cell  CLKSPC2R_K]
set_cell_property dont_utilize true [find -cell  CLKSPC_D]
set_cell_property dont_utilize true [find -cell  CLKSPC_H]
set_cell_property dont_utilize true [find -cell  CLKSPC_K]
set_cell_property dont_utilize true [find -cell  CLKSPL2R_D]
set_cell_property dont_utilize true [find -cell  CLKSPL2R_H]
set_cell_property dont_utilize true [find -cell  CLKSPL2R_K]
set_cell_property dont_utilize true [find -cell  CLKSPL_D]
set_cell_property dont_utilize true [find -cell  CLKSPL_H]
set_cell_property dont_utilize true [find -cell  CLKSPL_K]
set_cell_property dont_utilize true [find -cell  DEC24_DP_J]
set_cell_property dont_utilize true [find -cell  DEC38_DP_J]
set_cell_property dont_utilize true [find -cell  DELAYMUX0_A]
set_cell_property dont_utilize true [find -cell  DELAYMUX0_H]
set_cell_property dont_utilize true [find -cell  DELAYMUXN_A]
set_cell_property dont_utilize true [find -cell  DELAYMUXN_H]
set_cell_property dont_utilize true [find -cell  D_F_LMX0001_E]
set_cell_property dont_utilize true [find -cell  D_F_LMX0001_H]
set_cell_property dont_utilize true [find -cell  D_F_LMX0001_K]
set_cell_property dont_utilize true [find -cell  D_F_LMX0001_LPC_E]
set_cell_property dont_utilize true [find -cell  D_F_LMX0001_LPC_H]
set_cell_property dont_utilize true [find -cell  D_F_LMX0001_LPC_K]
set_cell_property dont_utilize true [find -cell  D_F_LPH0001_E]
set_cell_property dont_utilize true [find -cell  D_F_LPH0001_H]
set_cell_property dont_utilize true [find -cell  D_F_LPH0001_K]
set_cell_property dont_utilize true [find -cell  D_F_LPH0001_LPC_E]
set_cell_property dont_utilize true [find -cell  D_F_LPH0001_LPC_H]
set_cell_property dont_utilize true [find -cell  D_F_LPH0001_LPC_K]
set_cell_property dont_utilize true [find -cell  D_F_LPH0002_E]
set_cell_property dont_utilize true [find -cell  D_F_LPH0002_H]
set_cell_property dont_utilize true [find -cell  D_F_LPH0002_K]
set_cell_property dont_utilize true [find -cell  D_F_LPH1001_LPC_E]
set_cell_property dont_utilize true [find -cell  D_F_LPH1001_LPC_H]
set_cell_property dont_utilize true [find -cell  D_F_LPH1001_LPC_K]
set_cell_property dont_utilize true [find -cell  D_F_LPH2021_LPC_E]
set_cell_property dont_utilize true [find -cell  D_F_LPH2021_LPC_H]
set_cell_property dont_utilize true [find -cell  D_F_LPH2021_LPC_K]
set_cell_property dont_utilize true [find -cell  D_F_LPH4001_LPC_E]
set_cell_property dont_utilize true [find -cell  D_F_LPH4001_LPC_H]
set_cell_property dont_utilize true [find -cell  D_F_LPH4001_LPC_K]
set_cell_property dont_utilize true [find -cell  D_F_LPH4041_LPC_E]
set_cell_property dont_utilize true [find -cell  D_F_LPH4041_LPC_H]
set_cell_property dont_utilize true [find -cell  D_F_LPH4041_LPC_K]
set_cell_property dont_utilize true [find -cell  D_F_LPH8081_E]
set_cell_property dont_utilize true [find -cell  D_F_LPH8081_H]
set_cell_property dont_utilize true [find -cell  D_F_LPH8081_K]
set_cell_property dont_utilize true [find -cell  D_F_MPH0001_LPC_E]
set_cell_property dont_utilize true [find -cell  D_F_MPH0001_LPC_H]
set_cell_property dont_utilize true [find -cell  D_F_MPH0001_LPC_K]
set_cell_property dont_utilize true [find -cell  D_F_MPH1001_LPC_E]
set_cell_property dont_utilize true [find -cell  D_F_MPH1001_LPC_H]
set_cell_property dont_utilize true [find -cell  D_F_MPH1001_LPC_K]
set_cell_property dont_utilize true [find -cell  D_F_MPH2021_LPC_E]
set_cell_property dont_utilize true [find -cell  D_F_MPH2021_LPC_H]
set_cell_property dont_utilize true [find -cell  D_F_MPH2021_LPC_K]
set_cell_property dont_utilize true [find -cell  D_F_MPH4001_LPC_E]
set_cell_property dont_utilize true [find -cell  D_F_MPH4001_LPC_H]
set_cell_property dont_utilize true [find -cell  D_F_MPH4001_LPC_K]
set_cell_property dont_utilize true [find -cell  D_F_MPH4041_LPC_E]
set_cell_property dont_utilize true [find -cell  D_F_MPH4041_LPC_H]
set_cell_property dont_utilize true [find -cell  D_F_MPH4041_LPC_K]
set_cell_property dont_utilize true [find -cell  D_LDE0001_E]
set_cell_property dont_utilize true [find -cell  D_LDE0001_H]
set_cell_property dont_utilize true [find -cell  D_LDE0001_K]
set_cell_property dont_utilize true [find -cell  D_LDF0001_E]
set_cell_property dont_utilize true [find -cell  D_LDF0001_H]
set_cell_property dont_utilize true [find -cell  D_LDF0001_K]
set_cell_property dont_utilize true [find -cell  D_LDM8083_E]
set_cell_property dont_utilize true [find -cell  D_LDM8083_H]
set_cell_property dont_utilize true [find -cell  D_LDM8083_K]
set_cell_property dont_utilize true [find -cell  D_LDR0001_E]
set_cell_property dont_utilize true [find -cell  D_LDR0001_H]
set_cell_property dont_utilize true [find -cell  D_LDR0001_K]
set_cell_property dont_utilize true [find -cell  F_LMX0001_E]
set_cell_property dont_utilize true [find -cell  F_LMX0001_H]
set_cell_property dont_utilize true [find -cell  F_LMX0001_K]
set_cell_property dont_utilize true [find -cell  F_LMX0001_LPC_E]
set_cell_property dont_utilize true [find -cell  F_LMX0001_LPC_H]
set_cell_property dont_utilize true [find -cell  F_LMX0001_LPC_K]
set_cell_property dont_utilize true [find -cell  F_LPH0001_E]
set_cell_property dont_utilize true [find -cell  F_LPH0001_H]
set_cell_property dont_utilize true [find -cell  F_LPH0001_K]
set_cell_property dont_utilize true [find -cell  F_LPH0001_LPC_E]
set_cell_property dont_utilize true [find -cell  F_LPH0001_LPC_H]
set_cell_property dont_utilize true [find -cell  F_LPH0001_LPC_K]
set_cell_property dont_utilize true [find -cell  F_LPH0002_E]
set_cell_property dont_utilize true [find -cell  F_LPH0002_H]
set_cell_property dont_utilize true [find -cell  F_LPH0002_K]
set_cell_property dont_utilize true [find -cell  F_LPH1001_LPC_E]
set_cell_property dont_utilize true [find -cell  F_LPH1001_LPC_H]
set_cell_property dont_utilize true [find -cell  F_LPH1001_LPC_K]
set_cell_property dont_utilize true [find -cell  F_LPH4001_LPC_E]
set_cell_property dont_utilize true [find -cell  F_LPH4001_LPC_H]
set_cell_property dont_utilize true [find -cell  F_LPH4001_LPC_K]
set_cell_property dont_utilize true [find -cell  F_LPH6001_E]
set_cell_property dont_utilize true [find -cell  F_LPH6001_H]
set_cell_property dont_utilize true [find -cell  F_LPH6001_K]
set_cell_property dont_utilize true [find -cell  F_MPH0001_LPC_E]
set_cell_property dont_utilize true [find -cell  F_MPH0001_LPC_H]
set_cell_property dont_utilize true [find -cell  F_MPH0001_LPC_K]
set_cell_property dont_utilize true [find -cell  F_MPH0101_LPC_E]
set_cell_property dont_utilize true [find -cell  F_MPH0101_LPC_H]
set_cell_property dont_utilize true [find -cell  F_MPH0101_LPC_K]
set_cell_property dont_utilize true [find -cell  F_MPH1001_LPC_E]
set_cell_property dont_utilize true [find -cell  F_MPH1001_LPC_H]
set_cell_property dont_utilize true [find -cell  F_MPH1001_LPC_K]
set_cell_property dont_utilize true [find -cell  F_MPH4001_LPC_E]
set_cell_property dont_utilize true [find -cell  F_MPH4001_LPC_H]
set_cell_property dont_utilize true [find -cell  F_MPH4001_LPC_K]
set_cell_property dont_utilize true [find -cell  L2S0101_LPC_E]
set_cell_property dont_utilize true [find -cell  L2S0101_LPC_H]
set_cell_property dont_utilize true [find -cell  L2S0101_LPC_K]
set_cell_property dont_utilize true [find -cell  LDE0001_E]
set_cell_property dont_utilize true [find -cell  LDE0001_H]
set_cell_property dont_utilize true [find -cell  LDE0001_K]
set_cell_property dont_utilize true [find -cell  LDF0001_E]
set_cell_property dont_utilize true [find -cell  LDF0001_H]
set_cell_property dont_utilize true [find -cell  LDF0001_K]
set_cell_property dont_utilize true [find -cell  LDM8083_E]
set_cell_property dont_utilize true [find -cell  LDM8083_H]
set_cell_property dont_utilize true [find -cell  LDM8083_K]
set_cell_property dont_utilize true [find -cell  LDR0001_E]
set_cell_property dont_utilize true [find -cell  LDR0001_H]
set_cell_property dont_utilize true [find -cell  LDR0001_K]
set_cell_property dont_utilize true [find -cell  LMX0001_E]
set_cell_property dont_utilize true [find -cell  LMX0001_H]
set_cell_property dont_utilize true [find -cell  LMX0001_K]
set_cell_property dont_utilize true [find -cell  LMX0001_LPC_E]
set_cell_property dont_utilize true [find -cell  LMX0001_LPC_H]
set_cell_property dont_utilize true [find -cell  LMX0001_LPC_K]
set_cell_property dont_utilize true [find -cell  LPH0001_E]
set_cell_property dont_utilize true [find -cell  LPH0001_H]
set_cell_property dont_utilize true [find -cell  LPH0001_K]
set_cell_property dont_utilize true [find -cell  LPH0001_LPC_E]
set_cell_property dont_utilize true [find -cell  LPH0001_LPC_H]
set_cell_property dont_utilize true [find -cell  LPH0001_LPC_K]
set_cell_property dont_utilize true [find -cell  LPH0002_E]
set_cell_property dont_utilize true [find -cell  LPH0002_H]
set_cell_property dont_utilize true [find -cell  LPH0002_K]
set_cell_property dont_utilize true [find -cell  LPH0101_E]
set_cell_property dont_utilize true [find -cell  LPH0101_H]
set_cell_property dont_utilize true [find -cell  LPH0101_K]
set_cell_property dont_utilize true [find -cell  LPH0101_LPC_E]
set_cell_property dont_utilize true [find -cell  LPH0101_LPC_H]
set_cell_property dont_utilize true [find -cell  LPH0101_LPC_K]
set_cell_property dont_utilize true [find -cell  LPH1001_LPC_E]
set_cell_property dont_utilize true [find -cell  LPH1001_LPC_H]
set_cell_property dont_utilize true [find -cell  LPH1001_LPC_K]
set_cell_property dont_utilize true [find -cell  LPH4001_LPC_E]
set_cell_property dont_utilize true [find -cell  LPH4001_LPC_H]
set_cell_property dont_utilize true [find -cell  LPH4001_LPC_K]
set_cell_property dont_utilize true [find -cell  LPH6001_E]
set_cell_property dont_utilize true [find -cell  LPH6001_H]
set_cell_property dont_utilize true [find -cell  LPH6001_K]
set_cell_property dont_utilize true [find -cell  LSC0001_LPC_E]
set_cell_property dont_utilize true [find -cell  LSC0001_LPC_H]
set_cell_property dont_utilize true [find -cell  LSC0001_LPC_K]
set_cell_property dont_utilize true [find -cell  LTL0001_E]
set_cell_property dont_utilize true [find -cell  LTL0001_H]
set_cell_property dont_utilize true [find -cell  LTL0001_K]
set_cell_property dont_utilize true [find -cell  L_LPH0101_E]
set_cell_property dont_utilize true [find -cell  L_LPH0101_H]
set_cell_property dont_utilize true [find -cell  L_LPH0101_K]
set_cell_property dont_utilize true [find -cell  L_LPH0101_LPC_E]
set_cell_property dont_utilize true [find -cell  L_LPH0101_LPC_H]
set_cell_property dont_utilize true [find -cell  L_LPH0101_LPC_K]
set_cell_property dont_utilize true [find -cell  L_LTL0001_E]
set_cell_property dont_utilize true [find -cell  L_LTL0001_H]
set_cell_property dont_utilize true [find -cell  L_LTL0001_K]
set_cell_property dont_utilize true [find -cell  L_MPH0101_LPC_E]
set_cell_property dont_utilize true [find -cell  L_MPH0101_LPC_H]
set_cell_property dont_utilize true [find -cell  L_MPH0101_LPC_K]
set_cell_property dont_utilize true [find -cell  MPH0001_LPC_E]
set_cell_property dont_utilize true [find -cell  MPH0001_LPC_H]
set_cell_property dont_utilize true [find -cell  MPH0001_LPC_K]
set_cell_property dont_utilize true [find -cell  MPH0101_LPC_E]
set_cell_property dont_utilize true [find -cell  MPH0101_LPC_H]
set_cell_property dont_utilize true [find -cell  MPH0101_LPC_K]
set_cell_property dont_utilize true [find -cell  MPH1001_LPC_E]
set_cell_property dont_utilize true [find -cell  MPH1001_LPC_H]
set_cell_property dont_utilize true [find -cell  MPH1001_LPC_K]
set_cell_property dont_utilize true [find -cell  MPH4001_LPC_E]
set_cell_property dont_utilize true [find -cell  MPH4001_LPC_H]
set_cell_property dont_utilize true [find -cell  MPH4001_LPC_K]
set_cell_property dont_utilize true [find -cell  MUX41I_DP_F]
set_cell_property dont_utilize true [find -cell  MUX81I_DP_F]
set_cell_property dont_utilize true [find -cell  PG_LMX0001_E]
set_cell_property dont_utilize true [find -cell  PG_LMX0001_H]
set_cell_property dont_utilize true [find -cell  PG_LMX0001_K]
set_cell_property dont_utilize true [find -cell  PG_LMX0001_LPC_E]
set_cell_property dont_utilize true [find -cell  PG_LMX0001_LPC_H]
set_cell_property dont_utilize true [find -cell  PG_LMX0001_LPC_K]
set_cell_property dont_utilize true [find -cell  PG_LPH0001_E]
set_cell_property dont_utilize true [find -cell  PG_LPH0001_H]
set_cell_property dont_utilize true [find -cell  PG_LPH0001_K]
set_cell_property dont_utilize true [find -cell  PG_LPH0001_LPC_E]
set_cell_property dont_utilize true [find -cell  PG_LPH0001_LPC_H]
set_cell_property dont_utilize true [find -cell  PG_LPH0001_LPC_K]
set_cell_property dont_utilize true [find -cell  PG_LPH0002_E]
set_cell_property dont_utilize true [find -cell  PG_LPH0002_H]
set_cell_property dont_utilize true [find -cell  PG_LPH0002_K]
set_cell_property dont_utilize true [find -cell  PG_LPH1001_LPC_E]
set_cell_property dont_utilize true [find -cell  PG_LPH1001_LPC_H]
set_cell_property dont_utilize true [find -cell  PG_LPH1001_LPC_K]
set_cell_property dont_utilize true [find -cell  PG_LPH4001_LPC_E]
set_cell_property dont_utilize true [find -cell  PG_LPH4001_LPC_H]
set_cell_property dont_utilize true [find -cell  PG_LPH4001_LPC_K]
set_cell_property dont_utilize true [find -cell  PG_LPH6001_E]
set_cell_property dont_utilize true [find -cell  PG_LPH6001_H]
set_cell_property dont_utilize true [find -cell  PG_LPH6001_K]
```