VLSI User's Manual

October 12, 2023

Preface

This manual was prepared for use in the laboratory section of the course "VLSI Design Methodology," taught in the Department of Electrical and Computer Engineering at the University of Toronto. The laboratory section of this course introduces students to commercially available VLSI design software from Cadence and Synopsys, with a special emphasis on the TSMC 65nm technology available through the Canadian Microelectronics Corporation (CMC, `http://www.cmc.ca`).

Some of the material included here was modified from Cadence on-line documentation, Synopsys training course materials, CMC on-line documentation, and from deliverables for a joint CMC/Micronet BiCMOS contract. Although every effort has been made to ensure the correctness of this material, updates to design tools and technology design-kits necessitate occasional updates to the content of this document. Corrections and updates will be made available at `http://www.vrg.utoronto.ca/vrg/DesignManual`

Contents

Chapter 1

Introduction

1.1 Introduction: VLSI Design Methodology

This manual is intended to serve as a concise reference guide to VLSI design at the University of Toronto. We assume that the reader has had some background in electronics and digital circuit design.

A variety of design styles or alternatives is available to integrated circuit designers. Some examples include

- The gate array approach, where the designer specifies the interconnections to be completed on a partially-fabricated wafer
- The field-programmable gate-array approach, where a fully-fabricated packaged chip is electrically programmed to perform a specific function
- The "full custom" approach, involving the full customization of all mask layers.

The tools and techniques described within this manual will assist the designer in the last of these topics, known as hierarchical design methodology.

The primary motivation for hierarchical design is the fact that thus far, VLSI processing technology has advanced faster than design capability. VLSI technology permits the realization of very complex circuits and there is, therefore, a need for design methods that can manage the complexity in design and thus fully exploit available VLSI capabilities.

Hierarchical design involves the successive decomposition of a complex design into a series of lower complexity problems which can be solved independently. It is also known by such terms as structured design, top-down design and the Mead-Conway approach. Besides the abstraction design philosophy, the principle of regularity (the use of a few parts repeated frequently) is also applied as opposed to customizing each piece of design. In this case, what might be lost in circuit performance is more than compensated for in the reduced circuit development time and, because most of the parts have been predefined and used before, there is less chance for error in the circuit. The advantages of structured and regular design include easier human interpretation and easier design verification.

1.2 This Manual

This chapter describes some of the commercial software tools available here at the University of Toronto. Subsequent sections introduce some of the tools used for schematic entry, physical layout, verification, and simulation. Next, we describe some of the Cadence and

Synopsys tools used for design using higher levels of abstraction, allowing designers to perform behavioural or structural simulations of large systems, as well as synthesis and optimization. The last chapter describes the use of Cadence First Encounter to perform automated place-and-route of a synthesized design.

1.3 The CMC Sustainable Technology Configuration

Most of the computer-aided design tools and technology-specific design kits accessible to designers have been made available with help from *CMC Microsystems* (formerly the Canadian Microelectronics Corporation, or **CMC**), and are installed on the `/CMC` disk partition which should be accessible from all Solaris and Linux computers on various research and teaching networks within the Department of Electrical and Computer Engineering. This disk package is also known as the **STC**, or *Sustainable Technology Configuration*, a methodology for keeping design software at Canadian universities participating in microelectronics research up to a standard level.

The CAD tools are installed under the directory `/CMC/tools` and process technology design kits are installed under the directory `/CMC/kits`. Each tool and design kit is installed in a suitably-named directory with an extension corresponding to its version number; a symbolic link is created, named after the tool or technology (with no version number), pointing at the currently supported version. Symbolic links in appropriate tool library directories point to specific technology design kit directories. The tools are accessible to all researchers, but technology design kits are normally protected such that only designers who have signed a *Confidential Information and Intellectual Property Agreement* for the specific technology, as discussed in 2.4 “Obtaining Access to Technology Information” on page 16.

1.4 Version of Tools

New versions of CAD tools are frequently updated by the vendors, to address problems and to incorporate new features; new releases are typically available on a quarterly basis. Examples in this manual were tested with the following releases of software tools:

- **Cadence IC.617:** IC tools, for schematic, layout, verification, analog environment
- **Cadence IUS.820:** NClaunch/NCverilog/NCvhdl/NCsim for Verilog/VHDL simulation
- **Cadence INNOVUS.18.10.000:** Innovus Digital Implementation (formerly “EDI/-SOC”), for place-and-route
- **Mentor modelsim.10.7c:** for HDL simulation
- **Synopsys syn_vN-2017.09:** for design synthesis
- **Mentor calibre_2007.3_27.17:** for design verification

1.5 Versions of Design Kits

Technology design kits are updated periodically, but far less frequently than the CAD tools used for designing microchips. The examples in this manual were tested with these technology releases:

CRN65GP: 65nm technology node provided by TSMC

Chapter 2

Cadence and Synopsys Tools: Preparation

The objective of this and the next three chapters is to help the designer become familiar with some of the CAD tools available from Cadence and Synopsys. This chapter covers the bare basics: how to set-up your Unix account to access the tools and technology libraries; how to set-up your account so that your environment variables point to directories and libraries used by required CAD tools; and how to launch and exit some of the tools.

The Cadence design environment supports all of your design activities beginning with schematic capture, and all the way through to physical design layout, verification, and hopefully design tape-out.

The integrated circuit designer typically begins by drawing a transistor schematic for their design. Such a schematic may be used to generate a netlist, which is just a file listing the devices used in a design and a list of the connections existing between device pins. This netlist is used to run a circuit simulation in order to verify correct function and operation.

This schematic can be used to guide physical mask layout, and used as a guide for inter-device and inter-block connectivity. The layout versus schematic (LVS) comparison tools can be used to run automated checks on the schematic and its corresponding layout to check whether the two representations of the same circuit agree. Software can generate a netlist for the layout by using technology-specific rules to recognize devices based on how various mask layers overlap, and then detecting the inter-connectivity between devices by merging pieces of metal layers, vias, etc. into networks.

The transistors and other devices used to create the schematic might have slightly different characteristics from the devices used in the layout. This can be caused by a number of factors, but is usually due to the fact that the transistors used in the schematic use only an estimated diffusion area size. Moreover, the wires in our "ideal" schematic are just that: wires; they have no associated capacitance or resistance. A completed layout can be used to generate a more indicative netlist for your completed design, and will include all of the parasitic components. These are determined using technology-specific rules to compute, for example, the capacitance seen by each wire trace and the resistance of this wire trace. It is important to note that the capacitances and resistances extracted from a layout might not have been intended by the designer; rather, they are caused by the intrinsic properties of wires, parallel wires and wires running over a ground plane (e.g. the substrate). Post-layout simulation, a simulation of the parasitic-extracted design, is usually required before a final sign-off, to guarantee proper circuit operation before it is

actually manufactured.

Design rule checking (DRC) may be run on the layout view to ensure that there are no violations of process rules, which are defined by the process engineers at the fabrication facility (known familiarly as the "foundry") to guarantee that a sufficient percentage of ICs are functional after manufacturing. Design rules restrict transistor sizes (e.g. minimum channel length, minimum diffusion area), minimum spacing between geometries (e.g. wire-to-wire distance, contact to gate distance), and other geometric dimensions that can impact device operation or performance. Electrical Rules Checking (ERC) may be performed on the extracted layout view of a design, to ensure that standard rules have not been broken, e.g. power and ground wires are not shorted together, no floating nodes exist.

The final steps of the design process requires the conversion of the layout view of the design into a format acceptable in commercial fabrication houses, "GDSII" (otherwise known as "GDS" or "Stream") format. For historical reasons, the conversion to this format is known as "the tape-out" because integrated circuit designs were typically transferred from design computers to the fabrication facility via magnetic tape before network connectivity became ubiquitous.

2.1 Setting-up Your Environment for CMC CAD Tools

The majority of the CAD tools described in this manual may be set up by using some built-in `csh(1)` or `tcsh(1)` commands; most tools have associated "source" files that may be used to set up your design environment for those tools. Because other shells like `bash(1)` and `sh(1)` use incompatible syntax, we recommend that you not use these for your default login shell. [See on-line manuals for the `chsh(1)` command to change your default shell.]

You need to add the following lines to the `.cshrc` file in your home directory to instantiate Cadence IC6 version and associated CAD tools:

```
if ( -e /CMC/tools ) then
   source /CMC/tools/CSHRCs/Cadence.IC617
   source /CMC/tools/CSHRCs/Cadence.SOC
   source /CMC/tools/CSHRCs/MMSIM.72
   source /CMC/tools/CSHRCs/Cadence.EXT
   source /CMC/tools/CSHRCs/Synopsys.2017.09
   source /CMC/tools/CSHRCs/Cadence.INNOVUS.18
   source /CMC/tools/CSHRCs/Synopsys.Formality
   #source /CMC/tools/CSHRCs/Cadence.XCELIUM
   #setenv LD_PRELOAD /usr/lib/x86_64-linux-gnu/libstdc++.so.6
   source /CMC/tools/CSHRCs/Mentor.Calibre
       source /CMC/tools/CSHRCs/Cadence.INCISIVE.15
   source /CMC/tools/CSHRCs/Cadence.IUS.820
endif
```

Note: You need to re-login to the UNIX machine after making changes to `.cshrc`

You can ensure that Cadence tools are included in your account set-up by issuing the command

```
% echo $path
```

The system should respond by displaying a list of paths that are searched when you type a command at the system prompt. If your system is set-up to access Cadence CAD tools, and other necessary CAD tools, then you should see these (or similar) directory paths displayed, among others:

```
/CMC/tools/cadence/IC/tools/bin
/CMC/tools/cadence/IC/tools/dfII/bin
/CMC/tools/cadence/IC/tools/dfII/local/bin
/CMC/tools/meta/Latest/bin
/CMC/tools/synopsys/sim/sparcOS5/sim/bin}
```

(You might see version numbers following /CMC/tools/cadence/IC... indicating that a specific version of the tools are being used; this is okay.)

2.2 On-line Documentation

Full product documentation for the Cadence family of tools, including fully searchable manuals, are available on-line using the command

```
% cdsdoc
```

Issuing this command will open a window showing a list of folders, each folder containing documentation about a Cadence product. Double-click on the folder to open it and examine the available documentation. Double-click on any document and it will be opened using your default HTML browser; a new browser process or an existing browser process may be used.

Local information on Cadence products can also be accessed after starting the Cadence IC tool, from the command interpreter window (CIW) by choosing VRG Info. News, from Cadence technical contacts and local staff will also be accessible using this menu link. Technology-specific documentation, named for the technology being used, is often provided via another drop-down menu in the Cadence IC tool CIW. Any changes made to the `/CMC` disk partition are recorded in the file `/CMC/CHANGES.vrg`. Items are listed in reverse-chronological (recent changes first) order.

In addition, most of the process technology design kits, stored under `/CMC/kits`, will also have at least one file with the prefix "`==README`" describing any local modifications that were made. Some of the more complex technologies include files with names beginning with "`==README`." and ending with "`quickstart`" or "`hints`" – be sure to examine these for hints and for answers to some frequently asked questions related to these technologies.

2.3 Customizing Your Cadence IC Tool Design Environment

2.3.1 Customizing your CadenceIC tool start-up environment: The.cdsinit file

New VRG accounts are pre-configured with a simple initialization file.cdsinit, which can be found in your home directory. This default configuration file will also load a local

configuration file .cdsinit.local which it searches for in the directory where you start your Cadence session. This set-up allows you flexibility in the customization of your design environment for each technology you might work with. Users with accounts on other networks (EECG or ELE) should copy this default configuration file to their home directory using

```
% cp  /CMC/kits/VRGlocal/cdsinit  $HOME/.cdsinit
```

2.3.2 Customizing Cadence IC tool environment variables: The .cdsenv file

Command options and values for environment variables used by some of the Cadence tools can be set up in the .cdsenv file. Cadence tools will first read the default version of this file, located at

`/CMC/tools/cadence/IC/tools/dfII/local/.cdsenv`

and then it will read additional information from the `.cdsenv` file located in your home directory (if it exists) so you can override any of the built-in settings. Modifying the `.cdsenv` file in your home directory might be useful after you obtain some experience with the Cadence tools. For example, to select a default simulator or waveform viewer different from the default versions, you may use the following settings:

```
asimenv.startup  simulator          'string  \spectre"
asimenv.startup  cds\_ade\_wftool   'string  \awd"
asimenv.startup  projectDir         'string  \./simulation"
```

(This is an example only, and is used to illustrate the contents of the .cdsenv file. Do not make these changes at this time.)

2.4 Obtaining Access to Technology Information

Process technology design-kits are available for the following technologies: *bicmos, cmosp35, cmosp18, cmrf8sf, cmos90nm, cmos65nm,* and a few other specialized technologies. Some of these technologies (bicmos) are relatively old and have no special access requirements. Access to all other technologies is severely restricted, and may only be set up after designers sign a "Confidential Information and Intellectual Property Agreement" for each and obtain required counter-signatuers, normally from their faculty supervisor or course instructor. For details, see

`http://www.vrg.utoronto.ca/UofT_Access_Only/TechnologyAccess.html`

NOTE: These agreements are legal documents. Your signature indicates that you will comply with the terms of the agreement. Any information to which you are given access after signing a non-disclosure agreement must remain confidential and can not be copied from university computers, at risk of expulsion from the University and/or criminal prosecution.

In this manual, we will be using the *tsmc65* technology for design examples. Although the techniques described in this manual are generally applicable to other technologies, the emphasis here will be on *tsmc65*. Design-kits for other technologies might not support all of the features described here, or may require modifications to these techniques.

Once access to the technology design kit is achieved, please read any design kit documentation, often located under a directory named `\doc"` in the design-kit's home under

/CMC/kits/ but sometimes located elsewhere in the kit hierarchy. (Check the \==README" files in each kit's top directory for additional information.)

2.5 Creating a Project Directory

Each technology requires some technology-specific libraries, hence it is strongly recommended that you create a separate subdirectory under your UNIX login account for each technology you use, or for each major project, if you wish. You should start the Cadence IC tools from within a technology-specific subdirectory to ensure proper access to technology libraries. For example, to initialize a work-directory for CMOSP35 designs, I might type the following in a UNIX xterm or terminal window. (The portion of each line beginning with the '#' would not be typed; this is merely annotating what each line does.)

```
% mkdir $HOME/CRN65GP           #create a new work directory named CRN65GP
% chmod go-rwx $HOME/CRN65GP    #protect this directory!
% cd $HOME/CRN65GP              #change directories
% startCds -t ic6-crn65gp        #start IC6 Cadence tool with CRN65GP kit
```

The command startCds is a shell script which creates a file cds.lib only if it does not already exist in the directory from which you invoked the script. This cds.lib file holds a mapping between the names of design libraries and their actual locations in the UNIX filesystem hierarchy. You must have signed any non-disclosure forms (or other forms required for access to a technology) and have been granted access to the technology information via UNIX file and group permissions before attempting to use the startCds script successfully with a restricted technology.

2.6 Starting a Design Session

Assuming that you now have a correct account setup and access to the technology has been granted, you can start a Cadence design session. Change directories to your technology-specific or design-specific work directory and start a session using the command:

```
% startCds -t <technology>
```

The `-t <technology>` option is used to specify the technology you wish to use. For TSMC65, use ***ic6-crn65gp*** as the technology option. Upon successful start-up of the software, a Command Interpreter Window (CIW) will be displayed. This is your Cadence design session console, and will be used to display any warning or error messages from the software. (A record of the session will also be stored in a file in the subdirectory `$HOME/CDSlogs`; this may be useful for VRG staff to help trouble-shoot any problems you encounter.)

Near the top of the CIW is a set of titles corresponding to pull-down menus (e.g. File, Tools, Options, etc.). Most technology design-kits include a menu item specific to the technology or to the company who provides the technology. As well, the "VRG Info" menu shows some potentially useful information. At the far right edge of the menu banner in the CIW (and on all other windows in most Cadence tools) you will see a "Help" menu with context-sensitive information.

Earlier, we mentioned that local customization of Cadence tool start-up scripts is recorded in the text file

`/CMC/CHANGES.vrg`

If the contents of this file was updated since your most recent design session began, a pop-up window displaying its contents will appear on your screen when you start a new session. If this window appears, please read (at least) a few of the most recent items; each begins with a date code in the form YYYYMMDD indicating when the item was added, with the most recent entries listed first. (Needless to say, this window will always appear the first time you start a Cadence session.)

In later design sessions, if you note any unusual problems or symptoms, please be sure to review any messages in the CIW before reporting the issue to local staff.

2.6.1 Potential problem: "Warning: ridiculously long PATH truncated"

As mentioned earlier, you might need to manually add the following line to the .cshrc file in your home directory:

`source /CMC/tools/CSHRCs/Cadence`

If the addition of this line introduces errors upon login such as *"Warning: ridiculously long PATH truncated"*, one work-around is to edit the `.cshrc` file again and comment out three specific lines. Near the beginning of the file there are two lines testing and setting a shell environment variable named `LEVEL`. Change these two lines by adding a ('#') character to the beginning, that is, change

```
if ( ! $?LEVEL ) then
    setenv LEVEL 1
into
# if ( ! $?LEVEL ) then
#   setenv LEVEL 1
```

Further down in the file, you will have to comment-out the corresponding else ... endif lines and whatever is between them. If you have a recent .cshrc file, you will change

```
else
  # count the depth ...
  set level=\$LEVEL ...
endif
```

into

```
#else
#  # count the depth ...
#  set level=\$LEVEL ...
#endif
```

2.6.2 Displaying a design session on a Windows PC

Cadence design sessions may be displayed on a Windows PC only if the PC is running an X-Windows emulator (e.g. Hummingbird or Cygwin) or VNC Viewer. Older versions of Cadence tools required that your display adapter settings were set for 8-bit (pseudo-color) or to 24-bit (true-color), but these limitations have been removed in modern releases of the tools.

2.6.3 Connecting using SSH

When connecting to a server using SSH (Secure Shell protocol), it is necessary to "Enable X11 forwarding" from your SSH client tool, to ensure that all sub-windows gain permission to write to your X11 display. Newer command-line SSH clients use a "-Y" option; older clients used "-X".

2.7 Design Kit Layout

The Cadence design tools available here are technology independent, i.e. they do not include any information for specific process technologies.. The actual technology data is obtained from various manufacturing facilities, and are set up in the form of a technology "design kit", or "process design kit" (PDK) saved under the `/CMC/kits/` directory.

2.7.1 Older Design Kits

Originally, most design-kits used at University of Toronto were obtained from technology vendors with the assistance of CMC Microsystems, and were set-up following a template under the directory /CMC/kits/<technology>.<version>, where <technology>is the name of one of the CMC-distributed design kits, or it may be the name of a technology accessible only to specific researchers under full non-disclosure agreements with commercial fabrication facilities. As new versions of the design kits are released, new directories are created with different "<version>" numbers. A link is created to the currently-supported version directory and is named without the ".<version>" suffix. A typical layout of a technology design-kit will include the following files and directories:

README	A file describing the technology and its features. Information here will be useful to designers and to system administrators.
cds.lib	A file mapping specific technology-related library names to the UNIX file-system hierarchy. This file is copied to your work directory upon your first invocation of Cadence tools for a specific technology, only if a file by that name does not already exist, and only if you have been granted access permission to the technology. (If you have not been granted access to the technology before starting Cadence, you will see a warning message in a pop-up window indicating "Problem reading the cds.lib file. Maybe there is a problem with the cdsd daemon..." In this case, exit from Cadence, remove the cds.lib file from the work directory if it is empty, and reread Section Obtaining Access to Technology Information.

display.drf	A file loaded into your Layer Selection Window (LSW) during Cadence sessions, indicating colors and stipple-patterns used for displaying mask layers
doc/	Documentation directory
dfII_lib/	Directory holding Cadence dfII technology libraries, for device schematic symbols, standard cells, and sample structures
models/	Directory containing simulation models for HSPICE, SPECTRE, and Verilog simulations
skill/	Directory containing customized SKILL programs required for this technology
synopsys	Directory containing files related to Synopsys tools
<technology>.tf	A technology file which is normally used to compile the libraries stored under the dfII_lib/ directory
<tech>.strmMapTable	A file used during conversion of layout data for "tape-out"

2.7.2 Newer Design Kits

In recent times, we have been obtaining more design kits without the assistance of CMC Microsystems. These kits are often configured according to the specifications of the CAD groups within the companies providing the technologies, and our agreements with these companies often prevent any modifications within the design kit directories. However, we still try to provide a set-up similar to that of the older kits. For set-up details, please always review files in the kit directories which have names with a `\==README"` prefix.

2.8 Cadence Tools Design Storage Structure

2.8.1 Design storage in Cadence Tools

The hierarchy of data storage is organized as follows:

- A library contains a number of related designs called cells.
- A cell will have different representations called views.
- A cellview could be one of layout, schematic, extracted, symbol etc.

For example, in a design library *lib_tutorial*, you may have cells *circuit1* and *circuit2*. The cell *circuit1* may have several representations, including a layout cellview and a schematic cellview. The cell *circuit2* may contain a symbol cellview and a schematic cellview.

NOTE: This hierarchical structure is managed by Cadence. Although the actual storage structure in your UNIX account corresponds to this hierarchy, it is strongly recommended that any library management (renaming cells, deleting cells or libraries) be done only from within the Cadence system, using the Library Manager, to avoid confusing the Cadence tools.

NOTE: Many of the Cadence tools and technology-related SKILL routines rely on specific names for cellviews, so changing these from their defaults is not recommended; use only the word layout for mask-layout cell views; use only schematic for schematic cell views, and so on.

2.8.2 Data management

All existing libraries in the library path will be displayed in a ***Library Manager*** window, which can be invoked by typing the **F6** function key while the mouse cursor is in the CIW. This is where you access all your designs. The ***Library Manager*** displays all libraries and files, allows you to open, copy and delete files. It is possible to copy entire designs from one library to another. The ***Library Manager*** can also be invoked by selecting **Tools → Library Manager** in the CIW menu. There is also a ***Library Browser*** that can be invoked by selecting the Browse option in the *Open File* form. This tool does not allow any data manipulation but you can use it to find and open a cell view.

2.8.3 Library path

This is a list of UNIX directories where Cadence looks for the design libraries previously created. The predefined search path is set automatically as required by the technology initialization routines; the current working directory (where you start your Cadence session) is included in the default search path.

Note: The libraries displayed in the Library Browser and Library Manager can be located in many different UNIX directories.

Modifications to the library path may be made using the **Tools → Library Path Editor**. This provides a mechanism by which you may modify the paths to existing libraries, or add new libraries and paths as they become available.

2.8.4 Compacting Library Data

Defragmenting, or compacting, design libraries allows you to remove the free space which can accumulate in design files, and thus reclaim large amounts of disk space. To invoke the defragmentation program, execute the following UNIX command:
`squishDB [-v] [-Log] [-Backup] [-Path searchPath] [-Lib libraryName]`
where

- **-v** sets the message level to verbose
- **-Log** logs all messages into a file named ./translate.log
- **-Backup** keeps a backup of each file compacted. Default is to keep no backup.
- **-Lib** list of libraries to be compacted, enclosed in quotations. Default is all libraries.
- **-Path** library search path enclosed in quotations. Default is current directory.

Alternatively, from within a Cadence design session, select **CIW → File → Defragment Data → Library...** and then complete the form, indicating the name of a library you wish to defragment.

2.9 Creating a New Library and Design

1. Create a new library called TestLib In the CIW window, selectFile → New → Library... and enter the following values in the form:

Name	TestLib
Directory	(select the directory for your library; the default is your work directory.)

   Under *"Technology File"*, select *"Attach to an existing techfile"* if it is not already selected. This method is recommended because any updates to the technology library data (e.g. bug fixes made by VRG staff) will be reflected immediately in your own library, and because of disk-space savings: attaching to a library prevents the copying of many basic cells and verification files into your disk area. The *"Compile a new techfile"* method is used only if you plan to make your own modifications to library information; this is useful for researchers developing their own technologies, and is not recommended for novices.

2. Click on **OK** to create a new library. If you selected the "Attach" method for creating a new library, you will see a pop-up prompting you to select the technology library to which you wish to attach. In general, select the technology library named after the technology you are using. For **TSMC65**, select the ***tsmcN65*** technology library, then click on OK. If you selected the "Compile" method for creating a new library, Cadence will prompt you for the full path to the ASCII technology file to be compiled into your library. If a pathname appears in the pop-up, it should be safe to use it. Click **"OK"**. After the library is created, it will be accessible from the Library Browser and Library Manager.

3. Move the cursor into the CIW window and use the **CIW → Tools → Library Manager** pull-down item, or press F6 to invoke the Library Manager.

4. Click on *TestLib* in the *Library Browser*. If you selected the *"Attach"* method, you will see no cells in your new library, but the basic cells for the technology will still be accessible from the technology library to which your library is attached. If you selected the *"Compile"* method of library creation, you will see a list of cells inside the library; these cells are used for wiring and for the place and route process.

5. Next, create a new cellview: Select **CIW → File → New → Cellview...**

6. In the new pop-up window, select your library name: click-and-hold on the button beside *"Library Name"*, and move the cursor until your library is highlighted, then release the mouse button. Type a new cell name (e.g. MyDesign). Type the view name *"schematic"* or use the Tool selection method to choose *"Composer - Schematic"* to automatically fill in the View Name box. (This latter method is preferred, because view names must be typed properly; a typing mistake in the

view name could cause unexpected behavior during design sessions.) For other view types (e.g. layout or symbol), select an appropriate entry in the View Name box.

2.10 Exiting from a Cadence Design Session

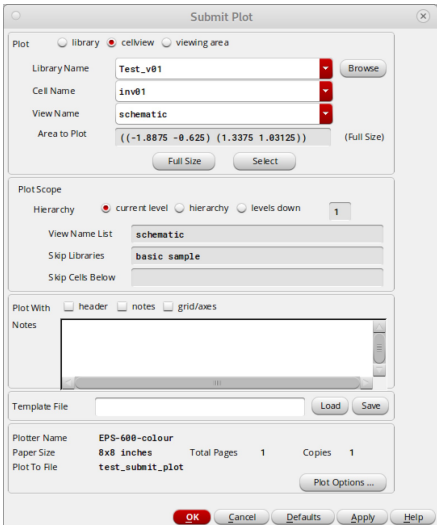
To exit from a Cadence design session, select **CIW** → **File** → **Exit** and then click on Yes when prompted *OK to exit icfb?* If you see a pop-up asking if you wish to save your display properties, it is safe to select Cancel, at which time your Cadence session will end. If, however, you see a pop-up asking if you wish to save your work, then Cadence has noticed that you have neglected to save some of your work. You may save all of the unsaved cells (to commit the designs to disk), none of them (to discard any changes), or you may select the cells to be saved before Cadence exits by turning on specific checkboxes.

2.11 Printing Schematic or Layout Views

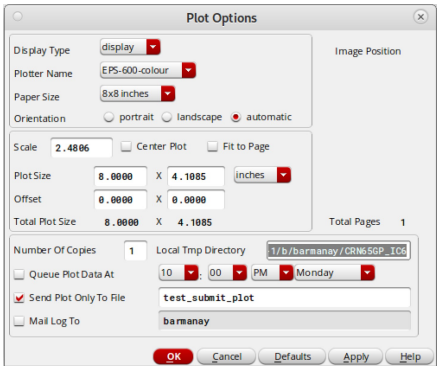
1. Open your schematic by selecting the cellview in the Library Manager.
2. Select **File** → **Print**. The Submit Plot window appears.
   (a) In the "Plot" section, verify that the cellview information is correct.
   (b) In the "Plot With" section, turn off the Header option.
   (c) You can safely ignore the "Template File".
   (d) The plotter name and paper size have default values. If these are not appropriate ...
3. Click on Plot Options... to pop up the Plot Options form. You have a choice of plotter name. To print to the Hewlett-Packard LaserJet in LP477, select lj477.
   (a) If you don't want Cadence to mail you a plot notification message, you can also turn off the *"Mail Log To"* option at the bottom of the form.
   (b) Click on OK to exit the options form.
4. Click on OK in the Submit Plot form. A message will be displayed in the CIW regarding success or failure.
5. If you want to obtain a plot file in Encapsulated PostScript format, suitable for importing into FrameMaker or other document preparation software, select the plotter name EPS, and indicate that the design is to be plotted to a file. Type the name of the file into which the EPS data will be saved.

2.11.1 Adding your own plotter configuration

After loading default plotter initialization data from `/CMC/kits/VRGlocal/cdsplotinit.vrg` the Cadence tools will read additional configuration data from the file `$HOME/.cdsplotinit` if it exists. You can customize your Cadence plotting needs by preparing this file. For details, see on-line Cadence documentation



(a) Submit Plot window



(b) Plot options window

in `cdnshelp`. EECG users need to copy the file `/CMC/kits/VRGlocal/cdsplotinit.eecg` into their own `$HOME/.cdsplotinit` to enable plotting.

2.12 Cadence Session Logs

A record of each of your Cadence design sessions - - all of the messages appearing in the CIW - - is saved in a file in the directory `$HOME/CDSlogs/` and is named `CDS.log.<process_ID>`. In theory, you can use a log file to recreate a previous session, but you might need assistance from VRG staff. In practice, these log files are only useful for VRG staff to help debug problems you might be experiencing in your design sessions. These log files could be very large, depending on the duration of your design sessions. It is suggested that you clear the CDSlogs directory at least once a week depending on how frequently you use Cadence. An example of commands you can add to your `$HOME/.logout` file to do this automatically when you logout is available in the file `/CMC/kits/VRGlocal/logout_sample`

2.13 TSMC-65nm Specific Items

2.13.1 Design Kit Libraries

Please add following 3 lines to cds.lib file in local project directory to gain access to the digital and IO directories.

```
DEFINE tcbn65gplus  /CMC/kits/tsmc_65nm_libs/OA_libs/tcbn65gplus/
    tcbn65gplus_200a/TSMCHOME/digital/Back_End/cdk_OA/tcbn65gplus
DEFINE tpan65gpgv2od3 /CMC/kits/tsmc_65nm_libs/OA_libs/tpan65gpgv2od3/
    tpan65gpgv2od3_200b/TSMCHOME/digital/Back_End/cdk_OA/tpan65gpgv2od3
DEFINE tpbn65v      /CMC/kits/tsmc_65nm_libs/OA_libs/tpbn65v/tpbn65v_200b/
    TSMCHOME/digital/Back_End/cdk_OA/tpbn65v
DEFINE tpfn65gpgv2od3 /CMC/kits/tsmc_65nm_libs/OA_libs/tpfn65gpgv2od3/
    tpfn65gpgv2od3_200c/TSMCHOME/digital/Back_End/cdk_OA/tpfn65gpgv2od3
```

Several libraries are provided with the current release of the TSMC65 Design Kit:

Library Name	Description
tsmcN65	Cells for schematic capture, layout extraction, netlisting and LVS
tcbn65gplus	Various, standard logic cells with symbols, schematics and layouts
tpzn65gpgv2	I/O pads and corner cells

2.13.2 Transistor models

MOS transistor models are available for Synopsys HSPICE and Cadence SPECTRE.

2.13.3 Documentation

Design rule related document can be found at following location
`/CMC/kits/tsmc65nm/CRN65GP-OA/DR/TN65CLDR001_2_3.pdf`
Model documentation can be found at following location
`/nfs/vrg/cmc/cmc/kits/tsmc_65nm/CRN65GP-OA/TN65CMSP018_V1.1/TN65CMSP018_1_1.pdf`
More application notes can be found at following location
`/nfs/vrg/cmc/cmc/kits/tsmc_65nm/CRN65GP-OA/TN65CMSP018K3_V1.0C/PDK_doc/TSMC_DOC_WM/PDK`

Chapter 3

Cadence: Schematic Entry

3.1 Power Supply Connections

Some technology-dependent design-kits allow for the use of global nodes in a design. When using the Cadence tools, global nodes are identified by an exclamation point following the node name: *gnd!, vdd! or vss!* for example. However, some libraries in the **TSMC65** design kit do not allow the use of global node names, but circuit simulators, such as Spice, require that at least a global ground node exists.

To overcome this limitation make sure that ground and power connections are created for each component in your design hierarchy. Then, global power connections can be made at the top level of your design hierarchy, such as in the test bench schematic. Alternatively, you can use global node names exclusively in your design, where supported.

3.2 Creating Your First Transistor-Level Schematic

This section will guide you though the steps needed to take to create your first Cadence schematic, a transistor schematic for a logic inverter as shown in Fig. 3.1. Steps will be provided for TSMC65 technology.

(a) Circuit diagram (b) Virtuoso schematic

Figure 3.1: Schematic of logic inverter

1. Create a new library, or use the existing library TestLib library from the previous chapter, for the technology node with which you wish to work. If creating a new library, use the Attach to existing techfile method described in the previous chapter

2. Create a new cell view in your library, in the CIW window, select **File** → **New** → **Cellview**

   The Create New File form appears; fill in the following values:

   Library Name: TestLib (or select your library from the Library drop-down selection list)
   Cell Name:inverter
   View Name: schematic (or select Composer-Schematic from the Tool drop-down selection list)

   Note: Make sure that you spell the view name correctly, otherwise the Cadence tools will notrecognize the new cellview as a schematic. In that case, tools and menus associated with a schematic will not be accessible.

   Click OK to create the new cell. A new schematic design entry window will open with the name of the library, cell, and cellview displayed in the windows title bar. A schematic specific pull-down menu should now be available along with a shortcut icon menu along the left size of the window.

3. Create and place an instance of an n-channel MOS transistor, nfet, in your schematic:

   (a) Select **Add** → **Instance** from the pull-down menu or use the ***i*** keyboard shortcut key to open the *Add Instance* dialog.

   (b) Click the Browse button to bring up the *Library Browser*.

   (c) In the *Library Browser* dialog window select the following values or fill in the appropriate fields in the *Add Instance* dialog window:

Library	**Cell**	**View**
tsmcN65	nch	symbol

   (d) In the *Add Instance* form specify the channel width and length of the nfet in meters. Dimensions can be specified in either SI units or scientific notation. Enter $400n$ or $400e-9$ to specify the width of the new nfet instance. Keep the default value of $60n$ for the length of the transistor instance.

   (e) Without closing the Add Instance dialog, return to the schematic window. Notice that the outline of the transistor symbol follows the mouse cursor. Left-click to place an instance of the nfet symbol. Transistor parameters will be displayed beside the placed symbol.

   Note: Multiple instances of a similarly sized transistor can be placed sequentially, until you abort by pressing the ESCAPE key or Cancel on the *Add Instance* dialog window. You can also change the size, or other parameters, of subsequent instances by bringing up the *Add Instance* dialog.

4. Repeat the above steps in order to place a pfet instance, *pch* that has a channel width of 0.8um and channel length of 60nm.

   Note: You can edit the properties of any symbol after it has been placed, by selecting the symbol and using the **q** keyboard shortcut key. This will bring up the Edit Object Properties dialog which allows you to modify that device's parameters.

5. Place schematic pins for the input, output, and power supplies connections:

(a) Select **Add→Pin** or use the **p** keyboard shortcut key.

(b) In the **Add→Pin** dialog enter the names for the input, output and power supply terminals in the Pin Names field: **in out vdd gnd**. This will allow you to place these pins in a sequential manner, starting with **in**; pin names will vanish from the Pin Names field as you place them.

(c) Before placing the pins specified in the Pin Names field, make sure to select the proper direction for each pin:

Pin Name	Pin Direction
in	input
out	output
vdd, vss	inputOutput

Note: HSpice is NOT case sensitive. Do not use case to differentiate between two distinct nets.

6. Zoom out so that you can see all of the symbols you have placed, select **Window→Fit** or press **f**

7. Connect symbols and pins using wires:

   (a) Select **Add→Wire (narrow)** or use the

   (b) Make the required connections, as shown in Schematic of logic inverter, by left-clicking for each end-point and turning point of the interconnecting wire; double-clicking will terminate the wire being currently drawn.

8. You can also connect different symbol or pin terminals by using common wire names; wires with identical names are considered to be implicitly connected. This might greatly simplify your schematic and make it easier to understand by greatly reducing the amount of wiring in the schematic.

   (a) Draw a wire segment from a symbol or pin terminal.

   (b) Select **Add→Wire Name** or the **l** keyboard shortcut key to bring up the *Add Wire Name* dialog window.

   (c) Enter the names of the net names in the Names field; net names will vanish from the Names field as you assign labels to wires.

   (d) Place the labels by clicking over the desired wire.

9. Run a schematic check: select **Check→Current Cellview**. Check the CIW window for any schematic errors or warnings identified. A detailed description of the schematic check is given in 3.6.

10. Save your design by selecting **Design→Save**, using the **S** keyboard shortcut key. You can also use the first icon at the left hand side of the schematic window, **Design→Check and save**, or the **X** keyboard shortcut key to perform an automatic schematic check and save.

11. Plot your schematic: select **Design→Plot Submit**. In the *Submit Plot* dialog window select an appropriate printer and plot size by following the **Plot Options** button at the bottom of the window.

3.3 Creating a Schematic Symbol

For hierarchical designs, smaller components are used to build up successively bigger designs. To be able to use the Inverter design as part of another schematic it is necessary to first create a symbol view of the schematic.

Note: You can also use this methodology to create a symbol view from an extracted layout cellview, later on during physical design.

1. Open the Inverter schematic.

2. Select **Design→Create Cellview→From Cellview...**

3. In the *Cellview from Cellview* dialog window keep the default settings, and click OK.

4. A *Symbol Generation Options* dialog window will pop-up. This window allows you to choose the positioning of the various pins on the four sides of the symbol, ordered top-to-bottom and left-to-right.
   Note: It is good practice to establish a pin ordering convention for all symbols. A typical convention is to put all inputs on the left side of the symbol, all outputs on the right, low voltage (ground) supply connections on the bottom, and high voltage (VDD) connections on the top.

5. Click the **Load/Save** selection box; in the expanded area of the dialog box select analog from the *Load Symbol Template Configuration* drop-down list and click the **Load** button. This will ensure that the proper symbol template is loaded for use with the Analog Design Environment, the Cadence simulation GUI.

6. Click **OK**. The newly created symbol cellview will be displayed.

3.4 Creating a Hierarchical Schematic

1. Create a new schematic cellview named *shift* in your library and then edit the *shift schematic*.

2. Place an array of flip-flops in your schematic, as shown in Schematic of four flip-flop shift circuit.

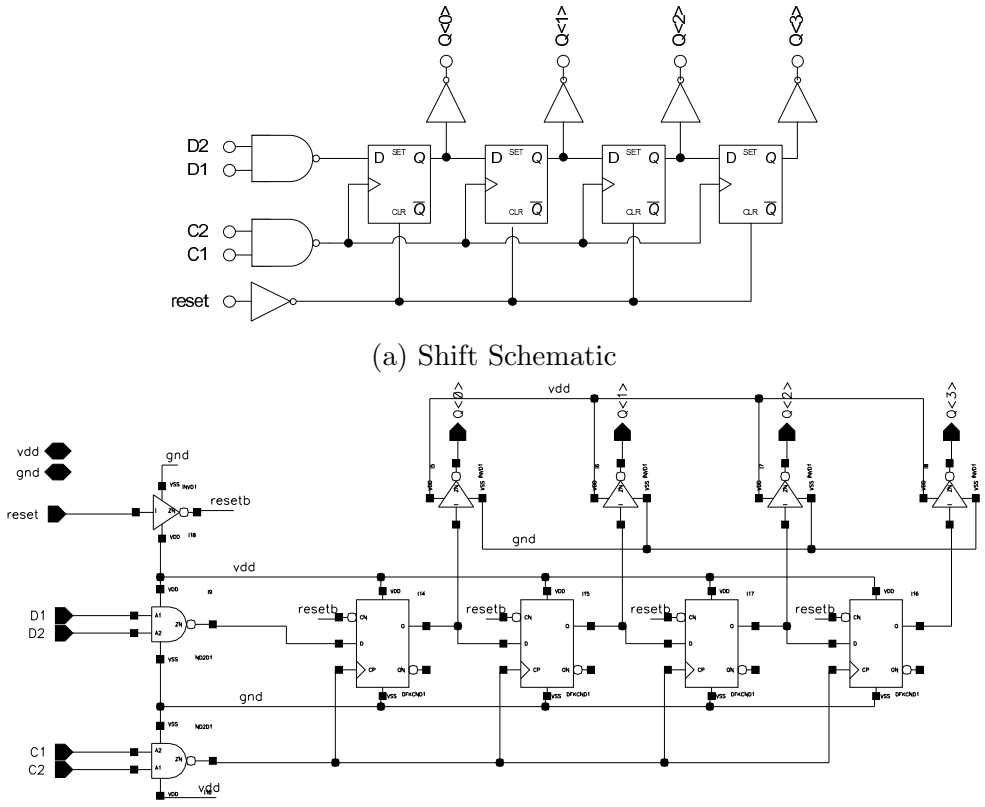
   (a) Select **Add→Instance** to open the *Add Instance* dialog window.

   (b) Select an instance of a positive clock-edge triggered D flip-flop with 2 drive strength: Using the *Library Browser* select

Library	Cell	View
tcbn65gplus	DFKCND1	symbol

   (c) In the *Add Instance* dialog window specify the size of the array at the bottom of the form to be Rows: 1, Columns: 4.

   (d) Place the first instance of the flip-flop.



(a) Shift Schematic

(b) Cadence Virtuoso schematic

Figure 3.2: 4-bit shift register design

(e) After you have placed the first instance, you should see outlines of all other instances in the array move along with your mouse cursor. The second instance placement indicates the spacing to be used for the second and successive instances. Place the second cell in order to automatically place a horizontal row of flip-flops.

(f) Click Cancel or press the ESC key to close the *Add Instance* dialog window.

3. Place an array of inverters, you will need to click Rotate in the Add Instance dialog window in order to orient the inverters properly:

Library	**Cell**	**View**
tcbn65gplus	INVD1	symbol

4. Place the rest of the gate instances as shown in Schematic of four flip-flop shift circuit using the following symbol for the NAND gates.

Library	**Cell**	**View**
tcbn65gplus	ND2D1	symbol

5. Add input and output pins to your design. Bring up the Add Pin dialog window and enter the following pin names: **D1**, **D2**, **C1**, **C2**, **Reset**, **Q<0:3>**.
Note: A bus is defined by labeling a wire with the bus name followed by the bit range of the bus: *busName<LSB:MSB>*. It is common practice to use a thick wire

to distinguish a bus from a wire. To bring out a single wire from the bus, connect a wire to the bus and label it specifying the bus name and bit number in angle brackets: for example *busName<bitNumber>*.

6. Interconnect the gates and pins together, as shown in Schematic of four flip-flop shift circuit, using wires by selecting **Add→Wire (narrow)** from the pull-down menu, or by using the **w** keyboard shortcut key.
   Note: If you want to place an array of wires, place one wire, select it, and then choose **Edit→Copy** from the pull-down menu or use the **c** keyboard shortcut key. Use the array option in the *Copy* dialog window.

   (a) Select input as the pin direction and place the input pins in the schematic.

   (b) Select output as the pin direction and select Bus Expansion, Multiple Placement, and click Rotate and place the output pins.
   Note: You will also need to create pins for the power connections if you are not using global net names for the power supplies.

7. Run a design check to establish hierarchical connectivity of your design and to add that information into the design's database. Select **Check→Current Cellview** and check the CIW for any errors and warnings. You should get four warnings related to the floating **qb** output pins for each of the four flip-flops. (You may ignore these.)
   Note: You can use **Check→Find Markers** to find and explain error or warning markers in the schematic.

8. Save your design.

3.5 Traversing the Design Hierarchy

When working with a hierarchical schematic you can always view the details of instance cells by using the **Descend** command. In the case of design kit cells, schematic views may not be available, but you can descend down in the hierarchy to see the cell's other available views, such as layout, extracted, etc.

1. Select an instance of the flip-flop cell. Press and hold the middle mouse button to invoke the Instance pop-up menu and select *Descend Read...* or use the **e** keyboard shortcut.

2. In the *Descend* pop-up dialog, select layout in the *View Name* drop down list and click **OK**. The mask layout for the flip-flop cell should now be displayed.
   Note: To view all mask detail contained within sub-cells of this layout, use the **F** keyboard shortcut. To turn off this detail use the **CTRL+F** keyboard shortcut.

3. To go back a level in the hierarchy select **Design→Hierarchy→Return** or use the **B** keyboard shortcut.

3.6 Running a Schematic Check

This feature runs some simple tests to ensure that a schematic you have drawn has no major errors, like floating wires, unconnected pins, shorted output pins, and others.

1. Open your *TestLib shift schematic* for editing.

2. Initially we need to examine and edit the rules used to perform the schematic check. Select **Check→Rules Setup**. The *Setup Schematic Rules Checks* pop-up dialog window will appear.

   (a) Select **Normal** from the *Packaged Checks* drop-down list.

   (b) In the *Logical* tab, select the **warning** radio button for the *Floating Output pins* check rule.

3. Run a design check on the schematic by selecting **Check→Current Cellview** or use the **x** keyboard shortcut.

4. The resulting pop-up dialog window summarizes the number of errors and warnings found in your schematic. The errors and warning will also be highlighted on the schematic. The CIW window will list the warnings and errors generated in detail. The following is a list of warnings that you should get for the TestLib shift schematic.

```
Extracting ``shift schematic''
Warning: Pin ``QN'' on instance ``I3'': floating output.
Warning: Pin ``QN'' on instance ``I2'': floating output.
Warning: Pin ``QN'' on instance ``I1'': floating output.
Warning: Pin ``QN'' on instance ``I0'': floating output.
```

5. You can also step through all warnings and errors one at a time with visual feedback on the schematic. Select **Check→Find Marker**... or use the **g** keyboard shortcut. The *Find Marker* pop-up dialog window will appear. Select **Zoom to Markers** in the dialog window to have the display zoom in (very closely) to the error/warning reported in the list.

6. Once you have verified that a set of warning/errors are acceptable, you can disable them from being checked by future checks by modifying the *Check Rules*. It is possible to get multiple warnings/errors for one problem, so re-run the check after repairing problems to ensure that all errors are repaired.

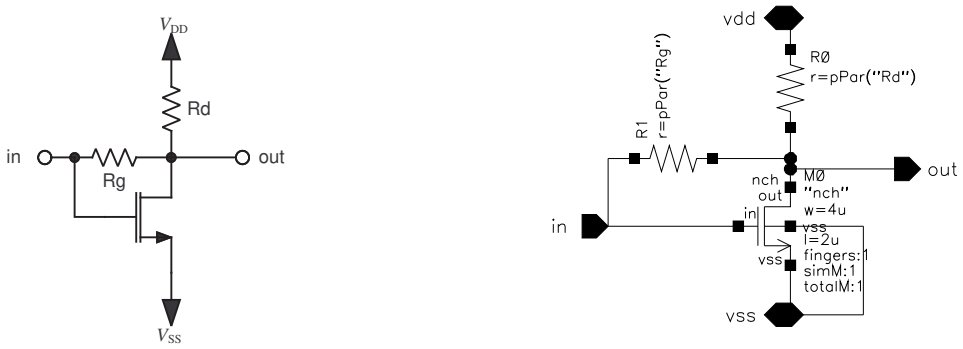
3.7 Pass Parameters and Variables

Quite often, a circuit structure is repeatedly in the top level design. Cadence Analog Artist, the circuit simulator interface, has an efficient way to handle multiple uses of a circuit structure where the difference in each occurrence is the value of one, or more, of its basic components (such as resistor values, transistor sizes, etc.). Cadence allows you to assign basic component values as pass parameters in the building block and lets you pass the desired component parameters to each block (in the form of a user-defined symbol) from a higher hierarchy level. This is analogous to a software program, where a function may be defined once, but called from a higher level function numerous times and with different parameters each time.

Another method to vary component values is through the use of component variables. This is used at the top-level design for parametric analysis. Parametric analysis allows you

to sweep specific component values during a schematic simulation. This is described later in a chapter on schematic simulation. In the following example, both pass parameters and top-level component variables are used in a design.

1. Create a new amplifier schematic in your TestLib library. Enter the schematic as shown in Schematic of CMOS amplifier. Use components *resistor* and *nch* from the *analogLib* and *tsmcN65* library respectively.



(a) amplifier circuit (b) Cadence Virtuoso circuit

Figure 3.3: Schematic of CMOS amplifier

2. Assign pass parameters for the resistor values: instead of typing numeric values in the *Add Instance* pop-up dialog window, use the following resistor "values": **pPar("Rd")** and **pPar("Rg")**, for the drain and gate resistors respectively.

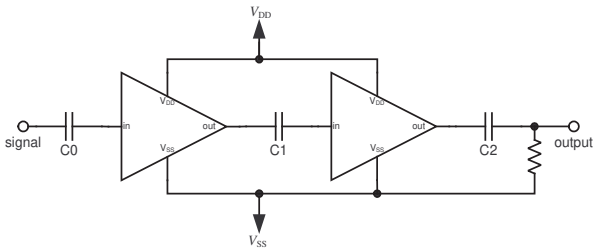
3. Create a symbol for the amplifier schematic.

4. Create a new *mul_stage schematic*. Enter the schematic as shown in Schematic of a multi-stage hierarchical CMOS amplifier. [Warning: Image ignored]

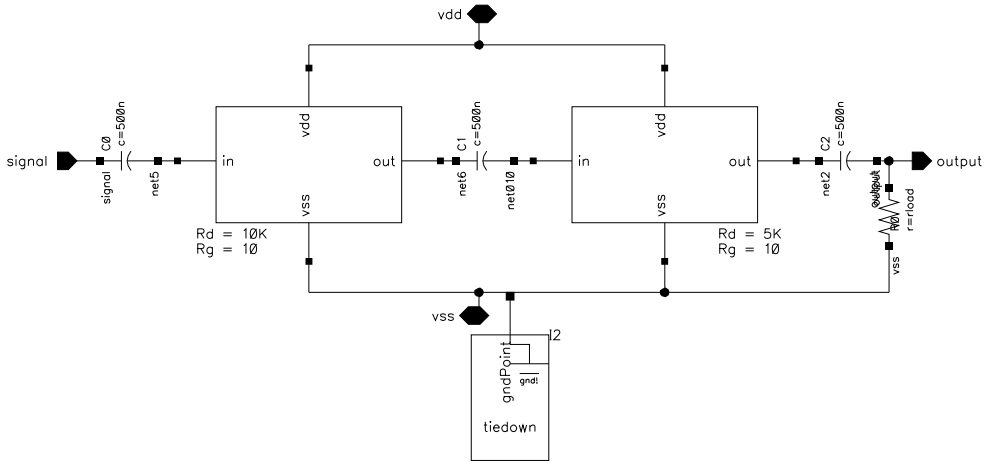
5. When you instantiate the amplifier symbol, you will find two CDF (*Component Description Format*) parameters - Rd and Rg - at the bottom of the Add Instance pop-up dialog. Enter the values as shown.

6. When placing the resistor, assign *rload* as the resistance value. This is now a component variable, which may be assigned a real value during simulation. (This topic will be discussed in a later chapter.) item Add wires to connect the *vdd* and *vss* terminals of your amplifier symbols.

7. Check and save the schematic.



(a) multi-stage amplifier circuit



(b) Cadence Virtuoso circuit

Figure 3.4: Schematic of a multi-stage hierarchical CMOS amplifier

3.8 Schematic Entry Keyboard Shortcuts, or *bindkeys*

Command	Bindkey	Menu Selection
Save	S	Design→Save
Check & Save	X	Design→Check and Save
Check Current Cellview	x	Check→Current Cellview
Find Markers	g	Check→Find Marker...
Add symbol instance	i	Add→Instance
Edit Object Properties	q	Edit→Properties→Objects...
Add pin	p	Add→Pin
Add wire	w	Add→Wire (narrow)
Label wire	l	All→Wire Name
Zoom to Fit	f	Window→Fit
Zoom to selection	z	Window→Zoom→Zoom In
Zoom out 2x	[	Window→Zoom→Zoom Out by 2
Zoom in 2x	]	Window→Zoom→Zoom In by 2
Descend Hierarchy (read)	e	Design→Hierarchy→Descend Read
Descend Hierarchy (edit)	E	Design→Hierarchy→Descend Edit
Ascend Hierarchy	B	Design→Hierarchy→Return

Chapter 4

Layout Verification: Introduction

The layout view of a design consists of the actual mask features (rectangles and polygons) which are to appear on the final masks used during the fabrication of a microchip. Cadence Virtuoso, the layout editor, allows each mask layer to be represented on the screen by a different colour or stipple pattern. These layer patterns are indicated in the Layer Selection Window (LSW) during a layout session. In this window, the *current layer* (i.e. the layer in which any new feature will be drawn) is highlighted. This current layer can be changed by left-clicking on the layer name in the LSW.

Note: There are several copies of some layers in the LSW. Normal drawing layers are denoted by a layer type of *drw*, for drawing. There are other types of layers as well, each layer type having a special purpose, such as *pin* for pins, *nt* for nets, *wg* for warnings, and *er* for errors. Designers should use only the *drw* layer to draw mask features.

4.1 Creating a Mask Layout

1. Create a new inverter layout cell view in your TestLib library: **File** → **New** → **Cellview**

   Library Name: **TestLib** or select your library from the Library drop-down selection list

   Cell Name: **inverter**

   View Name: **layout** or select Virtuoso from the Tool drop-down selection list

2. Adjust snap spacing for the layout as follows

   (a) press *e* to open *Display Option* window

   (b) change the Major, Minor, X-snap and Y-snap spacing as follows

Minor	**Major**	**X snap**	**Y snap**
0.005	0.05	0.005	0.005

3. Figure 4.1 shows one possible layout of a CMOS inverter. Try to reproduce this layout. The PMOS transistor has a W/L of 0.8μm / 0.06μm, while the NMOS has a W/L of 0.4μm / 0.06μm.

   The PMOS transistor consists of a polysilicon gate crossing active covered by p-plus in an n-well. The NMOS transistor consists of polysilicon gate crossing active covered by n-plus. Note also the n-well contacts (active covered by n-plus in the

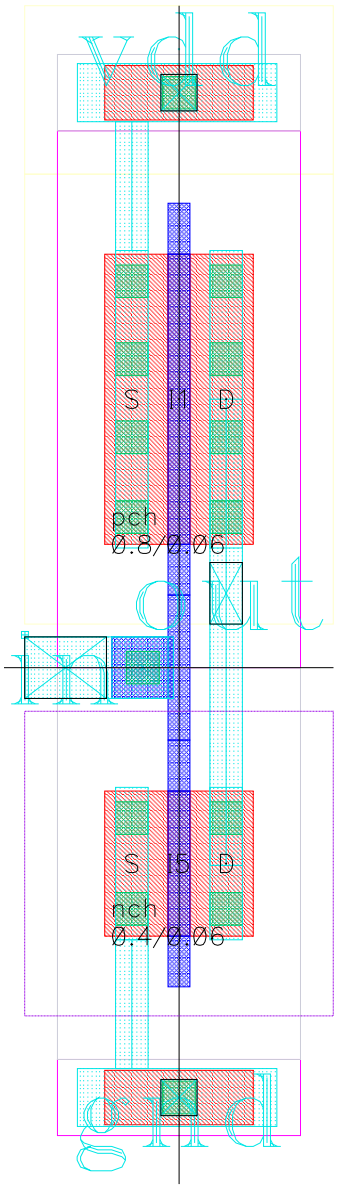


Figure 4.1: Example layout of a logic inverter

n-well) connected by metal to VDD near the top of this layout, and the substrate contacts (active covered by p-plus outside the n-well) connected by metal to VSS near the bottom of the layout. These represent the *bulk* connections for the MOS transistors.

Mask features are drawn in the layout editor by:

- left-clicking on a drawing layer in the LSW to make it the current layer
- invoking one of the **Create**→ commands (rectangle, polygon, or path)
- for a rectangle, left-clicking the diagonally opposite corners, or,
- for a polygon, left-clicking the polygon points in sequence, with a double-fast-click on the final point.
  Note: Any polygons drawn on the same layer that touch or overlap are considered to be electrically connected.

Alternatively, you can use user customizable template MOS transistor footprints from the *tsmcN65*:

(a) Select **Create→Instance** or use the **i** keyboard shortcut, which brings up the *Create Instance* pop-up dialog window.

(b) Select the **tsmcN65** library for the *TSMC65* technology node.

nch	NMOS template, no bulk connection
pch	PMOS template, no bulk connection

4. Specify the width and length of the gate, as well as any other required options and place the transistor instance.
Note: You might not be able to see the transistor instance details; if this is the case, then use **SHIFT+f** to show instance details.

5. Make sure to make the bulk connections for both NMOS and PMOS. The layout of standard cells can be used as reference design.

PMOS bulk connection

(a) Extend the NW to allow the placement of bulk contact.

(b) Create a N+ contact by drawing a rectangle with *NP drw* layer

(c) Define the active area using OD layer

(d) Place diffusion to metal1 contact using *CO drw* layer. Make sure the dimensions of the contact are allowed as per DRC rules

NMOS bulk connection: Ignore step 4.1 and follow steps 4.b to 4.d and replace the *NP drw* layer with *PP drw* layer.

4.1.1 Making connections using Path Stitching

It is possible, although extremely tedious, to connect polysilicon or metal connections on transistor together using polygon shapes for wires, vias, etc. However, a much easier alternative is to draw paths, which can be made to traverse several layers as the connection crosses over or under existing objects in the layout.

When drawing paths, it is very important to use a path whose width is an even multiple of design grid units. Since a path is defined by a set of coordinates along its center-line (with half of the path-width on either side), an odd number of snapping grid units in width will result in the edges of the path being off-grid (that is, on a 0.005 grid in the TSMC65 technology). When a design is submitted for fabrication to CMC, all paths are converted to polygons.

In some cases, you cannot draw a path on a single layer directly between two points due to intervening mask features, so you use path stitching to change layers within the path. Path stitching automatically changes the path from one layer to another, placing an appropriate contact to connect the two layers. The Path command selects the contact from the library technology file.

1. Turn on gravity to snap the cursor to objects

(a) Press **SHIFT+f** to display all levels.

(b) Select **Options→Layout Editor...** which brings up the *Layout Editor Options* pop-up dialog window.

(c) Check to make sure *Gravity* is selected.

(d) Change *Aperture* to **0.1**. The gravity aperture controls the distance at which the cursor snaps to objects: a setting of 0.1 means that the cursor should snap to an object when it is within 0.1 to that object.

(e) Change *Depth* to **2**. The cursor can snap to object within instances. The depth setting specifies how many layers down in the hierarchy the cursor should snap to.

(f) Click **OK**.

2. To draw a path using the *metal1* layer, click on the *m1 drw* layer in the LSW.

3. Select **Create→Path**, bringing up the *Create Path* popup dialog window.
Note: The path width is set to 90nm, which is defined in the technology file as a property of the metal1 layer.

4. Select **Design→Options→Display Options** and set *Snap Modes* to orthogonal.

5. Start making the connections between objects by clicking on an appropriate cell layer to start the path. Left click on a point where you'd like to change path directions.

6. Use the *Change To* field in the *Create Path* popup dialog window to change from *metal1* to *metal2*. Press *metal1* next to *Change To* in the *Create Path* popup dialog window. A list of layer names appears; these are the layers you can change to from metal1, based on the metal1 contacts defined in the library's technology file.

7. Slide the cursor to *metal2* and release the mouse button.

8. Move the cursor back into the cell view. A metal1 to metal2 contact appears on top of your cursor.

9. Click to anchor the contact.
Note: The width of the *metal2* path changes to 0.1μm, as defined in the technology file.

10. You can change to any of the given layers during path stitching. Double click on the same point to complete the path currently being drawn.

11. Complete the connections of the entire layout.

4.1.2 Creating Labels

Labels are a handy way of embedding more information in a layout, in such a way that will not affect the resulting layout. Labels serve the same purpose in layouts as comments do in software code.

1. Check the LSW for *text dg* (text drawing layer), *metal1 pn* (metal1 pin layer) and *metal2 pn* (metal2 pin layer). If these layers do not exist, you have to include them in the list of valid layers that are shown in the LSW.

   In the LSW, select **Edit→Set Valid Layers**. A popup window with all layers defined in the technology file is displayed. Look for the above layers and select them by clicking on the box after the layer name.

2. Create labels for information purposes, by clicking on the *text dg* layer in the LSW.

3. Select **Create→Label**, the *Label* popup dialog window will appear.

4. Type **vdd** in the *Label* field. Move the cursor over the power line at the top of the cell. Click to place the label.

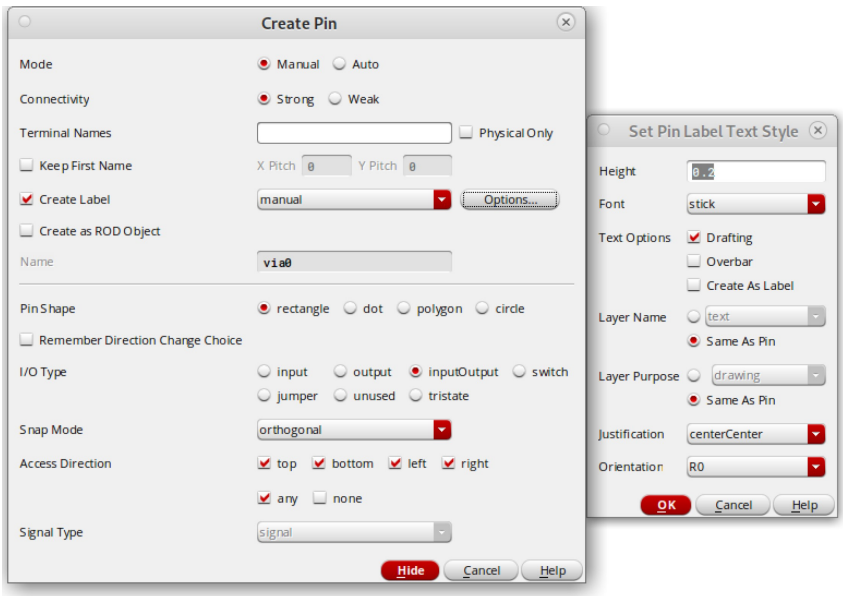
5. Create labels for **vss**, **in**, and **out**.

4.1.3 Creating Pins

Pins define the terminals that can be connected to during hierarchical layout or automatic place and route. Pins, as well, can serve to be the starting point for an automatic layout vs. schematic (LVS) comparison.

1. Click on the *metal1 pn* layer in the LSW. The pin must be drawn on the same layer as the underlying wire. The only difference is that the *pin* layer must be used for pins, instead of the *draw* layer.

2. Select **Create→Pin**, which will bring up the *Create Pin* popup dialog window. Select *Create Label* and choose *Options* to modify *Pin Label Text Style* as mentioned in below snapshot.



(a) Create Pin

3. In *Pin Label Text Style*, Important thing is to check if the *Layer Name* is selected as *Same as Pin* and *Layer Purpose* selected as *Same as Pin* . Other things such as text height and text justification can also be changed through this window.

4. Type **vdd gnd in out** as the terminal names.
   Note: You can specify any number of terminal names; Each pin you draw is assigned the next name in the *Terminal Names* field.

5. Set the *Access Direction* field accordingly. For instance, *vdd* and *vss* will have access from left and right only. (For more explanation in access direction, refer to the on-line Cadence manuals). Access direction is only necessary if you plan on using automatic routing.

6. Set the *I/O Type* to **inputOutput** for the *vdd* and *gnd* pins. Specify the *I/O Type* to **input** and **output** for the *in* and *out* pins, respectively. You might also need to change the *Access Direction* appropriately for the *in* and *out* pins.

7. select *M1 pin* layer from the LSW.

8. Draw the rectangle for the *vdd* pin coincident with the power line at the top of the inverter. The name *vdd* disappears from the *Create Pin* dialog window. The next name at the beginning of the list is *gnd*.

9. Press **ESC** to stop drawing pins and save your layout.

4.1.4 Hierarchical Layouts

Designs can typically be broken down into some kind of hierarchy, with the goal of having some basic leaf cells, and possibly nodes, repeated several times in the hierarchy. Such an approach tends to be much more efficient in terms of human layout effort and in disk size in terms of overall design. Placing instances of sub-cells may be accomplished in much the same way as described previously for schematic designs.

As an exercise, try creating an inverter chain. You may wish to modify your initial inverter layout such that the input and output of the inverter cell are on the same metal layer, which will minimize the amount of interconnect required between instances. Typically, designers will specifically engineer a cell's layout, so that some connections can be automatically made when different or similar cells are abutted against each other. Such an approach is especially useful for power rail connections and in general minimizes the amount of manual connection routing effort.

Layout instances of a cell can be made using **Create→Instance** or by using the **i** keyboard shortcut. Specify the *Library* and *Cell* that you wish to instantiate, and make sure that the *View* is set to be **layout**. Place instances in your layout by simply selecting where you'd like them to be placed. You can rotate, horizontally flip, or vertically flip the instance before you place it.

4.1.5 Display Level Control in Hierarchical Layouts

To change the amount of detail shown in a cell's view, you can specify the range of levels in the hierarchy that are to be visible. By default, only drawing layers at the current

level in the hierarchy are shown, that is level 0. Any cell instances are level 1, and cell instances inside these instances are level 2, and so on.

To change the levels of detail shown,

1. Zoom out to fit the whole layout in your layout window, using **Window→Fi**t or use the **f** keyboard shortcut.

2. Select **Design→Options→Display**, which will bring up the *Display Options* popup dialog window.

3. Change the *Display Levels* fields so that it will display layers 0 to 1. Click **Apply**.

   Note: Choosing to show levels 0 through 32, will display all levels of the hierarchy. Two keyboard shortcut keys allow you to quickly toggle from displaying level 0 only and levels 0 through 32; these shortcut keys are **SHIFT+f** and **CTRL+f**, respectively.

4.2 Layout Verification: Introduction

Ignore section 4.3 for TSMC 65nm technology node. Use section 4.6 on verification using **Calibre** tool from **Mentor** when using TSMC65nm technology.

For older technologies, simple verification tools work well to ensure that the maek layout of a design is correct, according to the "rules" for a specific technology. For newer technologies, particularly those with dimensions at 90 nanometres or smaller, the simple verification tools are not very efficient so computer run-times for verification can take prohibitively long times. Remainder of this chapter describes operation of advanced verification tool from Mentor (Calibre); the last section of this chapter describes the usage of tools from Cadence (*Assura*).

4.3 Cadence tools for layout verification

4.3.1 Design Rule Check (DRC)

Design rules are specified by foundries for each technology node; these rules roughly specify minimum dimensions and spacings of mask layout features in order to guarantee a desired manufactured die yield. Violation of design rules means that the probability of your design being manufactured correctly decreases. Indeed, some black box logic cells, which you can use for your logic designs, do have design rule violations, but these violations have been shown to achieve a desirable working die yield.

Note: Most foundries or fabrication partners, such as CMC or MOSIS or CMP, will not attempt to fabricate any chip designs which have design rule violations. In fact, CMC will only let you violate design rules if each design rule violation is explained in detail, and only if the manufacturing company approves these violations.

To run a DRC,

1. Open the layout view of the design you wish to verify.

2. Select **Calibre→Run nmDRC**, the *DRC* popup dialog window will appear.

3. Use the default values in the form and click **OK**.

   During the DRC run, the CIW will display status messages indicating the design rule currently being evaluated. Upon completion of the DRC, errors in the design are indicated on the layout via markers. The text output in the CIW indicates the violated design rules, if any, in addition to the highlighted errors in the layout.

4. To check which design rule is violated on the layout, select **Verify** → **Markers** → **Explain**. Point and click on the error marker you wish explained.

5. To check all DRC violations in sequence, select **Verify** → **Markers** → **Find** and select the *Zoom to Markers* option in the dialog window that pops up. This method allows you to use Cadence to find and zoom to each DRC error in sequence, which is useful if your layout happens to be large!

6. Correct all the errors until the layout passes DRC.

   Hint: To improve the speed of the DRC run, deselect the *Echo Commands* option in the *DRC* dialog window. This will disable the printing of DRC status messages to the CIW window and to the session log file. A summary outlining the results of the DRC run will still appear upon completion of the DRC run.

4.3.2 Layout Extraction

Layout extraction, also known as circuit extraction, is a procedure by which mask features drawn by a designer are recognized by the CAD tool in order to generate an equivalent circuit schematic representation from a layout. An extracted layout can be used to generate a circuit netlist for simulation purposes. Furthermore, parasitic capacitances and resistances due to the way transistors and other mask features are drawn can be optionally calculated during the layout extraction process.

To make layout extraction possible, technology-specific rules are supplied in order to define the devices and the interconnections between them: i.e. "polysilicon over active covered by p-plus'' may be recognized as an n-channel MOSFET in the CMOSP35 technology. Transistor device sizes, i.e. the transistors' W/L ratios, are directly calculated from recognized overlapping regions.

To extract a design,

1. Open the layout to be extracted.

2. Select **Verify**→**Extract**, the *Extractor* dialog window will appear.

   There is one optional switch which may be set for the CMOSP35 technology, parasitics, which will calculate the parasitic capacitance due to all drawn features.
   If no switch name is specified, the layout will be extracted without parasitics. This is sufficient for preliminary verification; during later stages of design, adding parasitics to the extraction will provide a more accurate netlist and hence a more accurate sign-off simulation.

3. Use the defaults in the form and click **OK**.

4. Cadence will indicate to you the extraction errors, if any, directly on the layout and in the CIW.

5. To check what rules are violated, use the same steps you used as when checking for DRC errors, **Verify→Markers→Find...**

6. Make necessary corrections on the layout and re-extract, repeating until no errors are reported.

   Note: To improve the speed of the extraction run, deselect the *Echo Commands* box in the *Extractor* dialog window. This will disable the print of status messages to the CIW, except for the final summary of violations.

7. You should also run a script to remove any floating nodes from the extracted layout; floating nodes will cause errors in Hspice and Spectre circuit simulations as they do not have a DC path to ground. Select **CMC Skill→Extract→Clean extracted view** and check the CIW for any errors or warnings.

Note: DRC and extraction may also be done in batch mode for large designs. See on-line documentation or consult VRG staff for details.

Macro Layout Extraction[1]

For designs with a large number of transistors, such as memories or custom digital chips, the extraction method described in the previous section (called flat extraction) can take several hours. A macro extraction methodology can reduce the extraction time to several seconds.

The macro extraction process tries to identify cells that have already been extracted flat and does not re-extract these cells. For example, we can create a layout for a 5-inverter ring oscillator by first creating the layout of a simple inverter and extracting it flat. Then we can place five separate instances of the inverter layout in a new (ring oscillator) layout cell view, add appropriate metal connections, and perform a macro extraction. The specific steps for this 5-inverter ring oscillator are as follows:

1. Create a layout for the basic inverter.

2. Add shape pins on all the terminals. Create the shape pins such that they cover as much of the metal as possible. This part is critical since during macro extraction, only the portions that the pin layers covers are considered as connection points. Multiple pins with the same name can be placed as long as they are electrically connected. Metal or poly that is left uncovered by a pin can potentially cause a short that will go *undetected* by the macro extraction method. Since internal nodes are not covered by a pin, there is always the possibility for an undetected short circuit when using macro extraction. Nevertheless, macro extraction is still useful for quickly extracting large cells.

3. Perform flat extraction on the basic CMOS inverter cell as described in the previous section.

4. Open the newly created extracted view of the inverter cell. Note the pin layer and the extracted metal. Close the extracted view.

[1]Special thanks to Kostas Pagiamtzis for preparing this section

5. Create a layout for a 5-inverter ring oscillator by placing five instances of the base inverter cell. Add metal to connect the base inverters in the proper configuration. Carefully add pins on all terminals as in step 2

6. In the layout cell view select **Verify→Extract**; the *Extractor* form appears. Choose **macro cell** as the *Extract Method* and click on **OK**

7. Cadence will report the extraction errors, if any, in the layout cell view and in the CIW; examine and fix the errors if necessary.

8. In the CIW, Cadence will report the number of macro cells it finds. Make sure that Cadence finds the number of macro cells that you expect. In this case we expect Cadence to find five macro cells.

9. Open the newly created extracted view of the ring oscillator. Identify the basic CMOS inverter cells and the metal connections. Close the extracted view.

10. The ring oscillator can now be used in other layout cell views with macro extraction.

If you plan to use macro extraction, it is a good idea to test it with small test layouts before applying it to a large design. If metal is left uncovered by a pin, there is a possibility that a connection/short can be made in the layout that is not detected by the macro extraction. Thus it is recommend that macro extraction is used as a way to speed up design and development, but final verification should always be performed with a full flat extraction of a design.

When performing a macro extraction it is possible to mark some instances as *flat*. Thus the extractor will descend one level into these cells. To identify an instance as *flat*, select the instance in the layout cell view, select **Edit→Properties**, and add the Property "*ivCellType*"with value *graphic*. The *graphic property* can also be set in the layout cell view and will affect all instances of the cell.

The *graphic property* only flattens a single level. Thus, to fully flatten a cell with many levels of hierarchy below it, all lower-level cells must have their *ivCellType* set to *graphic*.

Join Nets With Same Name

One option in the Extractor dialog window allows you to *Join Nets With Same Name.* This can be useful when running extractions on portions of a large design, where two wires will be joined by some as-yet un-drawn connections. ***Beware*** that this should **not** be used on final designs, as this might hide real errors.

4.3.3 Electrical Rules Check (ERC)

This section is for information only and not used in this course

ERC is another verification step which attempts to determine that no unusual features exist in the extracted view of a circuit layout. This procedure tries to locate floating devices, devices connected to only one net, floating nets short-circuits, and one-terminal nets.

1. While editing an extracted cell view, select **Verify→ERC** and an *ERC* popup dialog window appears.

2. Check the cell information on the form, and ensure that the library, cell and view names correspond to the extracted cell view you wish to check. If the fields are blank, you can click on **Sel by Cursor**, then click in the window displaying the extracted cell view.

3. If the *Rules File* is blank, type **divaERC.rul** in that space.

4. If the *Rules Library* is blank, click the check-box beside the blank field and type **cmosp35** in the field.

5. When ready, click on **Run**. You will be prompted when the ERC check is done.

To check what rules are violated:

1. Select ***Display Errors*** from the ERC form to highlight the ERC errors; select *Explain Error* and click on the marked error.

2. The object name, type and the ERC error message associated with the selected error will be displayed on the CIW or a pop-up window of your choice.

3. Make necessary corrections on the layout cell view, run DRC, re-extract and run ERC again.

4.3.4 Layout Versus Schematic (LVS) Verification

This section is for information only and not used in this course

Cadence provides additional verification tools that require additional designer input in order to verify a layout, or a schematic. Since typically a layout is drawn based on a circuit schematic, Cadence has a tool which is capable of comparing an extracted layout versus the original schematic.

1. Display the extracted and schematic views of a cell side by side (in two windows)

2. In the schematic or extracted view, select **Verify→LVS**, to bring up the Artist *LVS* popup dialog window.

3. Change the default *Run Directory* to one you desire, otherwise leave the rest of the form intact unless the schematic or extracted cellview names are not what you wanted to compare.

4. If the Rules File entry is blank, set it to **divaLVS.rul**.

5. Check all entries and click **Run**. You will be prompted when LVS is done.

6. While LVS is running, you can check the progress by viewing the log file:

7. Click the *Info* button at the bottom of the Artist *LVS* dialog window, a new *Display Run Information* popup dialog window will appear.

8. Click the *Log File* button to monitor the LVS progress.

9. After the LVS is done, click the *Output* button in the *Display Run Information* or the Artist *LVS* dialog window. The output log file of the LVS run will be shown in a text window. Near the bottom of the file you will find an indication as to whether the two circuit representations match or not.

10. To display unmatched nets, instances, parameters, etc., click the *Error Display* button in the Artist *LVS* dialog window.

11. Alternatively, you can use cross-probing to look at which devices, nodes, terminals, etc., in the extracted view correspond to the one you've selected in the schematic view, or vice versa:

    (a) Select **Verify→Probe** in the extracted or schematic view.

    (b) Set the *Probing Method* to cross probe matched.

    (c) Click anywhere in the schematic or extracted view to set it as the current window.

    (d) Click the *Add Device* or *Add Net* button and select the device or net you want to look at in the schematic.

    (e) The corresponding device or net will appear highlighted in the extracted view, if they have been found to match during LVS.
    Note: If you are using cells from the CMC-supplied black-box libraries, you must use macro LVS because there are no schematics for the black-box cells. Please refer to documentation in the files /CMC/kits/cmosp35/doc/CMOSP35lvsFlow or /CMC/kits/cmosp18/doc/CMOSP18lvsFlow for details.

4.4 Export and Import STREAM Data

STREAM (also known as GDS, GDS2, or, more properly, GDSII) format is an industry-standard language for describing mask layouts. This is a binary-data format (i.e. not generally readable as text), and as such can represent a complex design in a very compact form. Designs which are going to be submitted for fabrication through CMC or commercial foundries must be exported from Cadence in this format before submission.

Note: During translation between Cadence data format and STREAM format, a file called the *layerMap table* is used to indicate how layers are translated between the two systems. The *layerMap* file generally translates only the "drawing" layers, and not the "pin" or other layers, so during translation you might see warnings about ignored "layer-purpose" pairs. In most cases these are safe to ignore, but if you have any questions, please contact VRG staff for assistance.

Note: For some design kits, the manufacturer provides banner menu pull-down entries in the CIW to assist in the proper import or export of **GDSII** data files. If provided, please use the banner menu items instead of the procedures listed below.

4.4.1 Generating STREAM data (STREAM-out)

This section is for information only and not used in this course

1. In the CIW, select **File→Export→Stream**; the *Stream Out* popup dialog window appears.

2. Enter the following values:

Field name	Value
Library Name	TestLib (or your library name)
Top Cell Name	inverter (or your cell name)
View Name	layout
Output	StreamDB
Output File	inverter.strm
Units	micron
Error Message File:	PIPO.LOG.inverter.streamout

3. Click on the *User-Defined Data* button near the top of the *Stream Out* window and enter these values into the *Stream Out User-Defined Data* popup dialog window:

Field name	Value
Convert Pin To	geometry
Layer Map Table	/CMC/kits/cmosp35/cmosp35.strmMapTable

4. Click on the *Options* button in the *Stream Out* dialog window. Examine the various options presented; default values should work fine for now; click **OK**.

5. Click on **OK** in the *Stream Out* popup dialog window to begin the translation process.

6. Upon completion, a pop-up window will appear indicating success or failure of the conversion, and a brief summary will appear in the CIW.
   Note: Always examine the contents of the Error Message File specified in the Stream Out popup dialog window for warning and error messages.

4.4.2 Importing STREAM data (STREAM-in)

This section is for information only and not used in this course

There are some apparent inconsistencies in the way the STREAM-In procedures handle libraries which are created using the *Attach to existing techfile* method (see Section 2.9). The actual cause of this problem is that the STREAM-In procedure tries to open the technology file for the library to which the target library is attached, in **append** mode (i.e. STREAM-In tries to modify the system technology library, which could be dangerous in a multi-user environment). Because of this problem, we recommended that a library used as the target for the STREAM-In procedure be created using the *Compile a new techfile* method. After the STREAM-In procedures is complete, we can attach to an existing technology file.

1. Create a STREAM-In target library: In the CIW, select **File→New→Library**

2. In the pop-up fill form, type the name of the library to be created, **streamin**, and select the *Compile a new techfile* option. Click **OK** to continue.

3. In the *Load Technology File* popup dialog that appears, enter the full file name of the technology file to be compiled: usually in the form /CMC/kits/cmosp35/cmosp35.tf

4. Click **OK** to continue.

5. Examine the CIW for any error messages during the creation of the new library.

To import STREAM data into this library:

1. In the CIW, select **File→Import→Stream**; the *Stream In* popup dialog window appears.

2. Enter the following values:

Field name	**Value**
Input File	inverter.strm
Top Cell Name	inverter (or, leave this value empty)
Output	Opus DB(i.e. Cadence database format)
Library Name	streaminTest
ASCII Technology File Name	/CMC/kits/cmosp35/cmosp35.tf
Scale UU/DBU	0.001
Units	micron
Error Message File	PIPO.LOG.inverter.strmin

3. Click on the *User-Defined Data* button near the top dialog window and enter the following value into the *Stream In User-Defined Data* dialog window:

Field name	**Value**
Layer Map Table	/CMC/kits/cmosp35/cmosp35.strmMapTable

   Click **OK**.

4. Click on the Options button near the top of the Stream In dialog window and examine the various options; Click OK to confirm the default options.

5. Click on OK on the Stream In dialog window to begin the translation process.

6. Upon completion, a pop-up window will appear indicating success or failure of the conversion, and a brief summary will appear in the CIW.
   Note: Always examine the contents of the Error Message File specified in the Stream Out popup dialog window for warning and error messages.

7. Open the *Library Manager* using the CIW window, using **Tools → Library Manager**. In the *Library Manager* popup dialog window, select **View → Refresh** to ensure that your Cadence session has re-read the information on disk, which was created during the STREAM-in operation. Browse the streaminTest library and examine the top cell, inverter, to ensure that all mask features were transferred properly.

4.5 Export and Import CalTech Intermediate Form (CIF)

This section is for information only and not used in this course

CIF was used extensively in the early days of CAD software development at universities in fields related to VLSI design. Rather than being fully integrated into a complete design framework and sharing a common database (like modern design tools), early university-developed tools were stand-alone entities. Each of the components represented now in Cadence DFII (e.g. layout, DRC, Extraction) were individual computer programs designed to read mask layout information in the form of CIF, process the information, then output more CIF data or SPICE netlists or other information. The main advantage of CIF over industry-standard description languages is that it is readable by humans, and thus easy to parse by humans (with some basic knowledge) and early computer software. Some modern software still is not capable of reading or writing STREAM data but can read or write CIF, but these packages are becoming quite rare.

We mention CIF here, only to indicate that Cadence DFII can import and export data in CIF format. See on-line documentation for details if necessary, or contact VRG staff.

4.6 Verification using Mentor Calibre

For deep submicron technologies (i.e. those with dimensions of 90nm or smaller), the Cadence "diva" verification tools are less efficient than newer verification tools, notably Mentor Calibre and Cadence Assura. Design-kits for these newer technologies will include documentation describing the usage of these tools, but a brief desciption is included here.

4.6.1 Design Rule Check (DRC)

Design rules are specified by foundries for each technology node; these rules roughly specify minimum dimensions and spacings of mask layout features in order to guarantee a desired manufactured die yield. Violation of design rules means that the probability of your design being manufactured correctly decreases. Indeed, some black box logic cells, which you can use for your logic designs, do have design rule violations, but these violations have been shown to achieve a desirable working die yield.

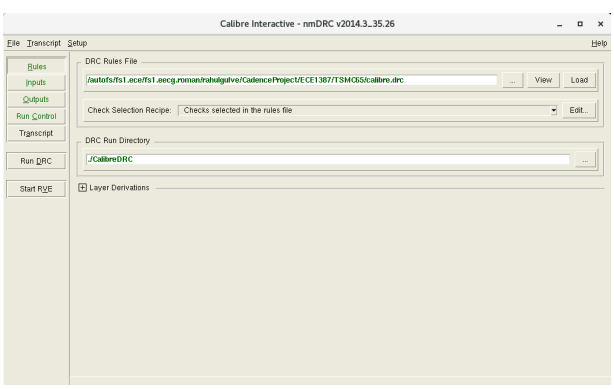
Note: Most foundries or fabrication partners, such as CMC or MOSIS or CMP, will not attempt to fabricate any chip designs which have design rule violations. In fact, CMC will only let you violate design rules if each design rule violation is explained in detail, and only if the manufacturing company approves these violations.

Note: Before we proceed we need to modify the drc rules file to be suitable for smaller designs. Copy the following file,
`/CMC/kits/tsmc_65nm/CRN65GP-OA/TN65CMSP018K3_V1.0C/Calibre/drc/calibre.drc`
to the local directory
`./calibre.drc`
And change comment the line that says
`#DEFINE FULL_CHIP // Turn on for chip lev...`
to
`//#DEFINE FULL_CHIP // Turn on for chip lev...`

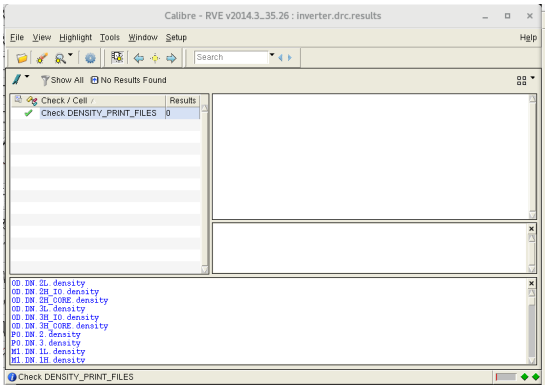
```
#DEFINE CHECK_LOW_DENSITY // Turn on for chip lev...
```
to
```
//#DEFINE CHECK_LOW_DENSITY // Turn on for chip lev...
```

To run a DRC,

1. Open the layout view of the design you wish to verify.
2. Select **Calibre** -> **Run nmDRC**, the *DRC* popup dialog window will appear.
3. If **Load Runset File** dialogue appears, press cancel
4. **Rules tab:** Make sure DRC Rules File and DRC Run Directory has been correctly filled as follows
   DRC Rules File: `.calibre.drc`
   DRC Run Directory: `./CalibreDRC`



(a) Calibre nmDRC rules setup



(b) Calibre RVE after successful DRC

5. **Inputs tab:** Make sure Top Cell, Library Name, View Name is correctly set up.
6. **Run DRC:** Click the button to run the DRC
7. After the run is complete, an RVE window opens up. It lists all the errors that should be resolved before moving on to next step

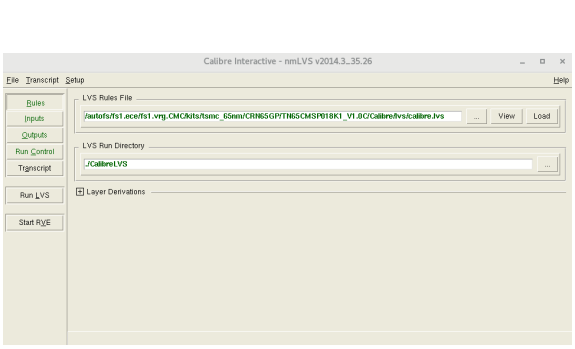
4.6.2 Layout Versus Schematic (LVS) Verification

Calibre provides additional verification tools that require additional designer input in order to verify a layout, or a schematic. Since typically a layout is drawn based on a circuit schematic, Calibre has a tool which is capable of comparing an extracted layout versus the original schematic.

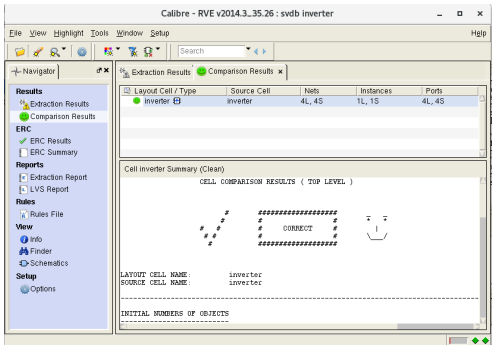
1. Display the layout and schematic views of a cell side by side (in two windows)
2. In the layout view, select **Calibre -> Run nmLVS**, to bring up the Artist *LVS* popup dialog window.
3. Open the layout view of the design you wish to verify.
4. Select **Calibre** -> **Run nmLVS**, the *LVS* popup dialog window will appear.

5. If **Load Runset File** dialogue appears, press cancel

6. **Rules tab:** Make sure LVS Rules File and LVS Run Directory has been correctly filled as follows
   LVS Rules File:
   `/CMC/kits/tsmc_65nm/CRN65GP-OA/TN65CMSP018K3_V1.0C/Calibre/lvs/calibre.lvs`
   LVS Run Directory: `./CalibreLVS`



(a) Calibre nmLVS rules setup



(b) Calibre RVE after successful LVS

7. **Inputs tab:** Make sure Top Cell, Library Name, View Name is correctly set up.

   (a) Layout: Make sure "Export from layout viewer" is checked

   (b) Netlist: Make sure "Export from schematic viewer"

8. Check all entries and click **Run**. RVE window will open when LVS is done.

4.6.3 Layout Extraction

Layout extraction, also known as circuit extraction, is a procedure by which mask features drawn by a designer are recognized by the CAD tool in order to generate an equivalent circuit schematic representation from a layout. An extracted layout can be used to generate a circuit netlist for simulation purposes. Furthermore, parasitic capacitances and resistances due to the way transistors and other mask features are drawn can be optionally calculated during the layout extraction process.

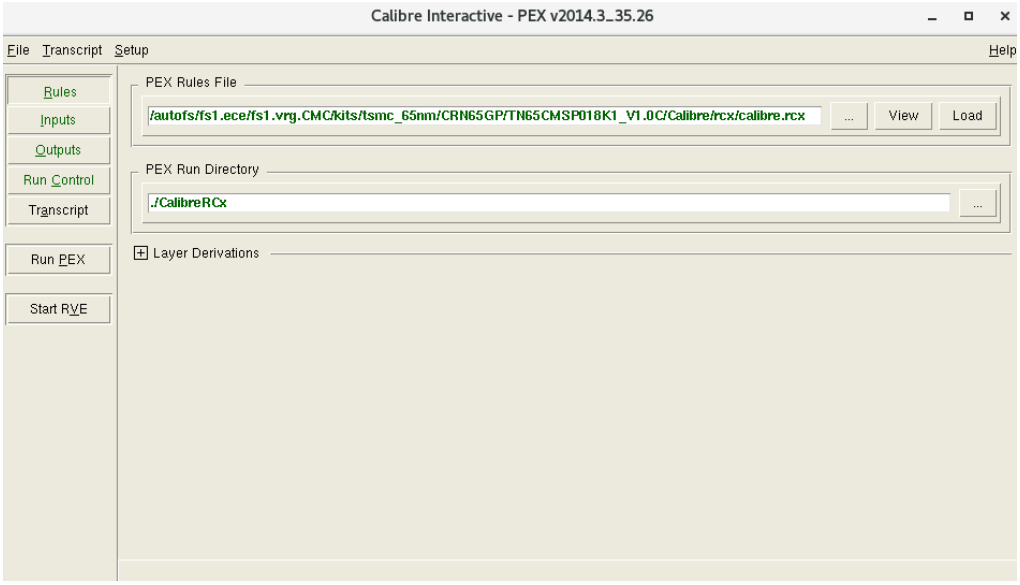
To make layout extraction possible, technology-specific rules are supplied in order to define the devices and the interconnections between them: i.e. "polysilicon over active covered by p-plus" may be recognized as an n-channel MOSFET in the CMOSP35 technology. Transistor device sizes, i.e. the transistors' W/L ratios, are directly calculated from recognized overlapping regions.

To extract a design,

1. Open the layout to be extracted.

2. Select Calibre -> Run PEX, the *Extractor* dialog window will appear.

3. If **Load Runset File** dialogue appears, press cancel



4. **Rules tab:** Make sure DRC Rules File and DRC Run Directory has been correctly filled as follows
   PEX Rules File:
   `/nfs/vrg/cmc/cmc/kits/tsmc_65nm/CRN65GP-OA/TN65CMSP018K3_V1.0C/Calibre/rcx/calibre.rcx`

   PEX Run Directory: *./CalibrePEX*

5. **Inputs tab:** Make sure Top Cell, Library Name, View Name is correctly set up.

   (a) Layout: Make sure "Export from layout viewer" is checked

   (b) Netlist: Make sure "Export from schematic viewer"

6. **Outputs tab:**

   (a) Extraction Type: Make sure following options are selected

      i. Transistor Level

         A. Choose hierarchical for big design for better speed

      ii. R+C

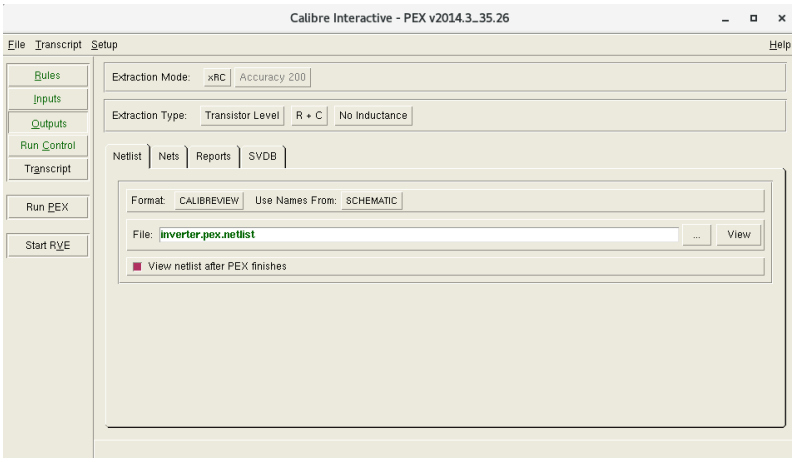
      iii. No Inductance

   (b) Netlist

      i. Format: CALIBREVIEW

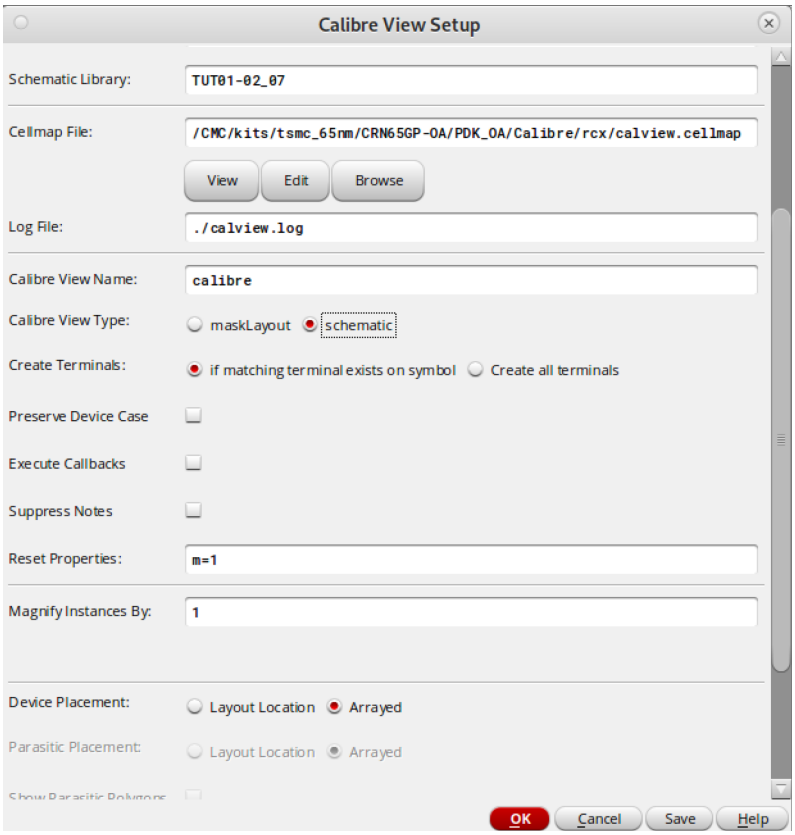
      ii. Use Names From: Schematic

7. Check all the entries and click *Run PEX*.

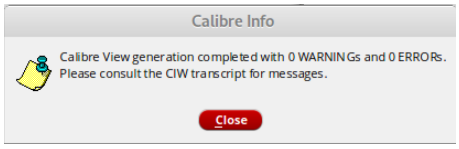


8. *Calibre View Setup* window will open after successful netlist generation.

   (a) Choose *Calibre View Type* as schematic
   (b) Choose *Device placement* as Arrayed
   (c) The other default values should work without any problems



9. press ok and the info window will open after completion.



10. The extracted view can be confirmed from the library explorer.

4.7 Cadence Assura Verification

This section is for information only and not used in this course

Some technology design-kits provide rule sets for verification using Cadence Assura, a verification package which is more advanced than the basic Cadence "*diva*". Again, an **Assura** banner menu item should become available in all Cadence layout windows if Assura is supported in the design-kit. Typical usage for a typical DRC run includes these steps:

1. Open a layout
2. Select from the banner menu **Assura→Run DRC...**
3. Review the default settings under each of the tabs in the "Run Assura DRC" window
4. Click **OK** to begin the DRC run
5. Upon completion, review and repair any errors.

Chapter 5

Cadence: Analog Simulation

Cadence provides several interfaces to simulators supported by Cadence and other vendors. In this chapter we examine the Analog Environment (also known as "Analog Artist") for simulation of schematic designs and extracted layouts using Cadence's SPECTRE and Synopsys's HSPICE simulators.

5.1 Creating a Test Bench Schematic

A test bench is the analog to a real physical laboratory. In a real lab, there are power supplies, function generators, and oscilloscopes; in our virtual test bench we use similar instruments, represented by various icons, to stimulate and observe the design under test (DUT). Typically, a test bench is used as a starting point for a schematic or extracted layout simulation.

Before you can run a Spice or Spectre simulation, you will need to create a symbol of the design to be simulated, the Device Under Test (DUT). This symbol for the DUT will then be instantiated in the test bench schematic.

1. Create a symbol for the *inverter* schematic, if one does not exist already. Open the *inverter schematic* and select **Design→Create Cellview→From Cellview** to create a symbol, as described in Section 3.3.

2. Create your test bench schematic:

   (a) Create a new schematic cell view in your *TestLib* library, give it a unique name to remind you that it is a test bench schematic for the inverter schematic: for example **tb_inverter**.

   (b) Place an instance of the **inverter** *symbol* into the test bench schematic.

   (c) Place an instance of a capacitor; you can find analog components in the *analogLib* library. Place an instance of the **cap** symbol, assign the capacitor a variable capacitance, ***cload***.

   (d) Connect the capacitor to the output of the inverter.

3. Add signal stimuli and power supplies to the **tb_inverter** schematic. Power supplies and signal stimuli can often be found in the library named after the technology; if they don't exist there, you can find them in the *analogLib* library:

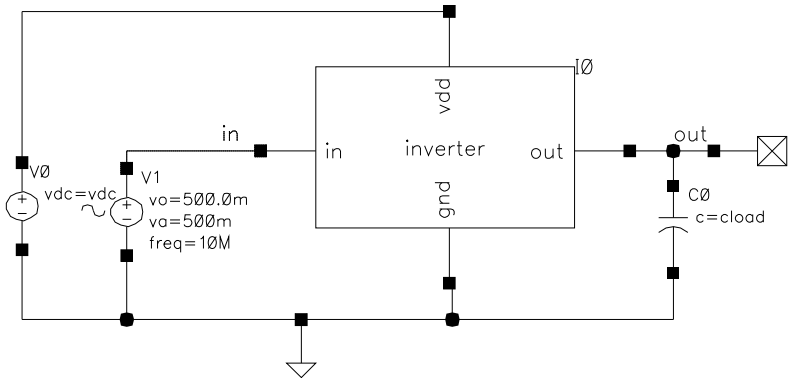


Figure 5.1: Simple inverter test bench schematic

(a) A sinusoid voltage source will be used to stimulate the input of the inverter. Place an instance of the **vsin** symbol in your schematic. Specify the values given in the following table in the *Add Instance* popup dialog window for the *vsin* symbol.

Field name	**Value**
AC magnitude	1
AC phase	0
DC voltage	500m
Offset voltage	500m
Amplitude	500m
Frequency	10M

(b) The inverter logic gate will not function without a DC power supply. Place an instance of the **vdc** symbol in your schematic. Specify the *DC voltage* to be a variable, **vdc**.

(c) Spice and Spectre simulations require that a reference node, also known as a ground node, be explicitly specified in the test bench. Place an instance of the **gnd** symbol in your schematic or create a net with the name **gnd!** -- the effect is the same.
Note: If you used global signal names for the power supplies (e.g. *vdd!* and *vss!*) then you will need to name the nets connected to the DC power supply and ground *vdd!* and *vss!*, respectively, in order to establish global power supply net connectivity.

(d) Wire up your test bench schematic so it looks like the schematic shown in Simple inverter test bench schematic

(e) Check and save your test bench.

5.2 Creating a Testbench Configuration

Cadence provides an interface to standard industry circuit simulation tools, such as Cadence *Spectre* and Synopsys *HSPICE* (among others). A testbench configuration can be created and used to set up these simulations. For example, one might set up a configuration for the simulation of a hierarchical design that uses only schematic cellviews for

all of the building blocks. Later, when some of the blocks are designed at the mask level, the configuration may be edited to include the extracted cellviews for those blocks.

There are several ways to create a testbench configuration; one method follows, but others are described in the Cadence documentation.

1. Prepare a schematic of the testbench as described in Section Creating a Test Bench Schematic

2. In the CIW, select **File→New→Cellview...** and fill in the *Library Name*, and *Cell Name*, then beside *View Name* type **config** (or, beside *Tool* select **Hierarchy-Editor**) and click **OK**. This will open two windows, one for the Cadence Hierarchy-Editor and another for setting up a New Configuration.

3. At the bottom of the New Configuration window, click **Use Template...**

4. In the *Use Template* window, select the simulator you wish to use, say **spectre**, then click **OK**,

5. Back in the *New Configuration* window, ensure that the *View* indicates **schematic**. (If it does not indicate that, then change it.). Click **OK**, then **File→Save**, then **File→Close**.

Now the testbench is ready for setting up a simulation. Open the configuration:

1. Open the *configured* testbench by selecting the **config** cellview from the *libManager* window or from the **CIW→Open...** menu. In the *Open Configuration or Top CellView* window, turn on the **yes** check-boxes for both the configuration and the top cell view, and click **OK**. The two windows should appear.

2. For now, you can ignore (or close) the *Configuration* window, but we will revisit this window in a later section, when we're ready to simulate an extracted cellview instead of the schematic cellview. Use the *Schematic* window to proceed with setting up a simulation.

5.3 Setting up a simulation using the Cadence Analog Design Environment

The Analog Design Environment (also called "ADE" or "Analog Artist" or just "Artist") provides a graphical user interface to simulators, which you can use to specify instance variables before simulation, specify variables you wish to sweep, optimize, etc. during simulation, and display the results of simulation using a GUI plotting environment and even back-annotate directly on the test bench schematic itself. To open the interface from the *Schematic* window, you can select **Tools→Analog Environment** from the command menu.

The window shown in Virtuoso Analog Environment GUI consists of four quadrants. The top-left quadrant specifies the schematic the window is tied to; each window session can only be tied to one schematic.

The left quadrant allows the designer to specify values for any Design Variables, that is any variables specified while instantiating design components.

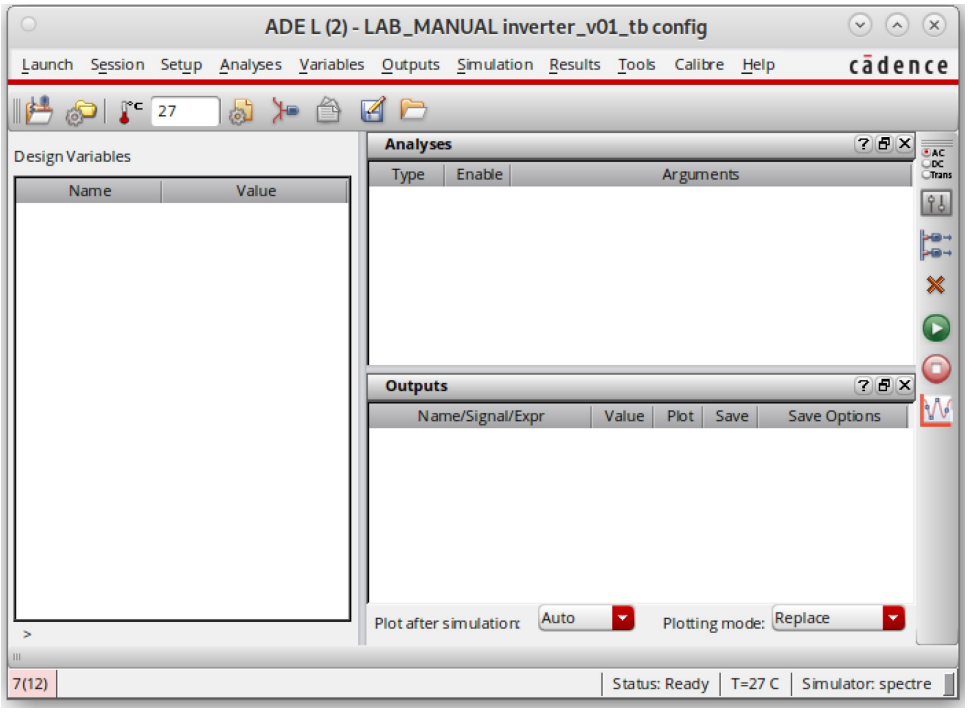


Figure 5.2: Virtuoso Analog Environment GUI

The right quadrant allows the designer to specify what type of circuit analysis to be run during simulation: DC analysis, AC analysis, transient analysis or noise analysis. You can specify several of the same types of analyses and then toggle whether or not they are enabled. You can edit variables. You can specify what types of measurements should also be plotted or displayed on the schematic automatically after the simulation completes.

The Analog Environment is well integrated within the Cadence design environment; if asked to specify instances, power supplies, or nodes by any of the popup dialog windows, you can simply point and click on the item in your schematic directly. You can also traverse the design hierarchy, starting at the test bench schematic to select nodes or items within your design hierarchy.

5.3.1 Including technology-specific device models

Technology specific device models are not automatically linked by the Analog Environment; you must specify them manually when you begin a new Analog Environment session. Hence, it's a good idea to save your session, so you won't have to redo these steps.

Note: Each Analog Environment session is specific to only one schematic. Hence, you will not be able to load session settings from another schematic.

1. Select **Setup→Simulator/Directory/Host**, which will bring up the *Choosing Simulator/Directory/Host* popup dialog window.

2. Make sure simulator is *spectre* for this particular technology.

3. The *Project Directory* field allows you to specify where all simulation files are to reside. If your design is large, then it will be beneficial to use /tmp as the root of

your project directory; /tmp is your host's local temporary storage.
Note: Using non-local network-mounted directories will severely limit the speed of your simulation for large designs. Remember that, by default, voltage values for every node in your netlist are stored during simulation; this can generate quite a lot of data during simulation!

5.4 Simulating a Design

5.4.1 Initializing Design Variables

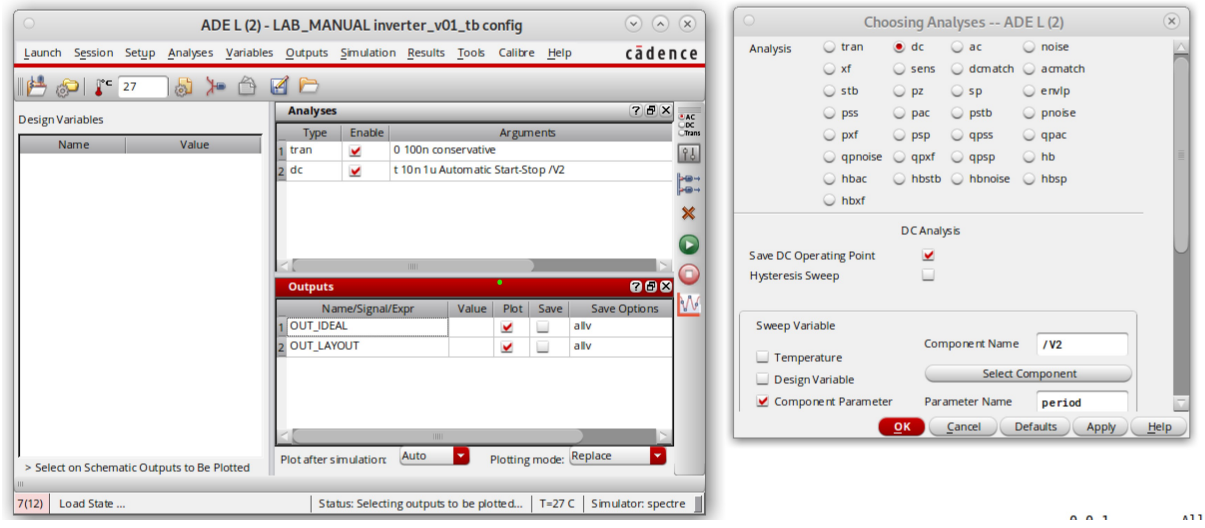
If you know what simulation variables need to be defined you can enter their values by selecting **Variables** → **Edit**, which will bring up the *Editing Design Variables* popup dialog window. You specify variables by entering a variable name into the *Name* field, a corresponding value into the *Value (Expr)* field, and clicking **Add**. Specify the following variable values:

Name	Value
cload	1p
vdc	1

Alternatively, you can use **Variables→Copy From Cellview** to "import" a list of variable names from the cellview.

5.4.2 Choosing a simulation analysis

Transient, AC and DC analysis can be run simultaneously. Units for analysis are in seconds, Volts, Amps, and Hertz depending on the analysis type.



ADEL simulation analysis and DC sweep setup

1. Select **Analyses** → **Choose**, bringing up the *Choosing Analyses* popup dialog window.

2. Select between transient, AC and DC analyses by toggling their radio buttons.

3. Specify three analyses: a transient analysis, an AC analysis, and a DC analysis with the following parameters:

Transient	Stop Time **1u**
AC	Start **1k**, Stop **100G**,
	Sweep Type **logarithmic**, **10** points per decade
DC	Save DC operating point
	Sweep Variable, **Component Parameter**

4. For a DC analysis, click on the **Select Component** button and choose the VSIN power supply in your test bench schematic. The *Source Name* field should be filled in automatically with the power supply's instance name.
   From parameter choose **dc** as a parameter and adjust the sweeping variable as:
   Start: 0
   Stop: 1
   Sweep Type: Linear
   Step Size: 0.05

5.4.3 Plotting Outputs

Voltage simulation data will be saved automatically for each node in the simulation. We will now specify which nodes we would like to have plotted when the simulation completes.

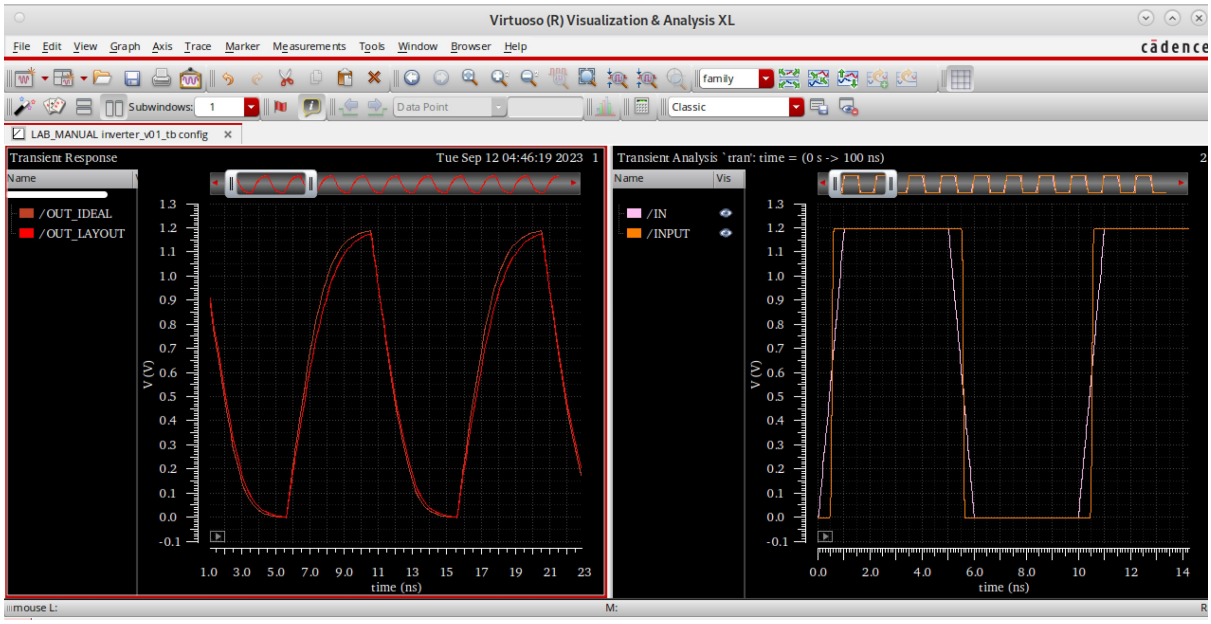
1. Select **Outputs→To Be Plotted→Select On Schematic**, which should bring your test bench schematic into focus.

2. Zoom out to fit the schematic to screen, shortcut key **f**.

3. Click on the input and output **wires** in the schematic, they will change color to indicate they have been selected. You have just chosen to plot the input and output voltage of the inverter.

4. To plot currents, click on the **terminal** you are interested in measuring the current going into or coming out of. A circle should appear around the terminal to indicate that it has been selected.
   Note: Double clicking on an instance will automatically select all terminals of the instance.

5. Press **ESC** when you have completed selecting measurements to plot.



Transient analysis plots

5.4.4 Saving Outputs

Currents, and some other outputs besides voltages, are not by default automatically saved during simulation. In order to be able to plot or view these outputs, you need to explicitly specify that they are to be saved during simulation.

Note: You can disable the automatic saving of all node voltages by selecting **Outputs** → **Save All** and toggling the *Select all node voltages* check box in the *Keep Options* popup dialog window that appears.

1. If you have already specified measurements to be plotted, simply double click on any output in the *Output* list that do not have a *yes* or *allv* (all voltages) in the *Save* column. A *Setting Outputs* popup dialog window will appear.

2. Make sure the **Saved** radio button is checked in the *Will Be* section of the dialog window.

3. Click **Change** to save the modification to the output.

You can also select outputs you wish to save by selecting **Outputs** → **To Be Saved** → **Select On Schematic** and selecting the outputs directly on the schematic itself.

Note: It does not matter if you specify the voltage outputs to be saved or not in this list. Notice the *Save* column for voltage nodes never changes from *allv*.

5.4.5 Selecting marching outputs

Sometimes, for long simulations, it is convenient to monitor how outputs are developing during the simulation, allowing for early termination of a simulation that is not proceeding as planned. This can be accomplished by selecting some outputs to be *Marching Outputs*:

1. If you have already specified measurements to be plotted, simply double click on any output in the *Output* list that do not have a *yes* or *allv* in the *March* column. A *Setting Outputs* popup dialog window will appear.

2. Make sure the **Marched** radio button is checked in the *Will Be* section of the dialog window.

3. Click **Change** to save the modification to the output.

You can also select outputs you wish to march by selecting **Outputs→To Be Marched→Select On Schematic** and selecting the outputs directly on the schematic itself.

Note: You will not see marching outputs if your simulation does not take longer than approximately five seconds to run.

5.4.6 Running a simulation

Once you have chosen the type of simulation you'd like to run, specified any outputs you'd like to plot, save or march, and specified any design variables you can proceed to run a simulation. Running a simulation is easy, simply select **Simulation → Run** or click on the **green light**.

Status messages are reported both in the simulation and the CIW windows. Marching outputs will appear in a separate window after the simulation has started. You may hear an audible tone once the simulation has completed, and you should see a status message in the simulation window indicating that the simulation was a success. Any outputs you have chosen to be plotted will be automatically shown once the simulation has completed. When ready to exit the *Analog Environment*, you can "save the state" -- the entire set-up for the simulation -- using **Session→Save State...** and click **OK.** Later, in a future design session, you can re-load that simulation state using **Session→Load State...**

5.4.7 Unsuccessful simulations

Occasionally you might see messages indicating that a simulation failed. One of the most common causes for a failure is that one or more of your MOS devices could not be matched with an appropriate Spice model. Spice *.MODEL* syntax provides a mechanism for specifying that a model is valid for a range of gate length and width; if you see messages indicating *unknown use of pch* or *nch* message, then it probably means that one of the devices has a length and width combination with no corresponding model. Verify that all of your devices fall within the ranges specified by your technology node.

5.4.8 Viewing the netlist

It is not necessary to generate or view a netlist before running a simulation as it will be generated automatically when required. However, we will create and view a netlist as an exercise; you might need to generate a netlist manually to help in debugging simulation runs, or during research projects when you wish to generate a netlist from within Cadence, but use Spice in stand-alone mode.

Lets look at a raw or informational netlist:

1. Select **Simulation→Netlist→Create Raw** in the Virtuoso *Analog Environment* dialog window.

2. Overlapping text windows will display the raw netlist information. There is netlist for the global design, and one for each level of the design hierarchy. Examine these files.

3. When done, close all netlist windows.

Let's take a look at the final simulation netlist,

1. Select **Simulation→Netlist→Create Final** in the Virtuoso *Analog Environment* dialog window.

2. A window displays the final netlist information. Observe the hierarchical structure and sub-circuit calls.

3. When done, close the netlist window.

Warning and error messages during netlisting

During netlisting, be sure to watch the CIW any warning or error messages, which should be examined carefully, and corrected. Common warning messages that may be safely ignored are "UNABLE TO OPEN FILE: pch.m" or "UNABLE TO OPEN FILE: nch.m". Some design kits provide separate model files (*.m) containing Spice or Spectre model information, but most of the design kits available under /CMC/kits have all model information in one file, so they do not use this format.

Common warning messages that need to be examined carefully include: "reference 0:pch not found", which indicates that the automatic model selection (based on the width and length specified for one of the devices in the design) has failed. To correct this problem, locate the device indicated (in this case, a p-channel MOSFET, with instance-ID 0), examine its properties and ensure that the length/width combination corresponds to one of the models specified by your technology's model files.

5.5 Displaying Output Waveforms

After a successful simulation, you can always add more voltage nodes to the list of outputs to be plotted. You can do so by selecting **Outputs→To Be Plotted→Select On Schematic** and then selecting nodes you'd like to plot directly on the schematic.

You can choose wires that are in a lower level of your design hierarchy, as well:

1. Simply **click-and-hold** the **middle mouse button** on the instance you'd like to descend into; select **Descend Read...** from the popup menu that appears, which will bring up the *Descend* popup dialog window.
   Note: You can also hover the mouse cursor over the instance you wish to descend into and use the **e** keyboard shortcut.

2. Select **Schematic** as the *View Name* from the drop down list in the *Descend* dialog window and click the **OK** button.

3. Select nets in this cellview, or keep descending the design hierarchy if you require.

4. You can return to the previous level in the design hierarchy by **pressing-and-holding** the **middle mouse button** in the schematic window and selecting **Return** or use the keyboard shortcut **CTRL+e**.

To plot the chosen outputs, select **Results→Plot Outputs** then **DC**, **Transient**, or **AC** depending on which simulation analysis results you'd like to view.

5.5.1 Printing Simulation Waveforms

In the waveform window, select **File→Print**, which will bring up the *Print* dialog window. Choose an appropriate printer or check off the **Print To File** check-box.

In the *Page Setup* tab, select *Media Size* to be **Letter**, and your preferred orientation. In the *Annotations* tab you can select what extra information you would like to have printed along with your plot.

5.6 Waveform Calculator

You can use the waveform calculator to build and display expressions containing your simulation output data; enter expression that can contain node voltages, port currents, model parameters, etc. The calculator can function either in *reverse Polish notation (RPN)* or using *infix notation*.

To start the calculator select **Tools → Calculator** in the Virtuoso *Analog Environment* dialog window. You can toggle between RPN and infix notation by selecting **Options → Set RPN** in the *Calculator* window.

Transient, AC, or DC voltage waveforms can be added to the expression field by selecting the appropriate tab (*tran*, *ac*, or *dc*); click on one of the radio buttons that appear in the tabbed list (e.g. *vt* or *it*, denoting voltage and current transients respectively); then, select a net or node from the schematic window that is brought into focus.

To plot an expression, simply click the button that looks like a squiggly line on graph paper, located under the expression field and to the left of the function window.

You can also save an expression to memory, for later recall; simply select **Memories → Table → New Memories**. This will add the current expression to the expression memory table, where you can also give a unique mnemonic for the expression. You can edit the equations stored in memory by selecting **Memories → Table → Edit**. To add a stored equation into the expression field, select **Memories → Select** and choose the mnemonic for the equation you wish to add. You can also save to disk and load from disk your saved equations by using **Memories → Load** and **Memories → Save**, respectively.

5.7 Parametric Analysis

The Virtuoso Analog Environment features a very useful tool called *parametric analysis*, which lets you sweep design variables during a simulation. You can display the simulation results in a family of curves. To perform a transient parametric analysis:

1. Select **Analyses → Choose**, and select a transient analysis *Stop Time* as **1u**. Disable all other analyses.

2. To limit the amount of simulation data that will be saved, disable the automatic saving of all node voltages. Select **Outputs →Save All** and uncheck the *Select all node voltages* check-box in the *Keep Options* popup dialog window.

3. Enable the saving of the in and out signals, by selecting **Outputs → Setup** and modifying the properties of the in and out signals appropriately.

4. Select **Tools** → **Parametric Analysis**, which will bring up the *Parametric Analysis* dialog window. The following steps will relate to this window.

5. Select **Setup** → **Pick Name for Variable** → **Sweep 1**, which will bring up a selection window with a list of design variables for your test bench.

6. Double click on **cload** to select it as the variable to be swept.

7. Enter the following values as sweep parameters for the *cload* variable:

From	To	Total Steps
10f	10p	5

8. You can see a list of design variables that will be swept, along with the values they will take during the simulation by selecting **Analysis→Show Sweep Sets→All**; click **OK** when you've done examining the list.

9. Optionally, you can add another range to sweep over:

10. Change the *Add Specification* cyclic field to **range**. Once you do so it will return back to *Add Specification*, but a second range entry will appear.

11. You can use these steps to add additional disjoint parametric ranges to be swept by the chosen design variable.

12. Select **Setup→Delete Range Specification** and double click on the added range, in the window that pops up, to delete it.

13. To run the parametric analysis, select Analysis Start. Status messages will appear in the *Parametric Analysis Window* as well as in the Virtuoso *Analog Environment* and CIW windows.

14. Explore the results in the waveform window; close the waveform window and the *Parametric Analysis* window when you're done.

5.8 Post-Layout Simulation

To simulate a physical layout, you must first run layout extraction to extract the devices, connectivity and layout parasitics. Before extraction, make sure you have all the input, output, power and ground pins on the proper pin layers at the top hierarchical level.

The steps involved in post-layout simulation are identical to those in the schematic simulation described earlier. In fact, the same testbench schematic can be used in both cases. However if you do not have a schematic for the layout, you can still create a symbol from the calibre cellview and place it in a testbench schematic as described earlier.

To simulate an extracted layout using your configured schematic test bench,

1. Open the **config** cellview for the testbench schematic, and start the Virtuoso Analog Environment.

2. In the *Hierarchy Editor* window, under the *Cell Bindings* section, find the cell for which you wish to use the **calibre** cellview in the simulation. On the line for that cell, under the column **View to Use**, right-click in that box, hover your mouse-cursor over **Set Cell View**, then select the **calibre** view. Select **File** → **Save** to save the new configuration.

3. In the *Schematic* window, select **Setup** →**Environmen**t in the Virtuoso Analog Environment window, set up the simulation or load a previously saved simulation state, and run the simulation. You can verify that this simulation used the calibre cellview by reviewing the netlist used in the simulation: in the *Analog Environment* window, select **Simulation** → **Netlist** → **Display Final** and review the comments in the netlist related to the calibre cell.

The configured schematic can be used to include cells for which you have no schematics or layouts, too. In the *Hierarchy Editor* window, you can **Set Cell View** to use a source file like a SPICE netlist or a netlist generated from advanced post-layout simulation tools.

Chapter 6

ModelSim: Digital RTL Simulation

6.1 Overview

In the next chapters, you will be working on designing your first digital custom accelerator using ASIC flow. Unlike analog block designs discussed in previous chapters, digital blocks are usually designed using digital ASIC flow tools which takes as an input Register Transfer Level (RTL) code and converts it into a full layout. RTL is usually written using High Level Descriptive languages (HDL), like System Verilog, Verilog, or VHDL. In this manual, we will be using System Verilog as our HDL language as it is one of the most commonly used language in industry nowadays. Digital ASIC flow is composed of 4 main stages (block diagram design, RTL development, Synthesis, and place and route). In the next chapters, we will discuss in detail every stage and how we can use them to get our final target layout.

6.2 Block Diagram

Block diagram design is the first stage of digital ASIC flow. In this stage, we usually sketch the architecture we are interested in developing/writing codes to using some shapes/figures to represent the main components for our design. The main target of this stage is to make sure we lay down all the main components of our design to make sure our design is modularized in a sense of each component can be verified on its own. Moreover, block diagram design helps you identify the flow of the system and the way it will work. For the sake of simplicity in this design, we will be implementing one simple component, which is 16:4 Encoder. The 16 to 4 Encoder consists of sixteen inputs Y15, Y14, ... , Y0 and four outputs A3, A2, A1, A0. At any time, only one of these 16 inputs can be '1' in order to get the respective binary code at the output.

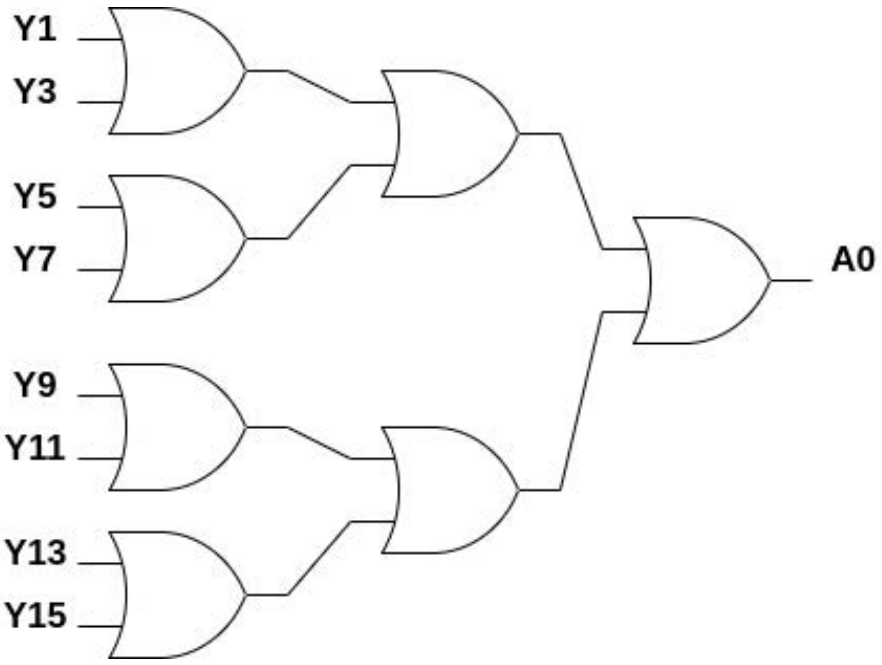
The table below shows the truth table of 16 to 4 encoder:

Y15	Y14	Y13	Y12	Y11	Y10	Y9	Y8	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	A3	A2	A1	A0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	**1**
0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	**1**	0
0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	**1**	**1**
0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	**1**	0	0
0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	**1**	0	**1**
0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	**1**	**1**	0
0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	**1**	**1**	**1**
0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	**1**	0	0	0
0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	**1**	0	0	**1**
0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	**1**	0	**1**	0
0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	**1**	0	**1**	**1**
0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	**1**	**1**	0	0
0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	**1**	0	**1**
0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	**1**	**1**	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	**1**	**1**	**1**

As you see from the truth table, A0 bit can be represented using the OR of all the odd inputs, e.g.

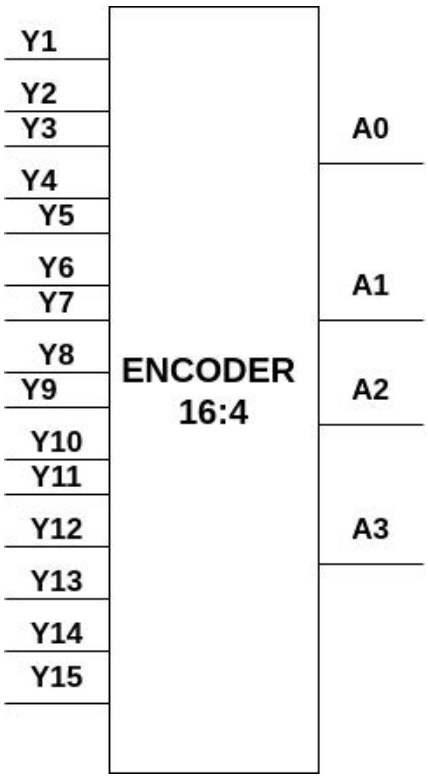
A0 = Y1 + Y3 + Y5 + Y7 + Y9 + Y11 + Y 13 + Y15

Hence, the block diagram for A0 will be the following:



Exercise: Follow the same logic as before to come up with the block diagram for the rest output bits

Finally, the digital symbol for the 16:4 encoder is as follows:



6.3 RTL Development

In this section, we will go through the second stage of the digital ASIC flow. We will focus mainly on designing our 16:4 encoder using System Verilog to model our block diagram. For learning the syntax of System Verilog, please refer to the following documentation.

`http://www.ece.uah.edu/~gaede/cpe526/SystemVerilog_3.1a.pdf`

The 16:4 encoder can be implemented using a for loop to check which bit of the input is asserted; hence, the index of this bit will represent the binary representation for the output as follows:

```
module encoder
#(
        parameter encoderWidth = 16    // Max Width of the input
 )
(
        input clk, // Clock
        input clk_en,  // Clock Enable
        input rst_n,   // Asynchronous reset active low

        input [encoderWidth-1:0] encoder_in,   // Encoder input

        // Encoder output
        output logic [$clog2(encoderWidth)-1:0] encoder_out

);
```

```
always_ff @(posedge clk)
begin
        if(~rst_n)
        begin
                 encoder_out = 'd0;
        end
        else if(clk_en)
        begin
                for (int i = 0; i < encoderWidth; i++)
                begin
                        if (encoder_in[i])
                        begin
                                encoder_out  = i;
                        end
                end
        end
end
endmodule
```

In order to create the SystemVerilog file for encoder.sv, please do the following: Go to your root directory and Run the following sequence of commands:

```
mkdir EncoderDigitalFlow
cd EncoderDigitalFlow
mkdir rtl
cd rtl
nano encoder.sv
```

and write the program shown above

Now we have the RTL ready for the 16:4 encoder. Before, we go to the next step for the digital flow, we will need to make sure we have verified our design. To do that, a Testbench is needed.

6.4 Testbench

In this section, we will create a testbench to verify the functionality of our RTL for the 16:4 encoder. The main objective of the testbench is to stimulate some known inputs that we know the output for and check the output wave to see if we receive the expected output or not. For example, if we feed **0b0000000000001000**. The output of the encoder should be **0b0100** since the 4^{th} bit from right is asserted. As shown in the next part, we have created a different test cases, like 4, 8, 15.

```
`timescale 1ps/1ps

module encoder_tb ();
```

```
localparam encoderWidth = 16;

logic clk;    // Clock
logic clk_en; // Clock Enable
logic rst_n;  // Asynchronous reset active low

logic [encoderWidth-1:0] encoder_in;        // Encoder input
logic [$clog2(encoderWidth)-1:0] encoder_out; // Encoder output

encoderTop uut (.*);

localparam h_period = 5; //duration for each bit = 5ns
localparam period = 10;

// Clock Generation
always
begin
    clk = 1'b1;
    #h_period;

    clk = 1'b0;
    #h_period;
end

initial
begin

    clk_en = 1'b1;
    rst_n = 1'b0;

    #period;
    rst_n = 1'b1;

    $display("***** TEST 1 *****");

    encoder_in = 'h0010; // 4
    #period;

    $display("Encoder in: ",encoder_in);
    $display("Encoder out: ",encoder_out);

    $display("***** TEST 2 *****");

    encoder_in = 'h0100; // 8
    #period;

    $display("Encoder in: ",encoder_in);
    $display("Encoder out: ",encoder_out);
```

```
        $display("***** TEST 3 *****");

        encoder_in = 'h8000; // 15
        #period;

        $display("Encoder in: ",encoder_in);
        $display("Encoder out: ",encoder_out);

    end
endmodule
```

In order to create the encoder_tb.sv, please do the following:

```
cd EncoderDigitalFlow/rtl
nano encoder_tb.sv
```

and use the code shown above as your testbench

Exercise: Implement a new testbench using random integer generator and automatic checking for the results instead of displaying the value and manually checking it

6.5 Simulations

Since we have developed our RTL and the Testbench for our 16:4 encoder, we are now ready to simulate the module to see if we have the correct functionality or not. We will be using ModelSim from Altera. In order to simulate your design, please do the following:

```
cd EncoderDigitalFlow/rtl
mkdir sim
cd sim
source /CMC/tools/CSHRCs/Mentor.Modelsim10.7c
vsim
```

1. Once the window opens up, you will click on

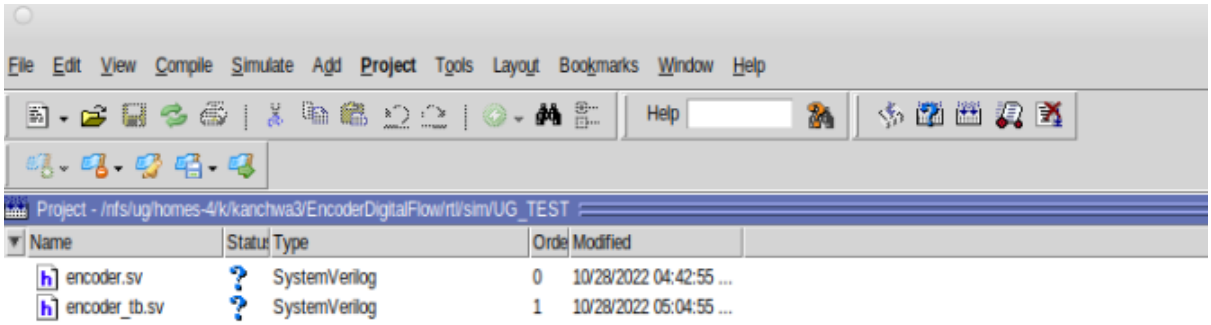
$$File \rightarrow New \rightarrow Project$$

2. Choose the current directory as the project location.

3. Type in a project name and click OK

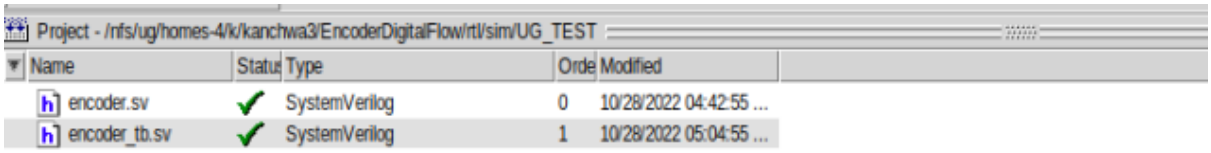
4. Click on Add existing button, and browse to add encoder.sv and encoder_tb.sv, and click OK

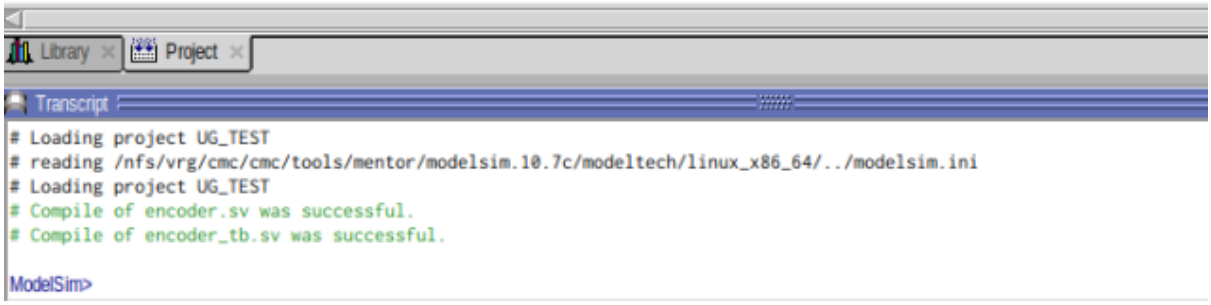
5. Your project should look like the following figure.



6. right click on encode.sv and choose compile selected

7. repeat the above step for encoder_tb.sv, and you should see the following confirmation





8. You have now compiled your RTL and Testbench, and are ready to simulate the design

9. On the left bar, go to Library, and click on work. There you should find the testbench name

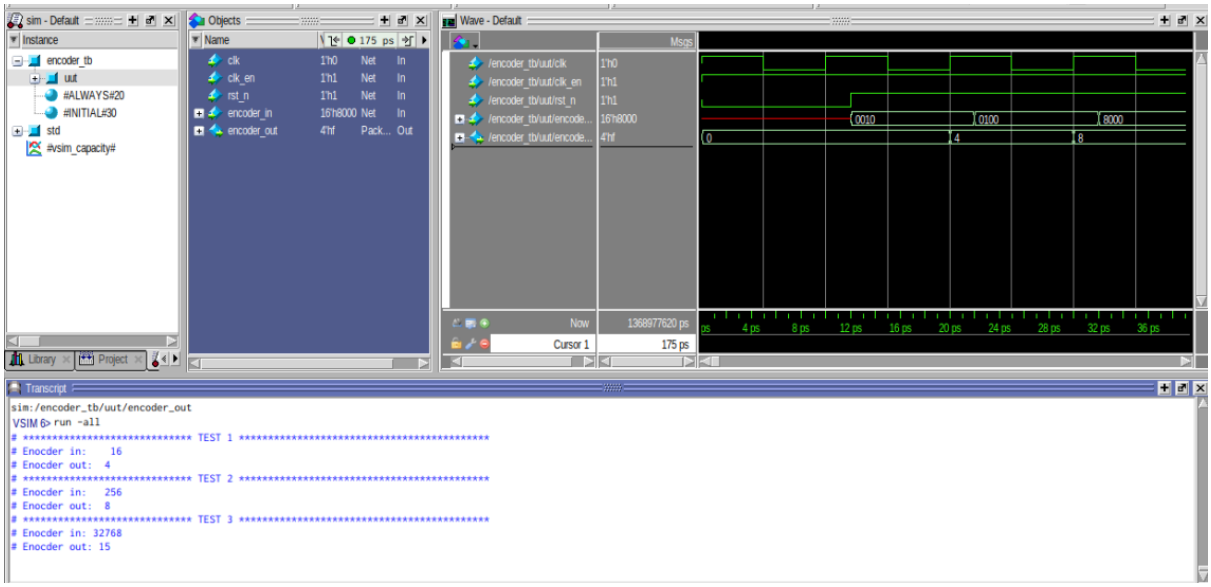
10. Right click on the testbench name and choose simulate

11. On the left bar, choose uut (Instance Name of the design, which is mentioned in the testbench), and in objects tab, select all the signals you want to see and press Ctrl+W to view the wave for it

12. Finally, run your simulation using the selected button in red, or by Clicking on

$$Simulate \rightarrow Run \rightarrow RunAll$$

13. You should see the following output



We have now verified the functionality of our model with the testbench. This verification step is called functional verification, which checks only the functionality of the design. We are ready to move the next two steps in the digital flow to get synthesis and place and route.

Chapter 7

Synopsys: Digital Synthesis

7.1 Overview

After finishing the RTL in chapter 6, in this chapter, you will be working on the third stage of the digital ASIC flow. In Synthesis stage, the RTL that was developed in chapter 6 will be mapped to digital circuit using standard cells from the technology we will be using (TSMC 65nm in our case). This linking process is happening with the help of Synopsys design vision software that allows you to link the standard cells of TSMC 65nm technology with the RTL you have developed. The output of the synthesis process is a HDL file that includes your mapped design using the standard cells (e.g. NAND, NOR, XOR, ... etc.).

7.2 TSMC65 scripts

You can find the Digital Flow scripts for TSMC65 in Quercus as a ZIP file. Download and extract this in a folder in your home directory. You will find the following structure of folders:

Folder name	Description
rtl	Includes the HDL code and testbenches for your design
FEOL	Includes the scripts used for Synthesis
BEOL	Includes the scripts used for Place and Route

7.3 Modifying Scripts

For Synthesis, there are two main files that we will be dealing with:

1. FEOL/rm_setup/common_setup.tcl

common_setup.tcl

```
puts "RM-Info: Running script [info script]\n"

##########################################################################################
# Variables common to all reference methodology scripts
# Script: common_setup.tcl
# Version: H-2013.03-SP2 (August 5, 2013)
# Copyright (C) 2007-2013 Synopsys, Inc. All rights reserved.
##########################################################################################

set MG "/guest/k/kanchwa3/EncoderDigitalFLow/TSMC65_Digital_Flow" #change this path to match your
directory

##########################################################################################

set DESIGN_NAME                   "encoderTop"  ;#  The name of the top-level design



set DESIGN_REF_DATA_PATH          "$MG/rtl"
                                       #  Absolute path prefix variable for library/design data.
                                      #  Use this variable to prefix the common absolute path
                                      #  to the common variables defined below.
                                      #  Absolute paths are mandatory for hierarchical
                                      #  reference methodology flow.

##########################################################################################
# Hierarchical Flow Design Variables
##########################################################################################

set HIERARCHICAL_DESIGNS           "" ;# List of hierarchical block design names "DesignA DesignB" ...
set HIERARCHICAL_CELLS             "" ;# List of hierarchical block cell instance names "u_DesignA
u_DesignB"
```

In this file, you have to include your design path and design name. This file also includes the technology files for the node used (i.e TSMC65nm).

2. FEOL/rm_dc_scripts/dc.tcl

dc.tcl

```
source -echo -verbose ./rm_setup/dc_setup.tcl


# Define the verification setup file for Formality
set_svf ${RESULTS_DIR}/${DCRM_SVF_OUTPUT_FILE}

#
define_design_lib WORK -path ./WORK


analyze -format sverilog ${rtl_image}
analyze -format sverilog ${RTL_SOURCE_FILES}

elaborate ${DESIGN_NAME}


write -hierarchy -format ddc -output ${RESULTS_DIR}/${DCRM_ELABORATED_DESIGN_DDC_OUTPUT_FILE}


if {[file exists [which ${DCRM_SDC_INPUT_FILE}]]} {
  puts "RM-Info: Reading SDC file [which ${DCRM_SDC_INPUT_FILE}]\n"
  read_sdc ${DCRM_SDC_INPUT_FILE}
}
if {[file exists [which ${DCRM_CONSTRAINTS_INPUT_FILE}]]} {
  puts "RM-Info: Sourcing script file [which ${DCRM_CONSTRAINTS_INPUT_FILE}]\n"
  source -echo -verbose ${DCRM_CONSTRAINTS_INPUT_FILE}
}


set ports_clock_root [filter_collection [get_attribute [get_clocks] sources] object_class==port]
create_clock -name "clk" -period 200 -waveform { 0 100  }  { clk  }
group_path -name REGOUT -to [all_outputs]
```

This is the script file that will be used in the Design Vision tool to analyze and elaborate your design. It will also define the options for synthesis, like optimization,... etc. Moreover, the scripts will write the area, timing reports in addition to the mapped verilog file.

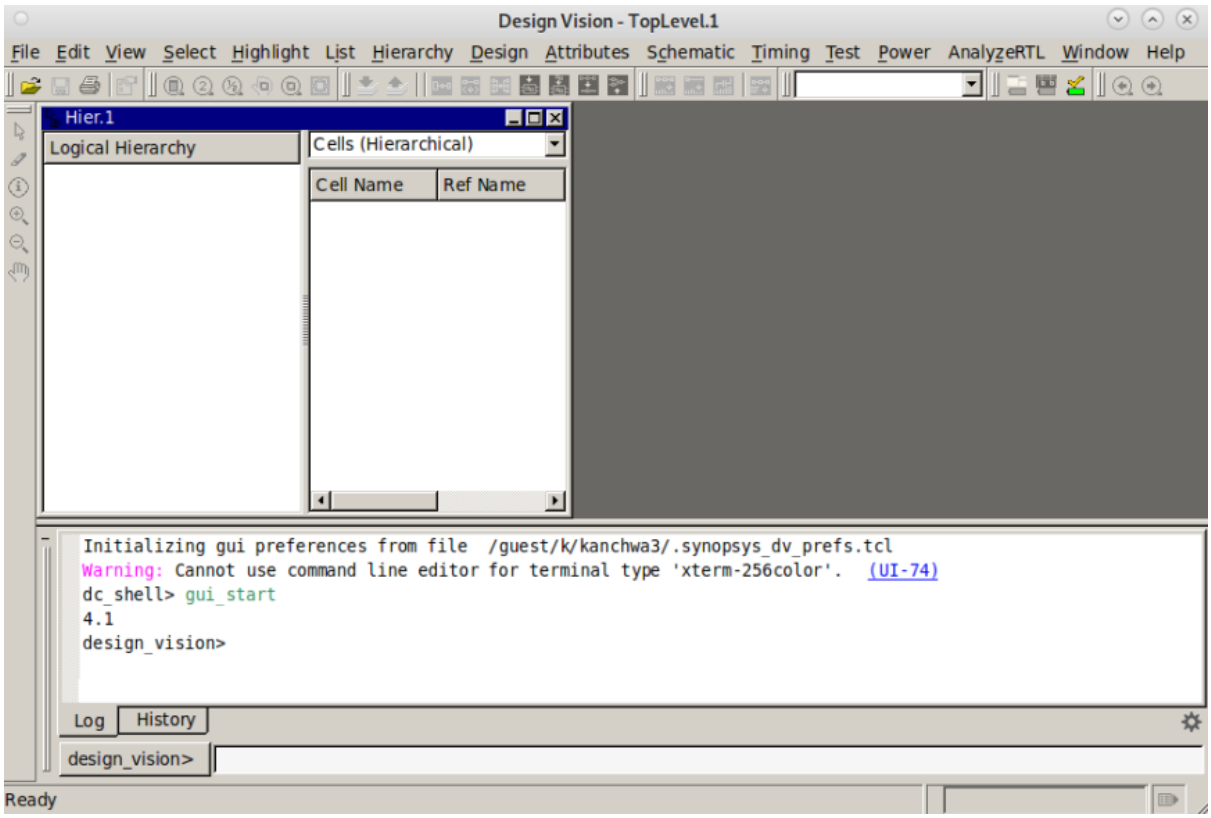
For this lab, we have modified the scripts for you. However, it is highly recommended that you go through the scripts to understand what kind of options you can modify.

7.4 Running Scripts

1. Download and extract the Digital Flow scripts folder in a folder within your home directory (if not already done so)

2. Run the following sequence of commands:

```
cd TSMC65_Digital_Flow/FEOL
source /CMC/tools/CSHRCs/Cadence.SOC
source /CMC/tools/CSHRCs/Synopsys.2017.09
source /CMC/tools/CSHRCs/Synopsys.Formality
design_vision
```



3. After design vision opens, Go to:

$$File \rightarrow ExecuteScripts$$

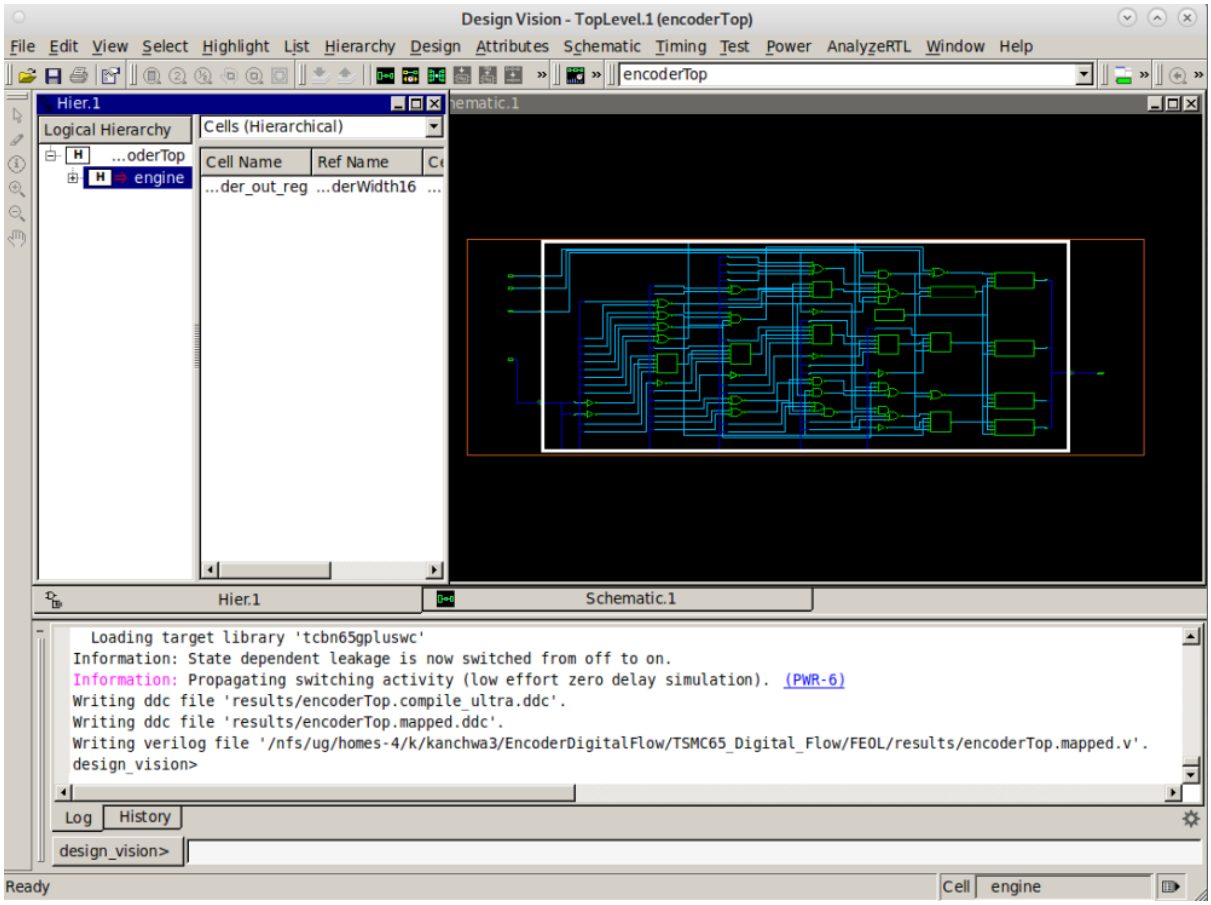
4. Then, choose "FEOL/rm_dc_scripts/dc.tcl" script

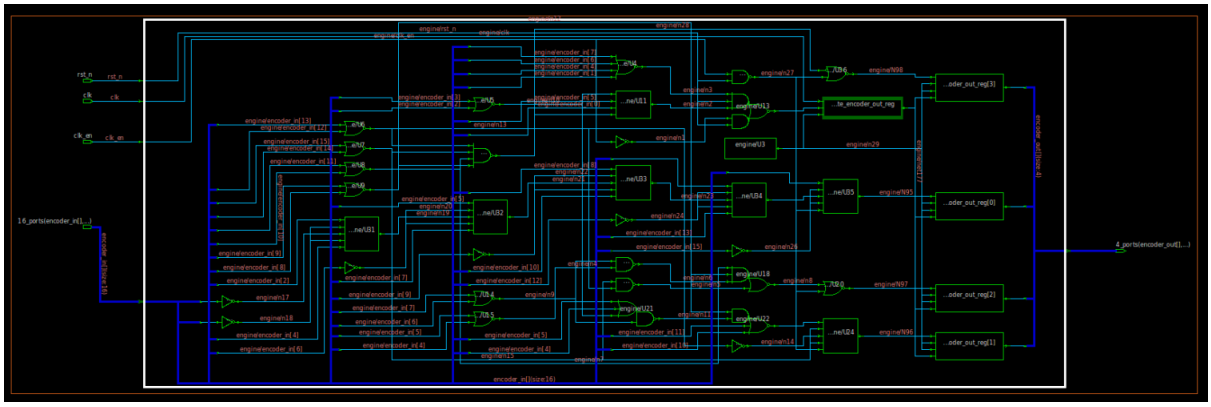
   This will perform the design synthesis and you should see the output in the terminal as well as the GUI for design vision.

5. Now after you synthesized your design, you can see the schematic for the output by going to the left bar:

   Logical Hierarchy → select design → right click → schematic view

   By clicking on the block diagram, you can go down a level to see the blocks within it.





As you can see the design is mainly composed of OR gates as expected in our block diagram in chapter 6. As previously mentioned, the scripts will also generate the timing reports and mapped verilog file for the design in the following paths:

File	Description
FEOL/results/encoderTop.mapped.v	This is the mapped file of the design using standard cells
FEOL/results/encoderTop.mapped.sdc	This is the design constraints file
FEOL/reports/encoderTop.mapped.timing.rpt	Timing report to know the max frequency
FEOL/reports/encoderTop.mapped.area.rpt	Area report to know the expected area
FEOL/reports/encoderTop.mapped.power.rpt	Power report to know the power dissipation of the block

encoderTop.mapped.timing.rpt

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

Operating Conditions: BCCOM   Library: tcbn65gplusbc
Wire Load Model Mode: segmented

  Startpoint: engine/encoder_out_reg[3]
              (rising edge-triggered flip-flop clocked by clk)
  Endpoint: encoder_out[3]
            (output port)
  Path Group: (none)
  Path Type: max

  Des/Clust/Port     Wire Load Model       Library
  ------------------------------------------------
  SNPS_CLOCK_GATE_HIGH_encoder_encoderWidth16 ZeroWireload tcbn65gplusbc
  encoder_encoderWidth16 ZeroWireload      tcbn65gplusbc
  encoderTop         ZeroWireload          tcbn65gplusbc

Attributes:
    d - dont_touch
    u - dont_use
   mo - map_only
   so - size_only
    i - ideal_net or ideal_network
  inf - infeasible path

  Point                                   Fanout     Trans      Incr       Path      Attributes
  ---------------------------------------------------------------------------------------------
  engine/encoder_out_reg[3]/CP (SDFQD0)                  0.00      0.00       0.00 r
  engine/encoder_out_reg[3]/Q (SDFQD0)                   0.01      0.04       0.04 r
  engine/encoder_out[3] (net)                 1                    0.00       0.04 r
  engine/encoder_out[3] (encoder_encoderWidth16)                   0.00       0.04 r
  encoder_out[3] (net)                                             0.00       0.04 r
  encoder_out[3] (out)                                   0.01      0.00       0.04 r
  data arrival time                                                           0.04
  ---------------------------------------------------------------------------------------------
  (Path is unconstrained)


1
```

Timing report: Unit is **ns**

As shown here, the critical path has 0.05 ns delay. Therefore, the design has max frequency of 20 GHZ (as it is just a simple design)

encoderTop.mapped.area.rpt

```
1
****************************************
Report : area
Design : encoderTop
Version: N-2017.09
Date   : Thu Sep 14 12:00:42 2023
****************************************

Library(s) Used:

    tcbn65gplusbc (File: /nfs/vrg/cmc/cmc/kits/tsmc_65nm_libs/OA_libs/-
tcbn65gplus/tcbn65gplus_200a/TSMCHOME/digital/Front_End/timing_power_noise/NLDM/-
tcbn65gplus_140b/tcbn65gplusbc.db)

Number of ports:                           50
Number of nets:                            85
Number of cells:                           41
Number of combinational cells:             34
Number of sequential cells:                 5
Number of macros/black boxes:               0
Number of buf/inv:                          8
Number of references:                       1

Combinational area:                63.360002
Buf/Inv area:                       8.640000
Noncombinational area:             41.040001
Macro/Black Box area:               0.000000
Net Interconnect area:      undefined  (Wire load has zero net area)

Total cell area:                  104.400003
Total area:                 undefined
1
```

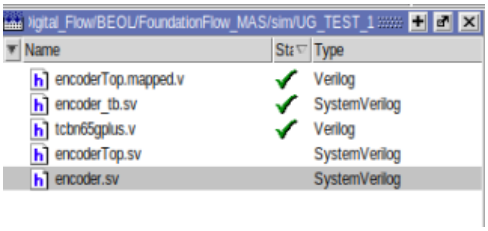
area report: Unit is μm^2

7.5 Post-Synthesis simulations

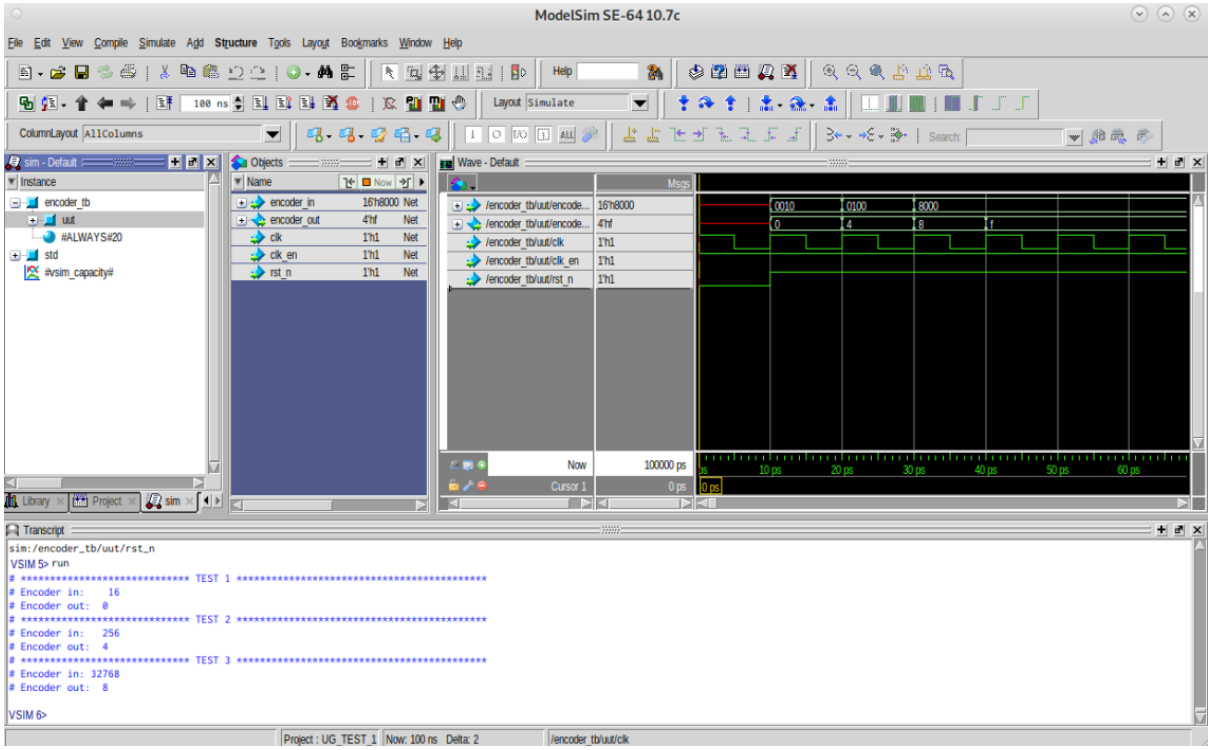
In this section, you will be simulating the design post-synthesis to make sure the synthesis process went well and the mapped design has the same functionality for the pre-synthesis RTL. In order to simulate the design, you will need to redo the steps in Chapter 6, section 5. An additional step you need to do is to add the technology file since the encoderTop.mapped.sv is now composed of standard cells that we need to define for ModelSim before simulations. Hence, you will need to compile the technology file in the following path:

`BEOL/FoundationFlow_MAS/sim/tcbn65gplus.v`

Your ModelSim after adding the technology file should look like as follows:



Now, after simulating the encoderTop.mapped.v file, you should get the following result, which is similar to the one you got from the functional simulations



Note: Make sure the name of the design inside the testbench is matching with the name of the module in the encoderTop.mapped.v file.

Now, that we have finished the synthesis process, we are now ready to move forward to the last step in digital ASIC design flow, which is place and route. In the next chapter, we will talk in detail about it.

Chapter 8

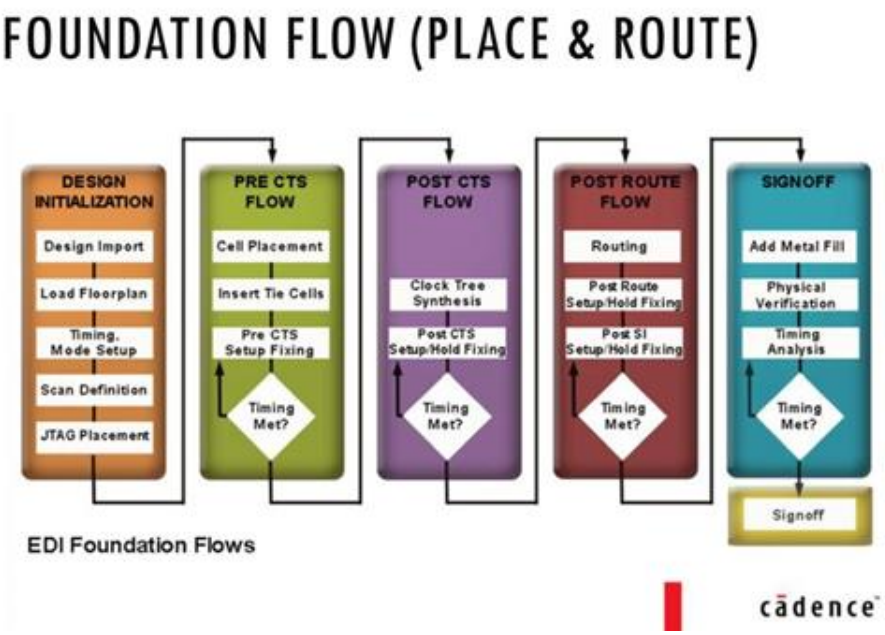
Cadence: Place and Route

8.1 Overview

In this chapter, you will be working on the final stage of the digital ASIC flow, which is place and route. In this stage, you will convert the standard cells into layout in order to generate the GDS that can be sent to fabrication. We will be using Cadence Innovus to help us transform the mapped output file from synthesis into layout. The output of this stage will be only the layout for the IP we have been working on which is the 16:4 encoder. Usually, when you are working on a large chip, you will need to integrate the IP we have designed with other blocks. However, this step requires us to import the output of Cadence Innovus into Cadence Virtuoso to make the custom layout wiring (which is not the scope of the chapter, and will be covered in tutorials).

8.2 Cadence Foundation Flow

The scripts for place and route using Cadence Innovus are created using the Foundation Flow from Cadence. The Cadence Innovus foundation flow is used to facilitate the place and route process using different TCL scripts. As shown in the next figure, these scripts are used to complete the floor planning, cells placement, routing of the cells, and optimization and sign off based on the constraint files, technology libraries, and the LEF files that describe the layout geometries.



The following table shows the structure of the folders for Place and Route:

Folder	Description
BEOL/FoundationFlow_MAS	This folder includes the foundation flow scripts starting from initialization to sign off that will generate the GDSII file
BEOL/SYSTEM_LEVEL_SCRIPTS	This folder includes the design specifications, such as the floor plan, pin placement, and power placement

8.3 Scripts

Floor planning is the first step in place and route. In this step, you define the boundaries and dimensions for your IP (e.g. width and height). Moreover, if you have SRAMs, then in this part, you also need to define where the SRAM will be placed. You choose the dimensions based on the expected area from the synthesis and the available space in the chip.

You should always aim for 70% or less design density during floor planning to allow space for routing in the later stages. This also ensures your design makes optimal use of the available area.

encoderTop_floorplan.tcl

```
set io_spacing 15
set sram_padding 2
set sram_xboundary  [expr $io_spacing + $sram_padding]
set sram_yboundary  [expr $io_spacing + $sram_padding]
set sram_r_boundary $io_spacing

# Show all the top level blocks before we proceed
setDrawView fplan

# Floorplan design area x-y dimension

set fp_x 650
set fp_y 150
floorPlan -site core -s $fp_x $fp_y $io_spacing $io_spacing $io_spacing
$io_spacing
```

Pin placement is the second step in place and route. In this step, you will be placing the pins (inputs and outputs) for your IP around the boundaries. It is important to note that pin placement is an important step, because if the pins placement is done properly, you will make the place and route process faster and it will likely be free from DRC and LVS errors.

encoderTop_place_pins.tcl

```
# TOP pin placement
########################################

editPin -side "TOP" \
-layer $pin_layer_h \
-spreadType start \
-start [expr $start_dist_top + $fplan_io_spacing ] [expr $fplan_height + 2 *
$fplan_io_spacing] \
-spreadDirection clockwise \
-pin $clk_rst \
-spacing $top_spacing \
-snap USERGRID \
-fixedPin 1

editPin -side "TOP" \
-layer $pin_layer_h \
-spreadType start \
-start [expr $start_dist_top + $fplan_io_spacing + $top_spacing*4] [expr
$fplan_height + 2 * $fplan_io_spacing] \
-spreadDirection clockwise \
-pin $encoderIn \
-spacing $top_spacing \
-snap USERGRID \
-fixedPin 1


########################################
# BOTTOM pin placement
########################################


editPin -side "BOTTOM" \
-layer $pin_layer_v \
-spreadType start \
-start [expr $start_dist_bottom  + $mask_Starting ] 0 \
-spreadDirection counterclockwise \
-pin $encoderOut \
-spacing $bottom_spacing \
-snap USERGRID \
-fixedPin 1
```

Power grid is the third step for the place and route. In this step, you will be placing the power grid, which will be used for the power connection for the standard cells and SRAMs if your design includes SRAMs.

encoderTop_power_grid.tcl

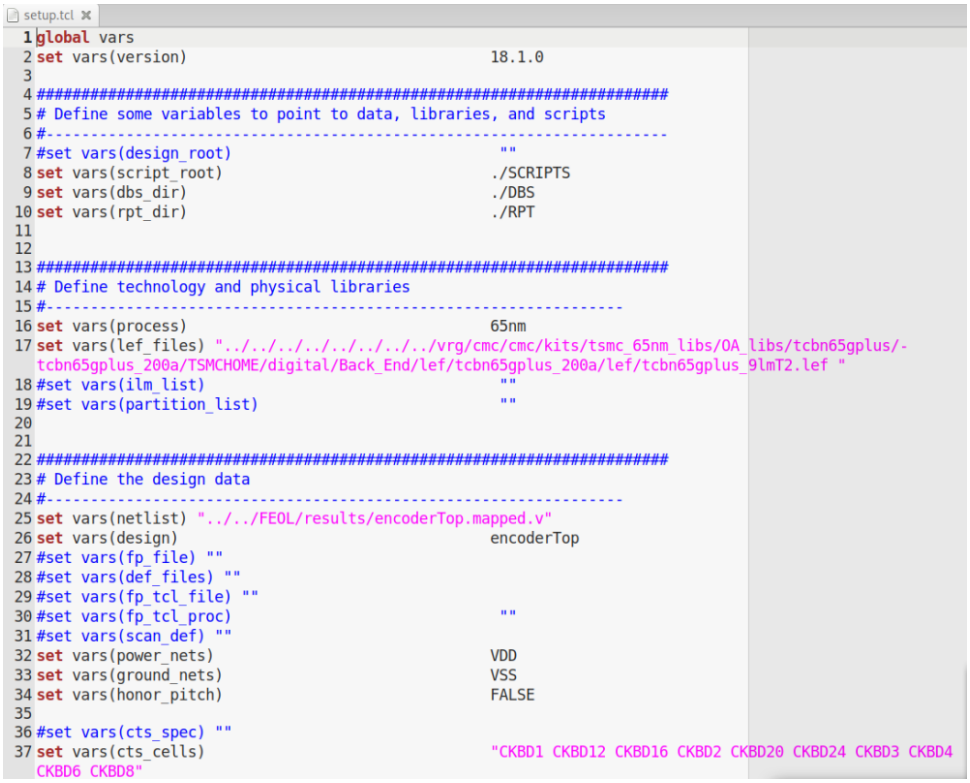
```
1
2
3 set power_nets [list  VSS VDD]
4
5
6 ########################
7 # Place Core Ring
8 ########################
9 setAddStripeMode -stacked_via_top_layer M8
10 setAddStripeMode -stacked_via_bottom_layer M1
11 addRing -skip_via_on_wire_shape Noshape -skip_via_on_pin Standardcell -type
   core_rings -jog_distance 0.2 -threshold 0.2 -nets $power_nets -follow core -
   layer {bottom M3 top M3 right M4 left M4} -width 5 -spacing 3 -offset 1.8
12
13
14 ##############################
15 # Add Power Grid
16 ##############################
17
18 # M8
19 setAddStripeMode -stacked_via_top_layer M8
20 setAddStripeMode -stacked_via_bottom_layer M4
21 addStripe -skip_via_on_wire_shape Noshape -extend_to design_boundary -
   block_ring_top_layer_limit M8 -max_same_layer_jog_length 6 -
   padcore_ring_bottom_layer_limit M7 -set_to_set_distance 20 -skip_via_on_pin
   Standardcell -padcore_ring_top_layer_limit M8 -spacing 6 -merge_stripes_value
   0.1 -layer M8 -block_ring_bottom_layer_limit M7 -width 4 -nets {VDD VSS}
22
23
24 # M7
25 setAddStripeMode -stacked_via_top_layer M8
26 setAddStripeMode -stacked_via_bottom_layer M4
27 addStripe -skip_via_on_wire_shape Noshape -extend_to design_boundary -
   block_ring_top_layer_limit M8 -max_same_layer_jog_length 6 -
   padcore_ring_bottom_layer_limit M7 -set_to_set_distance 20 -skip_via_on_pin
   Standardcell -padcore_ring_top_layer_limit M8 -spacing 6 -merge_stripes_value
   0.1 -direction horizontal -layer M7 -block_ring_bottom_layer_limit M7 -width
   4 -nets {VDD VSS}
```

The following table shows the scripts responsible for the steps listed above:

File	Description
BEOL/SYSTEM_LEVEL_SCRIPTS/encoderTop_floorplan.tcl	This is the script used for floor planning during place and route
BEOL/SYSTEM_LEVEL_SCRIPTS/encoderTop_place_pins.tcl	This is the script used for pin placement in the place and route
BEOL/SYSTEM_LEVEL_SCRIPTS/encoderTop_power_grid.tcl	This is the script used for power grid in the place and route

Back to the foundation flow, the foundation flow as mentioned is composed of different steps, starting from initialization till sign off. There are multiple scripts used to control the parameters for the design, for example the name of the module, the path for the design constraints and the mapped file from synthesis. These parameters can be adjusted in the file:

BEOL/FoundationFlow_MAS/setup.tcl



```
setup.tcl
1 global vars
2 set vars(version)                                  18.1.0
3
4 ######################################################################
5 # Define some variables to point to data, libraries, and scripts
6 #---------------------------------------------------------------------
7 #set vars(design_root)                              ""
8 set vars(script_root)                              ./SCRIPTS
9 set vars(dbs_dir)                                  ./DBS
10 set vars(rpt_dir)                                 ./RPT
11
12
13 ######################################################################
14 # Define technology and physical libraries
15 #---------------------------------------------------------------
16 set vars(process)                                  65nm
17 set vars(lef_files) "../../../../../../../../vrg/cmc/cmc/kits/tsmc_65nm_libs/OA_libs/tcbn65gplus/-
   tcbn65gplus_200a/TSMCHOME/digital/Back_End/lef/tcbn65gplus_200a/lef/tcbn65gplus_9lmT2.lef "
18 #set vars(ilm_list)                               ""
19 #set vars(partition_list)                         ""
20
21
22 ######################################################################
23 # Define the design data
24 #---------------------------------------------------------------
25 set vars(netlist) "../../FEOL/results/encoderTop.mapped.v"
26 set vars(design)                                   encoderTop
27 #set vars(fp_file) ""
28 #set vars(def_files) ""
29 #set vars(fp_tcl_file) ""
30 #set vars(fp_tcl_proc)                             ""
31 #set vars(scan_def) ""
32 set vars(power_nets)                               VDD
33 set vars(ground_nets)                              VSS
34 set vars(honor_pitch)                              FALSE
35
36 #set vars(cts_spec) ""
37 set vars(cts_cells)                                "CKBD1 CKBD12 CKBD16 CKBD2 CKBD20 CKBD24 CKBD3 CKBD4
   CKBD6 CKBD8"
```

Finally, the foundation flow scripts are in:

BEOL/FoundationFlow_MAS/PLUG/INNOVUS/

Note: These scripts are configured for the 16:4 encoder. However, you might need to visit and modify some paths and name of files to ensure these run without any errors. You can also update these if you want to use the flow for another IP.

The post_place.tcl and post_signoff.tcl files have outpaths that need to be updated to your directory.

8.4 Running Scripts

We have provided you with a make file that will run all the TCL files for the foundation flow one by one. Ensure the paths in these scripts are updated to your directory. To run these scripts, please do the following:

```
cd BEOL/FoundationFlow_MAS
source /CMC/tools/CSHRCs/Cadence
source /CMC/tools/CSHRCs/Cadence.SOC
source /CMC/tools/CSHRCs/Cadence.EXT
source /CMC/tools/CSHRCs/Cadence.INNOVUS.18
source /CMC/tools/CSHRCs/Cadence.INCISIVE.15

clean.tcl

make all
```

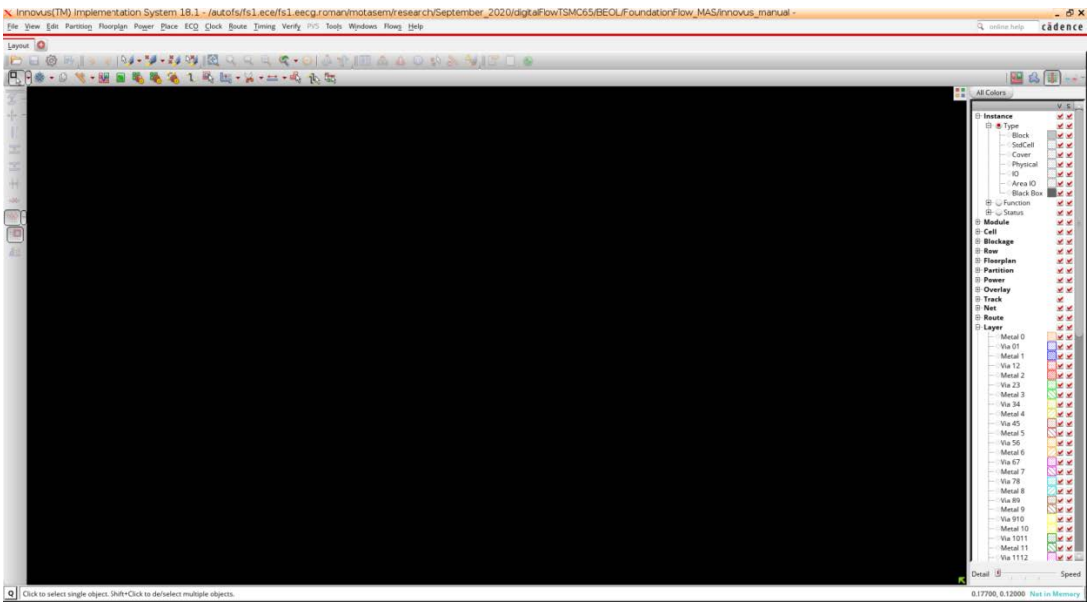
8.5 Output Files

The output files are divided into 2 main folders:

Folder	Description
BEOL/FoundationFlow_MAS/OUTPUT_FILES	This folder includes the final GDS file in addition to the verilog file that will be used for post-place and route verification
BEOL/FoundationFlow_MAS/DBS	This folder includes all Innovus data for every stage in the foundation flow (shows how the final GDS was progressing)

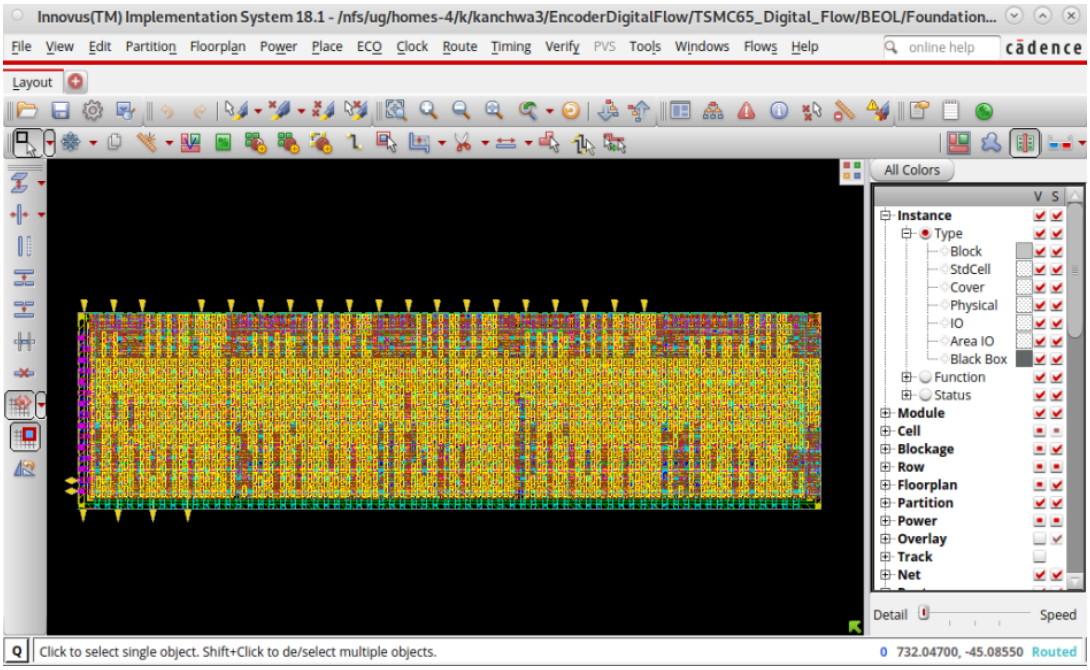
In order to see the layout of the sign off step, which is the last step in the foundation flow, you can do the following:

```
cd BEOL/FoundationFlow_MAS
innovus
```

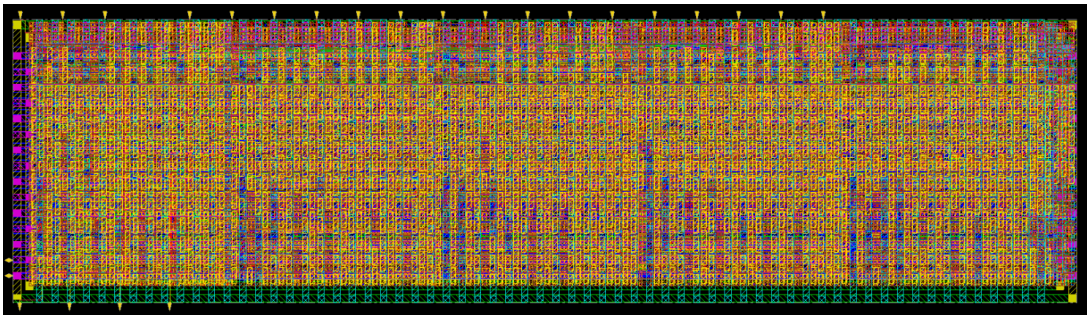


Once Innovus starts up, type the following in the terminal:

```
source DBS/signoff.enc
```

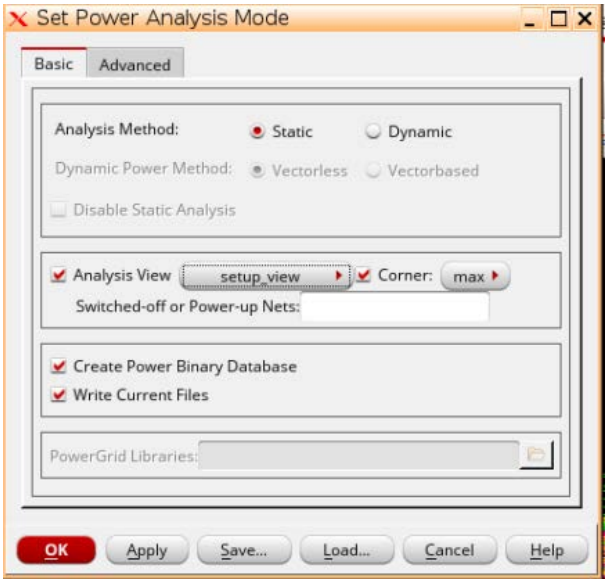


As shown, this is the final layout for the 16:4 encoder. The dimensions, power ring, and pin placements are based on the scripts we modified in section 3.

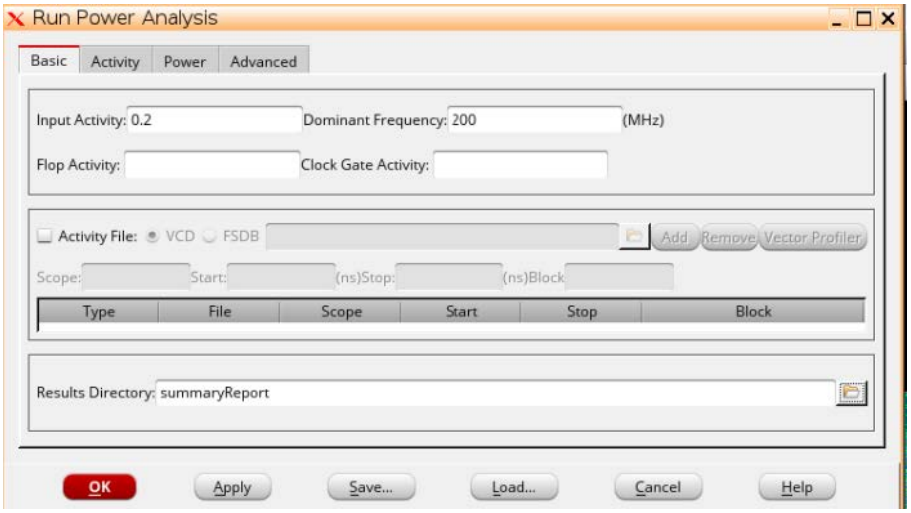


For power report, perform the following steps:

$$Power \rightarrow Power\ Analysis \rightarrow setup$$



Power → Power Analysis → run → choose frequency range and output location → ok



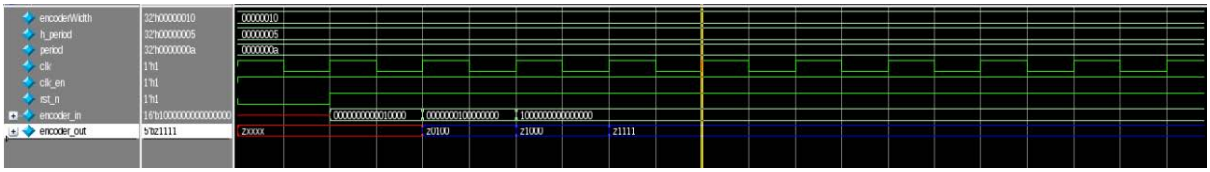
8.6 Post - place and route simulations

In this section, you will be simulating the design post-place and route to make sure the place and route process went well and the mapped design has the same functionality for the pre/post-synthesis RTL. In order to simulate the design, you will need to redo the steps in Chapter 7, section 6.

The post place and route verilog file can be found in:

`BEOL/FoundationFlow_MAS/OUTPUT_FILES/encoderTop.layout.v`

Now, create a new project as mentioned in chapter 6 and use the encoderTop.layout.v in addition to the technology file and testbench to verify your functionality. You should get the following, which is matching our functional simulations



Note: Make sure the name of the design inside the testbench is matching with the name of the module in the encoderTop.layout.v file.

Now, we have the GDS for the 16:4 encoder and we have verified it through all the stages for digital ASIC flow. You can import the GDS to cadence virtuous to do wiring connections to other IPs.

Exercise: Now that we have implemented the 16:4 encoder, choose another simple design (e.g. multiplier, adder, ... etc.) and try the same steps to familiarize with it.