Chapter 2

Cadence ICFB

CADENCE is a large company that offers a dizzying array of software for Electronic Design Automation (EDA) or Computer Aided Design (CAD) applications. Cadence CAD software is generally targeted at the design of electrical circuits, both digital and analog, and extending from extremely low-level VLSI design to the design of circuit boards for large systems. This book is primarily interested in digital integrated circuit (IC) design so we'll look primarily at those tools from the Cadence suite.

The custom design tutorials are a good starting point for custom analog IC design too

2.1 Cadence Design Framework

Many of the digital IC design tools from Cadence are grouped under a framework called Design Framework II (dfII). The dfII environment integrates a variety of design activities including schematic capture (Composer), simulation (Verilog-XL or NC_Verilog), layout design (Virtuoso and Virtuoso-XL, design rule checking (DRC) (Diva and Assura), layout versus schematic checking (LVS) (Diva and Assura), and abstract generation for standard cell generation (Abstract). These are all individual programs that perform a piece of the digital IC design process, but are all accessible (to a greater or lesser extent) through the dfII framework and the dfII user interface. Note that many of these programs were developed by separate companies that have been acquired by Cadence and folded into the dfII framework after that acquisition. Thus, some integrate better than others!

As we'll see, though, there are some pieces of the Cadence tool flow that are not linked into the dfII framework. Most notably place and route of standard cells with SOC Encounter, connection of large blocks with ICC Chip Assembly Router (CAR) and Verilog synthesis with BuildGates are done in separate programs with separate interfaces.

However, we'll start with the dfII tools in this tool flow, so we'll need to start up the dfII framework. The executable in the Cadence tool suite that starts up this framework is called icfb which stands for Integrated Circuit Front to Back design. If you were to set up your search path so that the Cadence tools were on your path, and execute the `icfb` command you would see the dfII framework start up.

Unfortunately, this wouldn't help you much! It turns out that having the tool framework is only half the battle. You also need detailed technology information about the devices you want to use for your design. This detailed design information includes technology information about the IC process that you are using and libraries of transistors, gates, or larger modules that you can use to build your circuits. This information includes many files of detailed (and somewhat inscrutable) information, and does not come from Cadence. Instead, it comes from the vendor of the IC process and from the vendor of the gate and module cells that you are using in your design. This collection design information is typically called a "Cadence Design Kit" or CDK.

We're using NCSU CDK v1.5

For this book we will use technology information for IC processes supported by the MOSIS chip fabrication service. This information has been assembled into a CDK by the good folks at North Carolina State University (NCSU). The NCSU CDK has detailed techology information for all the processes currently offered through MOSIS. These processes are available either in "vendor" rules which have have the actual specifics of the technology as offered by the vendor, or through abstracted rules known as Scalable CMOS or SCMOS rules. The SCMOS are scalable in the sense that a design done in the SCMOS rules should, theoretically, be useable in any of the MOSIS processes simply by changing a scalaing parameter. That means that the SCMOS rules are a little conservative compared to some of the vendor rules because they have to work for all the different vendors.

Of course, it's not quite that simple because as design features get smaller and smaller the IC structures don't scale at the same rate. But, it works pretty well. To handle the differences required by smaller geometry processes MOSIS has a number of modifiers to the SCMOS rules (SCMOS for "generic" SCMOS, SCMOS_SUBM for submicron processes, and SCMOS_DEEP for even smaller processes). For this class we'll be using the SCMOS_SUBM rules which will then be fabricated on the AMI C5N 0.5μ CMOS process.

We're using the SCMOS V8.0 rules.

But, that's getting ahead of ourselves a little bit. The important thing for now is that without the NCSU CDK, we won't have any technology information to work with. So, instead of starting up icfb directly, we'll start it up with the NCSU CDK already loaded. This will happen by calling Cadence from a script that we've written instead of calling the tool directly. This

script will start a new shell, set a bunch of required environment variables, and call the icfb tool with the right switches set. Other tools for the rest of this book will use similar scripts.

2.2 Starting Cadence

Before you start using cadence you need to complete the following steps:

First make a directory from which to run Cadence. This is important so that all of Cadence's files end up in a consistent location. It's also nice to have all of Cadence's setup and data files in a subdirectory and not clogging up your home directory. I recommend making an IC_CAD directory and then under that making a cadence directory. Later on we'll add to that by making separate directories for the other IC tools like Synopsys dc_shell, module complier, SOC Encounter and so on under that IC_CAD directory.

CAD tools can generate a lot of temporary and auxiliary files!

```
cd
mkdir IC_CAD
mkdir IC_CAD/cadence
```

Now it's handy to set a few environment variables. In particular you want to set your UNIX search path to include the directory that has the startup scripts for the CAD tools. You also need to set an environment variable that points to a location for class-specific modifications to the general Cadence configuration files. I recommend that you put these commands in your .cshrc or .tcshrc file so you won't have to retype them each time you start a shell. If you're using bash you'll have to adjust the syntax slightly. In general you want to set your path to point to where the tool startup scripts live, and set your **LOCAL_CADSETUP** variable to point to the directory that holds the local information. These locations are site- and semester-specific so check with your instructor for details of your system's organization!

All the Cadence and Synopsys CAD tools run on Solaris or Linux so if you don't have a good grasp of basic UNIX commands, now's the time to go learn them!

```
Set path = ($path <path-to-tool-scripts>)
setenv LOCAL_CADSETUP <path-to-local-setup-info>
```

As an example, these commands might be something like the following (again, check with your instructor or tool administrator for your local direcory information):

```
Set path = ($path /uusoc/facility/cad_common/local/bin/F07)
setenv LOCAL_CADSETUP /uusoc/facility/cad_common/local/class/6710
```

Finally, you need to copy one Cadence init file from the NCSU CSK directory so that things get initialized correctly. The file is called **.cdsinit**

(note the initial dot!). You can put it in your $HOME directory so that you'll always get that init file, or you can put it in the directory from which you start Cadence if you think you might ever want to start Cadence from a different directory for different projects or classes with a different **.cdsinit** file.

I recommend making this a symbolic link so that if the system-wide .cdsinit file is updated you'll see the new version automatically. You only need to do this once so that a link to the bbb.cdsinit file is in place before you start Cadence. Again, the specific installation paths are site- and semester-specific so be sure to check for the correct path!

```
ln -s <path-to-NCSU-CDK>/.cdsinit $HOME
or
cd $HOME/IC_CAD/cadence
ln -s <path-to-NCSU-CDK>/.cdsinit .
```

As an example, the <path-to-NCSU-CDK> might be /uusoc/facility/cad_common/NCSU/CDK-F07.

Now that you have your own cadence directory (called $HOME/IC_CAD/cadence if you've followed the directions up to this point), set your path, and linked the NCSU .cdsinit file either to $HOME or to $HOME/IC_CAD/cadence you're ready to start Cadence icfb with the NCSU CDK. Before starting the tool connect to your $HOME/IC_CAD/cadence directory (or where ever you wish to start Cadence from) first.

Start Cadence with the command:

```
cad-ncsu
```

We're using dfII from the IC v5.1.41 release

Of course, once you set this all up once, you should be able to jump right to the `cad-ncsu` step the next time you want to start Cadence.

You should see two windows once things get started up. The first is the **Command Interperter Window** or CIW. It's shown in Figure 2.1. The other is the **Library Manager** shown in Figure 2.2. The CIW is the main command interface for all the dfII tools. In practice you will probably not type commands into this window. Instead you'll use interfaces in each of the tools themselves. However, because most of the tools put their dignostic log information into the CIW, you will refer back to it often. Also, there are some things that just have to be done from this window. For now, just make sure that your CIW looks something like the one in Figure 2.1.

Cadence also keeps the log information in a CDS.log file which it puts in your $HOME directory

The **Library Manager** is a general interface to all the libraries and cells views that you'll use in dfII. Cells in dfII are individual circuits that you want to design separately. In dfII there is a notion of a "cell view" which

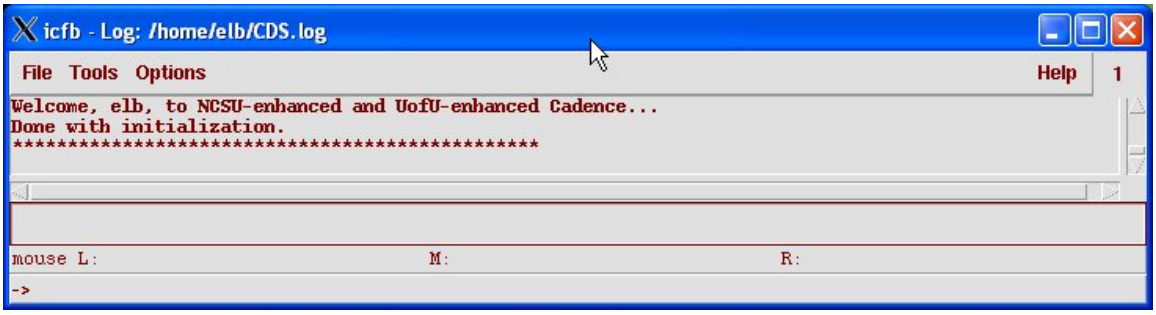

Figure 2.1: Command Interperter Window (CIW) for `cad-ncsu`

means that you can look at a cell in a number of different ways (in different views). For example, you might have a shematic view that shows the cell in terms of its components in a graphical schematic, or you might have a Verilog description of the cell as behavioral Verilog code. Both of these cell views can exist at the same time and are just alternate ways of looking at the same cell. The cell views that we'll eventually end up using in this tool flow are the following:

schematic: This view is a graphical schematic showing a cell as an interconnection of basic components, or as hierarchically defined components.

symbol: This view is a symbolic view of the cell that can be used to place an instance of this cell in another schematic. This is the primary mechanism for generaing hierarchy in a schematic.

cmos_sch: This is a schematic that consists of CMOS transistors. A cmos_sch view corresponds to a cell that is completely contained in a single standard cell. That is, it is a leaf-cell in the standard cell hierarchy that corrseponds to a cell in an existing library. It's important in some tool steps to differentiate the schematics that should be expanded by the netlisting process and the leaf cells where the netlisting should stop. That's the purpose of the cmos_sch view.

extracted: This view is generated by the circuit extraction process in the Cadence tools. It contains an extracted electrical netlist of the cell that the simultors can use to understand the electrical behavior of the cell.

analog-extracted: This view is generated from the extracted view and contains a little extra information for the analog simulator.

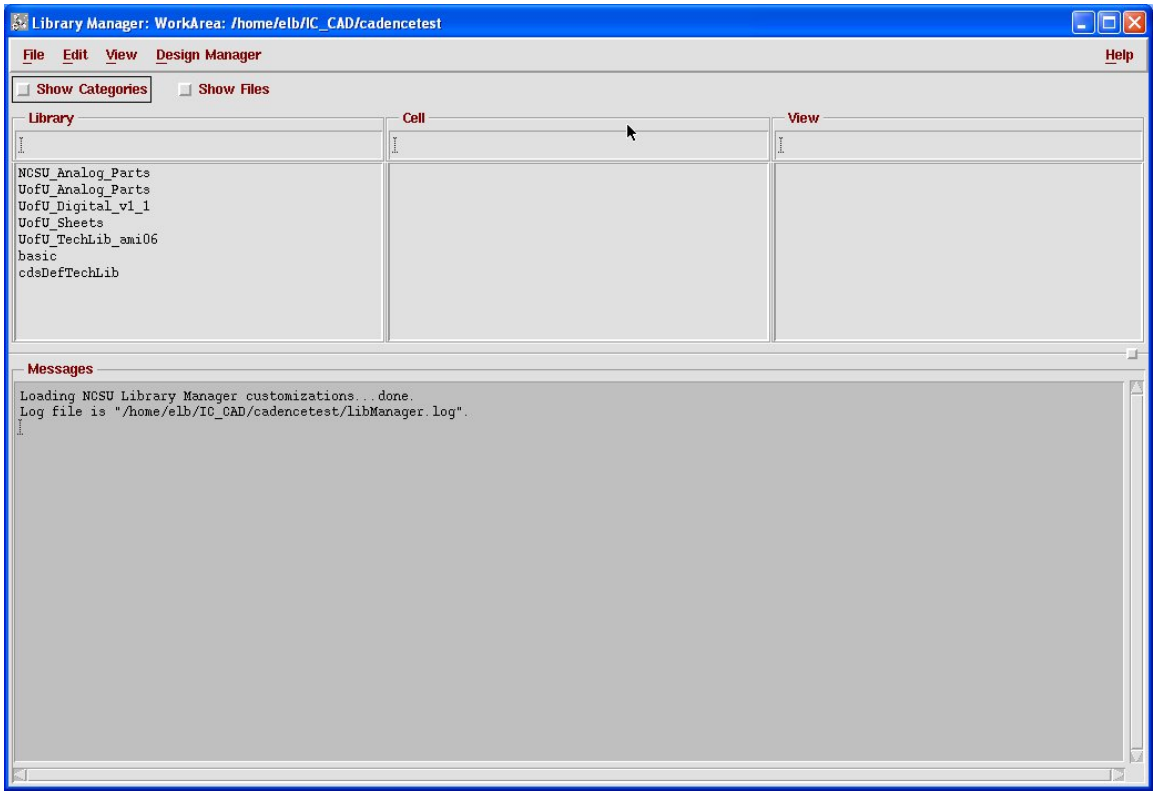

Figure 2.2: Library Manager window in `cad-ncsu`

behavioral: This view is Verilog code that describes the behavior of the cell.

layout: This view contains the composite layout information for a cmos_sch cell. This is the graphical information that the IC fabrication service uses to fabricate the cell on the silicon.

abstract: This layer takes the layout and extracts only the informaton that the place and route software needs to do the placement and routing. That is, it needs to know the physical dimensions of the cell, the connection points and layers, and any obstructions for the metal routing layers, but it doesn't need to know anything about the transistor layers. This view will be generated by the `abstract` process.

functional: This view is reserved for behavioral descriptions of CMOS transistors. It's used for a similar reason to the cmos_sch view: it lets the netlister know when it has hit a transistor. You won't need to create these views unless you're adding new transistor models to a library.

spectre: This view is used by the analog circuit netlister to generate an input file for the Spectre analog simulator. You won't need to create this view unless you're adding new transistor models to a library. There are a number of other similar views for other simulators that you also don't need to worry about.

A "library" is a collection of cells that are grouped together for some reason (being part of the same project, or part the same set of standard cells, for example). Libraries also have technology information attached to them so that the cells in the library refer to a consistent set of technolgy data. This technology information is linked rather than copied so that when updates are made on the techology, all libraries with that technology attached will see the updates. For example, all the standard gates cells that you'll be using (until you make your own!) are grouped into a library called **UofU_Digital_v1_1**. You will create libraries for each of your designs so that you can keep the design data separate for different projects. Think of librarires as directories that collect design dat together for a specific design. You could throw all your stuff into one directory, but it would be easier to find and use if you separate different designs into different libraries.

The UofU_Digital library uses the _v1_1 syntax to indicate version 1.1 of the library. Cadence doesn't like dots in cell names!

You should see a bunch of libraries already listed in the Library Manager. If you scroll around you should be able to see the following libraries if you are using the NCSU CDK:

If you're using a different CDK or PDK, you'll see different libraries in the default list

NCSU_TechLib_xxx: These are technology libraries for each of the MOSIS processes. The "xxx" will be filled in with information about which MOSIS process is being desctribed (**ami06** for the AMI C5N $.6\mu$ process, for example). We won't use these directly, and depending on how Cadence is set up for your class you might not see these at all. If you're not using the UofU packages, then you'll probably see all of these.

NCSU_Analog_Parts: This library contains components (transistors, resistors, capacitors, etc.) that you'll use for transistor-level design, and also some components for circuit-level simulation using spectre (in the Affirma analog circuit design environment). The switch-level transistor models in this library have zero delay for simulation.

NCSU_Digital_Parts: This library contains a variety of Boolean logic gates that you can use for gate-level design. Note that these gates do **not** have layout or place and route views so they **can not** be used for actually building chips! They are typically used in classes just for the initial "learn about the schematic capture tool" assignments.

basic: This is the Cadence built-in library which you won't use directly. It has basic components from which other parts are built.

cdsDefTechLib: A generic Cadence technology that we won't use.

If you're using a different CDK (Cadence Design Kit) or PDK (Process Deisgn Kit) you'll see different libraries. Also, you may see additional libraries for local additions or modifications to the default libraries. For example, at the University of Utah you'll see:

UofU_Analog_Parts: This is a library with copies of the transistor components from the **NCSU_Analog_Library**, but these transistors have 0.1ns of delay for switch level simulation.

UofU_TechLib_ami06: This is a technology library for AMI C5N 0.5μ library using the SCMOS_SUBM rules that we'll use in the tutorials. It's based on the NCSU technology library for this process, but has some local tweaks that make it a little more friendly to this flow.

UofU_Sheets: This library has graphics for schematic sheet borders that are specific to the University of Utah.

If you look at the cells in UofU_Digital_v1_1 you should see that each of them has a number of different cell views as defined previously

UofU_Digital_v1_1: This is a library of standard cells developed at the University of Utah for VLSI classes. It has the UofU_TechLib_ami06 technology attached to it so it can be used with the AMI C5N 0.6μ CMOS process through the SCMOS_SUBM design rules from MOSIS. This library will be enabled for viewing when it's needed.

UofU_Gates_v1_1: This is a library with only the gate (cmos_sch, behavioral, and symbol) views of the cells in the UofU_Digital_v1_1 library. It's used for initial assignments so that students can use the gates without seeing the other cell views.

UofU_Pads: This library (enabled for viewing when it's needed) contains I/O pad cells to be used with the AMI C5N CMOS process.

Unfortunately, you'll have to keep very careful track of when to use components out of each of these libraries. Some have very specific uses. The only way to handle this is just to pay attention and keep track!

Now that you've started Cadence using the `cad-ncsu` script, we can move on to using the individual EDA tools in the dfII suite...