Standard Cell Libraries

WARNING:

Students in this course are signed on a CIU (confidential information undertaking) stating that certain information you will access is proprietary and/or confidential, including information about process design kits (PDKs) and technology libraries. You may not disclose any of this information to non-authorized parties and/or share this information in any way. The information in this document is included and should not be shared outside this course!

1 Introduction

In this discussion, we will take a look at a real standard cell library, see what it includes and get the feel of what are first steps should be when getting a new technology library. For the purpose of this discussion, we will be using the ARM standard cell library, designed for the TSMC 65LP process.

It is important to note a few things here:

1. ARM is an *IP Provider* or *IP Vendor*. In other words, they sell IP to other companies, who can license the IP to make chips. In the case of the standard cell library we will be using, ARM has developed the standard cells for use with a specific technology – TSMC 65nm LP. Only the files and views describing the standard cells are provided by ARM – the technology is provided by TSMC.
2. In order to protect their IP, many technology and IP providers only provide *a part* of the library to Universities and small companies. In the case of the library we will be looking at, ARM has only supplied the views necessary for running synthesis and place and route. These are known as *Front-end Views*. The full library (including the *back-end views*) are not provided. To see an example of back-end views, please refer to the *Tower 180nm* version of this document.

1.1 Installation Path

On the RC3 cloud of BIU, all the design kits (PDKs) are installed under the `$kit_dir` directory (group dependent path). First, you will need to enter a project, which will set up your paths, licenses and environment variables. For this course, we will be using the TSMC 65nm LP and so run:

```
%> tsmc65
tsmc65 workarea setup exists.  Entering workspace
Project tsmc65 setup is complete. Entering your workspace
ws> echo $kit_dir/
 /data/tsmc/65LP/pdk/1.7a
```

The `$kit_dir` directory has many different files (unfortunately, what technology and IP vendors consider as being "in order" is often a big mess…). But this is the *technology installation* directory, so most of these files are what we need for running transistor level simulation and other technology features. In fact, we used a lot of the files here during courses, in which we ran Spice simulations and layout with Virtuoso.

We are not focusing on the PDK in this course (that's what we did in previous, circuit design courses), but rather on the digital libraries. These are installed one directory *above* the PDK, i.e., at `/data/tsmc/65LP/` or simply `$kit_dir/../..` . Specifically, the standard cell libraries are installed under a sub-directory called `dig_libs`. In addition to the standard cell libraries, this directory includes installations of other standard IPs, such as IOs, Bond Pads, Memories, etc.

Note that each vendor (library provider) will put their files in a different location. I suggest going to the installation area of the design kit (under /data/<vendor>/) and start looking for a directory with a name that has to do with "digital" or "library" or "standard cells" or "core" or something similar. For example, in the case of this library, it's called "dig_libs", short for "digital libraries".

In this course, we will be using standard cells provided by ARM, so look for a directory called "`ARM`". Actually, you will see three directories: `ARM_FEONLY`, `ARM_FEONLY_2`, and `ARM_BAK`. ARM_FEONLY (as the name hints) only provides access to the *front-end* views (i.e., no GDS, SPICE, etc.). Don't ask me why we have three directories with various names, but get used to the mess of library installations…

As you will notice, there are **a lot** of files and directories. In fact, it's quite a mess. Well, welcome to the real world – this is what it looks like. Get used to it. You can find our files under `$kit_dir/../../dig_libs/ARM_FEONLY/arm/tsmc/cln65lp`

Well, that's a very long path and hard to get to. Let's define a shell variable called lib that will provide us a shortcut to this directory.

First let's switch to a **tcsh** rather than **bash**:

```
% tcsh
% set arm = $kit_dir/../../dig_libs/ARM_FEONLY/arm/tsmc/cln65lp/
```

1.2 Track options

If we list the directory contents of the ARM standard cell library, we will see the following directories:

```
% ls $arm
arm_tech sc9_... sram_...
```

Lots of directories there. We've gotta now play the "guessing game". I think we can guess that `sram_...` directories have something to do with the memory compiler that we will play around with in a future exercise. If we look in the other directories, we will see `rf_...`, which is shorthand for "Register File", so it's also a memory and irrelevant for now. On the other hand, the `sc…` directories sound like… "Standard Cell". Good guess!

In fact, we saw there are “families” of sc… directories that start with a 9. My guess is that this number stands for the number of tracks in the library (i.e., “9-track library”).

Remember from the lecture that standard cell libraries are categorized by the number of tracks? The more horizontal tracks between VDD and GND, the wider the transistors can be, and therefore we have *faster* standard cells. However, they take up more area and they have higher leakage. So we should choose our library according to the speed vs. area/power trade-off. Dense cells are also harder to route, which could lead to less optimal routing – slower.

In this case, we chose to only install the 9-track library, which is kind of a sweet spot between speed and area. So we are looking at the directories with the structure `sc9_*` .

1.3 VT Option

So, you thought we were done… Think again! There are 3 sub-directories starting with `sc9_*`. We didn’t install the ones we won’t be using in this course, but here are the naming conventions for the various directories that the vendor provides:

- `base` – this is the basic standard cell library – it’s what we need!
- `eco` – this is the *engineering change order* library. Basically, spare cells for performing ECOs. ARM decided to put these cells in a separate directory.
- `pmk` – that stands for *power management kit*. These are all kinds of cells for power management that we won’t be needing in this course.

Okay, so we decided we need one of the sub-directories called `sc9_base_...`, but which one. Well, there are three options: `lvt`, `rvt`, `hvt`. Remember we discussed MTCMOS (Multi-Threshold CMOS) – the option to trade-off speed vs. leakage by just changing the VT of the transistors that make up our digital gates. Well, voila, here you have three libraries that are exactly the same, other than the VT of the transistors inside:

- `sc9_base_rvt`: Standard VT cells – the default choice.
- `Sc9_base_hvt`: High VT cells – slow, but low leakage.
- `Sc9_base_lvt`: Low VT cells – fast, but high leakage.

We will explore the RVT option, so let’s set another shell variable:

```
% set sc9rvt = $arm/sc9_base_rvt/r0p0
```

Note that the “r0p0” sub-directory refers to the revision version of this library – the only revision we have installed.

2 Library contents

So, what's inside our standard cell library? You can usually expect to find at the very least:

- **Documentation**: Description of the library, the files in the library, datasheets of the standard cells, and others.
- **Front End Views**: *Front End* is usually considered the part of the design flow before physical implementation. Therefore, anything used up until (and sometimes, including) synthesis, can be categorized as a *Front End View*. These would include **Verilog/Vital** behavioral descriptions, **Liberty** (`.lib`) timing/power/noise files, and design for test (**DFT**) views.
- **Back End Views**: *Backend* would usually be considered the physical implementation of the design, starting from synthesis and ending in physical verification (**DRC/LVS**). Therefore, the *Back End Views* will include the **layout** (GDS) of the cells, **abstracts** (LEF), **Spice** models, **power grid** models, and if you're lucky, **Virtuoso** libraries (OA database) of your library.

Note that, especially in Academia, library vendors will provide *only* Front End views of the libraries. Well, according to the categorization above, this is not accurate, so I will reformulate it as they will provide only *partial Back End views*. This means that they will remove anything they feel is more confidential, such as GDS and Spice models from the library. The full digital design flow can be carried out with this limited set of files (replacing GDS with LEF abstracts, for example), and during tapeout, the abstracts will be replaced with the actual GDS and DRC/LVS will be run again.

2.1 The ARM Standard Cell Library Contents

If we list the contents of the `$sc9rvt` directory, we will see a lot of familiar names, such as `verilog`, `lef`, `lib`, and `oa`. Indeed, these sub-directories hold the contents of the library, as we learned in the lecture. Many other names are less familiar, either because ARM decided to give them other names or because they are formats for tools of other vendors that we possibly didn't go over in class.

The bottom line is that each IP vendor provides library contents in a directory structure that they decide upon, and you will have to do a bit of "guessing" and "exploring" to find what you need. But, unquestionably, the first place to start is the documentation.

Note that if you want to see the contents of a "better organized" standard cell library, refer to the Tower 180nm version of this document.

3 Documentation and Datasheets

The way to start exploring any library is to read the documentation. There may be documents scattered around the whole directory structure, but a good place to start is the doc directory.

```
% ls $sc9rvt/doc/
sc9_cln65lp_base_rvt_cell_list.txt  sc9_cln65lp_base_rvt_databook.pdf  sc9_cln65lp_base_rvt_userguide.pdf
```

Well, that was very helpful. Here we have three really important documents:

- `sc9_cln65lp_base_rvt_userguide.pdf`: The user guide for this standard cell library.
- `sc9_cln65lp_base_rvt_databook.pdf`: The datasheet of the standard cells inside the library
- `sc9_cln65lp_base_rvt_cell_list.txt`: A text list of all the standard cells in the library.

3.1 User Guide

There is no better place to start than the user guide…

```
% acroread $sc9rvt/doc/sc9_cln65lp_base_rvt_userguide.pdf &
```

The user guide tells us how to use the library and the datasheets. A lot of it is pretty generic and straightforward, but useful. For example, on page 2, we have the definition of propagation delay used in the datasheets and .lib file:

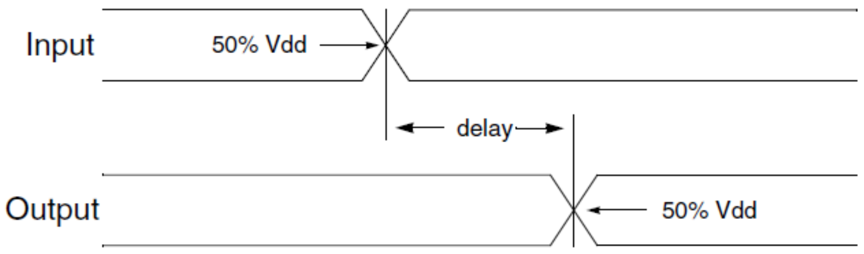


Figure 1: Propagation delay waveform from User Guide

Look familiar? This is defining that t_{pd} is measured from 50% VDD crossing at the input to 50% VDD crossing at the output. Actually, if you read the text, it says that this is just an example, and you have to look at the actual datasheet to find out the specific definition…

The first part of the user guide just has these generic definitions that you should already be familiar with. Propagation delay, rise/fall time, etc. Go over it to make sure you know what they're talking about.

3.1.1 Special Cells Section

The ARM 65nm User Guide has a section describing the special cells to use with the process. These include:

- **Antenna Fix cells**: Diodes used to fix Antenna Violations.
- **Delay Cells**: Cells to fix hold violations.

- **Endcap**: Cells to put at the edges of standard cell rows.
- **Decap** Fillers: Cells that fill the empty spaces in standard cell rows and add decoupling capacitance between VDD and GND.
- **Low-Power Cells**: Minimum sized gates for power saving.
- **Well-Taps**: Cells to connect the NWELL and P-SUB to VDD and GND, respectively.
- **Register file cells**: Special flip flops for synthesized memories.
- **Tie cells**: Cells to connect constant signals to VDD and GND.

3.1.2 Library Naming Convention

This is a critical part of the standard cell library documentation, especially with a non-standard naming convention, such as in the ARM library. It tells us what the (hard to read) name of the cell means. The ARM cells adhere to the following convention:

`[product]_[process corner]_[extraction]_[overlay]_[voltage]_{voltage2}_[temp]`

With the product field containing these fields:

`sc[cell height]{mc}_[process node name]_[toolkit name]_[threshold voltage]{_c[channel length]}`

Wow! How are we going to figure out what a gate is??? That's why they're going to pay you lots of money in the chip design industry!

Let's just give an example:

BUFH_X2P5B_A9TR:

- BUFH: high speed buffer
- X25P5: Drive strength is 2.5X (P stands for "point")
- B: The beta-ratio of the cell is balanced (tplh=tphl). This is a clock cell!
- A9: ARM 9-track library (all cells in this library will have this name)
- TR: Regular VT (all cells in this library will have this name)

3.2 Library Datasheet

Now open the document entitled `sc9_cln65lp_base_rvt_databook.pdf`:

```
% acroread $sc9rvt/doc/sc9_cln65lp_base_rvt_databook.pdf &
```

This document has general information about the standard cell library, such as:

- Types of logic gates included in the library.
- Operating Voltage and temperature ranges and operating conditions.
- Physical dimensions
- Characterization corners (PVT)
- Files included in the SC library
- Known problems and revision history

3.2.1 General Datasheet Information

The beginning of the datasheet includes general information about the standard cell library, such as the cell height (1.8 um), the number of tracks (9), and the layout grid resolution (5nm).

Later on, the characterization parameters (corners) are given. We can learn that the provided corners are:

- Typical: 1.2V, 25C
- Best Case (Fast): 1.32V, -40C
- Worst Case (Slow) 0.81V, 125C

The general section then explains how to read the particular datasheet of each cells, as described below.

The book then provides the datasheets for each of the standard cells in the library, categorized according to cell type/function.

3.2.2 Datasheet description of a gate

As an example, let's take the datasheet of an inverter on Page 385 of the PDF. The data provided is:

- Logic Equation and Truth table

Figure 61-1 Logic Equation

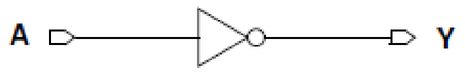
$$Y = \overline{A}$$

Table 61-1 Function Table

A	Y
0	1
1	0

- Gate schematic symbol (with pin names)

Figure 61-2 Functional Schematic



- Drive strength options with cell size

Table 61-2 Cell Size

Cell	Height (µm)	Width (µm)
INV_X0P5B_A9TR	1.8	0.6
INV_X0P5M_A9TR	1.8	0.6
INV_X0P6B_A9TR	1.8	0.6
INV_X0P6M_A9TR	1.8	0.6

- Input pin capacitance for all cell sizes:

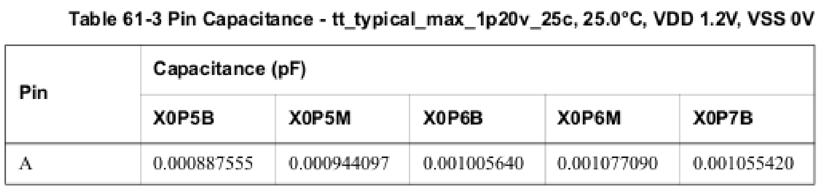


Table 61-3 Pin Capacitance - tt_typical_max_1p20v_25c, 25.0°C, VDD 1.2V, VSS 0V

Pin	Capacitance (pF)				
	X0P5B	X0P5M	X0P6B	X0P6M	X0P7B
A	0.000887555	0.000944097	0.001005640	0.001077090	0.001055420

- Delays, provides as intrinsic delays and then per-output cap delay function:

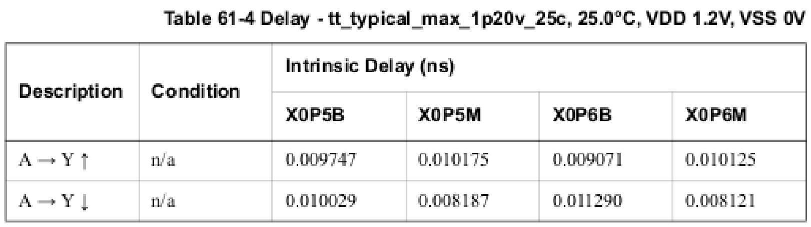


Table 61-4 Delay - tt_typical_max_1p20v_25c, 25.0°C, VDD 1.2V, VSS 0V

Description	Condition	Intrinsic Delay (ns)			
		X0P5B	X0P5M	X0P6B	X0P6M
A → Y ↑	n/a	0.009747	0.010175	0.009071	0.010125
A → Y ↓	n/a	0.010029	0.008187	0.011290	0.008121

- And tables with dynamic and leakage power consumption

4 Front End Views

4.1 Behavioral Models

In order to simulate the behavior of a standard cell, its functionality is described in a behavioral model. This can be either a Verilog file or a VHDL Vital file. In the case of the ARM library, only Verilog descriptions are provided.

Scroll through the Verilog behavioral model:

```
 % less $sc9rvt/verilog/sc9_cln65lp_base_rvt.v
```

Find the behavioral description of the smallest inverter. In "`less`", press "`/`" to search, and search for the string "`module INV_X0`".

The description is "strange". It defines the inputs and outputs of the inverter and the **Verilog** primitive of its behavior (`not`), as expected, but then has some macros. Don't worry, you are not expected to understand or know how to write these, but they are used by logic simulators to correctly simulate the standard cells and enable timing back-annotation, as we will learn later.

Note that there are two other **Verilog** files in the library. Possibly, you could find out what they are used for in the library datasheet.

4.2 Liberty (.lib) Timing/Power/Noise Models

Inspect the content of the liberty folder. In it, you will find a `.lib` file for each of the corners defined in the library datasheet. Let's open the typical timing corner. The naming convention for the files is `sc9_cln65lp_base_rvt_<process>_typical_max/min_<voltage>_<temperature>.lib`, so:

```
 % less $sc9rvt/lib/sc9_cln65lp_base_rvt_tt_typical_max_1p20v_25c.lib
```

Scroll through the file. The heading information contains interesting data, such as:

- The library name
- The measurement units
- The definition of **tpd** (i.e., 50% rise to 50% fall) and **trise/tfall**.
- Operating conditions of this corner (TT, 25C, 1.2V)
- Lookup tables for input transitions and output capacitances
- Wire Load Models

Afterwards, the cells are defined. Search for the minimum sized inverter (remember, "`/`", then "`INV_X0`")

```
cell(INV_X0P5B_A9TR) {
    area : ...

    pin(A) {
      capacitance : ...
    }
    pin(Y) {
      direction : output ;
      function : "(!A)'" ;
      max_capacitance : ...

      internal_power() {
        related_pin : "A" ;

        fall_power(pwr_tin_oload_7x7) {
           values(...
      timing() {
        related_pin : "A" ;
        timing_sense : negative_unate ;
        timing_type : combinational ;
        cell_fall(tmg_ntin_oload_7x7) { ...
```

A similar definition appears for every standard cell.

Note that the `db` directory contains the same information, but provided in the `db` format, which is what Synopsys tools (such as Design Compiler) use. A short and easy script can translate the `.lib` files into `.db` files and saved to use with Synopsys tools.

Note that almost the same information appears in the `lib-ccs-tn` and `lib-ecsm-t` directories. However, these use the CCS and ECSM timing models, respectively, rather than the NLDM model, used in the `lib` directory. For process technologies under 130nm, it is recommended to use either CCS or ECSM.

4.3 Other Front-end Views

The library contains other directories (such as "`tetramax`", "`volcano`", and "`voltagestorm`"), which are intended for use with other tools, beyond the scope of this course.

5 Back End Views

As mentioned previously, the ARM standard cell library is not equipped with full back-end views. Therefore, we will only discuss the LEF directory. For an example of a full back-end view library, see the Tower 180nm version of this document.

5.1 LEF

The Library Exchange Format is a file format used for two primary goals:

1. To provide design rules for place and route tools (**Tech-LEF**).
2. To provide gate abstracts for placing and routing standard cells and other IPs (**Cell-LEF**).

5.1.1 Gate Abstract LEF

Since this library is provided by an *IP Vendor* and not by the *foundry*, the standard cell LEF files only contain the gate abstracts and the **Tech-LEF** is in a separate file. Let's start by looking at the gate abstracts file:

```
%> less $sc9rvt/lef/sc9_cln65lp_base_rvt.lef
```

The beginning of the file contains the SITE definition for this library:

```
SITE sc9_cln65lp
  CLASS CORE ;
  SIZE 0.20 BY 1.80 ;
  SYMMETRY Y ;
END sc9_cln65lp
```

The SITE command defines the minimum grid unit of a placement block. This is similar to one "dot" in Legos. Every cell dimension has to be a multiple of this size. This allows row based placement of standard cells to work!

Now let's look at our nifty old inverter. Search for "MACRO INV_X0":

```
MACRO INV_X0P5B_A12TR
  CLASS CORE ;
  ORIGIN 0 0 ;
  FOREIGN INV_X0P5B_A9TR 0 0 ;
  SIZE 0.6 BY 1.8 ;
  SYMMETRY X Y ;
  SITE sc9_cln65lp ;
  PIN A
    DIRECTION INPUT ;
    USE SIGNAL ;
    ANTENNAMODEL OXIDE1 ;
      ...
    PORT
      LAYER M1 ;
        RECT ...
    END
  END A
  ...
```

What we see is that the LEF file describes all the relevant polygons for connecting a standard cell without actually going down into the “frontend” layers, such as diffusions and poly. It also includes the Antenna information to make sure you don’t burn your chip during fabrication.

5.1.2 Technology LEF

To find the technology LEF, we have to either go back to the PDK, provided directly by TSMC, or in this case, ARM also provides it for us, just in a separate directory.

The **Tech-LEF** can be found at the following locations:

```
%> less $arm/arm_tech/r2p0/lef/1p9m_6x2z/sc9_tech.lef
```

Pay attention that we are looking at a sub-directory called “`1p9m_6x2z`”. Strange name… It actually describes the *metal stack* of our process option, having **1 polysilicon** & **9 metal layers** in total, with **6 of them local interconnect** (X) layers and **2 ultra thick layers** (Z).

Inside this file, you will find definitions of all the routing layers and design rules (such as `SPACING` and `WIDTH`) that tell the router inside PnR tool how to check DRCs.

Pay attention. You need to read in the Tech-LEF before reading in the gate abstract LEF or the LEF of any other IP, so the PnR tool (Innovus) will know the definitions of the routing layers.

5.2 OA Database

Finally, if you’re lucky, your vendor will provide you with an OA database of your library. `OA` stands for “Open Access” and it is the “new” format defined by Cadence for working with other companies. Of course, Synopsys and Mentor Graphics don’t want to use Cadence’s format, and so they use others (such as “`MilkyWay`”) and Cadence itself keeps backward compatibility (circa 2006…) to their old format, known as `dfII` or `cdb`.