VLSI Design Course

LAB 0

Cell Design and Verification

Issue 1.1

Contents

1 Introduction

1.1 Lab overview

This is the first design lab for the VLSI Design course. The labs teach the practicalities of chip design using commercial CAD tools from Cadence Design Systems, and emphasizes the fundamentals of custom design.

This lab introduces you to the basics of how to use Cadence to design, simulate, and verify schematics and layout of logic gates. It also serves as a stand-alone tutorial to quickly get up to speed with the Cadence tools. This lab is divided into two parts.

In the first part of this lab, you will draw transistor-level schematics to build cells for NAND, NOT, and AND gates and create a symbol for each cell. You will then simulate each cell by applying digital inputs and checking that the outputs match your expectations.

In the second part of this lab, you will draw a layout for each gate you created in part 1. The layout indicates how the transistors and wires are physically arranged on the chip. Using design rules check (DRC) and layout versus schematic check (LVS), which is provided by the tool, the layout is checked to ensure it satisfies the design rules and that the transistors match the schematic.

Finally, you will apply what you have learned by independently designing NOR and OR gates.

2 Learning Objectives

At the end of this lab, you should be able to:

- Draw transistor-level cell schematics using Cadence Virtuoso
- Simulate transistor-level schematics using NC-Verilog
- Draw cell symbols using Cadence Virtuoso
- Draw cell layouts using Cadence Virtuoso
- Perform design rule checking (DRC) and fix discrepancies
- Perform layout vs. schematic (LVS) checking and fix discrepancies
- Construct, simulate, and verify hierarchical cells

3 Overview of VLSI CAD Tools

This set of laboratories uses the Cadence Electronic Design Automation (EDA) tools which when correctly set up are easy to use, as this tutorial will demonstrate.

There are two general strategies for chip design. Custom design involves specifying how every transistor is connected and physically arranged on the chip. Synthesized design involves describing the function of a digital chip in a hardware description language such as Verilog or VHDL and then using a computer-aided design tool to automatically generate a set of gates that perform this function, place the gates on the chip, and route the wires to connect the gates.

Most commercial designs are synthesized today because synthesis takes less engineering time. However, custom design gives more insight into how chips are built and into what to do when things go wrong. Custom design also offers higher performance, lower power, and smaller chip size.

4 Tool Setup

Cadence is installed on the burrow-rhel server at IU Luddy, located at

```
burrow-rhel.luddy.indiana.edu
```

Log into burrow-rhel from the lab computers or your own personal computing system with the following command (in a terminal):

```
ssh -Y uname@burrow-rhel.luddy.indiana.edu
```

where **uname** is your assigned Linux user name (ID).

The tools generate a bunch of files. It's best to keep them in one place. In your home directory, create some directories by typing:

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

In addition, the course files will be updated and are provided on GitHub. Cloning this repository will ensure you have these directories and will include any relevant files.

```
https://github.com/tdloveless/iu-vlsi.git
```

Ensure that you have the latest files prior to proceeding with any homework or lab activity.

5 Start Virtuoso

Before you start the Cadence tools, open a terminal and change into the cadence directory:

```
cd ~/IC_CAD/cadence
```

Start Cadence using the following command:

```
module load cadence
virtuoso
```

The above command starts the Virtuoso software and will open the Virtuoso Design Environment and the Library Manager windows.

A “What’s New” and a Library Manager window may open. You can **turn off** the *“What’s New”* window in the future by choosing **Edit • Off at Startup**.

You may see some warnings about “no route to host” and missing fonts in the terminal; you can safely ignore these. Scroll through the Virtuoso window and look at the messages displayed as the tool loads up. Get in the habit of watching for the messages and recognizing any that are out of the ordinary. This is very helpful when you encounter problems. At present, you shouldn’t see any warnings in Virtuoso.

5.1 Library Manager

All of your designs are stored in a *library*. If the Library Browser doesn’t open, choose Tools • Library Manager. You’ll use the Library Manager to manipulate your libraries. Don’t try to move libraries around or rename them directly in Linux; there is some funny behavior, and you are likely to break them.[1]

When you are all done, be sure to quit Cadence by choosing **File • Exit**… in the Virtuoso window. If you don’t, you may lose the work you’ve done and/or leave your library in a corrupted and unstable state. If this occurs, you may find a `panic.log` file in your home directory. The file will include directions to try to recover your work. Restart `virtuoso`. Before doing anything else, enter the instructions from `panic.log` into the Virtuoso window command line. For example:

```
dbOpenPanicCellView("lab3_xx" "test" "schematic")
```

If Cadence crashes, it might leave your locks on open files in place. If your lock shouldn’t be there, you can manually remove it by going into the library directory and deleting the `.cdslck` files. As always, be careful when deleting files.

Familiarize yourself with the Library Manager. Your cds.lib file includes many libraries from the North Carolina State University Cadence Design Kit supporting the different MOSIS processes. It also includes libraries from the University of Utah. The File menu allows you to create new libraries and cells within a library, while the Edit menu allows you to copy, rename, delete, and change the access permissions.

5.2 Create a library

Ensure the Library manger window is open and then create a library by doing the following:

- **Go to:** File • New • Library

  If the Library Manager window is not open, you can open it from the Virtuoso window by selecting: **Tools • Library Manager**.

- Name the library **lab0_xx**, where xx are your initials.

[1] Copying a library with `cp -r <srcpath> <dstpath>` does work correctly, and you will find this helpful when working with a partner on the final project.

Leave the path blank, and it will be put in your current working directory (~/IC_CAD/cadence). Set the following options.

- **Select:** "Attach to existing tech library"
- **Select:** sky130_fd_pr_main (also known as SkyWater 130 nm S130 (5M)).

sky130_fd_pr_main is a technology file for the SkyWater 130 nm Open Source 130 nm process, containing design rules for layout.

You should see the new library **lab0_xx** listed (left side) among other libraries in the Library Manager Window.

6 Part 1a: Schematic Entry

In this part, you will create transistor-level schematic for a 2-input NAND gate. Each gate or larger component is called a cell. A cell can have multiple views such as schematic, layout, extracted, etc.

6.1 Create schematic view

The schematic view for a cell built with CMOS transistors should be named **cmos_sch**. Later, you will build a view called **layout** to specify how the cell will be physically manufactured.

To create a schematic view, in the Library Manager, **highlight** the library **lab1_xx** and then:

- **Go to:** File • New • Cell View

In the New File window that opens:

- Confirm that **lab1_xx** is selected for **Library**
- Enter **nand2** for **Cell** name
- Enter **cmos_sch** for **View** name
- Enter **schematic** for **Type**
- Select **Schematic XL** for **Open with** and
- Ensure to **tick** the Checkbox **"Always use this application for this type of file"**
- Click *Ok*

You may get a window asking you to confirm that cmos_sch should be associated with this tool (click Yes), or one complaining that it failed to check out the license for Virtuoso_Schematic_Editor_L (Click Session).

The schematic editor window will open.

Your goal is to draw a gate like the one shown in Figure 1. We are working in a 0.130 µm process with λ = 0.065 µm. Our NAND gate will use 12 λ (0.78 µm) nMOS and pMOS transistors.

To create the schematic as shown in Figure 1, you will need to create instances of transistors (nmos and pmos), power, and pins and connect them using wires.

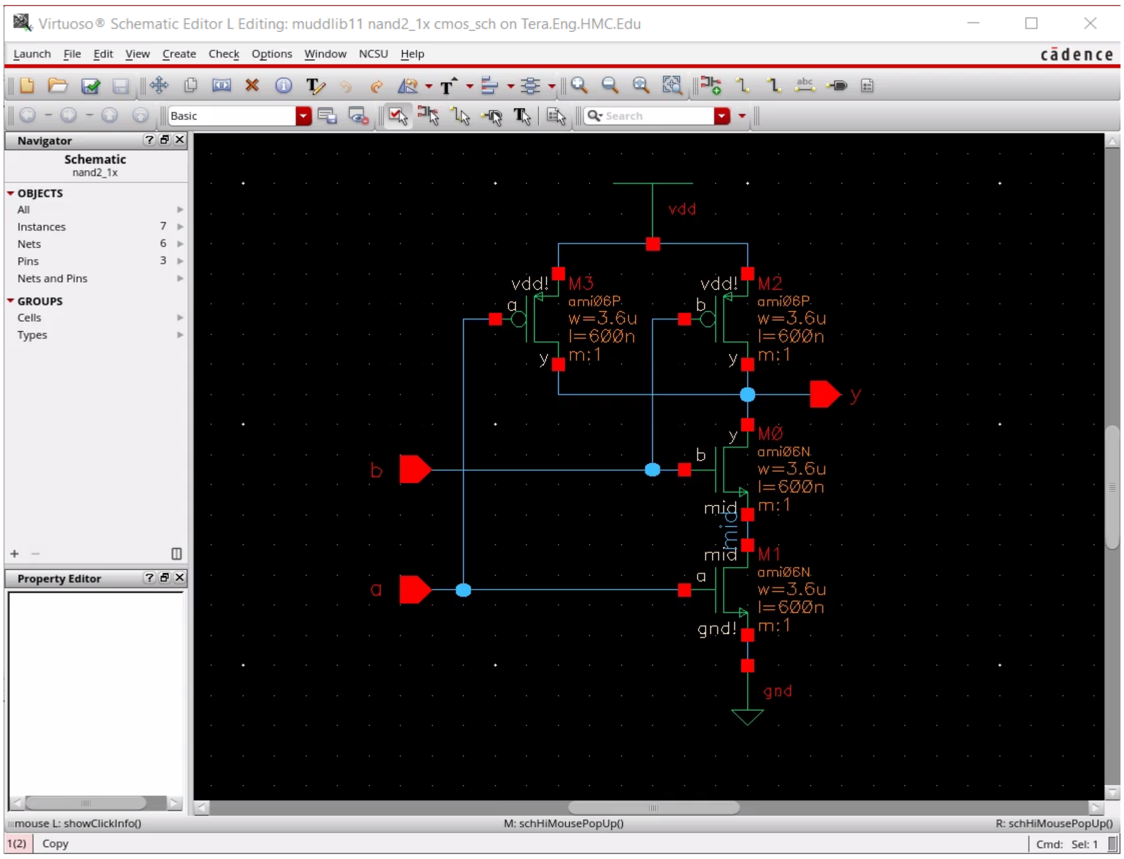

Virtuoso® Schematic Editor L Editing: muddlib11 nand2_1x cmos_sch on Tera.Eng.HMC.Edu
Launch File Edit View Create Check Options Window NCSU Help
cādence
Navigator
Schematic
nand2_1x
OBJECTS
All
Instances 7
Nets 6
Pins 3
Nets and Pins
GROUPS
Cells
Types
Property Editor
vdd
vdd! M3 ami06P w=3.6u l=600n m:1
vdd! M2 ami06P w=3.6u l=600n m:1
a
b
y
M0 ami06N w=3.6u l=600n m:1
mid
M1 ami06N w=3.6u l=600n m:1
gnd!
gnd
mouse L: showClickInfo()
M: schHiMousePopUp()
R: schHiMousePopUp()
1(2) Copy
Cmd: Sel: 1

Figure 1: nand2 cmos_sch

6.2 Create component instances

Create transistor and power instances, by selecting transistors from the installed library using these steps:

- **Go to:** Create • Instance (i)

On the **Component Browser/Add Instance** window that opens:

- Click **Browse** and **Select:** sky130_fd_pr_main for the **Library**
- **Select: nfet_01v8** for the **Cell**
    - This expands the Add Instance window
- Confirm **symbol** is selected for **View**
- Set **Width** to **0.78u** (u indicates microns).
- **Click** in the schematic editor window to drop the transistor.
- You can **click a second time** to place another transistor.

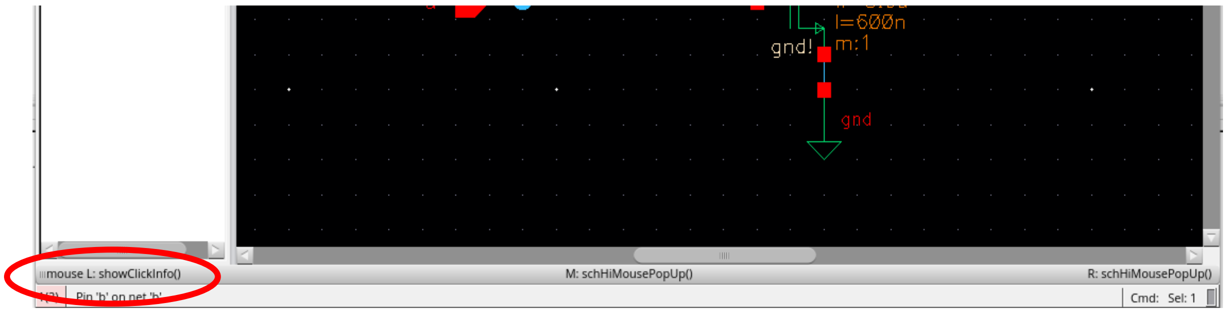
Return to the Component Browser window and choose pmos. Set the width to 3.6u and drop two pMOS transistors in the schematic editor window. Also drop gnd and vdd in the schematic editor window.

Move the elements around until they are in attractive locations.

6.3 Useful shortcuts

The following are shortcuts that will be useful when using Cadence tools.

- Exit a mode when in editor: *ctrl+c* OR *Esc* (press Esc twice if a dialog box is open)
- The bottom of the schematic editor window (see Figure 2) shows you the mode you are in. showClickInfo() is the normal mode
- Undo: *u* (note that saving clears your undo history)
- Copy: Edit • Copy (c)
- Move: Edit • Move (m)
- Delete: Edit • Delete (Del).
- Properties: Edit • Properties • Object (q)



mouse L: showClickInfo() M: schHiMousePopUp() R: schHiMousePopUp()
Pin 'b' on net 'b' Cmd: Sel: 1

Figure 2: Mode Indicator Text

6.4 Create pins

To create pins, do the following:

- Go to: Create • Pin… (Keyboard shortcut p).

On the Create Pin dialog box:

- Enter “a b” (no quotation marks, and a space between the two pin names).
- Set direction to “input.” The tools are case-sensitive, so use lower case everywhere.
- Place the pins, being sure that a is the bottom one. Pin order doesn’t matter logically, it does matter physically and electrically, so you will get errors if you reverse the order.
- Also place an output pin y.

6.5 Create wires

To create wires and connect the components, do the following:

- Choose Create • Wire (narrow) (hotkey: w).
- Click on each component and click again to draw a wire to where it should connect.

It is a good idea to name every net (wire) in a design as this will aid in tracking down a problem later on one of the unnamed nets. Every net in your schematic is connected to a named pin or to power or ground except the net between the two series nMOS transistors.

- Choose Create • Wire name… (l)
- Enter mid or something like that as the name
- Click on the wire to name it

It is unnecessary to label a wire connected to a named pin, power, or ground. If you do, it must have same name as the pin. The names of power and ground are vdd! and gnd! respectively.

6.6 Save Schematic

To save the schematic, Choose File • Check and Save to save your schematic.

You'll probably get one warning about a "solder dot on crossover" at the 4-way junction on the output node. To stop this, go to

- Check • Rules Setup… and click on the Physical tab in the dialog.
- Change Solder On CrossOver from "warning" to "ignored" and close the dialog.
- Then, click *Check and Save* again and the warning should be gone.

Fix any other warnings. A common mistake is wires that look like they might touch but don't actually connect. Delete the wire and redraw it.

Poke around the menus and familiarize yourself with the other capabilities of the schematic editor.

7 Part 1: Schematic Simulation

You will verify your schematic by simulating the design using the Cadence Spectre SPICE simulator.

7.1 Launch ADE-L

Launch the Analog Design Environment (ADE-L) by clicking the "Launch" menu. The first option in the dropdown menu is ADE L. The models for the S130 transistors should already be included, but you should double-check by navigating to "Setup-Model Libraries."

<Tutorial to be completed in class>

8 Part 1d: Create symbol

Each schematic can have a corresponding symbol to represent the cell in a higher level schematic. In this step, you will create a symbol for your 2-input NAND gate.

When creating your symbol, it is a good idea to keep everything aligned to the grid; this will make connecting symbols simpler and cleaner when you need it for another cell.

8.1 Open Virtuoso Symbol Editor

On your nand2 schematic editor window,

- **Go to:** Create • Cellview • From Cellview…
- Choose **cmos_sch** for **From View Name**
- Choose **symbol** for **To View Name**
- Ensure that **Tool / Data Type** is set to **schematicSymbol** and **click OK**,
- Click Ok on the Symbol Generation Options window that will open.

Cadence will then create a generic symbol as shown in Figure 6.

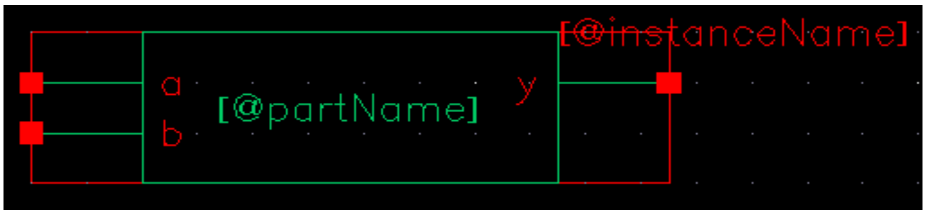


Figure 3: nand2 symbol

8.2 Modify symbol shape

Next, modify this symbol to something familiar and easy to read as shown in Figure 7. Pay attention to the dimensions of the symbol; the overall design will look more readable when symbols are of consistent sizes.

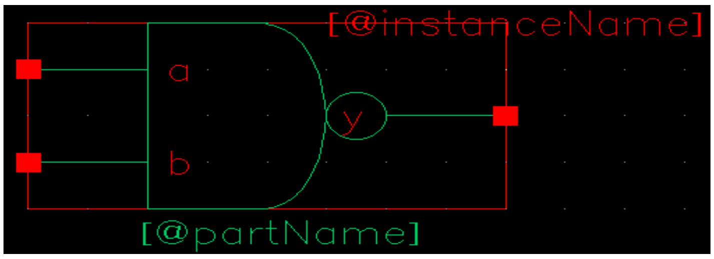


Figure 4: nand2 symbol final version

The green body of the NAND is formed from an open C-shaped polygon, a semicircle, and a small circle.

To form the semicircle,

- **Go to:** Create • Shape • Arc.

Experiment with the arc drawing tool.

To make the polygon

- **Go to:** Create • Shape • Line

To make the output bubble

- Create • Shape • Circle.

Move the lines and terminals around to make it pretty. The Edit • Stretch command may be helpful.

Finally,

- **Go to:** Create • Selection Box… and choose Automatic.

This creates a red box around the symbol that will define where to click to select the symbol when it appears in another schematic.

- Choose File • Check and Save when done.

9 Part 1e: NOT Gate

Next, design a NOT gate named **inv**. Draw the cmos_sch and the symbol, as shown in Figure 8. Make the pMOS width 10 λ (3u) and the nMOS width 7 λ (2.1u). Check and save when done.

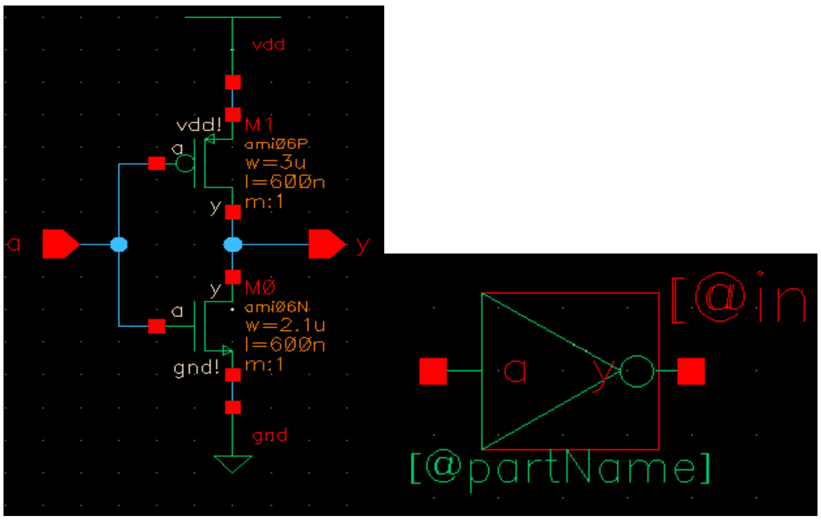


Figure 5: inv cmos_sch and symbol

10 Part 1f: Hierarchical Schematic

A CMOS AND gate consists of a NAND gate followed by a NOT gate. The schematic is constructed by connecting the symbols for the two gates that you have already drawn. This is an example of hierarchical design, reusing pre-existing components to save work.

- Create a schematic Cell named **and2** and enter schematic for the View.

This time, it is **schematic** and not cmos_sch because the cell will instantiate other cells rather than transistors.

- Add instances of **nand2** and **inv** from your **lab1_xx** library.
- Wire the two gates together and **create** ports on inputs **a** and **b** and output **y** as shown in Figure 9. Name the wire between the two gates. In Figure 9, we named it **yb**.

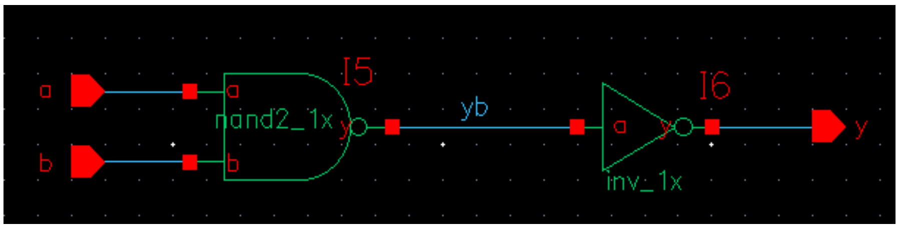


Figure 6: and2_1x schematic

Simulate your and2 gate to ensure it works.

Copy the **nand2.tv** and **testfixture.verilog** files to the run directory for your and2 gate and modify them to contain the proper vectors for the and2 function.

If you encounter netlister errors about connectivityLastUpdated, be sure you have checked and saved the schematics of all of the components.

Make a symbol for the and2. It should be similar to the nand2 but should leave off the output bubble. You may save yourself some time with judicious use of copy and paste or by using the copy command in the Library Manager.

11 Part 2a: Layout

The next step is to draw the layout for the 2-input NAND gate. Figure 10 shows the stick diagram of the layout. Recall that where red crosses green, we get an nMOS transistor and where red crosses yellow, we get a pMOS transistor.

The nand2 has two **nMOS transistors** in **series** between **GND** and **Y**. It has two **pMOS** transistors in **parallel** between **VDD** and **Y**. The blue power and ground busses run horizontally in metal1. A bus means a wire in this context.

The **green n+ diffusion** (**ndiff**) runs **parallel** to and just **above** the **GND** bus. The **yellow p+ diffusion** (**pdiff**) runs **parallel** to and just **below** the **VDD** bus.

The **inputs** run vertically on polysilicon, crossing diffusion to form the four transistors. **Metal1** and contacts are used within the cell to connect the transistors to GND, VDD, and the output Y.

The transistors fit on the same pitch as the metal tracks, so the cell must be 3 tracks wide to accommodate the three contacts on the pMOS transistors.

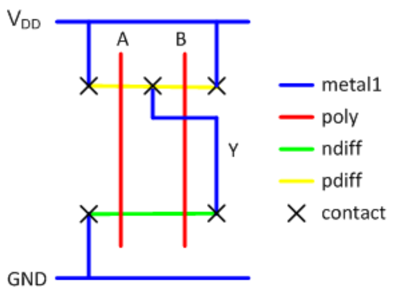


Figure 7: nand2 stick diagram

All cells should be drawn so that they “snap together” like LEGOs when placed adjacently. Therefore, we establish a set of rules for our cell library so they will satisfy design rules alone and also when connected together.

11.1 Design rule

The first step in planning a cell is to consider the metal routing that will go overhead.

Note that in this specification, 1 λ = 0.3u (0.3 µm)

In our system,

- metal2 will run vertically. Each contacted metal2 track has a pitch of 8 λ (4 λ width + 4 λ spacing to the next track).
- Metal3 will run horizontally. Each contacted metal3 track has a pitch of 10 λ (6 λ width + 4 λ spacing).
- To avoid wasted space, each cell should be a multiple of the track pitches in size. Hence, it should be a multiple of 8 λ wide and 10 λ tall.

All the cells in the same library need to be the same height so they can snap together. Thus, the cell height may be set by the tallest cell (generally a complex cell such as a flip-flop) with the widest transistors.

The cell height must also be tall enough to allow all of the horizontal wires in a datapath to pass overhead, which depends on the datapath being built. For the labs, we use a 10-track library with room for 10 horizontal metal3 lines to pass overhead. Based on the past experience, this is sufficient to easily build complex cells and to route interesting datapaths. Hence,

- the cells should be 10 tracks × 10 λ / track = **100 λ tall**.

The width of the cell depends on the connections inside the cell. Conveniently, the metal1 and metal2 pitch is the same (8 λ),

- from the nand2 stick diagram, the cell must be **3 tracks (24 λ) wide**.

Although metal1 can be 4 λ wide, we draw the GND and VDD busses 8 λ wide to carry the larger currents needed in the power supply.

In this context, there is ambiguity about *width* and *length* vs. *width* and *height*. We normally refer to a wire as having a length and width. The length is the long dimension and the width is the skinny dimension.

Thus, wires have a width of 8 and a length of 24 to run horizontally over the whole cell. But a rectangle has a width and a height, with the width being the x dimension and height being the y dimension. Thus, the width of the rectangles is 24 and the height is 8.

The origin of the cell is (x = 0; y = 0). Let us define x = 0 λ as the left side of the cell. Hence, x = 24 λ is the right side of the cell. See Figure 11.

Let us center the GND bus on y = 0. The bus has a width of 8, so it extends 4 above and below the center. Hence, the corners of the GND bus are (0, -4) and (24, 4). See Figure 11.

For a 10-track tall cell, the VDD bus must be 9 tracks higher, or centered at 90 λ. Hence, its corners are (0, 86 and 24, 94). See Figure 11.

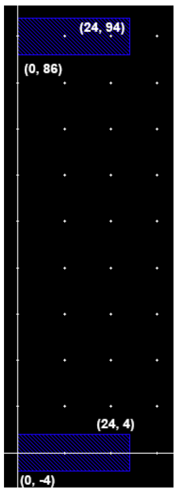


Figure 8: VDD and GND busses

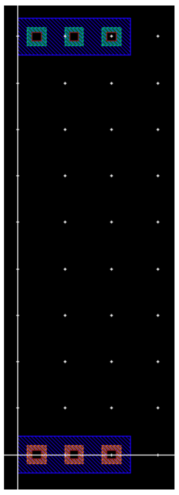


Figure 9: Well and substrate contacts

Each cell has **well** and **substrate** contacts centered under the VDD and GND busses, to tie the bodies of the transistors to the supply voltages. The pitch of the contacts is also 8 λ, so the number of contacts is the same as the number of tracks. These are shown in Figure 12.

The design rules call for a **spacing** of **4 λ** between **metal** and **diffusion**. To make cells snap together nicely, avoid putting any transistors or internal wires (excluding VDD and GND) closer than 2 λ to the left or right boundaries of the cell. Hence, adjacent cells will have a spacing of at least 4 λ between their internal contents.

To keep vertical metal 2 λ from the cell boundary, the leftmost vertical metal track should start at x = 2 and extend to x = 6. In other words, they are centered on x = 4. The track pitch is 8 λ, so the subsequent tracks are centered on x = 12, x = 20, etc.

Polysilicon has a width of 2 and is placed between the metal tracks to make transistors. Hence, it should be centered on x = 8, and x = 16. It has a minimum separation of 3 λ from unrelated diffusion. Hence, to avoid the substrate and well contacts, it must start 1 λ above GND and end 1 λ below VDD, as shown in Figure 13.

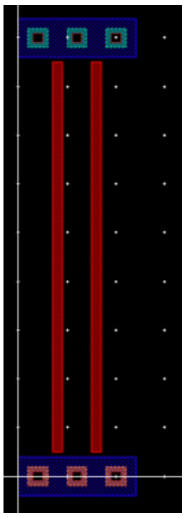


Figure 10: Polysilicon inputs

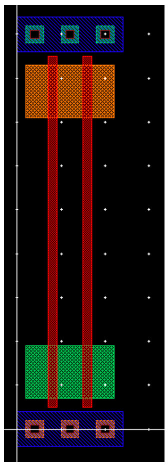


Figure 11: ndiff and pdiff

Polysilicon extends at least **2 λ** past diffusion when forming a transistor. Hence, the **ndiff** must start **3 λ above GND**. The **pdiff** must start **3 λ below VDD**. Like metal, beware of the ambiguity in the meaning of width of diffusion. Because poly is running vertically and diffusion is running horizontally, the width of a transistor corresponds to the height of the diffusion.

In our nand2 cell, the **heights** of the **ndiff** and **pdiff** are both **12 λ** because those are the widths of the nMOS and pMOS transistors in the schematic. In the schematic, we specified the width as 3.6u.

Recall that 1 λ = 0.3u (0.3 µm), so the height is 3.6u × 1 λ / 0.3u = 12 λ. Therefore, we need to draw strips of ndiff and pdiff starting and ending 2 λ from the ends of the cell (to give clearance to the next cell) and starting 2 λ above or below the polysilicon gates and extending vertically by 12 λ, as shown in Figure 14.

The **pMOS** transistors must be surrounded by an n-well by at least 6 λ on each side, and the well should be separated from the nMOS transistors by at least 6 λ. The well should be the same size in all cells to avoid interreference with nMOS in neighboring cells, and also to notch errors when cells are abutted. In our library, we choose to bring the n-well down to y = 40. This gives slightly more space for pMOS than nMOS transistors, which is a good choice because the pMOS are typically wider than nMOS. The n-well extends up to y = 96 to sufficiently enclose the well contacts. To surround transistors by 6, it must extend 4 λ left and right of the VDD bus, as shown in Figure 15.

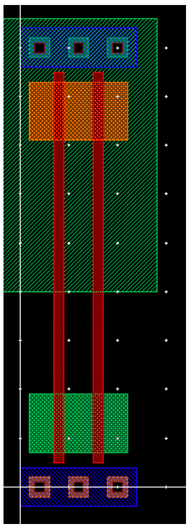


Figure 12: n-well

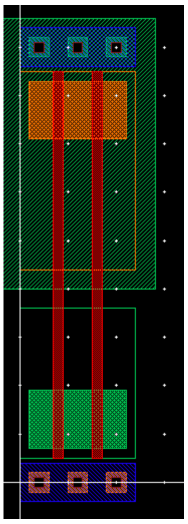


Figure 13: n-select and p-select

The MOSIS design rules call for **n-select** and **p-select layers** enclosing the **ndiff and pdiff** by at least **2 λ**. Each is a rectangle. The n-select starts 1 λ above GND and ends 4 λ below the n-well. The p-select starts 1 λ below VDD and ends 4 λ above the n-well boundary. Both extend the full width of the GND/VDD metal lines, as shown in Figure 16. Hence, when cells are abutted, the active regions also abut, and there are no notches.

All leaf cells resemble Figure 16, with power and ground, n-well, well and substrate contacts, horizontal strips of diffusion, and vertical polysilicon inputs. Different cells will have different widths, but the template is otherwise the same.

The cells are customized by adding metal1 and contacts to connect the power, ground, and inputs and outputs. Most cells can be built using no metal2 or metal3 wires inside the cell, so wires can easily be routed over the cell in these layers without interference. Stick diagrams show poly running purely vertically, but we often bend the poly to bring transistors closer together if there is no contact required between them. This is worth the trouble because it reduces the capacitance, improving speed and power consumption.

11.2 Build the layout

To start a layout of a 2-input NAND gate,

- First, **create a new cellview** for nand2 called **layout**. It should be of type layout and always open with **Layout GXL**.[2]

Click OK to accept nand2 cmos_sch when prompted about the Source View Definition. You may also get some warnings about Assura.

[2] If you use Layout L instead of GXL, you may have LVS errors later.

Your goal is to draw a layout exactly like the one shown in Figure 17. Neatness and precision are imperative to get a good layout. Keep in mind the principles described above so the reasons for the dimensions are clear.

The University of Utah technology file is configured on a half-lambda grid, so grid units are 0.15 µm. Take care that everything you do is an integer multiple of λ so you don't come to grief later on. The heavy dots are on a 10-λ grid and the smaller dots are on a 2-λ grid. **Also, units are listed in microns, so you will have to mentally convert 1 λ = 0.3 µm.**

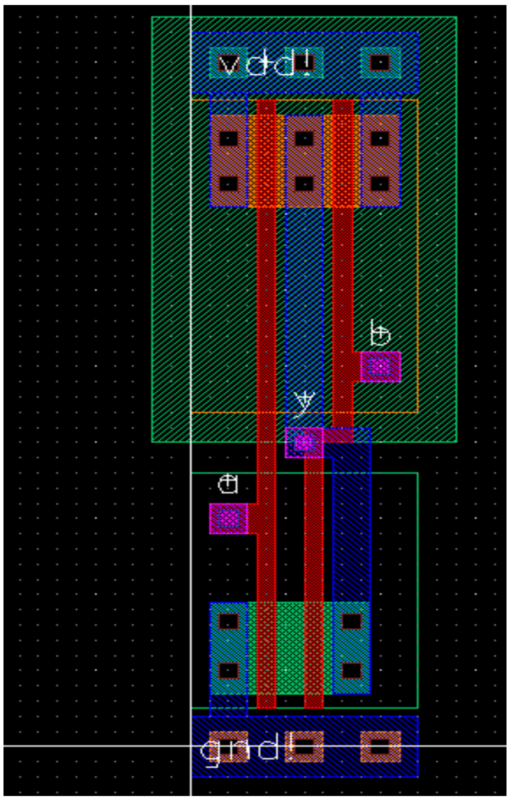


Figure 14: nand2 layout

11.2.1 Familiarize with the Layout editor

The layers pane is on the left side of the Layout editor window. For the labs, you will need **nwell, nactive, pactive, nselect, pselect, poly, metal1, metal2, metal3, cc (contact cut), via, and via2.**

In this design process, nactive and pactive stand for n+ diffusion and p+ diffusion, respectively. They must be surrounded by a rectangle of nselect or pselect, respectively; in a cleaner flow, the select layers might be automatically generated.

By default, the Layout editor snaps to a 0.5 λ (0.15 µm) grid. Getting off the lambda grid will cause you grief, so start by changing this.

- Chose Options • Display… (*e*) and set the X and Y snap spacing to 0.3 (the units are microns).
- Set the major grid spacing to 3 microns (10 λ)
- Set the minor grid spacing to 0.3 micron (1 λ).

Using the save option, you may save your display settings to a file and reload it whenever you restart Cadence or the Layout editor using the load option.

Some useful menu options when drawing the layout are:

- Zoom in and zoom out. There are several zoom commands under the window menu.
- Pan with the arrow keys.
- To create a rule for measurement, use Tools • Create Measurement (k).
    - Use the ruler to measure distances
    - Use Tools • Clear All Rulers to eliminate them when you are done.
- To stretch a drawing, use Edit • Stretch (s).
- To change the size or layer of a shape, use Edit • Properties (q)
- To merge multiple selected rectangles on a given layer into a single convenient polygon, use Edit • Basic • Merge (shift + m).

11.2.2 Power routing

Start by placing your power and ground busses in metal1.

- Click on metal1 in the layers pane. **Choose** Create • Shape • Rectangle.
- Draw a rectangle from (0, -1.2) to (7.2, 1.2). This is a rectangle 8 λ wide and 24 λ long, centered on the y-axis with the left edge on the x-axis.
- Draw a second rectangle 90 λ (27 microns) above the first.

11.2.3 Draw n-active and p-active

Next, draw the n-active and p-active. According to the schematic, the transistors are 12 λ wide, so the active rectangles should be 12 λ tall.

- Each should start 3 λ away from the ground or power bus.

11.2.4 Draw poly gates

Next, draw two poly gates.

- They should be 2 λ wide and extend 2 λ beyond the transistors in each direction.
- The gate on the right must bend because the spacing between the series nMOS transistors is only 3 λ, while the spacing between the parallel pMOS transistors is 6 λ. You can make the bend by drawing the poly wire as three separate rectangles.
- Add metal 1 wires to connect the transistors to power, ground, and the output.
- Then, add 2 × 2 λ contacts (cc) between metal and the n-active.

In cells like this where the diffusion is wide enough, placing multiple contacts reduces the series resistance and increases reliability in case one contact is malformed during manufacturing. The exact placement is unimportant, however, each contact cut must be separated by 3 λ from its neighbors. You minimize resistance by placing as many contact cuts as possible and keeping the distance from any bit of diffusion to the contact as small as possible.

11.2.5 Substrate and well taps

Recall that substrate and well taps are required to keep the diodes reverse biased. We will place taps under the power and ground wires on 8 λ centers.

- Draw three 4 × 4 λ squares of p-active underground, starting 2 λ from the edge.
- Place 2 × 2 λ contacts on the center of the taps to connect them to metal.
- Do the same with n-active under the power wire.

11.2.6 n-well and select layers

Now is a good time to draw the n-well and the select layers.

- The n-well should extend from 40 λ to 96 λ vertically and should extend 4 λ beyond the edge of the cell in both directions (-4 to 28 λ).
- The p-select should extend from 44 to 85 λ vertically and should be as wide as the power line (0 to 24 λ).
- The n-select should extend from 5 to 36 λ vertically and should also be as wide as the power line.
- **Another** rectangle of n-select must be drawn exactly covering the power line to surround the n-well taps.
- A rectangle of p-select should be drawn covering the ground line to surround the substrate taps.

11.2.7 Routing grid

Later, we will connect cells using vertical metal2 wires and horizontal metal3 wires.

- Metal2 is drawn on an 8 λ grid (width = 4 λ; spacing = 4 λ).
- Metal3 is drawn on a 10 λ grid (width = 6 λ; spacing = 4 λ).

Figure 18 shows the routing grid superimposed on the cell. The metal2 wires will run in the same columns as the well and substrate contacts, centered 4, 12, 20 λ, etc. right of the cell origin. The metal3 wires will run 0, 10, 20, 30, 40, 50, 60, 70, 80, and 90 λ up from the cell origin. Inputs and outputs should be placed on 4 × 4 λ squares of metal2 in the appropriate columns. For example, *a*, *b*, and *y* are all on this grid.

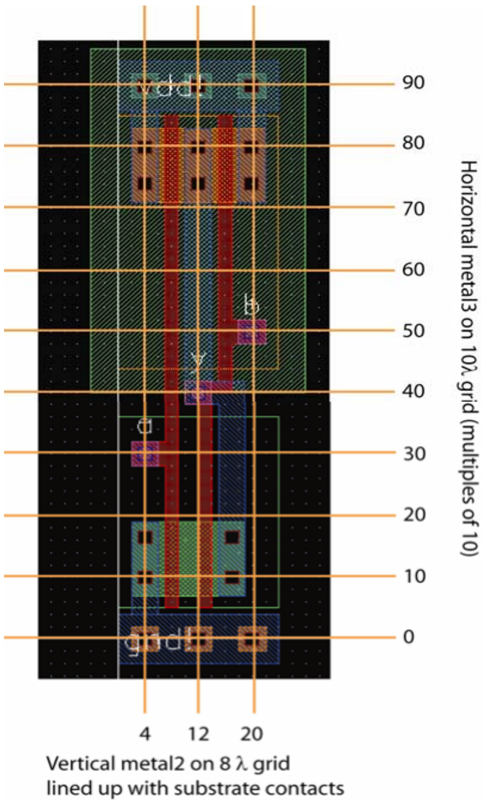

vdd!
b
y
a
gnd!
90
80
70
60
50
40
30
20
10
0
Horizontal metal3 on 10λ grid (multiples of 10)
4
12
20
Vertical metal2 on 8 λ grid
lined up with substrate contacts

Figure 15: Wiring grids and pin locations

11.2.8 Input and output pins

To create the inputs, you will need a stack going all the way from metal2 down to polysilicon: metal2, via, metal1, cc, and poly.

- The metal can be 4 × 4 λ, and the contacts/vias 2 × 2 λ, but the poly may need to extend a bit further to reach the main polysilicon lines.
- The output is already on metal1, so you only must add metal2 and a via.

Finally, place pins to define where the cell connects to other cells at the next level of the hierarchy. Select metal2 in the layer window.

- **Choose** Create • Pin…
  - Set the terminal name to a.
  - Make sure the following are chosen
    - Connectivity: weak
    - Pin Shape: rectangle
    - I/O Type: input.
  - Turn on Create Label.
  - Then, draw a 4 × 4 square of metal2 representing the pin on top of the metal2 already present for input a.
- Do the same for b.
- Do the same for y, but set it as an output.
- Also create large metal1 **inputOutput** pins called **vdd!** and **gnd!**. The pins should cover the entire power and ground busses. The "!! indicates a global net and is pronounced "bang."

12 Part 2b: Hierarchical Layout

To illustrate hierarchical layout, we will build an AND gate using a NAND and an inverter. Figure 22 shows what the completed AND gate should look like. Observe how the cells abut nicely, with no notches in the n-well or active, and a spacing of at least 4 λ between the metal and diffusion within each subcell.

12.1 What you should do

- First, draw the inverter layouts. Be sure that the nMOS and pMOS widths match the schematic. Check that it passes DRC and LVS.
- Next, create a new cell layout for the and2. Choose
  - Create • Instance and place the nand2 and inv layouts.
    - They show up as red bounding boxes.
    - Use Options • Display to look inside the box.
    - One option is to set the Stop Display Level to a big number (such as 10) to look inside cells.
    - But a more convenient option for us is to click on the Instance Pins box to show just the pins.
- Move the cells so that their power and ground busses abut.
- Place a 2 × 2 λ via2 on the **y** pin of the nand2 and on the **a** pin of the inv.

- Then, draw a metal3 wire connecting the two gates. Metal3 is thicker and has sloppier tolerances, so it must be 6 λ wide and extend 2 λ beyond each via2.

At this point, your design should resemble Figure 23.

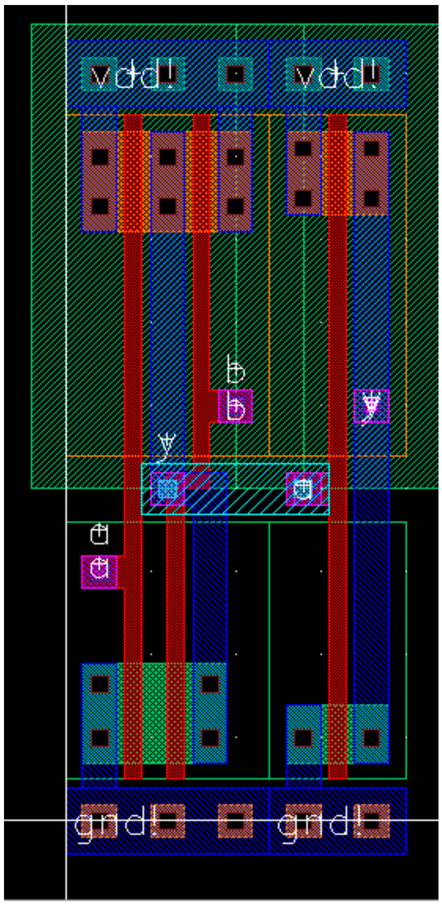


Figure 16: and2 layout

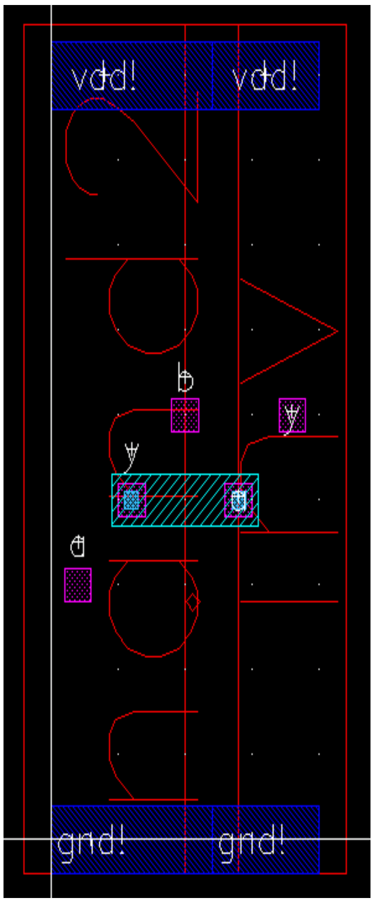


Figure 17: and2 hierarchical layout view

Now you will need new pins at this level of hierarchy for a, b, y (from the inverter output), **vdd!** and **gnd!** The inputs and outputs should be 4 × 4 metal2 squares on top of the pins from the lower level of hierarchy, while the power and ground should be metal pins covering the entire power/ground busses.

Check your design with DRC and LVS and fix any errors.

13 For Independent Practice

Now that you've learned the custom cell design flow, try it again yourself. Design schematics, symbols, and layouts for the following two gates. Simulate the schematics to prove that they work correctly. You will have to create your own System Verilog testbenches and test vector files. Check DRC and LVS. You will use your cells to complete the ALU in the next lab.

- 2-input NOR gate: nor2 (use transistor widths of 8 λ for the nMOS and 16 λ for the pMOS)
- 2-input OR gate: or2 (using a nor2 and an inv)

14 What to Turn In

Please provide a hard copy of each of the following items. You may wish to use the Windows Snipping Tool or the Mac Grab program to take screenshots of your designs.

1. Please indicate how many hours you spent on this lab. This will not affect your grade, but will be helpful for calibrating the workload for the future.
2. A printout of your nor2 schematic.
3. A printout of your nor2 layout.
4. A printout of your or2 schematic.
5. A printout of your or2 layout.
6. A printout of your or2 test vectors. Does the schematic pass simulation?
7. Does the or2 layout pass DRC? LVS?