Sample IC Design Using Cadence

Tools

Bilge Karaçalı

1 Introduction

This tutorial is intended to provide a step-by-step guidance to designing an integrated circuit using Cadence tools. With a sample design, it is hoped that students in VLSI design classes will be able to familiarize themselves with Cadence software.

2 Running Cadence

The Cadence software is supported by the AFS. In a command prompt, just type add cadence. The response you will get will look something like this:

---------------------------------------
Cadence Design Systems EDA Tools
---------------------------------------

Use the following commands to start Cadence applications:
---- Online Help ----

openbook       Cadence online documentation

---- Design Framework II (IC4.4.5) ----

icde &        Basic digital and analog design entry [12 MB]
icds &        Front end design [14 MB]
icms &        Front end analog, mixed signal and microwave design  [17 MB]
icca &        Cell based chip assembly [32 MB]

layout &      Basic layout with interactive DRC [14 MB]
layoutPlus &  Basic layout plus automated design tools [22 MB]

msfb &        Mixed-signal IC design [25 MB]
icfb &        Front to back design [38 MB]

---- Stand-alone tools ----

spectre <input file> &  Spectre circuit simulator
verilog <input file> &  Verilog HDL simulator
signalscan &       Analog/digital waveform display
You can then choose the program you need to run. I would personally advise you to run icfb since it gives you access to everything you could possibly need.

3 2-Input AND Gate

The design process of an IC circuit starts with a schematic design of the components. A two-input AND gate can be composed of two components: one two-input NAND gate, and one inverter. We will now do the schematic design of the two-input AND gate.

3.1 Creating a new library

First, we will create a new library for all of our designs. Bring up the Library Manager. Go to the File menu and choose new .. library. Type a name for the new library, say Mylibrary, and a path for it. Note that the path needs to exist for the Library Manager to put a library in it. Then, among the (three) choices for the technology library, click on Attach to an existing tech library and select the proper technology library from the list (e.g. TSMC 0.40u CMOS035). Click OK and observe that the new library appears in the Library list.

Now, we will create our first design cell in the library we have just created. Click on the newly created library. Go to File menu, and this time, select new .. cellview. In the Cell Name, write mynand, since we will design a nand gate, and choose Composer - Schematic from the Tool list. Note how the View Name turns into schematic. Click OK. We are now ready to build the
schematics of our 2-input NAND gate in the schematic window that appears.

3.2 Schematic design using Composer

In our design, we will need 2 NMOS transistors, and 2 PMOS transistors. Go to add pulldown menu, and select instance. This will bring up two windows, an Add instance window and a Component Browser. In the Component Browser, choose NCSU Analog Parts from the Library menu. Go to N Transistors, and click on nmos. Note how the Add instance window is updated. This loads an NMOS transistor to your mouse. Go to the schematic window of our NAND gate, and put two NMOS transistors in one column. Similarly, bring up two PMOS transistors, and place them in one row, right above the NMOS transistors. While we are at it, go to Supply Nets and bring up one vdd and one gnd. Now that we have all the instances we need for the design, you may hit the Esc key to get rid of these two instance windows. This is equivalent to clicking on the Cancel button.

Next, we need to add pins for our inputs and output. Go to the Add menu, and select Pin. This will bring up an Add Pin window. Let us call our two inputs A and B. In the Pin Names area, type A. Note the direction is input and that the mouse is loaded with our first input pin. Place the pin in the design window. Similarly, place another input pin named B. Now, create an output pin named AnandB by filling the name area and choosing output from the direction list.

After connecting the instances, pins and supply nets\(^1\), and modifying the

---

\(^1\)I trust you can figure out how to draw wires (I prefer narrow), move objects around, rotate them etc..
transistor sizes (select the object, go to Property .. object), our design looks like Figure 1. Go to Design menu, and click on Check and Save. Observe the action log in the icbf log window:

Extracting "mynand schematic"
Schematic check completed with no errors.
"Mylibrary mynand Schematic" saved.

In a similar way, design the inverter, in a separate cellview named myinv with inputs A and notA. The inverter should then look like Figure 2.

3.3 Hierarchy in schematic design
We will now build the schematic design or our 2-input AND gate. First, we need to extract the symbol views of the NAND gate and the inverter. Go to
Figure 2: The schematic design of an inverter

Design .. Create Cellview .. From Cellview. Make sure that the Tool / Data Type shows Composer - Symbol. Hit OK and observe the symbol view for the schematic design appear with designated input and output pins. Repeat the same procedure for both designs.

Since we have completed the schematics of our necessary components, it is time for us to test them. We will do that using the Analog Artist. But first, we need to construct the final schematic if our 2-input AND gate.

Create a new cellview (in schematic) and name it myand. Bring in one instance of the NAND gate and one instance of the inverter we have just designed. Then, add the input and output labels. The finished schematic should look like Figure 3. Make sure you verify and check your design, and extract a symbol view.
3.4 Testing the final schematic

Now, create another cellview (also in schematic), and call it `test_myand`. Import the `myand` instance we have just created. Also import one `vdd`, one `gnd`, one `vdc` and two `vpulse` instances. Arrange the components so that `vdc` is between `vdd` and `gnd` and has 5 volts (go to properties, etc.), `vpulse`’s go to A and B, with connections to `gnd` from the negative end. The `vpulse`’s need to be arranged so that one has twice the others period, both with \( v1 = 5 \text{ V} \) and \( v2 = 0 \text{ V} \). Make sure that you fill in all the time parameters. The circuit should now look like Figure 4. When you do a check and save, cadence will complain that one of the outputs is not connected to anywhere, but we will live with that. It is also a good idea to label the wires incident with the inputs and outputs of our design.

![Diagram of test circuit for 2-input AND gate](image)

Figure 4: The schematic design of test circuit for the 2-input AND gate

Next, bring on the Analog Environment from the Tools menu. Go to
Outputs .. To Be Plotted .. Select On Schematic, and click on the wires incident with A, B and AandB. observe that they appear in the Outputs window of the Analog Environment. Now, go to sf Analyses .. choose. In the window that appears, check tran and fill out From, To and By as 0, bigger period of the two pulse generators, and a small time step, like 1 nsec, respectively. Then, hit the green light to run the simulation. The resulting plots which should look like Figure 5 after some modifications on the plotting options tell us that our 2-input AND gate functions correctly (I assume you all know what the correct functioning of a 2-input AND gate is).

3.5 Layout using Virtuoso

Now that we know that our design strategy is accurate (since the schematic functioned accurately), we can start doing the layout of our components. Go to File .. New .. Cellview. Since we will do the inverter layout, type myinv in the cell name window, and select Vistuoso from the Tool menu. Observe that the View Name converts into layout. Hit OK, and we are ready to go in the new design window.

Hit i to bring up the instance add window, and hit the Browse button. Select the NCSU TechLib tsmc04 4M2P. Click on pmos and observe that it loads onto the mouse. Place the PMOS transistor layout in the window. Similarly, place an NMOS transistor in the layout below the PMOS transistor\(^2\). Make sure that you leave some space in between to draw polysilicon and metal lines etc.

\[^2\text{Most probably, you will be seeing only two boxes which say nmos and pmos. Hit shift-f and see what happens.}\]
Figure 5: The Analog Environment simulation result of the 2-input AND gate
We will create the connections by drawing rectangles. From the LSW window, select poly, select rectangle from the layout window (either from the create menu, or by hitting r or simply by clicking on the button right above the ruler at the left-hand-side of the layout window). If the polysilicons do not align properly, you can always move the transistors so that they do. So now, the gates are connected. Similarly, connect the drains by a metal2 rectangle. Note that since the drains are equipped with a metal1 via only, you need to put metal2 - metal1 via’s on them in order to be able to connect them using a metal2 rectangle. Go to Library Browser of create instance, and find m2 m1: this is the via between metal1 and metal2 we are looking for. Select it, and place one on top of each drain. Now we can connect the drains using a metal2 rectangle. We will use metal1 rectangles for input and output. Draw two metal1 rectangles, and connect one to the polysilicon that connects the gates, and the other one to the metal2 that connects the drains. Once you are done, label the input and output by selecting text from LSW and then clicking on the button that says [abcd].

The last thing we need to finish the layout is the VDD and ground rails. The VDD rail requires an nwell underneath, and the ground rail requires a pwell. Since the substrate is pwell, we don’t need to worry about it; but we do need to lay a nwell section right above the PMOS nwell. The two nwells need to touch each other to make sure that the bulk connection of the PMOS transistor is indeed VDD. Next, bring on the instance adding window, select ntap and put a few ntaps in the nwell you have just laid down. These will be the VDD rail. In order to connect each tap, cover them with a layer of metal1, and label the metal1 rectangle as vdd. Repeat the same procedure
for the ground rail using ptaps right below the NMOS and label it gnd. The completed layout should look like Figure 6. Save your design.

![Figure 6: The layout of the inverter](image)

### 3.6 Layout checks: DRC and LVS

Before we proceed further, we need to make sure that our design does not violate any of the design rules defined by the TSMC 0.40 micron process. Go to Verify .. DRC. Just hit OK and wait for the DRC to finish. Watch the cds.log output in the initial icbf window. If your design has DRC violations, they will appear boxed in the layout window, and the log file will have the descriptions. The most common errors are rectangles being too close, too narrow, etc.. You need to get rid of those errors (and warnings) before you proceed. It is also wise to do the DRC check in the course of doing the layout.
in order to avoid a mess of DRC errors at the end.

Now that our layout has passed the DRC check, we need to make sure that it is consistent with the schematic design. This step is not crucial for small designs like an inverter, but for a component with thousands of transistors, you really do not want to verify your layout some other way. The advantage of comparing the layout with the schematic design is the following: you know that the schematic worked. If you can verify that the layout will do the same thing as the schematic, you will be guaranteed (really? will the sun come up tomorrow too?) that the layout will operate properly. The comparison of the schematic design and the layout is done by the LVS check. The LVS check however requires that you create and extracted view of your layout. Go to Verify .. Extract and hit OK. The new window that comes up will contain the extracted view of your design (hit shift-f and verify that cadence recognized your transistors). In that window, go to Verify .. LVS. In the LVS window, fill in the opposite areas for Library, Cell and View. Note that the first two will be identical for the first two areas (thanks to our library structure), but one of the last areas will be schematic whereas the other one will be extracted. Once you have these filled, hit OK and wait for the LVS to finish. You need to get the LVS to succeed before you proceed. The most common errors that cause LVS to fail are mismatches between the input and output labels of the schematic design and the layout. LVS check with no errors tells us that we did the layout correctly.

Tracing the same steps as above, do the layout of the 2-input NAND gate. The layout should look like Figure 7. Make sure that your layout satisfies both DRC and LVS checks.
3.7 Hierarchy in layout

We will now do the layout of our 2-input AND gate using the verified layouts of our inverter and 2-input NAND gate. Create a new layout view for our AND gate (in the previous AND cell) selecting Virtuoso as the tool. Bring in the instance layouts of the inverter and the NAND gate we have just finished and verified. Connect the instances properly, do some adjustments, and relabel the inputs and the outputs properly, including vdd and gnd since lower hierarchy labels are not available to higher hierarchies. Your layout should eventually look like Figure 8. Save your design, run the DRC and LVS checks. As the design passes these checks with success, we are ready to extract the netlist of our 2-input AND gate layout.

Bring up the extracted view of the AND gate cell. Go to Tools .. Analog
Figure 8: The layout of the 2-input AND gate

Environment. First, make sure that the simulator is hspiceS in Setup .. Simulator/Directory/Host. Then, go to Simulation .. Netlist .. Create Final. Analog Artist will then create the netlist for the layout to be included in an hspiceS file. The netlist you get should be similar to the one below.

* # FILE NAME: /AFS/UNITY.NCSU.EDU/USERS/.../CADENCE/SIMULATION/
* myand/hspiceS/extracted/netlist/myand.c.raw
* Netlist output for hspiceS.
* Generated on Nov 16 01:26:50 2000
* File name: Mylibrary_myand_extracted.S.
* Subcircuit for cell: myand.
* Generated for: hspiceS.
* Generated on Nov 16 01:26:50 2000.

M4 2 B VDD VDD TSMC35P L=400E-9 W=900E-9 AD=900.000018087821E-15
\[ \text{AS=900.00018087821E-15 PD=2.90000002678426E-6 PS=2.90000002678426E-6 M=1} \]
\[ \text{M6 2 A VDD VDD TSMC35P L=400E-9 W=900E-9 AD=900.00018087821E-15} \]
\[ \text{M8 AANDB 2 VDD VDD TSMC35P L=400E-9 W=1.8E-6 AD=1.80000003617564E-12} \]
\[ \text{M10 AANDB 2 GND GND TSMC35N L=400E-9 W=600E-9 AD=759.99973110742E-15} \]
\[ \text{M12 2 B 1 GND TSMC35N L=400E-9 W=600E-9 AD=759.99973110742E-15} \]
\[ \text{M14 1 A GND GND TSMC35N L=400E-9 W=600E-9 AD=759.99973110742E-15} \]
\[ \text{+AS=759.99973110742E-15 PD=3.00000010611257E-6 PS=3.00000010611257E-6 M=1} \]
\[ \text{.lib "/ncsu/cadence/local/models/hspice/tsmc35/lmm0355v1.1" TT} \]
\[ \text{.lib "/ncsu/cadence/local/models/hspice/tsmc35/lmm0355v1.1" NMOS} \]
\[ \text{.lib "/ncsu/cadence/local/models/hspice/tsmc35/lmm0355v1.1" TT} \]
\[ \text{.lib "/ncsu/cadence/local/models/hspice/tsmc35/lmm0355v1.1" PMOS} \]
\[ \text{* Include files} \]
\[ \text{* End of Netlist} \]
\[ \text{.TEMP 25.0000} \]
\[ \text{.OP} \]
\[ \text{.save} \]
\[ \text{.OPTION INGOLD=2 ARTIST=2 PSF=2} \]
\[ \text{+ PROBE=0} \]
\[ \text{.END} \]

You can now use hspice to simulate your design by including that netlist you have created.

15