LAB TUTORIAL #3 – MANUAL LAYOUTS

A. Tino, Foreword: F. Campi

Background: ENSC450 focuses on building computation with CMOS logic. While majority of this course focuses on designing logic in VLSI, we have observed that all complex multi KGate logic may be simplified to synthesis, placing, synchronizing (clock tree synthesis), and interconnecting (routing) basic gates. These gates are simple CMOS circuits, implemented on a silicon substrate with the appropriate doping areas, SiO2 insulator layers, and metal/polysilicon interconnects.

In tutorial #2, we designed and simulated standard cells with HSpice. These CMOS circuit cells were designed at the Spice level however, which is an abstraction. We may realize this abstraction by physically designing these transistors on silicon as well.

Objective: Starting from a reference CMOS circuit, this tutorial will guide you through the layout of a simple inverter cell. We will learn how to draw diffusion areas (n+ and p+) per transistor, the Polysilicon gate, and the metal connection that routes IO signals and Vdd/Gnd to cell pins. Thereafter, we will extract a new circuit from the layout, and verify that our layout design corresponds to the actual circuit (referred to as **Layout-Versus-Schematic (LVS)** check). We will then have the correct information pertaining to cell area, pin position, and parasitic RC values which we guessed previously at the circuit level.

Tools: We will be using the **Cadence Virtuoso** IC development suite in this tutorial. This is the most utilized manual layout platform both in industry and academia. Virtuoso should not be confused with Cadence Innovus, which is a different EDA tool for implementing an **automated** layout from a VHDL description. The fundamental differences between Virtuoso and Innovus suite are:

a) Virtuoso is used for the implementation of small designs, based on circuit (Spice) representations, while Innovus is used for implementing large designs based on HDL representations.
b) Virtuoso requires the user to manually specify design area and perform manual metal interconnections (there are also ways to automate such routing but they are sub-optimal). Innovus only requires the user to specify the floorplan and constraints, and performs placing and routing automatically based on this information.

Most of the layout work in both academia and industry is completed manually, click and dragging silicon space to desired areas (i.e. diffusion, wells, contacts, polysilicon). This requires very specific training and expertise that is beyond the scope of this introductory VLSI CMOS course. Nevertheless, you must be aware of these basic steps and guidelines for such layout activities.

Basic Layout Steps

1. **Schematic Entry** – Based on the knowledge we have acquired so far in the course, our desired circuit/standard cell will be realized by means of an abstract schematic composed of nmos and pmos transistors. We will choose reference transistor parameters and try to match them in our layout. If this is not possible, we will return to the schematic to refine the values and match them in the layout. We use the schematic to verify that our layout represents the same circuit.

2. **Layout** – Using the schematic as a reference, the desired circuit will be implemented by designing the mask and patterns on the silicon space. We may complete this task by drawing polygons and/or rectangles of a given layer type. The interaction between polygons/rectangles describes the transistors and behaviour, and should match the schematic and its interconnections.

3. **Design Rule Check (DRC)** – Layout design is driven by technology specific design rules that guarantee that the drawn polygons/rectangles behave as expected, and that the lithography mask will be able to correctly reproduce the desired shapes. Rules describe the distance between layout shapes, and their legal dimensions for the given technology node.

   DRC specifically is a design step which verifies the spacing of our polygons/rectangles. If DRC is not successful, we must go back to the layout and fix the DRC violations. This step is very time costly and takes up the largest part of the design efforts: you will feel very relieved when your complete design undergoes a successful DRC check.

4. **Layout Versus Schematic (LVS)** - Although your design is compliant with the design rules, it must also implement the same physical behaviour as your schematic. LVS is an automated formal check that verifies that the geometries realized in the layout correspond to the schematic drawn, in all aspects. LVS is divided into two steps: 1) **extraction** where the layout shapes are analyzed in order to determine which electrical devices they represent, and 2) **LVS,** where the extraction of the layout is compared to the reference schematic, verifying that the user has implemented the desired layout. If there is no match, the designer may fix the layout, or change the schematic so that the layout matches the constraints.

Note that design rules and layout coordinates are in multiples of 0.005 microns in this technology (the manufacturing grid value). This is specified by the technology supplier.

1. Environment Setup

Before we begin designing our cells, we must set up our environment. As the case with the previous tutorials, ensure you use a new terminal which is configured to utilize these specific tools required of the activity. For our layout activities, we will be using the 180nm TSCM technology provided by CMC Microsystems.

Open a terminal:

```
source /etc/profile.d/ensc-cmc.csh
```

```
source /CMC/setups/CDS_setup.csh
```

Cd to your ensc450 course directory. In this directory:

```
mkdir CDS
```

```
mkdir CDS/LVS
```

```
cp /CMC/setups/cds.lib    CDS
```

`cd CDS` *****ensure you are ALWAYS in this folder before proceeding to launch Virtuoso**

2. Opening Virtuoso

In this section, we will use existing Skill code to design an inverter. You can start the Virtuoso Layout environment from the CDS directory with the command:

```
startCds -t cmosp18 -w .
```

-w specifies the working library, that should be found in your CDS directory (don't forget the . in the command above after -w indicating launching Virtuoso from the current directory)

-t specifies the technology we are using, in this case cmosp18 which is **TSCM 180nm**

Once the application launches (**THIS WILL TAKE A LONG TIME**), you will observe the following:

- **Icfb** console or command input window (CIW), with a blinking cursor on the command line. This has a menu function ribbon, a scrolling log window, and command line input.

- **Library Manager (LM)** window. If this does not launch, press F6 or click Tools > Library Manager. The LM window will display libraries included in your cds.lib file. If you left click a given library name, you will see a list of **cell views**. The vst_n18 seen is a standard cell library, where each cell may be viewed in several ways, including symbol, schematic, and layout.

    - If you do not open a layout cell view, you will only see an **abstract** view which substitutes the real layout and its contents (usually the case when submitted to the foundry for fabrication) - You will only see metal1 (blue) layer and a few pin names in white. When opening the CIW window for a standard cell, you may ignore the warnings.

- **What's New** window – just happily ignore and close this window.

Ensure you are in your ensc450 folder when you launch Virtuoso to ensure you can find the CDS folder.

Schematic Design

Next, we will create our own library from the icfb console:

In the icfb window, select **File > New > Library**. A pop-up window will display. Enter your <**libname**> in the field and click **Ok** (good idea to name this as **inverter**). Select **Attach to an existing techfile**. Press OK. The next pop-up will prompt for the technology. Select **cmosp18fix3.**[1] Select OK.

3. Designing Schematics

Virtuoso is a complete design suite, that supports the user in every step of circuit design. We should keep in mind that Virtuoso is a platform of reference also for analog design, however the complexity of the circuits and criticalness of Spice level design is much higher than our digital cells case.

It is possible and often convenient to edit a Spice schematic in Virtuoso for two main reasons:

1) The schematic can be used as a reference during layout to drive the layout implementation.

[1] If you cannot find the cmosp18fix3 library, recopy the cds.lib file into your CDS folder

2) At the end of the layout activity, a schematic will be extracted from the layout. A specific utility call LVS is used to compare the original schematic to the extracted layout circuit. This ensures that we do not have any functional errors in our layout.

This section will guide you on designing an inverter cell **schematic**.

In Virtuoso, select **File > New > Cellview**. Change the Library name to **inverter** (or the library name you specified previously), and in Cell Name **inv**:

- Library Name = <Your Library Name>
- Cell Name = <Your Cell Name, your choice>
- View Name = **schematic**
- Tool = **Composer – Schematic**

Press **OK**. You should then get a new window to draw your schematic.

Basic Navigation Rules for Schematic and Layout Editing (Important)

- In all Virtuoso tools, press **ESC** anytime you need to **exit** from a specific functionality, and return back control to the main window.
- You can zoom to a selected area by right clicking the mouse. Zoom out with SHIFT-Z. For zoom-to-fit, type **f**
- To move an instance (or a layout feature), select the instance and press **m** (which may take a while – check bottom left of the screen for mode Virtuoso is in). A shadow box will appear to guide you.
- To copy an instance (or layout feature) press **c** and use the same methodology as move.
- To delete a wire/instance or layout feature, select the instance and press **delete**

Adding Transistor Instances

In the new window, we will now add our inverter components starting with the PUN **pmos**. Type **i** (or **Add > Instance** in the GUI) which will launch a library browser. Browse for the **cmosp18** library (**not** cmosp18fix3). Select pfet. Under the **View** heading, set to **symbol**. Adjust the **width** in the "**Add instance**" window to **500nm**. We will not change L in this tutorial, and we do not need to change any other physical parameter. During the layout process, if we realize we can not physically fit the W/L we chose, we may go back to the schematic and fix the dimensions there.

Move the mouse over the drawing window, and left click to place your pmos transistor. Right-click the transistor to rotate if required. **Double check** that the width reflects 500nm on the transistor which you placed on the schematic. If the transistor reflects a different width, select the pfet and press 'q'. You may change the width in the pop-up window.

If you want to exit from the instance menu (or any other functionality), press ESC. Please be patient as the tools may be VERY slow at times.

Repeat the same procedure for the **nmos**: Select the nfet instance, place it, and set the desired width to **500nm** as well (we will be making life easier and sticking to one width dimension, however noting that we will be making a skewed inverter). Press the Esc key to exit.

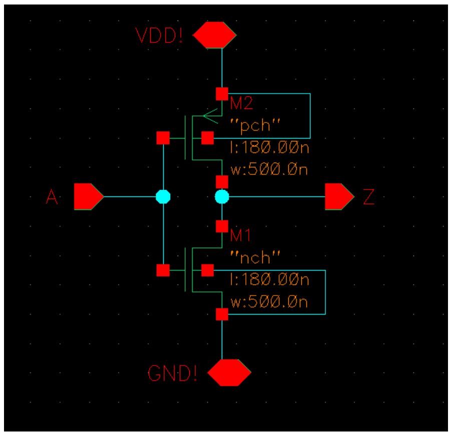


Fig. 1: Inverter Schematic

Adding Pins and Power Rails

Pins are added in a similar manner to instances, where wires are used to connect the pins to their appropriate instances and terminals. Power rail connections are added by using the specific pin names VDD! and GND!, and the corresponding power rail pin direction **InputOutput**.

Press **p**, or from the GUI select **Add > Pin** to add pins to the schematic. Once you left-click the schematic, your pin will be placed. Place the following four pins on the schematic:

- Pin Name = **VDD!**
- Direction = **InputOutput**

- Pin Name = **GND!**
- Direction = **InputOutput**

- Pin Name = **A**
- Direction = **input**

- Pin Name = **Z**
- Direction = **output**

Press cancel in the GUI, or ESC once you have completed this task.

Wire Connections

When you move the mouse over a Pin (represented as a red square), the pin will be highlighted. Move your pins the appropriate places on the schematic, as suggested in Fig. 1. Similarly, when hovering the mouse over a transistor symbol, it will also be highlighted so you may move the instance to a different location. To create connections between instances, press **w** (wire).

Click on the source pin, and draw a wire to the appropriate destination pin/terminal to create a wire connection. Draw all the necessary connections to form your inverter. If a wrong wire was placed, click ESC, highlight the wire and press delete.

To save your schematic, use **Design > Check and Save**. The results of the check will appear in the command window. Keep fixing the errors until you have a clean check in the icfb window log.

4. Simulating the Schematic Netlist

We want to simulate the schematic we have just designed to verify behaviour from both a functional and electrical point of view. In the Virtuoso GUI, select **Tools > Analog Environment > Simulation**, which will launch the window shown in Fig. 2.

To obtain the simulation window, select **Setup > Simulator/Directory/Host**. Select **hspiceS** as a reference simulator. Choose a project directory of your choice (you can use the default, or select a more appropriate build and simulation directory within your CDS working folder), and press **OK**. Next, select **Setup > Design** and under **Library Name** select your library name (inverter, in our case) and highlight inv under **Cell Name**.

Be patient, the initialization of this environment can take quite a while. From the Analog Environment window, you can then perform a simulation using **Simulation > Netlist > Create Final**. A window will pop-up, which generated a netlist for your schematic.

Note that this ASCII netlist file is located at the path described at the *top of the netlist window* - ***<simulationFolder>/<cell name>/hspiceS/schematic/netlist/hspiceFinal***

At this point, you will only be obtaining the circuit HSpice netlist, not stimuli or initialization parameters. It is possible to perform an entire spice simulation in Virtuoso, although we will not cover this in the current tutorial. However, we integrate this ASCII netlist in HSPICE with the techniques we learned in Phase 2's HSPICE tutorial.

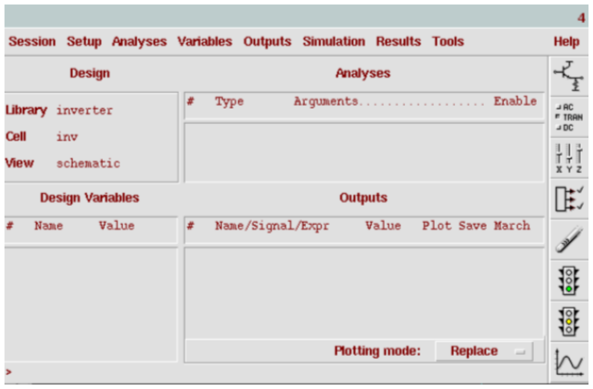

Session Setup Analyses Variables Outputs Simulation Results Tools Help
Design
Analyses
Library inverter
Cell inv
View schematic
# Type Arguments.................. Enable
Design Variables
# Name Value
Outputs
# Name/Signal/Expr Value Plot Save March
Plotting mode: Replace
AC TRAN DC

Fig. 2: Virtuoso Analog Environment Window

To simulate this netlist in HSpice, use the example found in:

/CMC/setups/ensc450/Layout/hspice/inv_schem.cir

Open a new terminal and *setup the hspice environment* as we did in tutorial 2. You may view the schematic waveforms and measure propagation delay and transition times with ezwave and the hpsice report generated. An example is provided in Fig. 3.

If you are satisfied with the behaviour of the designed schematic, feel free to move on to the layout. Note that while the circuit evaluation may be completed with HSpice at the ASCII level, we also defined this schematic for our circuit to support automated LVS. Although there is a way to define the circuit as ASCII and use it for LVS, we will only use the GUI schematic editor: this is the standard methodology for LVS, and we will use it for this course.

When completed with the Analog Design Environment, close the window using **Session > Quit**. It is a good idea to close this window as it takes up quite a bit of memory and may slow down the next design steps.

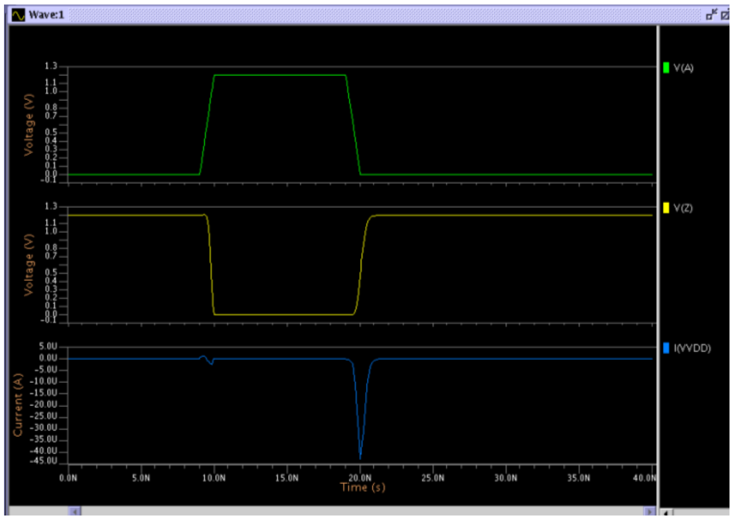


Fig. 3: Simulation of the HSpice generated netlist

Layout Design

5. Layout Design

Building a circuit from scratch is very complicated and time consuming. Even a small cell, such as an inverter, will take quite a bit of design time and interaction with the tool. Try not to perform everything in one session, and do not rush through the tutorial all in one step. Try to retain and understand the layout exercise as you will need it for your phase 3 stdcell designs.

Virtuoso Layout Editor

In the icfb window, create an empty layout view for your schematic: **File > New > Cell View**

- Library Name = <Your library name>
- Cell Name = <Your cell name>
- View Name = **layout**
- Tool = **Virtuoso**

Virtuoso is the main layout editor in Cadence. Again, this will take **QUITE SOME TIME TO FULLY LAUNCH**. The top and bottom status bars of the Virtuoso window provide useful information when editing and/or waiting for the tool to respond to your command. Specifically, the top window displays:

- X and Y coordinates of your cursor
- Number of selected objects
- Traveled distance in the X and Y direction
- The command in use

The bottom status bar displays the function of the mouse. This window looks very similar to the Schematic Editor, however the tools available for layout are very different.

Basic Navigation Rules for Schematic and Layout Editing

- For all Virtuoso tools, press Esc to exit from a specific functionality and return to the main window
- Zoom in by right-clicking the mouse. Zoom out with SHIFT-Z. Fit to screen using f
- To move an instance or layout feature, select the instance and press m, where you will be assisted with a pop-up window
- Copy an instance or layout feature, press c and use the same methodology as move
- To delete an instance in the layout, select it and press the delete button

After opening a layout, or creating a new layout, you will see a **Layer Selection Window (LSW)** appear showing all technology layer names and purpose layers. The LSW is a menu of physical layers that may be used in the layout. This LSW is a fundamental resource for the designer: when performing layout, we are rendering a 3D plot in 2D, so all physical and virtual layers of our layout will overlap in the Virtuoso GUI.

Using the LSW, you may select the drawing or entry layer (layer which you would like to draw objects on), specify the visibility layer (AV: All visible, NV: Non Visible), and set the selectable layers (AS: All selectable, NS: Non Selectable). LSW allows us to manipulate the layers which will greatly enhance and simplify our design and inspection.

The big challenge in designing a layout is to be compliant with all design rules, and as we are aware, there are quite a few of them. In order to make this procedure simpler, it is advisable to check every structure incrementally rather than designing the entire circuit and checking afterwards once complete. If too much of the design has been laid out, it may be impossible to fix all **DRC errors**.

In the following sections, you will be guided through a full design of an inverter, where Design Rules will be introduced and highlighted in red, where specifications may be found in the design manual. The TSMC 0.180um (180nm) design rule manual is located at:

/Lnx_STC/kits/cmosp18/doc/CMOSP18designRulesLogic.pdf

In order to set your layout design environment, in Virtuoso select the menu **Options > Display** and ensure that **Pin Names** are **on** under **Displays Control**, and the **grid control > X/Y snap Spacing** are **0.01** each (this means that the max resolution of your cursor will be 10nm, that is 2x the reference grid step of this technology and will be easier to measure/draw specific dimensions).

Refer back to the following points to help you during the layout (they may not make sense right now):

- In order to measure a dimension for a layer, select the starting point and press **k** (ruler) and stretch it to the target point. Click on the target point if you want the ruler to stick. You can remove all rulers by pressing **SHIFT-k**.
- To build a **rectangle** shape on the layout, select the appropriate **layer** on the **LSW**, place the cursor in the desired starting point, and press **r** (rectangle). You do not need to specify a net name. Keep an eye on the dimensions of the rectangle you are drawing described by **dX** and **dY** at the top of the layout drawing window. To draw a **polygon** shape (non-rectangle shape), press **P** and follow the same directions as specified in this point.

- If you would like to stretch an existing rectangle (this typically happens when you are trying to match design rules), you must **select one side** of the polygon/rectangle (**not** the entire shape). Activate this feature by pressing **F4** until you see the command window display **getTogglePartialSelect** at the bottom of the editor.
    - **getTogglePartialSelect nil** means that the feature is deactivated. The side will highlight with dashed lines.
    - Press **s** and pull the side of the rectangle to the desired place/position

- If you would like to know the layer of a given polygon, select the polygon with a left click, and press **q** (query). With the same command, **you may also change the layer**, which is useful if copying and pasting a nmos into a pmos (or vice versa, n-well to w-well etc)
- If you would like to select a layer that is underneath another polygon, use the LSW features for visibility. LSW allows the user to visualize, or select, one of the layers.

The gate

[**PO.W.1**] Draw a ruler **0.18** wide (**k**). Note that you may have to zoom first to obtain the appropriate scale. Select **polysilicon (drw)** from the **LSW** pallet and draw a rectangle by pressing **r**, and make it as wide as the ruler. The height of the rectangle is not important at this point, and may be adjusted later as indicated above. Fig. 4 provides an example. The width of the gate represents the length of the transistor gate (channel length). Note that in some libraries, to decrease leakage, it is possible to have larger gate, L. In our lab activity, we will stick to unity length transistors.

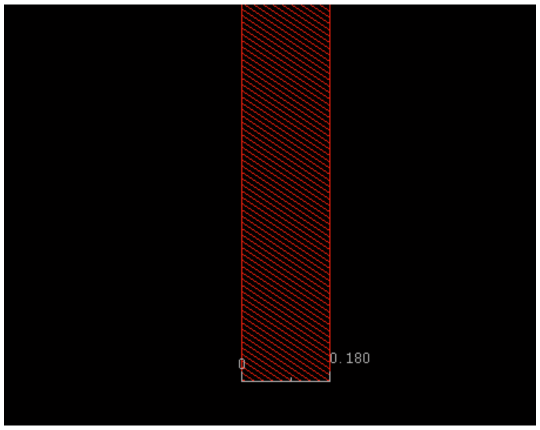
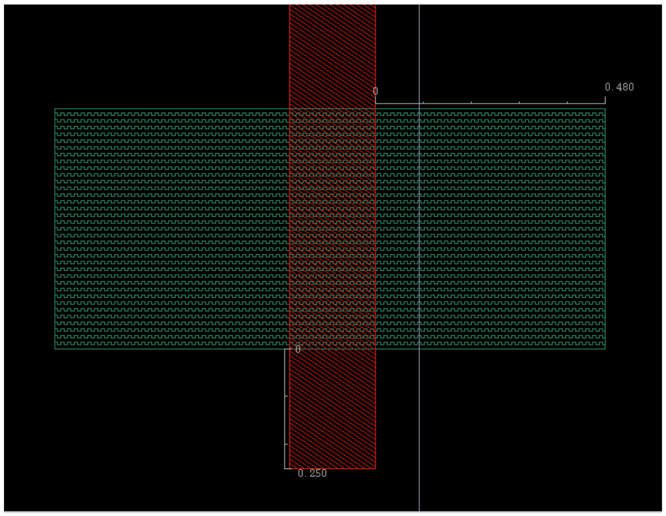


Fig. 4: Drawing the gate

The Drain and the Source

1) Active & Diffusion

[**P0.0.1**] Prior to drawing the doping region, the size of the transistor's active region must be established. The **active region will establish the width of our transistor**. In our case, draw a ruler that is **0.5** high. Draw another ruler which extends **0.48** from the polysilicon on both sides of the active. The gate must extend at least **0.25** from the active. Select **active (drw)** on the **LSW** pallet, and draw a rectangle (r) of active region perpendicular to the gate polysilicon as shown in Fig. 5. The dimensions of the transistors have now been defined – the **height of the active region represents your transistor width**.



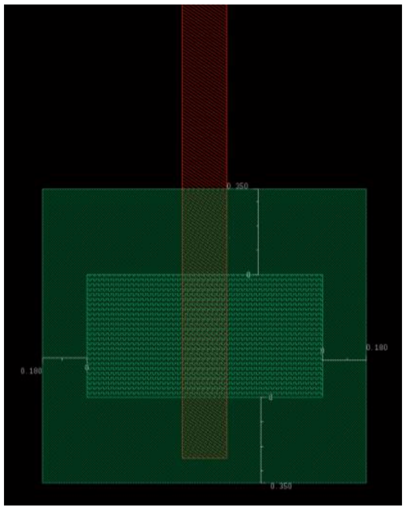


Fig. 5: Drawing the Active Region

Fig. 6: Drawing n+ region

[**NP.E.1 NP.C.5**] Once the active is defined it must be immersed with nplus diffusion - the n+ region around the active must be drawn. Following the design rules, n+ must overlap the active by >=**0.35** in the same direction as the channel (vertical), and by **0.18** perpendicular (horizontal) to the channel. Select **nplus (drw)** from the LSW and draw the rectangle, with the dimensions suggested in Fig. 6.

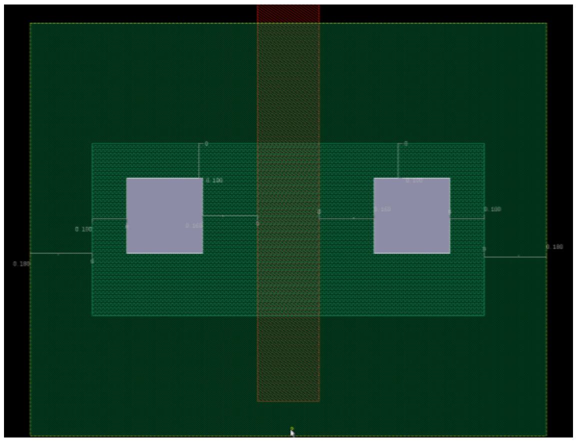


Fig. 7: Diffusion Contacts

2) Adding Contacts

[**CO.W.1, CO.E.1, CO.C.1**] Next, contacts must be added to the diffusion region of the transistor to define the source and drain. Although the source and drain are not defined now, the source will become clear once the substrate is determined and connected to the supply rail.

Select **contact (drw)** from the LSW window and draw a square with dimension **0.22x0.22** and the assistance of the ruler (k). The active-contact to gate spacing must be **> 0.16,** and active overlap of a contact **> 0.10**.

Note: You may need to extend and stretch the active and n+ layer horizontally to ensure that the layout rules are respected with the active/n+ and contacts. Specifically, the poly (gate) must be at least >= 0.16 from the contacts, and contact must be >= 0.1 from the edge of the active (top, sides etc). Do not adjust the height of the active as this defines transistor width. An example is provided in Fig. 7.

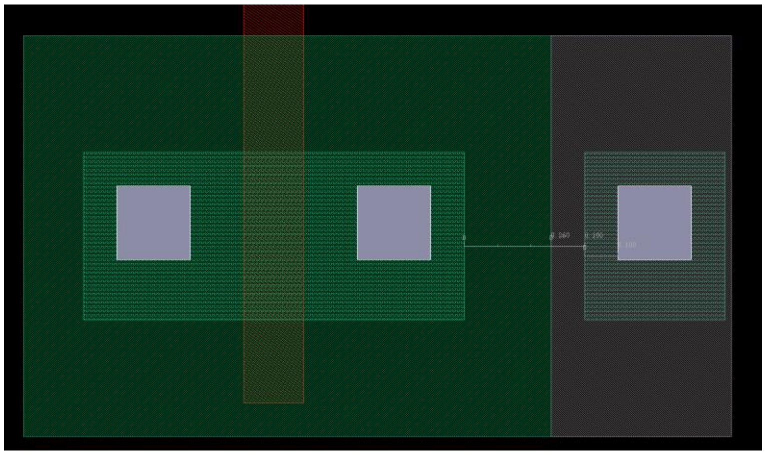


Fig. 8: Adding the Substrate

The substrate

[**PP.C.1, NP.C.2, PP.E.3**] Recall that mosfets are 4 terminal devices. To prevent latchup in 180nm, mos transistors require electrical connection to their backgates (n-well for pmos, p-sub for nmos). The pmos substrate must connect to a voltage greater or equal to its source, and the nmos to ak voltage less than or equal to its source.

The backgate of a nmos is made of **p+** and drawn adjacent to the source (for layout optimality). Fig. 8 provides a reference for adding the substrate, where the active and contact in the substrate follows the same design rules as described previously (i.e. contact from active edge >= 0.1). From rule **PP.C.1**, the active/n+ region must extend **>= 0.26** from the pplus (p+) region and hence you may need to stretch the right edge of the n+ layer to be >= **0.26** since the 0.18 measurement is not longer valid due to the newly added (adjacent) substrate.

Use a ruler to first measure the appropriate layout dimensions for p+/active region of the substrate:

- The p+ must extend **0.1** from the n+ region before placing active (**NP.C.2**). Draw and add the substrate's active layer.
- **0.1** of the active must be present prior to laying a contact (**0.22x0.22**)
- Another **0.1** of active must be present prior on the other end of the contact to active region
- At least **0.060** of p+ must extend past the active region (**PP.E.3**) (no other layer abutted)

Once you have measured the dimensions and drawn the pplus/active layer, it will be easiest to copy (c) a contact you made for the source/drain and drag that contact **0.1** from the edge of the active as measured (sides and top of active). Ensure your dimensions are correct.

Note: the size of the active in the substrate has no effect on transistor width. Therefore, it may be drawn with smaller dimensions than shown – minimized to fit tightly around the contact (i.e 0.1 all around).

It's a good idea to verify your design now. Let us perform a DRC check in Virtuoso by selecting **Verify > DRC**. When the DRC window opens, enable **Set Switches**, and select **do_drc** and **no_antenna_check** from the list (to select both entries, hold CTRL down) and press **OK**, and **OK** again to run the DRC Check.

Correct any errors listed in the icfb window. The only acceptable listed errors should be **floating contact, substrate/well soft connection, and floating poly1**. Correct all other errors. Once corrected, remove the markers with **Verify > Markers > Delete All**. See the DRC chapter of this tutorial for helpful features of this checking tool.

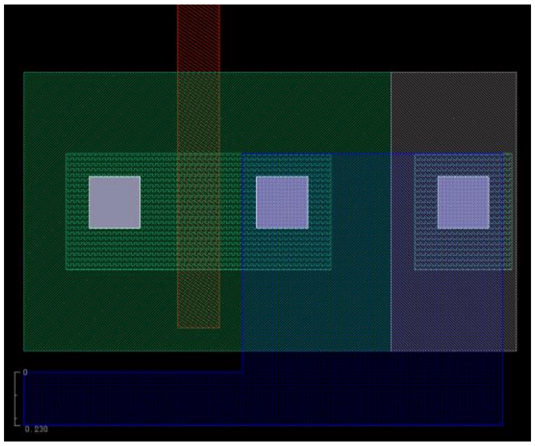


Fig. 9(a): Adding a power rail

The Power Rail

For the source and substrate, draw strips of **metal1 (drw)** as shown in Fig. 9. Acknowledge that by creating the substrate next to another transistor terminal, we have selected the transistor's source as the metal will be shared with the substrate and connected to the ground rail.

Metal1 width must be **>= 0.23** when drawing the supply rail, and extend **0.06** in each direction over the contact. We will use the polygon (press **P**) feature to draw the ground rail shape of Fig. 9. It is advisable to first draw a ruler to keep an eye on distances and avoid potential violations, then use the polygon feature to draw the rail shown in Fig. 9. Ensure that the ground rail overlaps with the source and body using proper dimensions (else the stretch feature will need to be used to correct the layer). Run a DRC check to ensure your metal1 has been laid out correctly without violating metal1/contact rules (ignore any floating contact and poly errors mentioned previously).

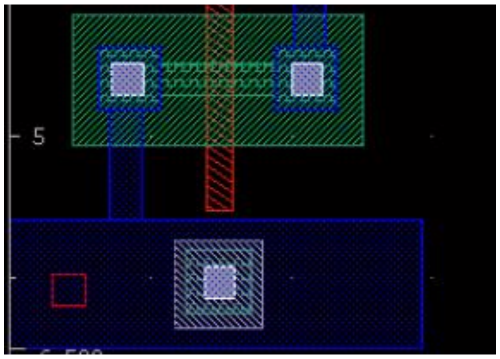


Fig. 9(b): Alternative substrate

An area optimized alternative for creating a substrate/body is shown in the bottom of Fig. 9(b). This substrate diffusion abides by layout rules to surround the contact, and may also be placed in the rail as opposed to the side of the diffusion as we have done in this tutorial (however the ground rail now possesses larger dimensions). Keep an open mind when performing the layout activity as there are many alternatives for implementing the same gate with different area overhead. A good exercise is to complete the inverter again, however considering this alternative substrate design.

Poly Line and Poly Box

[M1.A.1] The gate is composed of polysilicon. If we copy a contact to make our inverter's input A, it can not be placed directly on the gate as it is currently drawn: the contact is currently much bigger than the gate area. The gate must therefore be extended so that the contact fits. Extending the gate in the middle does not affect the transistor's channel length since length is defined by the area which overlaps with the active (**aka do not touch this region**).

An easy way to add a gate contact is by going to **Create > Contact** and selecting **M1_Poly** from the drop-down list. This will create a Parameterized Cell (pcell) which contains the minimum sized regions for contacting poly1 and metal1. Place the object above the nfet, adjacent to the poly as shown in Fig. 10. You may press *shift-f* to view the contents of the P-Cell you just placed. The metal1 contact drawn however will violate the minimum area rule for metal1. To eliminate this violation, draw a **metal1** strip over the existing metal1 square (remembering the metal1 over contact rule >**0.06**).

If you decide to create the contact with individual poly layers versus the Create > Contact shortcut mentioned above for contact and metal1 overlay, you may use the following layout rules:

Poly Box side >= 0.420um   Poly Width >= 0.180um   Poly Spacing >= 0.25um (gate to gate distance)
Min Metal1 Area >= 0.2025um^2 (sqrt(0.2025) = 0.420um)

Adding Pins to Layout

Next, we will add pins to our nfet layout. To do this, go to **Create > Pin**. Let us first create a pin for gnd. Enter **GND!** for the pin name (to match our inverter schematic), and specify the IO type as **InputOutput**, and **metal1_T** for pin type. If you would like to display the name of the pin, ensure **Display Pin Name** is selected before placing the metal pin on the layout. Proceed to place the pin on top of the metal ground rail as shown in Fig. 10. Next, create a rectangle over this pin using the metal1(pin) layer.

Next, we will create a pin for our gate. **Create > Pin**, enter **A** for pin name, IO type as **input**, and **metal1_T** as pin type. Place the pin on top of the poly box's **metal strip** which we laid out previously, and a metal1(pin) rectangle over the pin. Your layout should now resemble Fig. 10. Run another DRC to ensure the pin was placed correctly, ignoring the previous errors mention (however the floating poly error should now disappear).

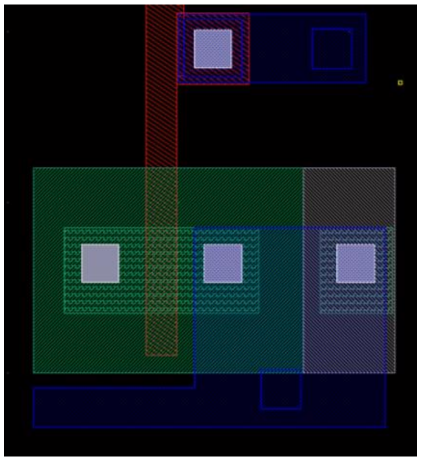
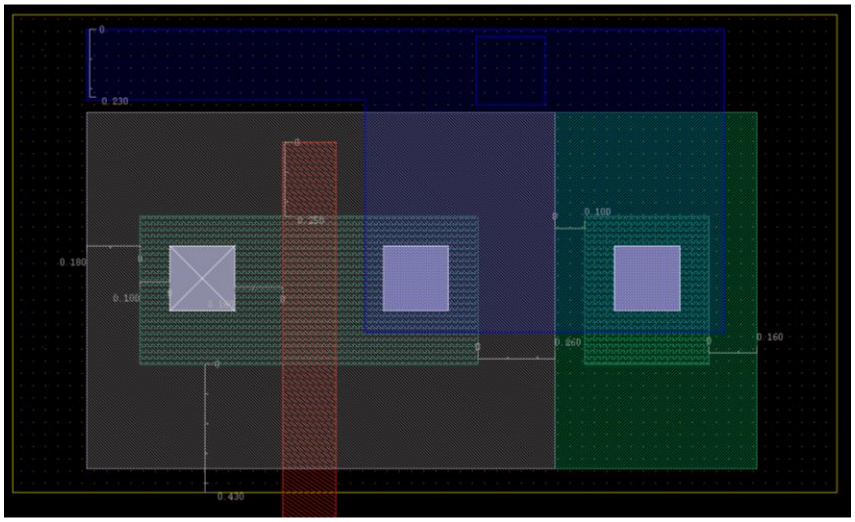


Fig. 10: Pins for gnd rail and inverter gate

PFet Design

Now that we have a DRC free nmos transistor, we can build a similar pmos transistor using the same rules as presented in the tutorial for nmos. Since we spent quite a bit of time and effort building DRC-free geometries for the nmos, we should reuse the shapes we created as much as we can for the pmos by copying and pasting, and changing the diffusion features. Although the pmos will be 100% equivalent to the nmos, this will obviously lead to asymmetrical inverter behaviour between the pull-up and pull-down network i.e. a skewed gate. Recall that we also used the same transistor widths for our schematic based inverter, so we should yield the same circuit once we finish our layout.



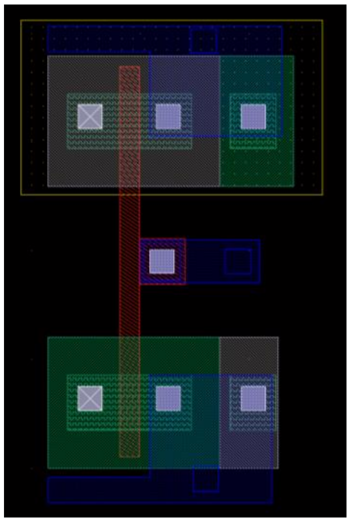


Fig. 11: Pfet layout Fig. 12:pfet and nfet

Before copying the nfet for the pfet design, you will need to place an nwell (drw) layer as the pmos is built on the nwell, where the black background of Virtuoso represents the p-substrate.

We can copy any stack of polygons by selecting them with the mouse, and using the ‘c’ (copy) command. You can change the layer of a polygon (as you will need to convert layers to other types for the pfet) by selecting the layer and using the command ‘q’ (query). You may resize using the ruler and stretch **s** feature to abide by layout rules. An example of the pfet and it’s layout layers and dimensions are displayed in Fig. 11, with a zoomed out version of the semi-final inverter shown in Fig. 12. Remember to place the **VDD!** pin (**metal_1**, type **InputOutput**) on the power rail. The following rules will aid in your design:

Nplus extension over p-active >= 0.180um, nplus distance to nwell boundary < 0.430um

Nwell overlap of Pplus >= 0.430um

As a final step, we will need to connect the drain of the nfet to the drain of the pfet with metal1 (shown in Fig. 13) to provide the gate output. Ensure metal >= 0.06 around contacts for output. Place the output pin on the metal interconnect – **Z**, **metal_1**, type **output**.

Feel free to implement the substrate for the pfet with the alternate design, as demonstrated in Fig. 9(b).

Our final inverter design is given in Fig. 13.

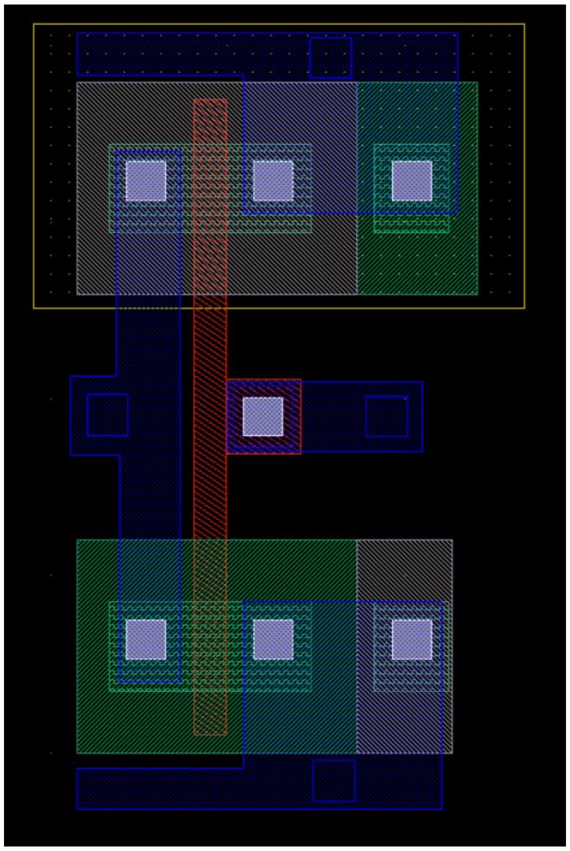
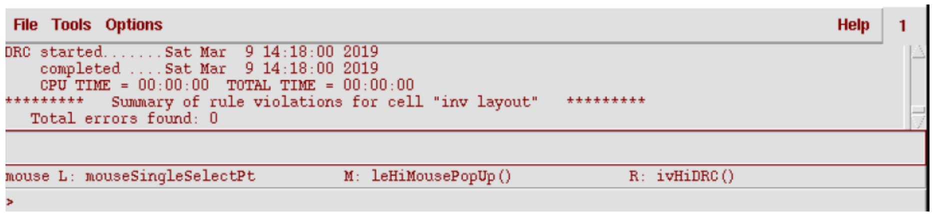


Fig. 13: Final Inverter Layout Design

DRC (Design Rule Check)



```
DRC started.......Sat Mar  9 14:18:00 2019
    completed ....Sat Mar  9 14:18:00 2019
    CPU TIME = 00:00:00  TOTAL TIME = 00:00:00
*********  Summary of rule violations for cell "inv layout"   *********
   Total errors found: 0
```

```
mouse L: mouseSingleSelectPt          M: leHiMousePopUp()          R: ivHiDRC()
>
```

Subsequent to designing a custom circuit layout (even during), we need to make sure that our implementation does not violate design rules. This is done by the utility **Design Rule Check (DRC)** as we have completed many times during the layout process. We **must** run DRC on any layout to ensure that the layout complies with the rules of the technology. If DRC is violated, then:

1. The mask may not be able to provide the expected design, leading to unwanted shorts and open circuits
2. The Spice models could be inaccurate, thus making our simulation unreliable. If the SPICE simulations are unreliable, then the STA in our synthesis tools will also be unreliable, and our IC will likely not function properly due to setup and hold violations which we can not view in the simple netlist of our design. Violations are induced by DRC errors, and are caused due to misalignment of real silicon shapes and the transistor behaviour modeled in Spice.

To perform DRC, in the layout open in the Cadence Virtuoso GUI, select **Verify > DRC**. Ensure:

- Checking Method: **flat**
- Checking limit: **full**
- Switch Names > Set Switches: **no_antenna_check**
- Join Nets with same name: **on**
- Echo commands: **Off**

Your layout window may show flashing violation markers describing the area of any DRC violations. The command window (icfb) will provide a textual report of the DRC violations, and will be helpful with the codes it provides pertaining to layers and measurement violations (check the manual referenced at the beginning of the tutorial to look-up codes). The following are also useful DRC features:

- **Verify > Markers > Explain** (or Find, Delete etc). Explain will allow you to obtain more info on the error/flashing marker you select on the GUI with a pop-up window.
- **Verify > Markers > Find**, and by selecting Next, you may cycle through all the markers.
- **Verify > Markers > Delete All** – gets rid of the flashing markers. They will be saved if you save your design using **Design > Save**.

It is recommended that you run DRC check often during layout design to make your design implementation as easy as possible.

Layout Versus Schematic (LVS)

The layout of the standard cell is fundamental information for digital design:

1. It specifies the shape of the standard cells as well as all pin positions, that will be used by automated P&R tools to perform connections during place&route.
2. From the layout, we can extract a new spice circuit for the cell containing real physical parameters of the layout. Since the layout has been drawn:
    a. We must check the functional equivalence between the starting circuit and the extracted circuit to ensure that we have actually designed the same circuit and accomplished our task.
    b. From the extracted circuit, we may have a precise estimation of circuit parameters from detailed simulations.

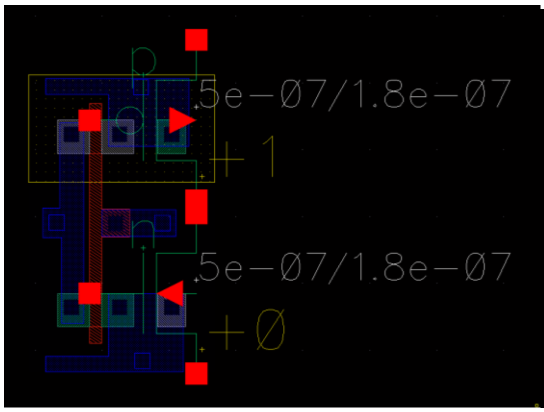
The step of extracting a circuit schematic from a layout and comparing it to the reference schematic is referred to as LVS. LVS is an essential and mandatory step in IC design.

Extracting the Schematic from Layout

From the Layout window, we may extract a layout schematic for LVS with **Verify > Extract**. In the pop-up, select:

- Extract Method: **flat**
- Join Nets with Same Name: **On**
- Echo Commands: **Off**

Shortly, you **should** see "total errors found: 0" in the icfb window, and a new view in your library called **extracted** which represents the structures extracted from the layout. To view the extracted circuit, in the icfb window, select **File > Open** and under View Name **extracted**. Press OK. Visually check the extracted design and verify that it makes sense to you. The extracted layout should resemble something similar to Fig. 14. Press **shift-f** and **ctrl-f** to change the extracted view to Fig. 15. You may also select **Options > Display** and select Nets to verify the connectivity of your layout.



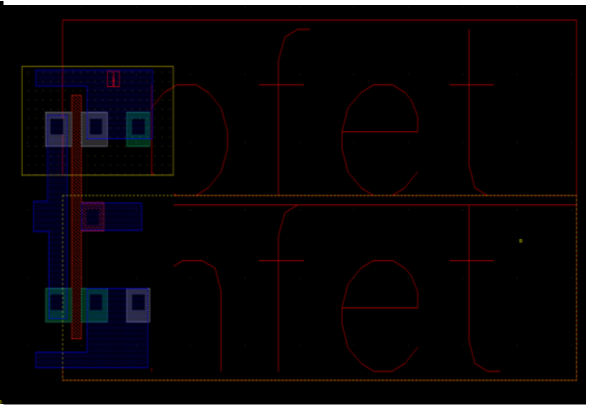


Fig. 14: Extracted View of Inverter

Fig. 15: Alternate Extracted View of Inverter

Note: It is possible to run Schematic extraction and LVS even if the DRC rules are not met. Unless you really understand what you're doing, it is not advisable as extracting such a layout is meaningless and manufacturable since it does not respect design rules. Most foundries will also perform a DRC on the design that you send in for fabrication, and will refuse to build silicon if the layout violates DRC. On the contrary, foundries do not perform LVS before fabrication as they don't have this schematic visibility for your design (and do not want this visibility either).

Layout Versus Schematic

Open the **extracted** view of your cell (inverter in this case) and select **Verify > LVS**. The artist LVS pop-up window will appear. Select the following parameters:

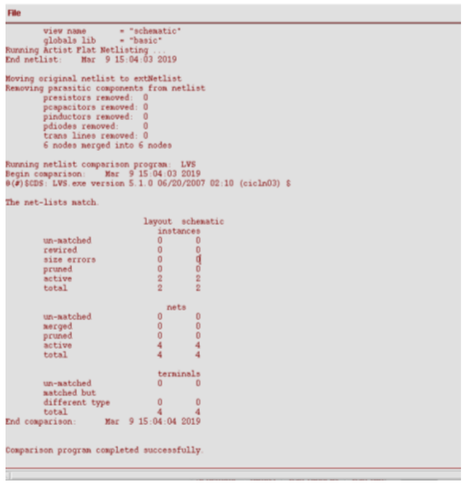
- Create Netlist: schematic and extracted (both selected)
- Library: <library name> for both   (should be inverter for this tutorial)
- Cell: <Cellname> for both (should be inv for this tutorial)
- View: schematic (on left)   extracted (on right – watch for typos)
- LVS Options: none should be selected, in particular disable Rewiring and Terminals if active.

Click Run in the window. The icfb window log will display "LVS job has started". After a moment, you will obtain a pop-up which says "**… has succeeded**". This does not mean that LVS matched though (you wish!), this implies only that the tool has finished analysis.

In order to see details of the run (esp in the case of a mismatch), select **info** in the pop-up. If passed, in the LVS pop-up:

1. If you select Log File, you will obtain a full log of LVS activity. The statement which you are looking for is: ***The net-lists match*** (you may need to scroll down a bit). After you obtain a summary of schematic details, if you do not have matching net-lists, something went wrong. Browse the errors – most of the time the error will be a missing pin in the layout, wires not connected properly in the schematic, or an overlooked naming of pins, directions etc in the schematic or layout. Fix the errors and re-run the steps for extraction to re-run this step.

2. If you select **Netlist**, you may generate an ASCII spice netlist (**not hspice compatible though**) of both the schematic and extracted view. If you are at a loss at understanding the reason for your LVS mismatch, you may manually compare these netlists and decipher the differences as seen by Viruoso.



```
        view name       = "schematic"
        globals lib     = "basic"
Running Artist Flat Netlisting ...
End netlist:    Mar  9 15:04:03 2019

Moving original netlist to extNetlist
Removing parasitic components from netlist
        presistors removed:  0
        pcapacitors removed: 0
        pinductors removed:  0
        pdiodes removed:     0
        trans lines removed: 0
        6 nodes merged into 6 nodes

Running netlist comparison program:  LVS
Begin comparison:    Mar  9 15:04:03 2019
@(#)$CDS: LVS.exe version 5.1.0 06/20/2007 02:10 (cicln03) $

The net-lists match.

                            layout  schematic
                               instances
        un-matched             0         0
        rewired                0         0
        size errors            0         0
        pruned                 0         0
        active                 2         2
        total                  2         2

                                 nets
        un-matched             0         0
        merged                 0         0
        pruned                 0         0
        active                 4         4
        total                  4         4

                               terminals
        un-matched             0         0
        matched but
        different type         0         0
        total                  4         4
End comparison:      Mar  9 15:04:04 2019


Comparison program completed successfully.
```

Fig. 16: Sample Netlist generated by LVS

Post Layout Simulation

Once completed a DRC and LVS free layout, we have a valid cell design. Once we have an extracted schematic, we may simulate it and compare the results against the schematic version. It is also good to verify the physical parameters of the layout, especially for more complex circuits in comparison to the inverter, to see any changes in electrical behaviour.

Open the **extracted** inverter schematic. We will now go through the same procedure as before, extracting a netlist from the layout this time (in extracted view).

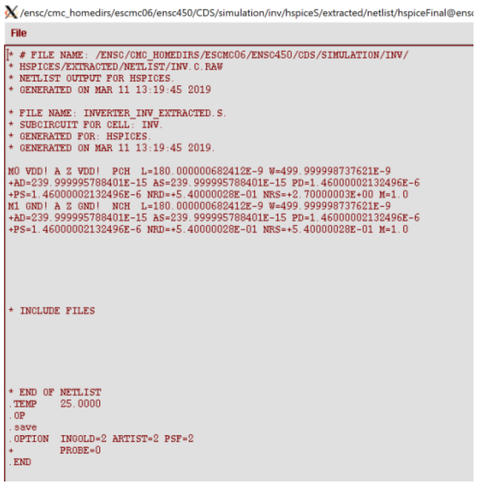
Select **Tools > Analog Environment.** In the pop-up, select **Setup > Simulator/Directory/Host**. Select hspiceS as a reference simulator. Choose a project directory of your choice (you can use the default, or select a more appropriate build and simulation directory in your CDS work directory). From the Analog Environment window, generate the netlist using **Simulation > Netlist > Create Final**.

Open up the extracted netlist in another terminal, which is reserved for hspice. The netlist is located in -
*<simulationFolder>/<cell name>/hspiceS/**extracted**/netlist/hspiceFinal*

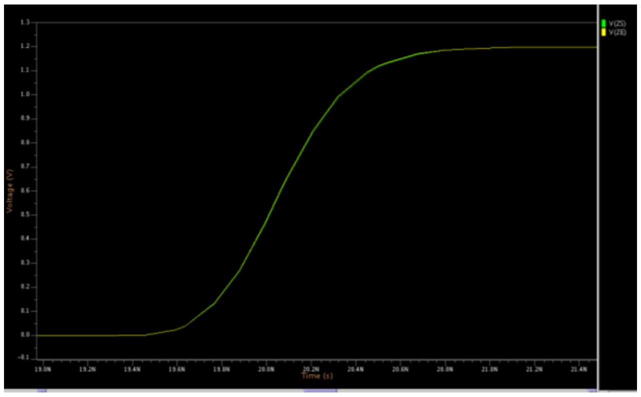
Copy and paste the circuit netlist into your spice deck. Use the file template provided to you as a reference:

/CMC/setups/ensc450/Layout/inv_layout.cir

Use both your schematic and layout/extracted netlists and integrate them into the template file. Depending on the names which you gave your cells/terminals, you may have to adapt the file to your design needs. Upon simulation, we can see that although the schematic and extraction yield similar results, even for such a small circuit as the inverter there are very small differences that can be appreciated: use the .meas statements to quantify the differences induced by layout vs schematic.


/ensc/cmc_homedirs/escmc06/ensc450/CDS/simulation/inv/hspiceS/extracted/netlist/hspiceFinal@ensc
File

```
* # FILE NAME: /ENSC/CMC_HOMEDIRS/ESCMC06/ENSC450/CDS/SIMULATION/INV/
* HSPICES/EXTRACTED/NETLIST/INV.C.RAW
* NETLIST OUTPUT FOR HSPICES.
* GENERATED ON MAR 11 13:19:45 2019

* FILE NAME: INVERTER_INV_EXTRACTED.S.
* SUBCIRCUIT FOR CELL: INV.
* GENERATED FOR: HSPICES.
* GENERATED ON MAR 11 13:19:45 2019.

M0 VDD! A Z VDD!  PCH  L=180.000000682412E-9 W=499.999998737621E-9
+AD=239.999995788401E-15 AS=239.999995788401E-15 PD=1.46000002132496E-6
+PS=1.46000002132496E-6 NRD=+5.40000028E-01 NRS=+2.70000003E+00 M=1.0
M1 GND! A Z GND!  NCH  L=180.000000682412E-9 W=499.999998737621E-9
+AD=239.999995788401E-15 AS=239.999995788401E-15 PD=1.46000002132496E-6
+PS=1.46000002132496E-6 NRD=+5.40000028E-01 NRS=+5.40000028E-01 M=1.0




* INCLUDE FILES




* END OF NETLIST
.TEMP    25.0000
.OP
.save
.OPTION  INGOLD=2 ARTIST=2 PSF=2
+        PROBE=0
.END
```


Voltage (V)
Time (s)