OSTBAYERISCHE TECHNISCHE HOCHSCHULE REGENSBURG

ELEKTRO- UND INFORMATIONSTECHNIK

Circuit Integration Internship (PSI)

Experiment No.: **SI1**
Experiment title: **Full-custom design of CMOS cells**
Experimental concept: **C. Schimpfle / D. Kohlert**

I. Experimental objective

In experiments SI1 to SI3, you will learn how to use a very powerful EDA (Electronic Design Automation) software package used in industry. You will demonstrate the use of the programs using complete design flows for fully custom and standard cell designs.

II. Brief description of the experiment

In the first experiment, you will familiarize yourself with the layout program (CADENCE Virtuoso). You will be asked to design a CMOS inverter with minimal area requirements while observing all geometric design rules.

During the course of the experiment (Chapter V.3), different topics are addressed:

- Using Virtuoso
  - Introduction to the layout program VIRTUOSO XL
- Layout of n- and p-channel transistor with SKILL
  - Here, the basic transistor rectangles are dimensioned and placed. This requires a thorough understanding of the design rules. The placement of the rectangles is done via scripts.
- Contact holes
  - The contact holes are specified as finished cells due to time constraints, the required capabilities would correspond to V.3.2
- Design of the busbars for VDD and VSS
  - This section covers the construction of the supply rails with the substrate contacts. These are created based on the predefined contact holes in the classic "hand" layout. This also introduces these software functions.
- Structure of the inverter
  - Here the placement and wiring of several basic cells to form a complete standard cell is carried out.
- Structure of the ring oscillator
  - Here the placement and wiring of several inverters to a 5-stage ring oscillator is carried out

III. Experimental preparation

1. Familiarize yourself with the process used! Complete all listed process steps!

2. Study the design rules. Enter the information for version 3.2 in the protocol file (psi1_prot.ods) and answer the questions for versions 3.3 and 3.4.

3. Find out about (static) CMOS logic (e.g. in /1/) !

IV. Design styles

When designing an integrated circuit, several requirements must be met: The correct function of the circuit must be ensured throughout all design steps, the constraints imposed by the technology must be considered, and the technological capabilities must be utilized as effectively as possible. At the same time, the time and personnel required for the design should be as minimal as possible. All of these requirements are difficult to meet simultaneously. Therefore, methods have been developed that offer the best possible compromise solutions depending on the specific design task.

A distinction is made between three different design styles:
- dem**full-custom design**,
- dem**cell-oriented design**and
- dem**Array design**.
-

The**fully customized design**is the design style with the most degrees of freedom. Each transistor can be dimensioned and placed individually. The traces connecting the transistors can be optimally designed with respect to electrical constraints (e.g., signal propagation delays, crosstalk, etc.). The characteristic features of this design style are:

- high optimization potential,
- optimal use of technology,
- high risk of errors,
- high creation and correction effort,
- low degree of automation,
- long “time to market”.

(The "time to market" is the time between submitting the customer specifications and delivering the first functional circuit prototype.) Overall, the characteristics listed show that this design style only makes sense if the circuit created with it can be sold in large quantities over a longer period of time. The high development effort is also worthwhile for the creation of cell libraries and the design of memory cells, i.e., structures that are frequently used in other circuits.

At the**cell-oriented design**uses pre-designed basic layouts (library cells). These cells are verified for correctness of their design and their electrical behavior is known. While in the so-called **Macrocell design**There are usually no restrictions regarding the geometry of the cell layout, the cell layout in the so-called**Standard cell design**subject to considerable limitations. Usually, rectangular cells of identical height are used, which have supply lines at the top and bottom edges. By simply lining them up (**Abutment**) of these standard cells, more complex functional blocks can be realized. The voltage supply to the cell rows is comb-like from the left and right sides. Figure 1 shows a simple example of a standard cell design.

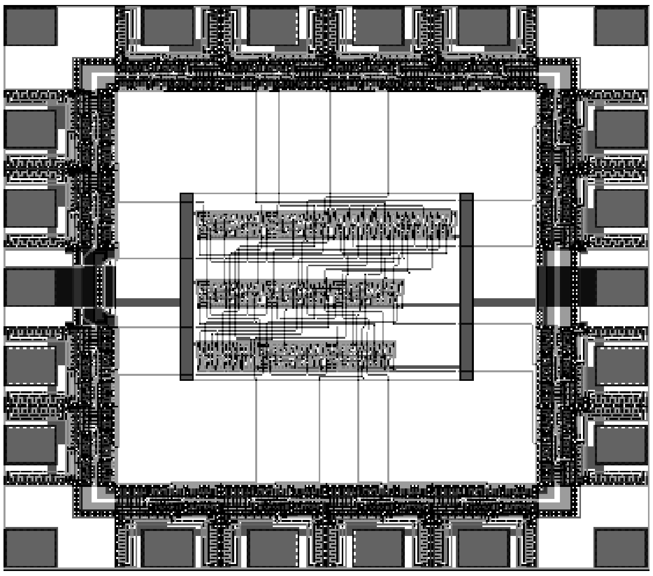


Fig. 1: Simple standard cell design (L-Edit demo example)

Between the individual cell rows there are wiring channels in which the wiring between the cells is carried out (**Channel Routing**).

The characteristic properties of the standard cell design are:
- simple cell arrangement,
- simple wiring,
- high degree of automation,
- few optimization options,
- poor utilization of chip area,
- separate treatment of the chip edge necessary,
- shorter “time to market” than with fully customized design.

The standard cell design is widely used today for the design of digital circuits.

To further shorten the time to market, design styles were sought that allow for the prefabrication of silicon wafers. This approach is used in the so-called**Gate array design**In this design style, wafers are prefabricated to the extent that all transistors are already present. The chip is "personalized" by wiring the transistors to customer specifications. The main advantage of this design style over the previously mentioned is the shortened production time due to prefabrication. Of the three design styles mentioned, this one generally has the shortest time to market.

V. Experimental procedure

V.1 Logging in on the PC

V.2 Calling Virtuoso

V.3 CMOS inverter layout:

Task
Your task is to create the layout of a static CMOS inverter. This layout should satisfy all geometric design rules according to the SI lecture while occupying as little space as possible.

V.3.1 Using Virtuoso
The operation of Virtuoso is described in the Virtuoso Quick Start Guide (VirtuosoTut.pdf).

V.3.2 Layout of n- and p-channel transistors with SKILL
The process used (Skywater 130nm) has design rules that make manual layout extremely laborious. It is therefore more sensible to consider the size and position of the required rectangles based on a sketch and the relevant design rules, and then create them automatically using a SKILL script. This is especially true for the two transistors. Once the transistors are drawn, the situation is easier, as you can orient yourself using the positions of the transistors.

SKILL is Cadence's own scripting language with all the features of a programming language and the ability to create drawing objects. In our case, only the commands for creating and placing rectangles and some simple calculations are required.

Example: SKILL script to create a rectangle in the "diff" "drawing" layer

The first two lines always remain the same:

```
cur_view = geGetEditCellView( hiGetCurrentWindow( ) )
println( cur_view )
```

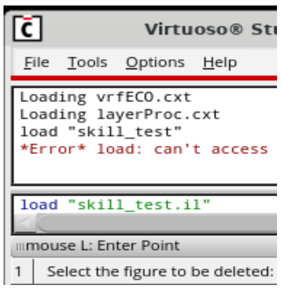
For drawing, the lower left and upper right corners as well as the layer. The drawing command is:

```
dbCreateRect(cur_view '("diff" "drawing") list(xmin : ymin xmax :
ymax)
```

The following script draws a rectangle in the drawing layer "Diffusion" with the corner coordinates 0µm ; 0µm (bottom left) and 50nm ; 100nm (top right):

```
cur_view = geGetEditCellView( hiGetCurrentWindow( ) )
println( cur_view          )
; rectangle     diffusion
xmin  =  0
ymin  =  0
xmax =  0.1
ymax =  0.05
dbCreateRect(cur_view '("diff" "drawing") list(xmin : ymin xmax : ymax) )
```

The SKILL script must in the Virtuoso working directory.
The adjacent figure shows the call:

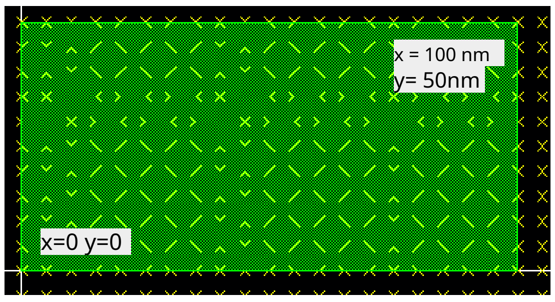


```
Loading vrfECO.cxt
Loading layerProc.cxt
load "skill_test"
*Error* load: can't access
```

```
load "skill_test.il"
```

mouse L: Enter Point
1 Select the figure to be deleted:

The following rectangle is created (diff = green):



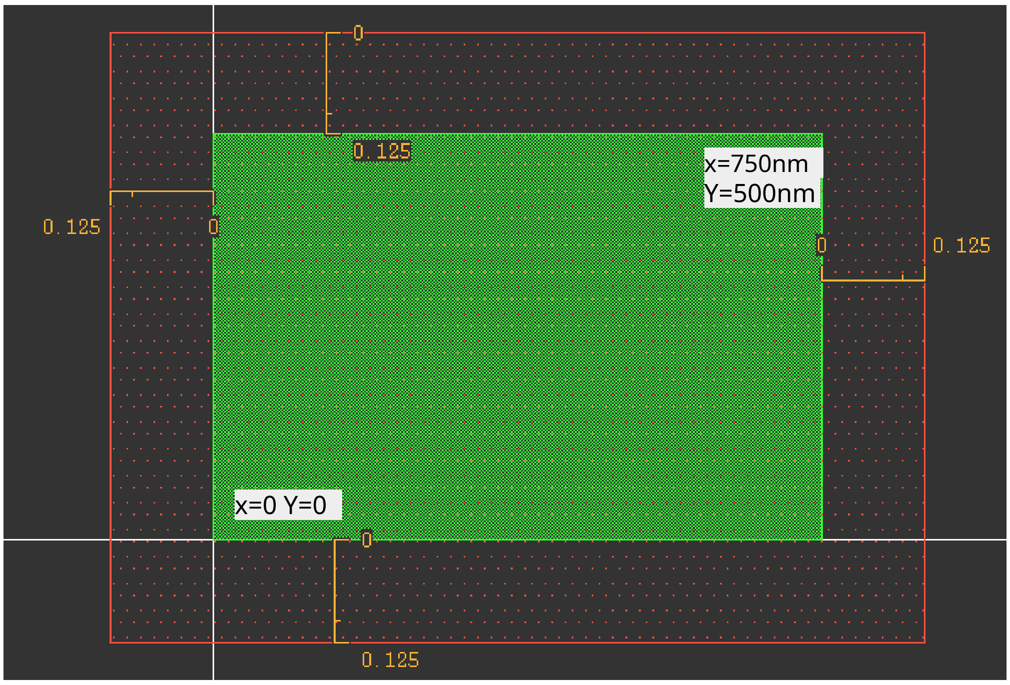
The ability to calculate coordinates can be used profitably for many purposes.

For example, there is a design rule (n_psd.5a) that requires the nsdm layer to overlap the diff layer by 125 nm on each side. The following script shows the creation of the rectangle in the nsdm layer based on the rectangle's coordinates in diff :

```
cur_view = geGetEditCellView( hiGetCurrentWindow( ) )
println( cur_view )

n_psd_5a = 0.125

;Rectangle diff
xmin = 0
ymin = 0
xmax = 0.75
ymax = 0.5
dbCreateRect( cur_view '("diff" "drawing") list(xmin : ymin                    xmax : ymax) )

;n_psd
xmin = -n_psd_5a
ymin = -n_psd_5a
xmax = 0.75 + n_psd_5a
ymax = 0.5 + n_psd_5a
dbCreateRect( cur_view '("nsdm" "drawing") list(xmin : ymin                    xmax : ymax) )
```

The name of the design rule for the overlap can be used as a variable name, resulting in flexible scripts that can even be easily adapted to changed design rules.

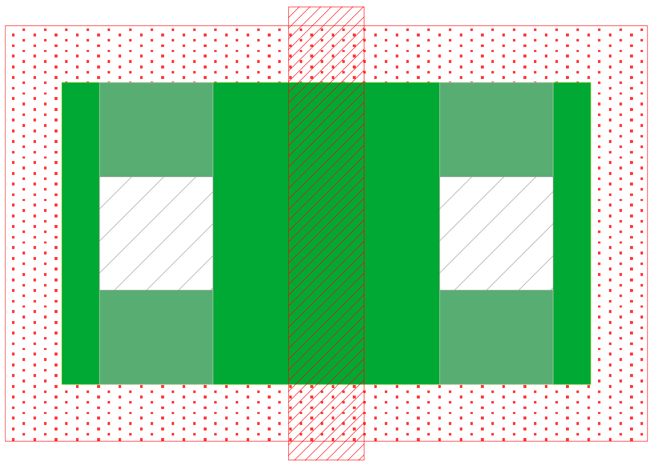
Result:



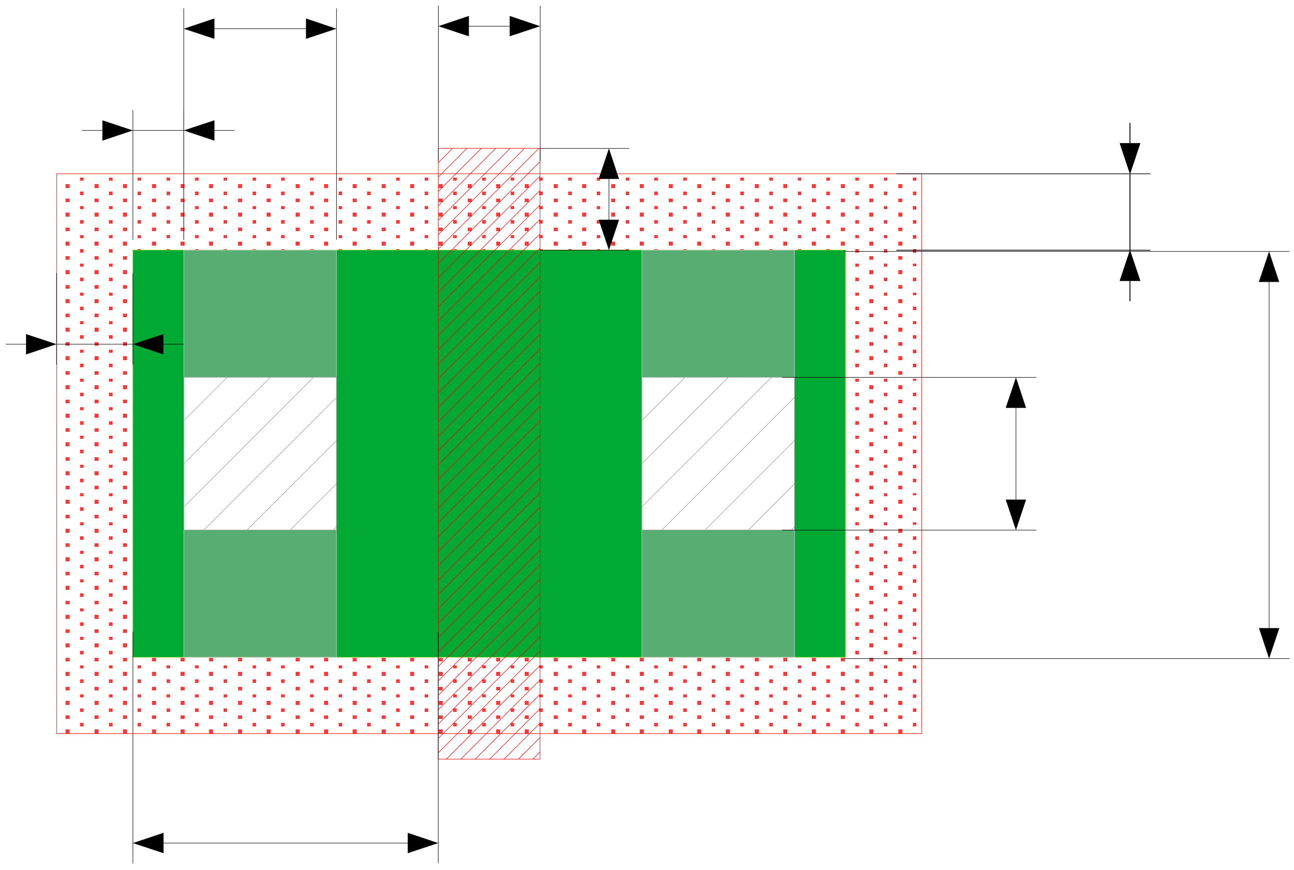
n-channel field-effect transistor

The following figure shows the structure of an n-channel field-effect transistor:

Label all rectangles in the log file with the name of the corresponding layer!



In the report file, enter the name of the corresponding design rule and the numerical value for each of the dimension lines drawn in the following drawing!



A SKILL file (nfet_psi.il) is provided. This file is a simple text file and can be edited with a text editor.
Except for the diff and nsdm layers, all rectangles are present with the necessary information.

The dimensions for the diff and nsdm layers still need to be entered. Add the necessary information to the layers!
Use the name of the respective design rule as the variable name, so that the drawing instructions for the rectangles only contain variable names!
The lower left corner of diff should be the zero point.

Open a new cell in Virtuoso called nfet_psi (Layout) and run the script!

Test your script and perform a design rule check! If error messages occur, correct the file until no more error messages appear! When testing the cells, remember to delete the existing cell in the layout before running the script.

Save your cell.

Copy your script additions and layout into the log file psi1_prot.ods. Creating a .png file is described in the tutorial.

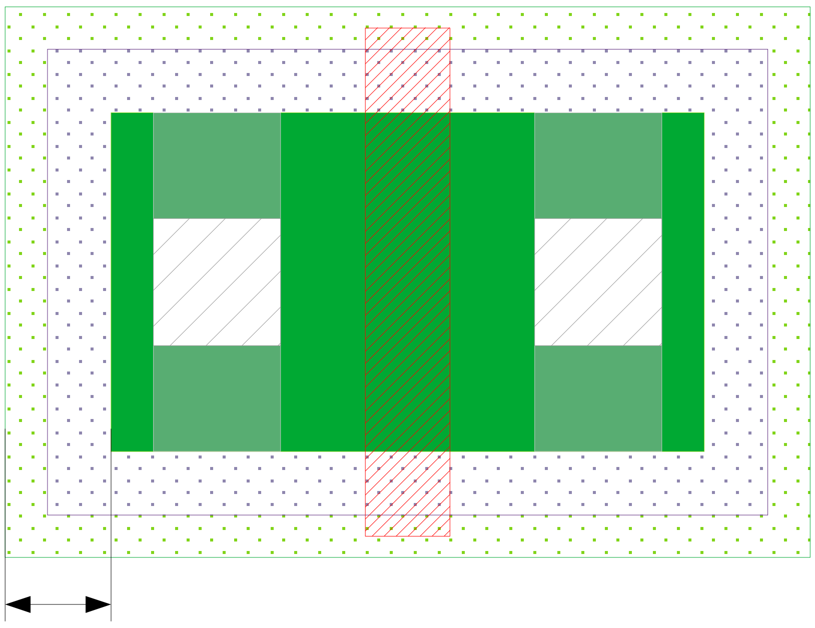
Testing the script

Design Rule Check
If error messages occur, the file must be corrected until no more error messages occur

When testing the cells, please note that you must delete the existing cell in the layout before running the script.

p-channel field-effect transistor

The following figure shows the structure of a p-channel field-effect transistor:



In the log file, label all rectangles with the names of the corresponding layers! Enter the name of the relevant design layer for the dimension line in the log file.

Rule and the value !

The SKILL file (pfet_psi.il) is also predefined. Here, too, the
diff and psdm layers must be added.
Test your script and perform a design rule check! If error messages occur, correct the file until no more error messages appear!

P-channel transistors generally have higher resistance than n-FETs for the same dimensions. Therefore, it makes sense to choose a larger channel width for p-FETs.
Which variable must be doubled to create a p-FET with double the channel width? Perform the modification and verify the result with the DRC!
Save your cell!
Copy the script and layout into the log file psi1_prot.ods

V.3.3 Contact holes

In order to contact the transistors, contact holes are required, which have their own design rules.

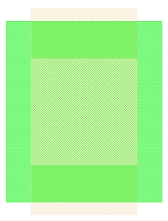
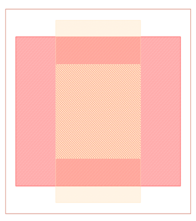
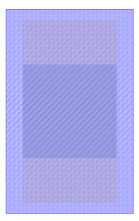
Question V3.3.1:
Which metal layer can be used to contact diffusion regions and polysilicon lines?

Question V3.3.2:
Which levels are required for contact between the lowest metal level and a diffusion region?

Question V3.3.3:
What levels are required for contact between the lowest metal level and a polysilicon line?

Question V3.3.4:
What levels are required for contact between the lowest metal level and a Metal 1 line?

Due to time constraints, the cells are made available for the contacts:

Contact Diffusion: diff_ct_psi 	Contact Poly: poly_ct_psi 	Contact Metal1 : m1_ct_psi 

All cells are now available to layout an inverter.
However, the "power rails" for supplying the inverter and the substrate contacts are still missing.

V.3.4 Design of the busbars for VDD and VSS

TAP contact p-substrate

Question V.3.4.1:
Which layers are needed to create a substrate contact for the p-type substrate? Which of the two supply voltages must be connected?

Create an empty cell named p_tapcon_psi !
First, a connection between li1 and TAP must be created. This can be created from the diff_ct_psi cell:
- Load cell diff_ct_psi into layout
- Rotate cell so that the li1 track becomes horizontal
- Cell diff_ct_psi "flatten (displayed levels)" to be able to edit the individual rectangles

- Transfer rectangle from layer diff to layer tap.

The width of the power rail should be approximately the same as that of the FETs. This allows two contacts to be placed side by side in the cell.

Question V.3.4.2:
Which design rule specifies the minimum distance from licon, and how large is it?
Which design rule specifies the minimum distance from mcon, and how large is it?

Copy the structure you've created so far and place it next to the existing one. Move the second cell so that the spacing matches the minimum spacing! You can draw the required spacing beforehand using the ruler.
Close the gap in the layers (easiest with "Stretch")! Run a DRC and correct any errors!

Question V.3.4.3:
The "psdm" layer must be above the "tap" layer. Which design rule specifies the overlap value, and how large is it?

Mark the appropriate distances with Rulers and draw the rectangle:
- Draw a rectangle around "tap" that roughly fits
- Exact adjustment of the rulers with "Stretch" Carry out

a DRC, eliminate any errors!

The existing cell contacts the "tap" layer with "li1." However, for large-area wiring, it must be possible to connect to "metal1."
Place a suitable contact (see default) over each "tap"-"li1" contact. Stacked contacts are allowed.
Run a DRC and correct any errors!
Copy the layout into the log file using a screenshot!

TAP contact n-well
Save the cell "p_tapcon_psi" under the name "n_tapcon_psi". Move the rectangle in the "psdm" layer to the "nsdm" layer.
Install the n-well according to the answer above and perform a DRC!

Question V.3.4.4:
Which design rule is relevant for the placement of the n-well?

An error message appears. Correct the p-well dimensions according to the error message!

Run a DRC and save the cell! Copy the layout into the log file using a
screenshot.

V.3.5 Structure of the inverter

Create a new cell (name inv_psi).
Position the cells you've created so far so that you can create an inverter with minimal dimensions! Note that in standard cells, all inputs and outputs must be routed to the Metal1 level! The power supply is provided via the power rails developed above.

Wire the cells to create a functional circuit. Wiring can be done using both rectangles and the "wire" function.
Run a DRC and save the cell! Copy the layout to the log file!

V.3.6 Structure of the ring oscillator

Create a new cell (name ringo_psi).
Position the cells you've created so far to create a 5-stage ring oscillator with minimal dimensions! The power supply is provided via the power rails developed above.

Wire the cells to create a functional circuit. Wiring can be done using both rectangles and the "wire" function.
Run a DRC and save the cell! Copy the layout to the log file!