# Edición de Circuitos en Electric VLSI y simulación en gnucap

Leandro Marsó

Córdoba

*elleandro@gmail.com*

11 de abril de 2015

# Contenido

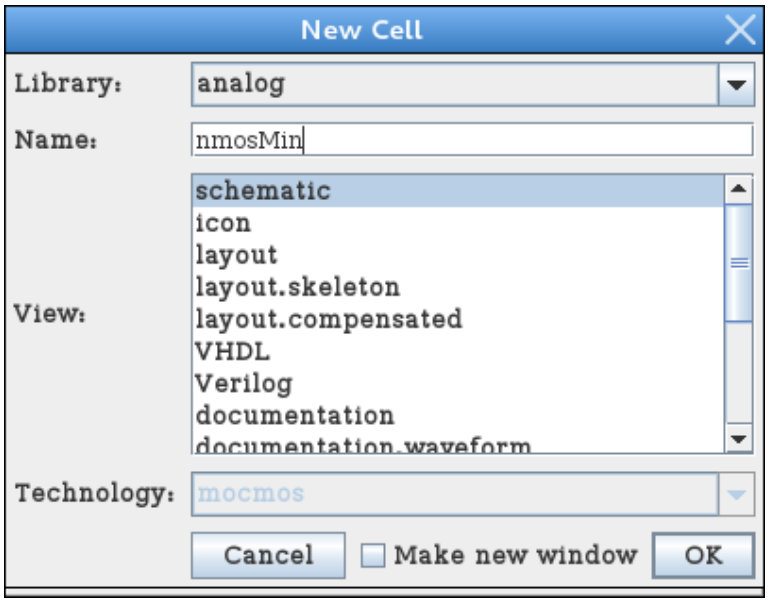
# Edición de Circuitos Esquemáticos

# Creación de una librería

Creamos la librería que contendrá las celdas que vamos a diseñar:
File $\rightarrow$ New library
Y elegimos el nombre **analog**

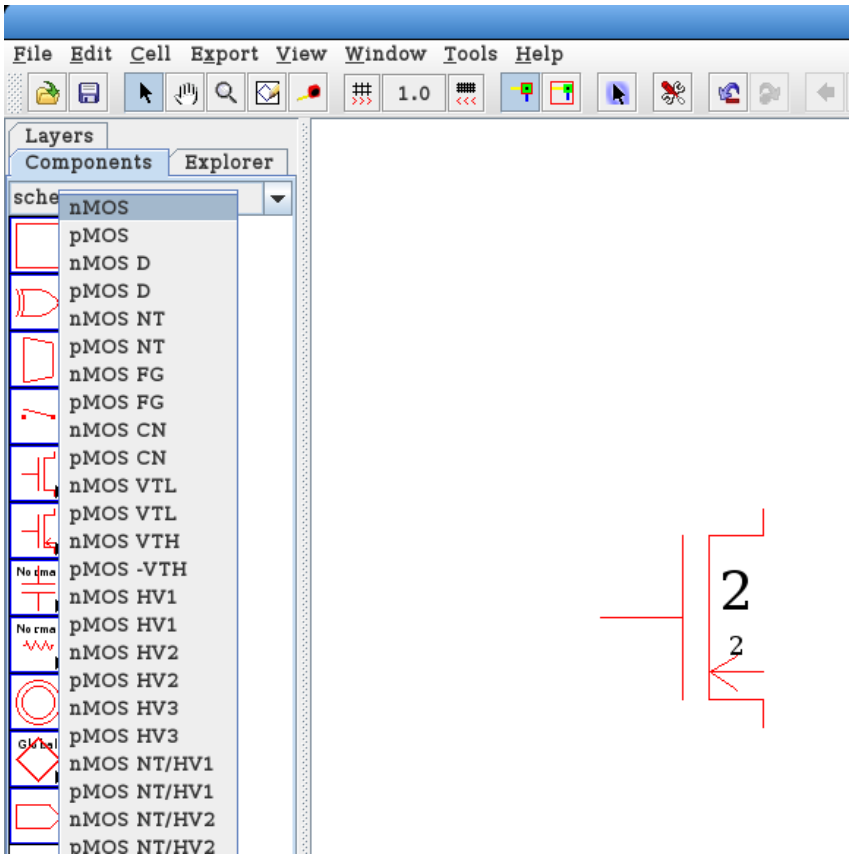
# Nueva Celda

Creamos el esquemático de la nueva celda, con el nombre **nmosMin**:
Cell → New Cell. Y en View: elegimos **schematic**



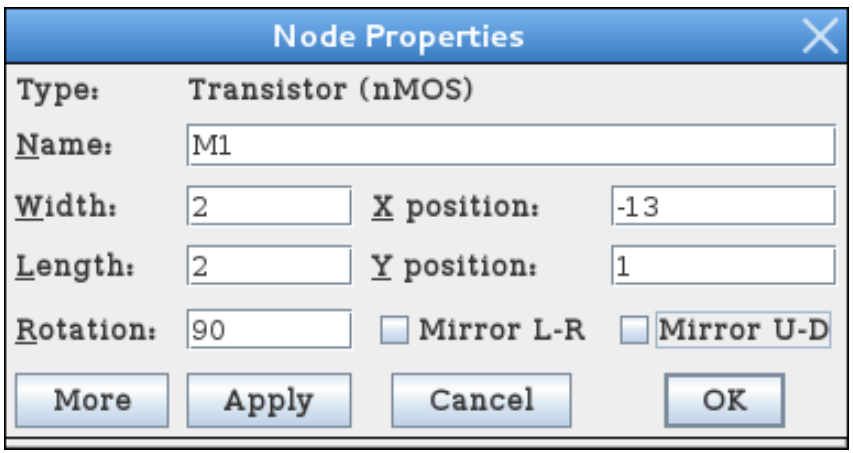
# Instanciar un transistor

Vamos a la pestaña Components y seleccionamos un transistor nmos de cuatro terminales y lo instanciamos en el plano de trabajo:



# Editar propiedades

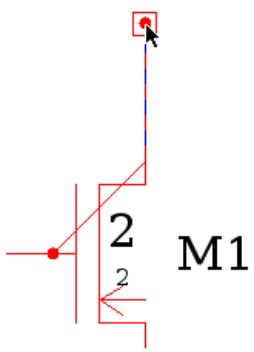
Para cambiar las propiedades del transistor hacemos **Ctrl-I** y cambiamos el nombre del transistor, y dejamos el ancho y largo en 2:



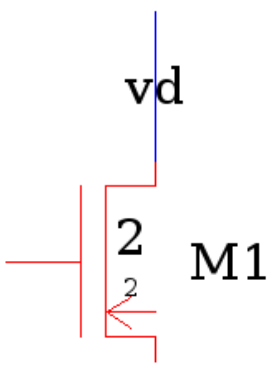
# Conectar y crear puertos

Ahora vamos a crear puertos y exportarlos para ejemplificar el uso del simulador y una convención para los nombres.

Conectamos el drenador haciendo click izquierdo sobre el mismo y luego hacemos click derecho sobre otro lado para conectar:

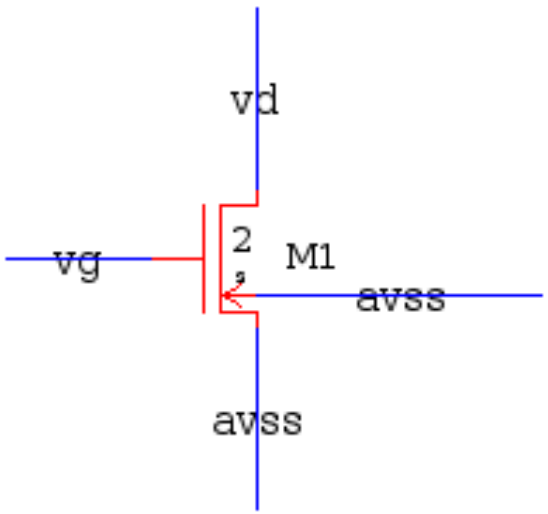



Nombramos el cable haciendo click izquierdo exactamente en el centro del mismo y luego **Ctrl-l** para cambiarle el nombre:

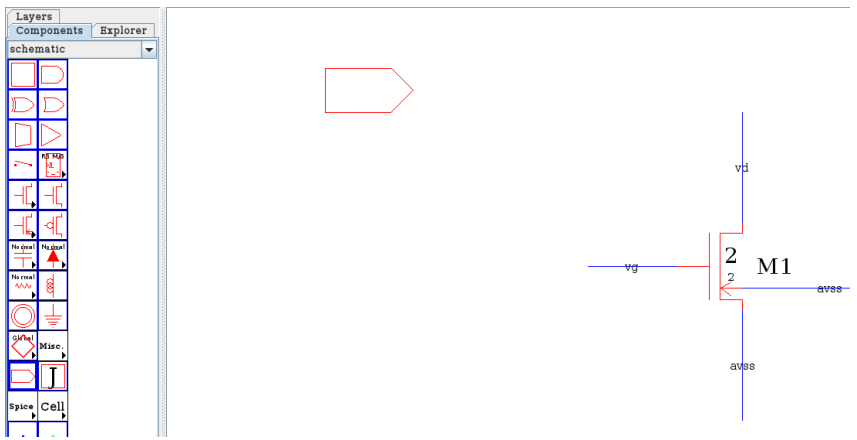


# Conectar y crear puertos

Nombramos todos los puertos del transistor de la siguiente forma:

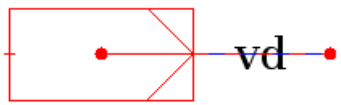


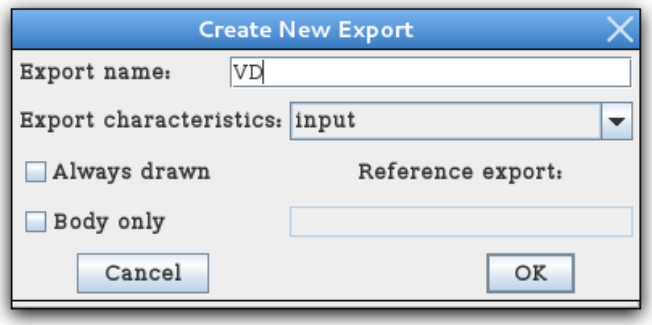

Ahora vamos a crear los puertos de esta celda, instanciamos un elemento llamado **off-page** como muestra la figura:



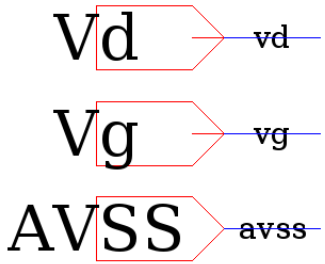
# Conectar y crear puertos

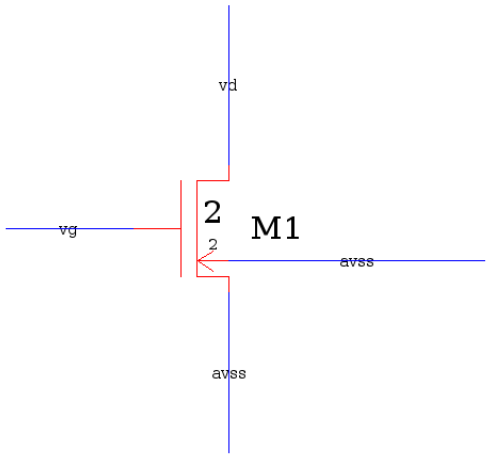
Creamos el puerto haciendo un click izquierdo sobre el elemento **off-page** y presionamos **Ctrl-E** para exportar este puerto y darle un nombre. Además conectamos un cable y le damos el mismo nombre que el del drenador del transistor:




Lo mismo hacemos para los otros puertos, como mostramos en la siguiente figura:

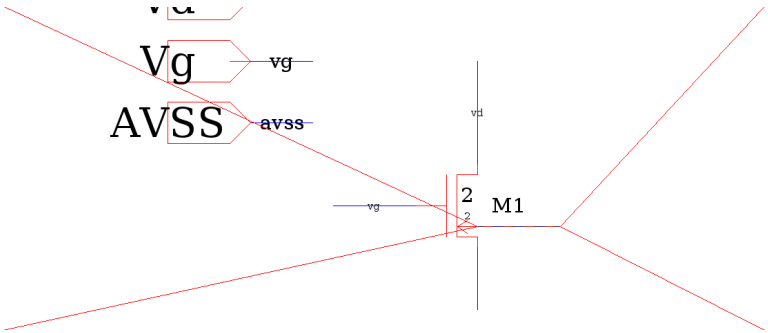


# Detectar errores de conexión

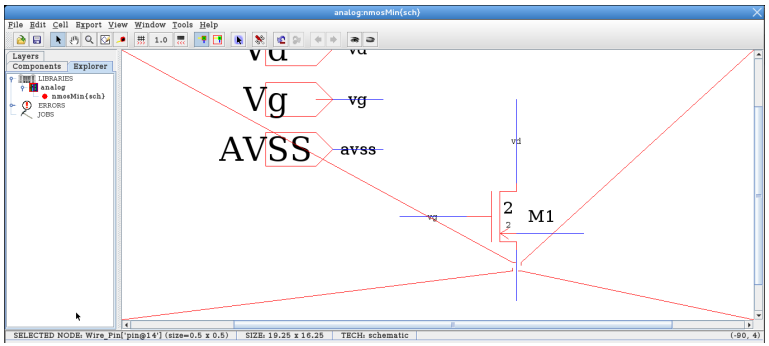
Si presionamos **F5** durante la edición, nos realizará un chequeo de errores, que pueden ser detallados presionando **Shift** $+ <$ o **Shift** $+ >$. Damos dos ejemplos típicos:

## Cables desconectados



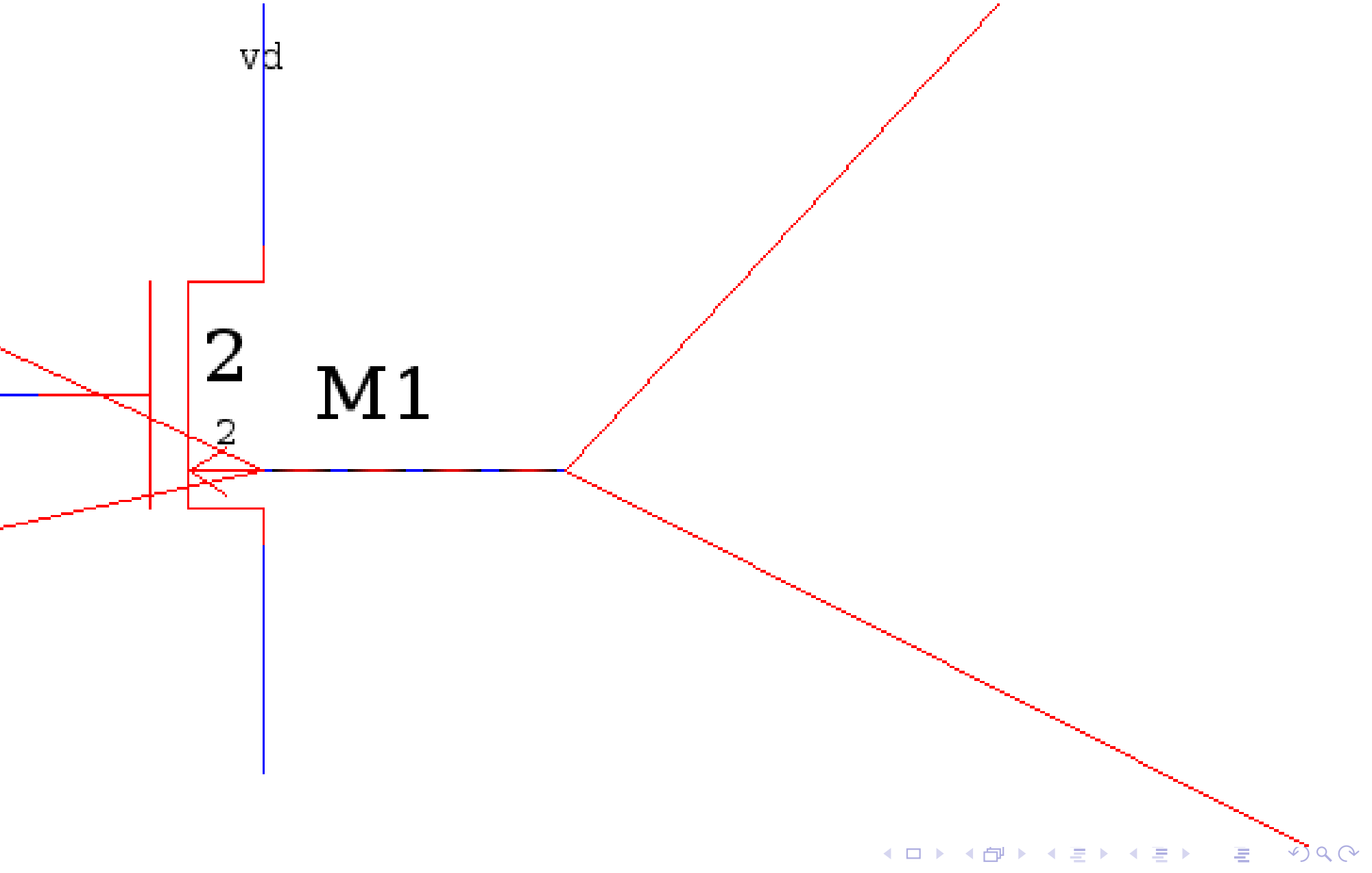

Este error aparece cuando un cable no tiene conexión o nombre.

## Pin innecesario

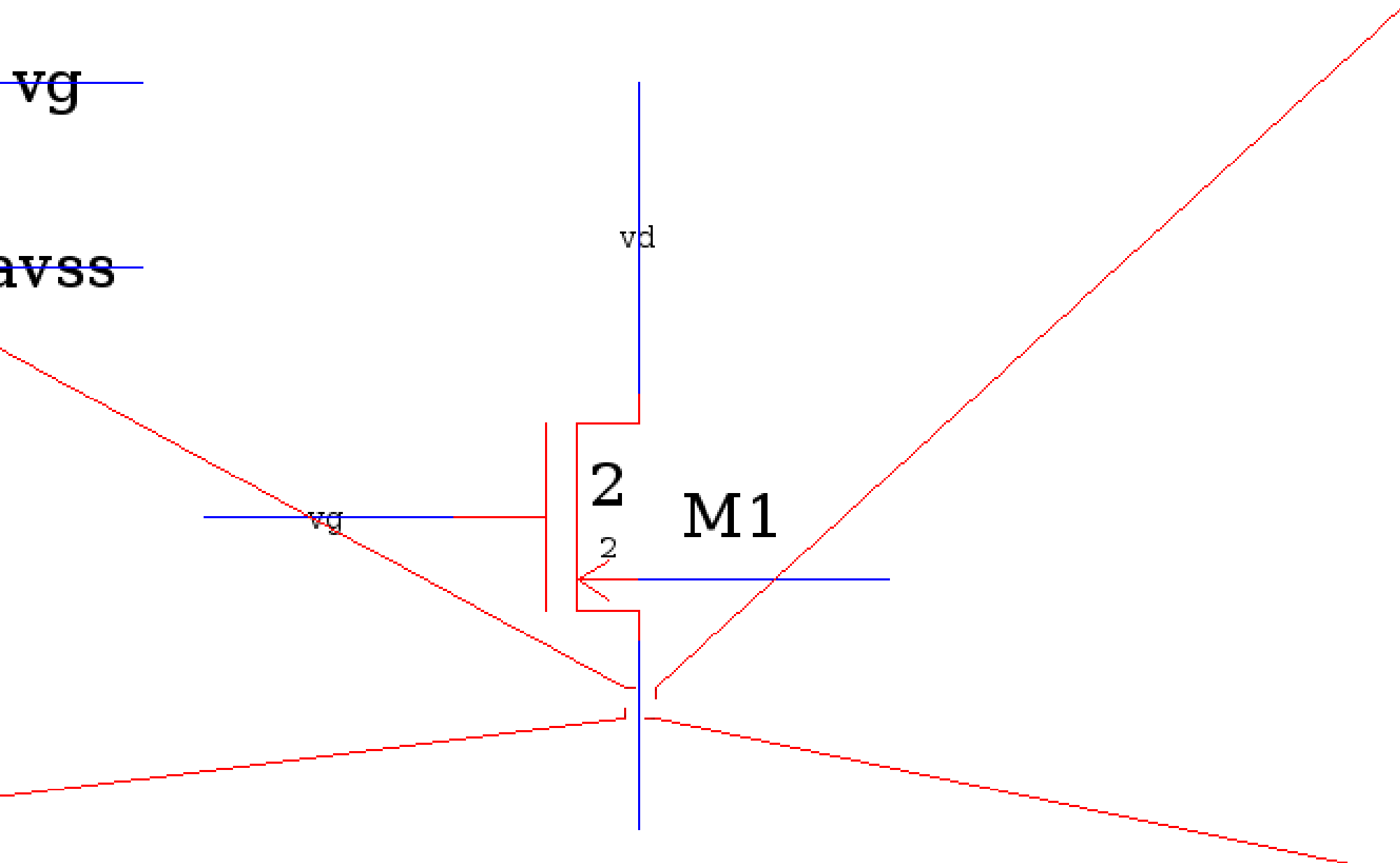



Este error se produce cuando hacemos un cable sobre la misma línea recta en dos tramos, es decir, presionando dos veces el click derecho para realizar un cable recto.

# Detectar errores de conexión



# Detectar errores de conexión



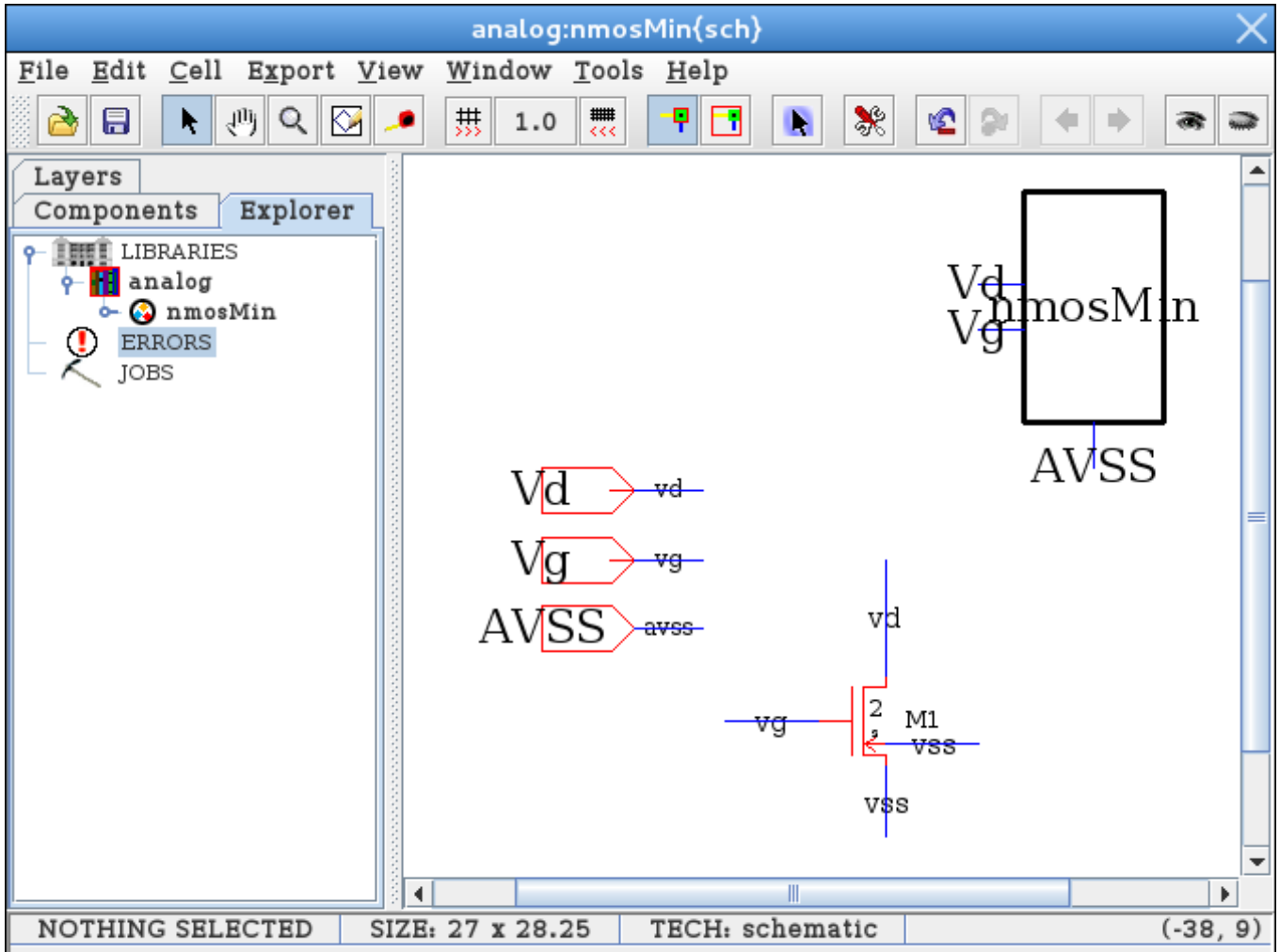
# Corrección de errores

Para corregir el error de pines extra:
**Edit** $\rightarrow$ **Cleanup cell** $\rightarrow$ **Cleanup Pin**

# Crear un ícono

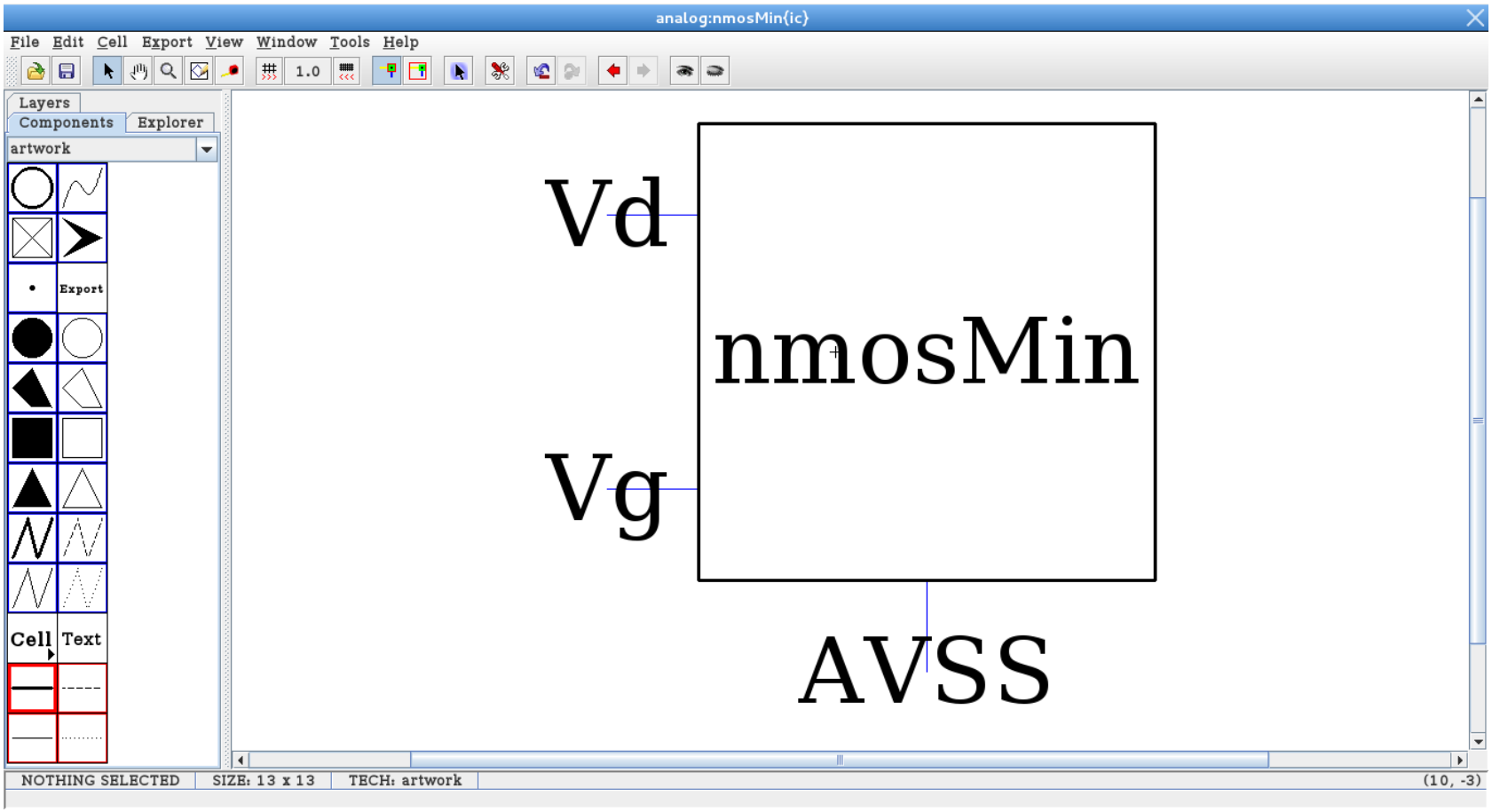
Luego de finalizar con el diseño esquemático, queremos tener un ícono para ser instanciado en otros esquemáticos:

**View → Make Icon View**



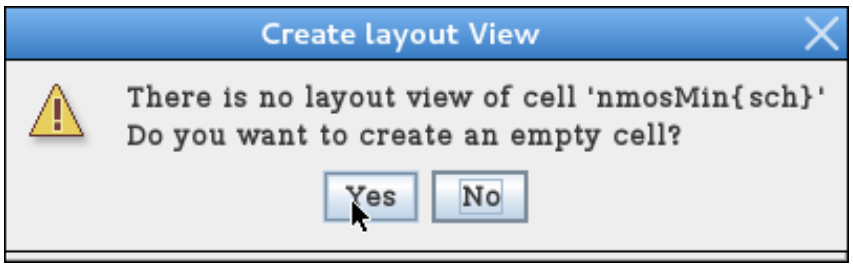
# Editar el ícono

Para editar este ícono hacemos: **View** $\rightarrow$ **Edit Icon View**
Seleccionamos el rectángulo y con **Ctrl-i** le ajustamos el valor de X para que sea cuadrado. Además movemos los puertos del ícono para separarlos y centrarlos:



# Creación del layout

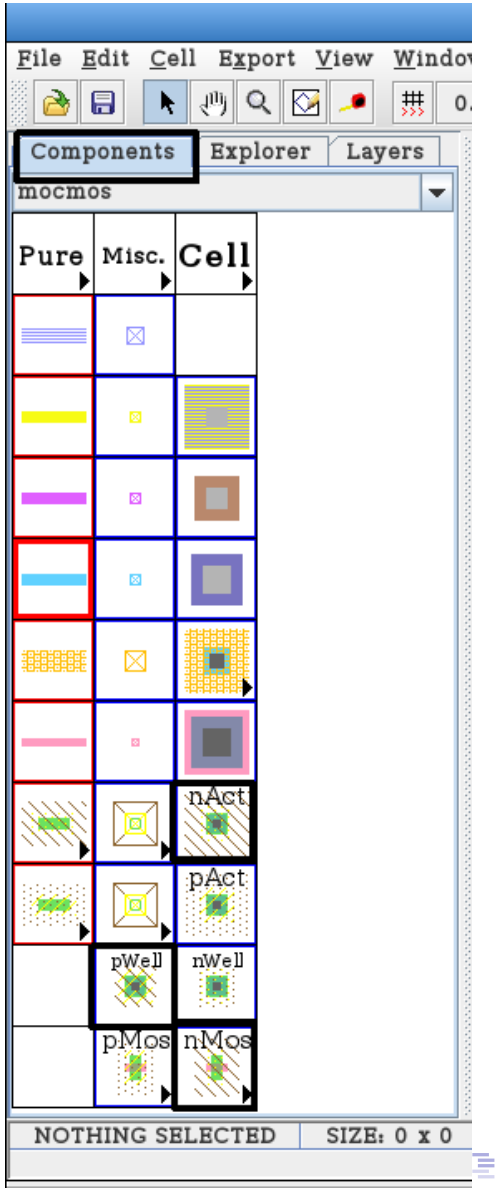
Para realizar la versión en **layout** de esta celda, hacemos:
**View** $\rightarrow$ **Edit Layout View**



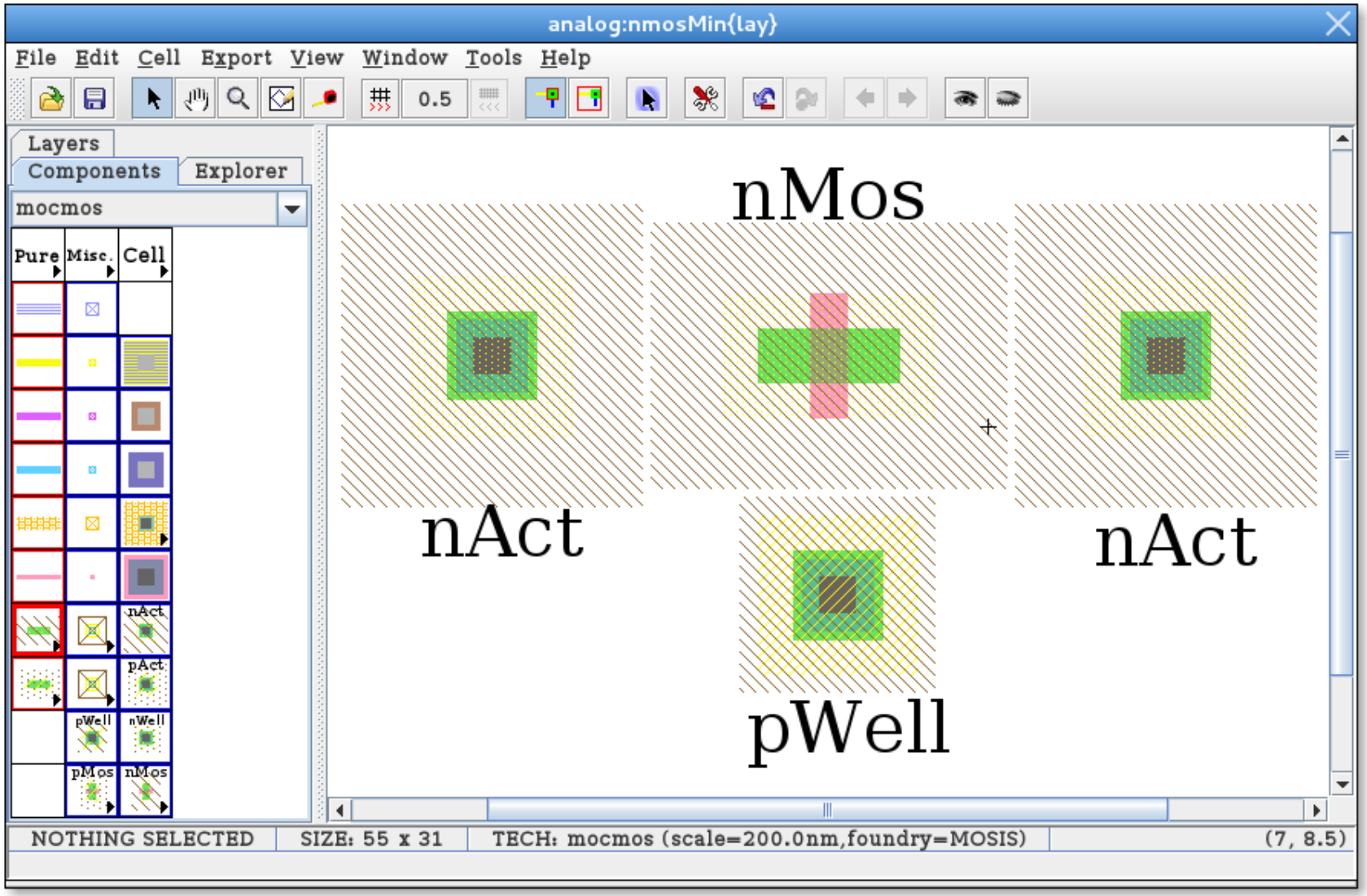

Y elegimos crear una celda vacía de layout.

# Creación del layout - Instanciar elementos

Vamos a instanciar un transistor tipo n (**nMos**), los contactos a Metal1 (**nAct**) para este transistor, y el contacto a bulk necesario (**pWell**). Esos elementos los encontramos en la pestaña de **Components**, como lo señalamos en la figura:

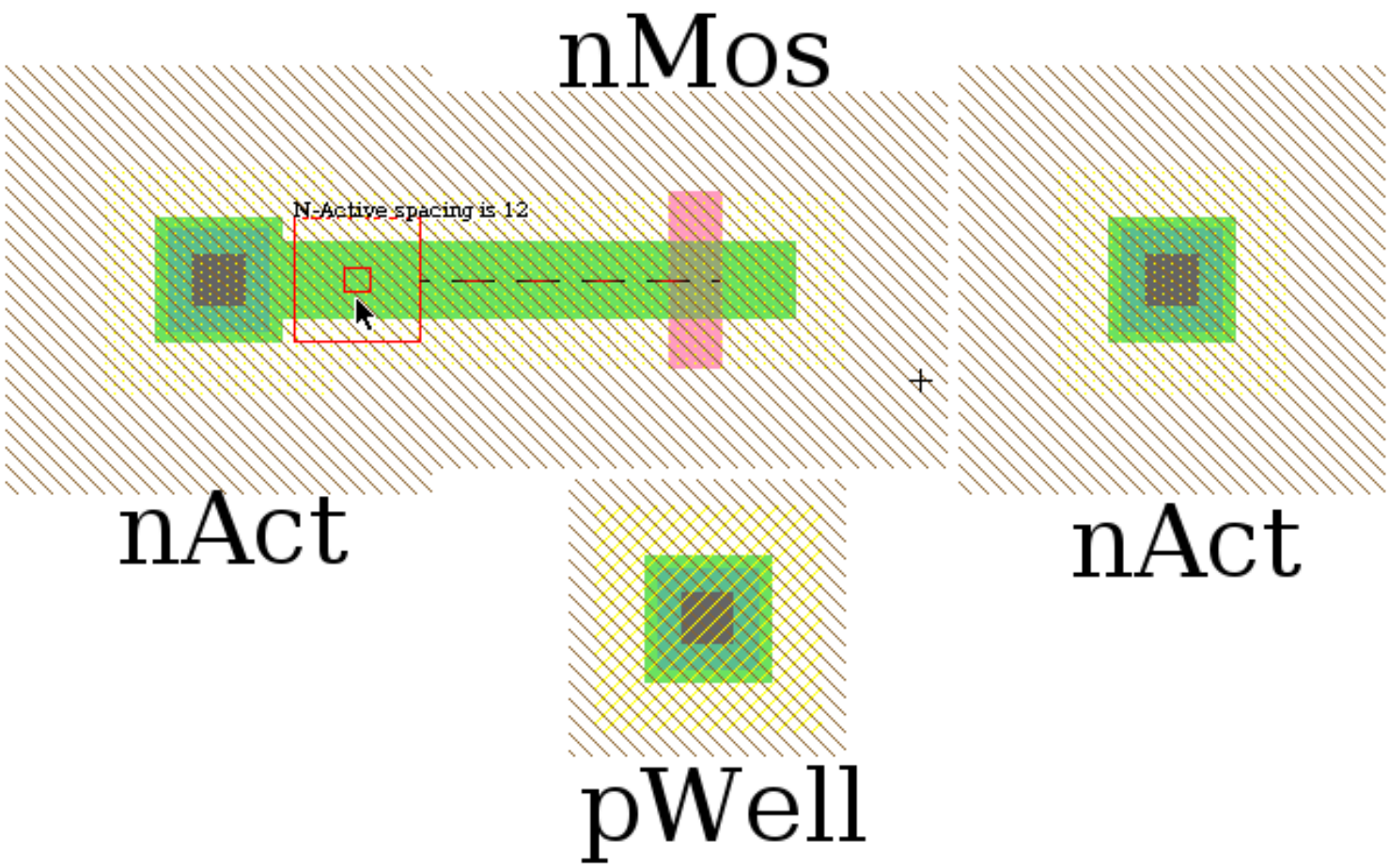


# Creación del layout - Instanciar elementos



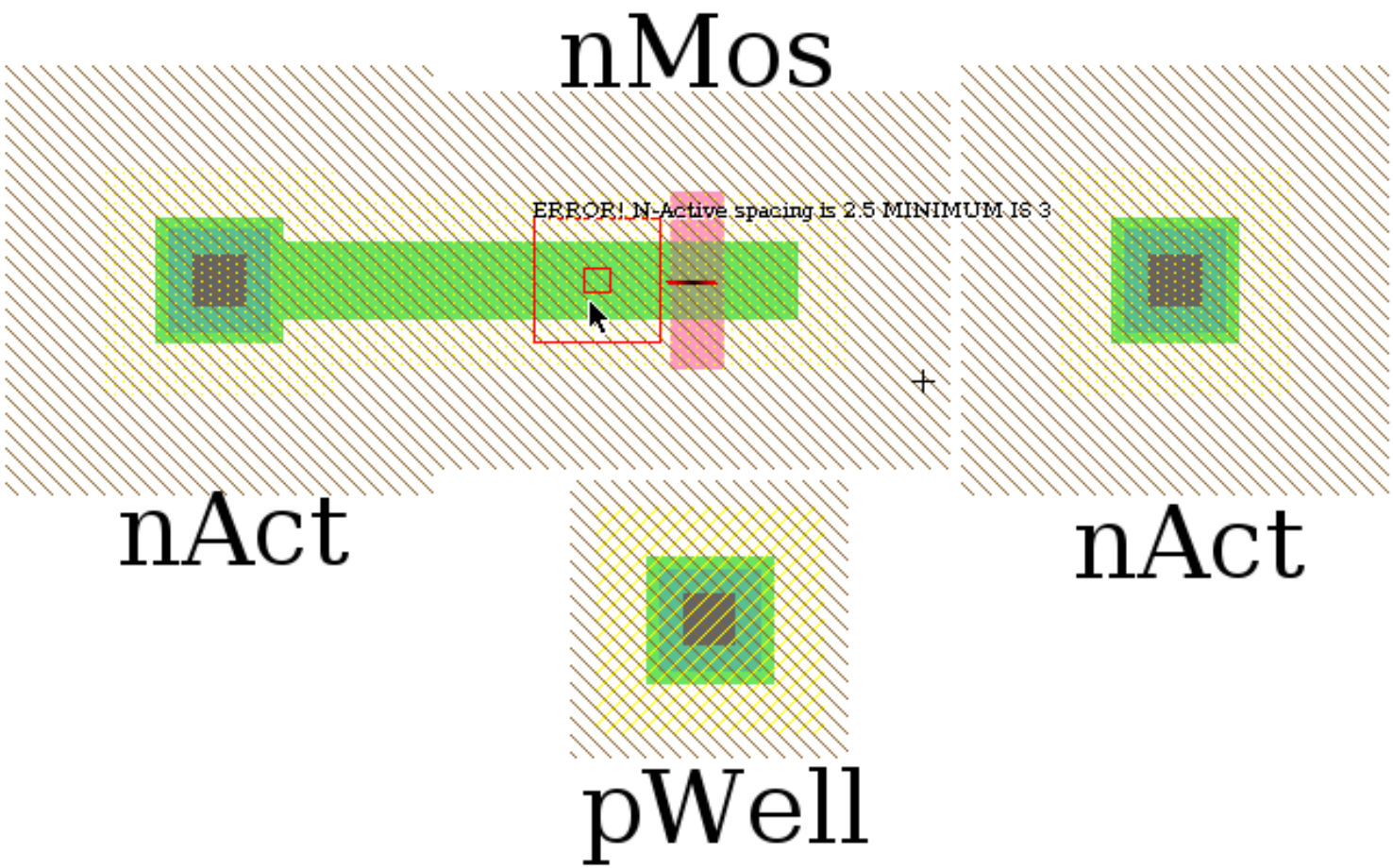
# Creación del layout - Contactos a Metal 1

Haciendo click izquierdo al contacto y arrastrando hacia la derecha para acercarlo al transistor, vemos una leyenda que nos dá información sobre la regla de DRC.



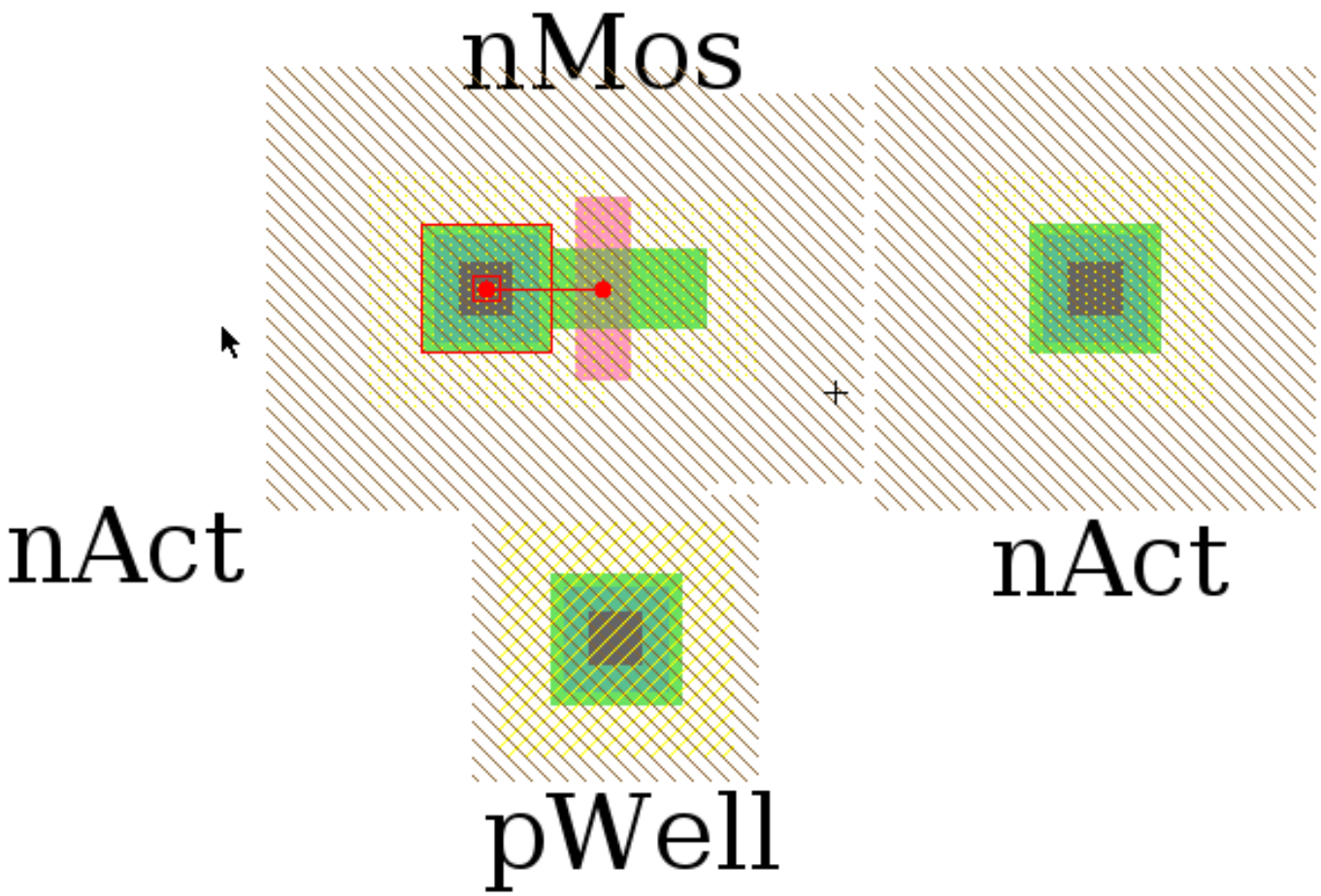
# Creación del layout - Contactos a Metal 1

Lo acercamos tanto como se pueda, hasta que nos indique que no cumple una regla de DRC:



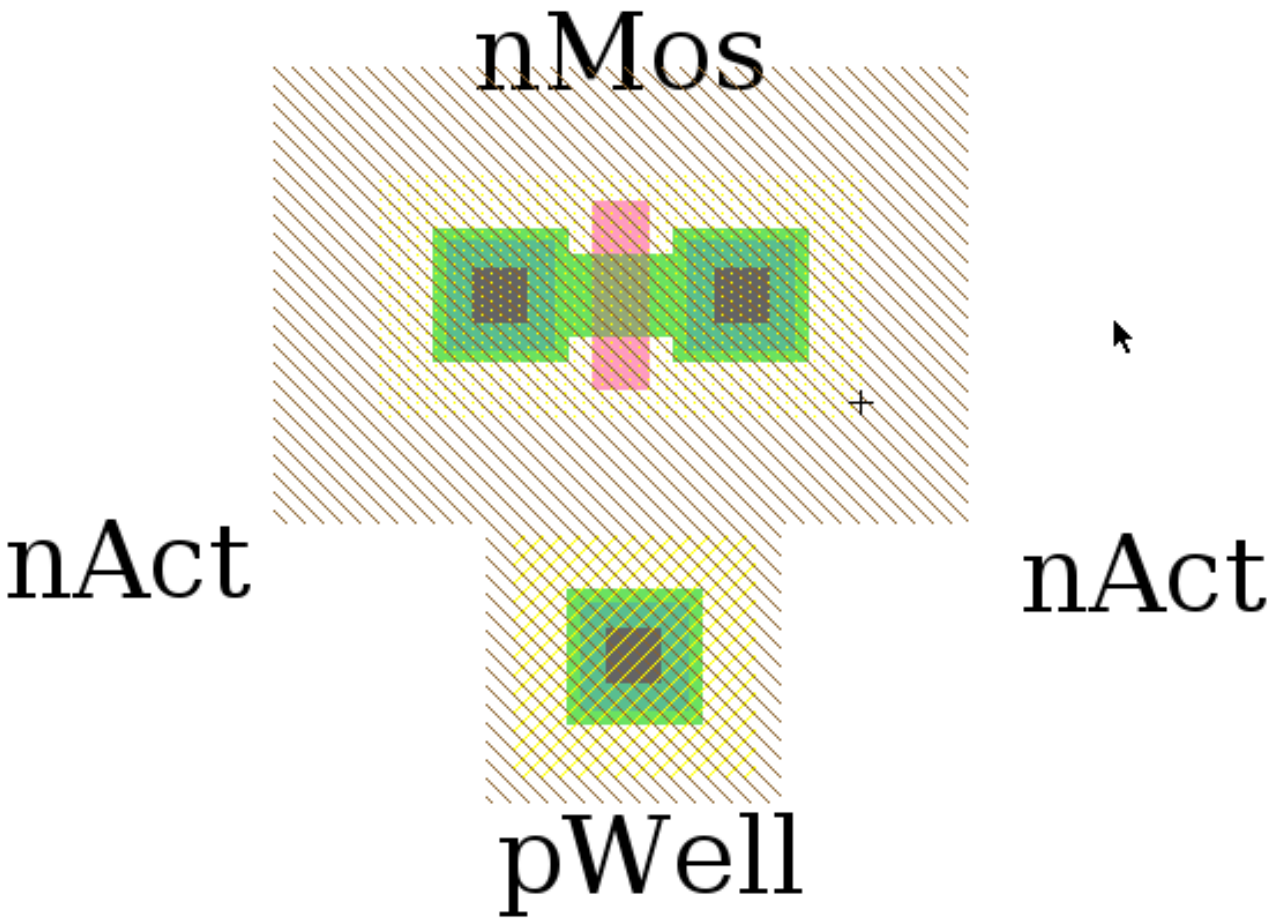
# Creación del layout - Contactos a Metal 1

Asi queda bien conectado:



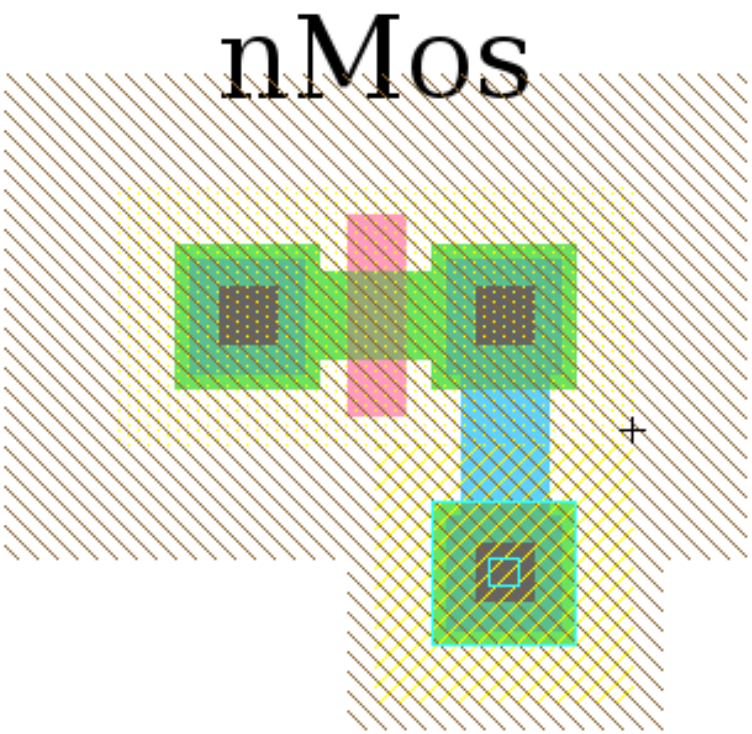
# Creación del layout - Contactos a Metal 1

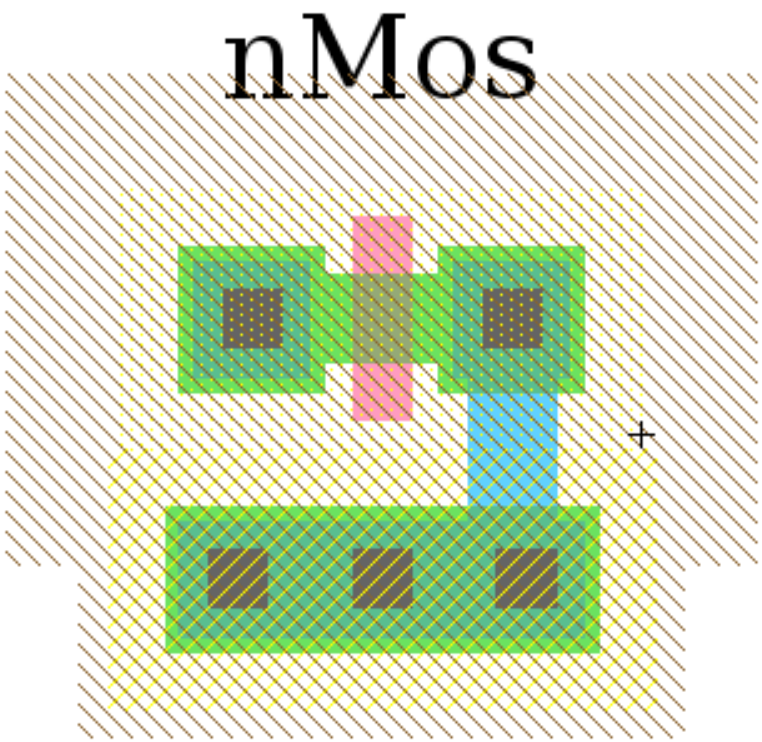
Hacemos lo mismo con el otro terminal:



# Creación del layout - Contactos a bulk

De igual forma conectamos el bulk[1] al source (siempre presionando **F5** para chequear DRC):

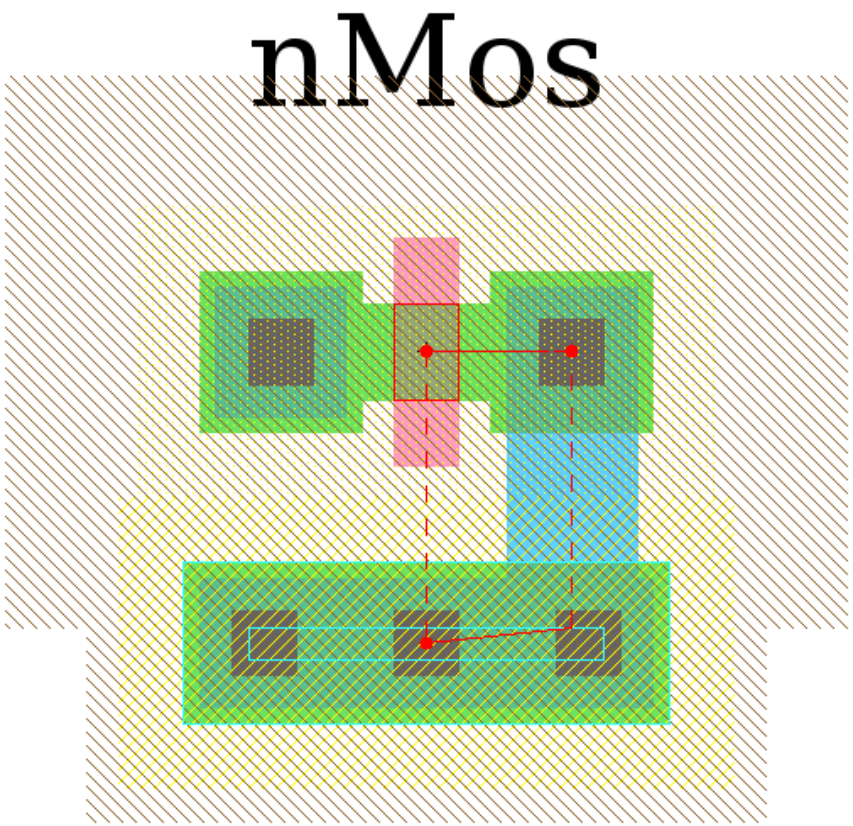



[1] Modificamos el valor de X al contacto a bulk para que sea más grande

# Conexión simbólica del *body* del transistor

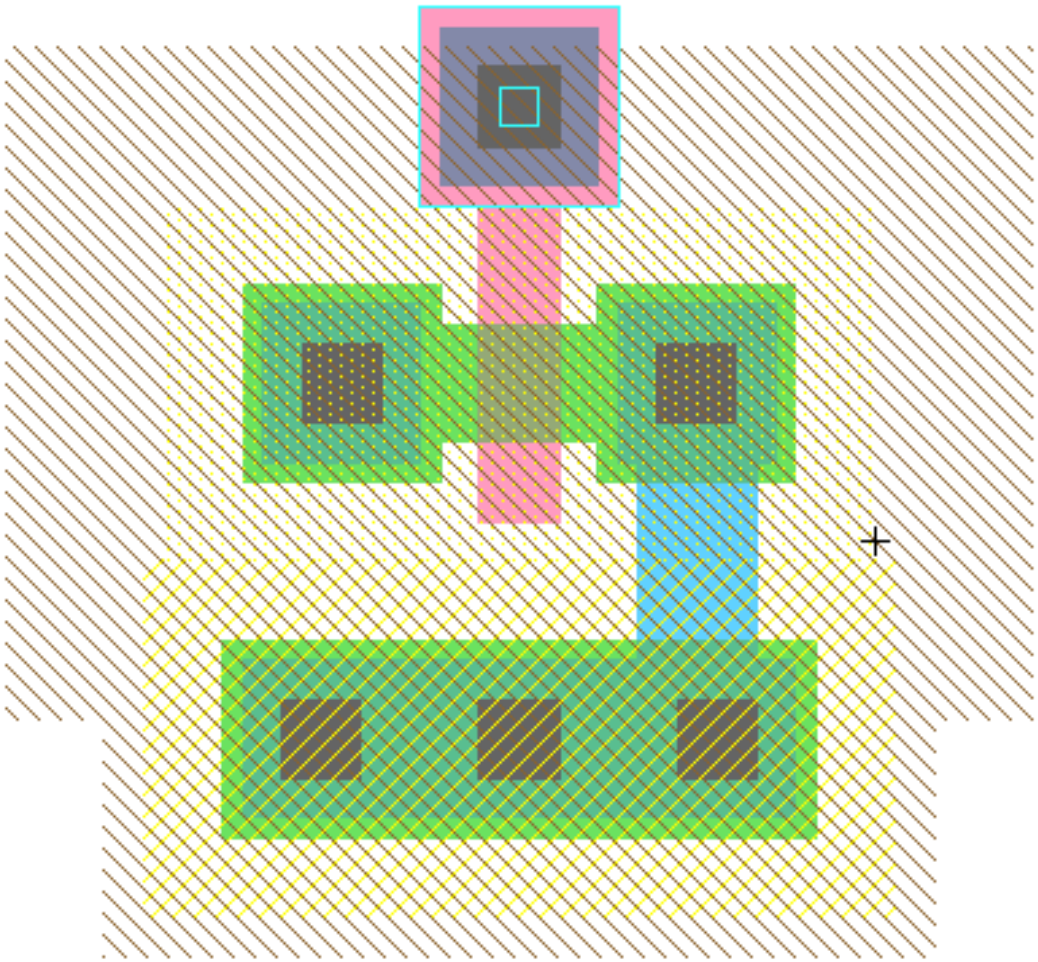
Seleccionar *Toggle Special Select*, hacer click izquierdo sobre el transistor y luego click derecho sobre los contactos a bulk[2].




[2]Para mas explicaciones ver **Body Checking Section** (Capítulo 9 del manual de Electric)

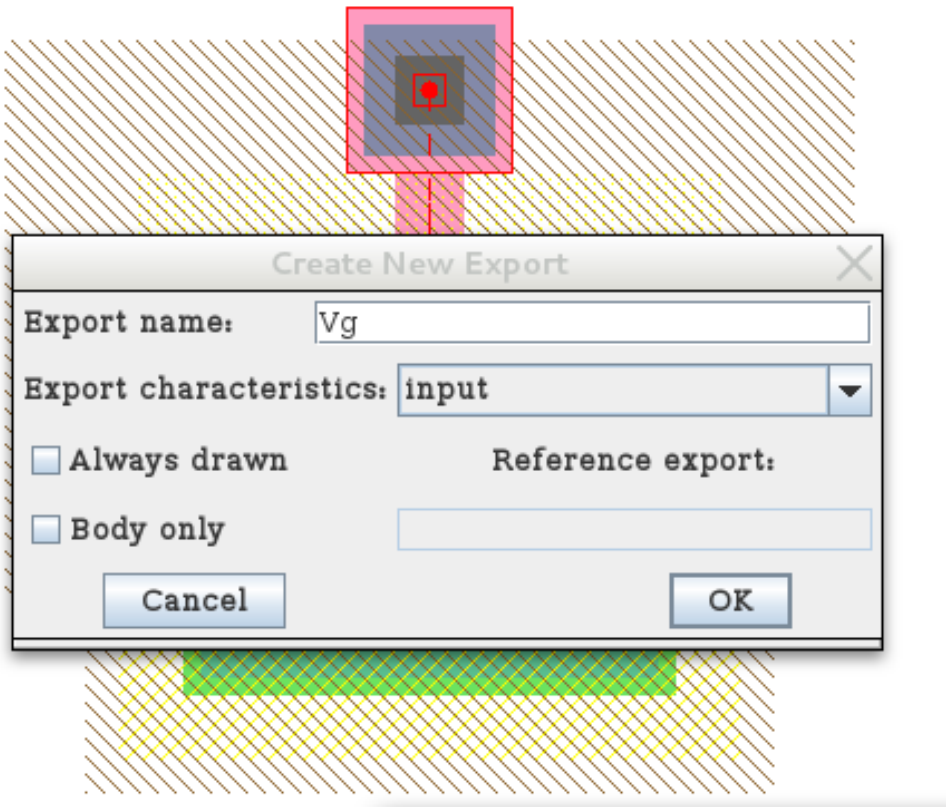
# Creación del layout - Contacto a poly

Instanciamos un contacto a poly y lo conectamos al gate:



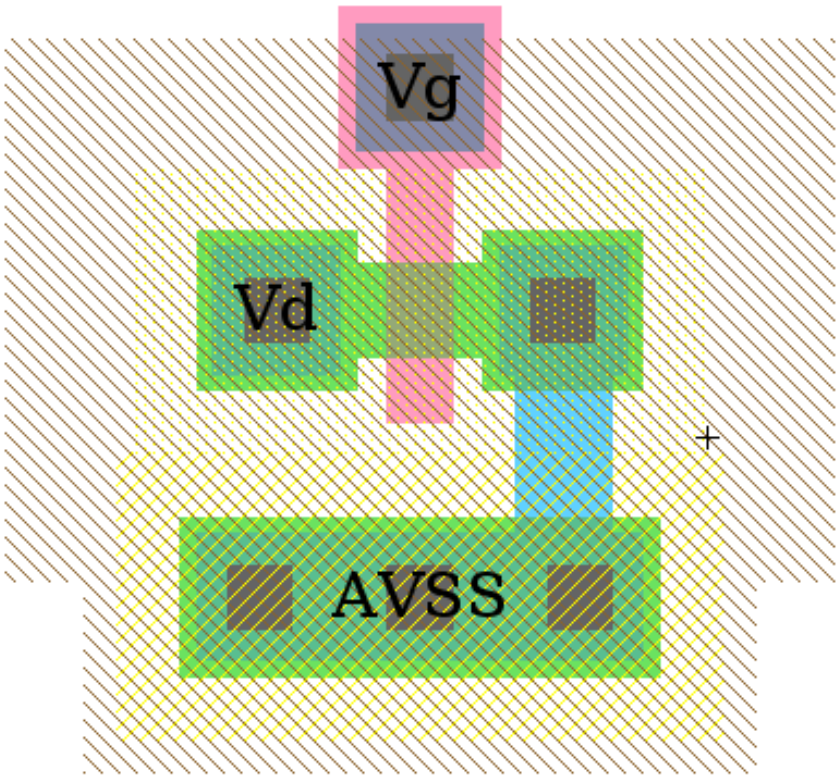
# Creación del layout - Puertos

Creamos los puertos con los mismos nombres que en el esquemático, seleccionamos primero el lugar donde queremos crear el puerto, y luego presionamos **Ctrl-e**:



# Creación del layout - Puertos

De la misma forma creamos el puerto para **Vd**, **Vg** y **AVSS**:



# Comprobación de errores de DRC

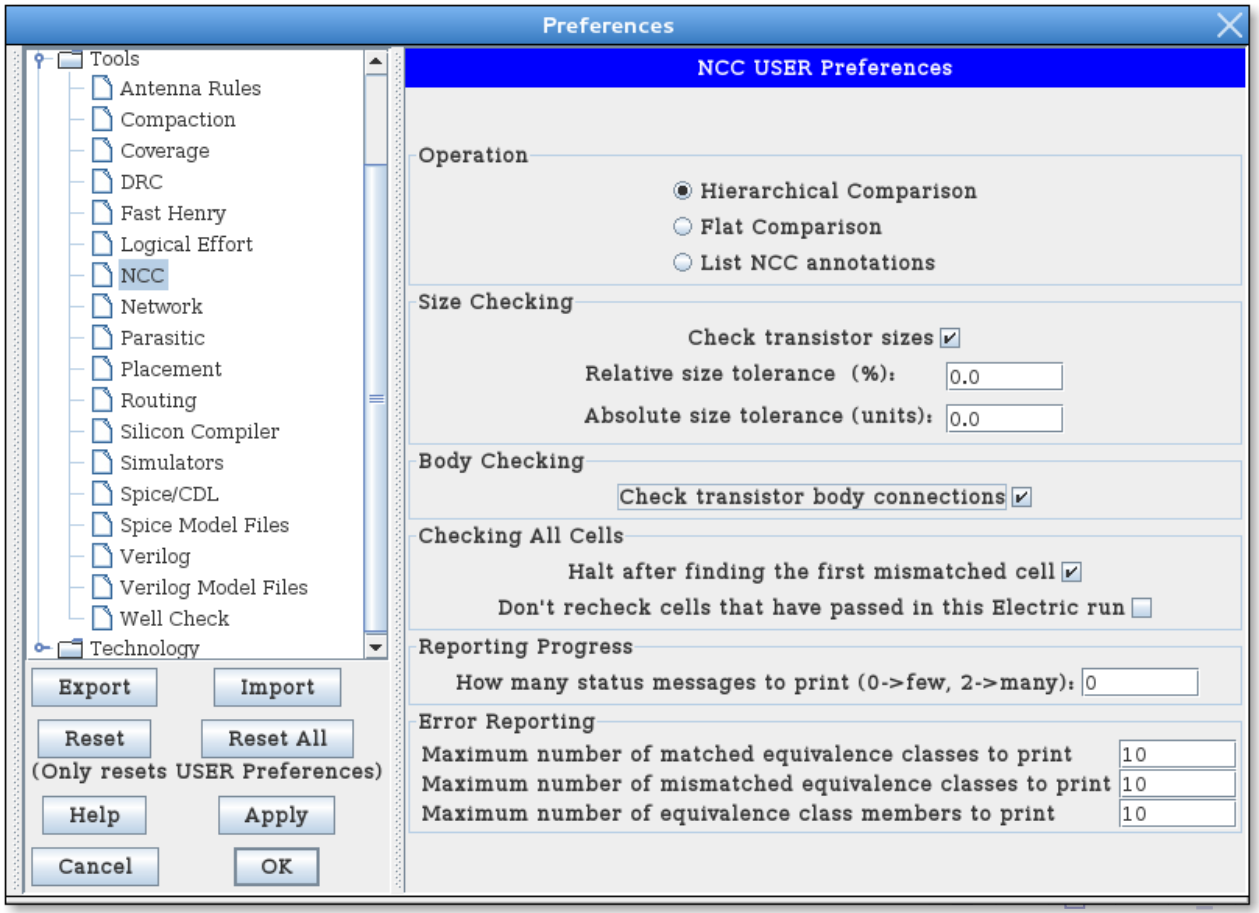
Durante la edición del *layout*, es necesario asegurarnos que no haya violacioes a las reglas de diseño, para eso presionamos **F5** y correjimos los errores a cada paso.

# Chequéo errores de LVS

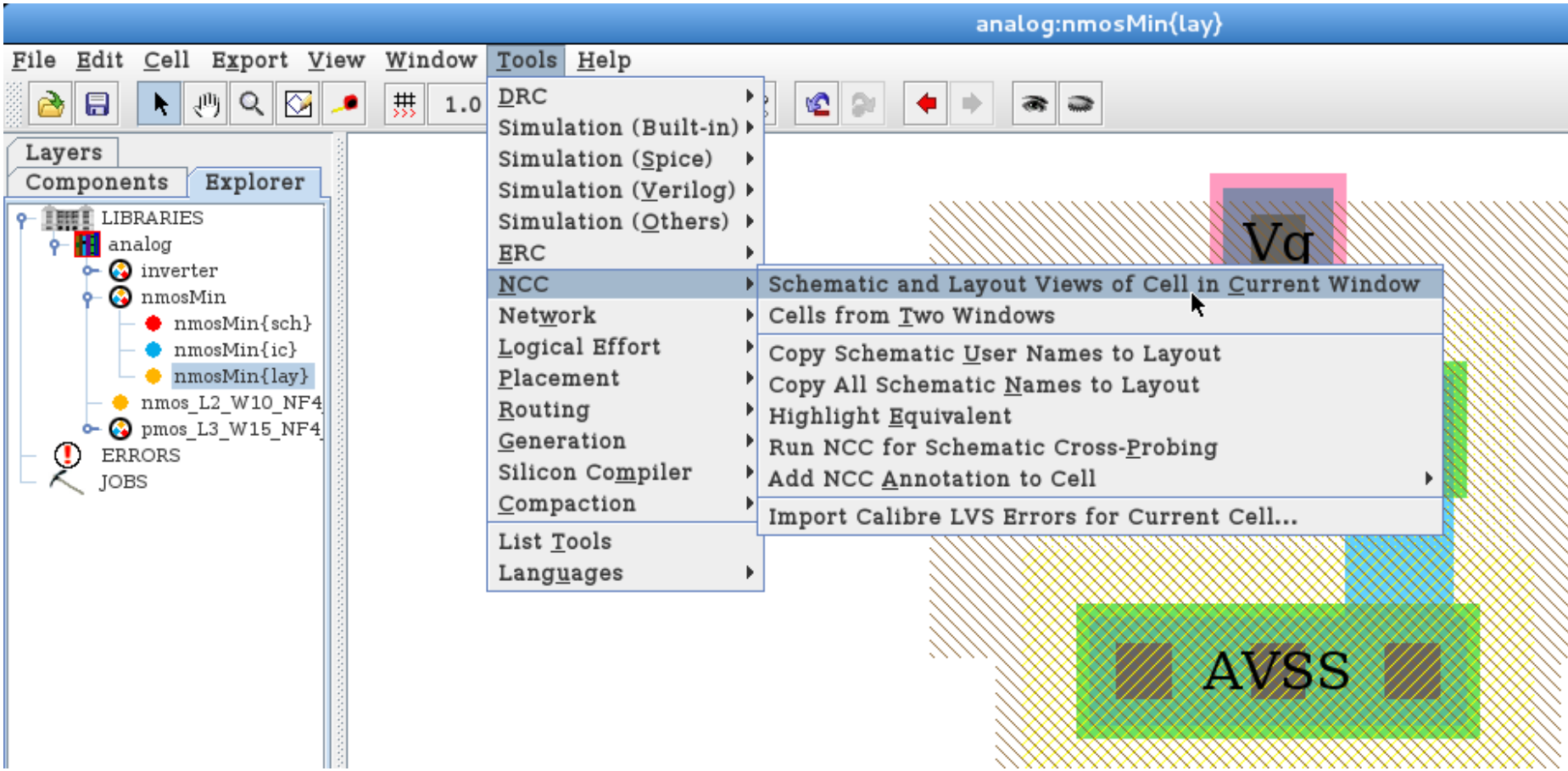
**Electric** cuenta con una herramienta llamada **NCC** (*Network Consistency Check* que nos permite realizar una verificación de equivalencia entre el esquemático y el *layout* de la misma celda. También sirve para hacer equivalencia entre esquemáticos, entre *layouts*, y otras vistas de la misma celda.

# Verificación de equivalencia entre el esquemático y el layout (LVS)

Primero configuramos correctamente la herramienta para que el LVS se haga correctamente. Vamos a **File** → **Preferences** → **NCC** y nos aseguramos seleccionar donde dice **Check transistor sizes** y en **Check transistor body connections**, como mostramos en la figura:



# Realizar el LVS



# Mensaje de LVS correcto

Si el esquemático y el *layout* son equivalentes, el mensaje será:

```
================================6================================
Hierarchical NCC every cell in the design: cell 'nmosMin{sch}'  cell 'nmosMin{lay}'
Comparing: analog:nmosMin{sch} with: analog:nmosMin{lay}
  exports match, topologies match, sizes match in 0.004 seconds.
  Summary for all cells: exports match, topologies match, sizes match
  NCC command completed in: 0.006 seconds.
```

# Extracción de los parásitos del layout

Realizamos ahora una extracción del circuito representado por el *layout* para crear un *netlist* SPICE, que incluya los elementos parásitos representados por los contactos, vías y metales utilizados para realizar las interconexiones.

**Extracción de parásitos**

**Tools → Simulation (Spice) → Write Spice Deck**

Luego especificamos donde guardar el archivo, y damos nombre al archivo. Creamos una carpeta llamada `sim` y le ponemos el nombre `nmosMin_lay.spi`. Luego examinar el archivo y ver el circuito y las resistores y capacitores que aparecen.

# Simulación del *Layout*

Creamos una nueva celda de *layout* llamada `nmosMin_tb` donde instanciamos nuestro transistor.

**Instanciar una celda de *layout***

**Components** → **Cell** → **nmosMin**{**lay**} (y ubicarla en el plano)

Exportamos los tres puertos con los nombre Vg, Vd y vss, y luego creamos el netlist para simulación:

**Creación del netlist para simulación**

**Tools** → **Simulation (Spice)** → **Write Spice Deck**

# Simulación del *Layout* - Creación de un testbench

Creamos el archivo `simulacion.gnucap` con el siguiente contenido: Descargar el testbench. Descargar el archivo fuentes.spi.

# Análisis DC

Para cualquier tipo de simulación, debo especificar si quiero ver o guardar algunas señales. Por ejemplo, recién hemos utilizado:

**Selecciono que señales mostrar**

```
print dc V(Vd) V(Vg) I(Vdd)
```

Para lanzar la simulación haciendo un análisis en DC:

**Comando de simulación**

```
dc Vdd 0 3.3 0.05 Vin 0.00 3.3 0.66
```

Documentación completa de este análisis y las opciones:

`http://www.gnucap.org/dokuwiki/doku.php?id=gnucap:manual:commands:dc`

# Lanzar la simulación

Para comenzar la simulación, desde una consola en el directorio `sim` hacemos:

```
$gnucap -i simulacion.gnucap
```

Al finalizar la simulación, **gnucap** nos devolverá el control y podremos continuar con mas simulaciones, o salir del programa presionando **Ctrl-d**.

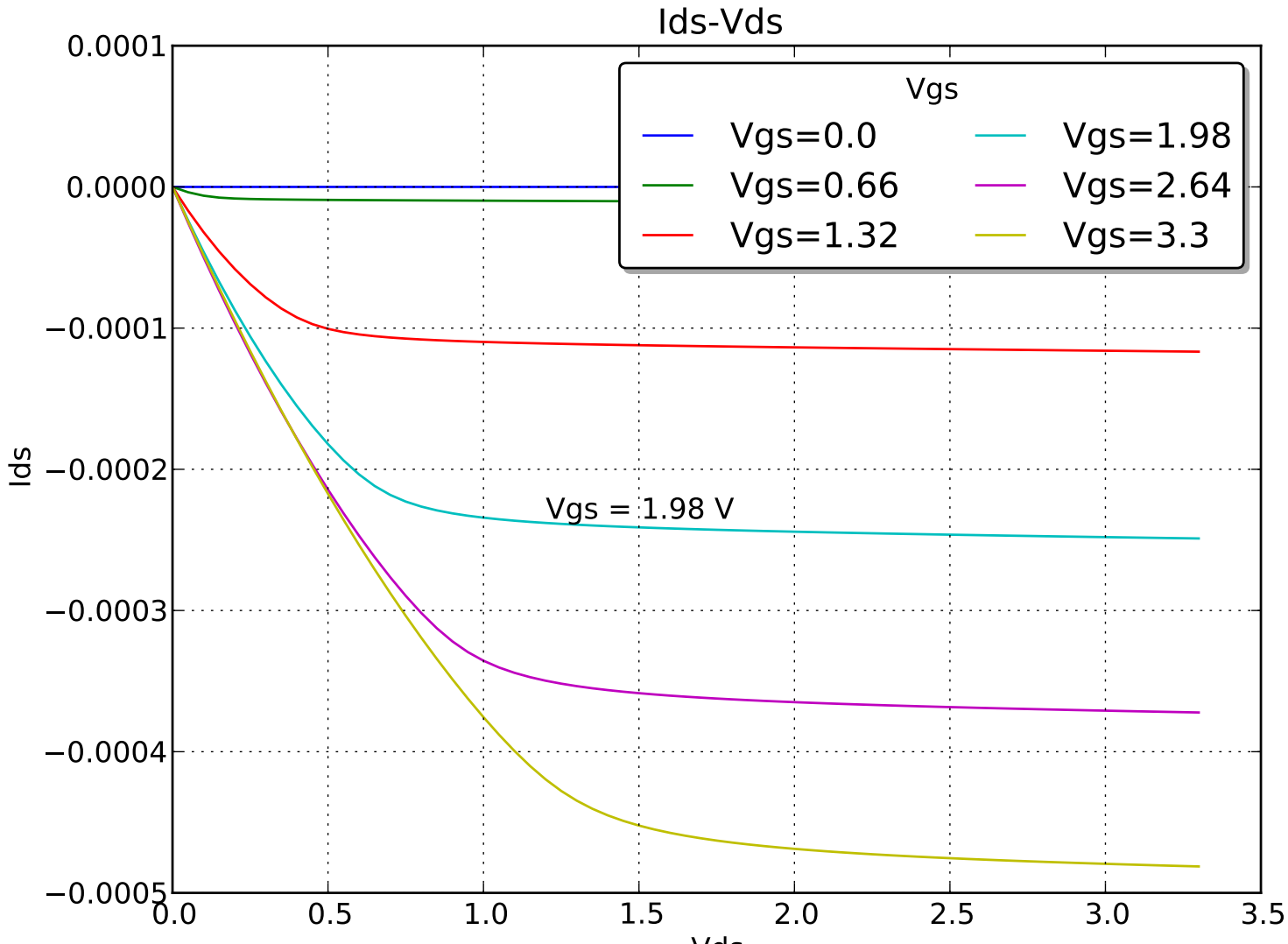
# Visualización de la simulación

Para visualizar los datos de la simulación que guardamos en el archivo simulacion.out optamos por utilizar un script en Python:

```
$python/plot_transistor_curves.py sim/simulacion.out
```

El resultado de esta simulación se puede ver en la filmina siguiente.

# Caracterización de los transistores de canal N y P por medio de simulación (familia de curvas de Id/Vds)



# Simulación del esquemático

La simulación del circuito esquemático se realiza de la misma forma que el *layout*, sólo que se debe crear el netlist Spice desde la vista de esquemático.

# Simulaciones de punto de operación

Para este comando, también es necesario determinar las señales que queremos ver, lo hacemos con el comando **print**

## Comando para agregar valores que se mostrarán

* Tensión en todos los nodos del circuito:

**print op v(nodes)**

* Agrega a la lista la corriente que da la fuente:

**print op +i(Vdd)**

# Más sobre el comando Print

Por ejemplo, cuando está instanciado el circuito lm_2stage_opamp, con nombre Xlm_2stag_0:

## Ver si un transistor está saturado

* Instanciación del circuito:
* Xlm_2stag_0 ( vss i_bias vdd out ) lm_2stage_opamp
* Para acceder a un elemento dentro del circuito, por ejemplo el transistor MM1:
* MM1.Xlm_2stag_0
* Por ejemplo quiero ver si está saturado ese transistor:

print op saturated(MM1.Xlm_2stag_0)

Más info sobre como llamar a los elementos internos de los circuitos en: `http://gnucap.org/gnucap-man-html/gnucap-man083.html`

# Simulaciones de punto de operación

Sintáxis:

```
op start stop stepsize {options ...}
```

## Algunos ejemplos

* Hacer una simulación a 27 grados:
op 27
* Hacer un barrido de temperatura desde -40 a 180 en pasos de 10 grados, haciendo una simulación de OP en cada paso:
op -40 180 10

Existen muchas opciones, por ejemplo que el barrido sea logarítmico, inverso, cíclico, etc. Para mas opciones ver:

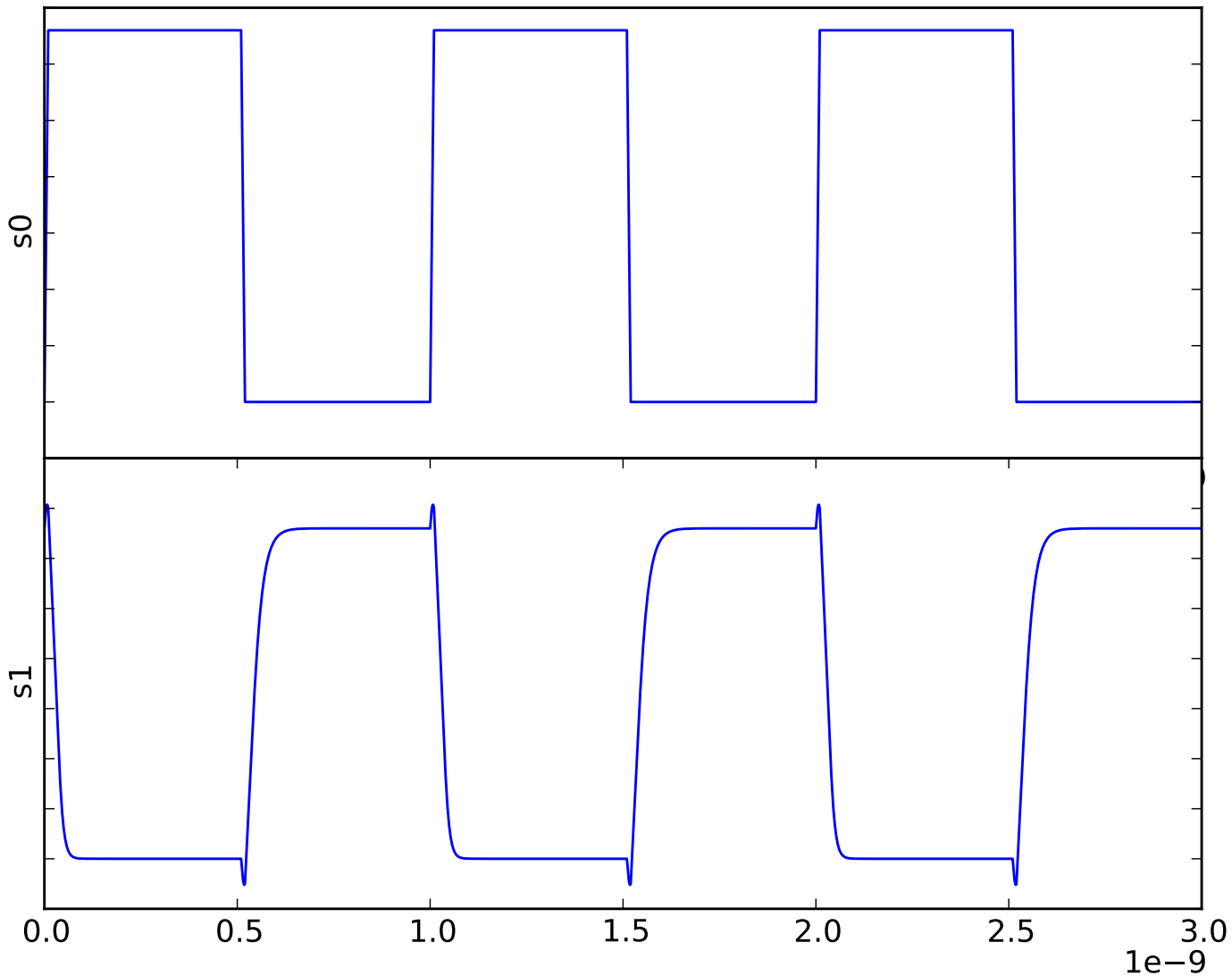
http://www.gnucap.org/dokuwiki/doku.php?id=gnucap:manual:commands:op

# Régimen transitorio

Simulación de un inversor: Hacemos la extracción del layout de nuestro circuito **inv** y luego creamos el testbench para gnucap:
Ver el testbench.
Ver el archivo de exitación.
Ver el archivo con la alimentación general
Para lanzar la simulación, hacemos:

```
$gnucap -i inv.gnucap # Desde el directorio
                      # donde estan todos los archivos
```

# Resultado de la simulación



# Análisis AC

Este es un análisis de pequeña señal, que nos permite hacer un barrido en frecuencia. Es obligatorio antes de realizar este análisis, hacer una simulación de punto de operación.

**Sintáxis**

ac options ... start stop stepsize options ...

Documentación: `http://www.gnucap.org/dokuwiki/doku.php?id=gnucap:manual:commands:ac`

# Transformada de Fourier

Realiza un análisis transitorio, pero muestra los resultados en el dominio de la frecuencia.

## Sintáxis

fourier start stop stepsize options ...

Documentación:

`http://www.gnucap.org/dokuwiki/doku.php?id=gnucap:manual:commands:fourier`

# Instalar paquetes extras

Desde una consola:

**Instalar mas herramientas**

```
sudo apt-get install perl python git easygit
```

# Fin