

アナログ開発ツールの使い方 (OpenRule1um)

森 瑞紀 (Mizuki Mori) , <https://github.com/3zki>

今村 謙之 (Noritsuna Imamura)

Rev.1

本日のルール

- チップ搭載優先順位
 - 本日中にレイアウトのDRC,LVSまで終了した方の中で
 - 申込時に「製造あり」の枠で参加された方
 - 申込時に「製造抽選」の枠で抽選に漏れた方
 - 申込時に「製造なし」の枠で参加された方

今後の予定

- 本日
 - 提出物：回路図、レイアウト、githubのアカウント、連絡先（チップ送付先）
- 数日中
 - 提出物：公開用の一言など
- 来年一月～二月
 - イベント：チップが届きますので、お渡し会を兼ねたチップ測定会を開催します

アジェンダ

- 開発ツールの紹介、PDKの中身
- アナログLSIの設計フロー
- アナログLSI設計デモンストレーション (CMOSインバータ)

アナログLSIの設計フロー

どのような設計をするか、どのような開発ツールを使用するか

アナログLSIの設計フロー

1. 回路図（テストベンチ）を描く
2. シミュレーションをする
3. 回路図を基にレイアウトを描く
4. レイアウトを検証する (DRC, LVS)
5. レイアウトを基に寄生成分を考慮したシミュレーションをする OpenRule1um では未対応
6. フレームに載せる

そもそもLSI回路設計に必要なものとは？

- **プロセスデザインキット PDK**
 - シミュレーションモデルライブラリ (**SPICE**)
 - 検証ツール用ルールファイル (DRC, LVSなど)
 - スタンダードセルライブラリ
 - レイアウト (GDS)
 - ネットリスト(**SPICE**)
 - 自動配置配線ルール(LEF)
 - Verilogシミュレーションライブラリ (LIB, V)
 - 一部指定されたレイアウト (フレームなど)
- **PDKで指定された各種ツール** (EDAツール)
 - **回路図エディタ** or Verilogコンパイラ
 - **回路図エディタ** : xscem, Glade, LTspice, KiCAD
 - **レイアウトエディタ** or 自動レイアウトツール
 - **レイアウトエディタ** : klayout, Glade
 - **SPICEシミュレータ** or Verilogシミュレータ
 - **SPICEシミュレータ** : ngspice
 - **検証ツール**
 - **検証ツール** : klayout

PDKの場所

公式レポジトリ

- Glade(回路図&レイアウト), KiCAD(回路図)用PDK
<https://github.com/MakeLSI/OpenRule1um>
- Klayout用PDK (レイアウトエディタ)
<https://github.com/mineda-support/OpenRule1um>
- Qflow用PDK (デジタル自動配置配線)
<https://github.com/mineda-support/OR1>
- Glade, KiCAD, Klayout用スタンダードセル
https://github.com/MakeLSI/OpenRule1um_StdCell
- LTspice用PDK (回路図 ,スタセルシンボル)
https://github.com/MakeLSI/OpenRule1um_StdCell_LTspiceSymbol
- OR1実測データ (Spiceモデルあり)
<https://github.com/MakeLSI/Measure>

ISHI-KAIオリジナル

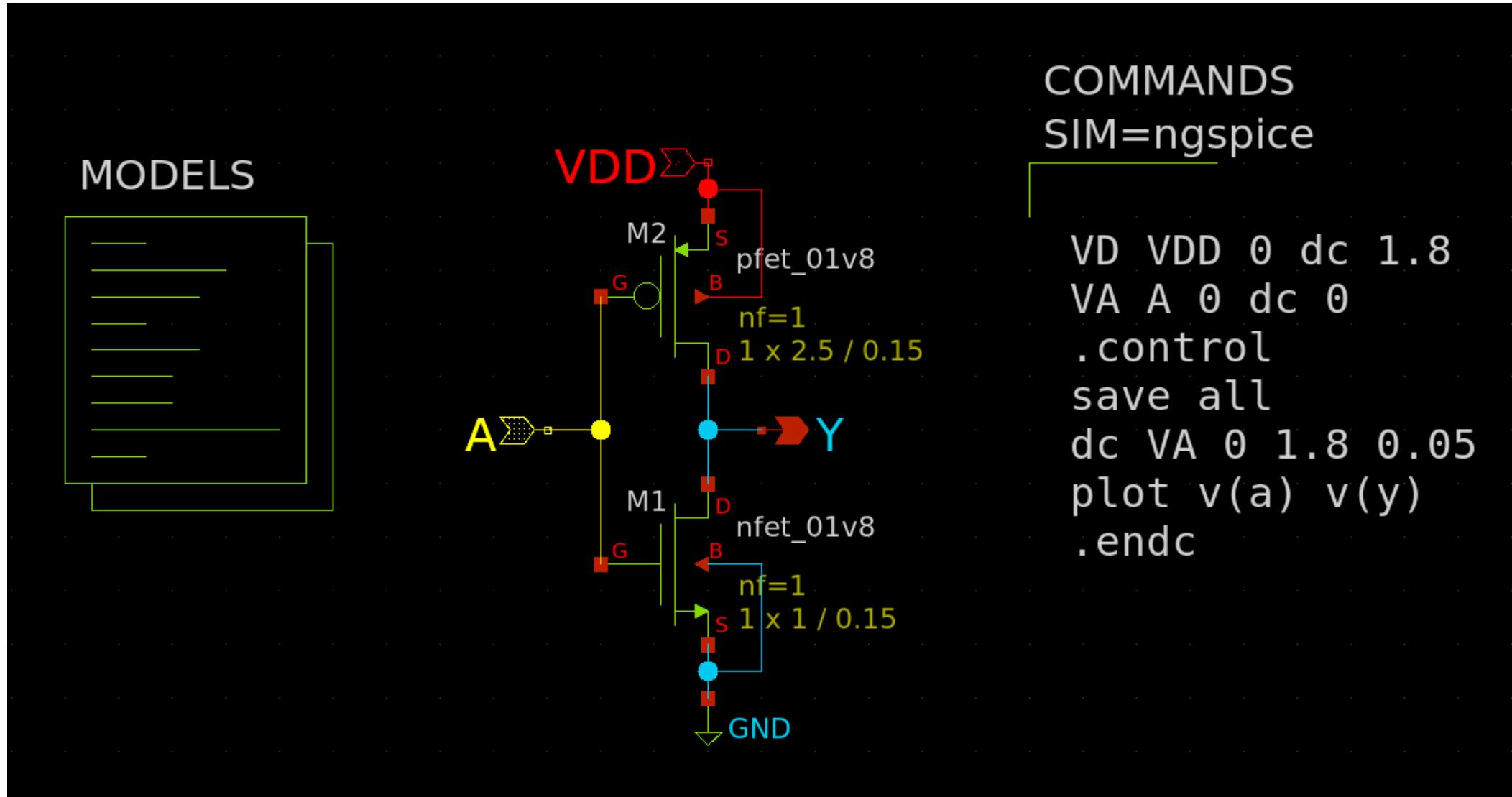
- Xschem, Klayout用PDK **[NEW!]**
https://github.com/ishi-kai/OpenRule1umPDK_SetupEDA
- Xschem用PDK (開発ベータ版)
https://github.com/3zki/OpenRule1umPDK_Xschem

ローカルでのセットアップ方法

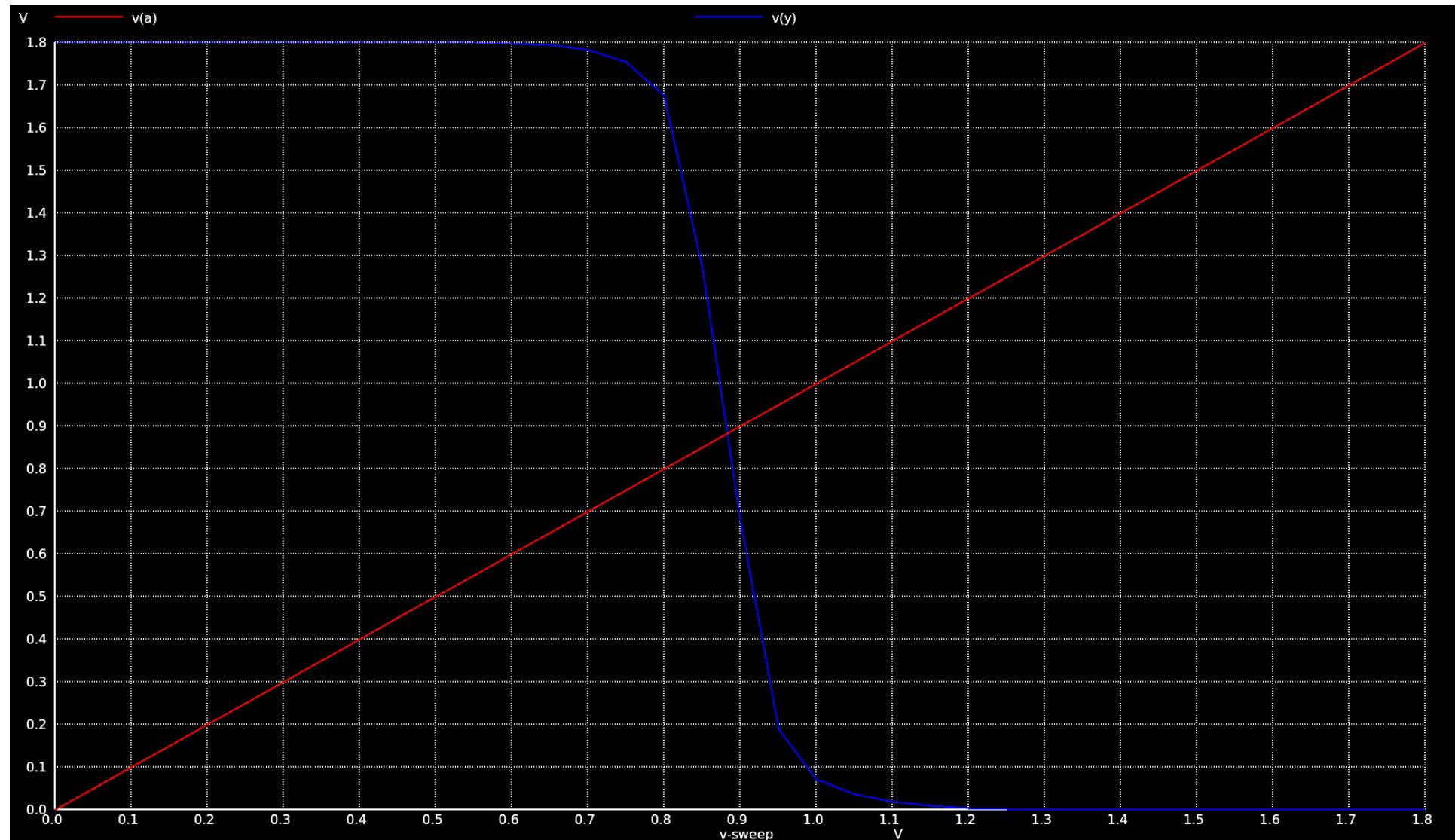
https://github.com/ishi-kai/OpenRule1umPDK_setupEDA

- ここに必要なものが全てが入っています。

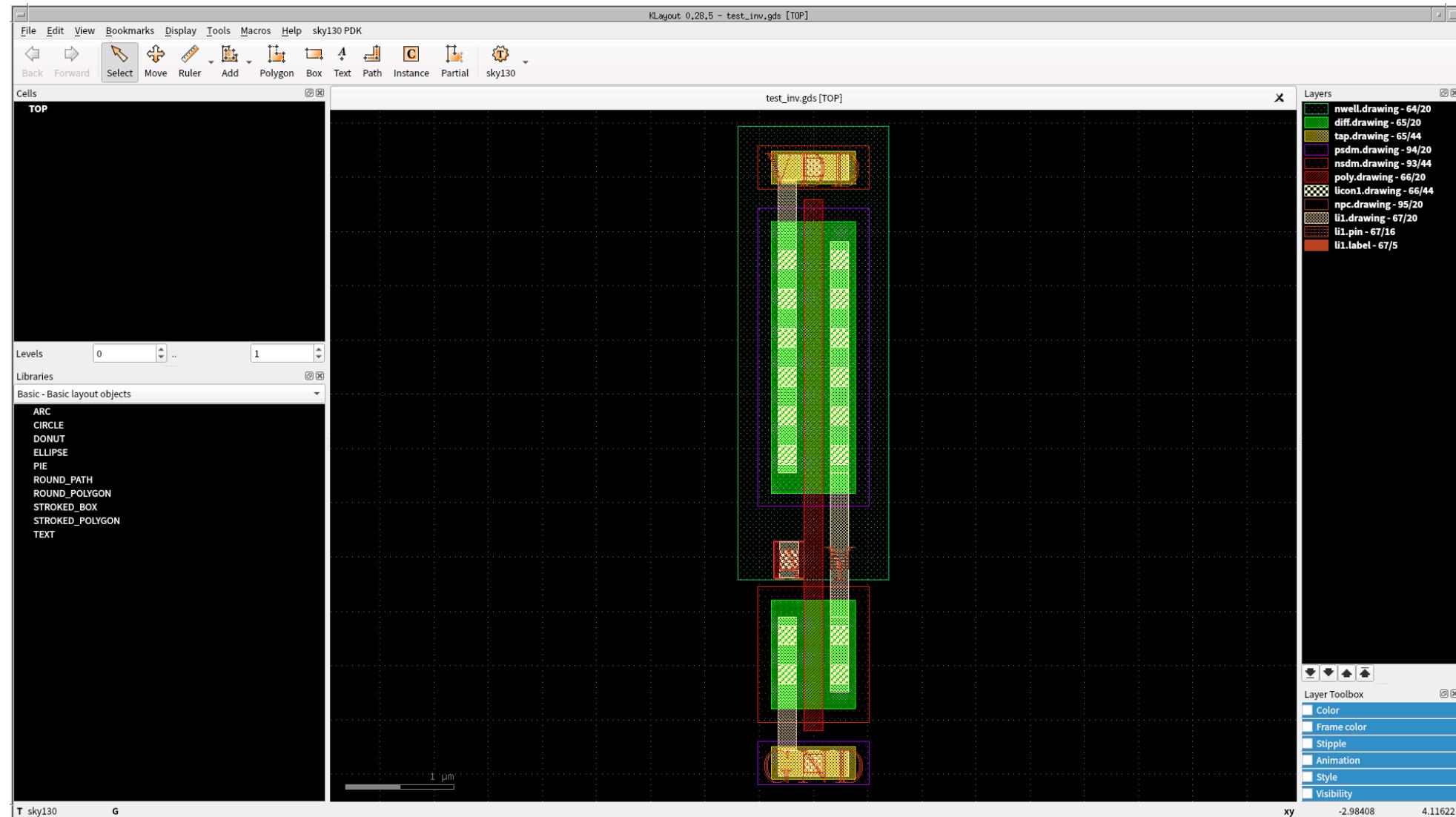
回路図(ベンチマーク)を描いて…



論理検証をして…

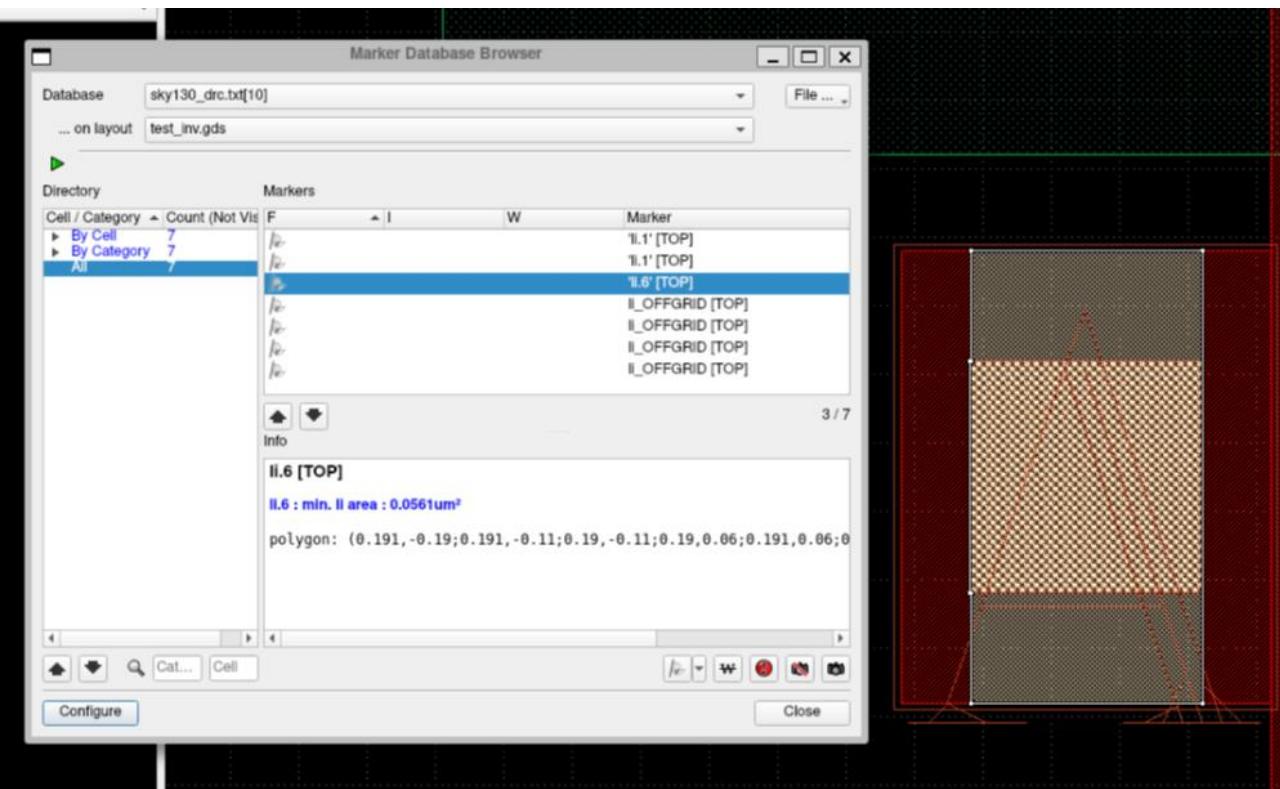


レイアウトを描いて…

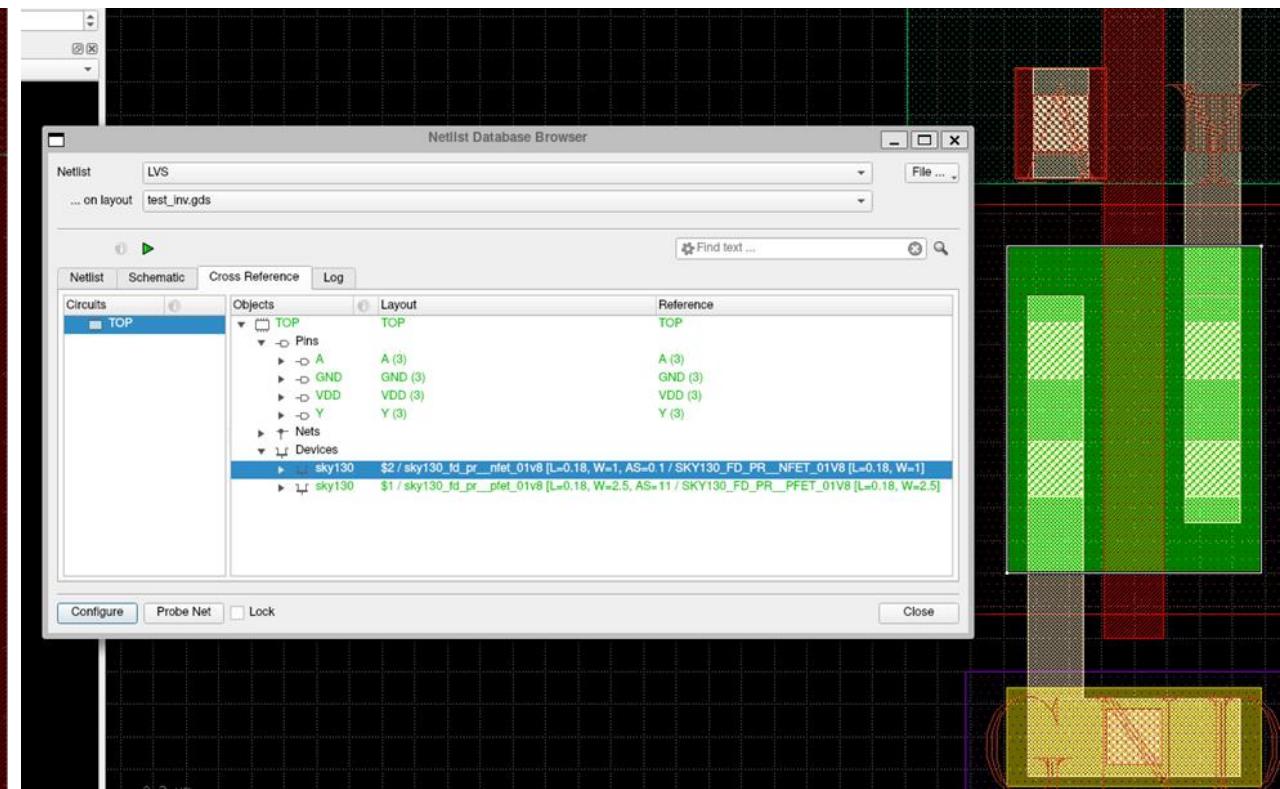


レイアウトを検証して…

DRC



LVS



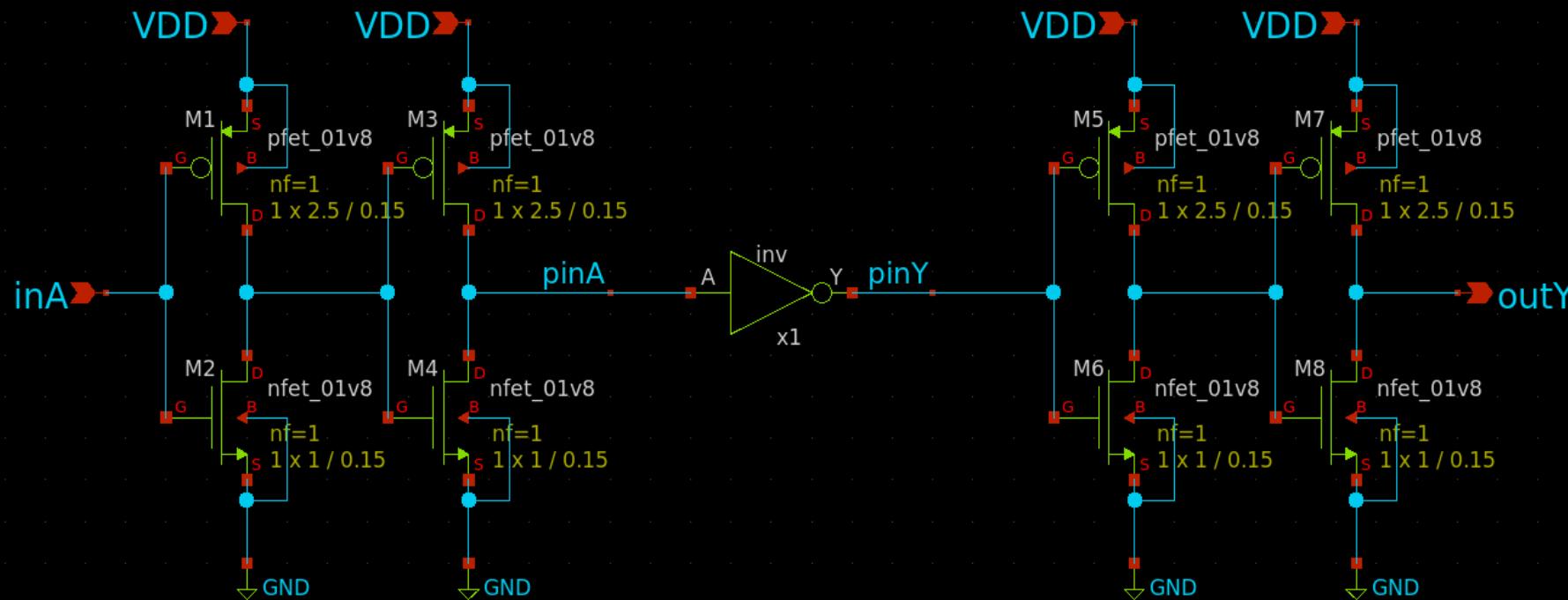
寄生成分を抽出して…

```
 Magic-PEX
File Edit View Search Terminal Help
Loading sky130A Device Generator Menu ...
Loading "/home/user/.klayout/macros/sky130_magic_pex.tcl" from command line.
Warning: Calma reading is not undoable! I hope that's OK.
Library written using GDS-II Release 6.0
Library name: LIB
Reading "TOP".
CIF file read warning: CIF style sky130(): units rescaled by factor of 5 / 1
Extracting TOP into TOP.ext:
exttosim finished.
exttospice finished.
exttospice finished.
* NGSPICE file created from TOP.ext - technology: sky130A

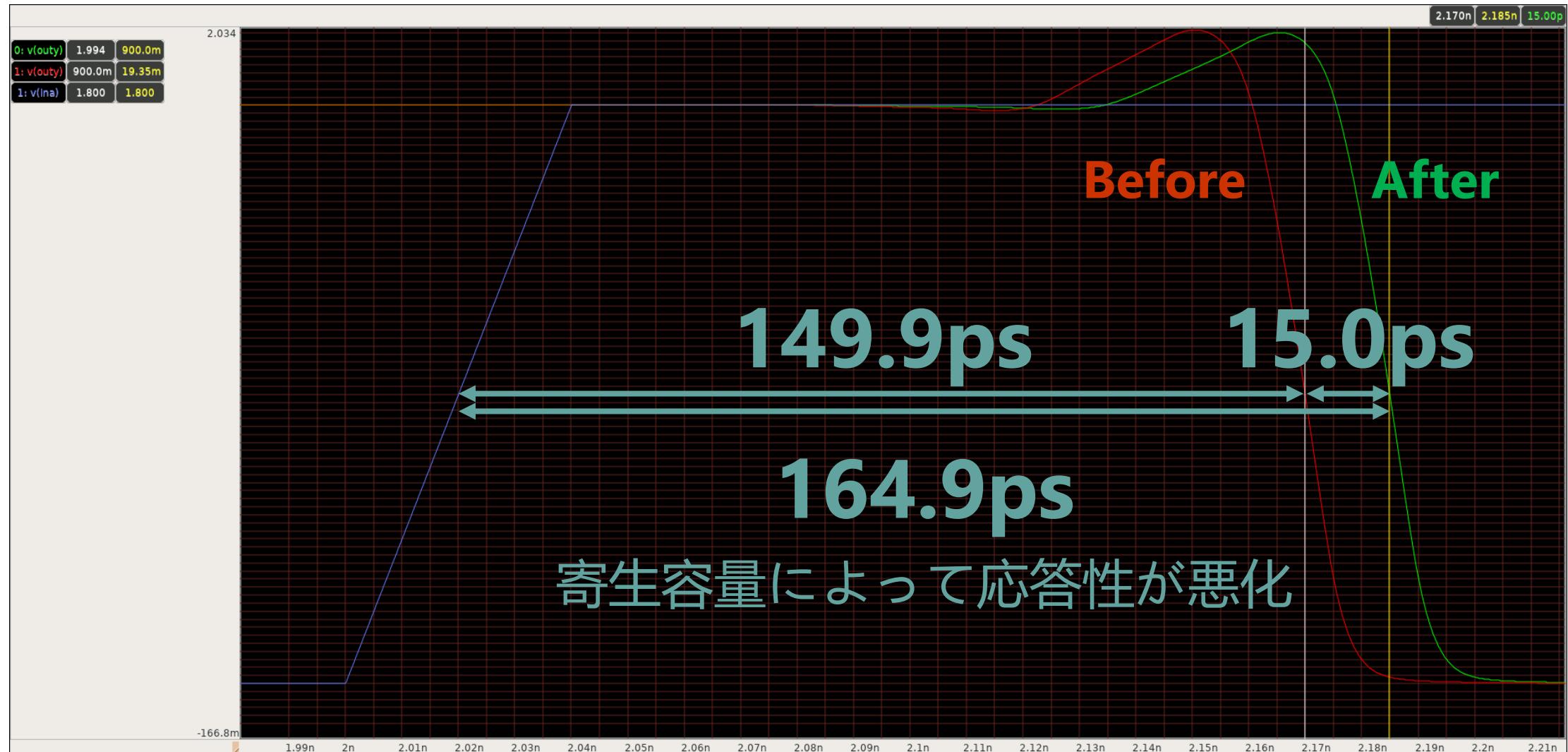
.subckt TOP A Y VDD GND
X0 Y A VDD VDD sky130_fd_pr_pfet_01v8 ad=7.5e+11p pd=5.6e+06u as=7.5e+11p ps=5.
6e+06u w=2.5e+06u l=180000u
X1 Y A GND GND sky130_fd_pr_nfet_01v8 ad=3e+11p pd=2.6e+06u as=3e+11p ps=2.6e+0
6u w=1e+06u l=180000u
C0 A Y 0.05fF
C1 VDD Y 0.17fF
C2 A VDD 0.18fF
.ends
```

ベンチマークを作成して…

```
ngspice
VA inA 0 pulse(0 1.8 0 40p 40p 1n 2n) dc 0
VD VDD 0 dc 1.8
.include ~/TOP_pex_extracted.spice
.control
.tran 1p 4n
.wrdata ~/inv_bench.txt v(ina) v(outy)
.write ~/inv_test_pex.raw
.endc
```



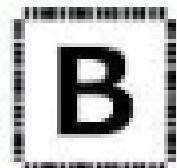
ポストレイアウトシミュレーションをする



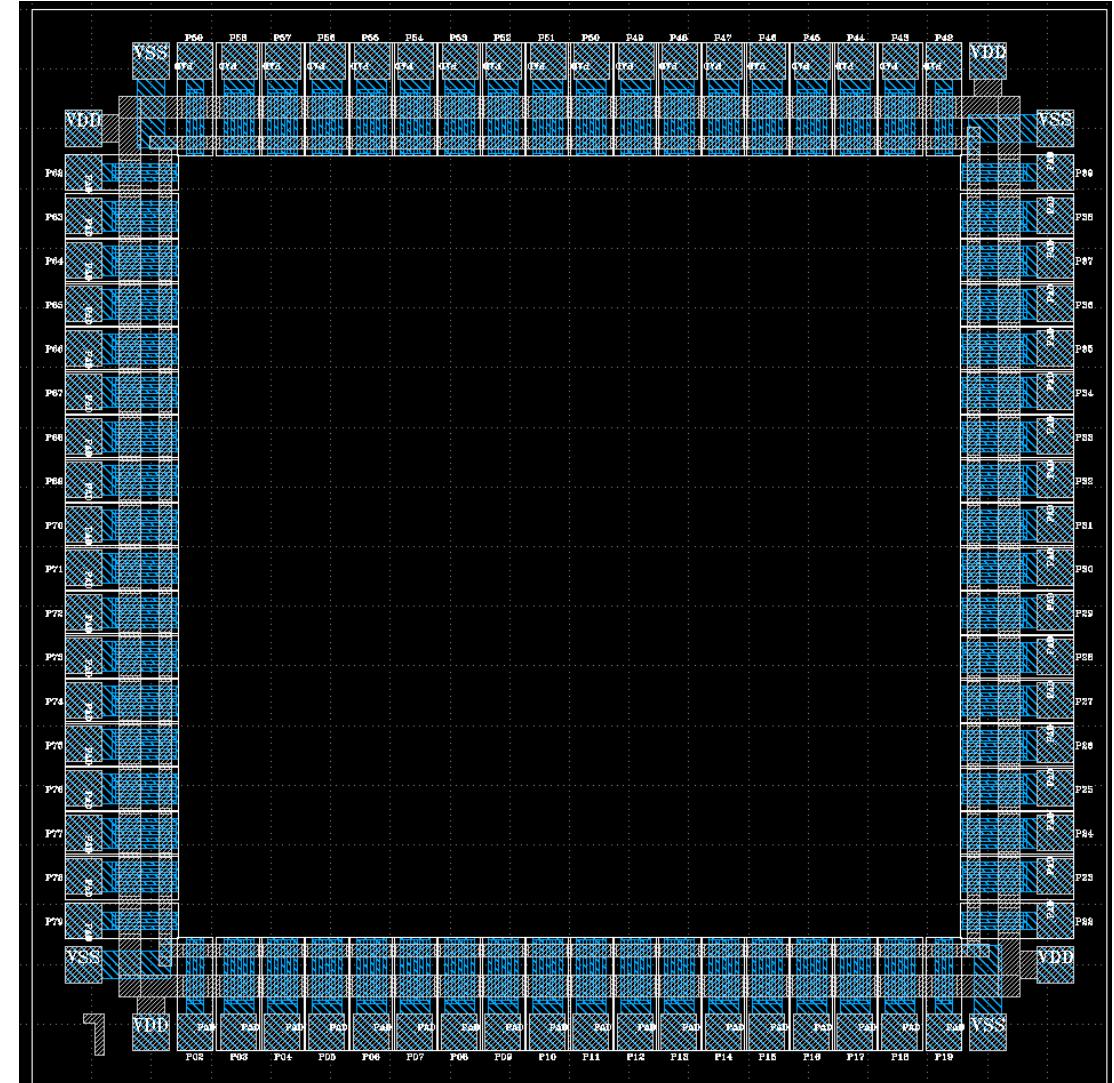
完成したら指定のフレームに回路を載せる（テープアウト対象者のみ）

指定のフレームにレイアウトを載せる

- 80pin(ユーザが使用可能なピンは72pin)
 - 自分が使用するピンについてはピンリストを参照



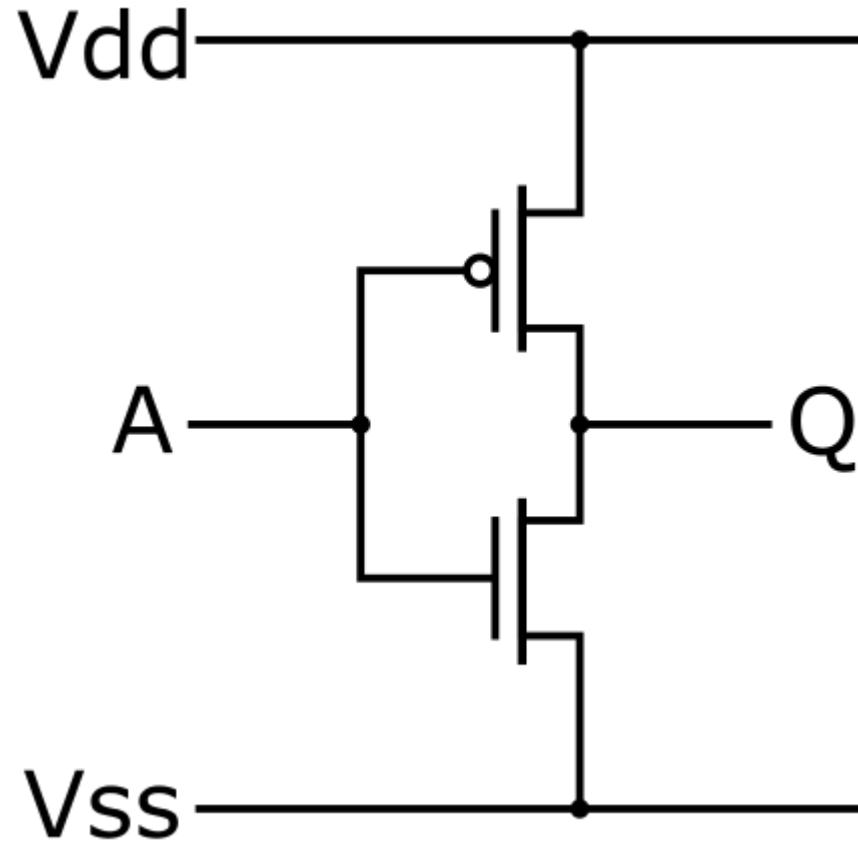
チップサイズ1.8×1.8(mm)
LQFP 80pin 対応モデル
SOPAD(座標固定)



アナログLSI設計デモンストレーション

CMOSインバータ作成

CMOSインバータ



入力A	出力Q
L	H
H	L

↓

入力A	出力Q
Vss	Vdd
Vdd	Vss

アナログLSIの設計フロー

1. 回路図を描く
2. シミュレーションをする
3. 回路図を基にレイアウトを描く
4. レイアウトを検証する
5. レイアウトを基に寄生成分を考慮したシミュレーションをする
6. (フレームに載せる)

寄生成分データが
無いため割愛

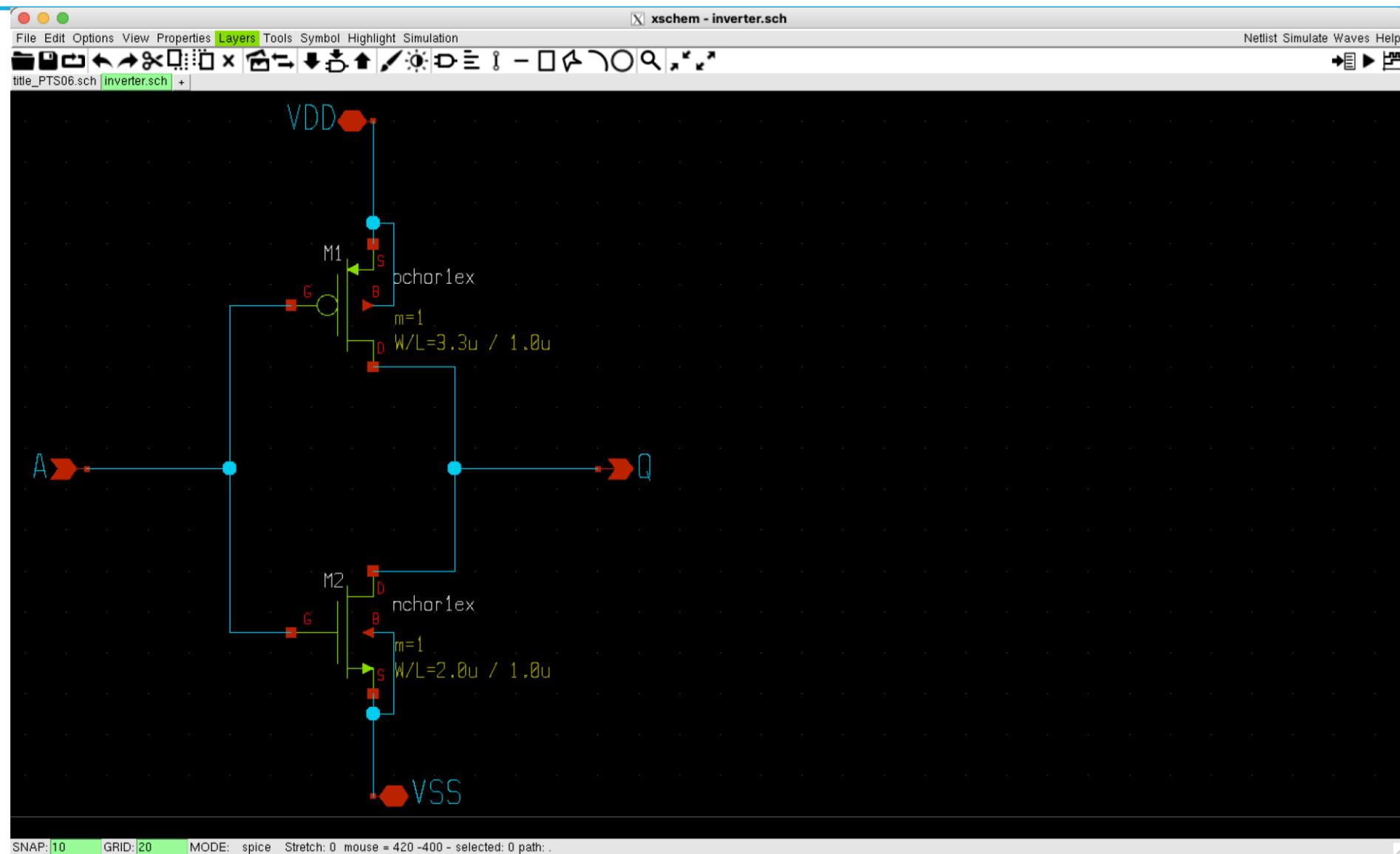
xschem使用上の留意点

- デバイスのシンボルは一部のみ
 - モデル名を変更する必要がある場合もある（P-FET,N-FETでは不要）

論理検証

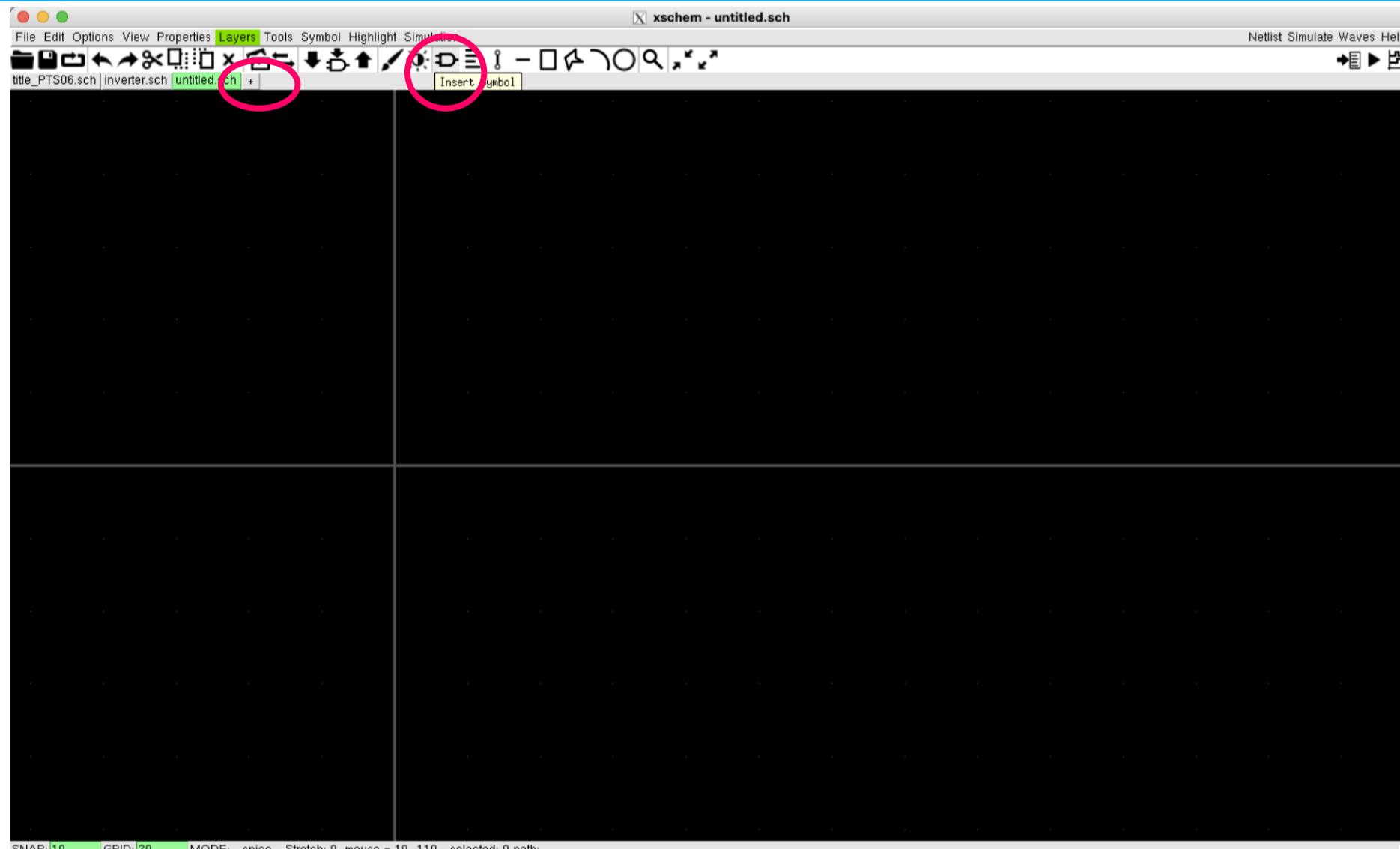
- ・ インバータでは入力に対して反転した出力が確認できれば良い
 - ・ 入力 0 V → 出力 Vdd V, 入力 Vdd V → 出力 0 V
 - ・ DC解析でCMOSインバータの動作を見てみる
- ・ DC解析 電圧を変動させたときの定常応答
- ・ tran解析 時間を変動させたときの過渡応答
- ・ AC解析 周波数を変動させたときの定常応答

CMOSインバーター回路の例

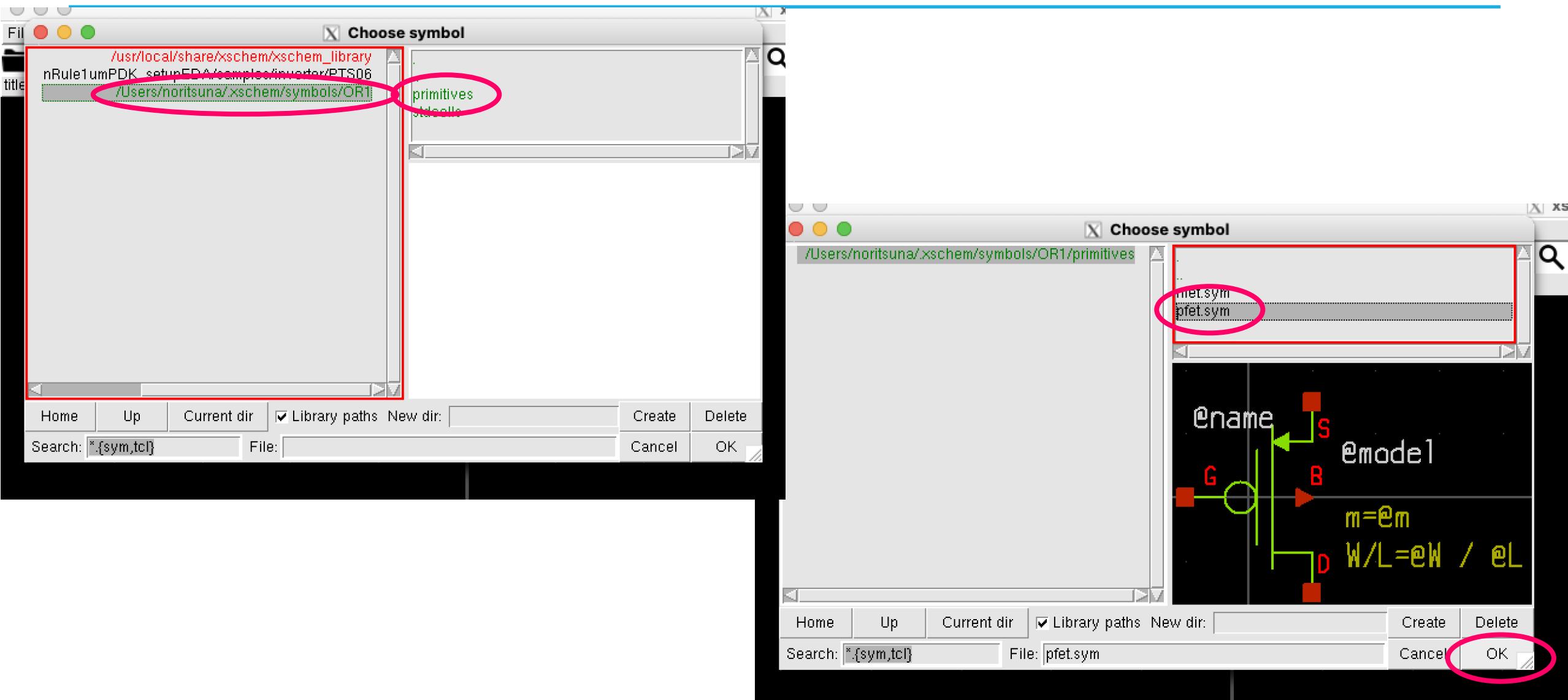


SNAP: 10 GRID: 20 MODE: spice Stretch: 0 mouse = 420 -400 - selected: 0 path: .

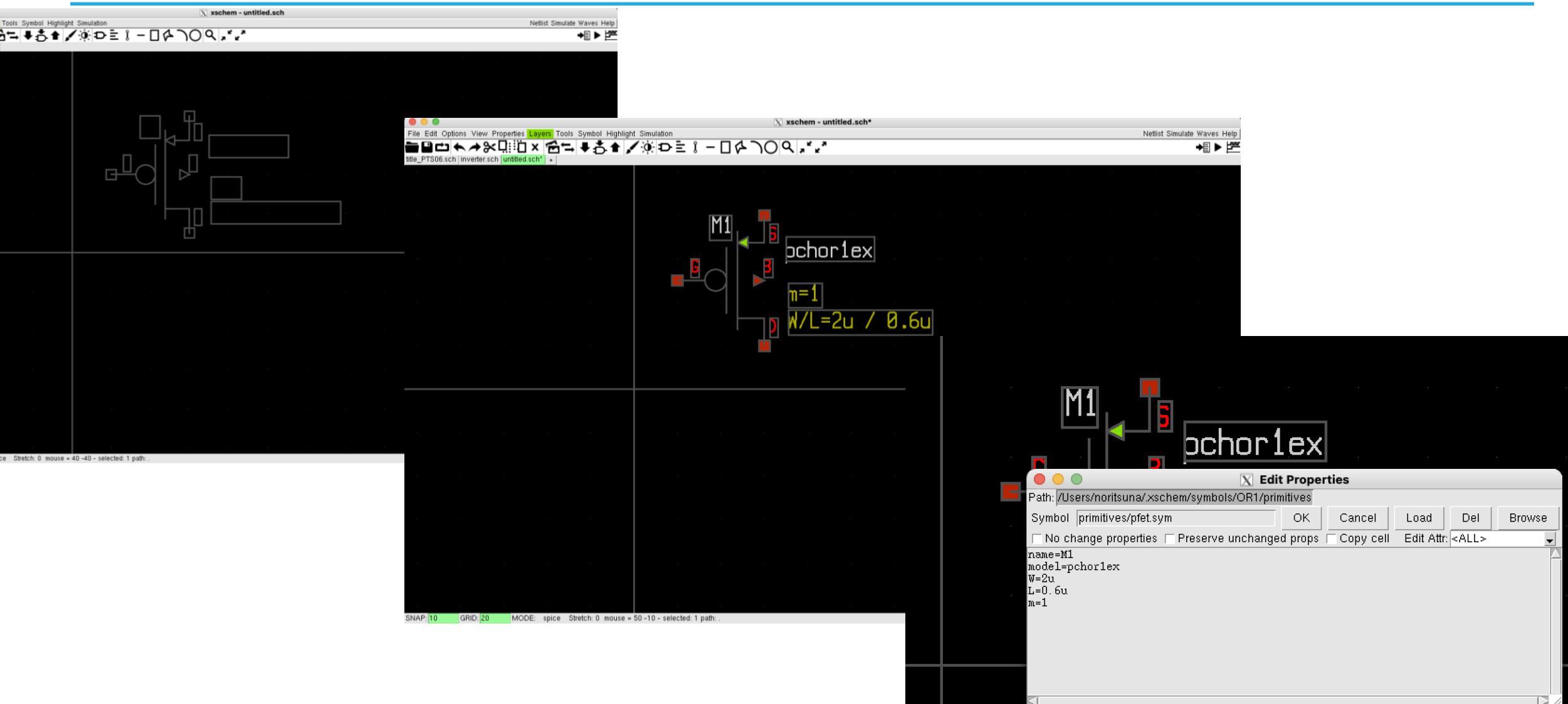
xschem使い方：新規回路図とMODULEウィンドウ



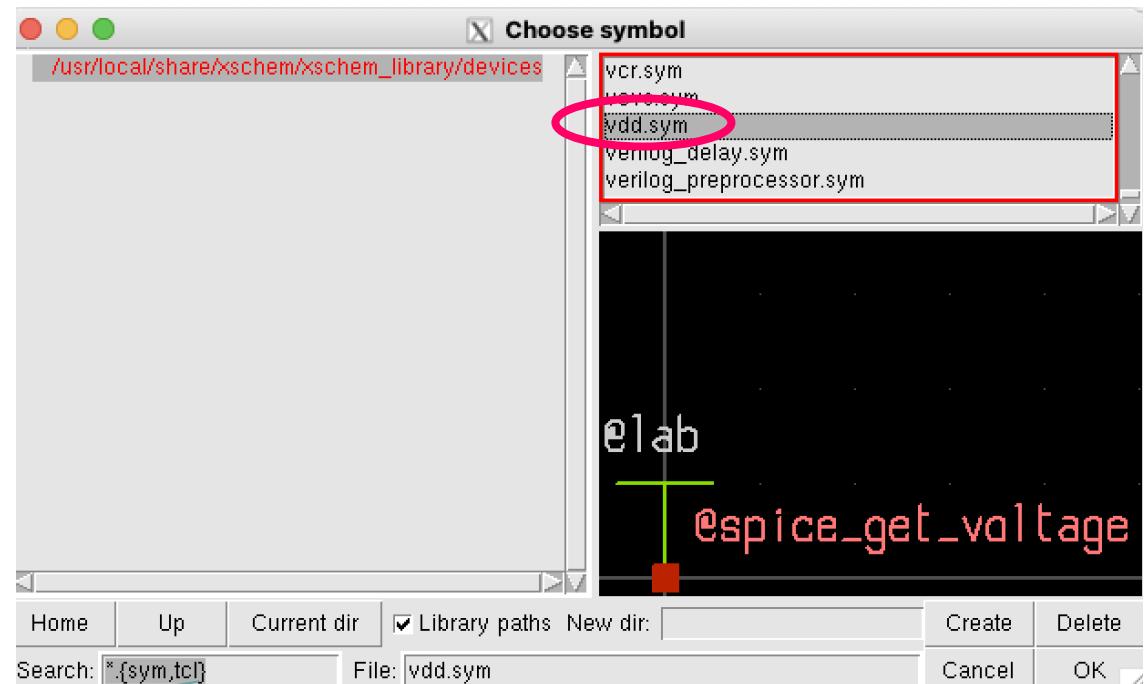
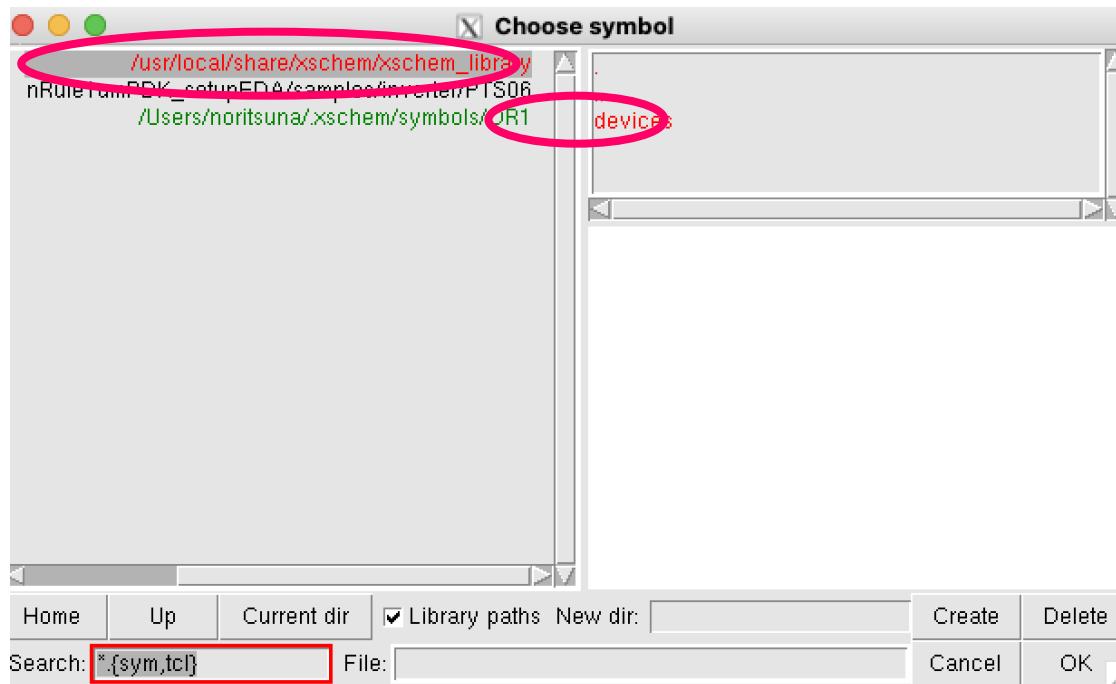
xschem使い方： OR1のMODULEの選択：P-FET



xschem使い方： OR1のMODULEの配置とプロパティ

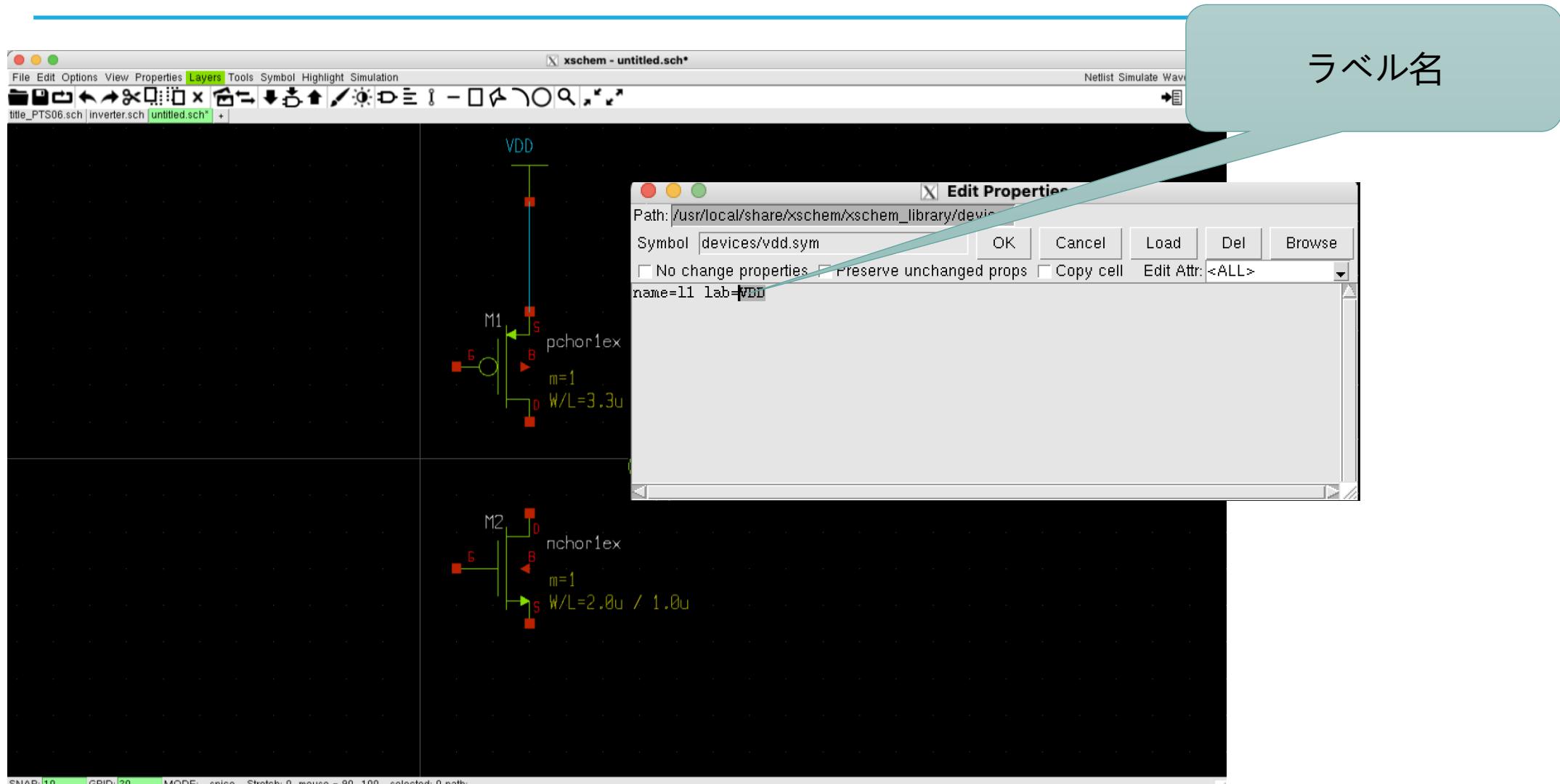


xschem使い方：標準のMODULEの選択：VDD

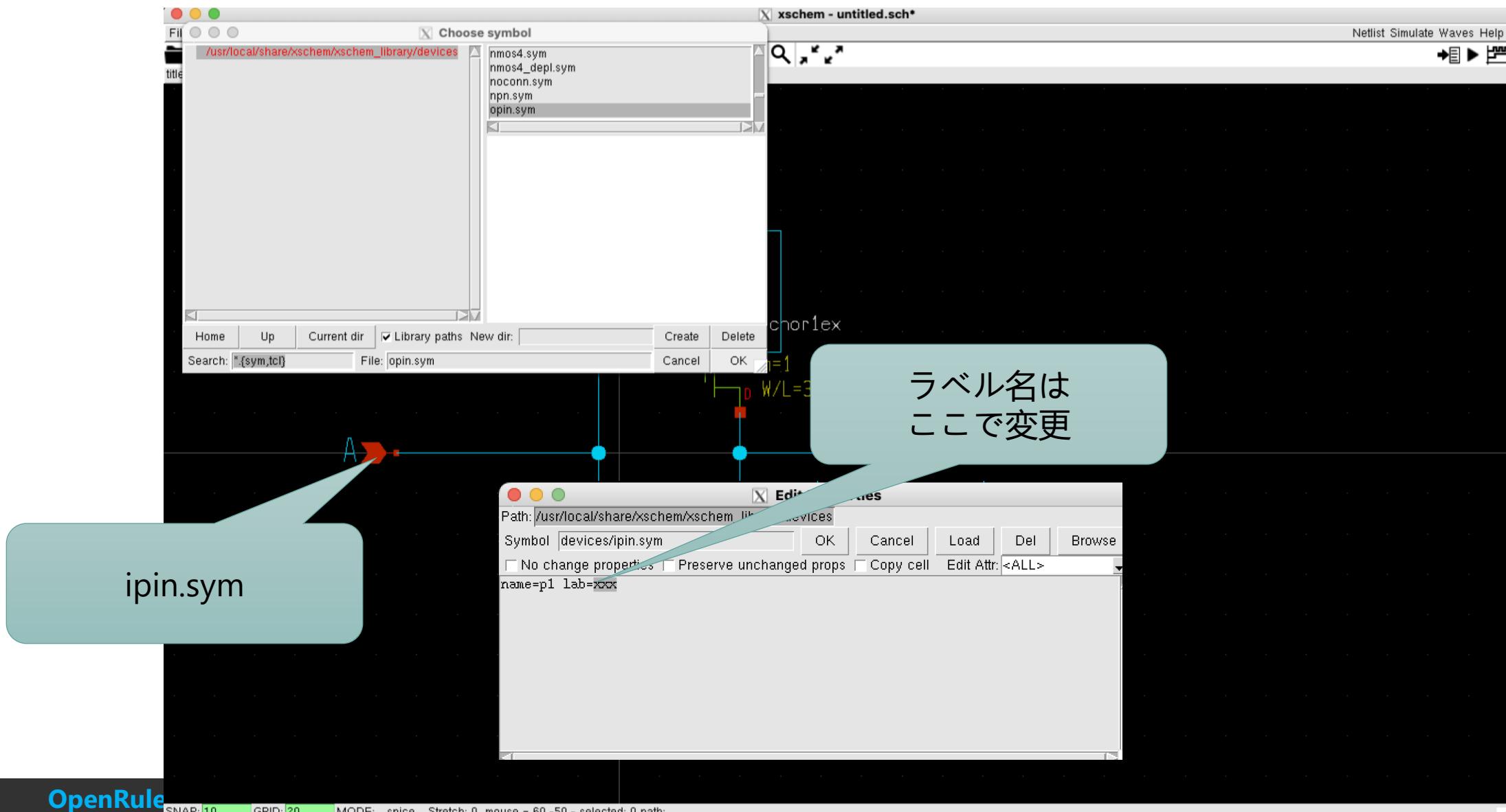


検索も可能

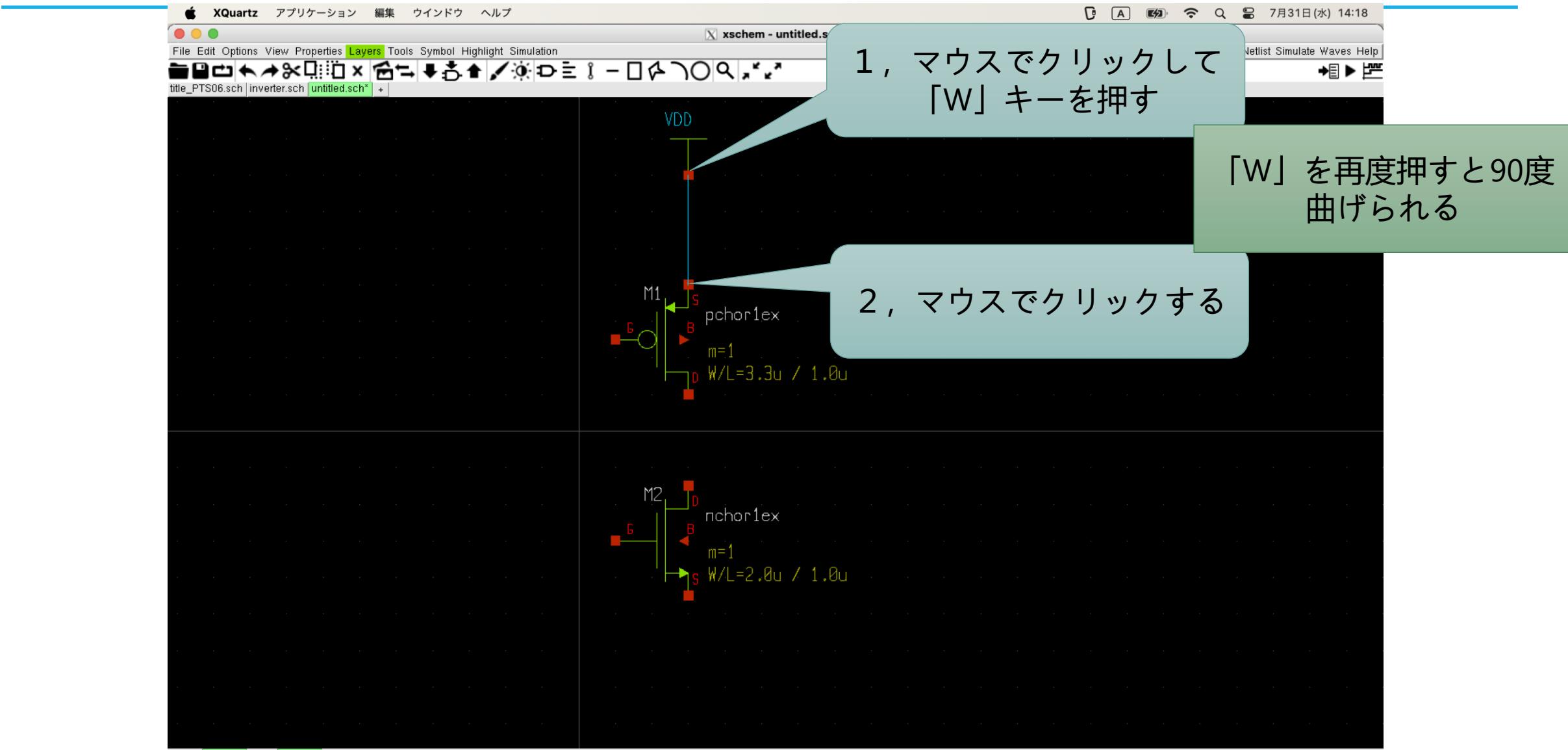
xschem使い方：標準のMODULEの配置とプロパティ：VDD



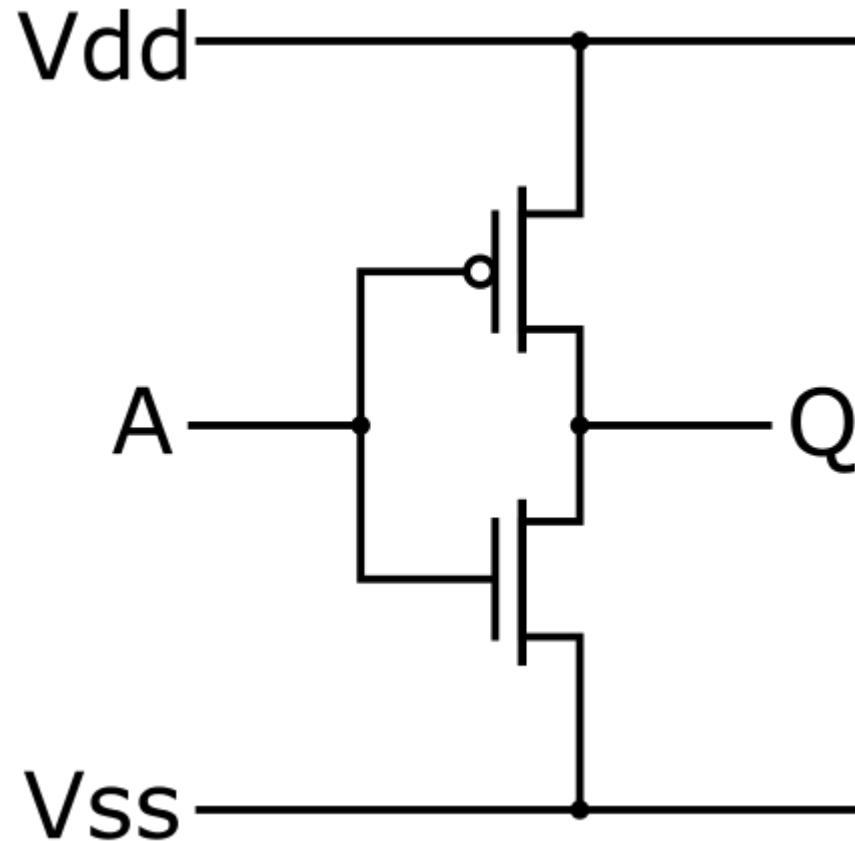
xschem使い方：標準のMODULEの選択：ipin, opin, iopin



xschem使い方： wireの引き方



ハンズオン：CMOSインバータを作る

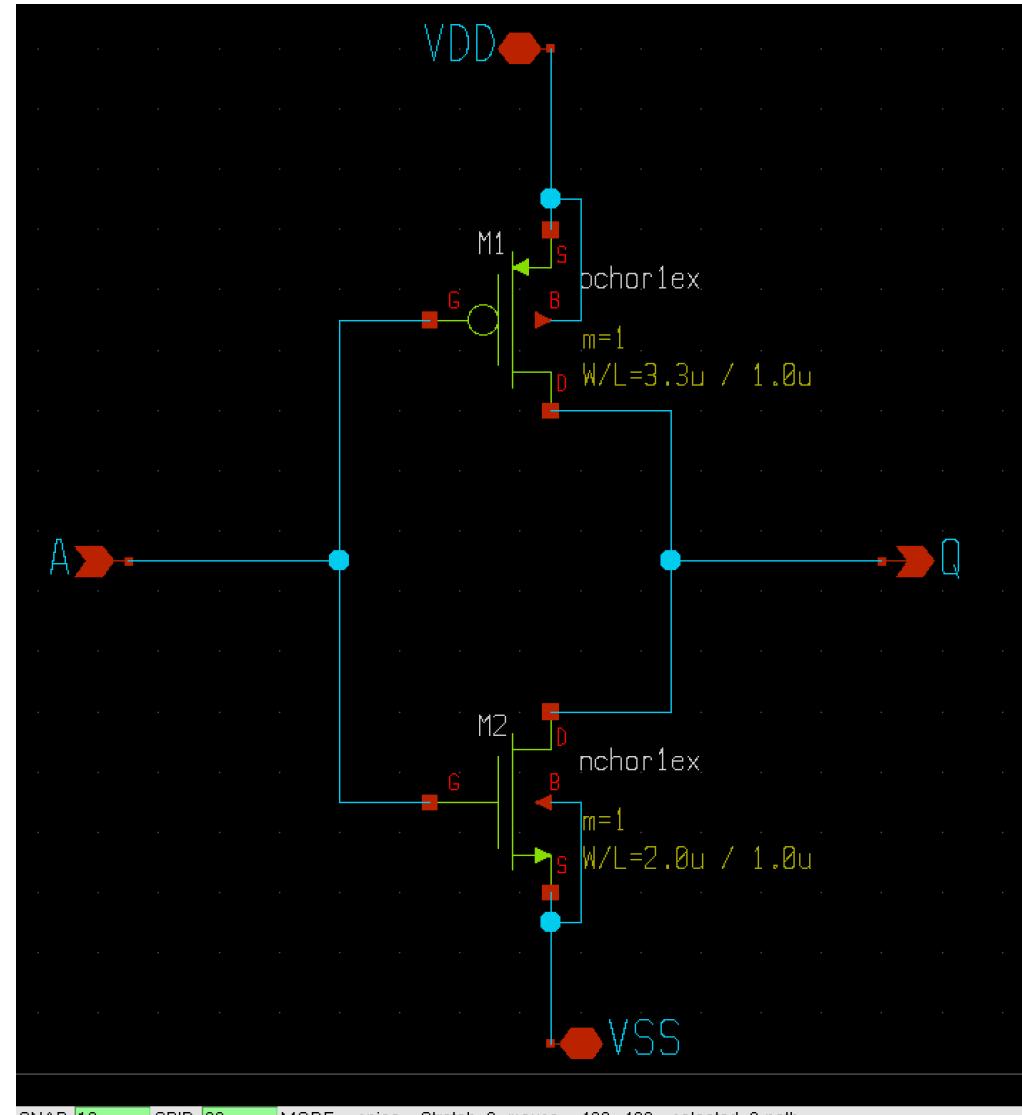


入力A	出力Q
L	H
H	L

↓

入力A	出力Q
V_{ss}	V_{dd}
V_{dd}	V_{ss}

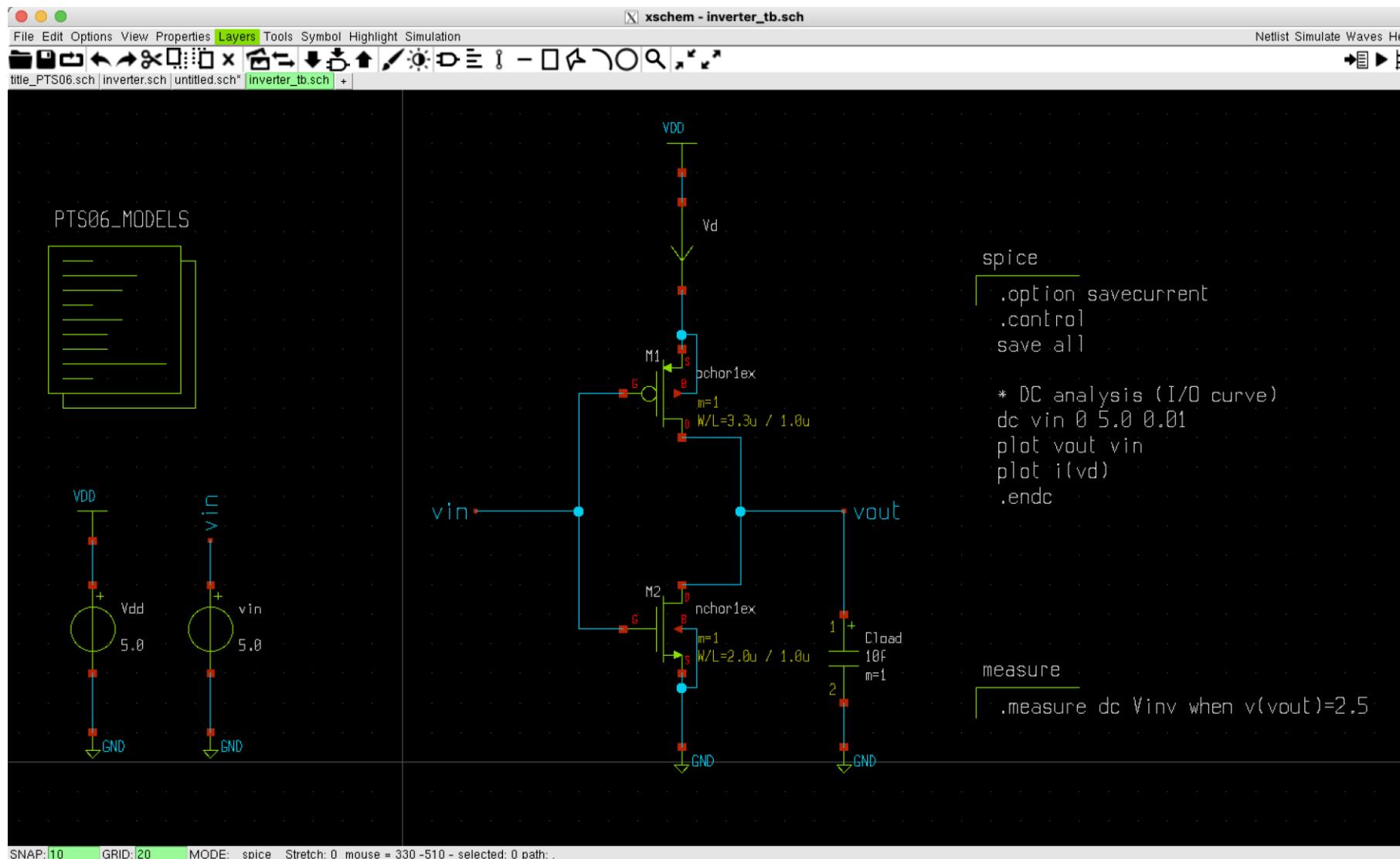
ハンズオン：CMOSインバータのサンプル



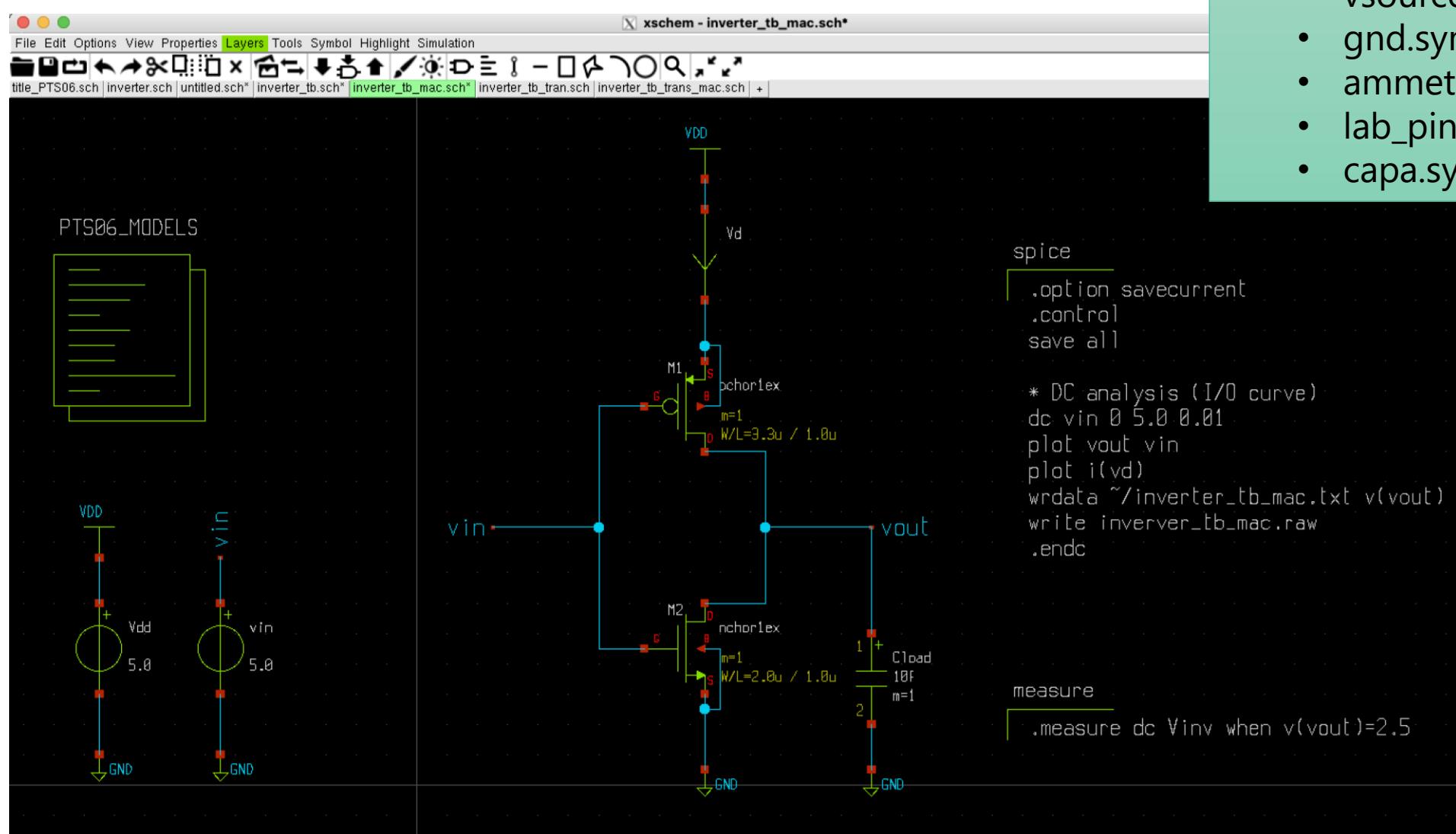
アナログLSIの設計フロー

1. 回路図を描く
- 2. シミュレーションをする**
3. 回路図を基にレイアウトを描く
4. レイアウトを検証する
5. レイアウトを基に寄生成分を考慮したシミュレーションをする
6. (フレームに載せる)

xschem : ベンチマーク例



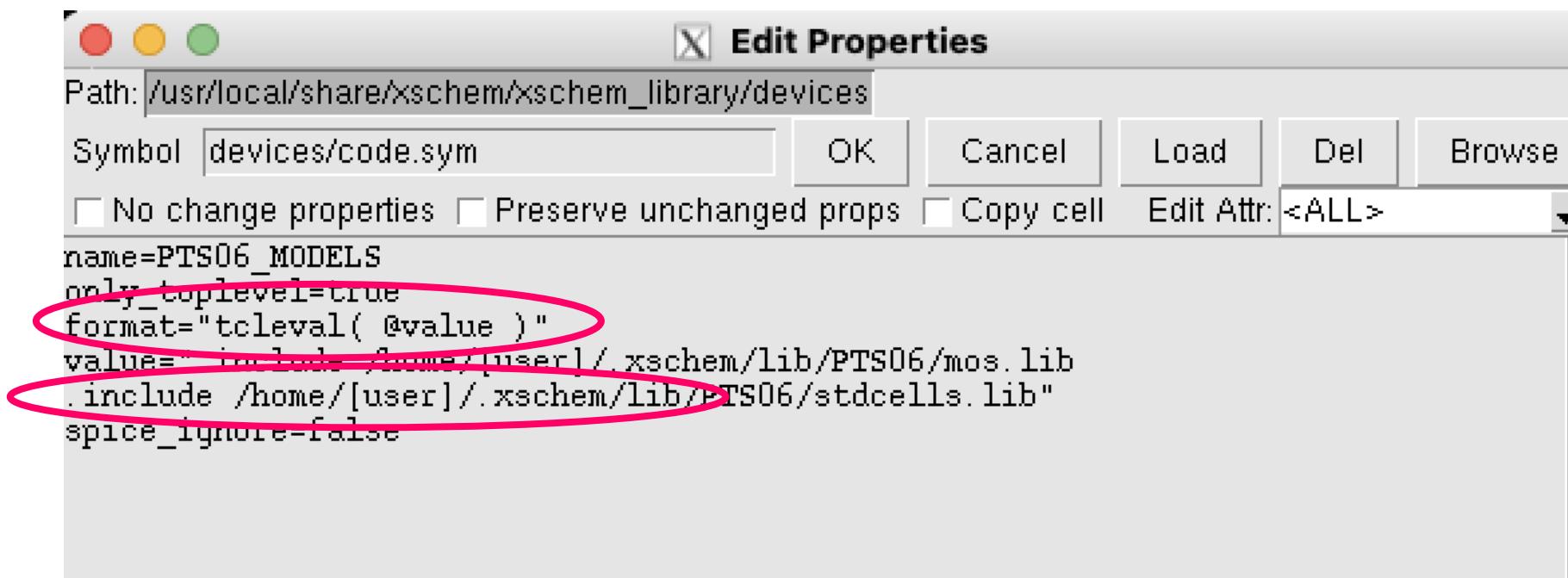
xschem : ベンチマーク



- キーワード
 - code.sym
 - code_shown.sym
 - vsource.sym
 - gnd.sym
 - ammeter.sym
 - lab_pin.sym
 - capa.sym

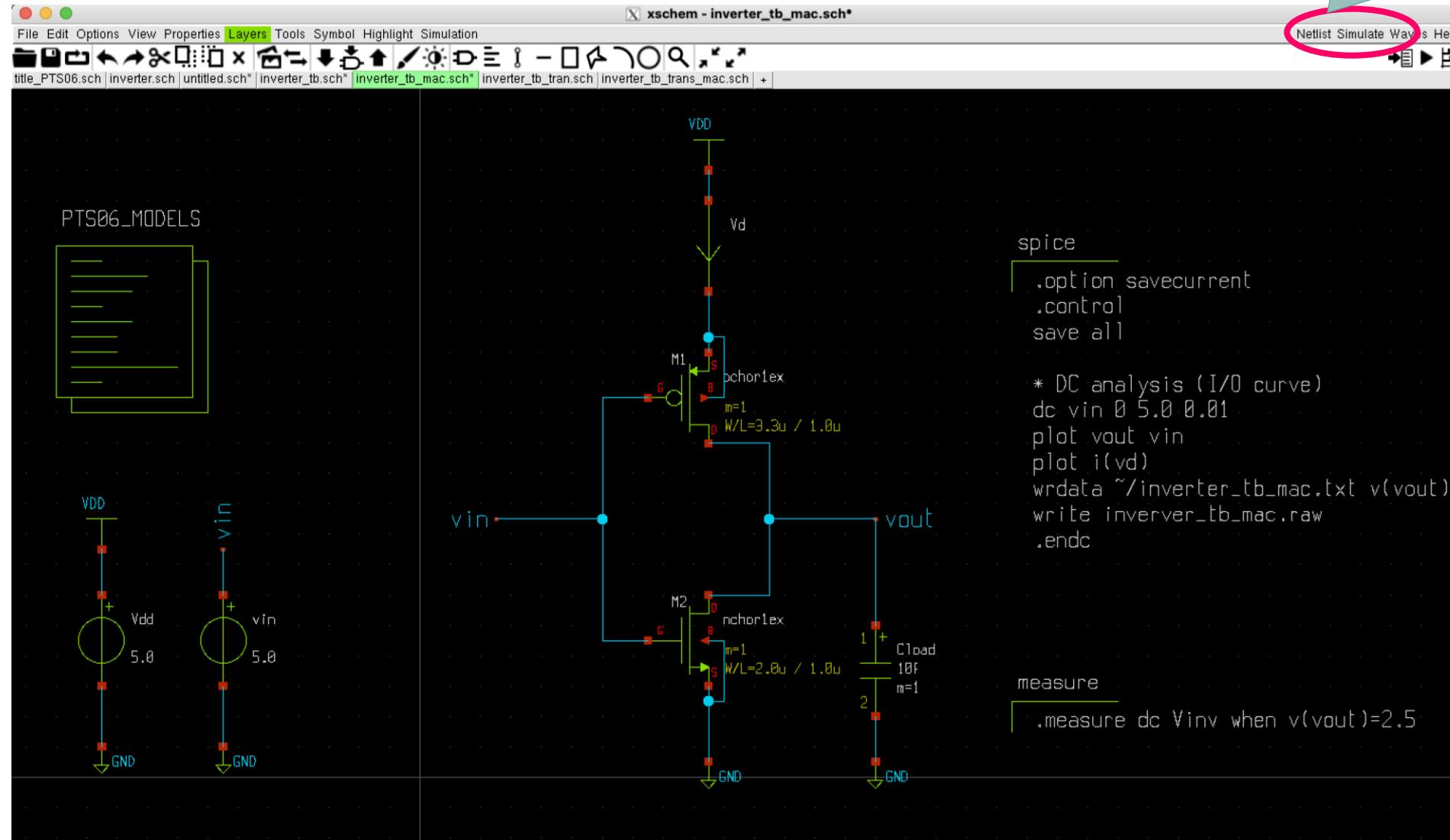
xschem : PTS06_MODELSを記述する際の注意点

- code.symを使用する際、以下の行を追記する必要がある
 - format="tcleval(@value)"
 - includeは絶対パス（フルパス）である必要がある。相対パスは使えない。

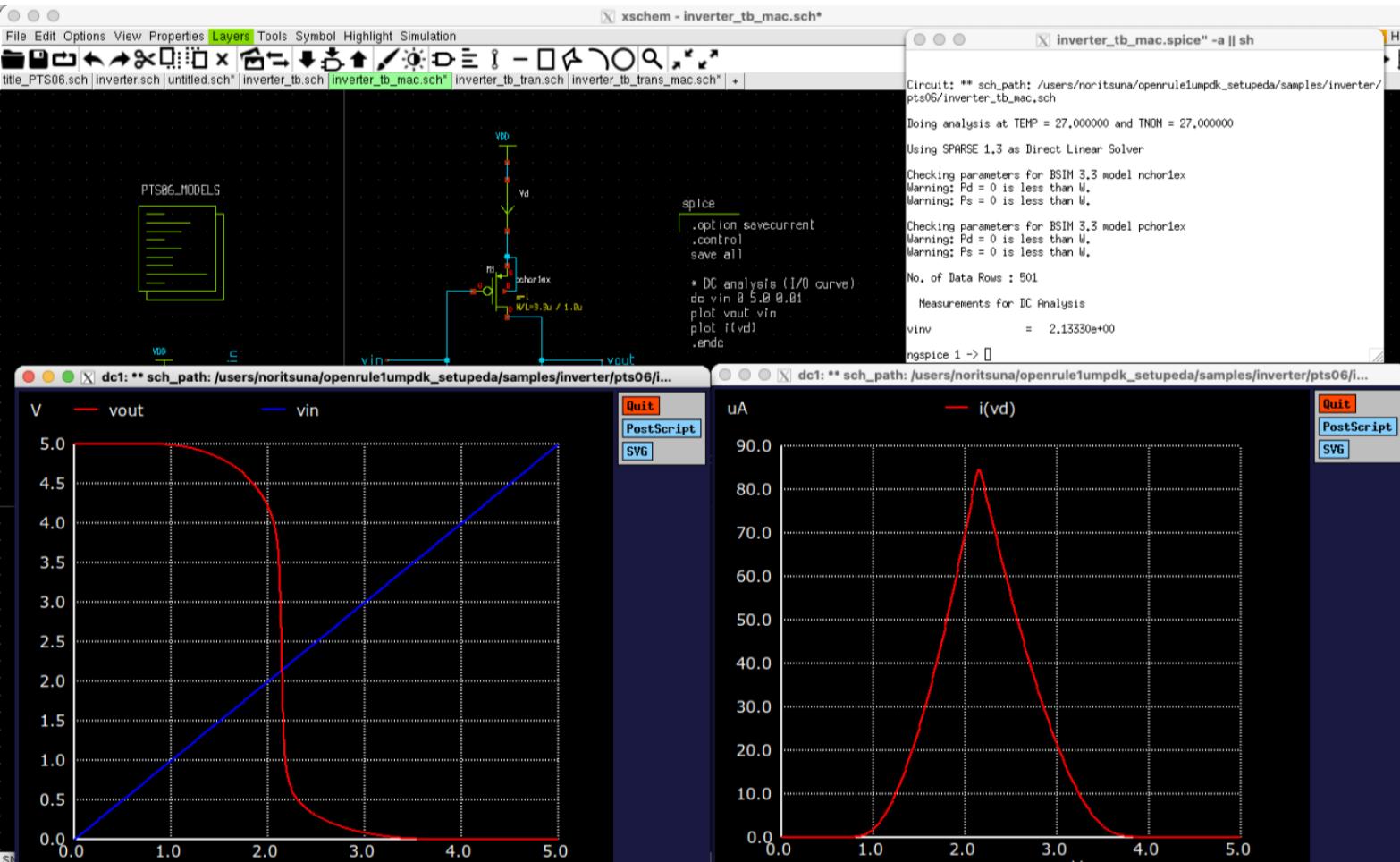


ngspice : シミュレーションの実行

netlist→simulateの
順にクリック



ngspice : シミュレーション結果



入力A = 0 V → 出力Q = 5.0 V

入力A = 5.0 V → 出力Q = 0 V

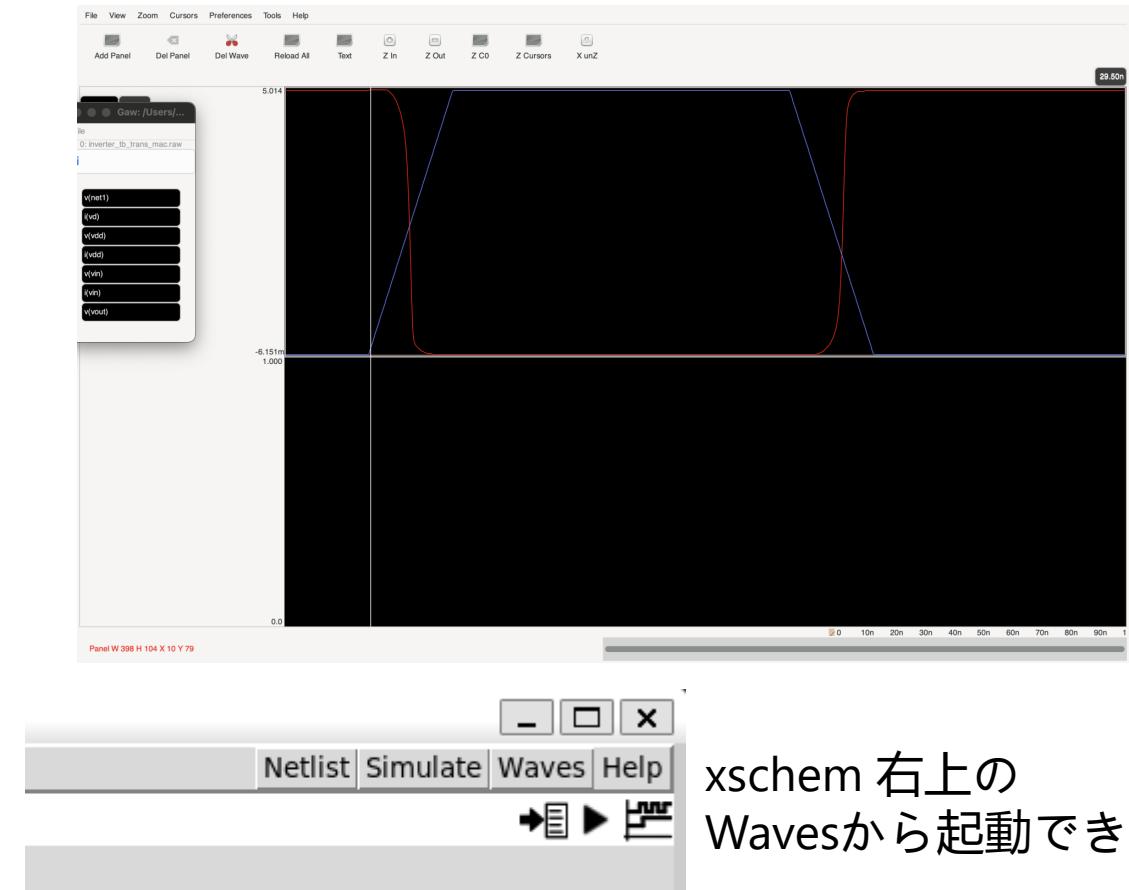
- インバータとして機能している

ngspice : wrdata と gawによる波形表示

wrdataを使用するとテキスト出力ができる

```
0.00000000e+00 3.2999999e+00
5.0000000e-02 3.2999999e+00
1.0000000e-01 3.2999997e+00
1.5000000e-01 3.2999990e+00
2.0000000e-01 3.29999960e+00
2.5000000e-01 3.29999842e+00
3.0000000e-01 3.29999375e+00
3.5000000e-01 3.29997564e+00
4.0000000e-01 3.29990878e+00
4.5000000e-01 3.29968137e+00
5.0000000e-01 3.29900073e+00
5.5000000e-01 3.29728006e+00
.
.
.
.
```

writeでrawファイルを出力すると gaw で波形表示ができる

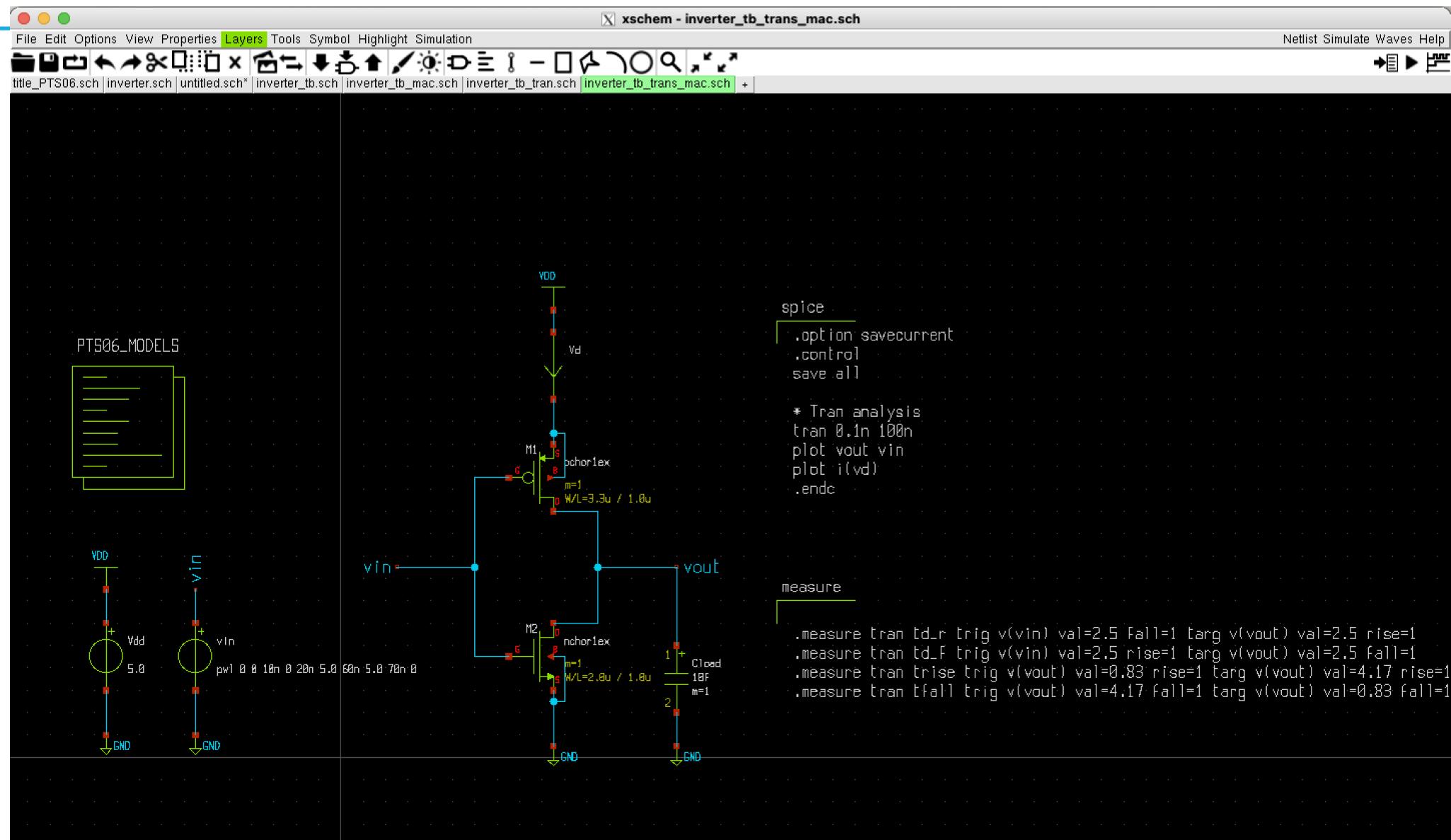


xschem 右上の
Wavesから起動できる

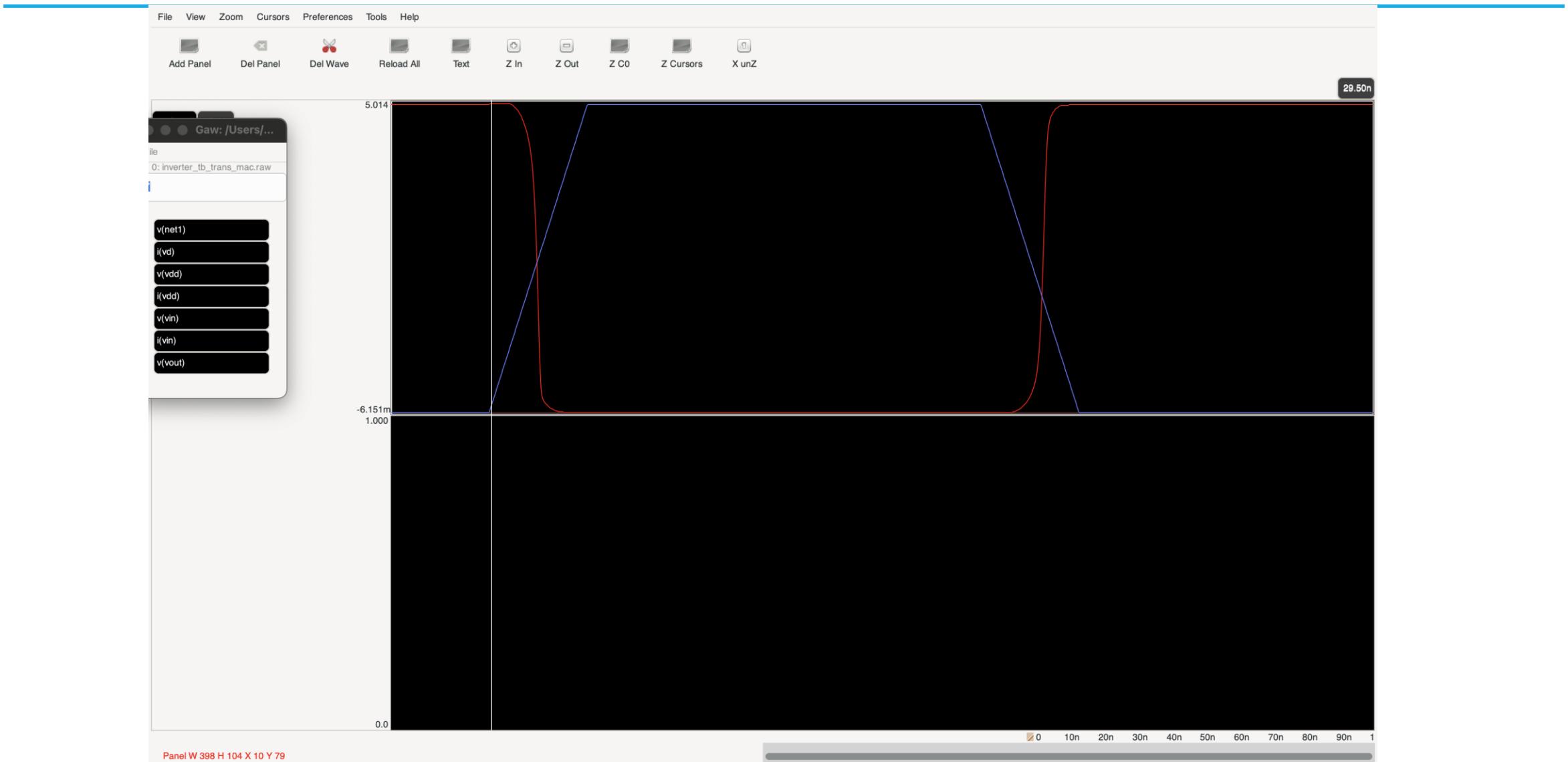
ngspice : 過渡応答、遅延時間

- 過渡応答から遅延時間を見たい場合はtran解析を用いる
- 出力段に負荷を設定しないと正しい過渡応答が見れない
今回は入力段・出力段共に何も接続しない状態を見る

ngspice : 過渡解析ベンチマーク例



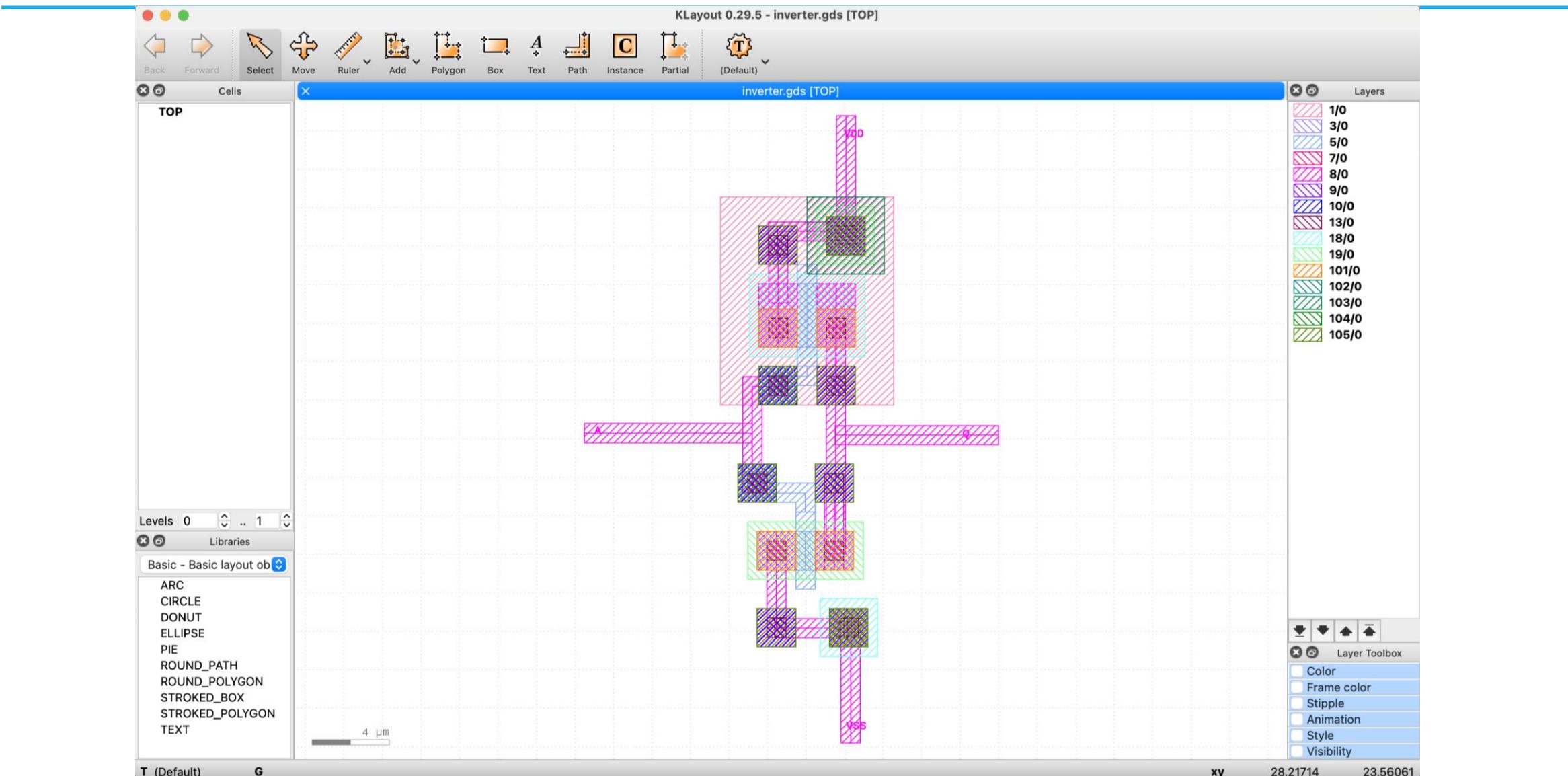
ngspice : 過渡解析シミュレーション結果



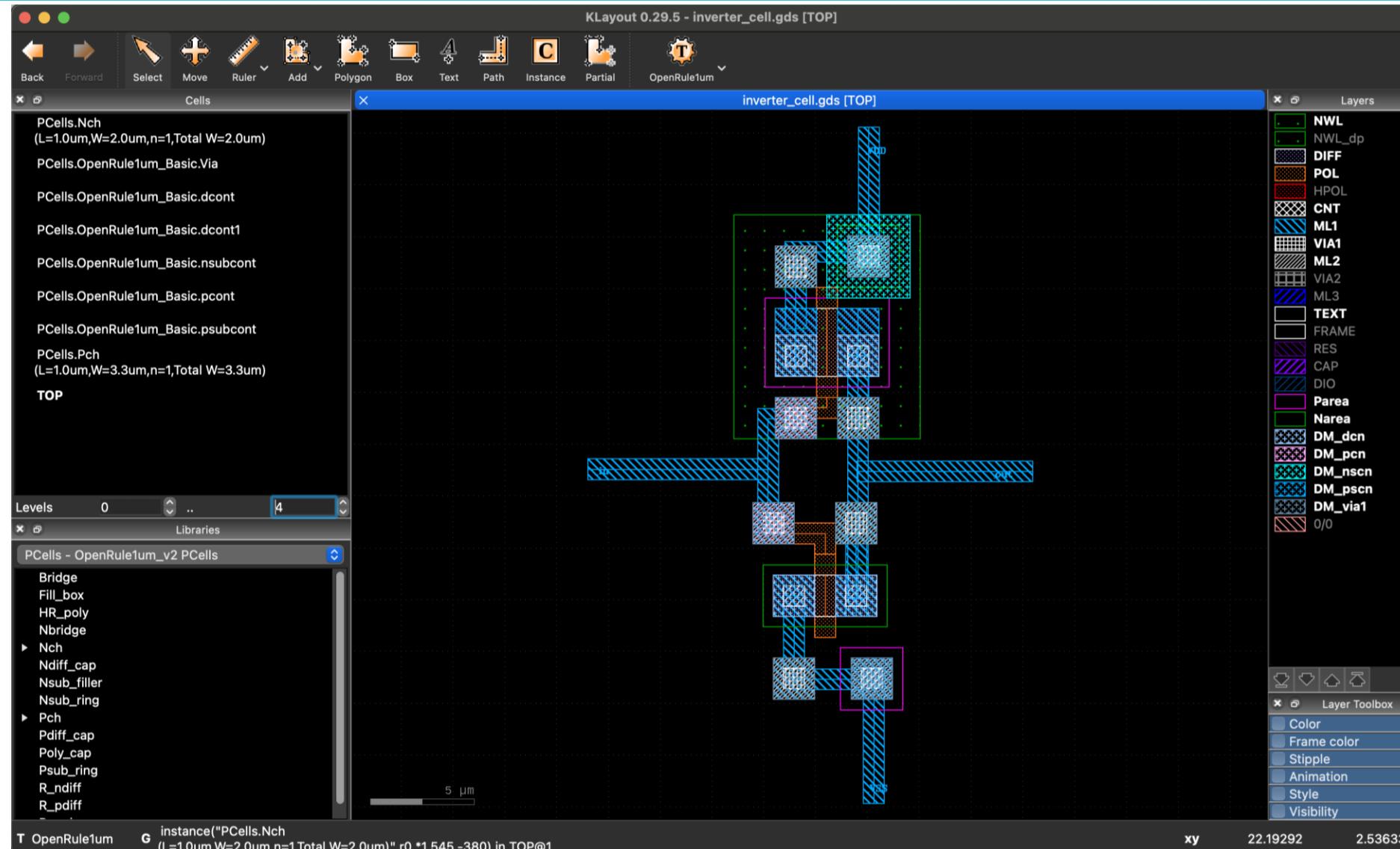
アナログLSIの設計フロー

1. 回路図を描く
2. シミュレーションをする
- 3. 回路図を基にレイアウトを描く**
4. レイアウトを検証する
5. レイアウトを基に寄生成分を考慮したシミュレーションをする
6. (フレームに載せる)

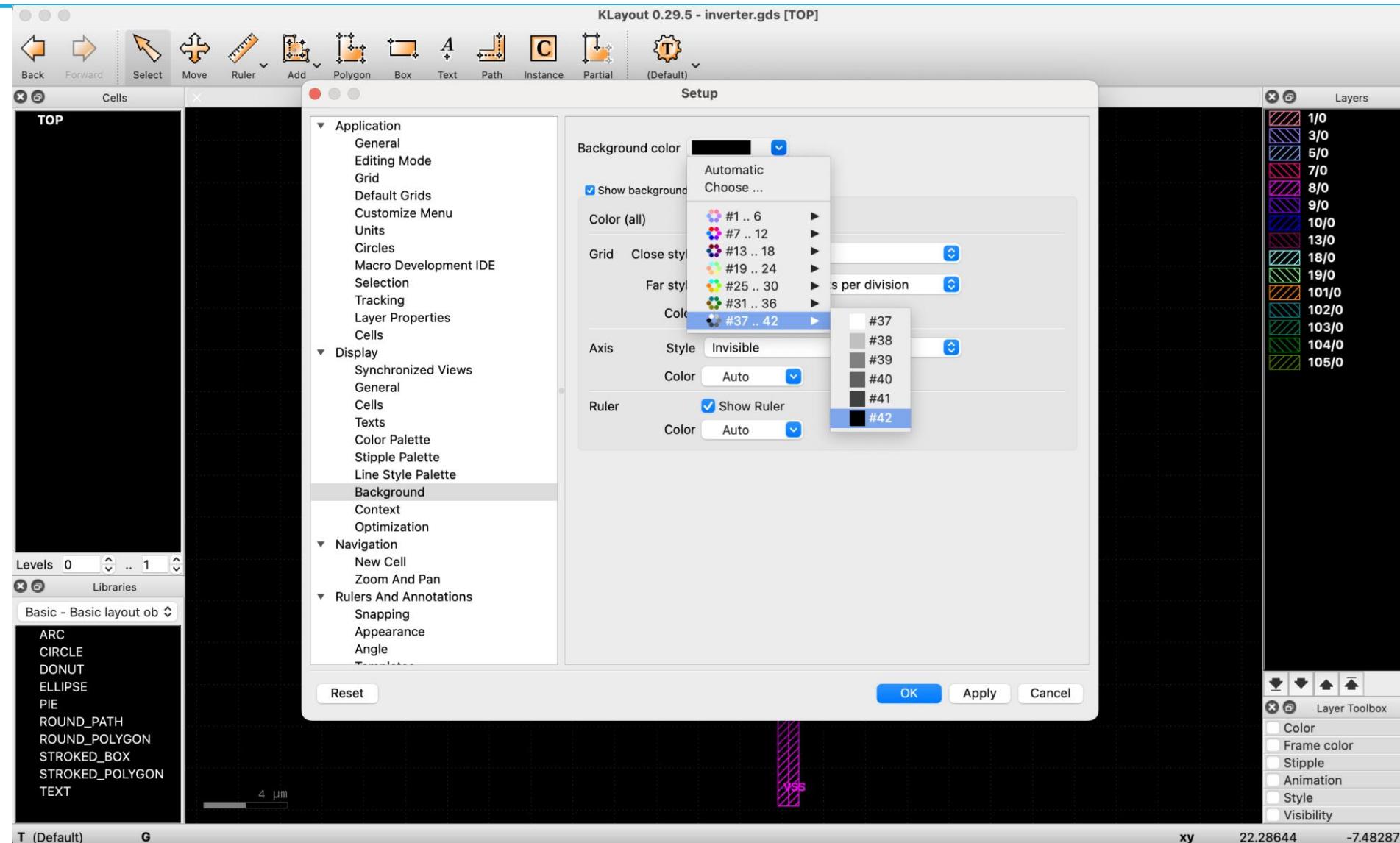
KLayout : 公式PDKのレイアウト画面



KLayout : バックグラウンドが「黒」のレイアウト画面

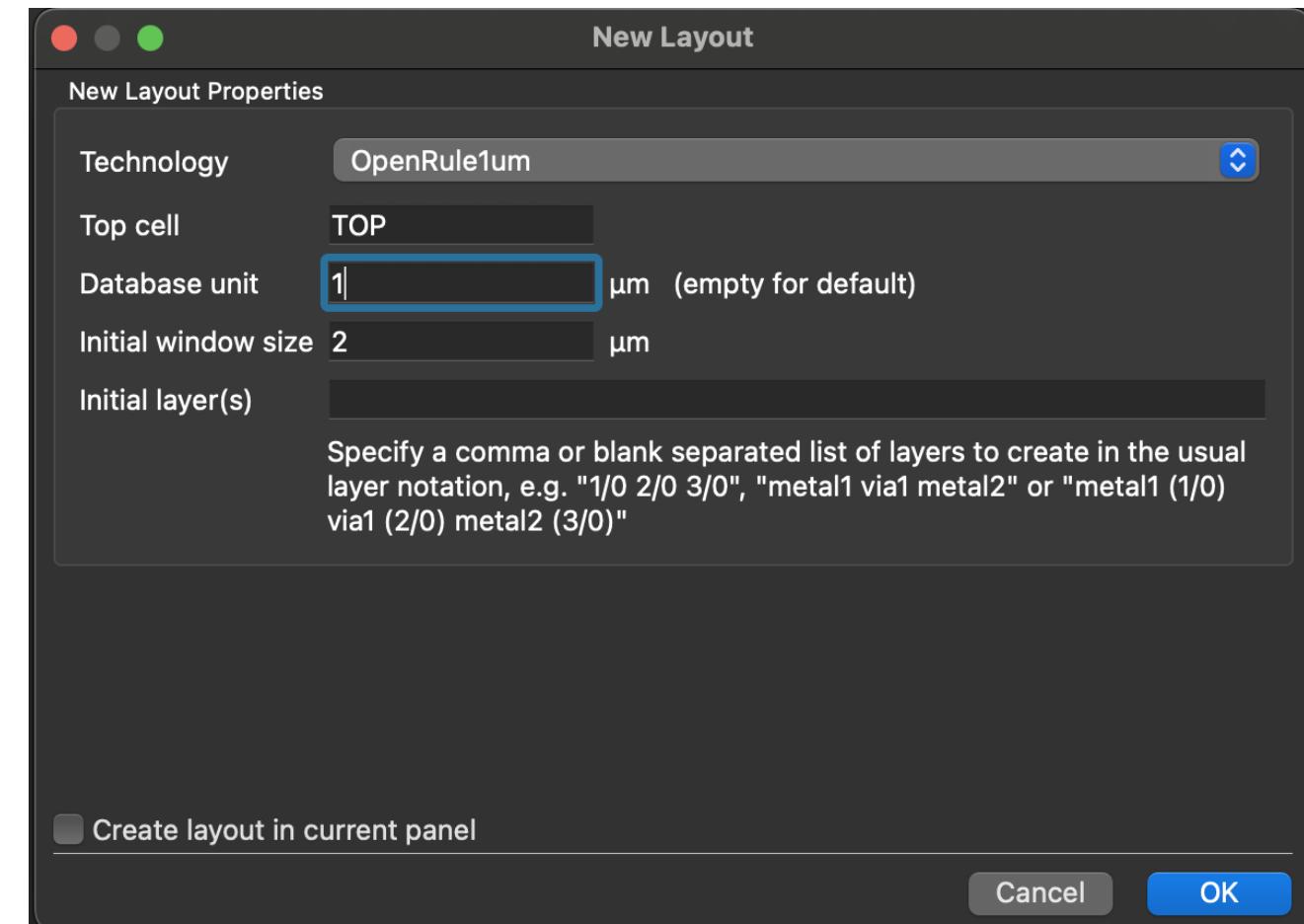


KLayout : バックグラウンドを「黒」にする



Klayout : レイアウトを描く上での注意

- Technologyを OpenRule1umにする
- Database unit を 1にする

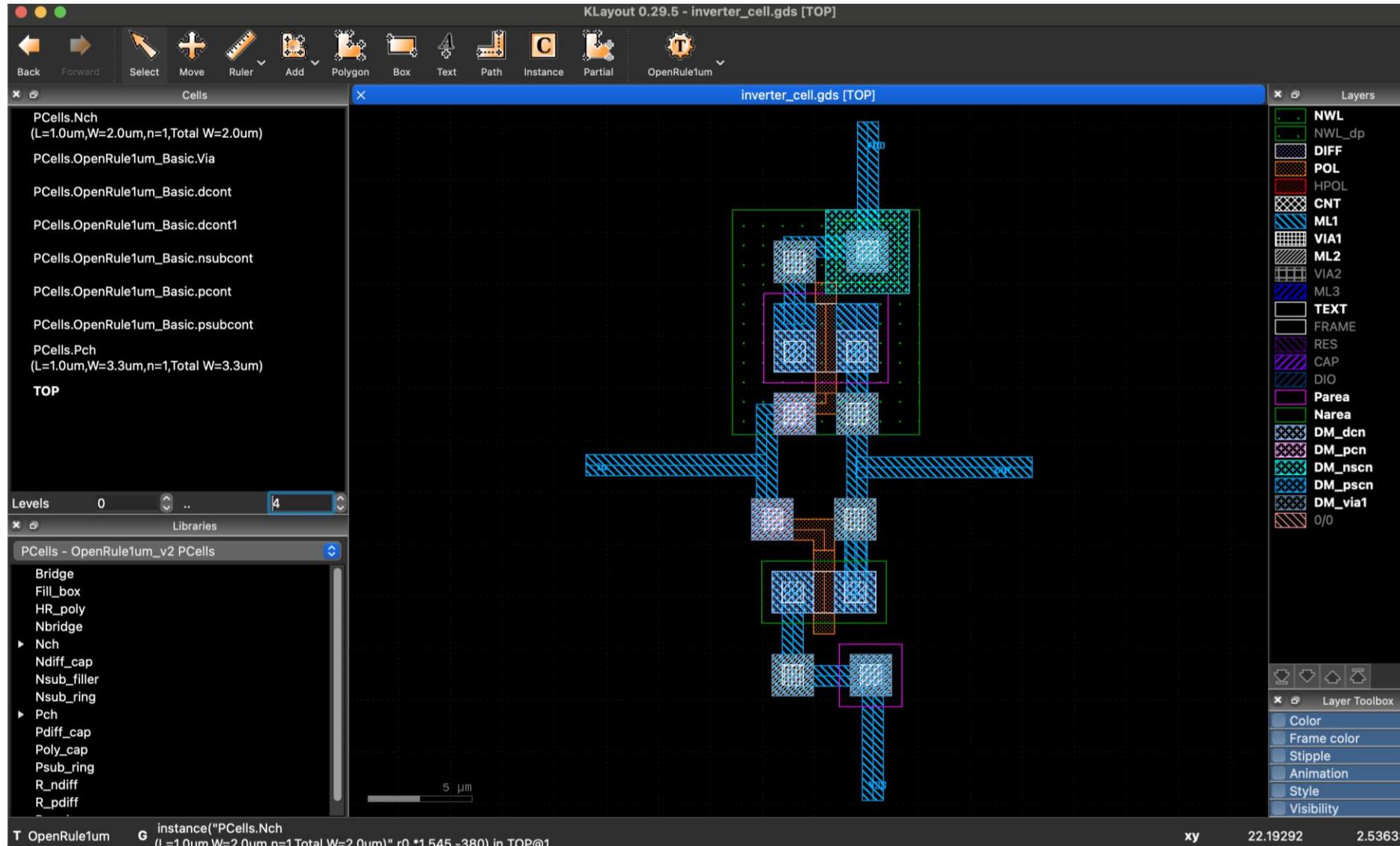


Klayout : デザインルールの場所

- 必ずデザインルールを守る

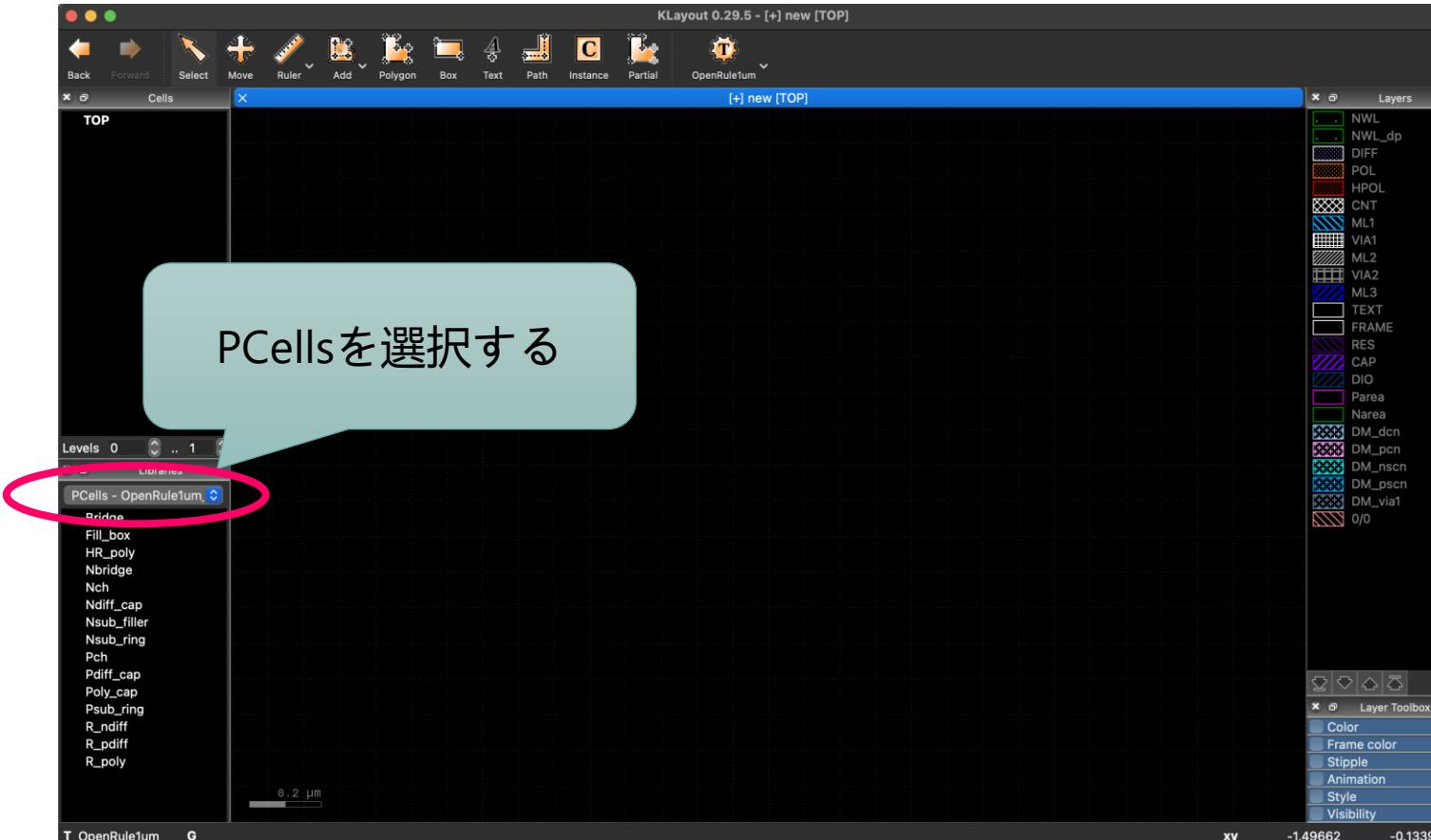
<https://github.com/mineda-support/OpenRule1um>

Klayout : 完成サンプル



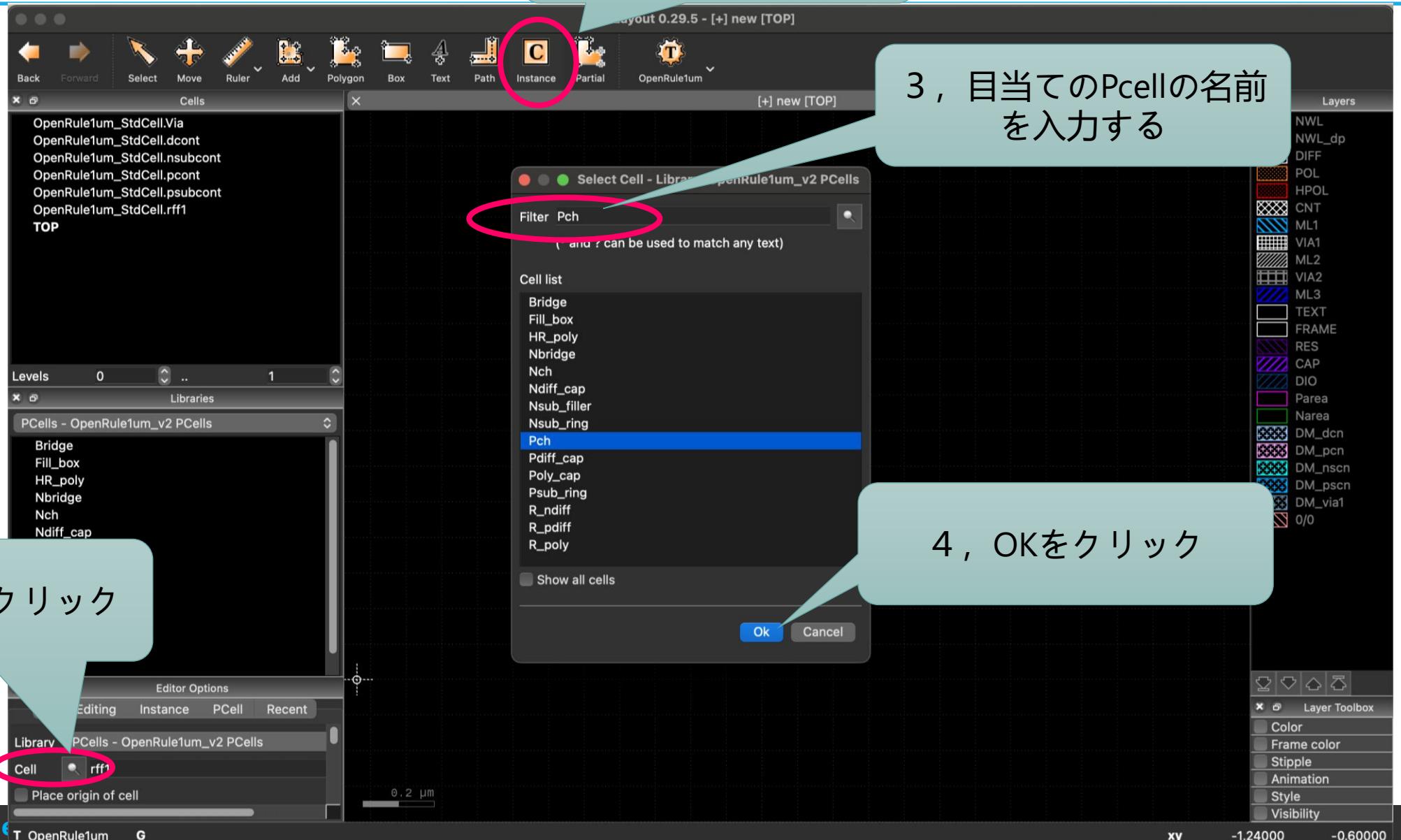
KLayout : P-Cellの使い方

- P-Cell（パラメータ化セル）とは、EDAで使用される再利用可能で構成可能なブロックのことです。P-Cellは、集積回路のレイアウトパターンを特定の設計要件に応じて調整可能なパラメータで作成することができます。これにより、各プロジェクトごとにレイアウト全体を再設計する必要がなくなり、設計プロセスの効率と柔軟性が大幅に向上します。



Klayout : P-Cellの使い方

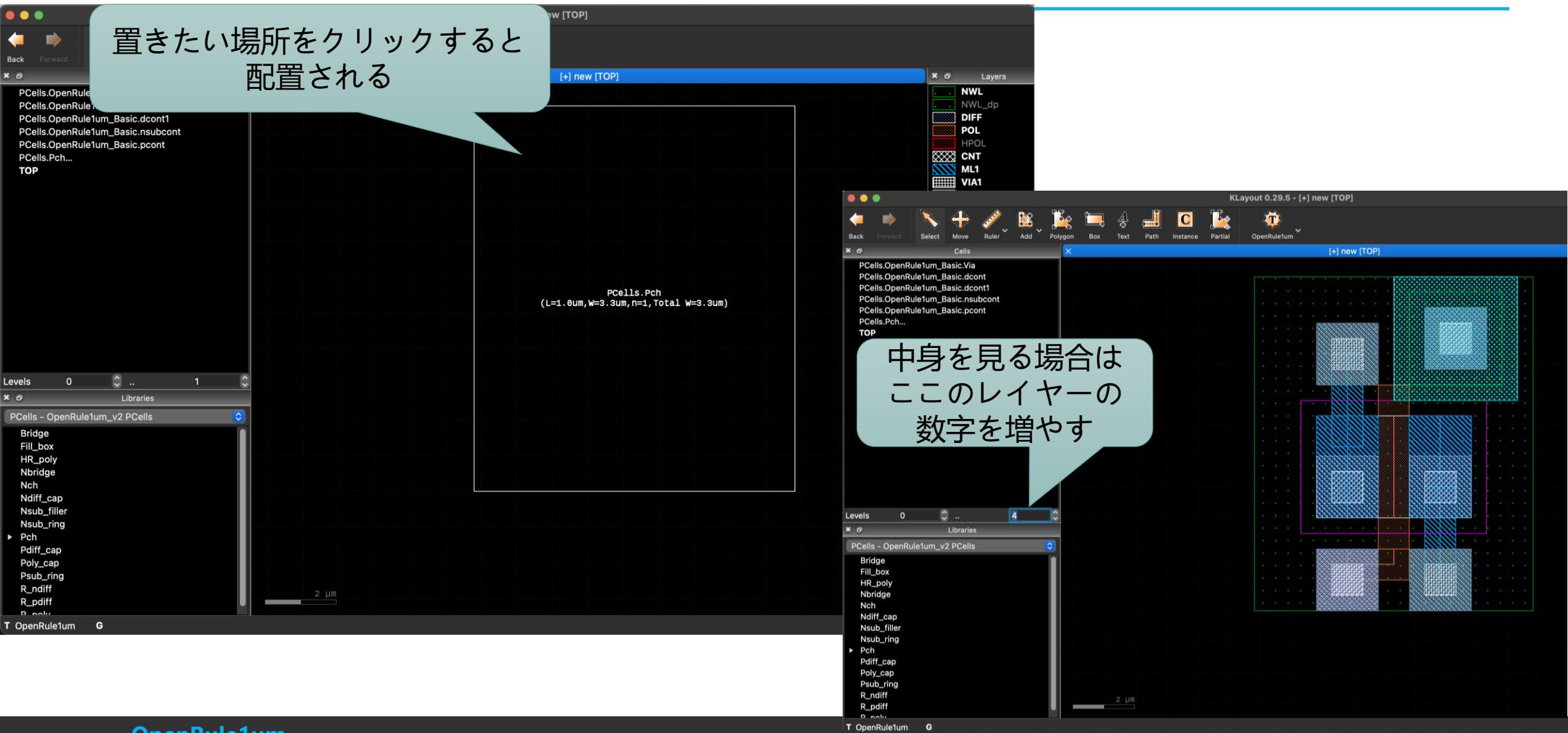
1 , Instanceをクリック



KLayout : P-Cellの使い方

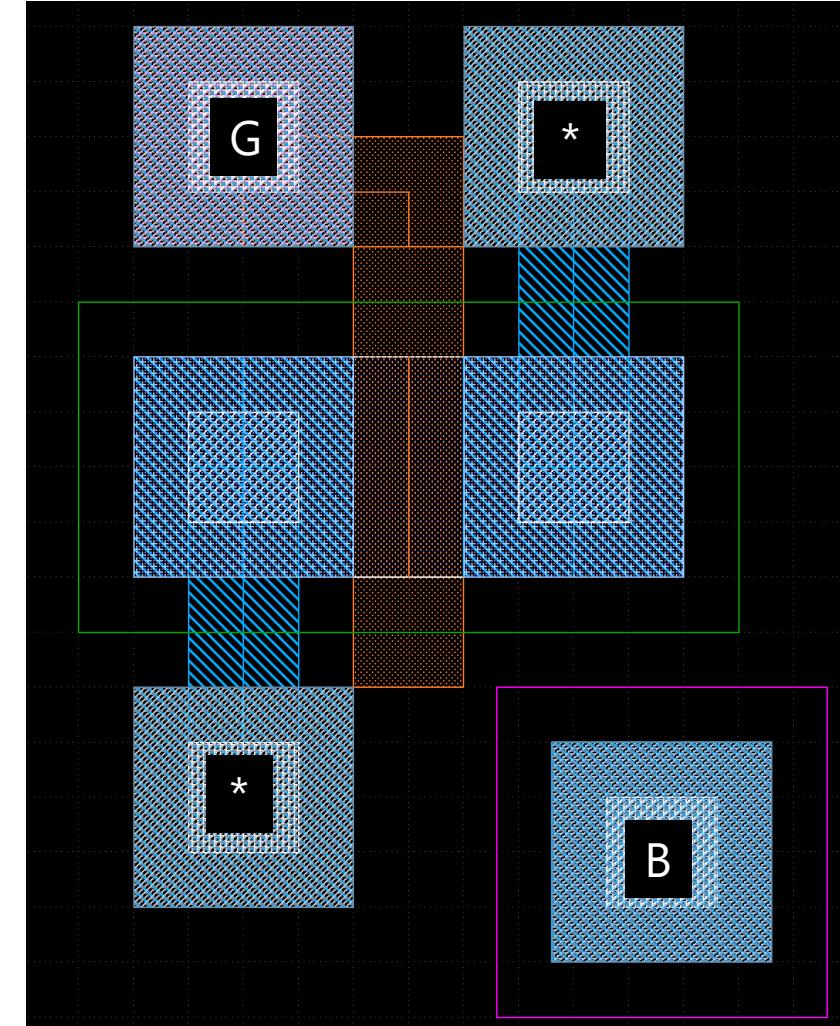
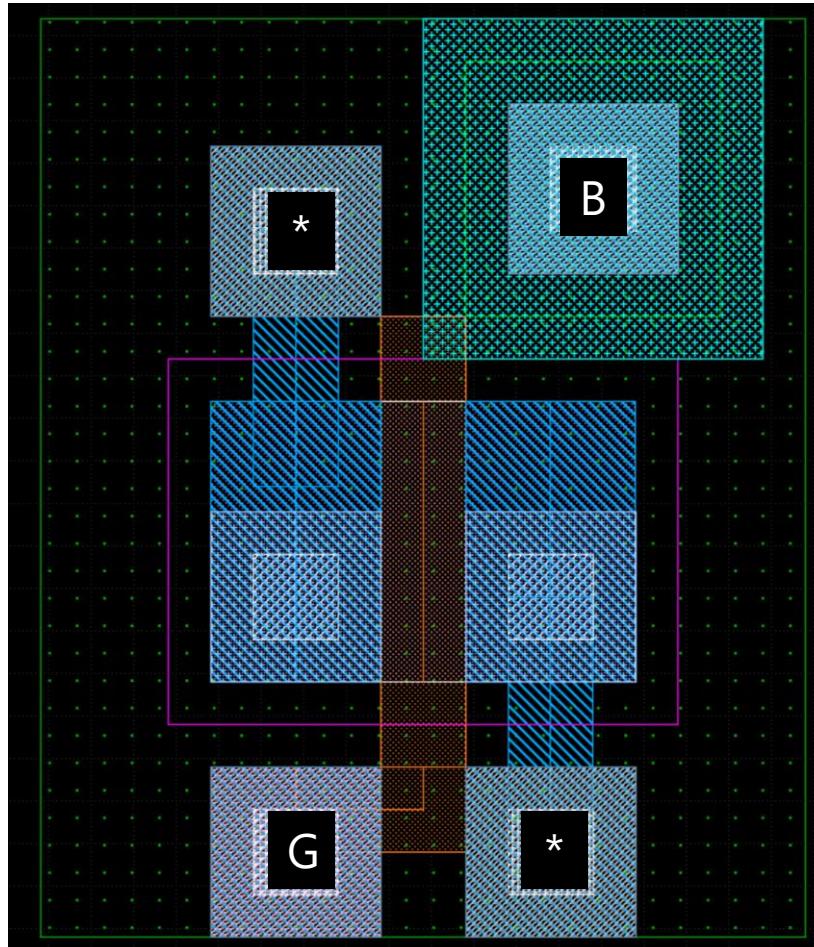


Klayout : P-Cellの使い方



Klayout : タップ付 PFET と NFET

- [*] どちらをドレインにしてどちらをソースにするかは任意。



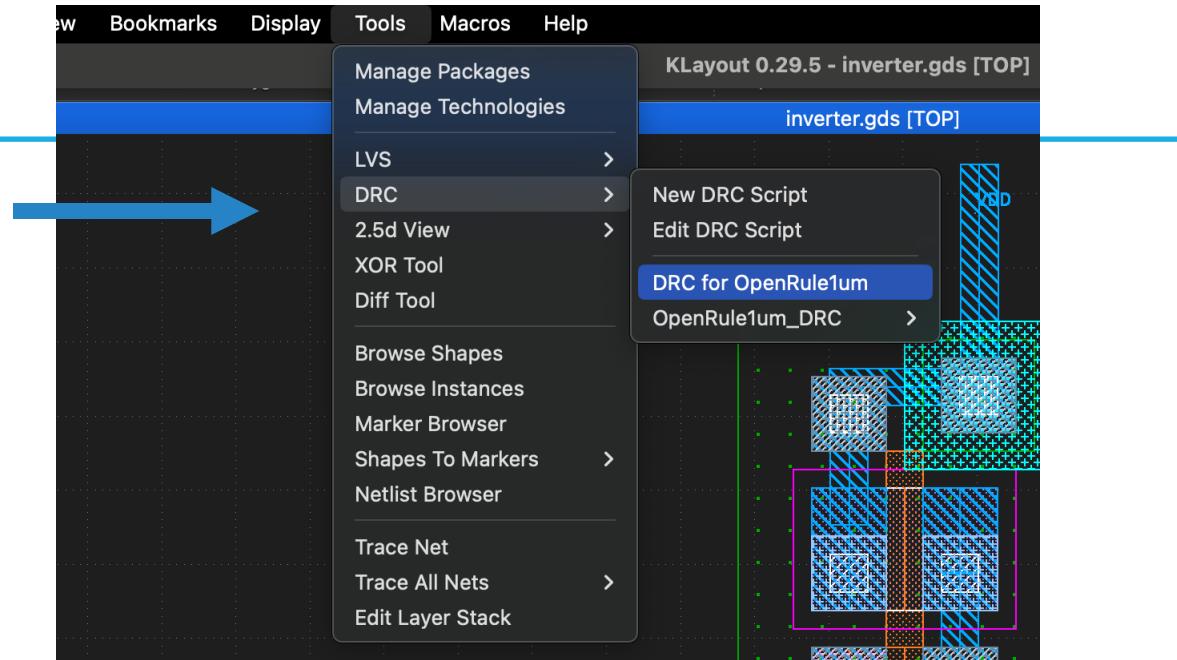
アナログLSIの設計フロー

1. 回路図を描く
2. シミュレーションをする
3. 回路図を基にレイアウトを描く
- 4. レイアウトを検証する**
5. レイアウトを基に寄生成分を考慮したシミュレーションをする
6. (フレームに載せる)

Klayout : レイアウト検証

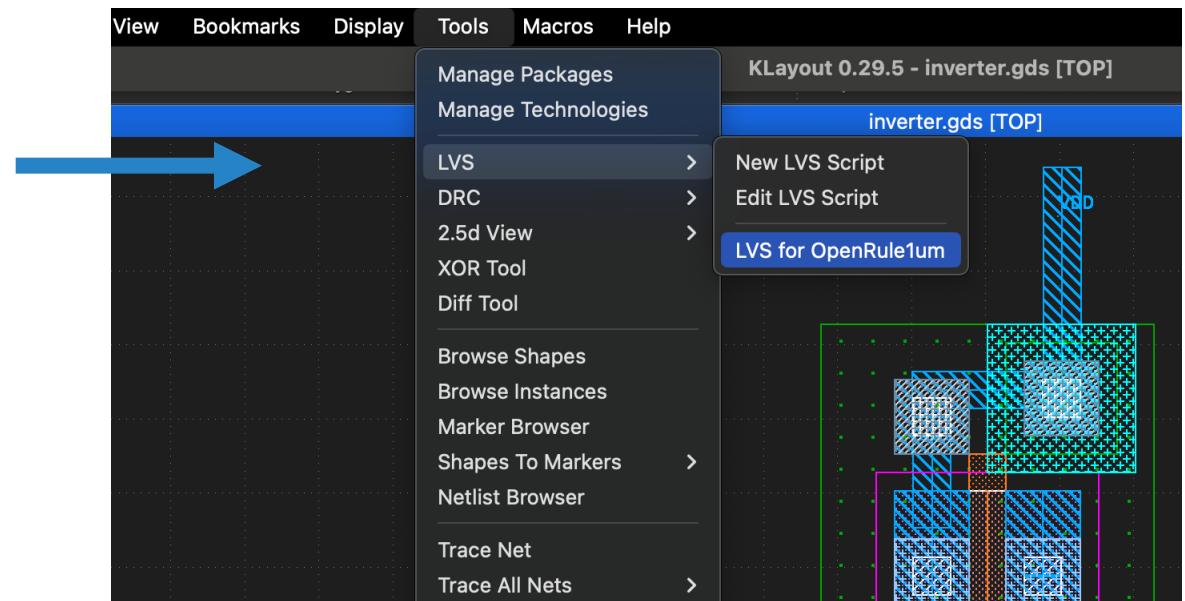
Design Rules Check (DRC)

- 指定されたデザインルールから違反していないか検証

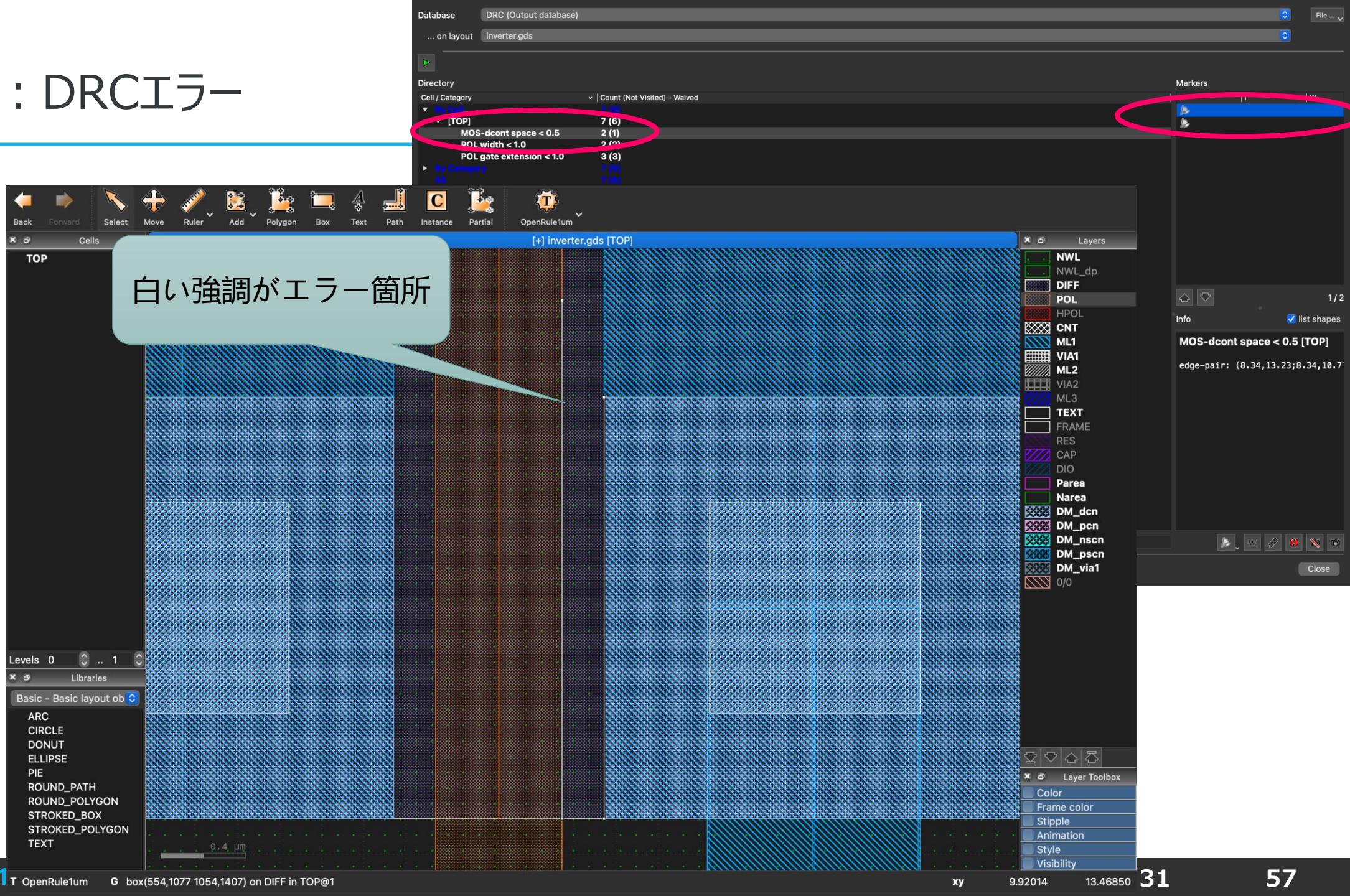


Layout Versus Schematic (LVS)

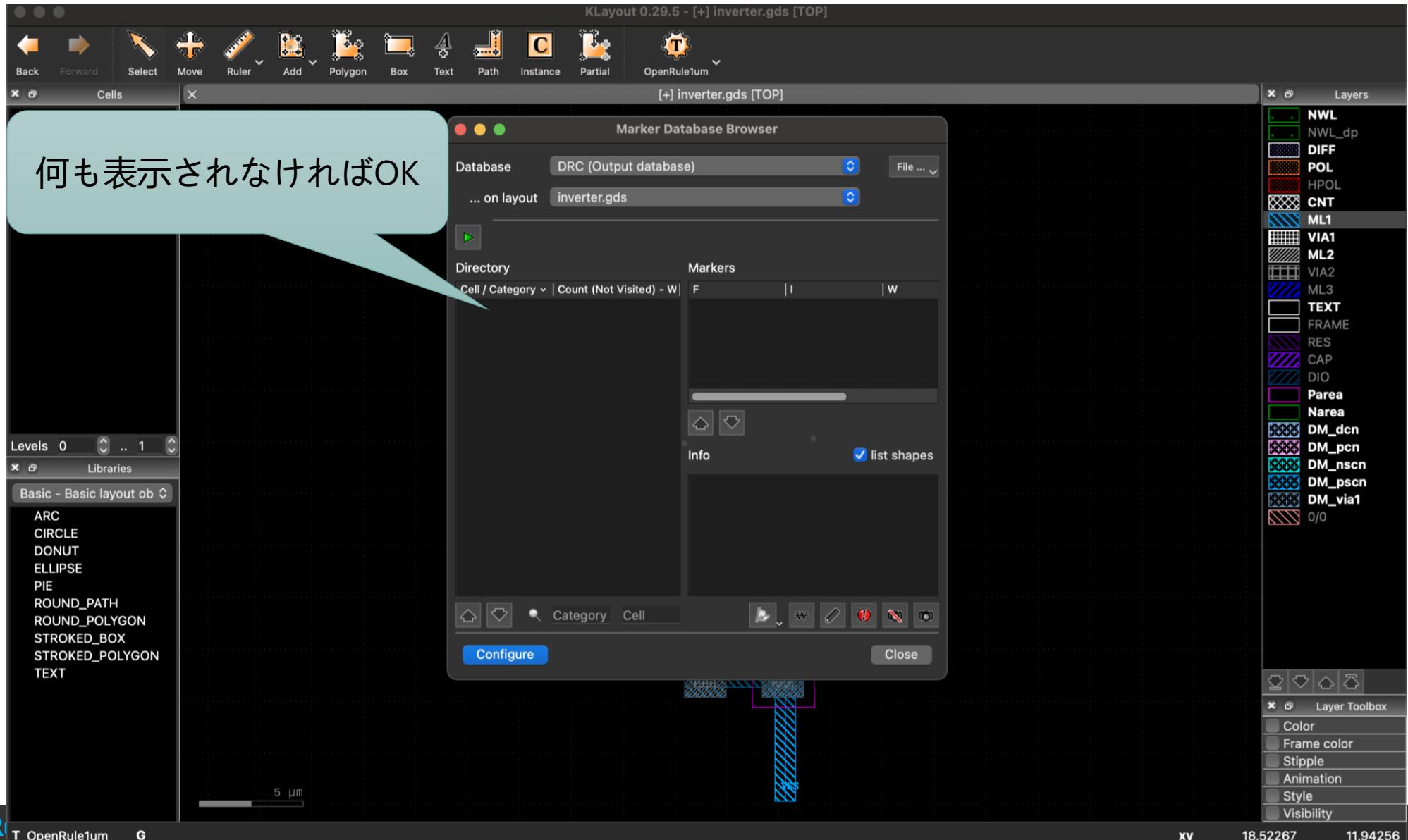
- レイアウトが回路図通りに描けているか検証



Klayout : DRCエラー

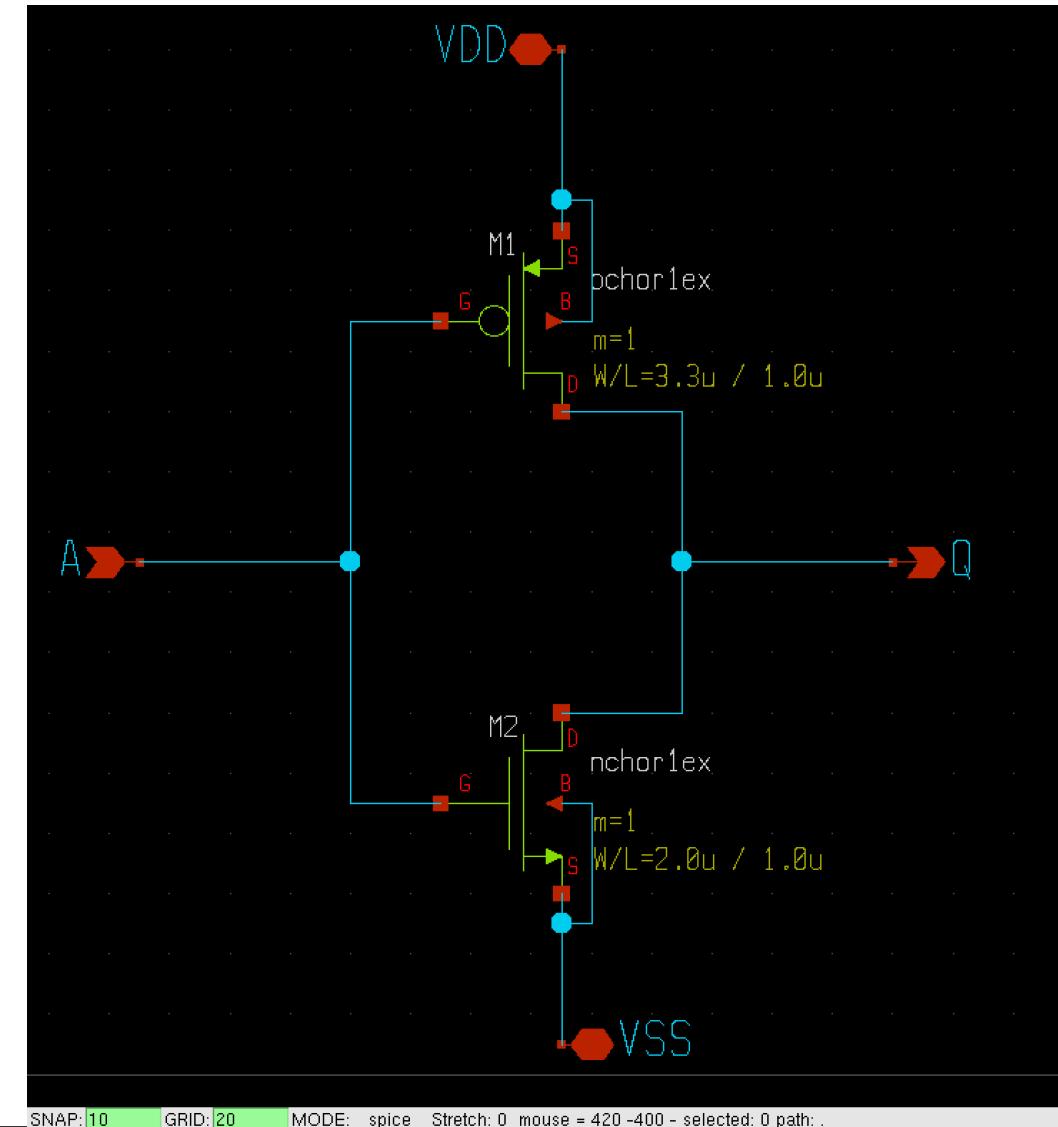


Klayout : DRC OK



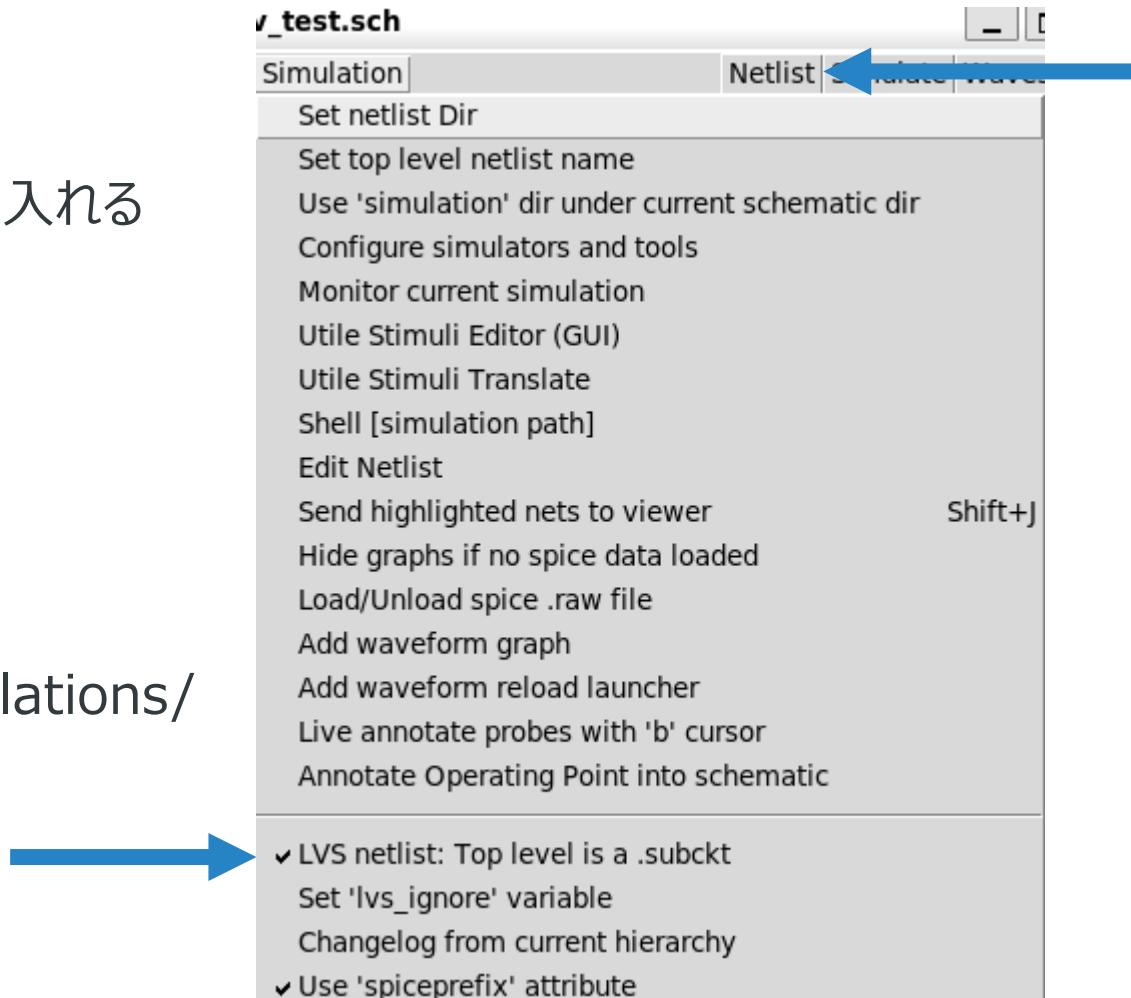
Klayout : LVS用ネットリストの出力方法 (1/2)

- xschem上であらかじめLVS用の回路図を作成しておく。
- ピンはipin, iopin, opin のみを使用する。
 - vdd.symやgnd.symを使用するとうまく通らない。
- ファイルを同じ名前にする 例: nand.sch, nand.gds

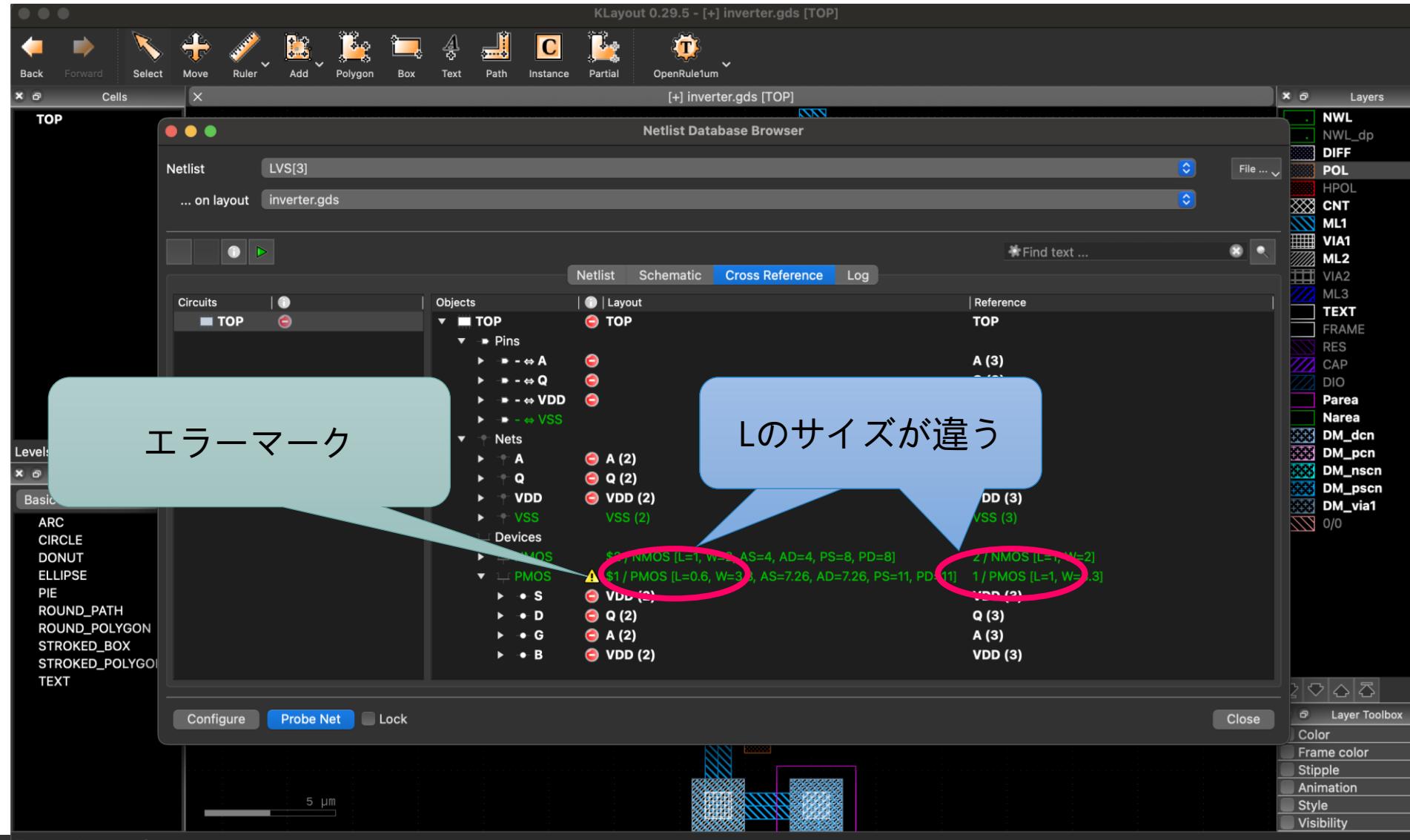


Klayout : LVS用ネットリストの出力方法 (2/2)

- xschem上で**LVS用**のネットリストを出力する。
 1. メニュー Simulation > LVS netlist に✓を入れる
 2. Netlistを押す
- 終わったら閉じて良い
- デフォルトの生成場所は ~/.xschem/simulations/



Klayout : LVS NG



Klayout : LVS OK

