

アナログ開発ツールの使い方（東海理化 = TR10）

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

今村 謙之 (Noritsuna Imamura)

Rev.1

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 TR10 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

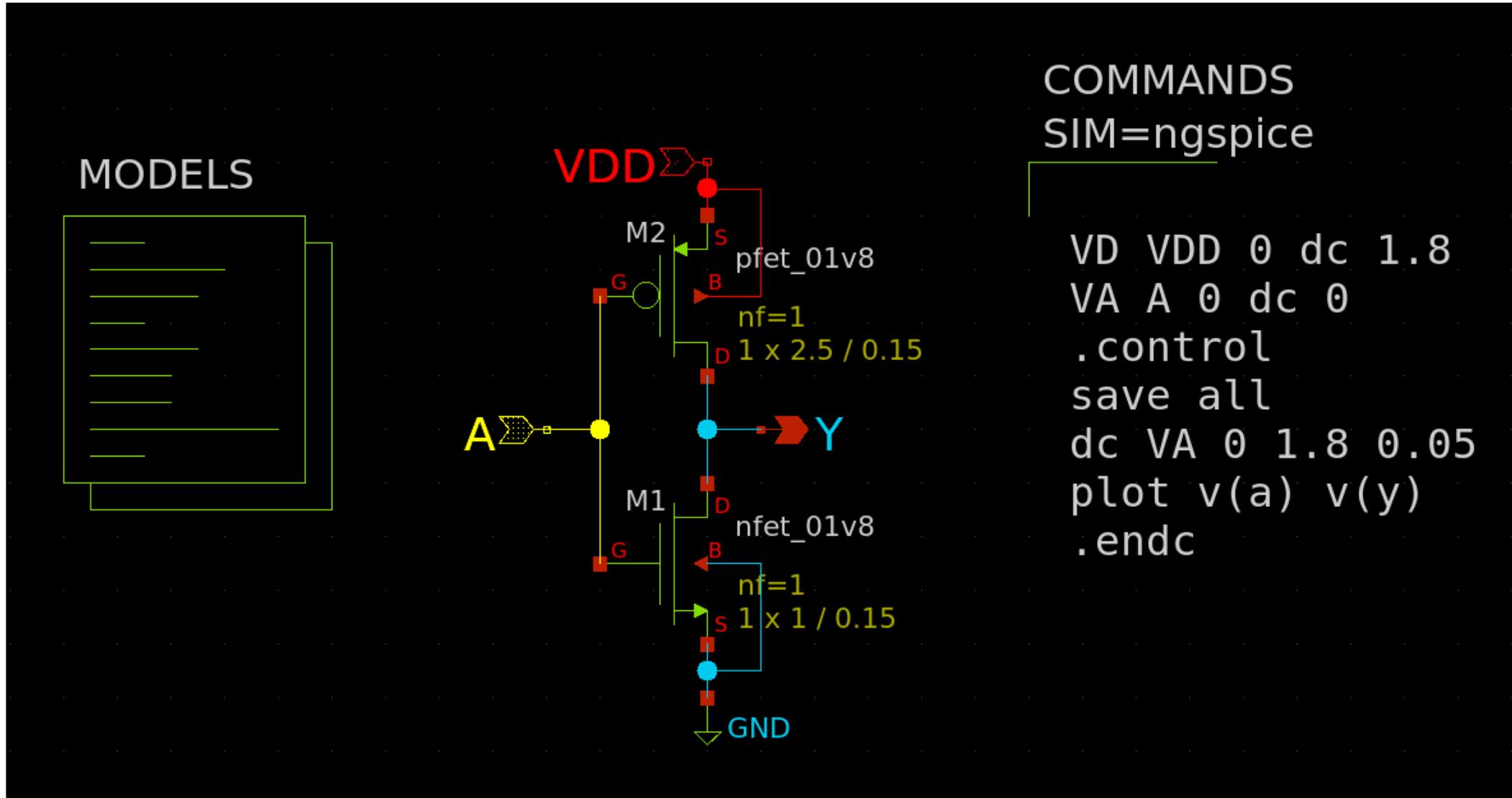
- Xschem, Klayout用PDK
https://github.com/ishi-kai/OpenRule1umPDK_SetupEDA

ローカルでのセットアップ方法

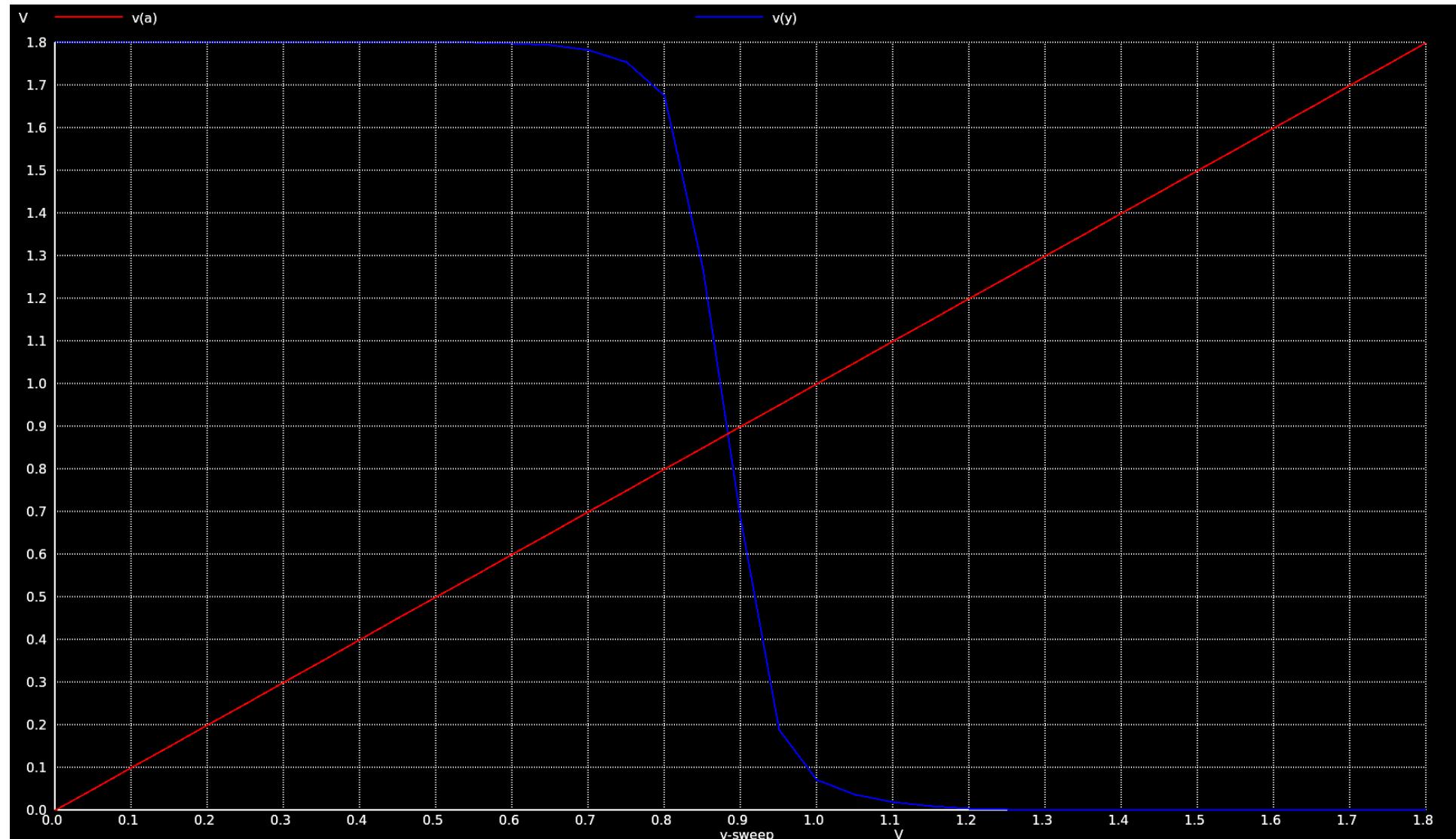
https://github.com/ishi-kai/OpenRule1umPDK_setupEDA

- ここに必要なものが全てが入っています。

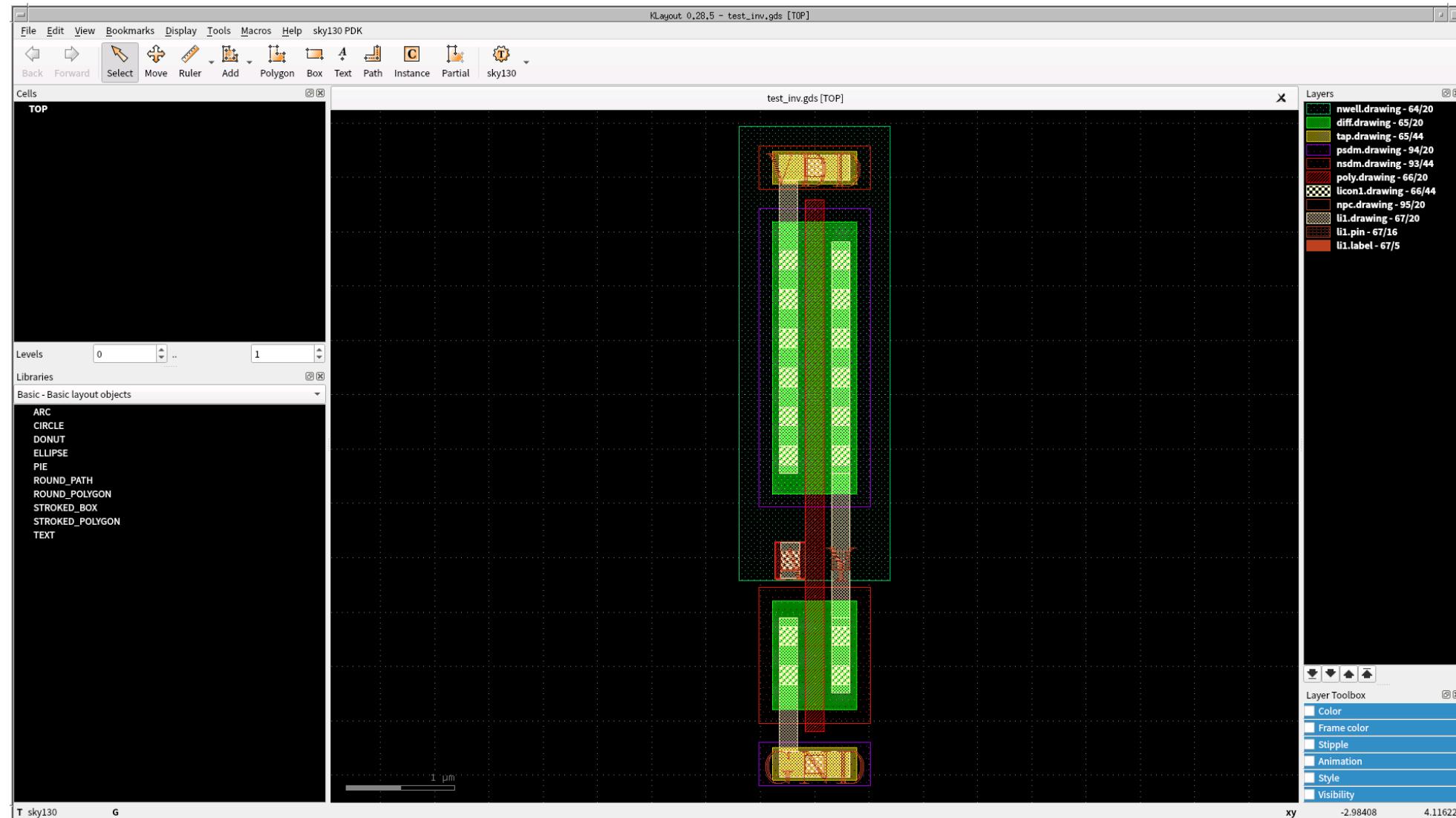
回路図(ベンチマーク)を描いて…



論理検証をして…

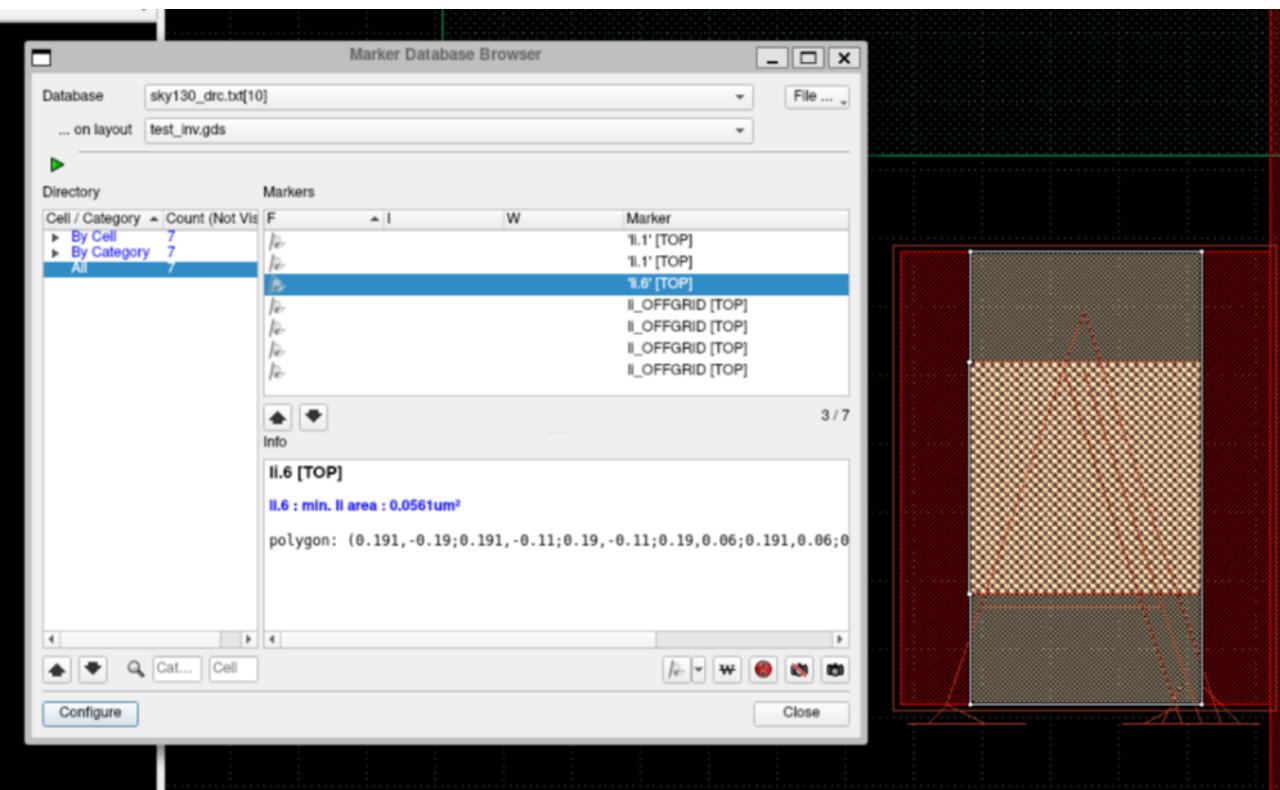


レイアウトを描いて…

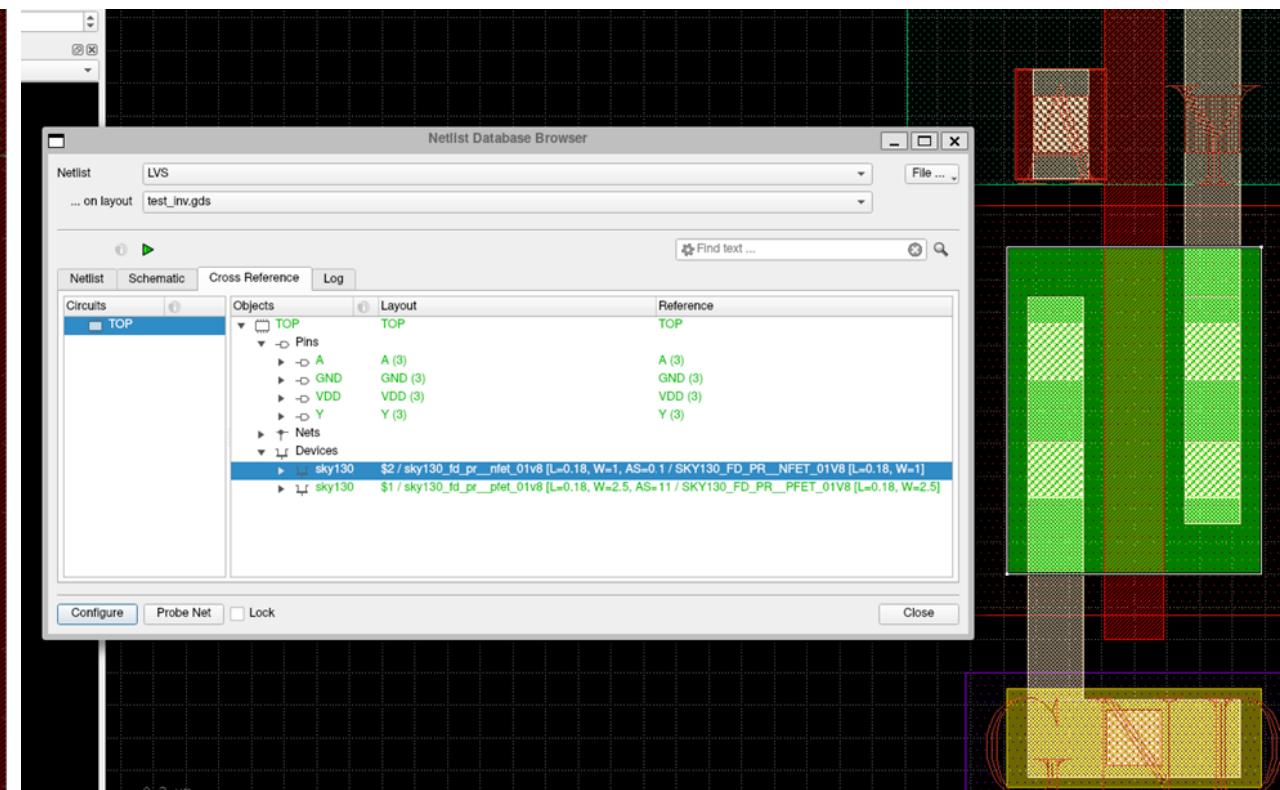


レイアウトを検証して…

DRC



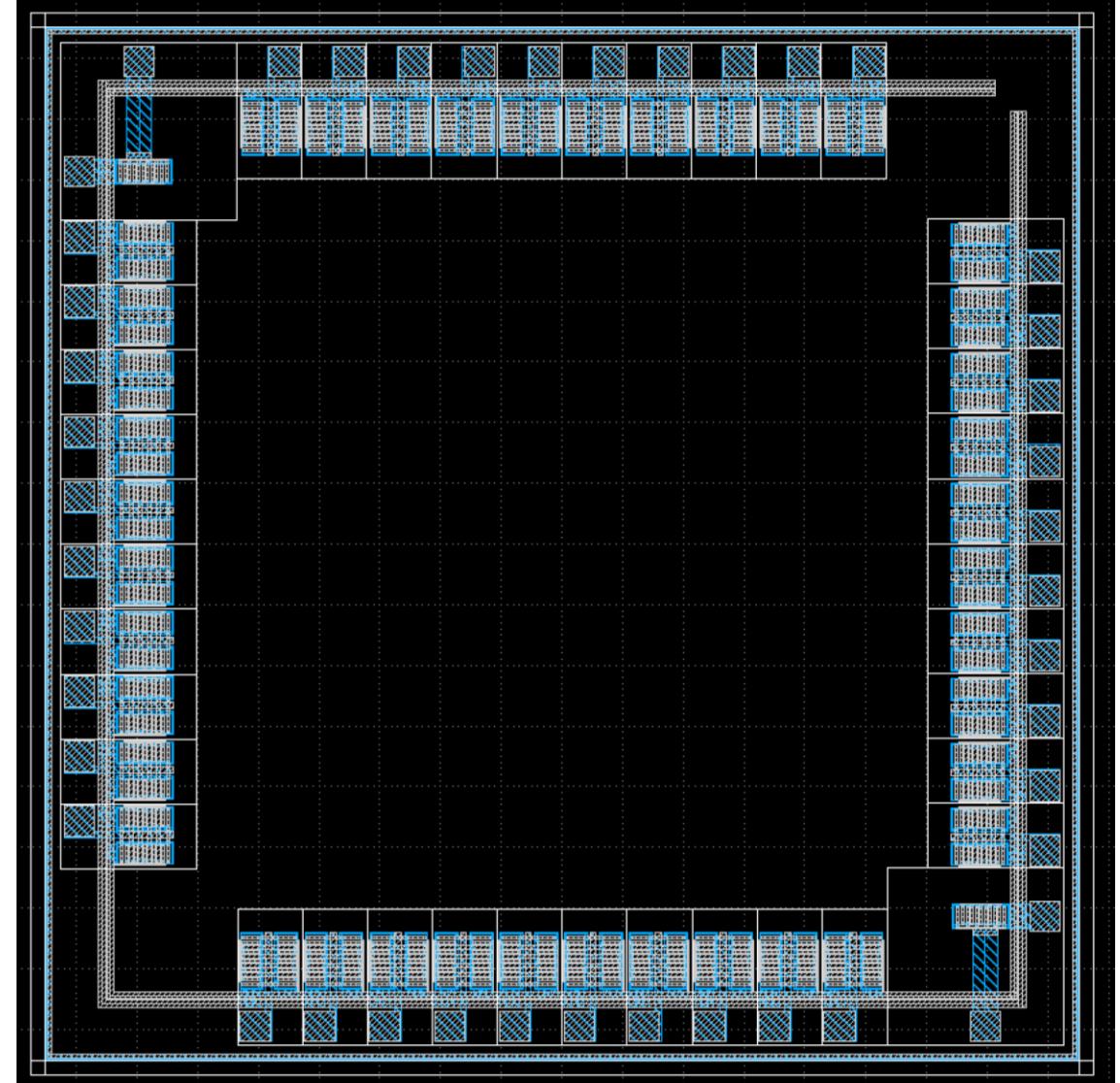
LVS



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

- 44ピン(ユーザが使用可能なピンは40ピン)
 - 自分が使用するピンについてはピンリストを参照



九州大学大学院システム情報科学府附属価値創造型半導体人材育成センター主催 2025年度実習シリーズで採用



2025_九州大学価値創造型半導体人材育成センター_実習シリーズ③

集積回路設計・評価ハンズオンセミナー

1

九州地域では、TSMC/JASMをはじめとする半導体製造メーカーの集積が進んでおり、「オール九州」での半導体エコシステム構築が始まっています。しかし、その中核を担う設計人材がとても少ないとことが問題となっています。そこで九州大学価値創造型半導体人材育成センターでは、設計人材育成のため、半導体の設計から評価までの技術を一気通貫で習得できるセミナーを開催します。

内容:CMOS集積回路(Inverter)をOpen Toolにより設計し、レイアウトします。各自のレイアウトは東海理化のシャトルサービスにより試作します。また、出来上がったICチップを測定し、評価します。ICチップは持ち帰ることができます。

日時場所:設計ハンズオンセミナー: 2025年9月24日～25日、九州大学 W2-325号室

ICチップお渡し会＆評価実習: 2026年3月(未定)、九州大学内

※ 9月、3月の2回で1つのセミナーです。WindowsノートPCが必要です。

または、こちら



申込み方法:
・【件名】に参加したいセミナーの名称を記入ください。

・お名前、お名前(ローマ字)、ご所属、メールアドレス をお知らせください。

申し込み先: class_program-at-ecsvc.ed.kyushu-u.ac.jp -at- =@

締め切り:7月23日
先着順で、定員になり次第しめきりとさせていただきます

ご協力:OPEN SUSI, ISHI-KAI, AIST solutions、株式会社東海理化



書籍紹介コーナー

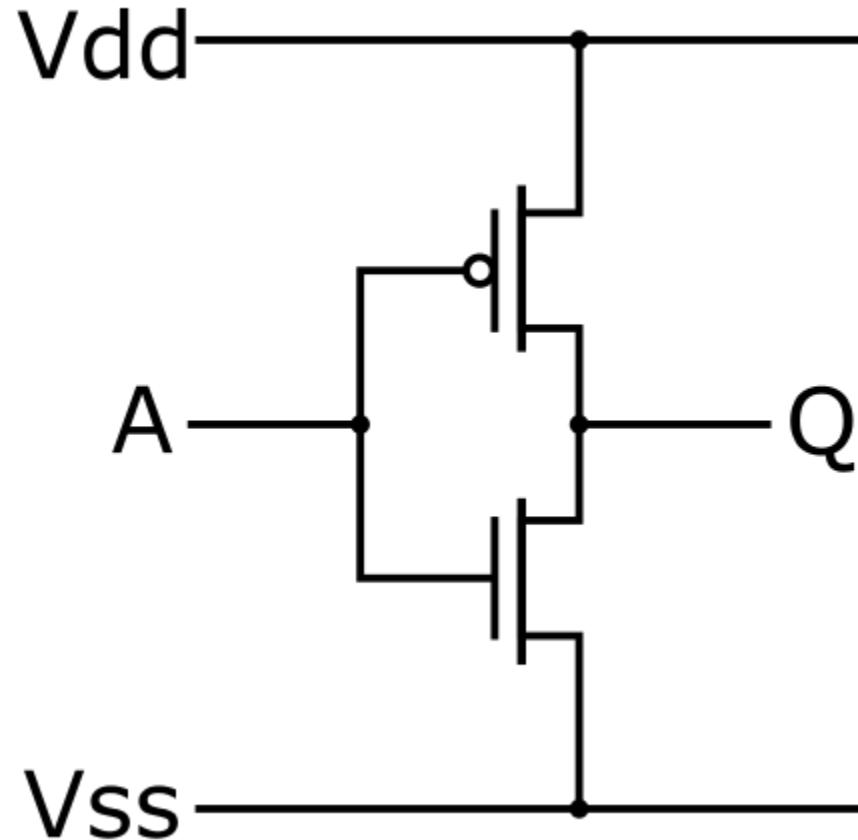
- タイトル : Open Source Silicon Magazine vol.1
 - サブタイトル : はじめてのIC 設計～オリジナルのインバータを作ってみよう～
- 内容
 - 本日の解説から、さらに一步『手前』や端折ってしまっている部分を解説した内容
 - 技術書典「第10回刺され！技術書アワード」のニュースタンダード部門のファイナリストに選出されました！



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

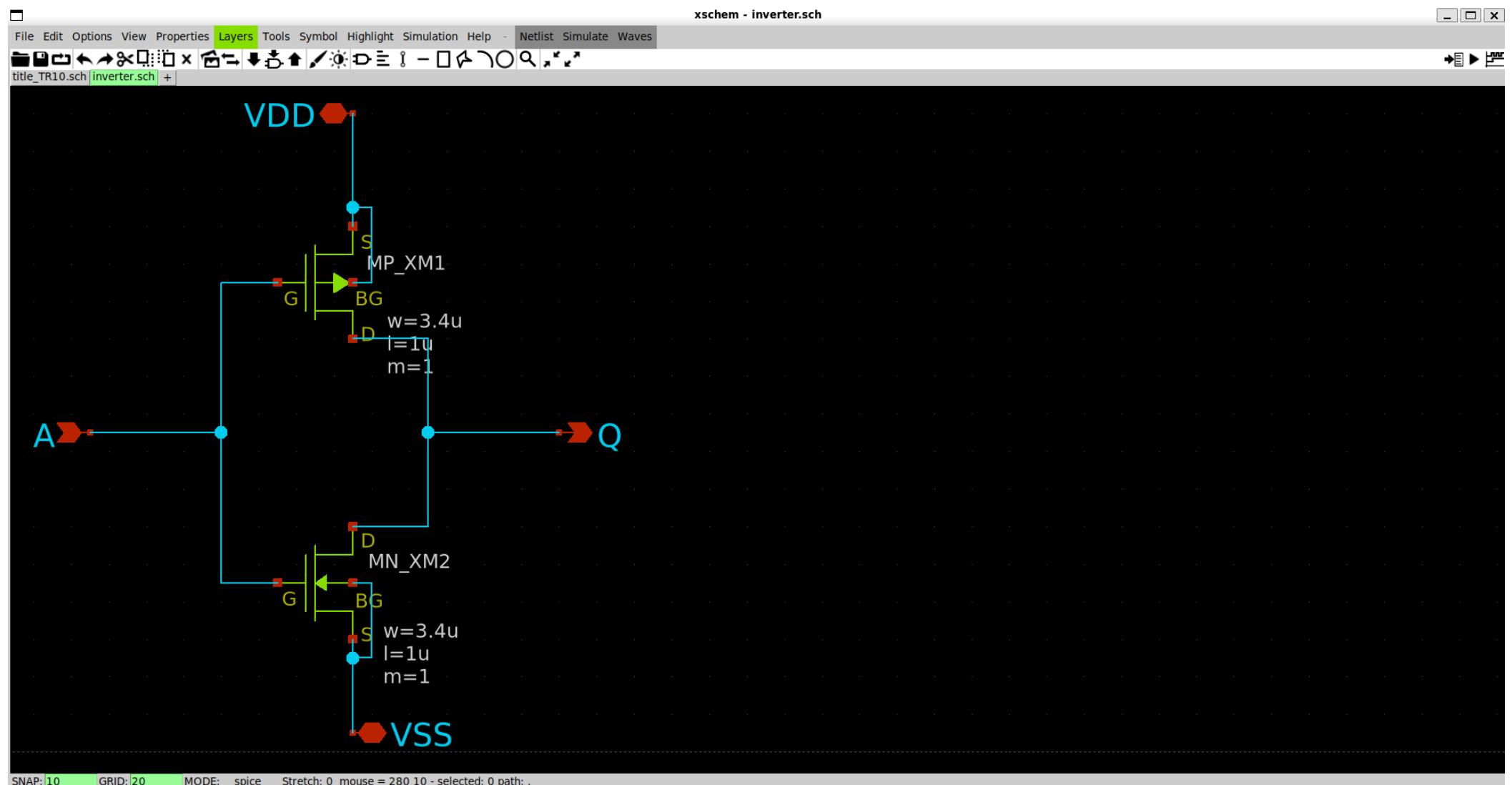
xschem使用上の留意点

- 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

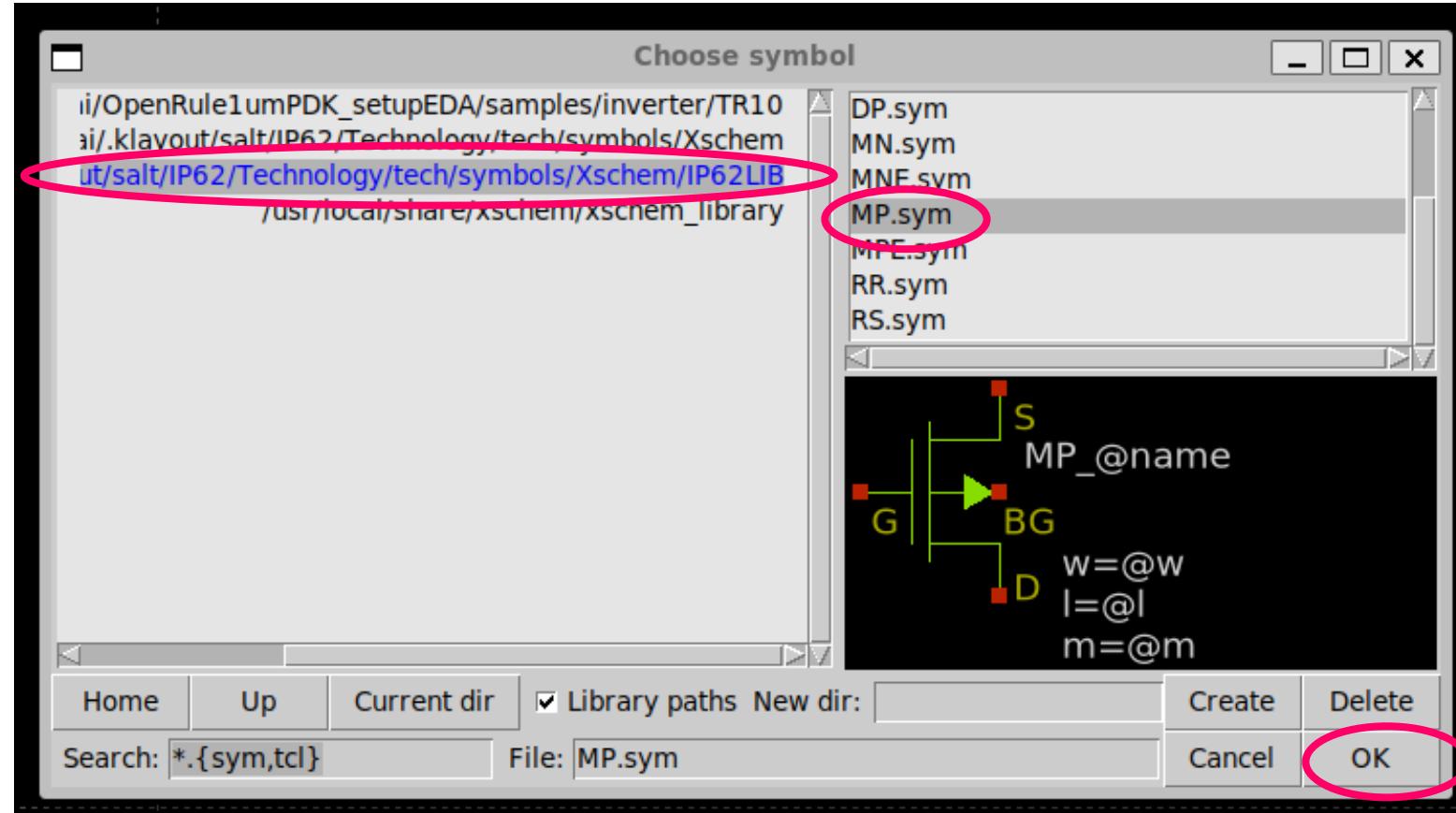
CMOSインバーター回路の例



xschem使い方：新規回路図とMODULEウィンドウ



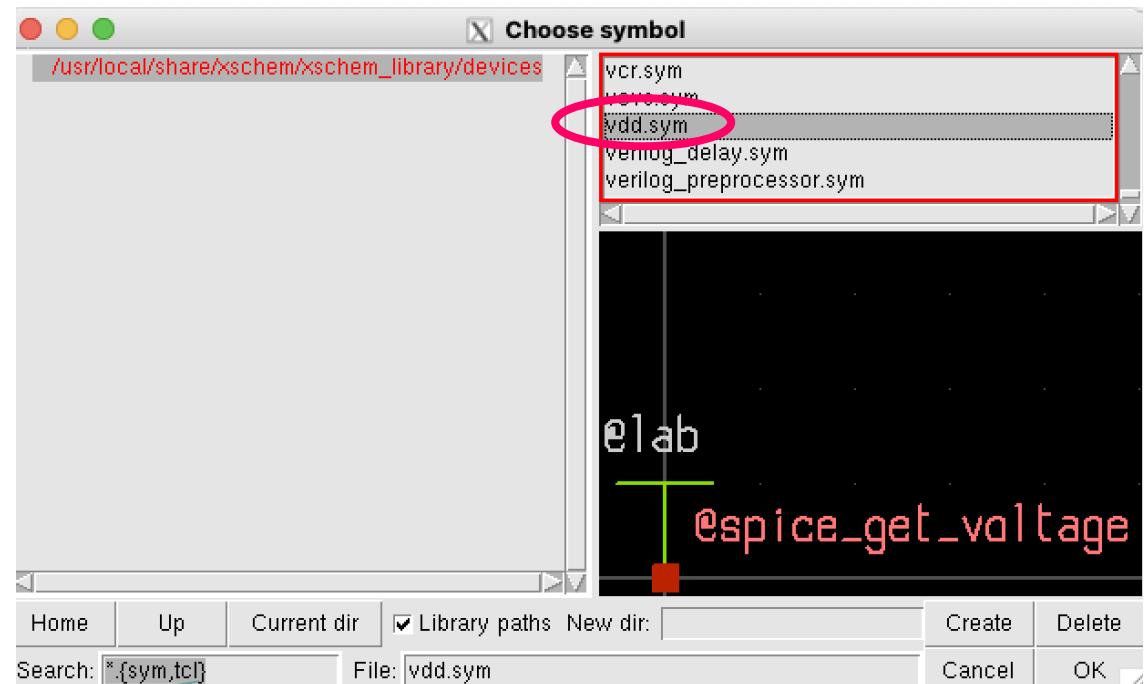
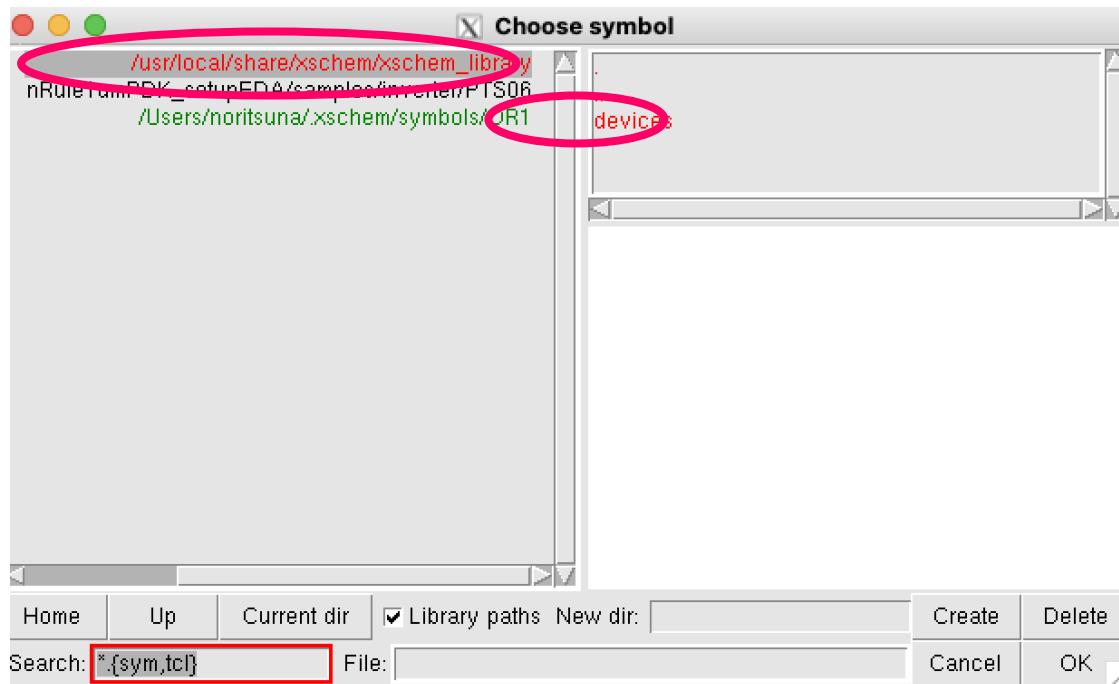
xschem使い方：IP62LIBのMODULEの選択：P-FET



xschem使い方：IP62LIBのMODULEの配置とプロパティ

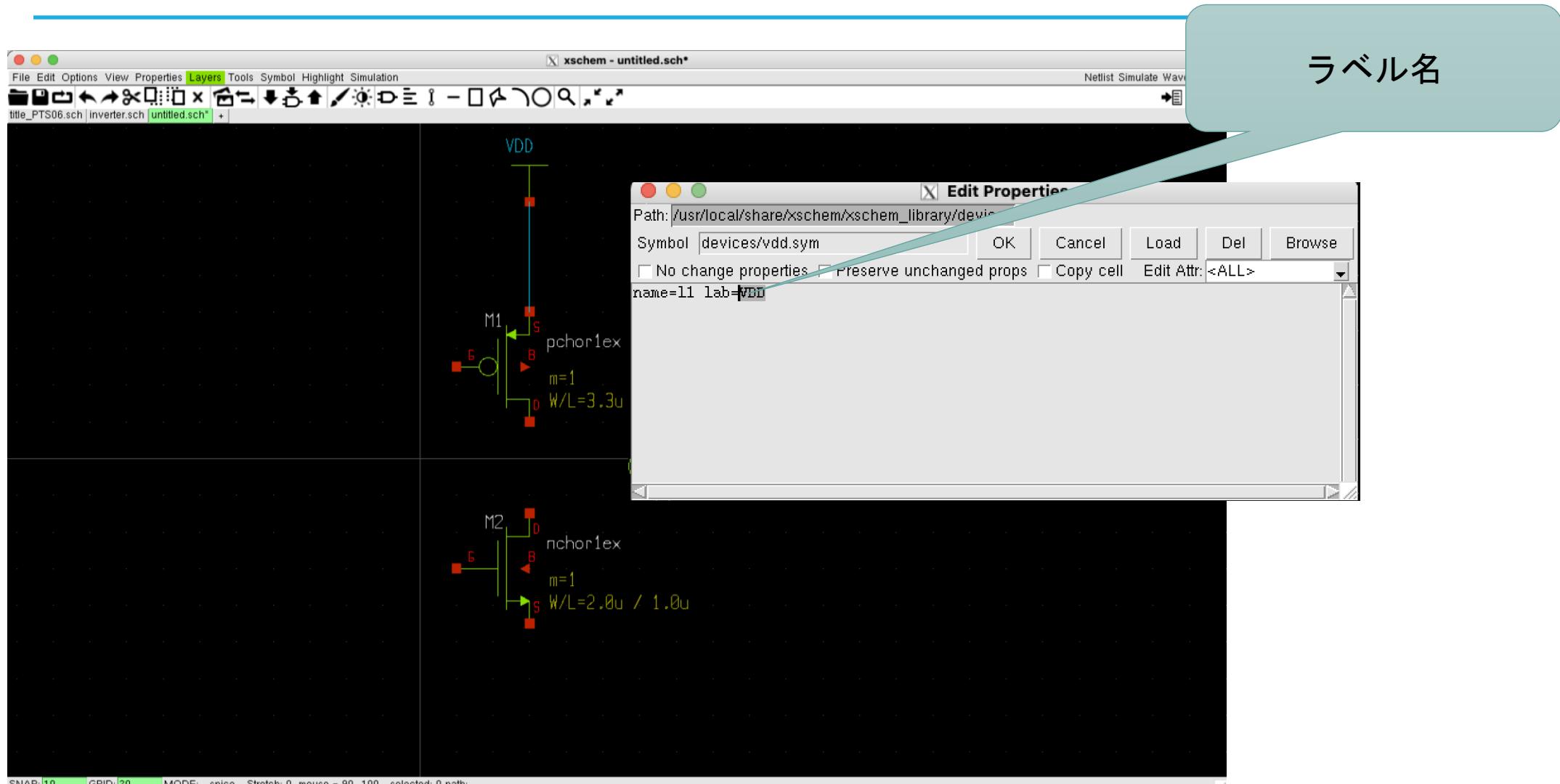


xschem使い方：標準のMODULEの選択：VDD

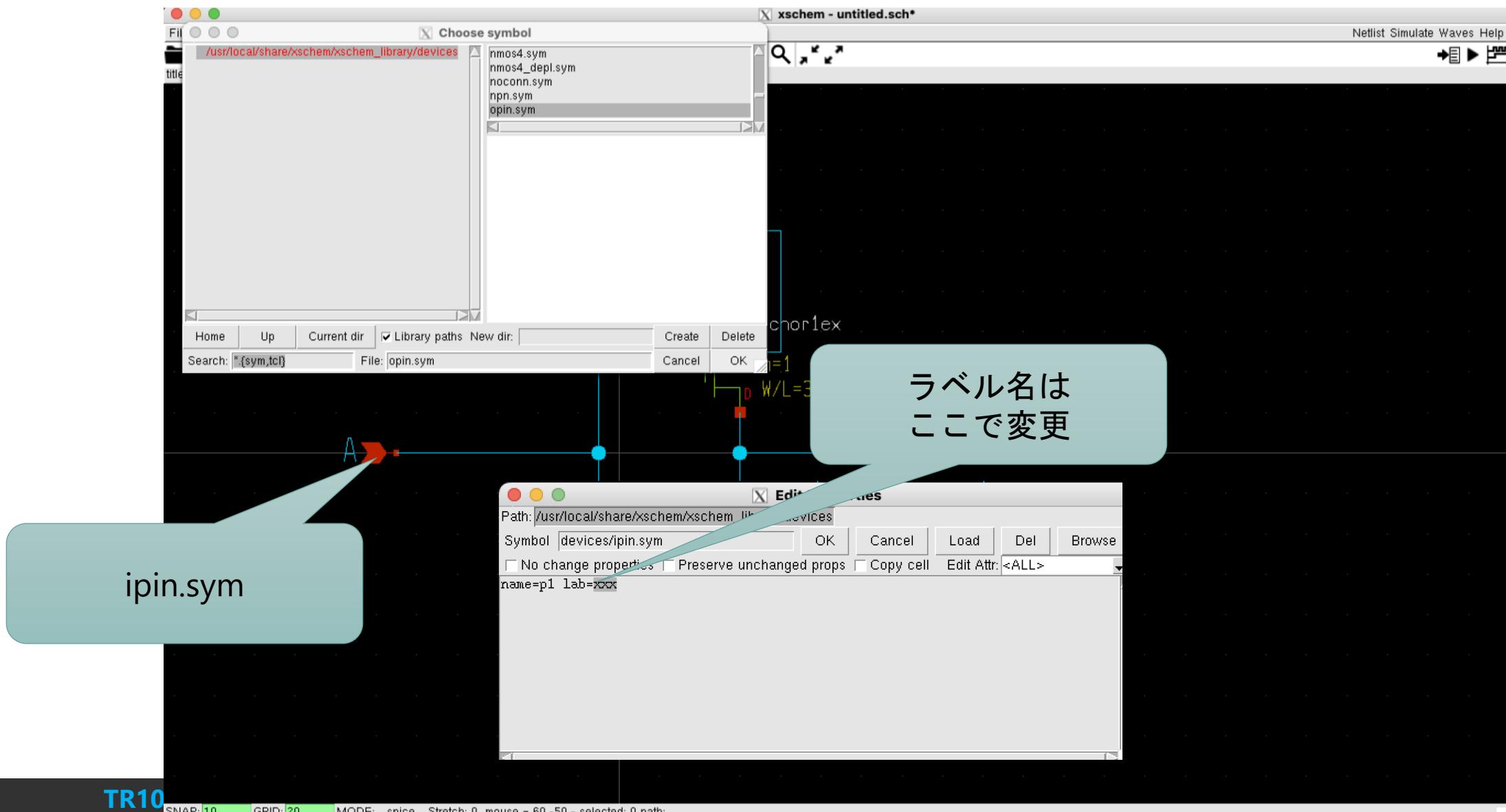


検索も可能

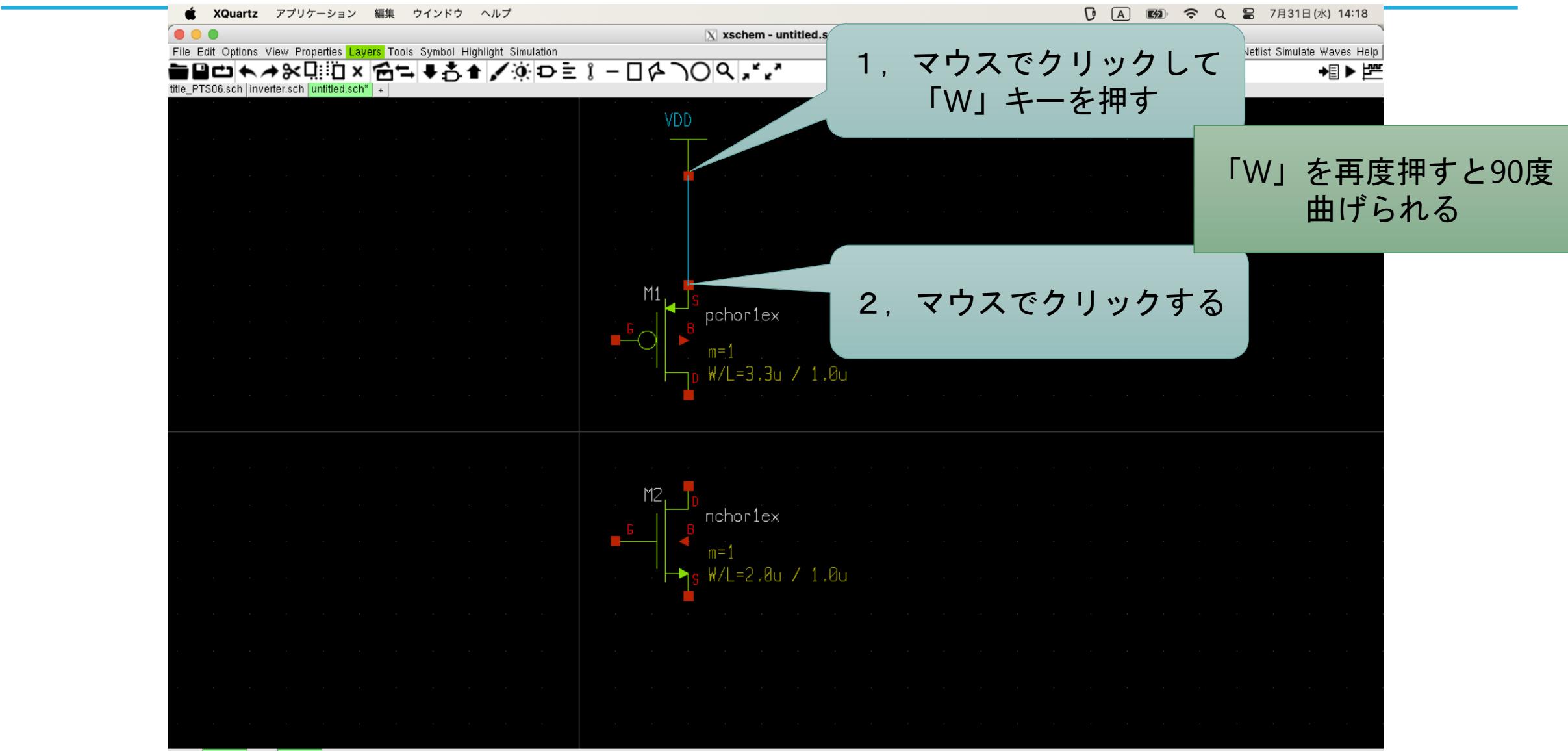
xschem使い方：標準のMODULEの配置とプロパティ：VDD



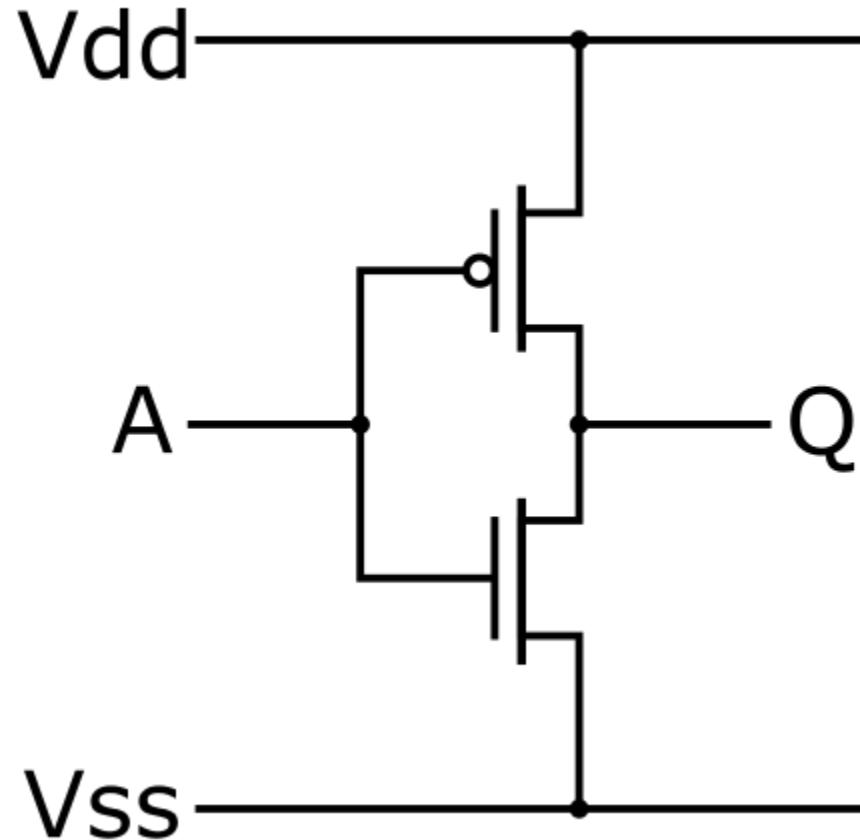
xschem使い方：標準のMODULEの選択：ipin, opin, iopin



xschem使い方： wireの引き方



ハンズオン：CMOSインバータを作る

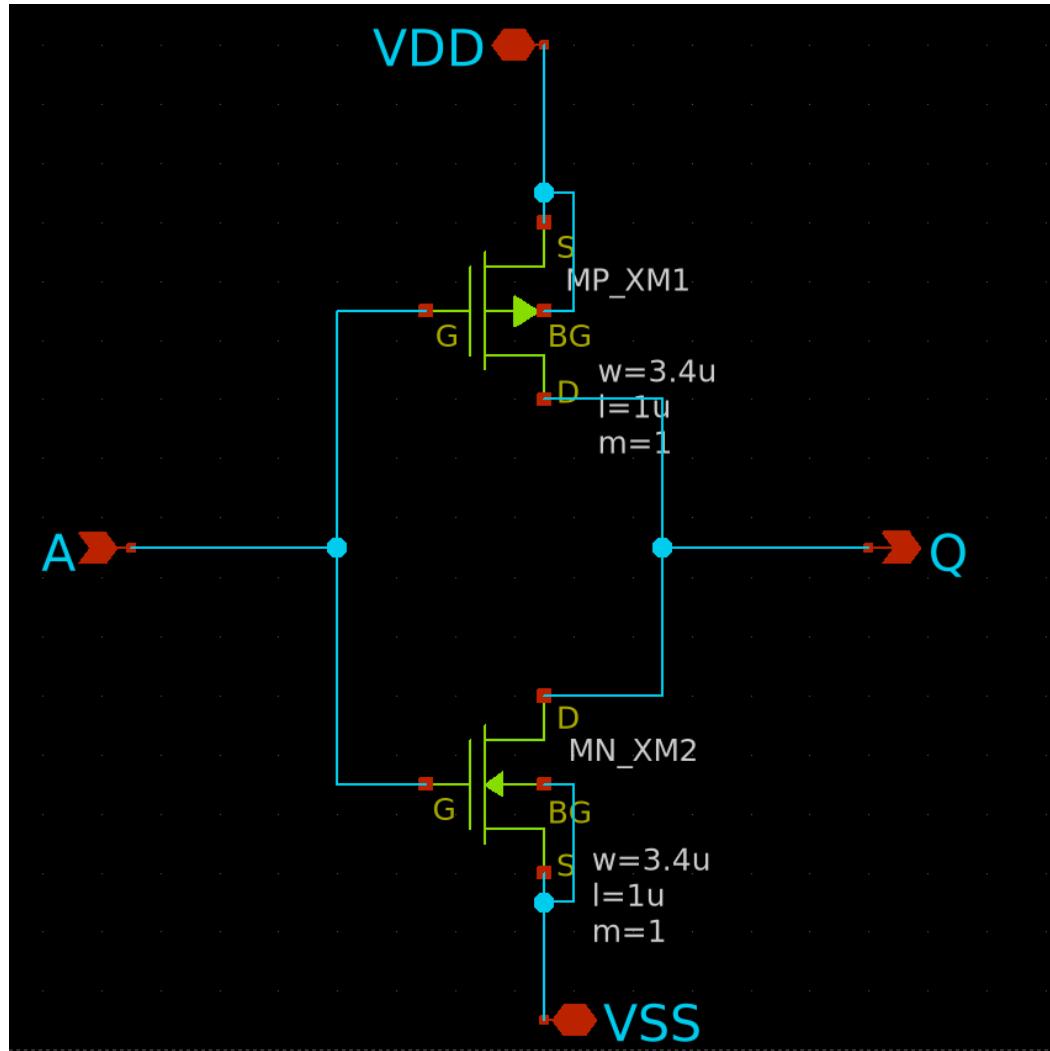


入力A	出力Q
L	H
H	L

↓

入力A	出力Q
V_{ss}	V_{dd}
V_{dd}	V_{ss}

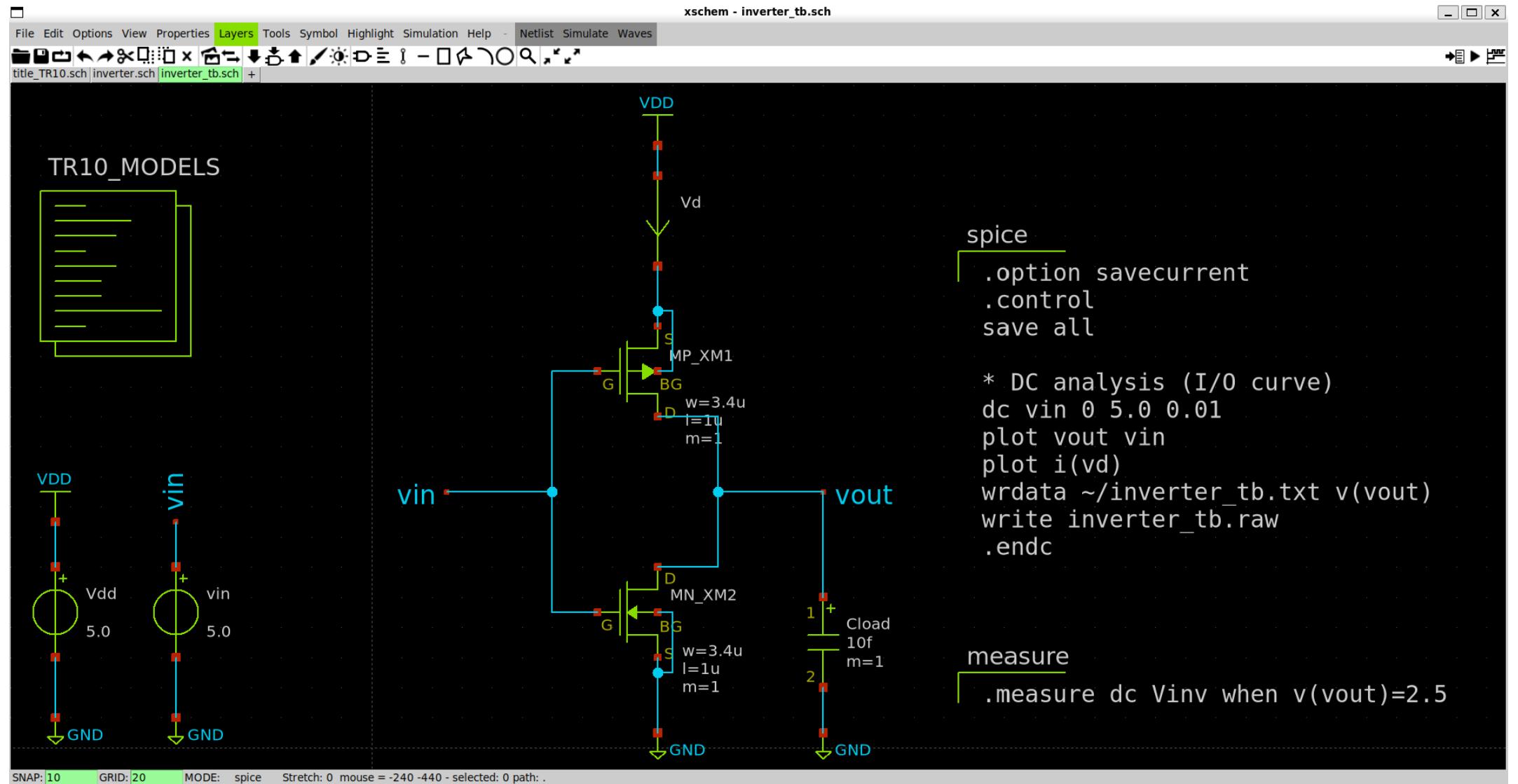
ハンズオン：CMOSインバータのサンプル



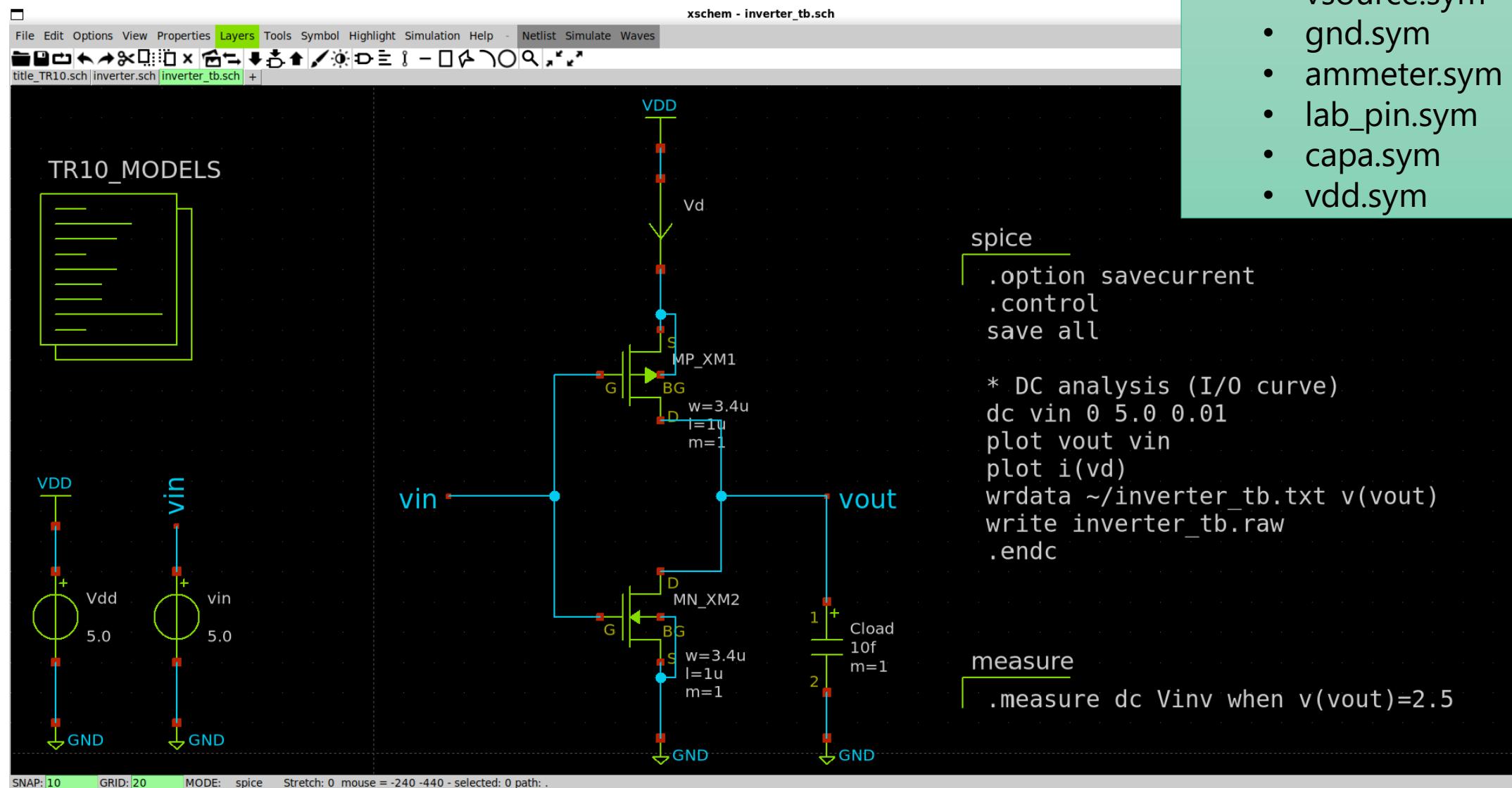
アナログLSIの設計フロー

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



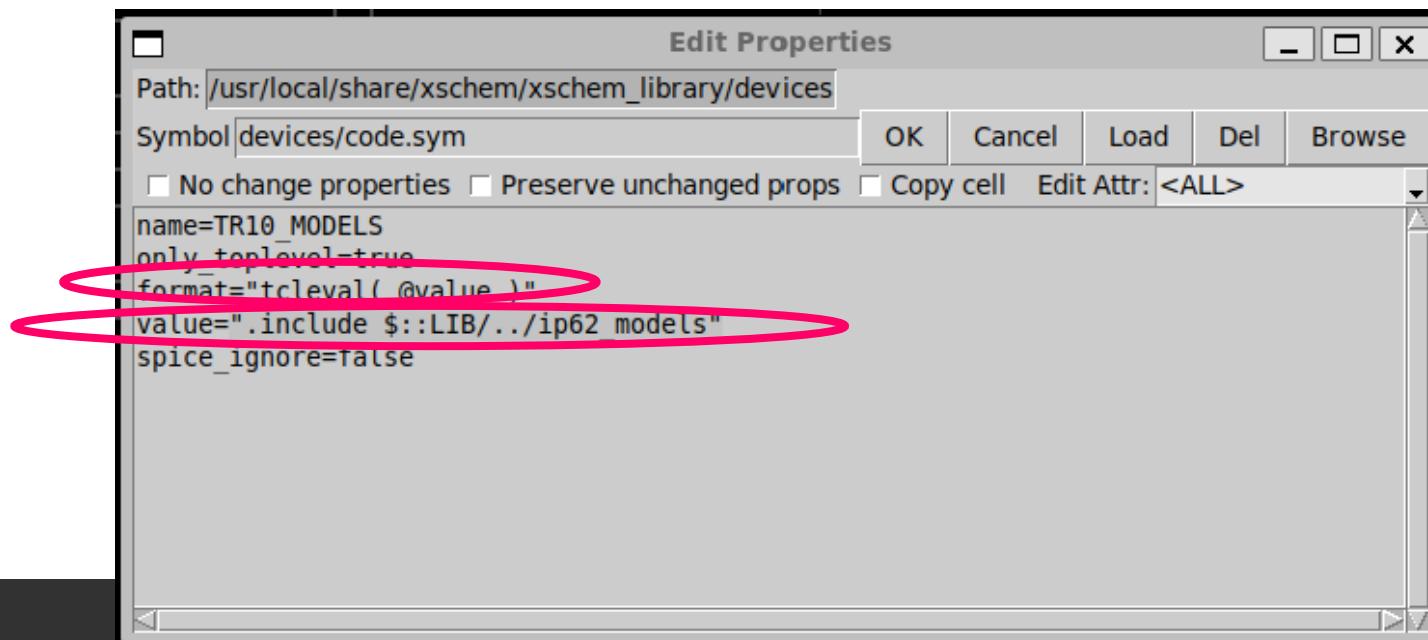
xschem : ベンチマーク



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

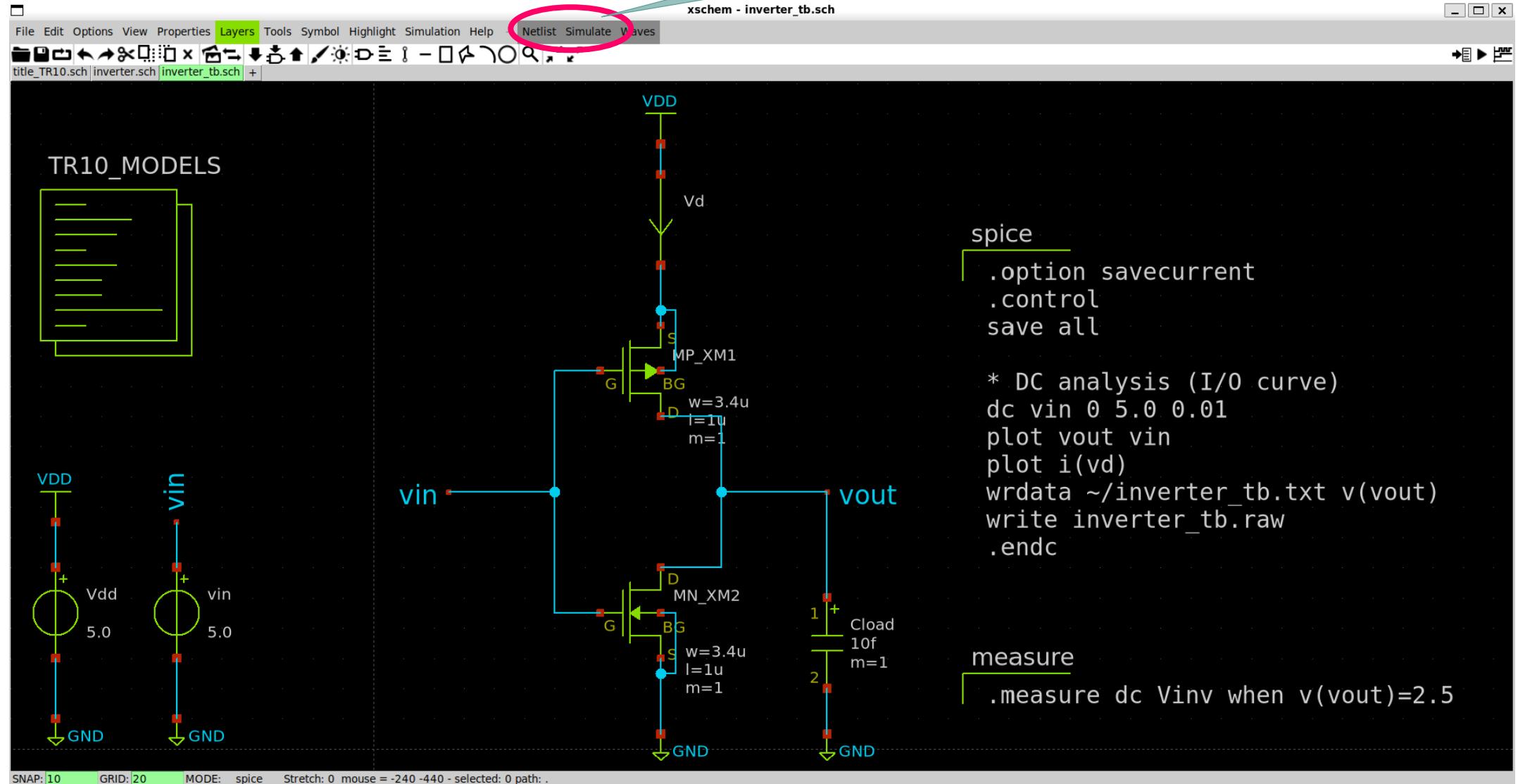
xschem : TR10_MODELSを記述する際の注意点×

- code.symを使用する際、以下の行を追記する必要がある
 - format="tclevl(@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。
 - 「\$::LIB」は特殊パスのため利用可能

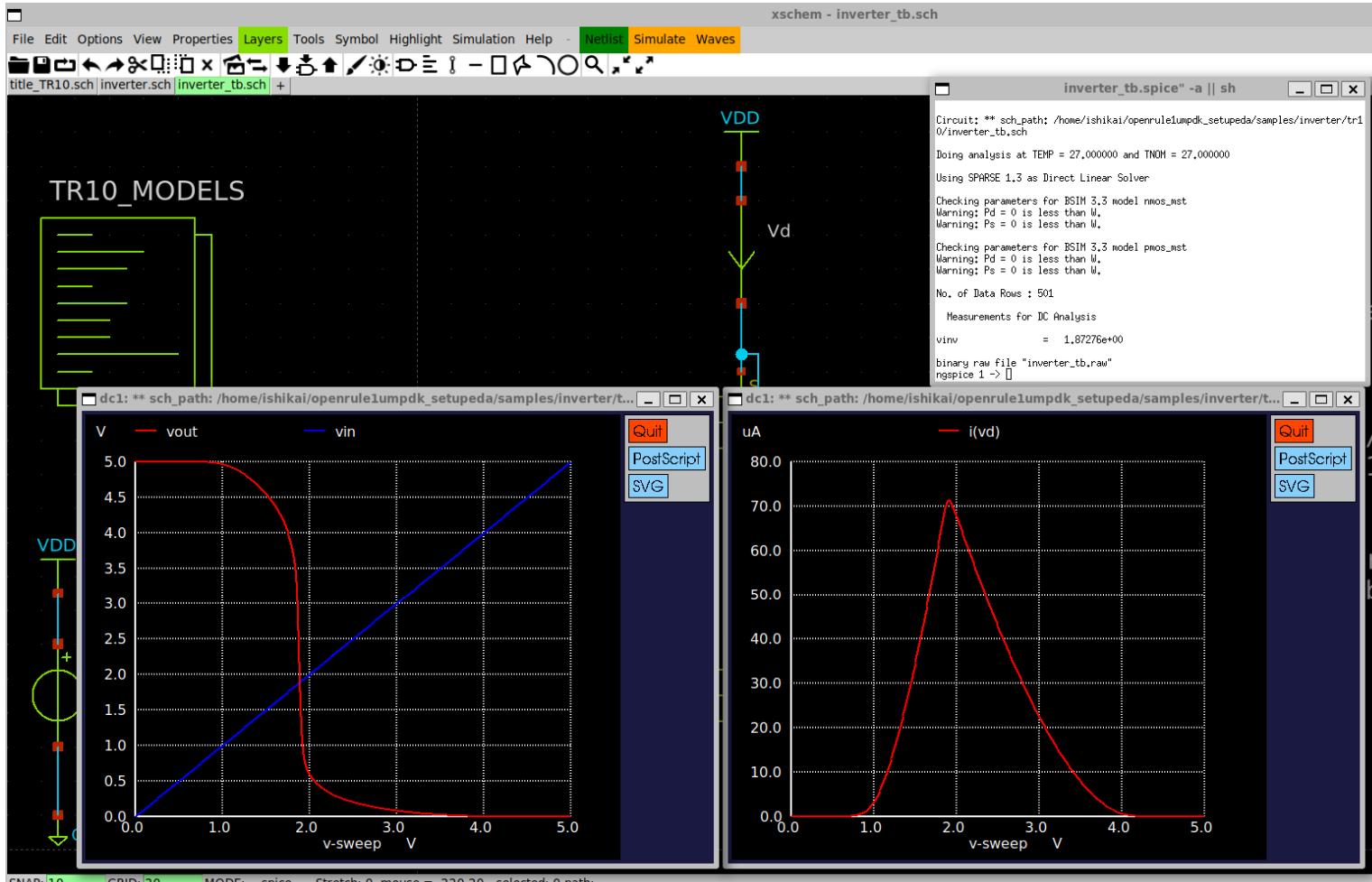


ngspice : シミュレーションの実行

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

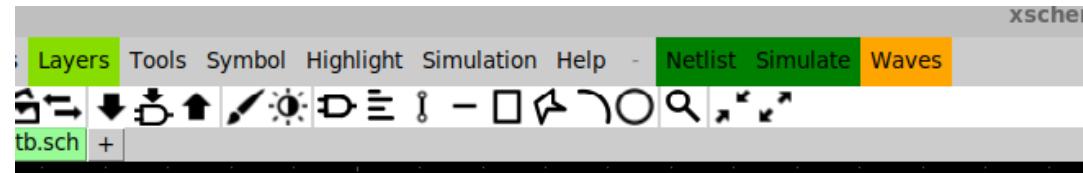
wrdataを使用するとテキスト出力ができる

```
0.0000000e+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 3.2999960e+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
```

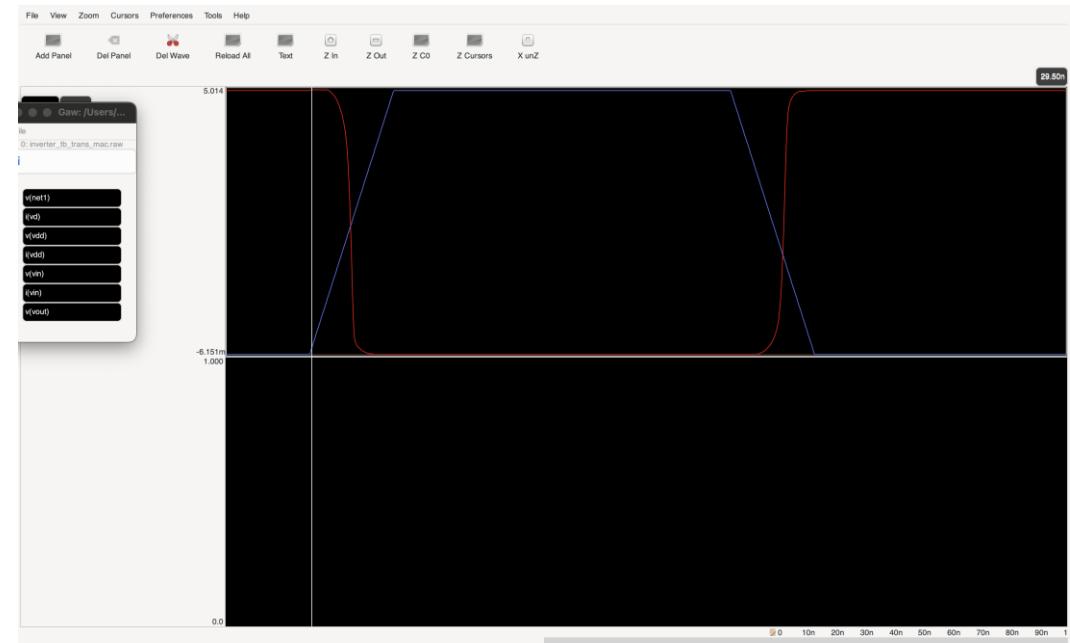
▪

▪

▪



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

xschem - inverter_tb_trans.sch

File Edit Options View Properties Layers Tools Symbol Highlight Simulation Help Netlist Simulate Waves

title_TR10.sch|inverter.sch|inverter_tb.sch|inverter_tb_trans.sch +

TR10_MODELS

VDD Vdd 5.0 GND

vin vin pwl 0 0 10n 0 20n 5.0 60n 5.0 70n 0 GND

vin

MP_XM1 MP_XM2

MP_XM1: Gate (G), Source (S), Drain (D), Body (BG)

MP_XM2: Gate (G), Source (S), Drain (D), Body (BG)

W: 3.4u, L: 1u, M: 1

Cload: 10f, M: 1

VDD Vd

vout

spice

```
.option savecurrent
.control
save all

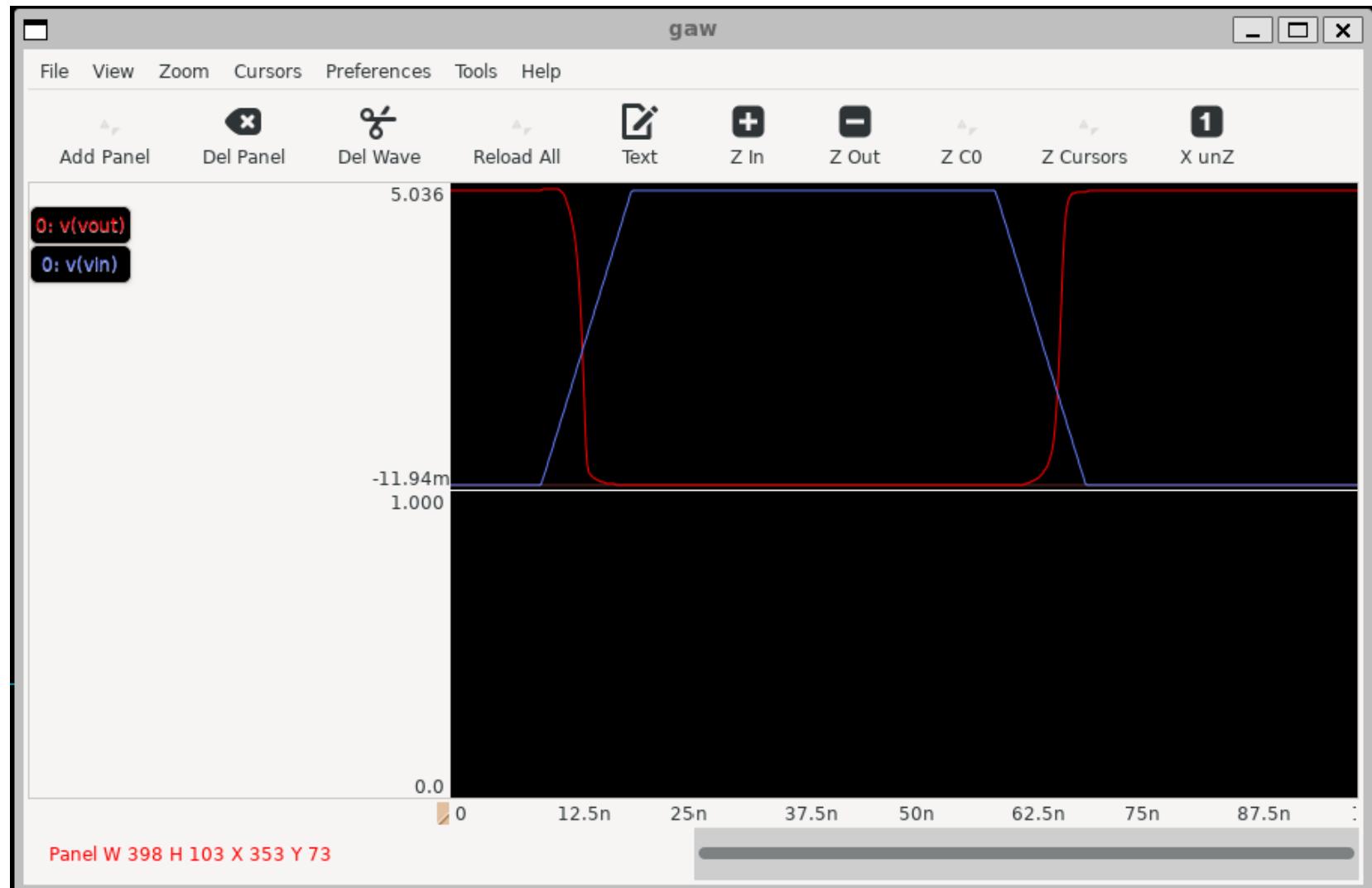
* Tran analysis
tran 0.1n 100n
plot vout vin
plot i(vd)
wrdata ~/inverter_tb_tran.txt v(vout)
write inverter_tb_trans.raw
.endc

measure

.measure tran td_r trig v(vin) val=2.5 fall=1 targ v(vout) val=2.5 rise=1
.measure tran td_f trig v(vin) val=2.5 rise=1 targ v(vout) val=2.5 fall=1
.measure tran trise trig v(vout) val=0.83 rise=1 targ v(vout) val=4.17 rise=1
.measure tran tfall trig v(vout) val=4.17 fall=1 targ v(vout) val=0.83 fall=1
```

SNAP: 10 GRID: 20 MODE: spice Stretch: 0 mouse = -250 -410 - selected: 0 path: .

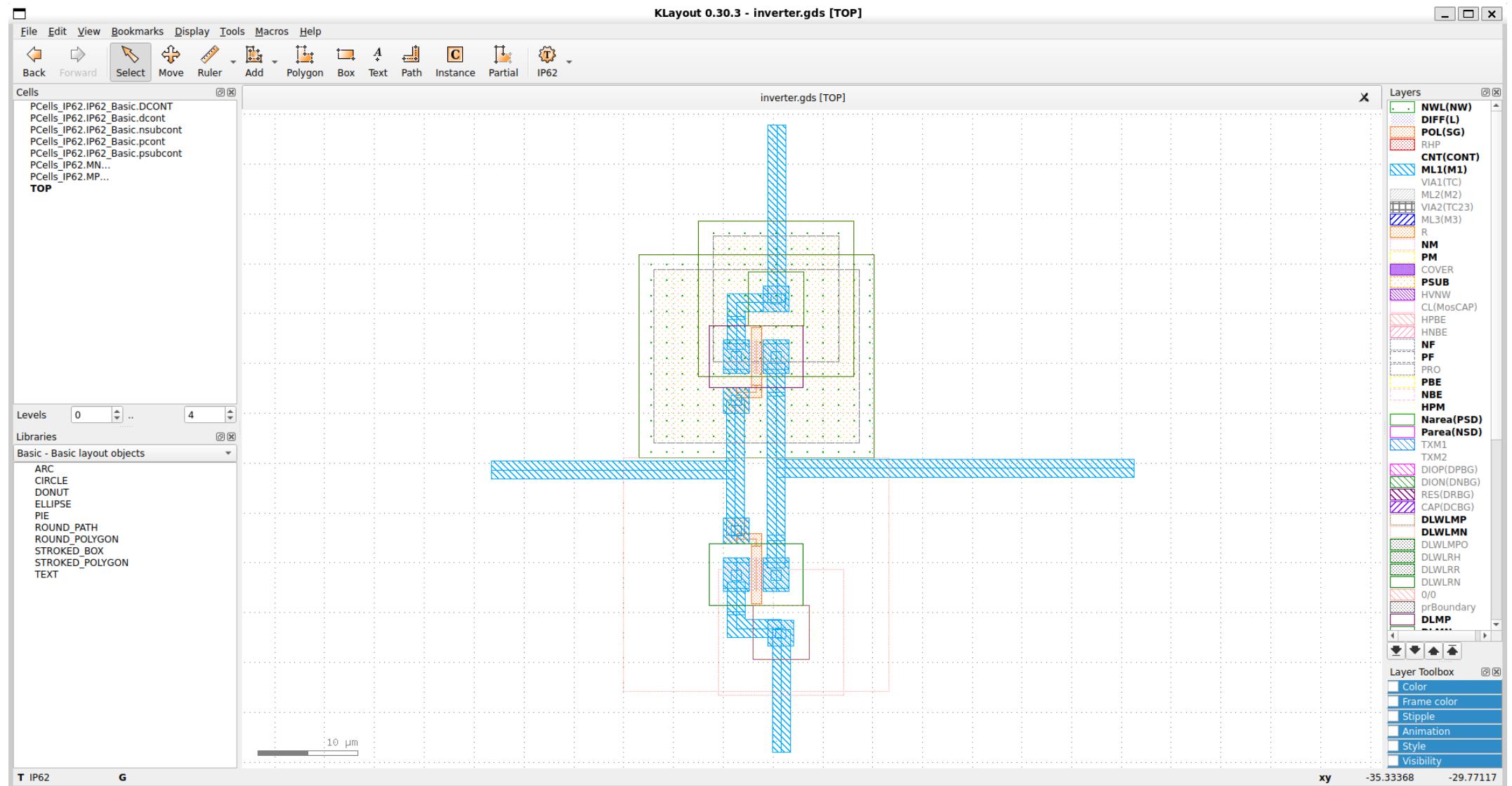
ngspice : 過渡解析シミュレーション結果



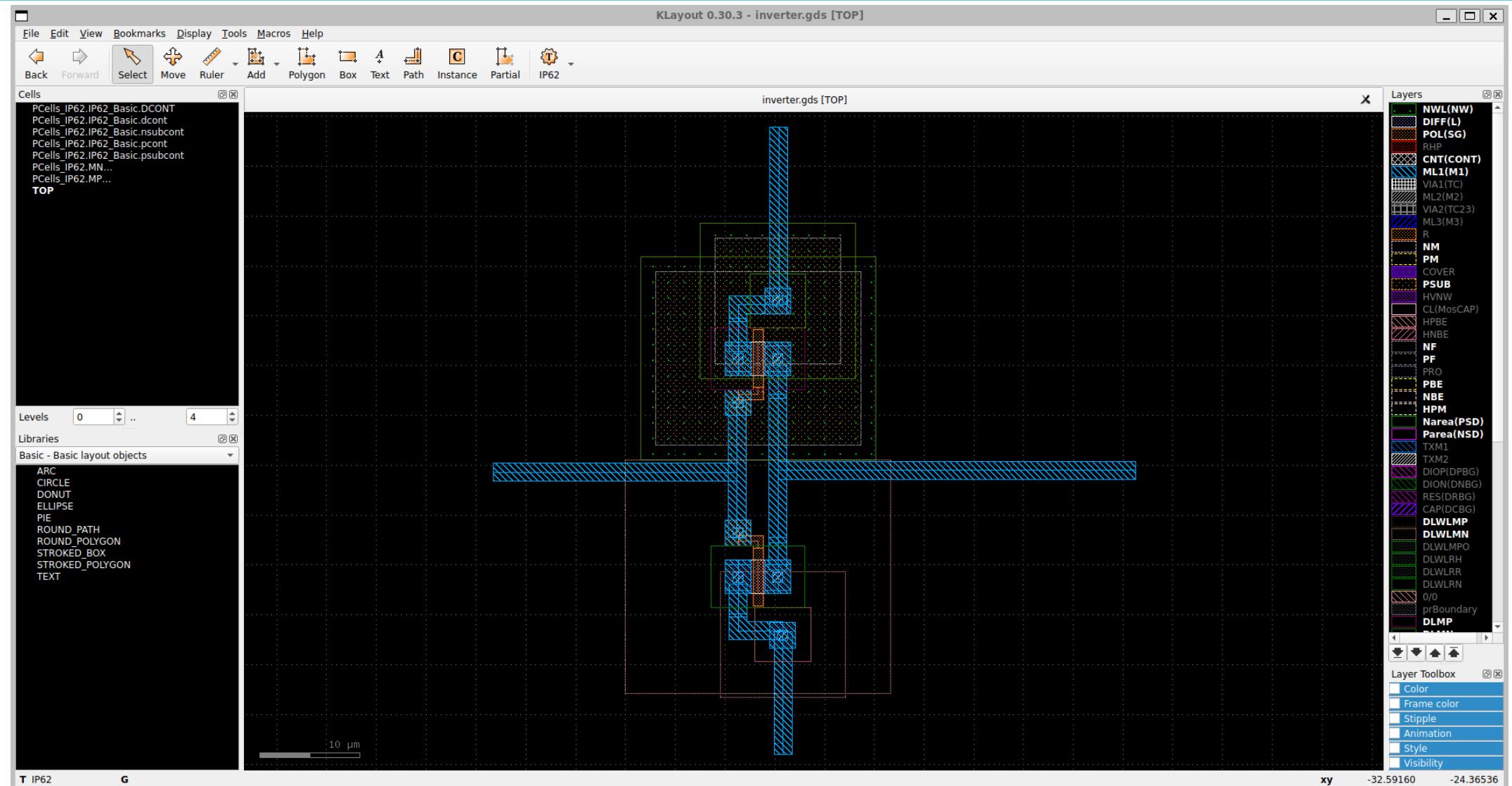
アナログLSIの設計フロー

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)

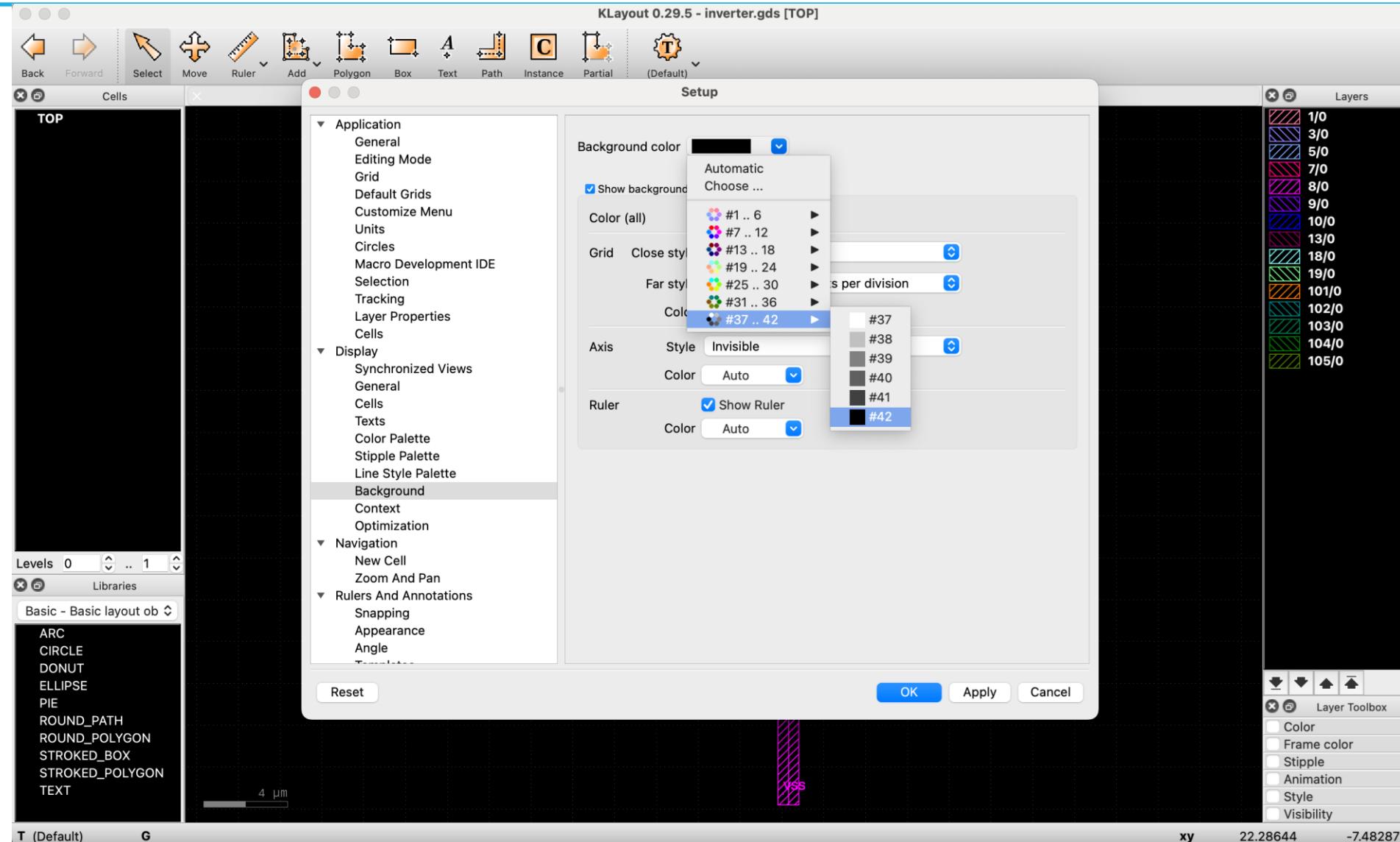
Klayout : 公式PDKのレイアウト画面



Klayout : バックグラウンドが「黒」のレイアウト画面

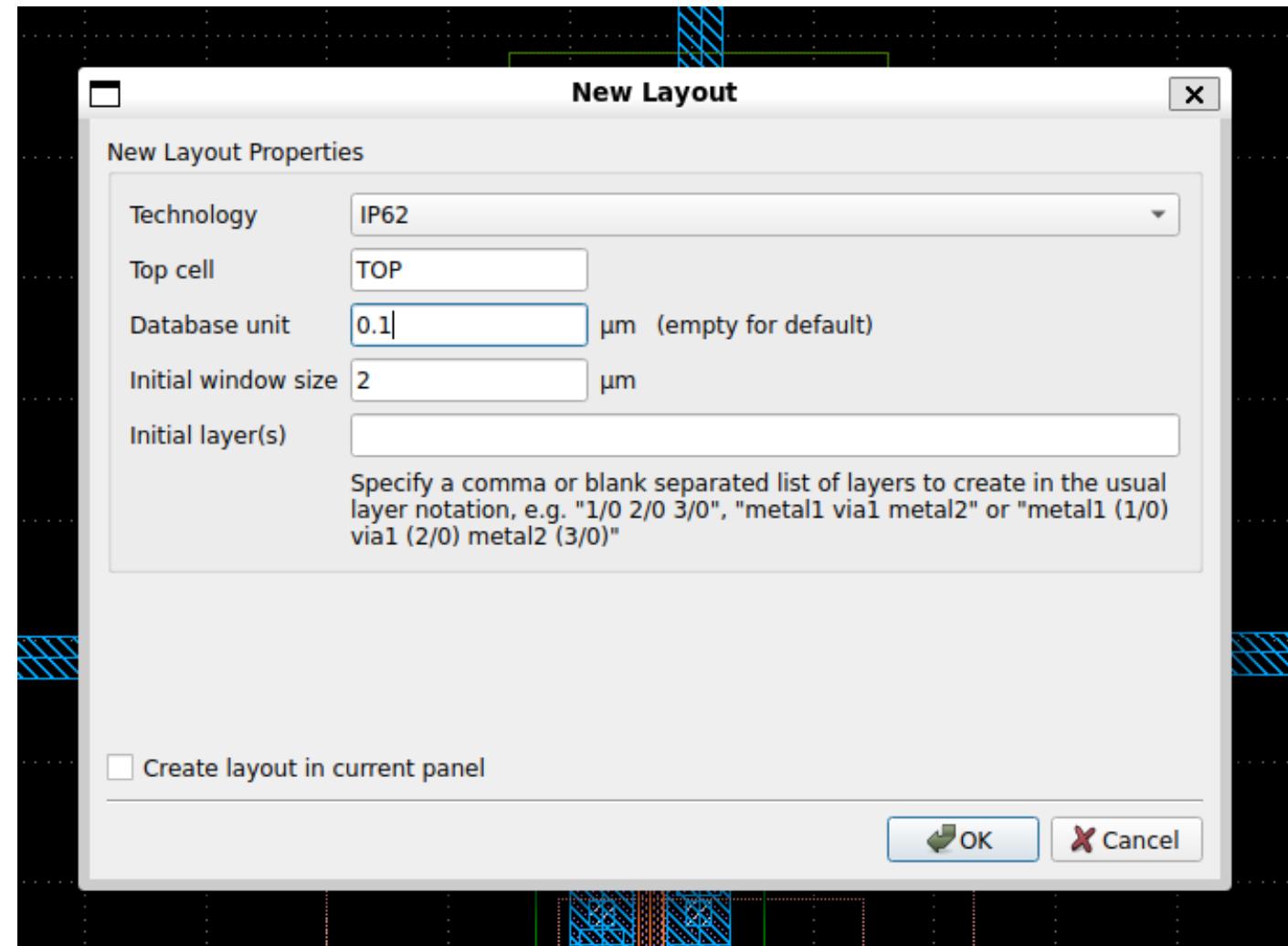


KLayout : バックグラウンドを「黒」にする



Klayout : レイアウトを描く上での注意

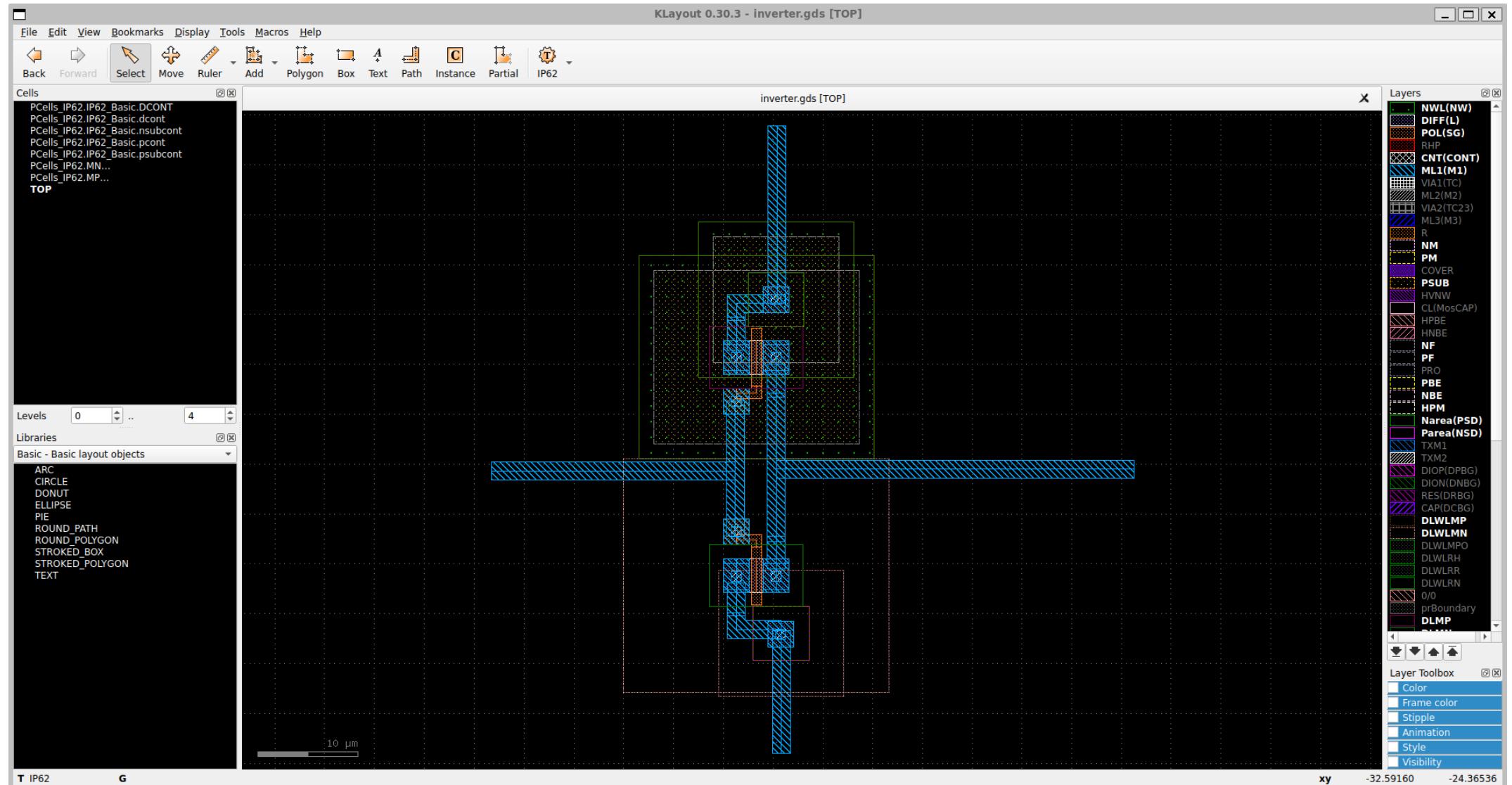
- Technologyを IP62にする
- Database unit を 0.1にする



Klayout : デザインルールの場所

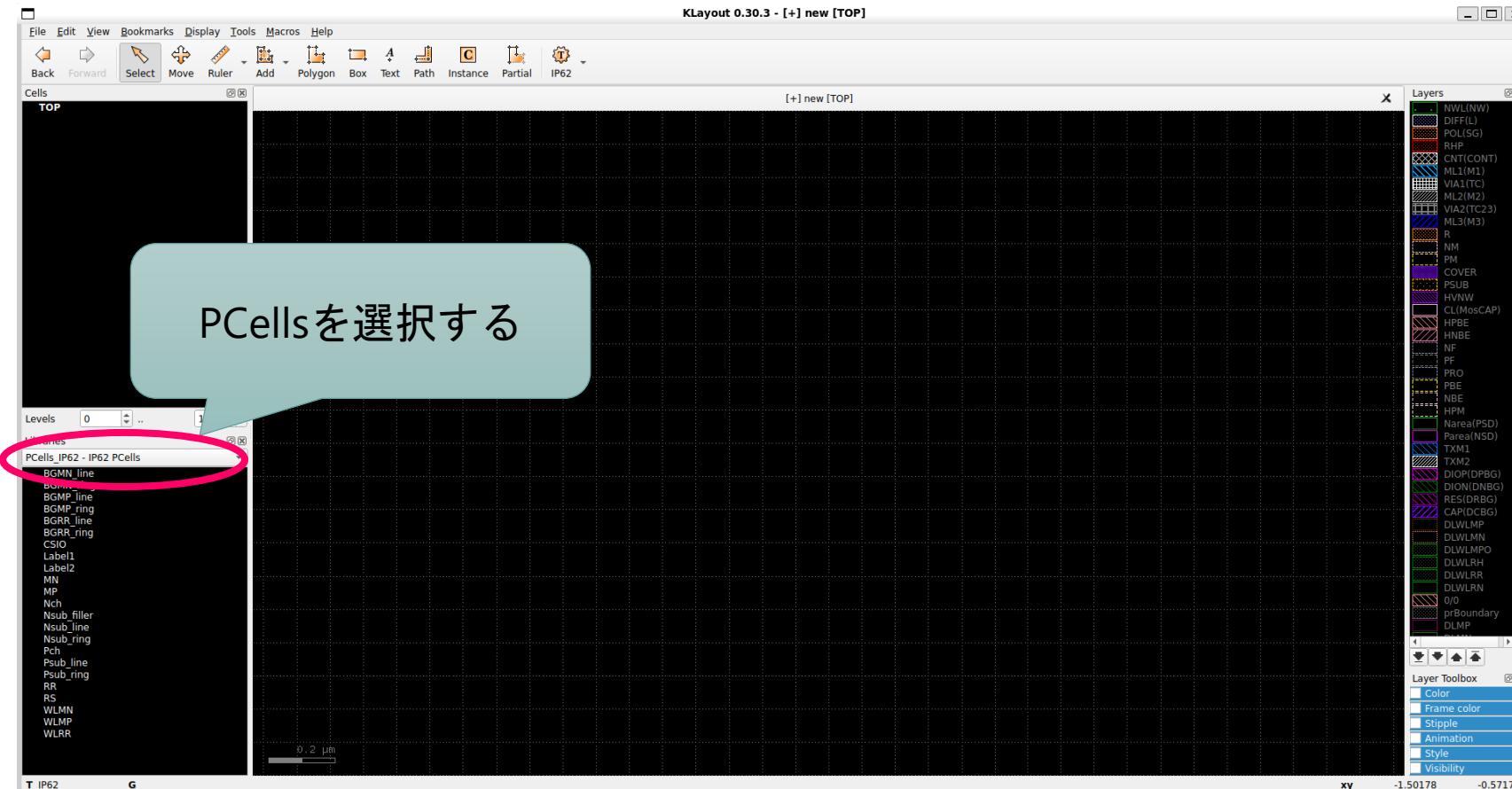
- ~/src/OpenIP62/IP62/Technology/doc
 - デザインルールマニュアル
 - OS00_リファレンスマニュアル.pdf
 - 必要に応じて参照すること
 - OS01_インストールマニュアル.pdf
 - OS02_回路simガイドライン.pdf
 - OS03_レイアウト検証ガイドライン.pdf
 - OS04_ESD保護素子ガイドライン.pdf
 - OS05_スタンダードセルラインナップ.pdf
 - OS06_素子接続ガイドライン.pdf
- ~/src/OpenIP62/IP62/Tools
 - 抵抗・容量計算ツール(オープン版).xlsx
 - 抵抗やコンデンサを利用する場合に容量計算する

Klayout : 完成サンプル



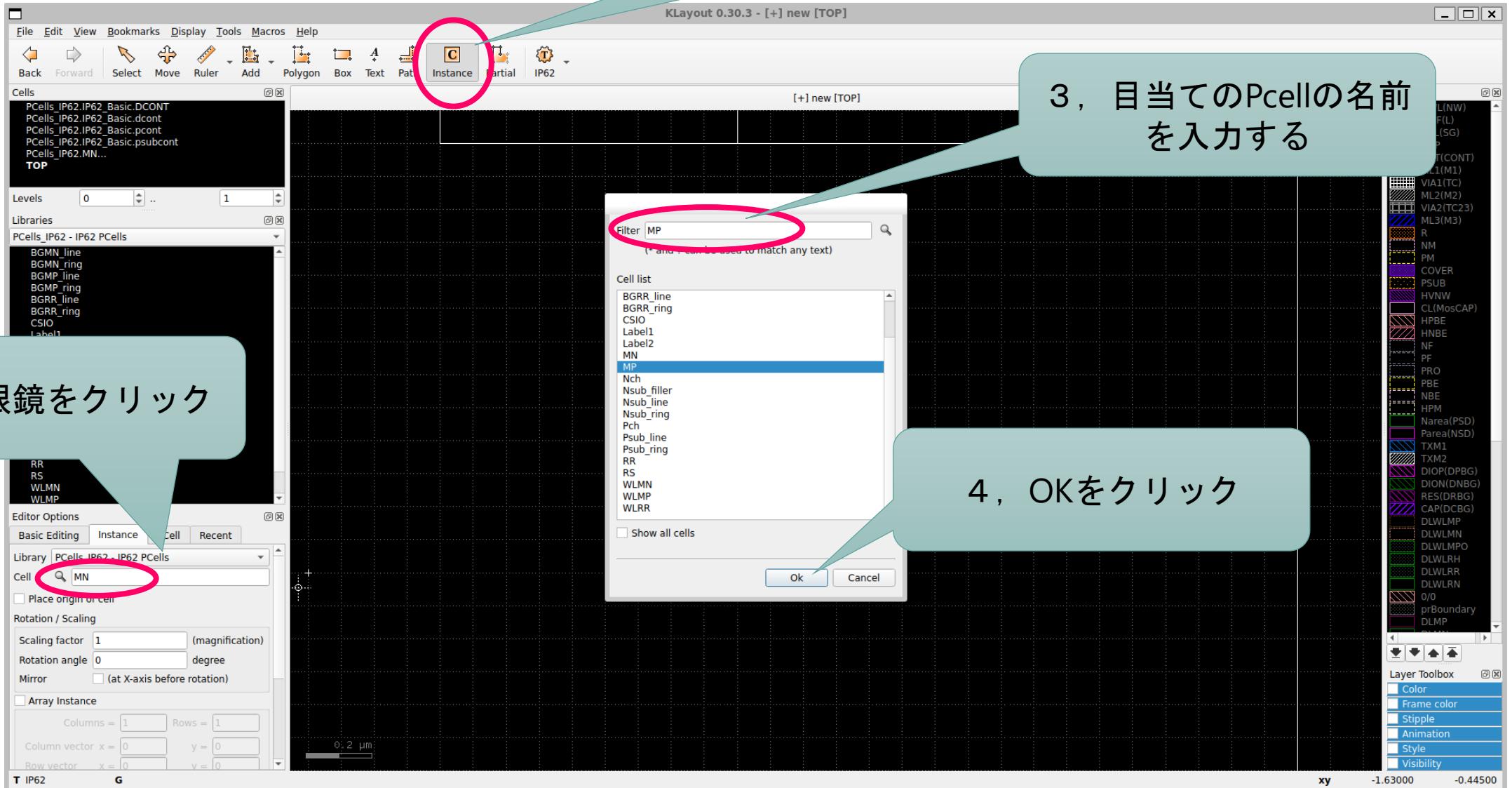
KLayout : P-Cellの使い方

- 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

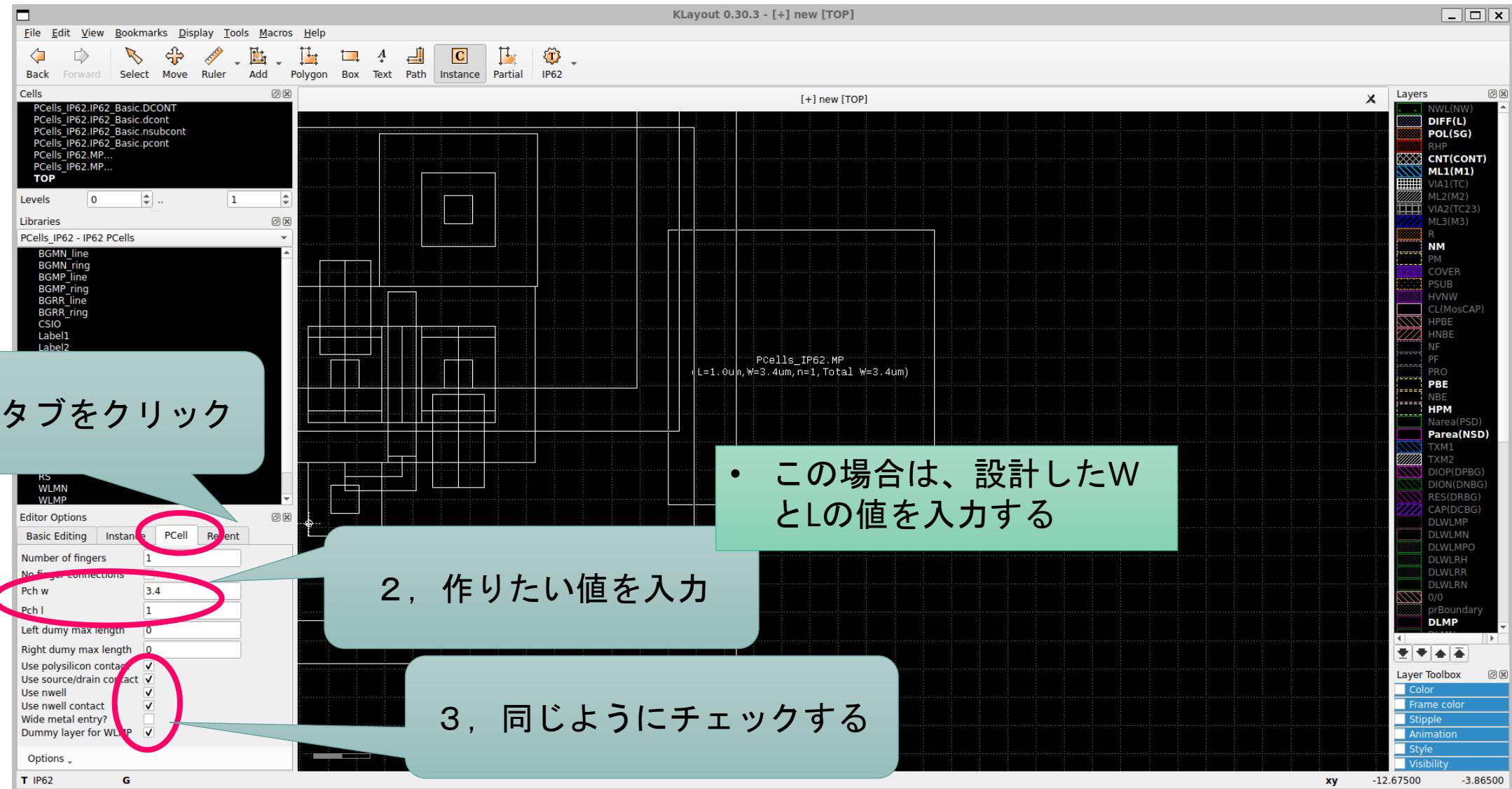


2, 虫眼鏡をクリック

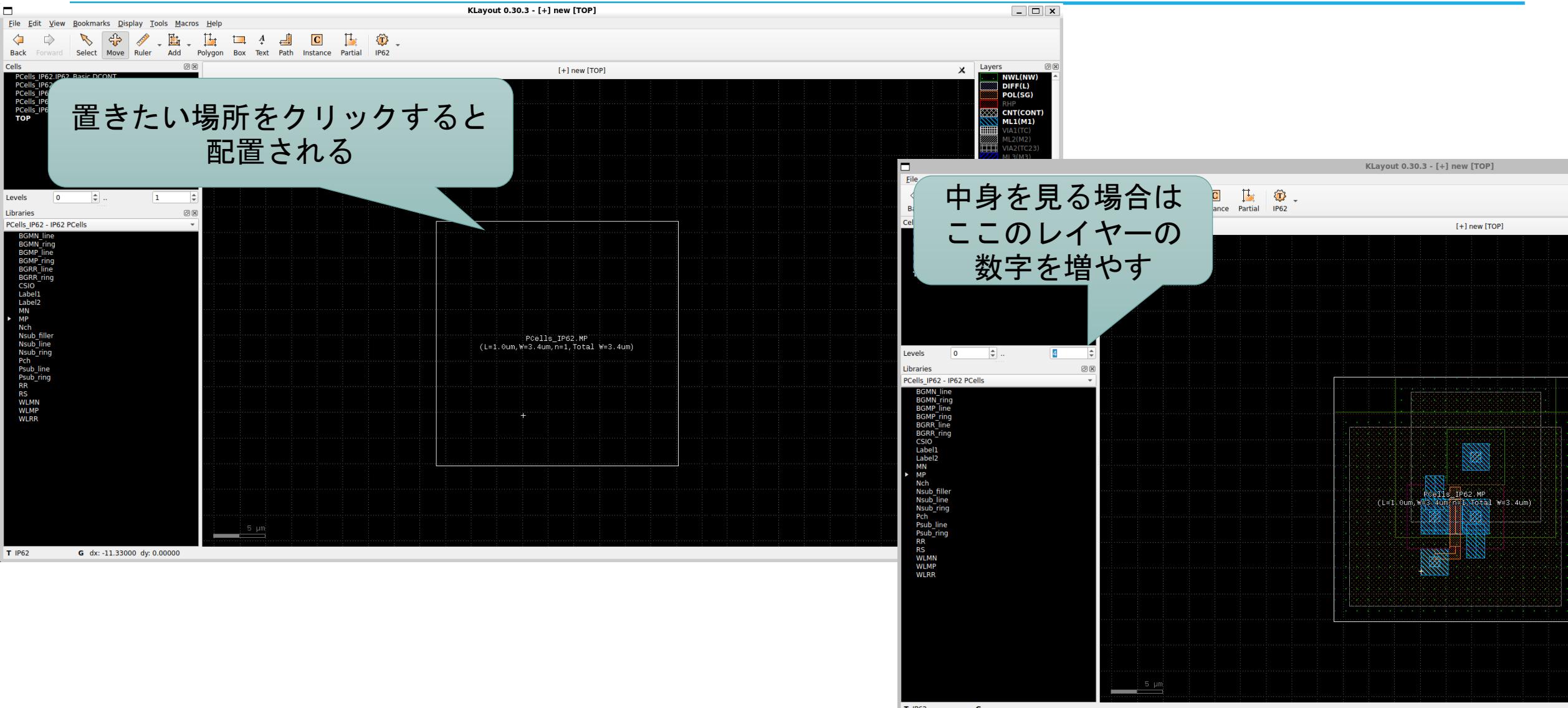
3, 目当てのPcellの名前
を入力する

4, OKをクリック

Klayout : P-Cellの使い方

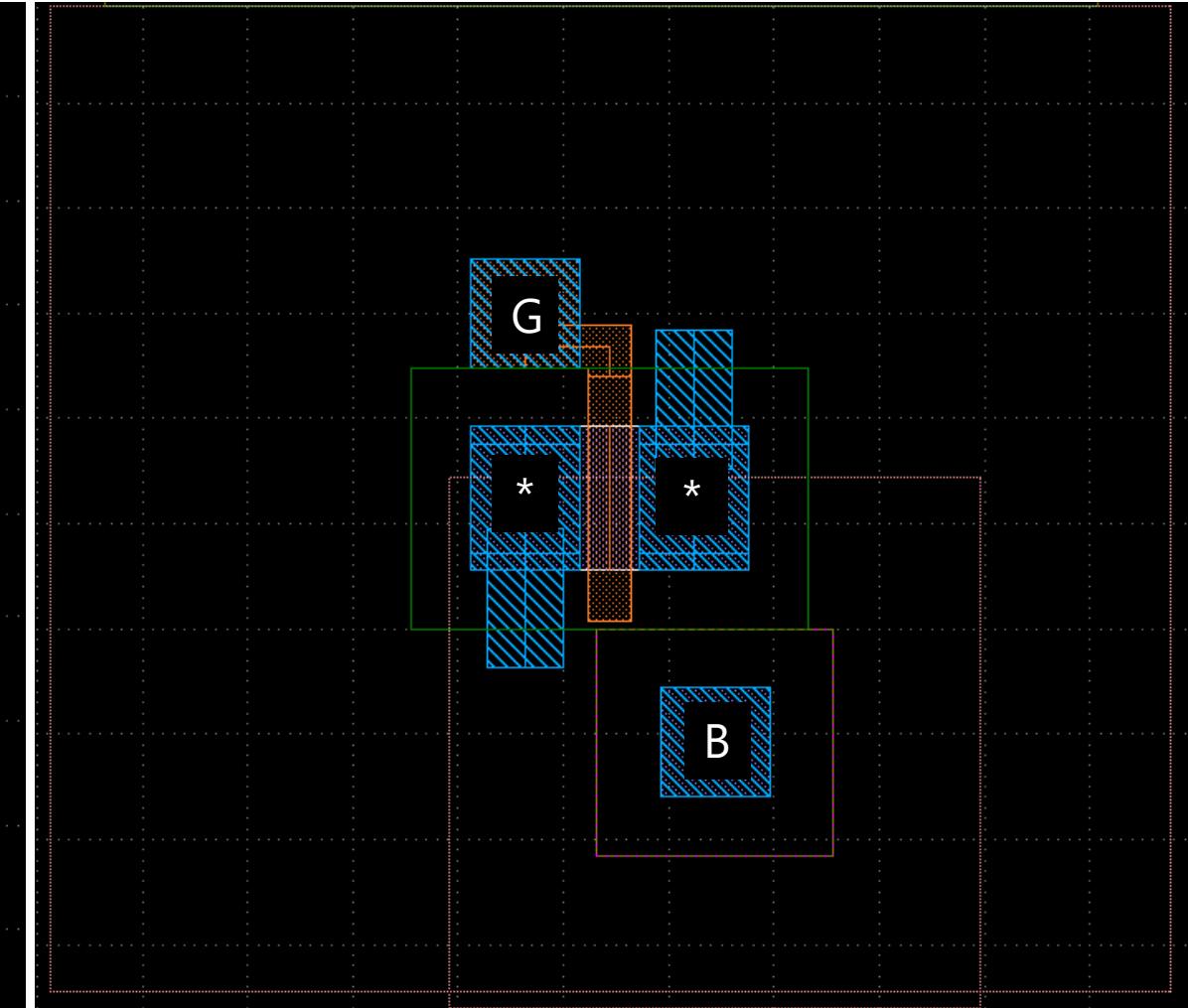
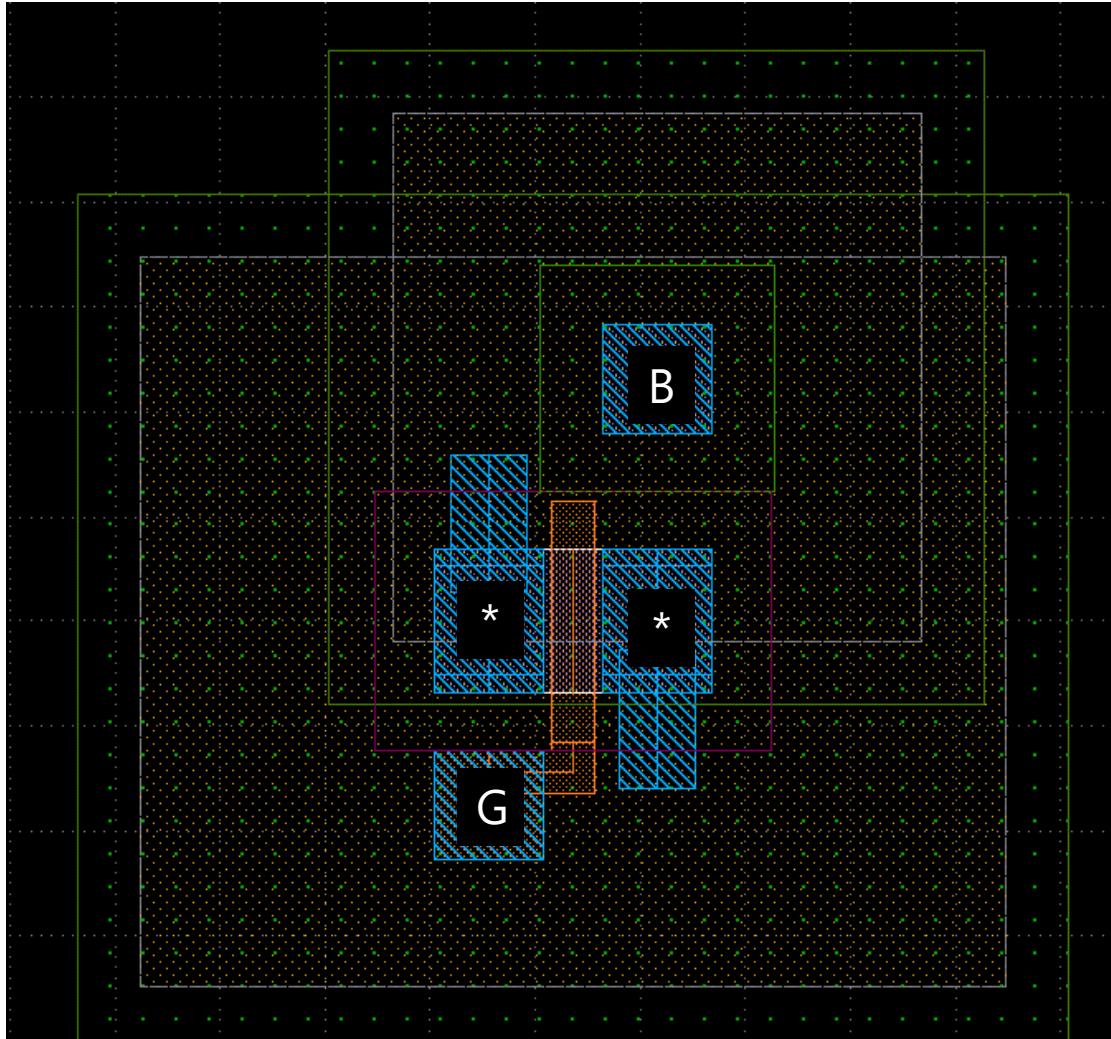


KLayout : P-Cellの使い方



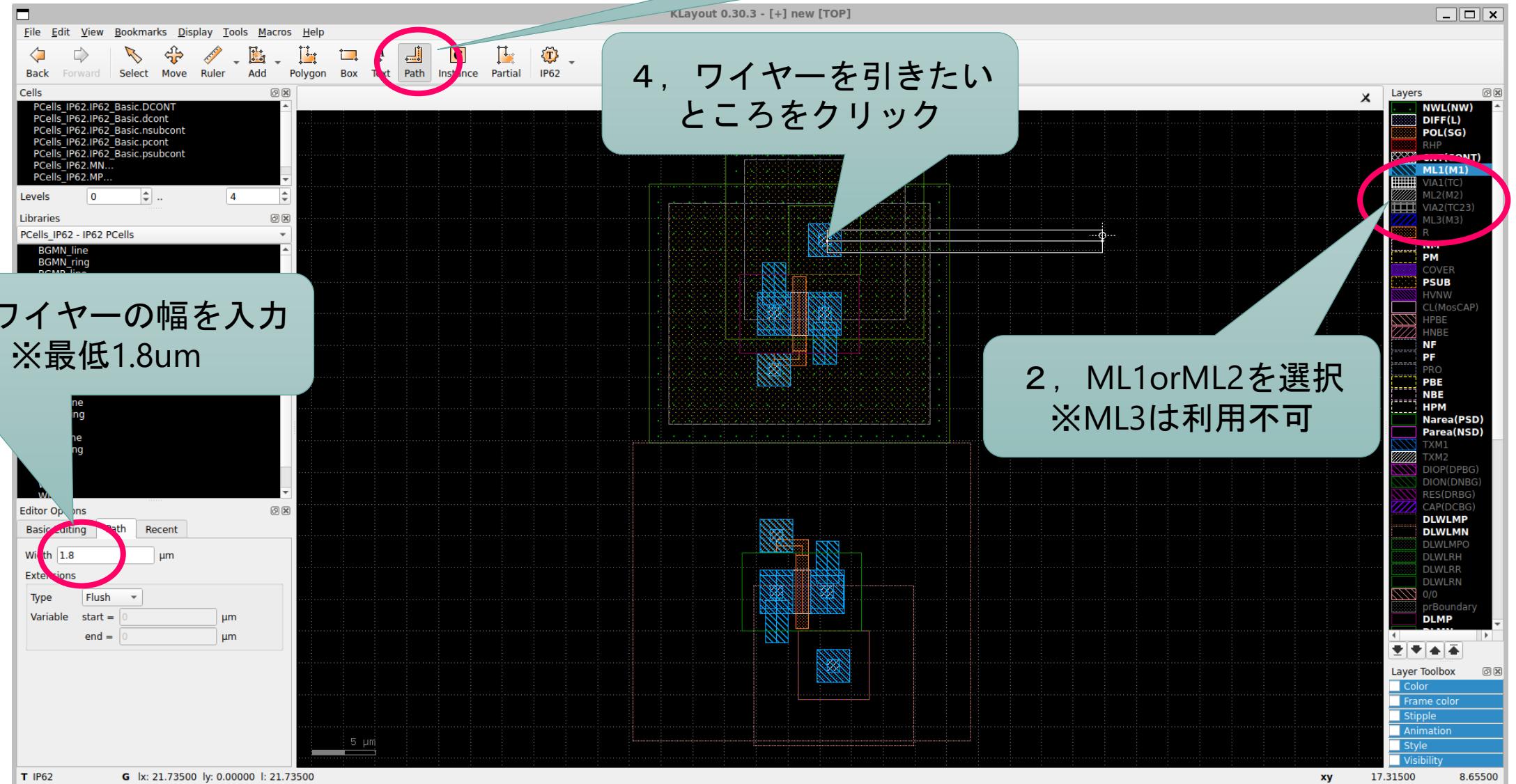
Klayout : タップ付 PFET と NFET

- [*] 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。

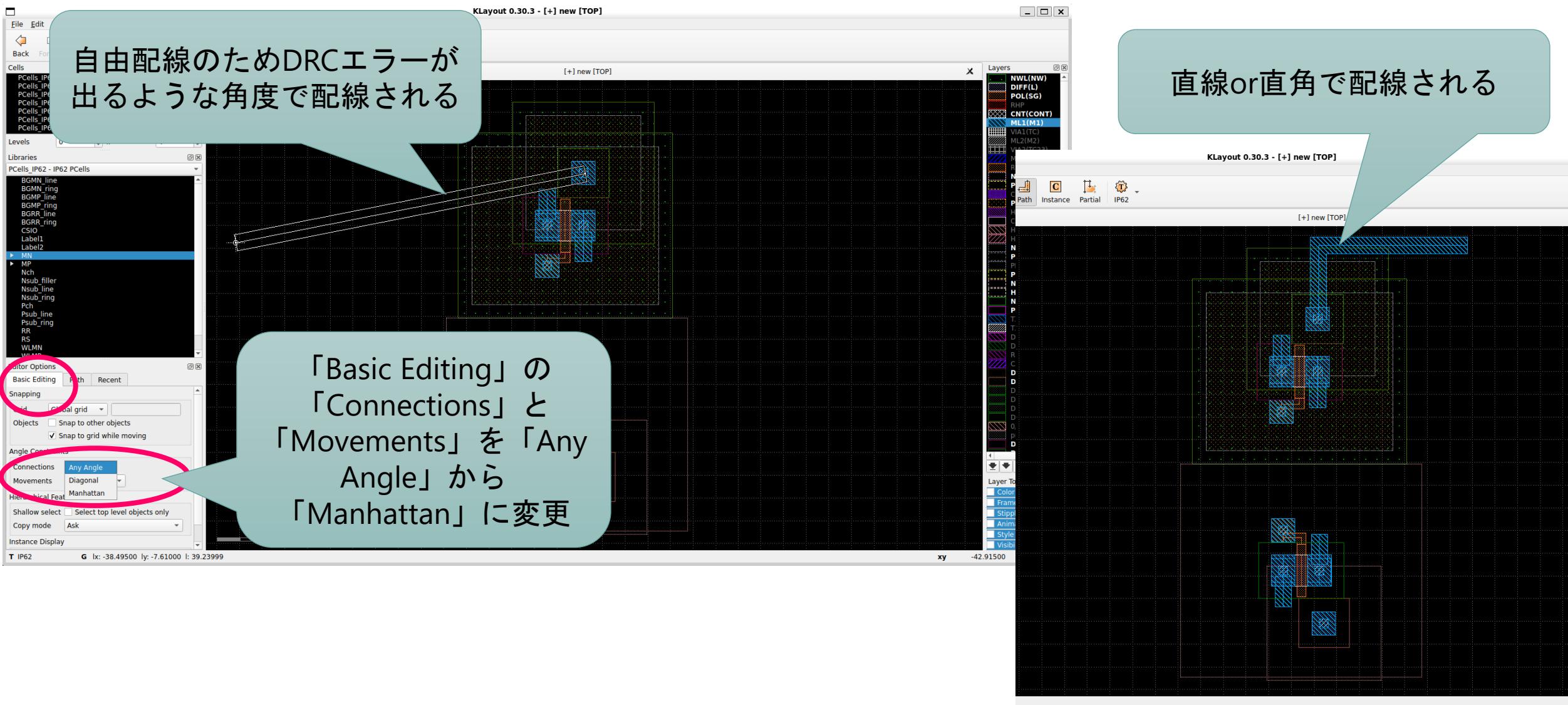


KLayout : ネットリスト（ワイヤー）の引き方

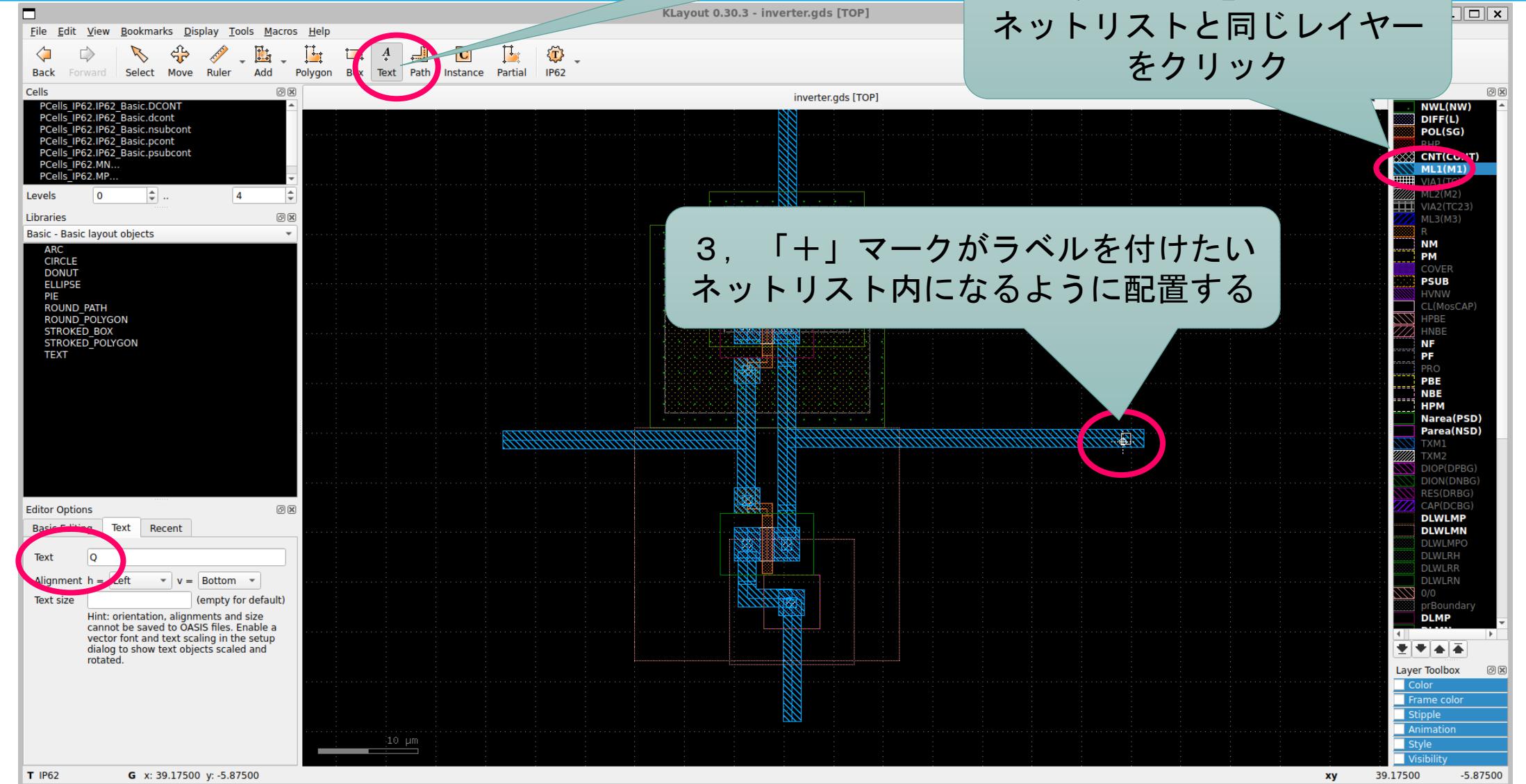
1, Pathをクリック



KLayout : ネットリスト（ワイヤー）の設定



Klayout : ラベルの付け方



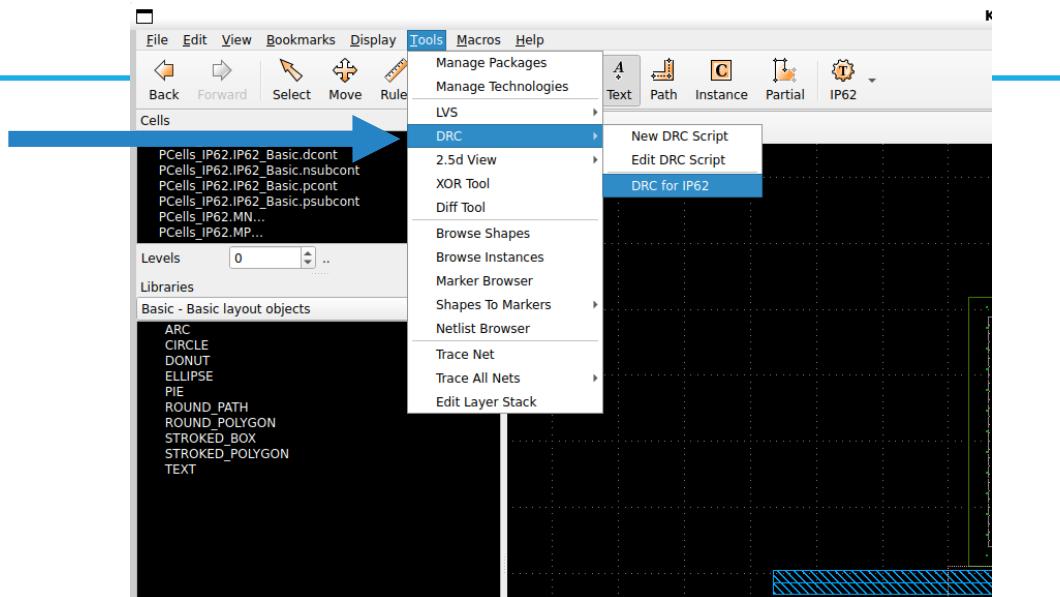
アナログLSIの設計フロー

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

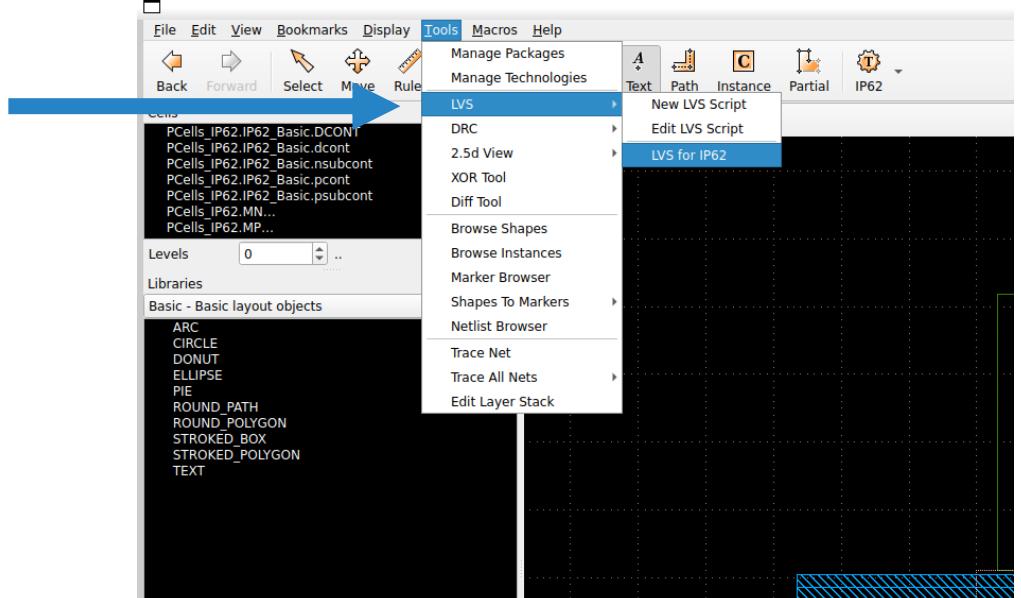
Design Rules Check (DRC)

- 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)

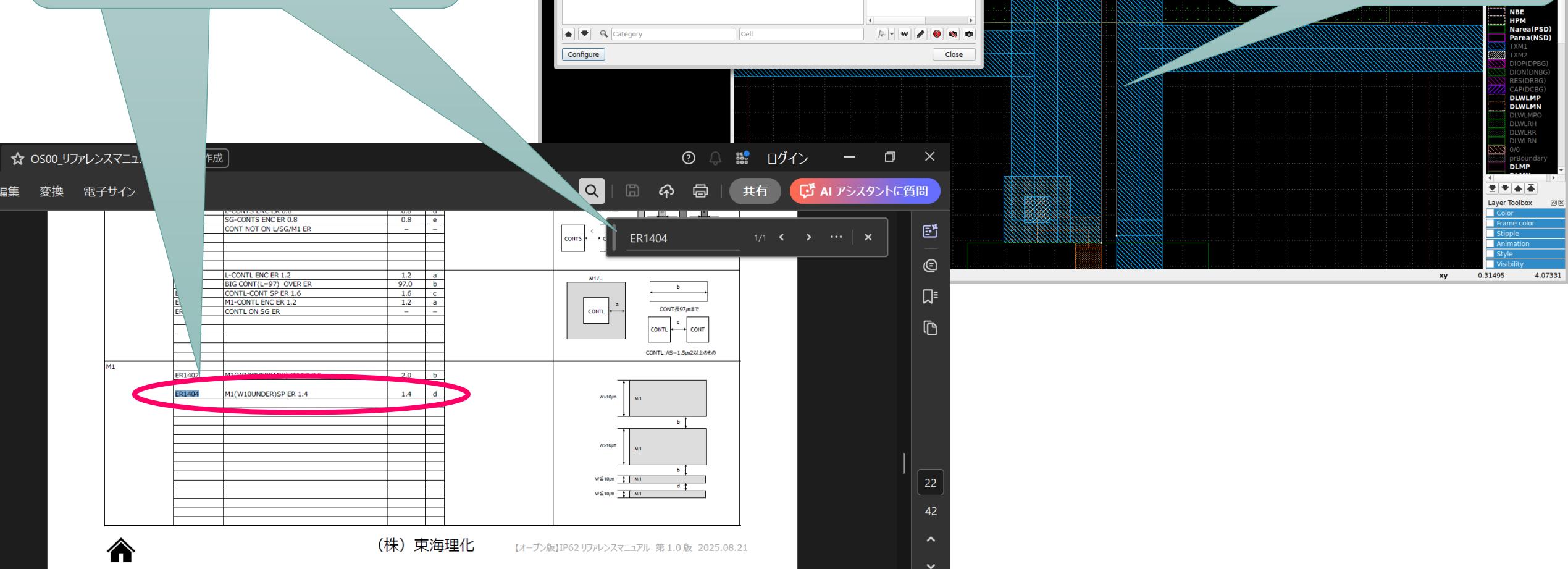
- レイアウトが回路図通りに描けているか検証



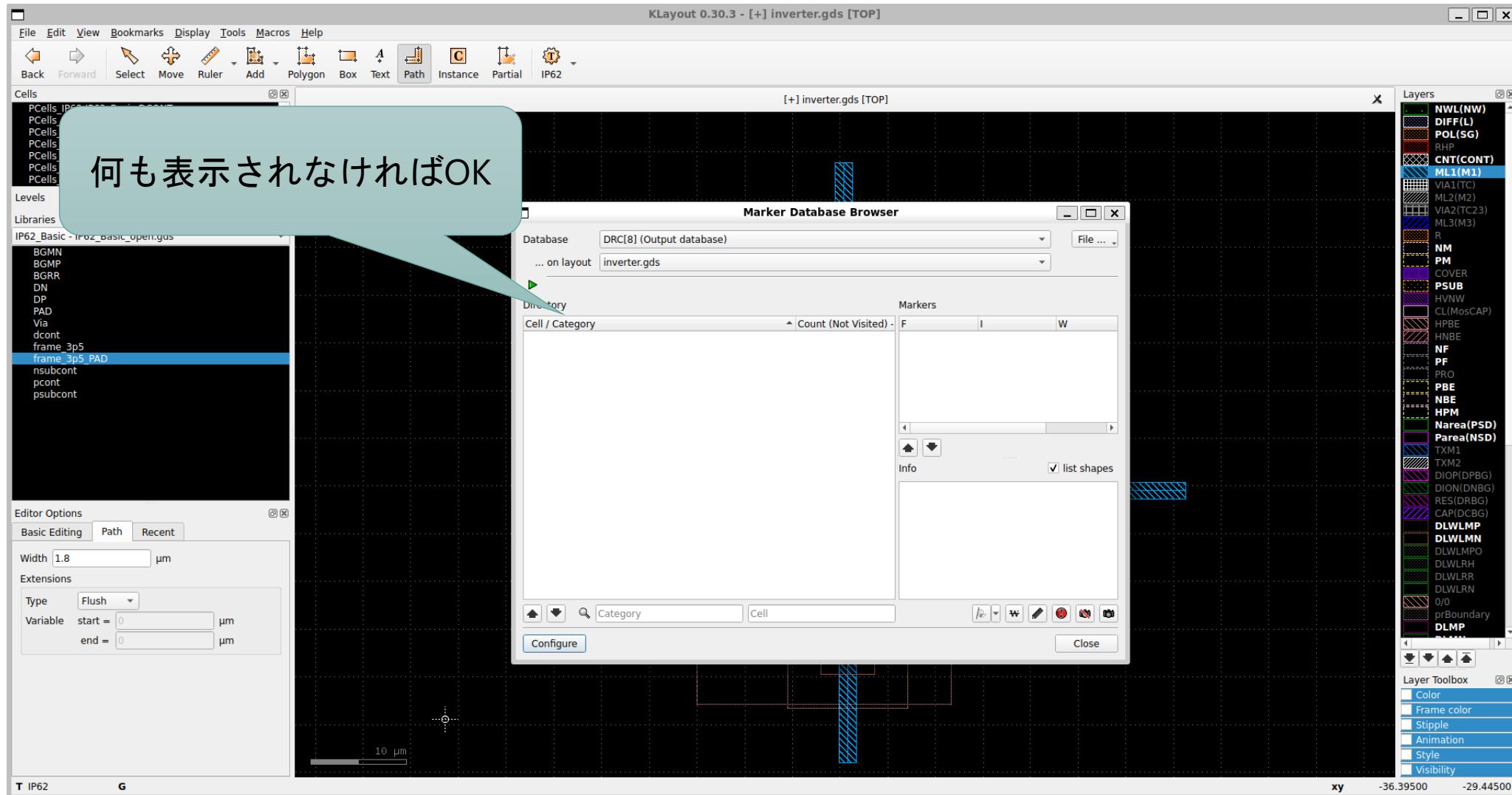
Klayout : DRCエラー

エラー番号をレファレンスマニュアルで検索すると
エラー内容が解説されている

旗をクリックすると
白いでエラー箇所が
強調される

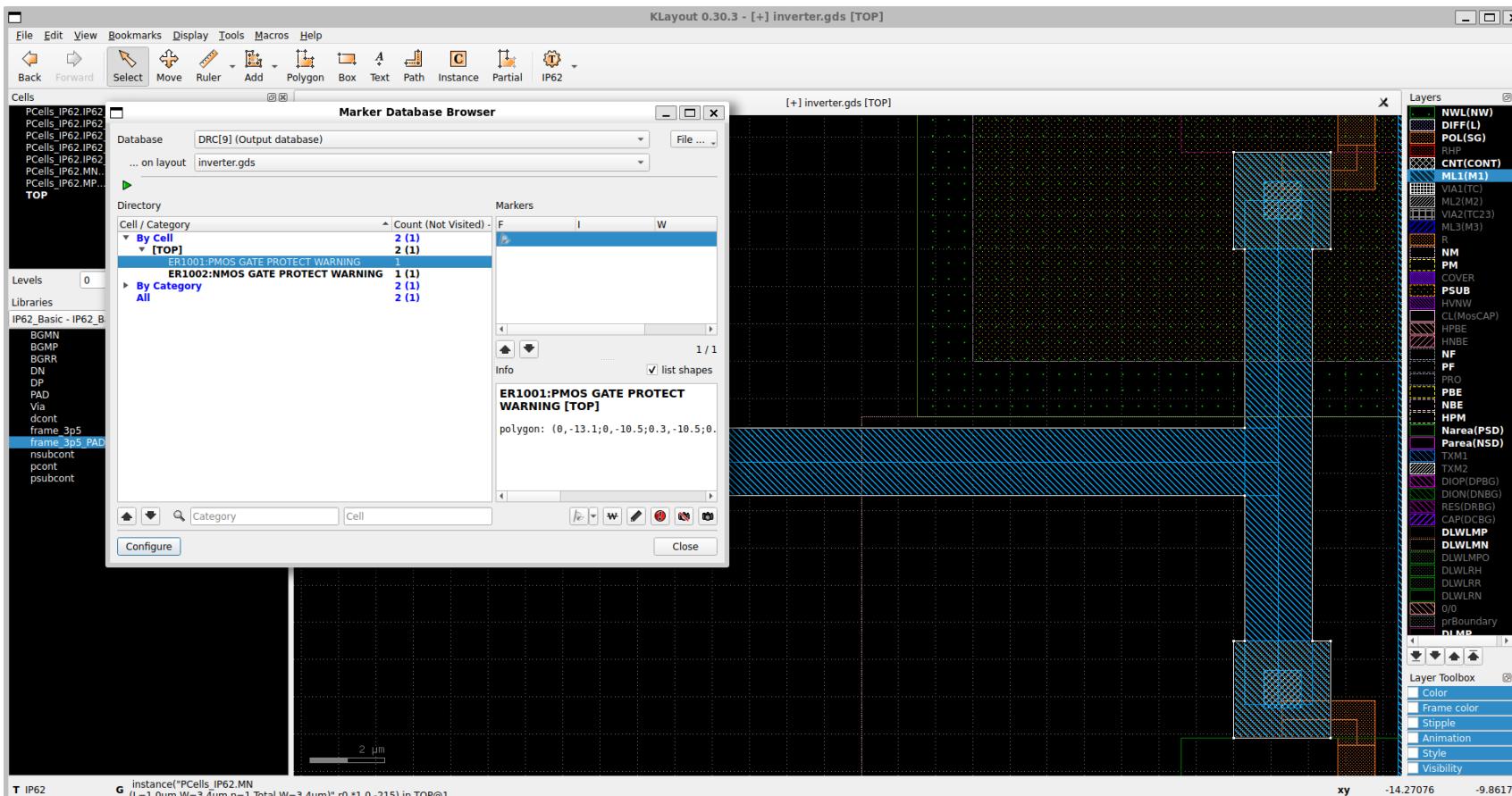


Klayout : DRC OK



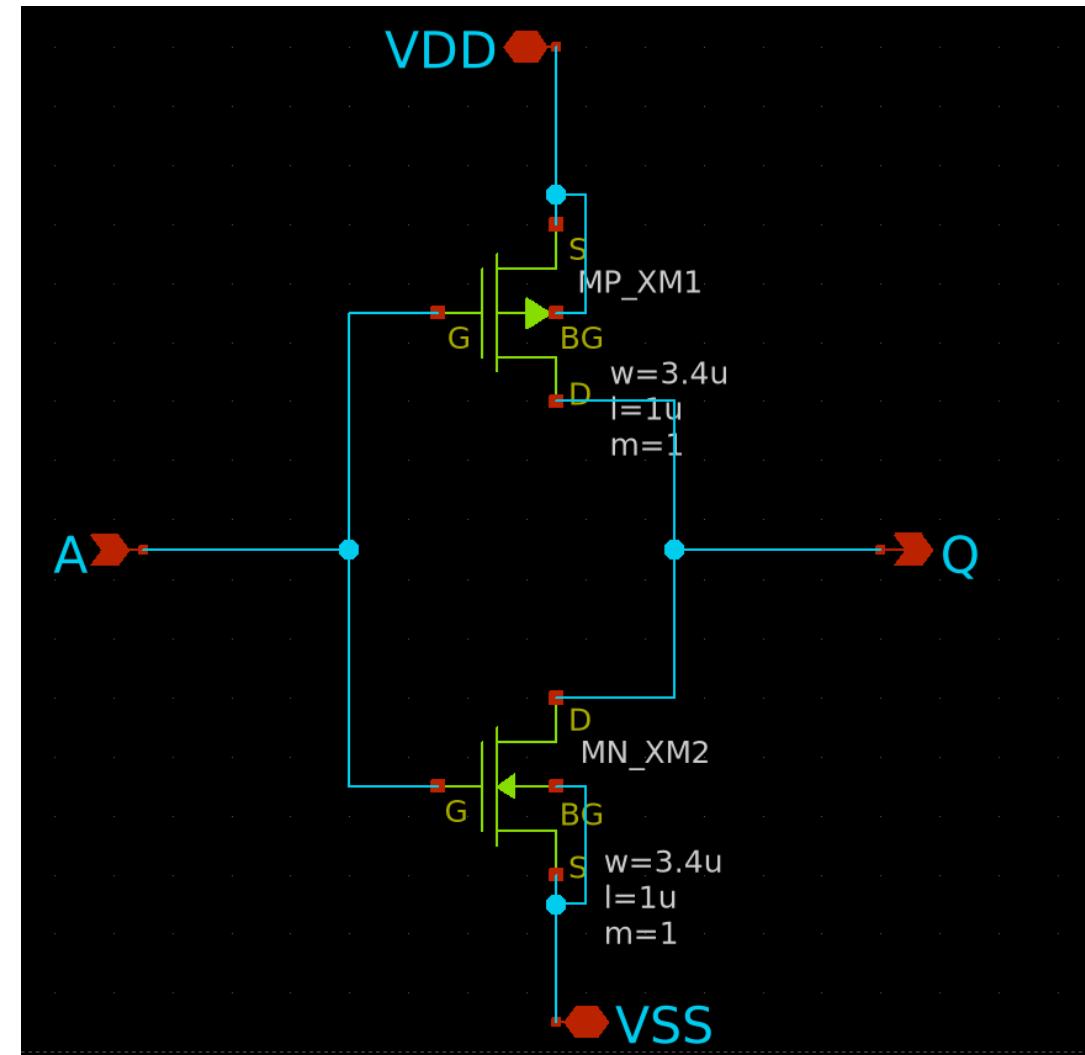
Klayout : DRC WARNING

- ER1001, ER1002のWARNING
 - アンテナルルエラーのため無視してもよい



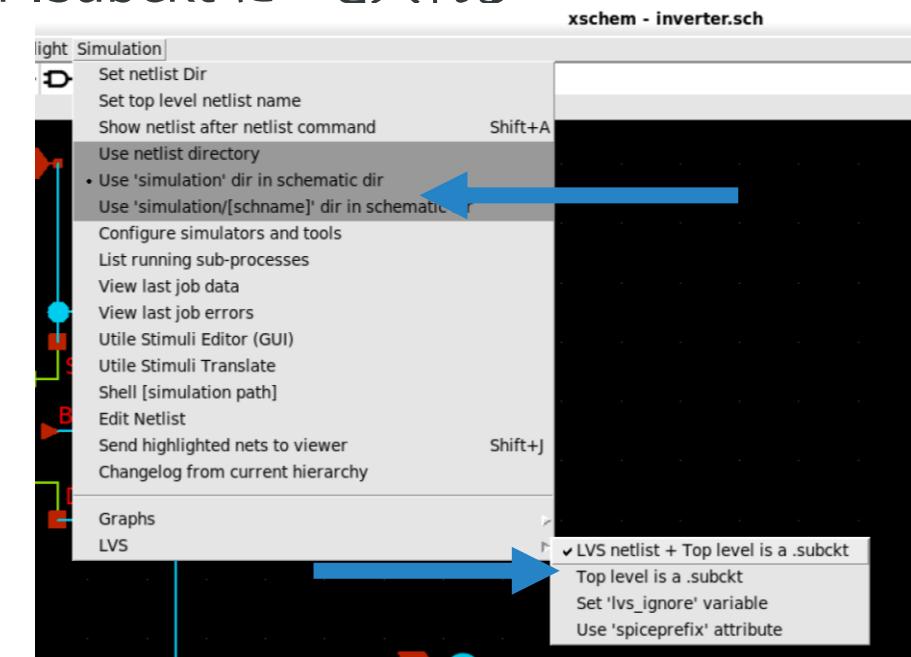
Klayout : LVS用ネットリストの出力方法 (1/3)

- 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 (2/3)

- xschem上で**LVS用**のネットリストを出力する。
 1. メニュー Simulation > Use 'simulation' dir in schematic dir に✓を入れる
 2. メニュー Simulation > LVS > Use 'spiceprefix' attribute の✓を外す
 3. メニュー Simulation > LVS > LVS netlist + Top level is a .subckt に✓を入れる
 4. 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実行Dir]/simulation/



Klayout : LVS用ネットリストの出力方法 (3/3)

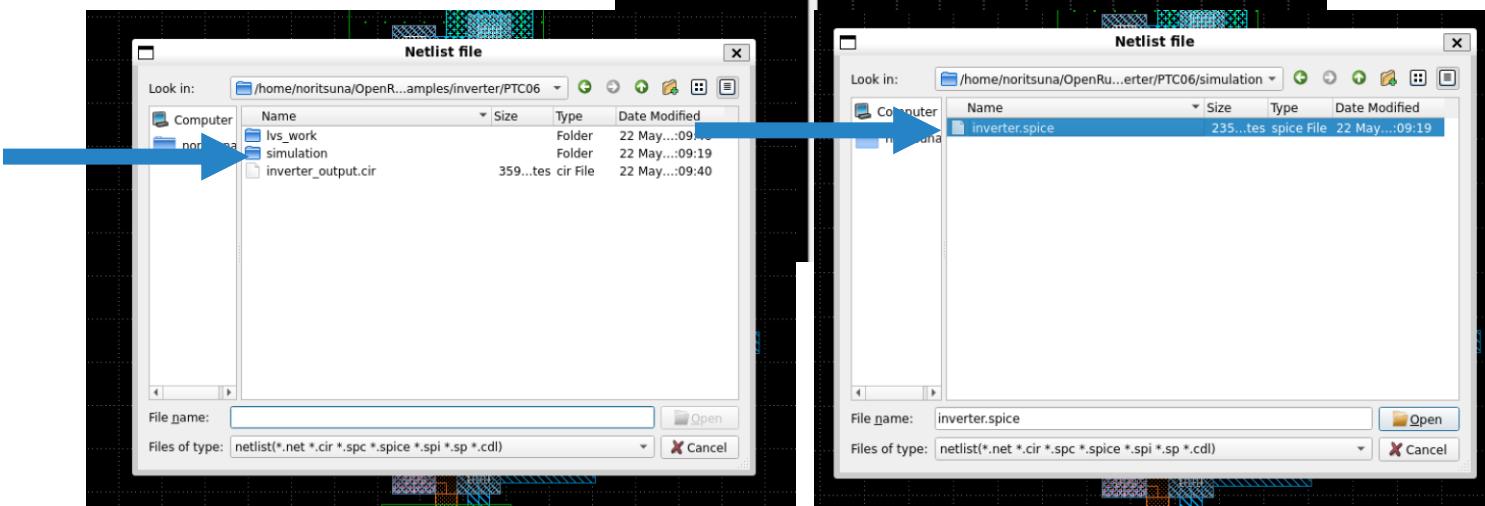
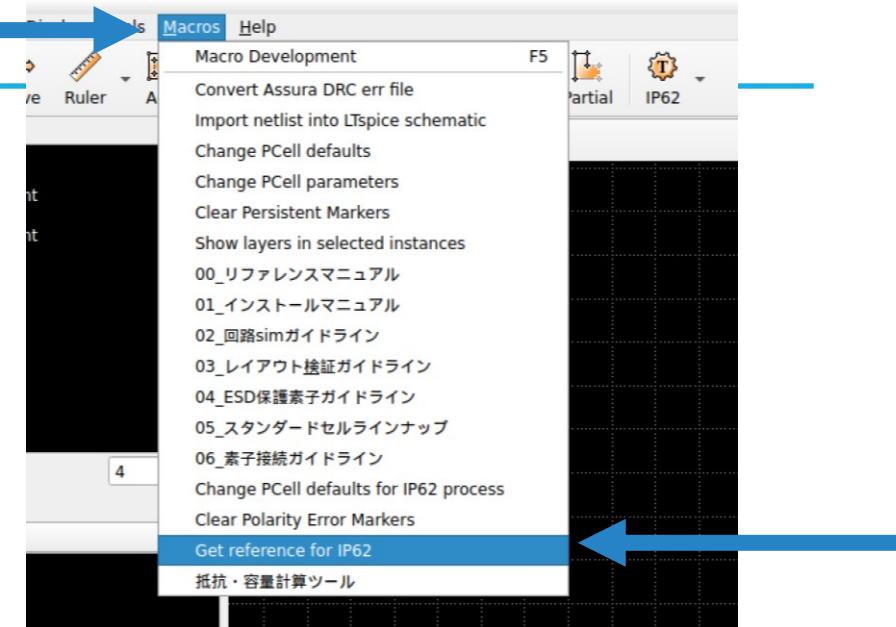
- klayout上で**LVS用**のネットリストを出力する。

1. メニュー Macros > Get reference for IP62 を実行する

2. Dialogウィンドウが出る

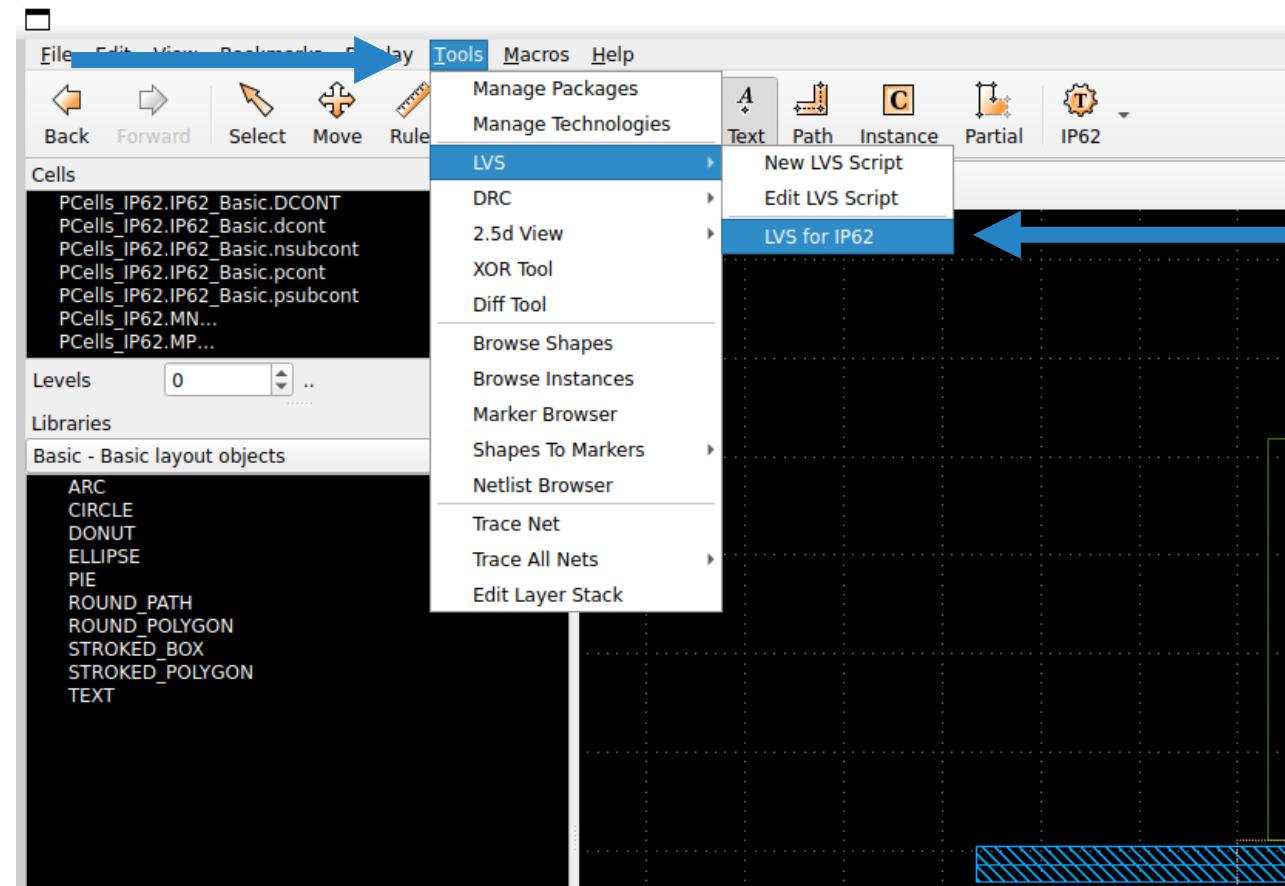
1. simulation ディレクトリを選択する

2. inverter.spice ファイルを選択する

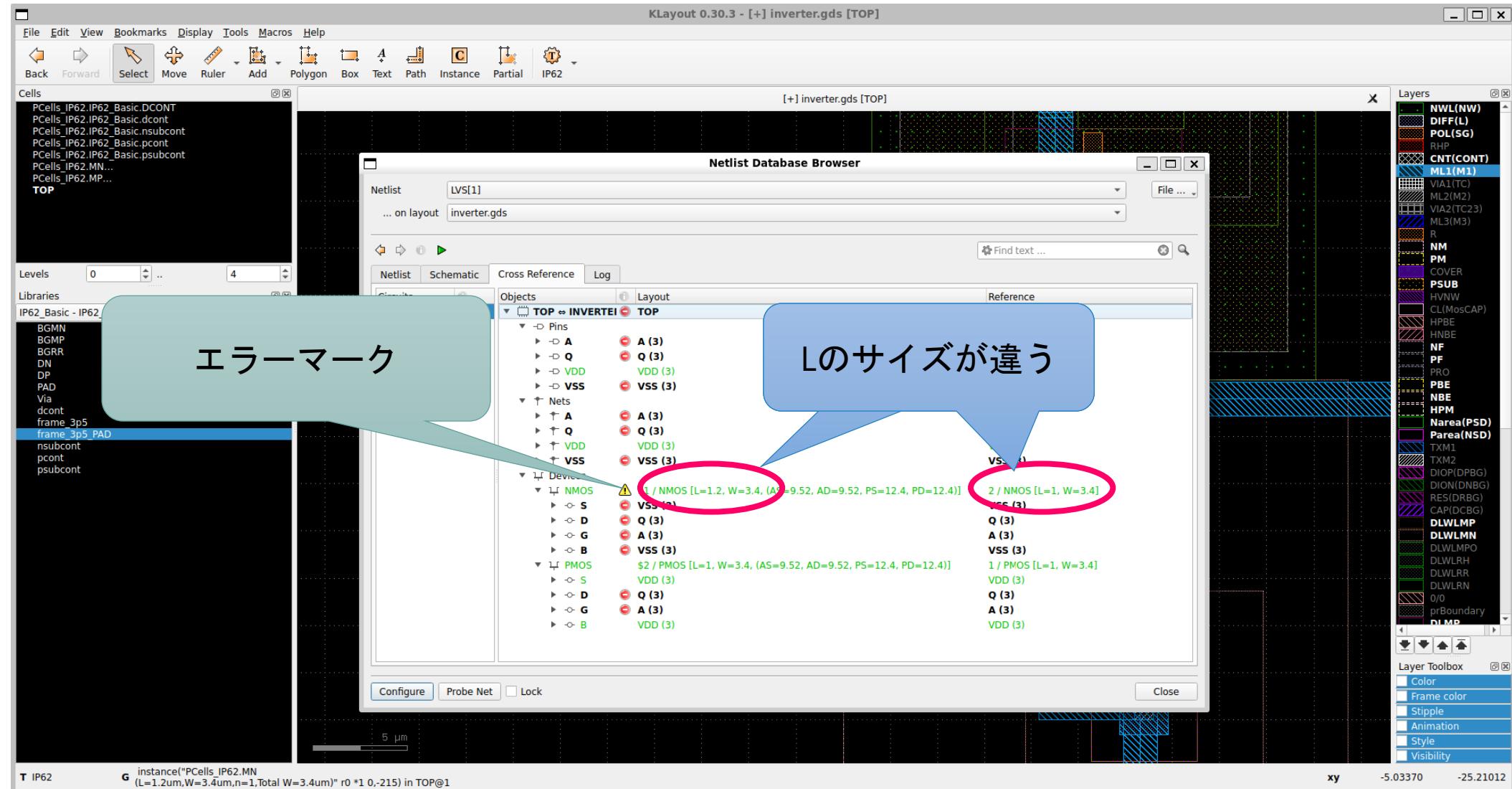


Klayout : LVSの実行

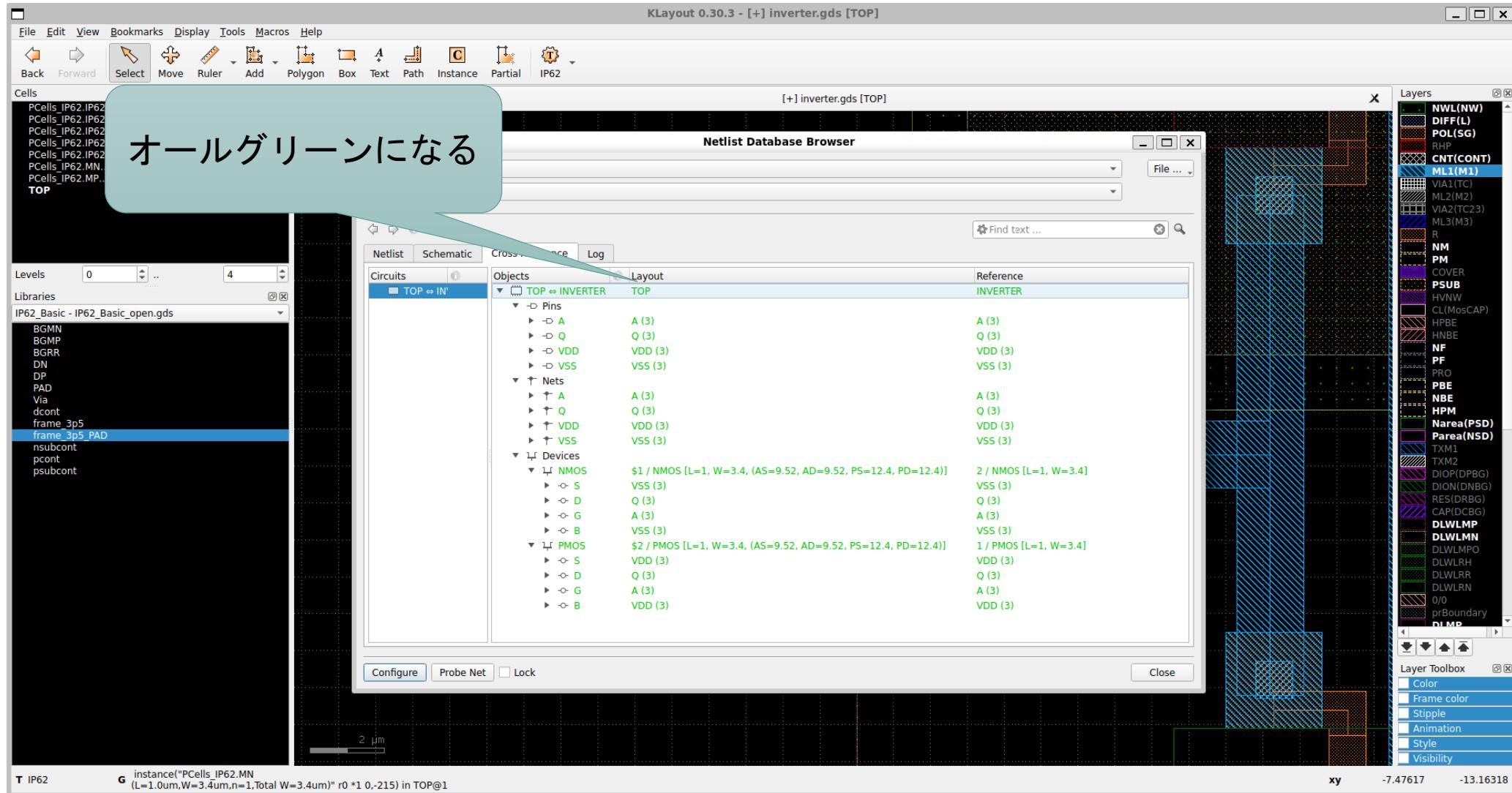
- klayout上でネットリスト比較を実行する。
 1. メニュー Tools > LVS > LVS for IP62 を実行する



Klayout : LVS NG



Klayout : LVS OK



提出

- 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）
 - 1, 自分のGithub上にリポジトリを作ってもらって、schファイルとgdsファイルをアップロードする
 - 2, READMEに感想などを記述する
 - 3, DiscordでgithubのURLを告知する
 - 4, ConnpassのIDもお願いします。
- 数日中
 - 提出物：公開用の一言など
- 半年後
 - 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