

# Virtuoso IC61 ローム 0.18 $\mu$ m PDK を使用した回路設計法

東京大学 VDEC 飯塚 哲也

e-mail: iizuka@vdec.u-tokyo.ac.jp

Ver.2 — 2019 年 3 月 28 日

## 1 背景

VDEC の主力プロセスであるローム 0.18 $\mu$ m プロセスの PDK を用いた、主にアナログ回路設計向けの設計手順について解説する。

本マニュアルでは、まず 2 入力 NAND 回路の回路図設計・レイアウト設計手順を解説する。その後、作成した NAND 回路図に修正を加えた場合を想定したレイアウト修正法について解説する。次に、作成した NAND 回路を使用したリングオシレータ回路を設計し、階層設計手順についても説明を行う。

本マニュアルでは、簡単な設定手順から、回路図設計、回路シミュレーション、レイアウト設計、レイアウト検証、配線 RC 抽出および抽出後のシミュレーション作業について基本的な手順をまとめた。いくつかの手順については簡単のため私が独自に作成したメニュー等から実行することになる。

また、本書の後半では Virtuoso Layout GXL を使用した自動配置配線の手順について解説を行い、最後に Virtuoso ADE GXL を使用した回路パラメタの自動最適化の手順についても解説を行う。最終的に得られたアンプ回路のレイアウト設計については本書では解説を行わないが、それまでに学習した手順を参考に各自実施し、性能検証等を行うこと。一通り演習が終わったら、アンプに限らず、講義で扱った教科書や Web 等を参考に面白そうな回路を作成し、シミュレーション検証、レイアウト設計等を行ってみること。

本書で使用するツールは以下のものとする。

**回路図設計** Cadence Virtuoso Schematic L

**回路シミュレーション** Synopsys HSPICE および HSIM (Synopsys の Fast SPICE シミュレータ。ちょっと古いが、使い方が HSPICE とほとんど同じなのでこれを使う。)

**回路最適化** Cadence Virtuoso ADE GXL (設計仕様に対する回路パラメタの最適化が可能。)

**波形ビューア** Synopsys Spice eXplore (Synopsys の提供する波形ビューア。Cosmos Scope を使っても良い。)

**レイアウト設計** Cadence Virtuoso Layout GXL (レイアウトエディタ上での自動配置配線についても触れる。)

**レイアウト検証 (LVS/DRC)** Mentor Calibre/Calibre Interactive

**寄生成分抽出 (PEX, LPE)** Synopsys Hercules/StarRC

本マニュアルでは、別途 VDEC と NDA 契約を締結していることを前提とする。

## 2 CAD ツールおよび PDK の設定

まず、研究室の Linux 環境に合わせた環境設定ファイルを読み込む必要がある。具体的には各種コマンドの場所 (Path)、CAD 起動のためのライセンスの設定などが正しく読み込まれていることが必要になる。以下のコマンドで雛形となるシェルスクリプトをコピーする。以下ではシェルとして zsh を使うことを前提とする。既に独自で設定を書き換えている場合は適宜バックアップを取るなどして、コピーした後で再度自分なりに編集すること。また、zsh 以外の環境を使用する場合には各自で適切に設定を行うこと。

```
% cp /usr1/iizuka/cadence/ROHM018/forNewComer/.zshrc ~/
```

コピーが完了したら設定ファイルを読み込む。

```
% source ~/.zshrc
```

以降はログイン後に自動的にホームディレクトリの .zshrc が読み込まれるため改めて実行する必要はない。.zshrc を編集した場合には再度読み込む必要がある。

次に設計ディレクトリを各自のホームディレクトリ以下に作成する。以降の作業は全てこのディレクトリの下で行う。

```
% cd  
% mkdir design
```

次に設計ディレクトリに移動し、必要なファイルを一通りコピーする。設定のためのスクリプトを作成してるので以下を実行する。これは最初に一度だけ実行すれば良い。

```
% cd design  
% /usr1/iizuka/cadence/ROHM018/forNewComer/setup_rohm018.sh
```

メッセージが表示されるので、何がどこにコピーされたか確認しておくこと。

コピーされた cds.lib ファイルに設計に必要なライブラリが一通り記載されており、virtuoso を起動するとここに記載されたライブラリが自動的に読み込まれる。続いて、コマンドラインから

```
% virtuoso &
```

として Virtuoso を立ち上げ、一緒に立ち上がる Library Manager の Library の欄に cds.lib の中身が表示されていることを確認する。Library Manager を別途起動するには、Tools > Library Manager とすればよい。

#### Tips:

Virtuoso を使用して回路設計を行っていると、Library Manager を使用する機会が多い。そのため Virtuoso の起動の度、ほぼ毎回 Tools > Library Manager として Library Manager を起動することになる。これが面倒な場合は、Virtuoso を起動するディレクトリ内の .cdsinit ファイルに以下の 1 行を追加しておく事で、Virtuoso 起動時に自動で Library Manager を立ち上げてくれる。上で読み込んだ設定ファイルではこの記述が既に追加されている。

```
ddsOpenLibManager()
```

## 3 Virtuoso Schematic L による回路図設計

### 3.1 設計ライブラリの新規作成

まず、これから設計を行うライブラリを新規作成する。ここでは、~/design 以下に testPDK と言うライブラリを作成することにする。

1. Library Manager から、File > New > Library として、New Library ウィンドウを立ち上げる。
2. Name 欄に testPDK と入力し、OK をクリックする。
3. Technology File for New Library というウィンドウが立ち上がる所以、Compile an ASCII technology file を選択し、OK をクリックする。
4. ASCII Technology File のところに Browse ボタンを使用して rohm180technology\_ic6.tf を選択して、OK をクリックする。

以上で新規ライブラリが作成された。

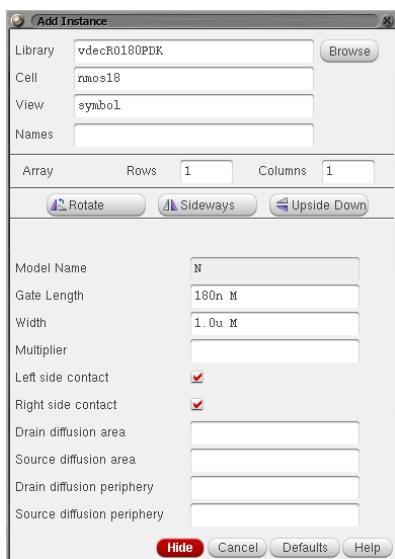


図 1: Add Instance ウィンドウ

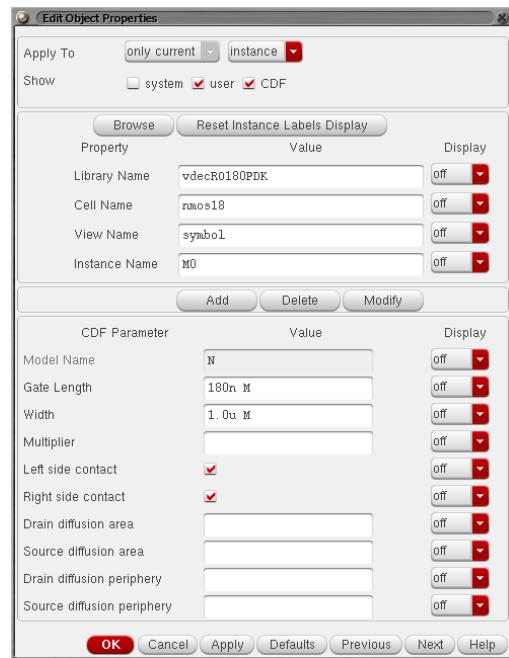


図 2: トランジスタ Property の変更

### 3.2 2 入力 NAND 回路図の設計

設計する 2 入力 NAND 回路の回路図を設計する。まず Library Manager 上で Library 欄の testPDK を選択した状態で、File > New > Cell View として New File ウィンドウを立ち上げる。この状態で、Cell 欄に NAND2 と入力し、Type 欄が schematic になっていることを確認して、OK をクリックすると空の（真っ黒の）回路図エディタが立ち上がる。ライセンスに関するウィンドウが立ち上がった場合には Yes または Always をクリックする。Always をクリックすれば以降はライセンスに関する小窓は出てこない。

このエディタ上に 2 入力 NAND 回路の回路図を記述する。

#### Tips:

今後、セル名と入出力ピン名は基本的に全て大文字を使うこと。今後いろいろな CAD ツールを使用するが、大文字、小文字の取り扱いが各社異なるため、混乱が生じないようにするためである。

1. 起動した回路図エディタ上で “i” として、図 1 に示される Add Instance ウィンドウから Browse ボタンを使用し、vdecR0180PDK ライブラリの nmos18 の symbol を選択する。
2. Add Instance ウィンドウに各種プロパティが表示される。ここで所望の Gate Length、Width 等を選択してもよいが、ここではひとまずこのまま回路図エディタ上の任意の位置に選択した NMOS トランジスタを配置する。
3. エスケープキーを押し、一度 Add Instance モードを抜ける。
4. 配置したトランジスタをクリックで選択し、“q” キーで、図 2 に示される Edit Object Properties ウィンドウを開く。

ここで、ゲートサイズの指定を行う。モデル名にはシンボルに対応したモデル名がデフォルトで入力されているため、改めて入力する必要はない（モデル名は編集しないこと）。ゲート長・ゲート幅はデフォルトで、180nm・1.0μm が入力されているが、まず試しにゲート長を 190nm と入力してみることにする。190nm と入力した後、マウスまたは Tab キーで他のフィールドに移動すると、190 と入力した箇所が自動的に 200nm に変更される事が分かる。これは本 PDK に含まれる P-Cell の仕様で

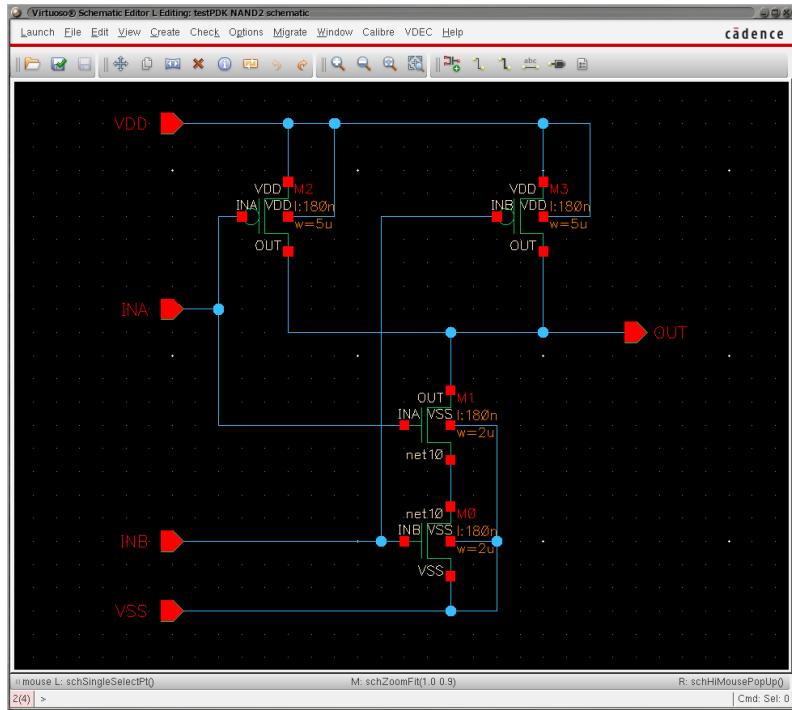


図 3: 2 入力 NAND 回路図

- ゲート長・ゲート幅の入力ステップサイズは 20nm とする

と設定されているためである。それよりも小さいステップの値を入力した場合には、本 PDK では入力された値を自動的に繰り上げる。この様な制約を持っている理由は、最小グリッドサイズが 0.02 であるレイヤにオフグリッドエラーが発生するのを防ぐためである。もし 0.01 ステップでトランジスタのゲートサイズを指定したい場合には、本 PDK のトランジスタシンボルを使用することはできない。他の(例えば名倉先生提供の)トランジスタシンボルを使用する等で回避してください。

また、トランジスタのゲート長に例えば 130nm と入力して他のフィールドに移動すると、ゲート長が自動的に 180nm に変更される事が分かる。本 PDK では、トランジスタのゲート長・ゲート幅の最小値はそれぞれ 180nm・220nm に設定されており、それより小さい値を入力した場合には自動的に許容される最小値に置き換えられる。

上記の仕様が確認できたら、トランジスタのゲート長を 180nm に入力しなおす。今回は NMOS のゲート幅 Width は  $2\mu\text{m}$  を使用するので、ゲート幅のフィールドに  $2\mu\text{m}$  と入力し、OK をクリックして Properties ウィンドウを閉じる。

上記の手順を繰り返して、図 3 に示すような 2 入力 NAND 回路図を作成する。PMOS トランジスタには pmos18 のシンボルを使用し、PMOS トランジスタのゲートサイズは、ゲート長 180nm、ゲート幅  $5\mu\text{m}$  とする。配線を接続する際は “w” として、各シンボルの端子を接続する。ピンを置く際は “p” として INA、INB と言う 2 入力 (Direction は input) と、OUT と言う 1 出力 (Direction は output) 端子を置く。電源・グラウンドとしては同じようにピンとして VDD、VSS と言う 2 つの入力ピン (Direction は input) を置くのがおすすめであるので、ここではそのようにする。

回路図エディタウィンドウ左上の、フロッピーディスクにチェックマークがついたボタンで Check and Save を行う。ワーニングやエラーがある場合にはポップアップが立ち上がる。ワーニングは無視してもよい場合が多いが、基本的には全て無くなるようにすること。Check and Save がうまく行くと、トランジスタの各端子に接続されているネット名が表示される。結線の確認に使用するとよい。

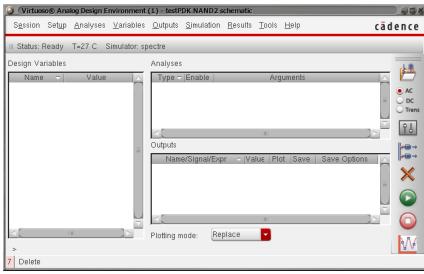


図 4: Analog Design Environment ウィンドウ

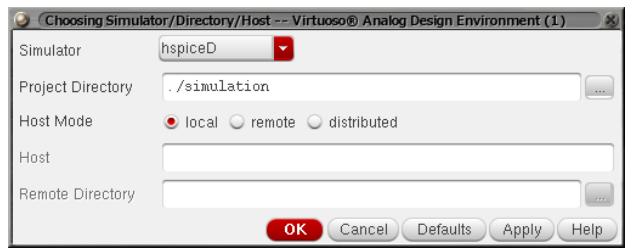


図 5: Choosing Simulator/Directory/Host ウィンドウ

### 3.3 回路シミュレーション

設計した 2 入力 NAND の回路シミュレーションを Synopsys 社の HSPICE を使用して行う。本マニュアルでは HSPICE シミュレーションに必要なファイルの中身について詳細に説明しない。HSPICE コマンドの詳細については別途 HSPICE のマニュアル等を参考にすること。ただし、書いてある内容からおおよそ想像がつくはずである。

以下では、研究室独自のメニューを使用してトランジスタレベルネットリストを出力し、コマンドラインから HSPICE を実行して、Synopsys 社の Spice eXplore という波形ビューアを使用して波形を確認するフローを例として示す。

#### 3.3.1 ネットリストの出力

まずは一般的なネットリスト出力手順について説明し、その後研究室独自の VDEC メニューの使用方法について説明する。

回路図エディタウィンドウ左上のメニューから Launch > ADE L として、図 4 に示す Analog Design Environment(ADE) ウィンドウを起動する。ライセンスに関するウィンドウが立ち上がった場合には Yes また Always をクリックする。ADE ウィンドウのメニューから Setup > Simulator/Directory/Host... として、図 5 に示す Choosing Simulator/Directory/Host ウィンドウを開く。同ウィンドウで、Simulator に hspiceD を選択する。Project Directory はデフォルトで ~/simulation となっており、ここではひとまずそのまま使用する。その後 OK をクリックすると少々時間がかかるが設定が完了する。設定が完了したら ADE ウィンドウのメニューから Simulation > Netlist > Create と実行すると、ネットリストが生成され、内容を表示するウィンドウが立ち上がる。所望のネットリストになっているかどうかを簡単にチェックして、ネットリストが表示されたウィンドウは閉じてしまつてよい。

以上が一般的な方法であるが、今後は以下の方法で簡便にネットリストを作成するものとする。

回路図エディタウィンドウ右上に VDEC というメニューがあるので（自作のメニューであるため、通常は存在しない）、VDEC > Run Netlister として、図 6 に示す VDEC Netlister ウィンドウを起動する。ネットリストを出力したい対象の Library Name、Cell Name が選択されていることを確認し、View Name が schematic となっている事を確認した後、左側の Make Netlist ボタンをクリックすることでネットリストが出力される。出力されたネットリストは図 7 に示すようなウィンドウに表示される。NAND2 の場合は 4 個のトランジスタ (m から始まる行) が正しく含まれていることを確認する。その後、このウィンドウは閉じてしまつてよい。

ネットリストが出力されるディレクトリは、同ウィンドウの最上段の Run Directory に示されているが、デフォルトでは Virtuoso を起動したディレクトリ以下に「simulation/ライブラリ名/セル名」というディレクトリを自動的に作成し、そのディレクトリ内に netlist という名前のファイルが作成される。ネットリストのファイル名はウィンドウ最下段の Output Nelist File から変更することもできる。



図 6: VDEC Netlister ウィンドウ

```

## Generated for hspiceD
## Generated on: Mar 27 08:16:12 2018
## Design library name: testPDK
## Design cell name: NAND2
## Design view name: schematic

TEMP 25
OPTION
+ ARTIST=2
+ INGOLD=2
+ PARHIER=LOCAL
+ PSF=2

** Library name: testPDK
** Cell name: NAND2
** View name: schematic
m1 net1 net2 net3 net4 net5 N L=180e-9 W=2e-6
m2 net6 net7 net8 net9 N L=180e-9 W=2e-6
m3 net10 net11 net12 net13 net14 P L=180e-9 W=5e-6
m2 net15 net16 net17 net18 P L=180e-9 W=5e-6
END

```

図 7: 生成された Netlist ウィンドウ

### Tips:

Analog Design Environment を起動した後に、Setup > Simulator/Directory/Host から Simulator と Directory の設定を行ったが、それが面倒な場合は各自の Home Directory に .cdsenv というファイルを作成し、以下を記述する(既にある場合はそこに追記する)。

```
asimenv.startup simulator string "hspiceD"
asimenv.startup projectDir string "~/simulation"
```

以上を記述後に Virtuoso を再起動すると、次回以降は Setup > Simulator/Directory/Host の設定がされた状態で ADE を起動することができる(もちろん projectDir の場所は任意)。ちなみに、.cdsenv ファイルではなく .cdsinit ファイルで設定を行いたければ以下を Virtuoso を起動するディレクトリの .cdsinit ファイルに記載すると同じ設定ができる。

```
envSetVal("asimenv.startup" "simulator" 'string "hspiceD")
envSetVal("asimenv.startup" "projectDir" 'string "~/simulation" )
```

### 3.3.2 HSPICE の実行

次に、HSPICE を実行して回路シミュレーションを行う。回路シミュレータ HSPICE の細かい使用法については研究室の Workstation に置いてあるマニュアルを参照するほか、Google 等でも調べられる。もちろん先輩に聞くのが一番早いかも知れない。

前節で VDEC メニューより出力した NAND2 のネットリストファイルは以下のパスに出力される。

```
~/design/simulation/testPDK/NAND2/netlist
```

cd コマンドで上記ディレクトリに移動する。今回は HSPICE 実行に必要な SPICE Input ファイルの例を作成してあるので、以下のコマンドでコピーする。

```
% cp /usr1/iizuka/cadence/ROHM018/forNewComer/nand2.sp .
```

ファイルの内容を次ページに示す。

次にターミナル上から

```
% hspice -i nand2.sp -o nand2
```

とやってみると、HSPICE が実行できることが確認できる。

```
>info: ***** hspice job concluded
```

と表示されれば無事実行が完了している。

```
*****
* SPICE input file for NAND2
*****
.options ingold=2 brief
.option post probe
*****
* Probe option
*****
.probe v(*) $ 本例では全ての信号電圧を出力する $
*****
* SPICE model files
*****
$ モデルファイルの指定 $
.include "/usr1/iizuka/cadence/ROHM018/rules/spice/hspice/bu40n1.mdl"
.lib "/usr1/iizuka/cadence/ROHM018/rules/spice/hspice/bu40n1.skw" NT
.lib "/usr1/iizuka/cadence/ROHM018/rules/spice/hspice/bu40n1.skw" PT
*****
* Technology dependent parameters
*****
.temp 25
.param operatingVoltage = 1.8
.param timeIncrement = 10p
.param runHoldLength = 10n
.param transitionTime = 0.05n
*****
* HSIM Settings
*****
.param HSIMANALOG=3
.param HSIMSPEED=1
.param HSIMOUTPUTFLUSH=5n
*****
* Stimula
*****
.tran timeIncrement 'runHoldLength * 10'

vINA INA 0 pulse (
+ 0 'operatingVoltage'
+ 'runHoldLength * 1'
+ 'transitionTime' 'transitionTime'
+ 'runHoldLength * 1 - transitionTime' 'runHoldLength * 2'
+
)
vINB INB 0 pulse (
+ 0 'operatingVoltage'
+ 'runHoldLength * 1'
+ 'transitionTime' 'transitionTime'
+ 'runHoldLength * 2 - transitionTime' 'runHoldLength * 4'
+
)
*****
* Power supply
*****
vdd vdd vss dc operatingVoltage
vss vss 0 dc 0.00
*****
* Top level circuit definition
*****
.include "./netlist" $ ここも必要に応じて書き換える $
.end
```

SPICE Input ファイルは毎回必要な信号入力等を手で記述する必要があるので、独自の回路を作った場合には、このファイルを参考に必要な変更を加えること。

### 3.3.3 HSIM の実行

次に、試しに HSIM を実行して回路シミュレーションを行う。HSIM は高速 SPICE シミュレータであり、HSPICE よりも精度は劣るが、実行時間は早い。また精度と実行時間の関係がある程度調整可能である。そのため、大規模な回路の動作を簡易的に確認する等の場合には有用である。

ここで、先に示した SPICE Input ファイルのうち HSIM Setting とコメントを記載した部分以外は全て HSPICE 用の入力ファイルとまったく同じものが使用できる。HSIM 用の設定記述は、先に試したように、残っていても HSPICE の実行時には自動的に無視されるので問題無い。この部分で HSIM の精度設定などを行っている。それぞれの意味は以下のとおり。

**HSIMANALOG** HSIM の精度設定。値が高いほど精度が高い。値の範囲は -1 から 3。

**HSIMSPEED** HSIM の精度設定。値が高いほどシミュレーションスピードが早く精度が悪い。値の範囲は 0 から 8。

**HSIMOUTPUTFLUSH** HSIM はデフォルトではシミュレーション結果をメモリに保存し、メモリがいっぱいにならたらディスクに書き出す。このため、シミュレーション途中で波形を見たい場合等に困ることがある。この設定をしておくと、指定した時間シミュレーション上の時間が経過する度に結果を波形ファイルに書き込むのでシミュレーション途中でも波形の確認ができる。値は適宜指定する。

本設定は HSIM を使用する上での最低限の設定であるが、大抵の場合これだけ設定しておけば問題なく使用できる。他にも便利な使い方が多数あるが、詳細な使用法・設定法は HSIM のマニュアルを参照されたい。上記の設定は .param で指定してあるため、この記述を入れたままでも HSPICE の入力ファイルとしても問題なく使用できることは一つの使いやすいポイントである。

同じ nand2.sp を使用して、以下のコマンドをターミナル上に入力し HSIM を実行する。

```
% hsim -i nand2.sp -o nand2
```

実行を行うとログメッセージがターミナル上に流れ、実行が終了する。実行が終了すると nand2.fsdb というファイルが生成されるので ls コマンド等で確認する。.fsdb ファイルも通常の波形ビューアで問題なく読み込める。この程度の回路規模だと実行時間にはほとんど差がない… どころか、HSPICE の方が速いくらいである。違う回路や精度設定等を変えてみてどのようになるか自分で確認してみるとよい。

### 3.3.4 シミュレーション波形の確認

シミュレーション結果の波形を波形ビューアを使用して確認する。ここでは Spice eXplore に含まれる Wave View (wv) と言う Synopsys 社の波形ビューアを使用する。Cosmos Scope (cscope) 等の他のビューアを用いても問題ない。

コマンドラインから

```
% wv &
```

と入力し、波形ビューアを呼び出す。(sx & として起動すると、より高機能な状態で起動するが、今回は波形を見る事しかしないため、よりシンプルな GUI で起動する。) 実行すると、図 8 に示すウィンドウが立ち上がる。

左上のフォルダボタンをクリックし、立ち上がるファイルブラウザから nand2.tr0 (HSIM の結果を見る場合は nand2.fsdb) ファイルを選択して OK をクリックする。Output View 欄に表示される D0:nand2.tr0 と言うところをダブルクリックし、下に toplevel という文字が表示されたらそこを選択するとさらに下の欄に出力した信号名が表示される。表示したい信号名をダブルクリックすると右側に波形が表示される。二つの入力 v(ina) と v(inb)、および出力信号 v(out) を表示すると、図 9 の様に波形が表示されるので、2 入力 NAND 回路として動作している事を確認する。(波形ビューアの表示色等は実際に起動する画面と少々異なります。各自 Config > Preferences 等でお好みに設定してください。) 所望の動作が確認できた場合は次のレイアウト設計に進む。動作が正しくない場合には回路が正しいかを確認し、再度ネットリスト出力、シミュレーション、波形の確認を行い正しい動作をする回路を設計する。

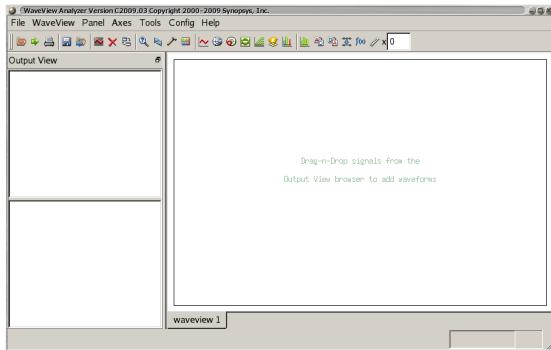


図 8: Wave View 起動画面

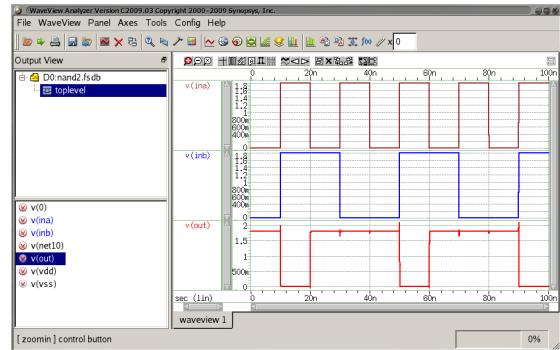


図 9: 2 入力 NAND 回路のシミュレーション波形

## 4 DC/AC シミュレーションの実施

少々唐突ではあるが、ここで HSPICE を用いた DC および AC シミュレーションの実行について説明を行う。先に行ったシミュレーションでは時間で変化する波形を確認するために Transient のシミュレーションを行ったが、特にアナログ回路設計の際には、トランジスタのバイアス条件を決定するための DC シミュレーションと、回路の周波数特性を検証するための AC シミュレーションも重要なとなる。先と同様に、ここでは HSPICE の細かい使用法や入力ファイルの中身の詳細については説明しない。

まず、基本的な DC シミュレーションの HSPICE 実行に必要な SPICE Input ファイルの例を作成してあるので、以下のコマンドでコピーする。

```
% cp /usr1/iizuka/cadence/ROHM018/forNewComer/dc.sp .
```

ファイルの内容を次ページに示す。この例では、NMOS トランジスター個の電圧・電流特性を確認している。アナログ回路設計では特に重要な特性であるので、新しいプロセス技術等に触れる機会では必ず最初に実行すべきシミュレーションである。先に出力した nand2 の netlist を見て既に分かっているとおり、m から始まる行でトランジスタを指定することができる。この例では、HSPICE 入力ファイルの中で直接トランジスタを指定している (Transistor Definition の箇所)。慣れれば簡単な回路であれば毎回 Virtuoso で回路図を書かなくてもテキストベースで指定した方が早いこともある。

シミュレーション実行は前節と同じ以下のディレクトリで実行すれば良い（ただし、この節の DC/AC シミュレーションはどこでやっても実行できる）。

```
~/design/simulation/testPDK/NAND2/
```

先と同様にターミナル上から以下のコマンドで HSPICE が実行できる。

```
% hspice -i dc.sp -o dc
```

次に結果を確認する。先と同様に WaveView を使い、dc.sw0 を選択する。toplevel を選択すると、i1(m0) ~i4(m0) までの電流波形が確認できる。m0 は使用したトランジスタのインスタンス名であり、i1~i4 はそれぞれ

- i1: ドレインに流れ込む電流
- i2: ゲートに流れ込む電流 (ROHM プロセスではゲートリークがモデルされていないためゼロ)
- i3: ソースから流れ出す電流
- i4: 基板に流れ込む電流

を示している（ドレイン側とソース側とで電流の向きが違うのに注意）。通常は i1 だけ見れば充分であることが多い。波形ビューワで i1(m0) を指定すると講義等で見たことのある  $I_d-V_{ds}$  特性が見られるはずである。

```
*****
*
* SPICE input file for DC simulation
*
*****
.options ingold=2 brief
.option post probe
*****
* Probe option
*****
.probe dc i1(m0) i2(m0) i3(m0) i4(m0)
*****
* SPICE model files
*****
.include "/usr1/iizuka/cadence/ROHM018/rules/spice/hspice/bu40n1.mdl"
.lib "/usr1/iizuka/cadence/ROHM018/rules/spice/hspice/bu40n1.skw" NT
.lib "/usr1/iizuka/cadence/ROHM018/rules/spice/hspice/bu40n1.skw" PT
*****
* Technology dependent parameters
*****
.temp 25
.param operatingVoltage = 1.8
.param plotNum = 20
*****
* Transistor definition
*****
m0 drain gate source vss N L=180e-9 W=2e-6 M=1
*****
* Stimula
*****
v1 drain vss dc 0
v2 gate vss dc 0
v3 source vss dc 0
*****
* Power supply
*****
vdd vdd vss dc operatingVoltage
vss vss 0 dc 0.00
*****
* Sweep v1 and v2
*****
.dc v1 0.0 operatingVoltage 'operatingVoltage / plotNum'
+ v2 0.0 operatingVoltage 'operatingVoltage / plotNum'
.end
```

次に、ACシミュレーションの実行を試してみる。こちらも入力ファイル例を作成してあるので、以下のコマンドでコピーする。

```
% cp /usr1/iizuka/cadence/ROHM018/forNewComer/dcac.sp .
```

ファイルの内容を次ページに示す。この例では、NMOS トランジスタ一個と負荷抵抗を用いたソース接地増幅回路のシミュレーションを行っている（Circuit Definition の箇所を確認すること）。ファイル最下部において、DC および AC シミュレーションを実行している。ゲートに接続されている電圧源 V2 の指定方法に注意する。適当な DC バイアス電圧の指定とともに、ac 1 の指定があるが、これは 1V 振幅の AC 信号をこの端子に印可し、シミュレーションを実行するという意味である。ただし、実際に  $\pm 1V$  の正弦波が印可されるわけではなく、あくまで小信号のシミュレーションにおいて得られる特性に対して 1V 入力の結果が得られるだけ（ここで指定する振幅を大きくしたり小さくしたりしてもシミュレーションの結果として得られるゲインの値は変わらない）であるので、後々のゲインの計算等の都合を考えると 1 を入力しておくのが最も簡便である。

```
*****
*
* SPICE input file for DC/AC simulation
*
*****
.options ingold=2 brief
.option post probe acout=1

*****
* Probe option
*****
.probe dc v(drain)
.probe ac v(drain) vdb(drain,gate) gain=par('vdb(drain)-vdb(gate)')
.probe ac vp(drain) vp(drain,gate) phase=par('vp(drain)-vp(gate)')

*****
* SPICE model files
*****
.include "/usr1/iizuka/cadence/ROHM018/rules/spice/hspice/bu40n1.mdl"
.lib "/usr1/iizuka/cadence/ROHM018/rules/spice/hspice/bu40n1.skw" NT
.lib "/usr1/iizuka/cadence/ROHM018/rules/spice/hspice/bu40n1.skw" PT

*****
* Technology dependent parameters
*****
.temp 25
.param operatingVoltage = 1.8
.param plotNum = 10
.param startFreq = 10MEG
.param stopFreq = 10G

*****
* Circuit definition
*****
m0 drain gate source vss N L=180e-9 W=2e-6 M=1
r1 drain vdd 10k

*****
* Stimula
*****
v2 gate vss dc 'operatingVoltage * 0.4' ac 1
v3 source vss dc 0

*****
* Power supply
*****
vdd vdd vss dc operatingVoltage
vss vss 0 dc 0.00

*****
* Sweep v1 and v2
*****
.dc v2 0.0 operatingVoltage 'operatingVoltage / plotNum'
.ac dec plotNum startFreq stopFreq

.end
```

いくつか AC シミュレーションの実行ファイルに関して注意点がある。入力ファイルの上部、.option の行に acout=1 というオプションが追加されている。これはその下にある.probe による波形出力の計算方法を変えるものであり、デフォルト（acout=1 を明示的に指定しない場合）は acout=0 が指定されている。このオプションにより、vdb(drain,gate) および vp(drain,gate) の計算方法が変わる。より明示的に計算した結果である、gain、phase についても波形を出力しているので、オプションや先に指定した AC 振幅の値を変えて挙動の違いを見てみるとよい。詳細は HSPICE のマニュアルや Google で得られる情報を参考にされたい。

以下のコマンドで HSPICE を実行する。

```
% hspice -i dcac.sp -o dcac
```

次に結果を確認する。まずは DC シミュレーションの結果を確認する。WaveView で dcac.sw0 を開き、v(drain) を確認する。V2 の DC 電圧の変化に対して、ドレイン電圧が変化していることが分かる。グラフの形状から

おおよそ 0.7V 程度のバイアスを与えると、大きな小信号ゲインが得られそうだと分かる。その周囲での傾きを計算すると、ざっくり 4~6 倍程度（極性は反転）のゲインが得られそうだと分かる。

今回のシミュレーションファイルでは、AC シミュレーション時の DC バイアス電圧 (V2) が、ちょうどゲインが大きく取れる付近になる様に設定してあるので、このまま AC シミュレーションの結果を確認する。WaveView で dcac.ac0 を開き、gain (または vdb(drain,gate)) と phase (または vp(drain,gate)) の波形を確認する。波形ビューワ上の横軸で右クリックし、Log Scale に変更することで、馴染みのある Lowpass 特性が観測できる。DC ゲインを確認すると約 15dB 程度となっており、6 倍弱のゲインが実現できている事が分かる。これは先にざっくり見積もった値と合致する。また、位相が 180° から 90° に向かって変化していくことも分かる。V2 の DC 電圧や負荷抵抗値を変化させて挙動をより詳しく見てみるとよい。

## 5 Monte Carlo シミュレーションの実施

実際の集積回路製造プロセスにはかならず「ばらつき」が存在する。そのため、製造された回路上では全てのトランジスタがそれぞれ僅かに異なる性能を持つことになる。しかしながら、これまでのシミュレーションでは全てのトランジスタは (W, L が同じであれば) 全く同じ特性を持つものとして計算される。そのため、実際に測定するとシミュレーションとは異なる結果となり、最悪の場合には回路が動作しないことになる。この「プロセスばらつき」を加味したシミュレーションを行うために必要なのが Monte Carlo (モンテカルロ) シミュレーションである<sup>1</sup>。

ここでは先に作成した NAND の回路を利用してモンテカルロシミュレーション法に慣れておく（トランジスタが 4つしかないのあまり面白くないですが）。

作成した NAND 回路のネットリストを使用するため、以下のディレクトリに移動する。

```
~/design/simulation/testPDK/NAND2/
```

例となるファイルがあるので、以下のコマンドでコピーする。

```
% cp /usr1/iizuka/cadence/ROHM018/forNewComer/nand2_process.sp .
```

ファイルの内容を次ページに示す（前と同じ部分は省略する）。

先に使用した Transient シミュレーションの内容とほとんど同じだが、.tran を指定するところに、モンテカルロを実行する回数等の指示があり、最下部にばらつきの具合を指定するための Variation Block が指定されている。<sup>2</sup>トランジスタの W や Vth0 の値をランダムに変化させながら、monte=8 で指定された 8 回のシミュレーションを実行する。

これまで同様、以下のコマンドで HSPICE を実行する。

```
% hspice -i nand2_process.sp -o nand2_process
```

また同様に波形を確認する。出力 v(out) を観測し、信号の遷移部分（立ち上がりでも立ち下がりでも良い）を拡大すると、8 本の波形が僅かに異なる形状で出力されていることが分かる。NAND の例ではトランジスタ数が少ないためそれほど大きな差は観測できないが、回路規模が大きくなるとそれだけばらつきの影響も大きくなる。また、シミュレーション時間も長くなる。monte=で指定する回数を増やして再実行し結果を見ておいても良い。また波形ビューワによって遷移時間のばらつきを観測するなど、ツールの使い方に慣れておくと良い。

<sup>1</sup>なぜモンテカルロと呼ばれるのかとかは検索すれば分かります。

<sup>2</sup>TSMC 社等の世界的にも標準なプロセス技術においては、ばらつきの程度はシミュレーションモデル内に取り込まれて提供されることが通常である。TSMC 社等では製造会社側から提供される情報であるためばらつき具合の情報はかなり正確であるが、ROHM プロセスではばらつきの情報が提供されておらず、こちらで「妥当と思われる」値を入力している。過去にいくつかの限られたケースに置いて実測と比較して「大きく外れていらない」ことは確認済みではあるが、残念ながらこれが真実である保証は無い。

```
*****
*
* SPICE input file for NAND2 with process variation
*
*****
... 中略 ...
*****
*
* Stimula
*****
.tran timeIncrement 'runHoldLength * 10' sweep monte=8 firstrun=1
    ... 中略 ...
*****
*
* Variation Block
*****
.Variation
Option Ignore_Global_Variation=Yes
.Local_Variation
NMOS N
+ Vth0 = '1e-3*3.635/(sqrt(get_E(M)*get_E(W)*get_E(L)*1e12))',
+ Wint = '0.373/(sqrt(get_E(M)*get_E(W)*get_E(L)*1e12))', %
PMOS P
+ Vth0 = '1e-3*4.432/(sqrt(get_E(M)*get_E(W)*get_E(L)*1e12))',
+ Wint = '0.326/(sqrt(get_E(M)*get_E(W)*get_E(L)*1e12))', %
.End_Local_Variation
.End_Variation
.end
```

## 6 热雜音シミュレーションの実施

先のプロセスばらつき以外にもこれまでのシミュレーションでは加味されていない効果として、トランジスタ自身から生じる雑音がある。<sup>3</sup>ここでは熱雑音<sup>4</sup>を加味したシミュレーション方法について実行方法を知つておく。ここでもNAND回路を利用するため、先と同じディレクトリにてシミュレーションを実行すること。

また例となるファイルがあるので、以下のコマンドでコピーする。

```
% cp /usr1/iizuka/cadence/ROHM018/forNewComer/nand2_noise.sp .
```

ファイルの内容を下に示す（前と同じ部分は省略する）。

Transientシミュレーションの指定箇所の下に、ノイズを印可するための指定がある。プロセスばらつきの際と同様に、モンテカルロ（mc）法を用いてランダムに雑音を加えて16回のシミュレーションを行う。各指定の詳細についてはHSPICEのマニュアル等を参照されたいが、fmin, fmaxでどの周波数範囲まで熱雑音を加味するか指定するため、実際のシミュレーション対象回路上での信号帯域等（デジタル信号では立ち上がり・立ち下がり時間など）をよく検討して設定する必要がある。

```
*****
*
* SPICE input file for NAND2 with thermal noise
*
*****
... 中略 ...
*****
*
* Stimula
*****
.tran timeIncrement 'runHoldLength * 10'
.trannoise v(OUT) method=mc samples=16 seed=1 fmin=1 fmax=1.4G scale=1
.probe trannoise onoise v(INA) v(OUT)
... 中略 ...
.end
```

<sup>3</sup>それ以外にも他の回路や外部からの雑音、電源雑音等もあるがそれらは必要に応じてまた別の方法で取り込むことになる。

<sup>4</sup>フリッカ雑音もあるが、残念ながらROHMプロセスでは雑音のモデルが提供されておらず、正確な熱雑音・フリッカ雑音を加味したシミュレーションは出来ない。こちらもTSMC社等の製造会社ではかなり正確なモデルが提供されている。

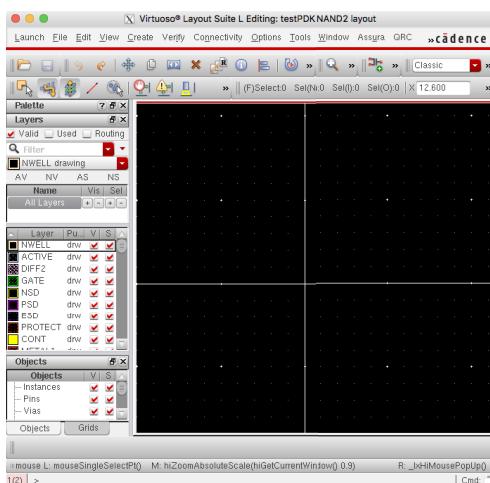


図 10: LayoutL 起動画面

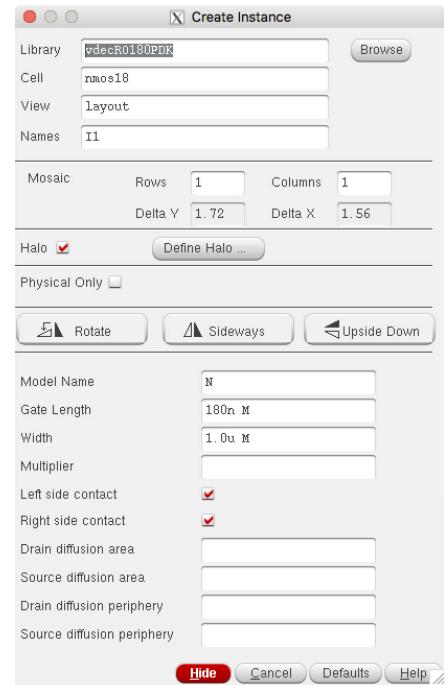


図 11: NMOS Pcell の呼び出し

これまで同様、以下のコマンドで HSPICE を実行する。

```
% hspice -i nand2_noise.sp -o nand2_noise
```

また同様に波形を確認する。出力  $v(out)$  を観測し、信号の遷移部分（立ち上がりでも立ち下がりでも良い）を拡大すると、先と同様に僅かに異なる形状で 16 本の波形が出力されていることが分かる。これまた NAND の例ではトランジスタ数が少ないためそれほど大きな差は観測できないが、回路規模が大きくなる（例えば NAND を直列に何段も繋ぐ）と、最終段ではそれだけ雑音の影響も大きくなる。また、やはりシミュレーション時間も長くなる。samples=で指定する回数を増やして再実行し結果を見ておいても良い。また波形ビューワによって遷移時間のばらつきを観測するなど、ツールの使い方に慣れておくと良い。

## 7 Virtuoso Layout L によるレイアウト設計

上で回路図を作成した 2 入力 NAND 回路のレイアウトを設計する。

1. Library Manager 上で testPDK ライブラリを選択した状態で File > New > Cell View として New File ウィンドウを立ち上げる。
2. 立ち上がったウィンドウで Cell 欄に NAND2、Type 欄に layout を選択して OK をクリックすると図 10 に示すような、空の（真っ黒の）レイアウトエディタが立ち上がる。ライセンスに関するウィンドウが立ち上がった場合には Yes または Always をクリックする。
3. まずは必要なトランジスタを呼び出す。先に作成した 2 入力 NAND 回路では 2 つの NMOS トランジスタ、2 つの PMOS トランジスタを使用している。まず画面上で  $i$  キー（Create / Instance）で Create Instance ウィンドウを呼び出し、右上の Browse ボタンをクリックして、Library: vdecRO180PDK、Cell: nmos18（1.8V コア用 NMOS トランジスタ）、View: layout と選択する。図 11 に示すように、Model Name、Gate Length、Width にデフォルトの値が入る。
4. この状態でマウスカーソルを Layout L ウィンドウ上に移動してみると、赤い枠で nmos18 のレイアウトが表示される。適当な位置（座標 0,0）でクリックしてみる。画面の拡大にはマウスの右クリックでド

ラッグ（領域選択）して希望の範囲を拡大することができる。Shift-Z でズームアウト、Ctrl-Z でズームインも可能である。

5. クリックした段階では赤い枠のままなので、中身が見えるようにする。layout L ウィンドウが選択された状態で、Shift-F とすると中身が見える。また、Ctrl-F とすると元の階層表示に戻る。
6. 図 12 に nmos18 をデフォルトのまま置いた例を示す。緑色が GATE、左右の赤い部分が METAL1、白い部分が ACTIVE（拡散層）、大外の黄色枠が NSD（N 型 Implant）、黄色く塗りつぶされている部分が CONT（コンタクト）であり、対応するレイヤについてはウィンドウの左側に示されている。CONT を経由して ACTIVE と METAL1 が接続しており、左右の METAL1 はそれぞれトランジスタの Source/Drain になっている（レイアウトは左右対称なので、どちらがどちらかは厳密ではない）。ESC キーにより、現在実行しているコマンドを抜けることができる（困ったらとりあえず ESC を連打する）。
7. 今置いてあるトランジスタは、先に作成した回路図で使用したトランジスタのサイズと異なっており、適切なサイズに修正する。レイアウト上でトランジスタのどこかをクリックし、トランジスタレイアウトが選択された状態（白枠で囲われる）にし、q (Edit / Properties...) キーを押す。図 13 に示すウィンドウが立ち上がるるので、中央付近の Parameter タブをクリックで選択する。Schematic では  $2\mu\text{m}$  を用いたので、Width の欄に  $2\mu$  と入力し、下部の OK をクリックすると、レイアウト上でトランジスタの W が正しく  $2\mu\text{m}$  となることが確認できる。
8. 次にもう一つの NMOS トランジスタを i キーを使って呼び出す。今回は最初から Create Instance ウィンドウ上で Width 欄に  $2\mu$  と入力する。（先ほど修正したので自動的に  $2\mu$  と入っているかも知れない。）先に置いた NMOS と、Source/Drain がぴったり重なるように置く。デフォルトの状態だとクリックできる位置のグリッドが粗く、ぴったり重ならないかも知れない。その時は、数字の 1 キーや 2 キーを押してグリッドの設定を変更する事ができる。ここでは一度 2 キーを押してみると、ぴったり合わせることができる。置いた結果を図 14 に示す。
9. k キーで Ruler を出すことができる。適当な二点をクリックすることでその間の距離を測ることができる。Design Rule を満たすために必要な間隔や幅を計測するために必須の機能である。u キーで Undo、Shift-u キーで Redo が可能である。また、適宜左上のフロッピーディスクのマークをクリックし、レイアウトの保存を行うこと。ただし一度保存するとその時点から Undo できなくなるので注意。
10. 引き続いて PMOS トランジスタと同じ手順で置いていけば良いのだが、操作に慣れるため別の方法で PMOS トランジスタを置いてみる。まず、上で置いた二つの NMOS トランジスタを左クリックによるドラッグで二つとも取り囲み両者を同時に選択した状態とする。（今の場合は Ctrl-A により全選択してもよい。）
11. 次に c キーにより Copy コマンドを実行する。選択された NMOS トランジスタの適当な位置をクリックして掴み、Zoom-In、Zoom-Out や矢印キーを使って上方向に移動する。この際、移動できる方向は上下左右等、決まった方向のみに設定されているが、0 キーを押すことで移動方向を自由にすることができる。改めて 0 キーを押すと元の設定に戻る。今回は真上に移動したいため、垂直に移動し、適当に上に離れた位置で再度クリックするとコピーされたトランジスタのレイアウトが配置される。この状態でコピーされたレイアウトがそのまま選択された状態となっているので、q キーにより Property を開く。誤って選択解除してしまった場合には改めて上の二つのトランジスタを左クリックのドラッグによって同時に選択し q とする。
12. 立ち上がる Edit Instance Properties ウィンドウの上の方にある Common の欄のチェックボックスをチェックする。これにより二つのトランジスタの Property を同時に変更することができる。（Ruler が選択されるとおかしくなる場合は、Ruler だけをまず選択し、Delete キーで削除した後、コピーした二つのトランジスタを改めて選択する。Shift-k で全ての Ruler を消すこともできる。）

13. Common のチェックを入れた後、Attribute タブの Cell 欄を pmos18 に変更する。Parameter タブに移動し、Width 欄を回路図入力の際に用いた 5u に変更する。その後下の OK をクリックする。すると図 15 に示すように、選択していたトランジスタが PMOS に変化し、W が 5μm となる。
14. 後はこれらのトランジスタの接続を完成させれば良いが、その前にトランジスタの基板側の接続を行うための基板コンタクトを作成する。○キーを押すと図 16 に示す Create Via ウィンドウが立ち上がる所以、まず M1\_PACT を選択し、Column を適当に 8 くらいにしたうえで、NMOS の下に配置する。このとき、NMOS PCell の黄色線の NSD と、M1\_PACT コンタクトのピンク線の PSD がぴったり重なるようにおけば良い。次に同じように M1\_NACT を選択し、同様に Column 数を 8 として PMOS の上部分に置く。このときも PSD と NSD がぴったり重なるように置く。ESC キーでコマンドを抜ける。ここで最下部に配置した M1\_PACT 部分が VSS に、最上部に配置した M1\_NACT 部分が VDD に接続されることとなる。
15. さらに、PMOS 側基板コンタクトと PMOS を NWELL レイヤで取り囲む。左側のレイヤ選択部分で NWELL をクリックし選択した状態として、画面上で r キー (Rectangle) を入力する。上部に配置した M1\_NACT を充分取り囲むように NWELL パターンを作成する。パターンを開始したい位置で一度左クリックし、対角線上の位置で再度クリックするとパターンが作成される。間違えてしまった場合は適宜選択したうえで Delete キーで削除できる。ここまでできあがつたレイアウトを図 17 に示す。
16. 次に、各トランジスタの接続を行う。まずはゲートの接続を行う。レイヤ選択部分で GATE レイヤを選択した後 PMOS トランジスタのゲート下端部分に Zoom-In し、p キー (Path) によりパス作成モードに入る。この状態で F3 キーにより作成するパスの太さを数値で指定できるウィンドウが立ち上がる所以、トランジスタのゲートの L に合わせて 0.18 と入力する。デフォルトの太さのまま Path を引いた後、パターンを選択し、q キーによって Property を開いて太さを変更しても良い。ゲートパターンの中心で一度クリックしてまっすぐ下に引いて PMOS と NMOS を接続したいところだが、選択できるグリッドが粗すぎる場合には、一度 ESC でコマンドを抜けて、1 キーを押してグリッドを細かくしてから再度 p コマンドを行う。もう一個の PMOS/NMOS ペアも同様に接続する。p コマンドを使っても良いし、作成した GATE 配線をコピーしても良い。
17. 次にゲートに対して信号電圧を印加するためのコンタクトを作成する。信号は基本的に金属配線 (METAL1 ~5) により伝達されるので、ゲートから METAL1 への接続を作成しておく。先ほどと同様○キーにより Create Via ウィンドウを立ち上げ、M1\_GATE を選択する。先ほど作成した GATE の左右外側にそれぞれ適当な位置に GATE レイヤが重なるように配置する。
18. 次に PMOS/NMOS トランジスタの Source/Drain の接続を行う。レイヤ選択部分から METAL1 を選択し、p キーもしくは r キーを使って回路図通りに接続を行う。ここまでで作成したレイアウトを図 18 に示す。
19. 最後に各端子へのラベル作成を行う。このままではどの配線が回路図のどの端子に対応しているかツールは分かってくれないため適切に端子名を付けてあげる。先ほど作成した回路図上に配置した端子名 (VDD、VSS、INA、INB) を一つずつレイアウト上に配置していく。レイヤ選択部分で METAL1 を選択した状態で 1 キー (Label) を押し、立ち上がる Create Label ウィンドウで Label 欄にまずは INA と入力し、Height 欄に見やすさのため 0.2~0.4 程度の値を入力する。INA と INB はトランジスタのゲートへの入力信号であるので、先ほど配置したゲートコンタクトの METAL1 部分をクリックしてラベルを配置する。図 19 に INA のラベルを作成した例を示す。同様に INB を反対側のゲートコンタクトに配置し、VDD を最上部の M1\_NACT の METAL1 部分に、VSS を最下部の M1\_PACT の METAL1 部分に配置する。これでレイアウトが完成する。

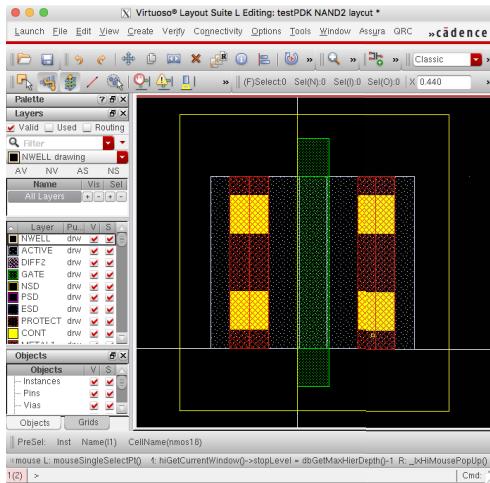


図 12: NMOS のレイアウトを置いたところ

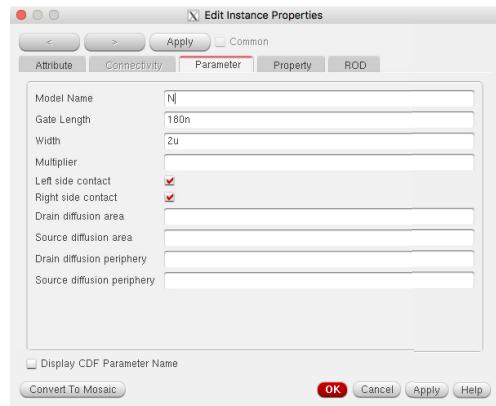


図 13: NMOS Property の変更

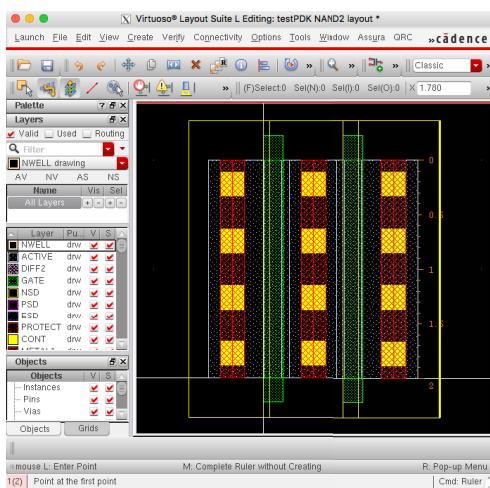


図 14: 二つ目の NMOS を置いたところ

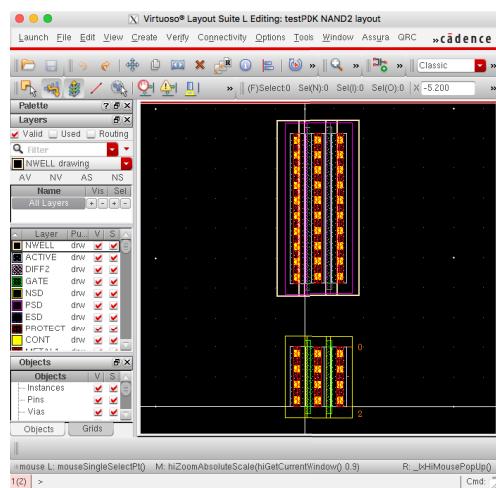


図 15: コピーした NMOS を PMOS に変更した

### Tips:

Display Options ウィンドウから毎回 X/Y Snap Spacing を変更するのが面倒な場合は各自の Home Directory に .cdsenv というファイルを作成し、以下を記述する。(既にある場合はそこに追記する。研究室の設計フローではデフォルトで以下の設定が入っている。)

```
layout xSnapSpacing float 0.02
layout ySnapSpacing float 0.02
```

以上を記述後に Virtuoso を再起動すると、次回以降は起動時の X/Y Snap Spacing が指定した値となる。ROHM 0.18μm プロセスでは 0.02 もしくは 0.04 をデフォルト値として指定しておくのがおすすめである。また、ROHM 0.18μm プロセスでレイアウトを行っていると、Snap Spacing を変更したい場面がしばしば出てくる(レイヤごとに最小グリッドが違うため)。その時にいちいち Display Options ウィンドウを立ち上げて変更するのが面倒であれば、例えば以下の様なコマンドを、Virtuoso を起動するディレクトリの .cdsinit ファイルに追記する。

```
hiSetBindKey("Layout" "<Key>1" "setSnapSpacing(0.010)" )
hiSetBindKey("Layout" "<Key>2" "setSnapSpacing(0.020)" )
hiSetBindKey("Layout" "<Key>3" "setSnapSpacing(0.040)" )
```

以上を記述後に Virtuoso を再起動すると、レイアウトウィンドウ上で “1” キーで X/Y Snap Spacing を 0.01 に、“2” キーで 0.02 に、“3” キーで 0.04 に変更することができるようになる。好みで設定するとよい。ただし上記のショートカットを用いるとしばしば自分が今どのグリッド設定でレイアウトしているのか分からなくなることがあるので注意が必要である。

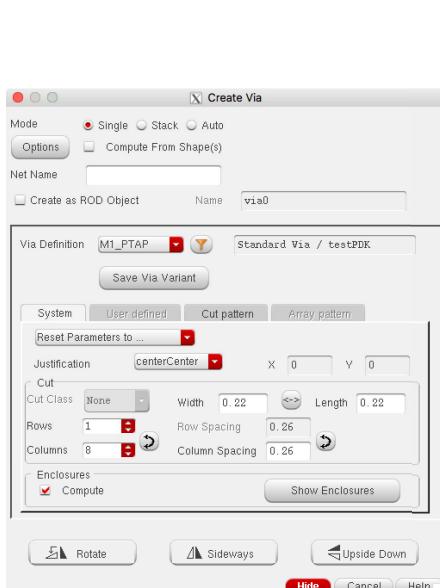


図 16: Create Via ウィンドウ

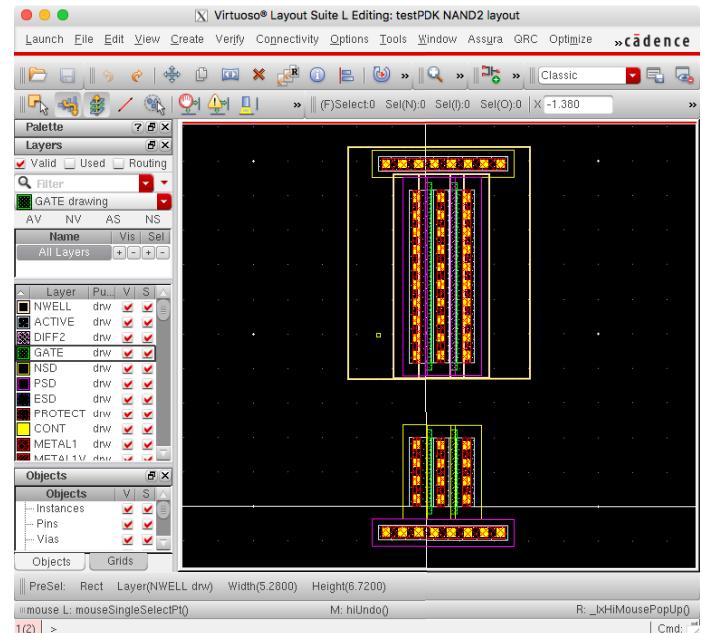


図 17: 基板コンタクトと NWELL まで作成した NAND2 レイアウト

#### Tips:

このトレーニングで使用している ROHM 社の  $0.18\mu\text{m}$  に特有の問題として、レイヤ毎に許される最小グリッドが $0.01$  だったり  $0.02$  だったりすることが挙げられる。(書面で渡される設計ルールドキュメントの PAGE A-6 を参照。) これによって、適当なグリッド設定で設計するとオフグリッドエラーが大量に発生してしまうことがある。通常は  $0.02$  もしくは  $0.04$  くらいのグリッドで設計する事が望ましいが、やむを得ず変更せざるを得ないことがあるので、慣れないいうちは頻繁に DRC チェックをかける等、ミスした状態で設計を進めてエラーをため込まないような配慮が必要になる。慣れてくればそれほど気にならない。

## 8 Calibre Interactive による DRC/LVS の実行

本 PDK により設計した Schematic、Layout の DRC/LVS は Calibre Interactive から GUI を使用して直接実行できる。以下に実行手順について説明する。

### 8.1 前準備

この部分については研究室トレーニングでは既に設定されているためやる必要は無い。ただし、後々のため設定方法について知っておくことは重要であるので、記載しておく。

1. 環境変数に以下の値を設定する。以下は csh/tcsh 系の場合の例

```
setenv MGC_HOME "/usr/CAD/mentor/ixl_cal_2009.2_36.21"
setenv CALIBRE_IGNORE_DRC_CHECK_MAP 1
setenv CDS_Netlisting_Mode "Analog"
```

MGC\_HOME に関しては各自の設定に合わせて Calibre のインストールディレクトリを指定すること。

2. virtuoso を起動するディレクトリに .cdsinit というファイルを作成し(既にある場合には追記する)、以下の行を追加する。

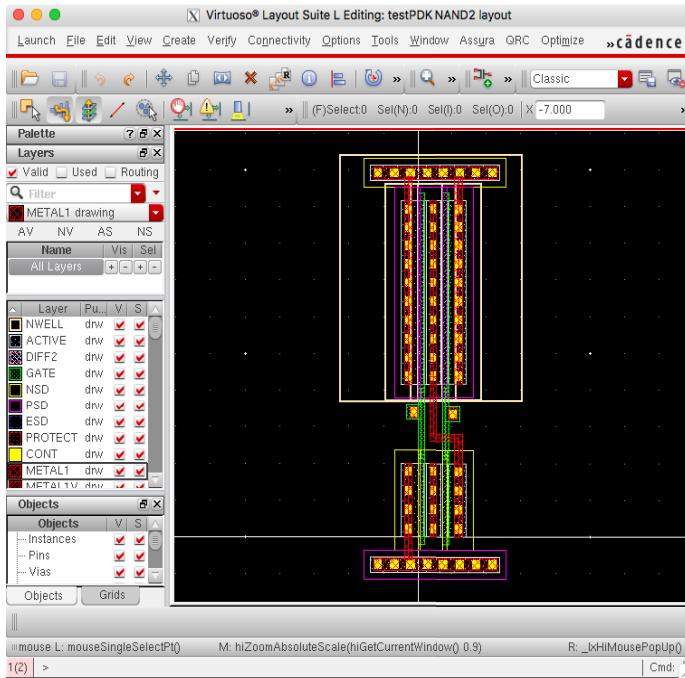


図 18: ゲートコンタクトと配線接続を完了したレイアウト

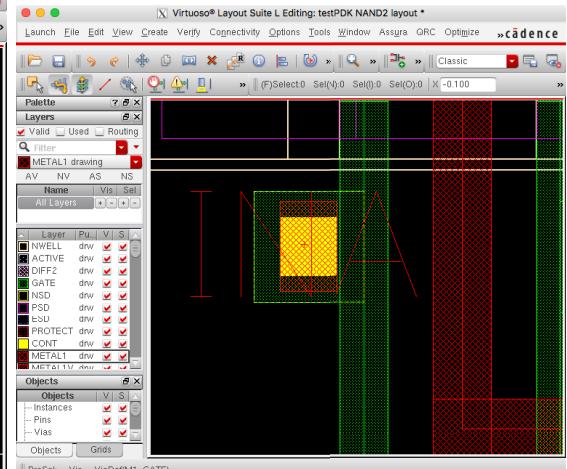


図 19: ラベルの作成

```
load(strcat(getShellEnvVar("MGC_HOME") "/shared/pkgs/icv.ixl/tools/queryskl/calibre.sk1"))
```

- 1 番で設定した環境変数を有効にした(シェルを source し直した)ターミナル上で Virtuoso を起動し直すこと。

## 8.2 DRC の実行

まず、作成したレイアウトに Design Rule 違反が無いかを Calibre DRC (Design Rule Check) を使用してチェックする。

- Virtuoso を起動し、DRC を実行したい layout view を開く。本演習では先に作成した NAND2 のレイアウトを Library Manager から開く。すでに開いていればそのまま進めて良い。
- レイアウトエディタメニューの右上に Calibre というメニューが出ているので、そこから Run DRC (バージョンによっては Run nmDRC) を選択する。
- その後起動するウインドウの Load Runset File の右上の「...」ボタンをクリックし、drcROHM018.cmd を選択する。このファイルに必要な設定が全て書かれている。
- 図 20 に示すウインドウが立ち上がるるので、左の Rules ボタンをクリックし、DRC Rules File に DRC ルールファイルとして /usr1/iizuka/cadence/ROHM018/rules/verification/drc.rul が指定されていることを確認する。(ここに中間ファイルがたくさん出力される。)
- 左の Run DRC ボタンをクリックし Calibre を実行する。
- Calibre 実行終了後、自動的に、図 21 に示す Calibre RVE が起動するのでエラーをチェックする。

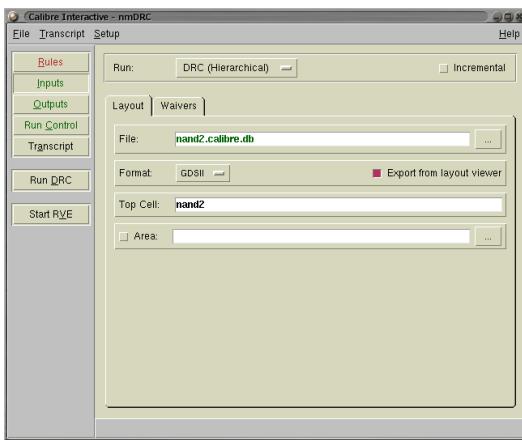


図 20: Calibre Interactive nmDRC 起動画面

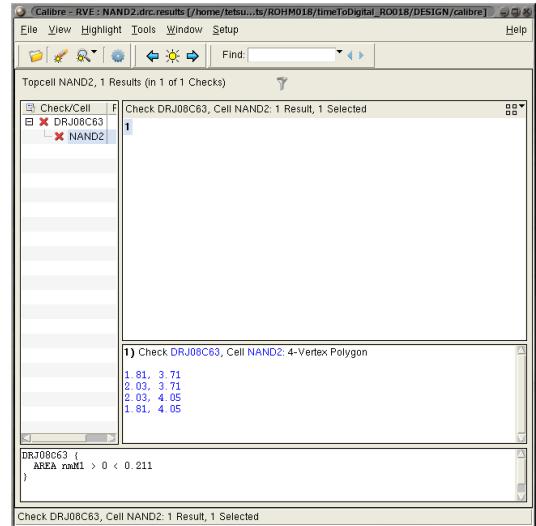


図 21: Calibre RVE による DRC エラーの表示

8. 初めて起動したときのみ、Welcome to Calibre RVE と言う小さいウィンドウが出てくるので、これは OK をクリックして閉じてしまつてよい。
9. エラーがある場合には、Calibre RVE の中央のウィンドウに表示されるエラーレベル番号をダブルクリックすることで、レイアウトエディタ上のエラー箇所が Zoom 表示される。この機能を活用し、紙で配られる Design Rule ドキュメントを参照しながらエラー箇所の特定・修正を行う。
10. レイアウトエディタでエラー箇所を修正したら、再度図 20 のウィンドウで Run DRC ボタンをクリックすると、DRC が再実行される。
11. 以上の操作を繰り返し、DRC エラーが 0 になるまで修正作業を行う。ここでは METAL1 の最小面積違反に関するエラーが残るが、ひとまず無視する。
12. エラーの修正が終わったら、Calibre Interactive を閉じる。Calibre Interactive のウィンドウのメニューから File > Exit とするか、ウィンドウ右上の×印をクリックする。
13. Runset File を保存しますか? と聞かれるので、特に設定を変えた場合などはここで Yes として、設定ファイルを保存しておくと以後は Calibre Interactive 起動時に保存した設定ファイルを読み込むことでルールファイルの指定などを行えるので便利である。
14. その他、DRC の階層実行や、Multi-Thread の設定等を Calibre Interactive の GUI から設定できるので、いろいろといじってみるとよい。変更した設定は上記の Runset File に保存することができる。

### 8.3 LVS の実行

Design Rule 違反が無いことが確認できたら、続いて Calibre LVS (Layout Versus Schematic) を用いてレイアウトと回路図が等価なものになっているかを確認する。

1. LVS を実行したい layout view を開く。この時 LVS 対象となる schematic view が同じ Cell 名で存在する必要がある。本演習では作成した NAND2 回路を対象とする。既に DRC の実行時に開いている場合はそのまままでよい。
2. レイアウトエディタの右上に Calibre というメニューが出ているので、そこから Run LVS (バージョンによっては Run nmLVS) を選択する。

3. その後起動するウィンドウの Load Runset File の右上の「...」ボタンをクリックし、lvsROHM018.cmd を選択する。このファイルに必要な設定が全て書かれている。
4. 図 22 に示すウィンドウが立ち上がるるので、左の Rules ボタンをクリックし、LVS Rules File に LVS ルールファイルとして /usr1/iizuka/cadence/ROHM018/rules/verification/lvs.rul が指定されていることを確認する。
5. LVS Run Directory に、Calibre を実行するディレクトリとして ./calibre が指定されていることを確認する。(ここに中間ファイルがたくさん出力される。)
6. 左の Inputs ボタンをクリックし、右側のウィンドウの Netlist タブ内の 2 段目の Export from schematic viewer にチェックが入っていることを確認する。
7. 左の Run LVS ボタンをクリックし Calibre を実行する。
8. Calibre 実行終了後、自動的に図 23 に示す Calibre RVE が起動するのでエラーをチェックする。
9. Calibre RVE に表示されているログに CORRECT と示されていて、緑色の笑顔マークが表示されれば LVS が Pass している。エラーがある場合には、図 23 に示すようにエラーが表示されている。
10. エラーメッセージをクリックし、Discrepancy (不一致) 箇所を特定する。右側中段のメッセージ中にクリックできる箇所があるので、そこをクリックすると、レイアウトエディタ上の不一致箇所がハイライト表示されるので、この機能を駆使して LVS エラーを修正する。
11. レイアウトを修正したら、再度図 22 のウィンドウで Run LVS ボタンをクリックすると、LVS が再実行される。
12. 以上の操作を繰り返し、LVS 結果が CORRECT になるまで修正作業を行う。
13. エラーの修正が終わったら、Calibre Interactive を閉じる。Calibre Interactive のウィンドウのメニューから File > Exit とするか、ウィンドウ右上の×印をクリックする。
14. Runset File を保存しますか? と聞かれるので、特に設定を変更した場合などはここで Yes として、設定ファイルを保存しておくと以後は Calibre Interactive 起動時に保存した設定ファイルを読み込むことでルールファイルの指定などを行えるので便利である。
15. その他、LVS の階層実行や、Multi-Thread の設定等を Calibre Interactive の GUI から設定できるので、いろいろといじってみるとよい。変更した設定は上記の Runset File に保存することができる。

## 9 Star-RCXT による寄生成分抽出 (LPE/PEX) の実行

ROHM 社 0.18μm プロセスではレイアウトの寄生成分（容量、抵抗）の抽出のために、Synopsys 社の Hercules/Star-RCXT というツールを用いる。通常はコマンドラインから実行する事が多いツールであるが、当研究室では Virtuoso のメニューから実行可能としたスクリプトを作成したため、その使用方法を以下で説明する。

1. LPE (Layout Parasitic Extraction) /PEX (Parasitic EXtraction) を実行したい layout view を開く。この時 LVS 対象となる schematic view が同じ Cell 名で存在する必要がある。本演習では作成した NAND2 回路を対象とする。既に DRC/LVS の実行時に開いている場合はそのままよい。
2. レイアウトエディタの右上に VDEC というメニューが出ているので、そこから Run Star-RCXT LPE... を選択する。

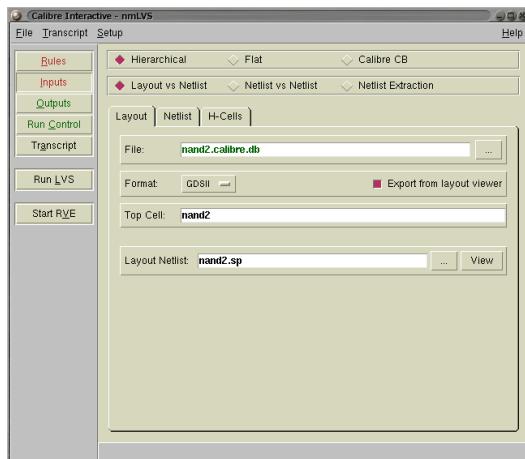


図 22: Calibre Interactive nmLVS 起動画面

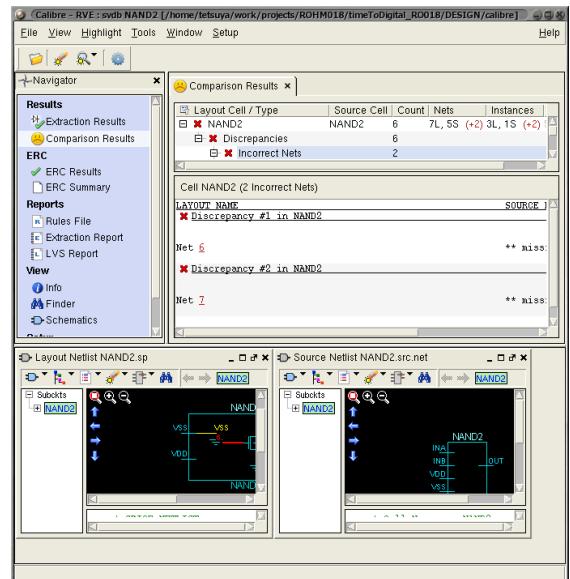


図 23: Calibre RVE による LVS エラーの表示

3. 図 24 に示すウィンドウが立ち上がる。基本的な設定は全て自動的に入力されているので、下部上段の Execute Hercules LVS/Star-RCXT LPE ボタンをクリックすることで寄生成分抽出が実行できる。Virtuoso を起動したときに最初に立ち上がる横長のウィンドウにいくつかログメッセージが表示され、コマンドが実行される。ここでは変更する必要は無いが、LPE Extraction Mode を RC とすることで、寄生抵抗の抽出まで実行できる。また LPE Precision の数字を変更することで容量抽出の精度を低く (100) または高く (400) することができる。
4. 実行が終わると、Hercules/Star-RCXT LVS/LPE Program Completed Successfully というメッセージが小さいウィンドウで表示されるので OK をクリックする。上手く行かない場合は、図 24 のウィンドウ下部の Show Log File や LVS Errors ボタンをクリックし、エラー内容の確認などを行う。上手く行った場合は Show LPE Netlist ボタンをクリックすることで生成されたネットリストを確認する事ができる。回路図から生成したネットリストではトランジスタ (m から始まる行) しか含まれていなかつたが、配線に起因する容量 (c から始まる行) が追加されていることが分かる。

## 10 寄生成分抽出後のシミュレーションの実行

前節で生成した寄生成分抽出後のネットリストに対して、改めてシミュレーションを実行し、性能の変化を確認する。今回の NAND2 の例では遅延が僅かに変化する事が予想されるのでそれを確認する。アンプ等のアナログ回路ではバンド幅や安定性に大きな変化が生じることがあるのでより詳細な検証が必要となる。

1. 抽出したネットリストは、Virtuoso を起動したディレクトリの下に、star/ライブラリ名/セル名/セル名.spf という名前で保存される。今回の NAND2 の例では

```
% ~/design/star/testPDK/NAND2/NAND2.spf
```

という名前で保存されているはずである。同じディレクトリに Star-RCXT の実行ログなどが保存されている。

2. 回路図作成後にシミュレーションを実行したディレクトリに移動し、まずは先ほど行ったシミュレーション結果を別のファイル名でバックアップする。

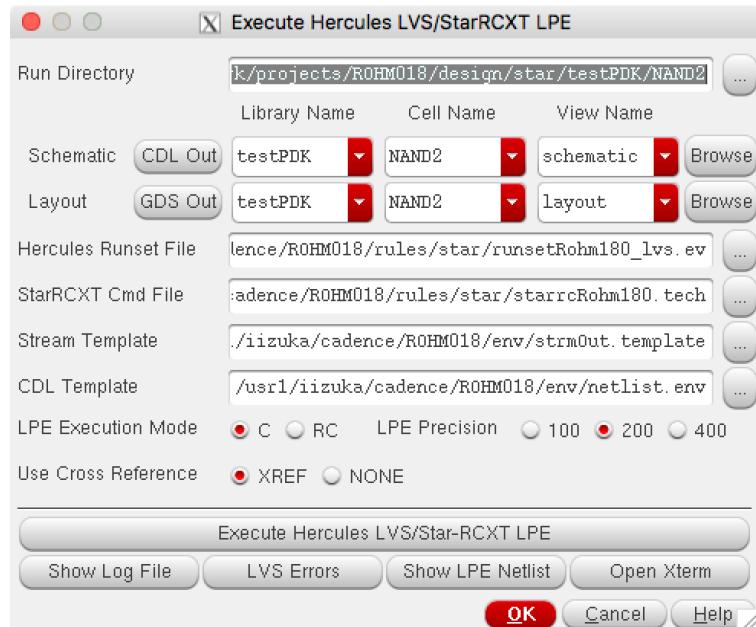


図 24: VDEC Star-RCXT 実行メニュー画面

```
% cp NAND2.tr0 NAND2_pre.tr0
% cp NAND2.fsdb NAND2_pre.fsdb
```

(pre はレイアウトの「前」という意味。逆にレイアウト後のシミュレーションを post simulation と呼んだりもする。)

3. HSPICE/HSIM 入力ファイル内のネットリスト読み込み部分を書き換え、抽出後のネットリストを読み込むように修正する。具体的には最後の部分の .include で始まる行を以下の様に書き換える。

```
.include "~/design/star/testPDK/NAND2/NAND2.spf"
```

これにより、シミュレーション実行時に抽出後のネットリストが読み込まれる。

4. 第 3.3 節を参考に、HSPICE もしくは HSIM シミュレーションを実行し、wv 等の波形ビューワでレイアウト前後の波形を読み込み比較する。

## 11 Virtuoso Layout GXL によるレイアウト設計

設計した 2 入力 NAND 回路のレイアウトを別の方法で作成してみる。

先に説明した通り、Library Manager 上で File > New > Cell View として New File ウィンドウから layout ビューを作成してもよいが、PDK を使用したフローでは、Virtuoso Layout GXL という高機能版のレイアウトエディタを使用して、接続関係を確認しながらレイアウトを行うことができる。ここでは、Virtuoso Layout GXL を使用した Connectivity-Driven レイアウト手法について解説する。以下で解説するいくつかの機能は Virtuoso Layout L では有効にならないため、注意が必要である。

以下で Virtuoso Layout GXL を使用したレイアウト作成手順について示す。

1. まず、前節で作成したレイアウトをバックアップする。Library Manager 上で先ほど作成した layout view を選択し、右クリックで出てくるメニューから Rename... を選択し、To View の欄に layout\_backup (バックアップであることが分かれば名前は何でも良い) と入力して OK とする。

2. 先ほど作成した 2 入力 NAND 回路の Schematic Editor ウィンドウから、メニュー左上の Launch > Launch Layout GXL として立ち上がる Startup Option ウィンドウで Layout に Create New を、Configuration に Automatic を選択し OK とする。
3. 次に立ち上がる New File ウィンドウから、Library に testPDK、Cell に NAND2、Type に layout が選択されていることを確認して OK をクリックする。
4. すると、Schematic Editor ウィンドウとともに、新規に Layout Editor のウィンドウが立ち上がる。ライセンスに関するウィンドウが立ち上がった場合には、Yes または Always をクリックする。
5. Layout Editor ウィンドウ下、一番左側の Generate All From Source ボタンを押して図 25 に示す Generate Layout ウィンドウを立ち上げる。
6. I/O Pins の Layer のところで METAL1 を選択して Apply ボタンを押すと、その下にリスト表示されている端子の Layer 欄が全て METAL1 になる。端子ごとにレイヤを変更したい場合にはリストの中の変えたい端子を選択して、下の Layer から必要なレイヤを選択し、Update をクリックすると個別に変更することが可能である。今回は全ての端子を METAL1 で作成する。
7. さらに、その下の Pin Label Shape: で、Label を選択し、右側の Pin Label Options... で、どのレイヤで Pin Label を作るかを選択する。ここでは、Height を 0.2 とし、Layer Name、Layer Purpose を両方共 Same As Pin に変更する。その他の設定は好みで変更してよい。
8. この状態で OK をクリックするとレイアウト上に Schematic Editor に配置した必要なトランジスタおよびピンが作成される。この時、Layout Editor 上に配置されるトランジスタのサイズは、自動的に Schematic Editor で入力したサイズになっている。(デフォルトで Display の Stop Level が 0 になっているため、トランジスタの中身が見えない場合には、“d”キーを押して立ち上がるウィンドウの左下の Display Levels の Stop Level を 32 等に変更すること。)
9. この時、CIW 上にいろいろと Warning メッセージが出るが、これらは無視してよい。
10. この操作は ROHM 0.18μm 特有のものであるが、メニューから Options > Display を選び(または “d” キーを押して)、Display Options を立ち上げ、右上の X Snap Spacing, Y Snap Spacing を 0.02(もしくは 0.04) としておく。**ここは必ず 0.02 の倍数にすること。**これは、オフグリッドエラーを防ぐためである。

以上の操作で、図 26 に示すように、Schematic Editor 上で配置したような位置にトランジスタが配置される。水色の枠は Boundary 用のレイヤで、今のところは特に気にしなくてよいが、ひとまずこの枠の中でレイアウトを行うものと思えばよい。(枠のサイズは後で自由に変更できるので、あまり気にしなくてよい。)

あとは、今までどおりのレイアウト手順で、**m** キーによる Move や **r** キーによる Rectangle、**p** キーによる Path を駆使して NAND2 のレイアウトを作成すればよいが、以下では Layout GXL から使用可能になつたいくつかの便利な機能を紹介しながらレイアウトを作成していく。実際には各自の好みなどもあるので、自分のやりやすいように適宜必要な機能を使用してレイアウトを行えばよい。

## 11.1 Connectivity Driven レイアウト

Layout GXL では、トランジスタや端子の接続関係を確認しながらレイアウトを行うことができる。以下の手順でその様子を確認する。

1. 自動で生成されたレイアウト中の右側の PMOS トランジスタを Move コマンド (“m” キー) で Move してみる。すると、図 27 に示すようにトランジスタの各端子から、オレンジ色の線(ライトライン)が表示される事が分かる。この線が各端子の接続関係を示しており、この線にしたがってレイアウトを行えば基本的には LVS エラーは発生しないはずである。

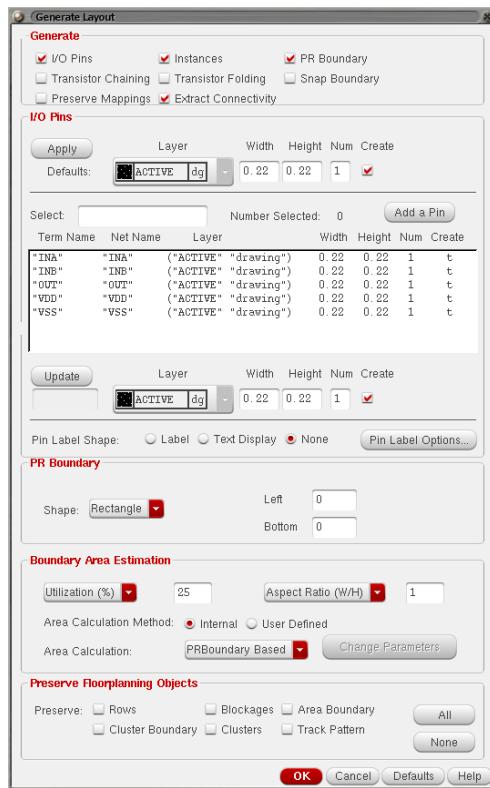


図 25: Generate Layout ウィンドウ

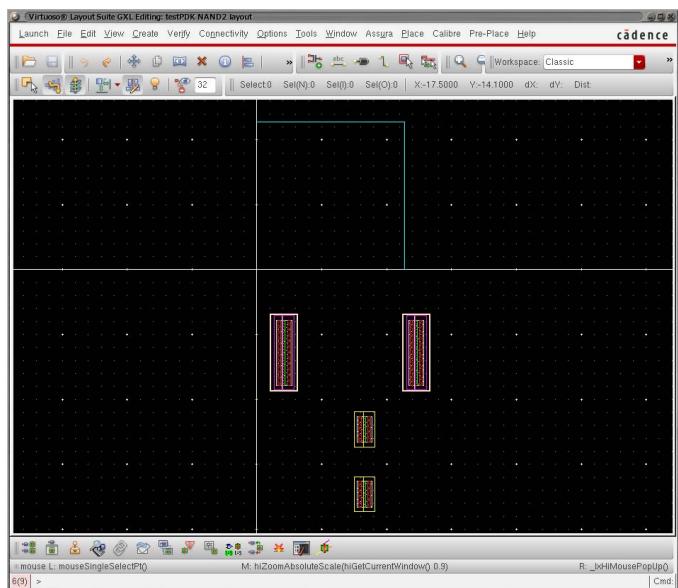


図 26: 生成された NAND2 のレイアウト

2. Move した PMOS トランジスタは適当な位置に置いてしまうか、エスケープキーで Move コマンドを抜けておく。
3. 次に、レイアウトウィンドウの上部のメニューの右端、Help ボタンと cadence ロゴの間の何もないところでマウスの右クリックをすると、図 28 のようなメニューが現れるので、ここから Annotation Browser を選択する。すると図 29 に示すように、まだ接続が完了していない端子がリスト表示される。このウィンドウを用いて端子の接続関係を確認することができる。
4. Annotation Browser を起動するとレイアウト領域が狭くなるので、レイアウトしづらくなる。閉じたいときは Annotation Browser と描いてある右上の位置の×印をクリックすれば閉じることができる。本演習では Annotation Browser は立ち上げたままにしておくことをおすすめする。

## 11.2 Design Rule Driven (DRD) レイアウト

Layout GXL では、Design Rule を確認しながらレイアウトを行うことができる。以下の手順で、Design Rule Driven レイアウトを有効にし、DRD レイアウトの様子を確認する。

1. 次に、レイアウトエディタの原点付近を拡大する(マウスの右ボタンダラッグで領域選択)。すると、図 30 に示すように、2 入力 NAND 回路に必要なメタル端子が配置されていることが分かる。このうちの一つのメタルの Rectangle を Move コマンドで移動して他の Rectangle に近づけてみる。
2. 前節の様にフライトラインが表示される事が分かる。ひとまずエスケープキーで Move コマンドを抜ける。
3. 次に、レイアウトエディタのメニューの Options > DRD Edit... から、図 31 に示す DRD Options ウィンドウを表示する。一番上の Interactive Mode が Off になっているところを Notify に変更し、OK をクリックして DRD Options ウィンドウを閉じる。

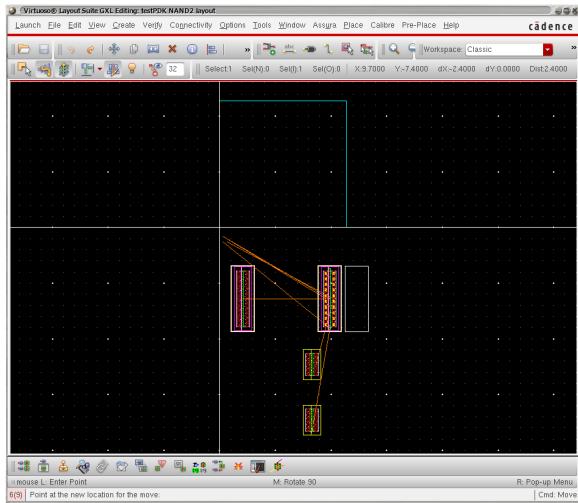


図 27: 配線接続のフライトライン

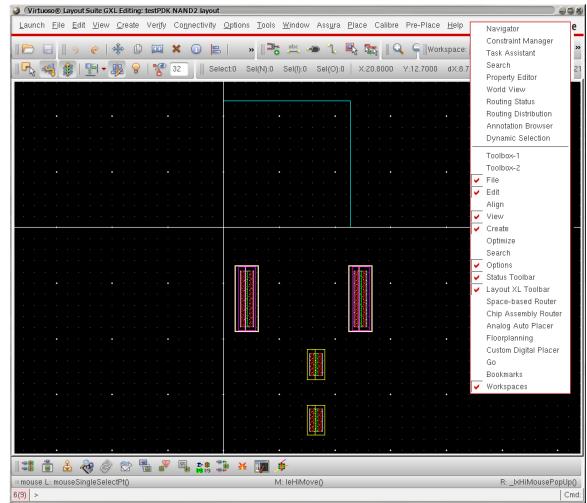


図 28: 右クリックでメニューを表示

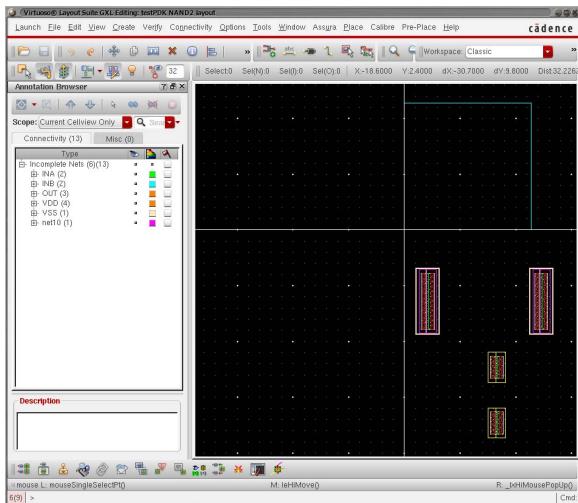


図 29: Annotation Browser の起動

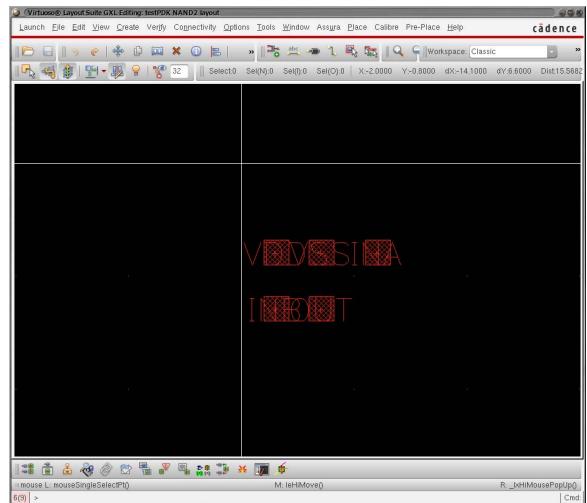


図 30: ピンの周辺を拡大

4. 再度、メタル端子のうち、一つの Rectangle を Move コマンドで移動して他の Rectangle に近づけてみる。図 32 に示すように、二つのメタル間の距離が Design Rule 以下になった場合に、DRC 違反箇所を表示してくれる事が分かる。ついでに、端子の Rectangle の minArea 違反も表示されている。
5. 次に、再度レイアウトエディタのメニューの Options > DRD Edit... から、DRD Options ウィンドウを表示し、Interactive Mode を Notify から Enforce に変更する。この状態で再度、Move コマンドでメタル端子のうち、一つの Rectangle を移動して他の Rectangle に近づけてみる。Design Rule で指定された minSpacing よりもメタル端子を近づけようとしても、それ以上メタルレイヤが移動しない事が分かる。Enforce の設定では、デザインルールに違反する様な距離にそもそもメタルを置くことを許可しない。
6. 本書では、Interactive Mode は Notify をおすすめしておく。(Enforce では「Design Rule 違反だけど、一旦ここに置いておいて...」というレイアウトもできないので不便な事があるため。)好みによって設定を行うとよい。また、DRD Options ウィンドウの Post Edit Checking というチェックボックスがあるが、これをチェックしておくと、Move や Copy の最中だけでなく、レイヤを配置した後もデザインルール違反を表示する。こちらも好みによって使用するとよいが、余計な違反まで表示することが多く、本書では Post Edit Checking は使用しないことをおすすめしておく。

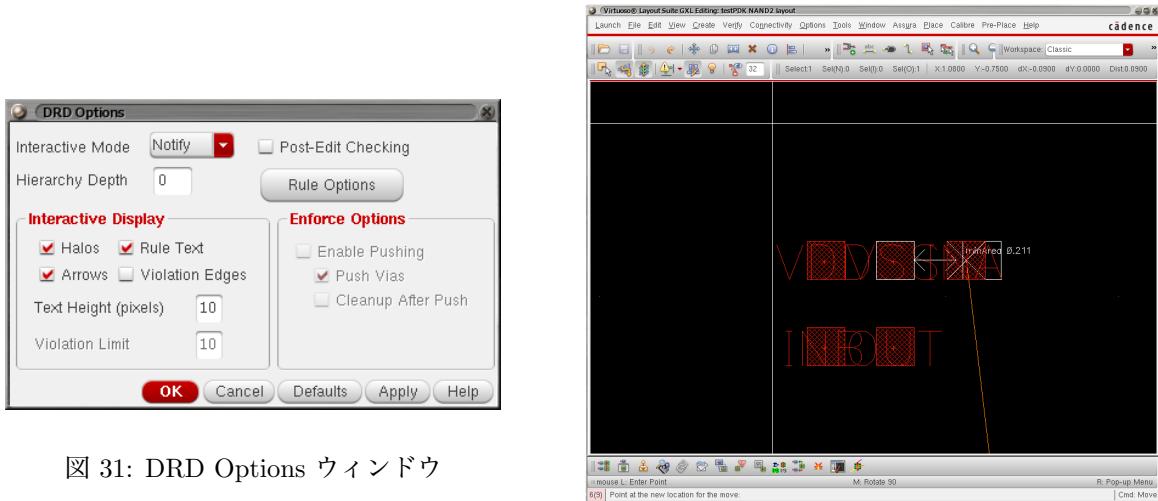


図 31: DRD Options ウィンドウ

図 32: Design Rule 違反箇所の表示

- 今後の演習のために、再度レイアウトエディタのメニューの Options > DRD Edit... から、DRD Options ウィンドウを表示し、Interactive Mode を Enforce から Notify に戻しておく。また、Design Rule の情報を見ながら階層を降りて表示するかどうかを、同じ DRD Options ウィンドウの Hierarchy Depth で設定しているので、必要に応じて値を増やしておくとよい。本演習では後々階層設計について触れるので、値を 10 程度にしておくとよい。

#### Tips:

DRD Options ウィンドウから毎回 DRD の設定を変更するのが面倒な場合は各自の Home Directory に .cdsenv というファイルを作成し、以下を記述する(既にある場合はそこに追記する)。

```
layout drdEditMode cyclic "notify"
layout drdEditDisplayMarkers boolean nil
layout drdEditHierDepth int 10
```

以上を記述後に Virtuoso を再起動すると、次回以降は起動時の DRD Options の Interactive Mode の設定が Notify になり、Post Edit Checking が Off になり、Hierarchy Depth が 10 になる。

### 11.3 Auto Abutment

Layout GXL では、接続関係のある二つのトランジスタを接触して配置すると自動でソース・ドレインをマージする Auto Abutment (アバットメント) 機能がある。この機能について確認する。

- 図 33 に示すように、右側の PMOS レイアウトを Move コマンドで水平に移動し、左側の PMOS レイアウトに近づける。
- ソース・ドレインの領域が重なるようにして、そこに配置する。(このとき、ソース・ドレインの重なりがぴったり重なっていなくてもよい。)
- PMOS を配置すると、図 34 に示すようにソース・ドレインが共有される。

このとき、二つの PMOS で共有されたソース・ドレイン端子のコンタクトがそのまま配置された状態で Auto Abutment がなされる点に注意が必要である。NMOS 側の Auto Abutment については、次節の Align コマンドを実行した後に行う。その時との挙動の違いに注目すること。

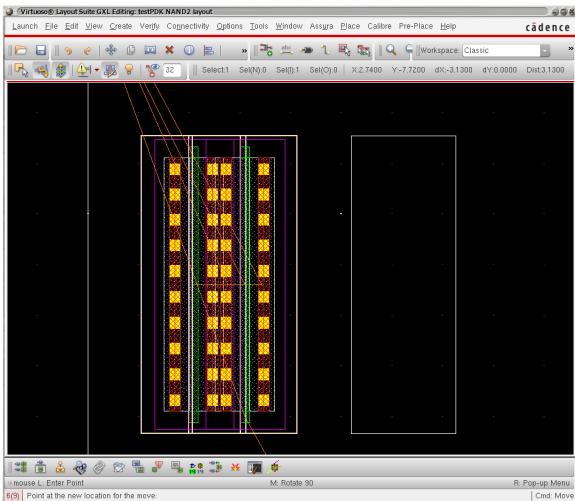


図 33: PMOS トランジスタを Move して近づける

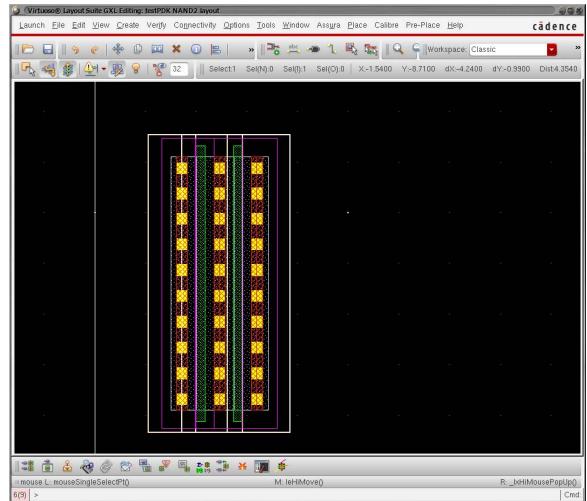


図 34: 近接して配置すると自動で Merge する

## 11.4 Align コマンド

レイアウトを行っていると、複数のパターンを規則正しく並べなければいけない場面が多くある。例えば今回の設計例では、現状縦に並んでいる二つの NMOS トランジスタを横に高さ方向を揃えて並べたい。この時、普通は片方の NMOS トランジスタを Move して、もう片方の NMOS トランジスタときちんと並ぶように画面の拡大・縮小を駆使して移動させていくことが通常である。しかし、並べたいオブジェクトが多数に渡る場合等には作業が煩雑になる。Layout GXL では、新たに Align コマンドが使用できる。Align コマンドの使用法について以下に簡単に説明する。

1. Align コマンドの効果をわかりやすくするため、上側の NMOS トランジスタを図 35 に示すように、Move コマンドで少し左側に移動させておく。
2. レイアウトエディタのメニューから Edit > Advanced > Align... と実行し、図 36 に示すウィンドウを立ち上げる。
3. Align ウィンドウの上半分の Reference の欄で、Object を選択し Instance を選択する。さらに、Align Direction から Horizontal を選択し、その下の Use に Top Edge を選択する。
4. Set New Reference ボタンを押した後、レイアウトエディタ上で、先ほど移動した NMOS トランジスタを選択すると、図 37 に示すように、Align コマンドの Reference となる位置がピンク色の線で表示される。
5. Align ウィンドウに戻り、中央の Align Using の欄で Object と Instance を選択し、Use に Top Edge を選択する。
6. 再度レイアウトエディタ側に戻り、並べたいトランジスタ（ここでは、下側のトランジスタを左クリックで選択する）。
7. Align ウィンドウの Apply ボタンをクリックすると、図 38 に示すように二つの NMOS トランジスタが正しく並べられる。
8. エスケープキーを押し、Align コマンドを抜ける。

実際には 2 つのトランジスタを並べる程度の作業には、直接 Move コマンドを使用した方が早いと思われるが、数多くのオブジェクトを並べたり、一定の Pitch で並べたりするような場合には、Align コマンドが便利になってくる。状況に合わせて使用するとよい。

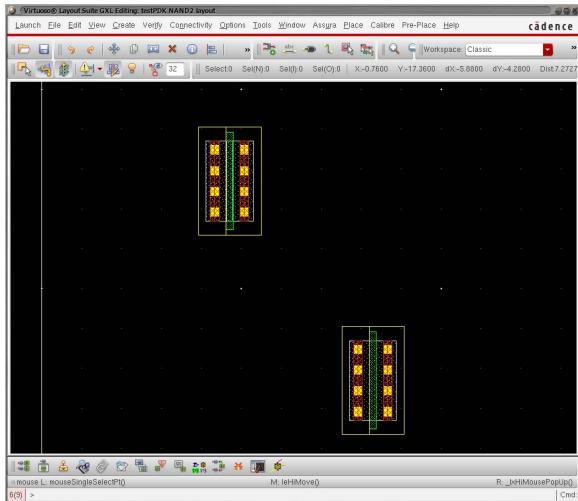


図 35: 上側の NMOS トランジスタを少し左に Move しておく

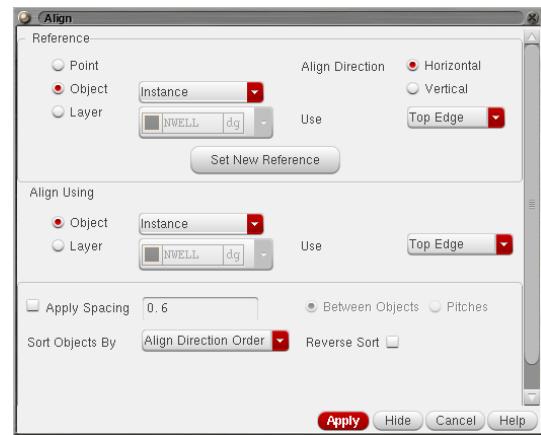


図 36: Align ウィンドウ

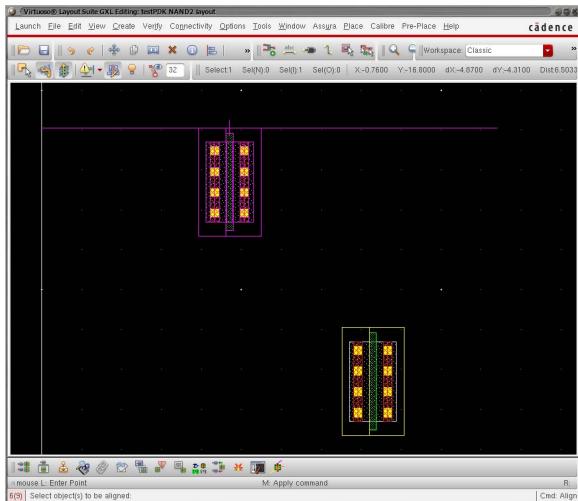


図 37: Reference の設定

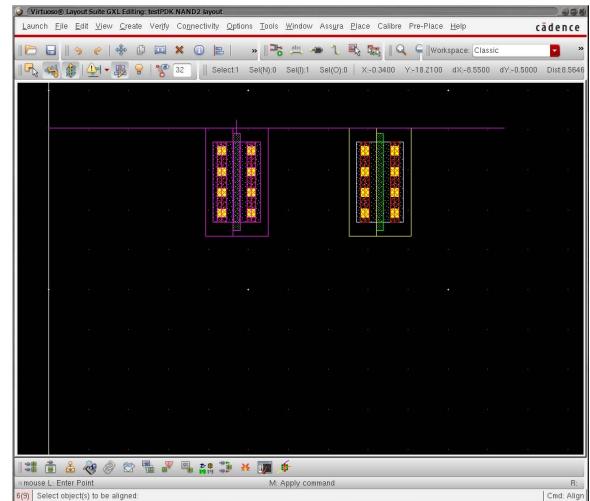


図 38: NMOS トランジスタが並べられた

ここで、NMOS トランジスタについても Auto Abutment を行ってみる。先ほど並べたトランジスタの片方を Move して、もう一方のトランジスタにソース・ドレインが重なるように配置する。すると、図 39 に示すように二つのトランジスタのソース・ドレインが共有される。この時、PMOS トランジスタの場合と異なり、共有された拡散領域のコンタクトが消え、二つのゲート間の距離が自動的に最小スペースとなっている事が分かる。これは、NMOS 側の二つのトランジスタ間の接続は直列接続であり、他の接続が存在しないことから、コンタクトが不要であると自動で判断されるためである。これにより二つのトランジスタ間の不要な寄生容量を削減することができる。(Auto Abutment を使用する際は、実は上下の位置がきちんと合っていなくても、勝手に並べてくれる。そのため、実際には Align 作業できちんと位置合わせを行わなくても、Move コマンドで拡散領域を適当に重ねて配置すれば、二つのトランジスタはきちんと並ぶ。)

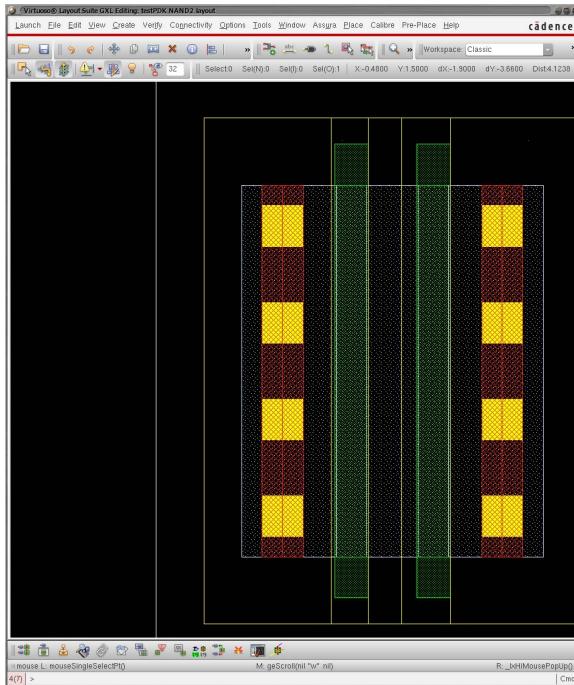


図 39: NMOS の Abutment による接続

#### Tips:

Auto Abutment で拡散共有を行う場合に、上記の NMOS トランジスタのケースの様に、直列接続したトランジスタの拡散共有が最小幅で行われてしまい、レイアウトの規則性や容易性が損なわれて、気に入らないという人がいるかと思われる。(自分がそうである。)

これを避ける方法は今のところ二つある。

一つは、Auto Abutment の機能自体を無効にしてしまうことである。Auto Abutment 機能はレイアウトエディタのメニューから Options > Layout XL... と実行し、Layout XL Options ウィンドウを開き、その中の Generation タブの中央の Abut transistors のチェックを外すことで無効にできる。この方法の欠点は、レイアウト中の全てのトランジスタの Auto Abutment が無効になってしまうため、

- 共有した拡散にコンタクトが必要な場合 → Move 時にきちんと位置を合わせて配置する
- 共有した拡散にコンタクトが必要ない場合 → Move 時にきちんと位置合わせをした後、トランジスタの Properties の Parameter タブの中の Left side contact または Right side contact のチェックを適宜外して不要なコンタクトを外す

といった作業が必要となることである。

もう一つの方法は、Auto Abutment をしたくないトランジスタの P-Cell を Flatten することで、ただの Rectangle の集まりにしてしまうことである。方法は、Flatten したいトランジスタを選択した後、レイアウトエディタのメニューから Edit > Hierarchy > Flatten... と実行して起動する Flatten ウィンドウから、Flatten Mode に One Level、Flatten PCell にチェックを入れて OK とする。

この方法の欠点は、PCell を Flatten してしまうと、Connectivity の情報が使用できなくなることと、サイズを変更する場合等には PCell の Parameter として編集することができなくなる点である。

ここでは、上記の作業は行わず、最小間隔で拡散が共有された形でレイアウトを行う。

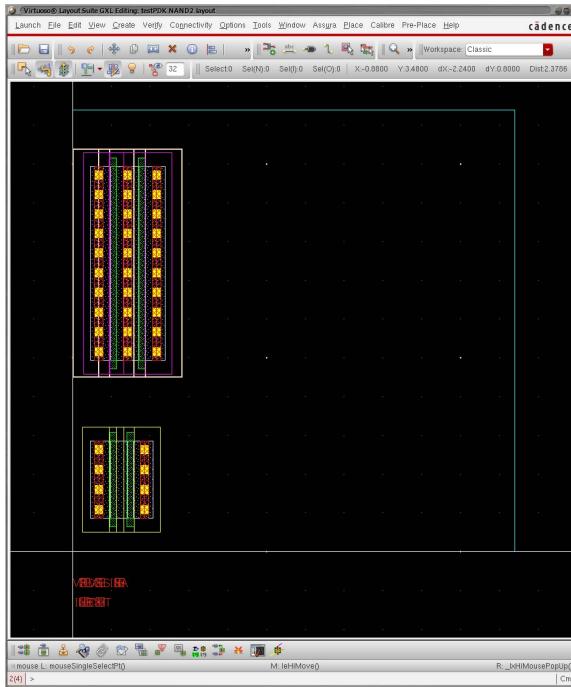


図 40: トランジスタの配置

## 11.5 Multi-Part Path の作成

ここから、いよいよ NAND のレイアウトを作成していく。先ほどまでの作業で Auto Abut された二つの PMOS および、二つの NMOS を、図 40 に示すように、水色の Boundary の中に収まるように Move コマンドを使用して移動する。本演習では細かい座標を特に指定しないので、各自適当な位置に配置すること。オフグリッドエラーを避けるため、Move を行う前に X/Y Snap Spacing が 0.02 もしくは 0.04 になっていることを Options > Display で確認すること。

次に、基板コンタクトのレイアウトを作成する(実際には作業の順番はどうでもよい)。今まででは、基板コンタクト用のインスタンスを作成し、それをアレイで並べる等の作業を行うことが多かったが、新しくリリースされたテクノロジファイルを使用すると、Multi-Part Path という特殊な Path を作成することができる(Layout GXL だけでなく、Layout L 上でも作成可能)。この Path を使用して基板コンタクトを作成する。

1. レイアウトエディタ上のメニューから Create > Multipart Path と実行する。
2. 何も起きてないように見えるが、レイアウトエディタ右下隅に Cmd: Multipart Path と表示され、コマンドが有効になっている事が分かる。
3. この状態で F3 キーを押して、図 41 に示す Multipart Path のメニューを表示させる。
4. 右上の MPP Template から、PguardRingWideli を選択する。
5. 図 42 に示すように、基板コンタクトを置きたい位置に通常の Path を引く要領で Multipart Path を作成する。本例では X 軸に沿うように適当な長さ引いた。さらに、Design Rule 違反を避けるため、NMOS トランジスタの NSD レイヤが Multipart Path の PSD レイヤと接するように NMOS トランジスタを移動しておいた。
6. 続いて N-Well コンタクトを作成する。同じ要領で、Create Multipart Path ウィンドウから、MPP Template で NguardRingWideli を選択する。

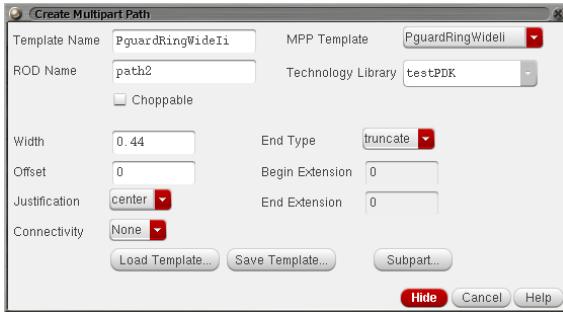


図 41: Create Multipart Path メニュー

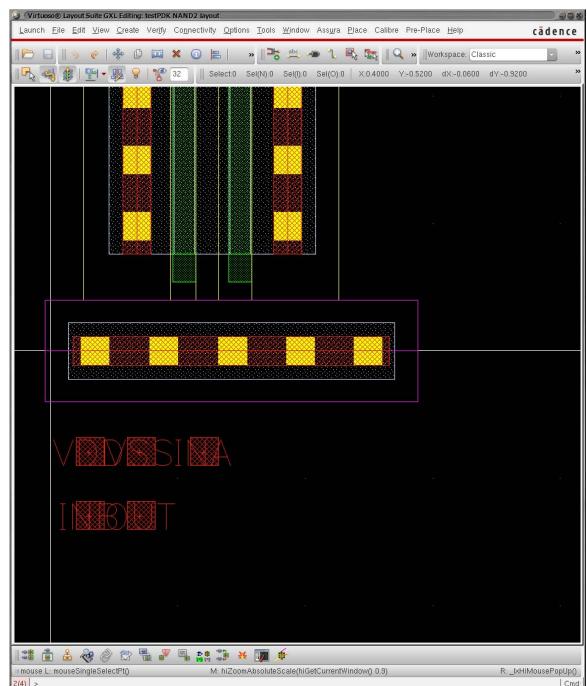


図 42: 基板コンタクト側 Multipart Path の作成

7. さらに図 43 に示すように、N-Well コンタクトを Path を引く要領で作成する。本例では PMOS の PSD レイヤと Multipart Path の NSD レイヤがちょうど接するように Y 方向の位置を合わせ、また、N-Well レイヤがちょうど PMOS の左右の N-Well レイヤと接するように X 方向の長さを調整して引いた。
8. エスケープキーで Multipart Path コマンドを抜ける。

以上の手順で、図 44 に全体像を示すように基板コンタクトを作成する。Multipart Path は通常のパスと同じように **m** キーによる Move や **s** キーによる Stretch ができるので、それらのコマンドを駆使して好きな位置に調整すること。

#### Tips:

メニューから Create > Multipart Path... と毎回実行するのが面倒な場合は Virtuoso を起動するディレクトリにある .cdsinit に以下を追記しておく。

```
hiSetBindKey("Layout" "Ctrl<Key>m" "leHiCreateMPP()")
```

以上を記述後に Virtuoso を再起動すると、次回以降は Ctrl-m キーで Multipart Path コマンドが実行される。キー設定は各自のお好みに合わせて設定すること。

## 11.6 Create Wire による配線接続

続いて各端子の間の配線接続を行う。従来どおり **p** キーによる Path や **r** キーによる Rectangle を用いて各端子間を接続してもよいが、Layout GXL では Create Wire という新しい配線方法があるのでその使用方法について紹介する。

1. メニューから、Create > Wire (または Ctrl+Shift+w) Create Wire コマンドを実行する。
2. 接続したい端子をクリックする。例えば NMOS の左側の端子をクリックしてみると、するとどのレイヤで接続するかの選択肢が出る(図 45)ので、METAL1 を選択して OK をクリック。
3. Path を引く要領でマウスを移動させると、図 46 に示すように、選択したレイヤで Path が引かれる。このとき接続先へのフライラインが同時に表示される。

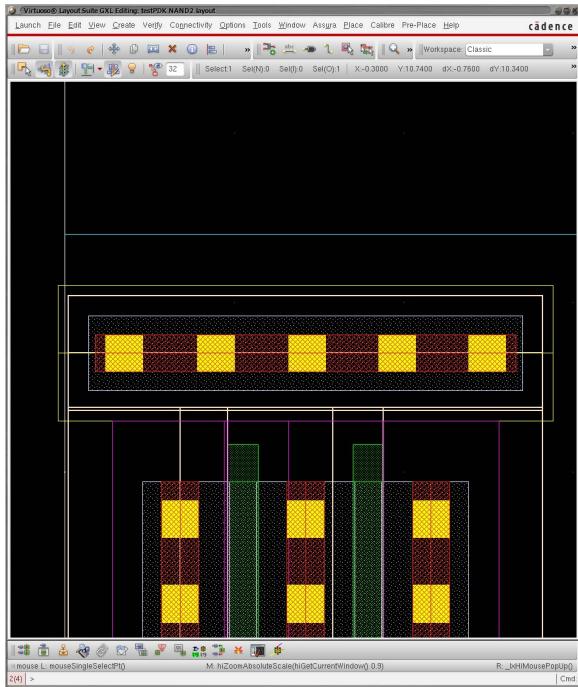


図 43: N-Well コンタクト側 Multipart Path の作成

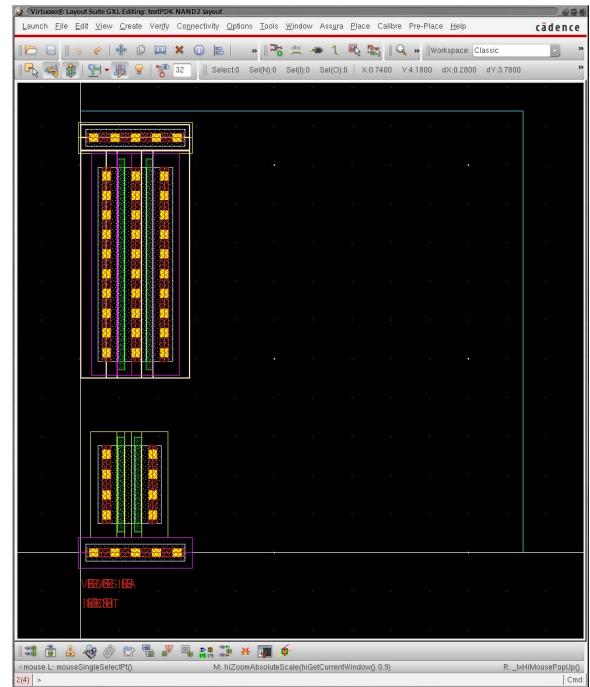


図 44: 基板コンタクト作成後 NAND2 レイアウト

4. 試しに、フライラインが表示されている先ではない間違った端子に配線をつないでみる。すると、図 47 に示すように、接続した箇所に点滅する黄色い表示が発生し、さらに、左側の Annotation Browser に Shorts という表示が追加され、レイアウト中に配線の Short 箇所があることが示される。(これらの機能は普通に Path を描いても同じように表示される。)
5. では、間違った配線を Delete キーで削除し、再度配線を引き直す。まず間違った配線を選択し、Delete キーを押すと配線が削除され、Annotation Browser から Short の表示が消える事が分かる。その後、1 番からの動作を繰り返し、図 48 に示すように正しい箇所に配線を接続する。配線接続が完了すると、Annotation Browser 中で接続された配線名の後ろの括弧内の数字が一つ減る事が分かる。(本例では OUT 配線。)

以上で OUT 配線の接続が完了したが、Create Wire には配線の乗り換え時に便利な機能があるので、OUT 配線を削除してさらに解説を加える。

1. 先ほど配線した OUT 配線を選択し Delete キーで削除する。
2. Create > Wire コマンドを実行し、NMOS の左側端子をクリックする
3. 配線レイヤで METAL1 を選択し、OK をクリックする。
4. METAL1 配線を適当な位置まで引き、右クリックすると、図 49 に示すようなメニューが出てくるので、Via Up を選択する。
5. すると、自動的に VIA1 と METAL2 が表示されるので、好きな位置でクリックする。
6. すると、その後の配線は METAL2 で引くことができる。横に移動して再度右クリックし次は Via Down を選択すると、再度 VIA1 が表示され、好きな位置でクリックするとその後は METAL1 を配線することができる。
7. 以上の手順で、図 50 に示すように、一度 METAL2 に乗り換えて OUT 配線を接続する。

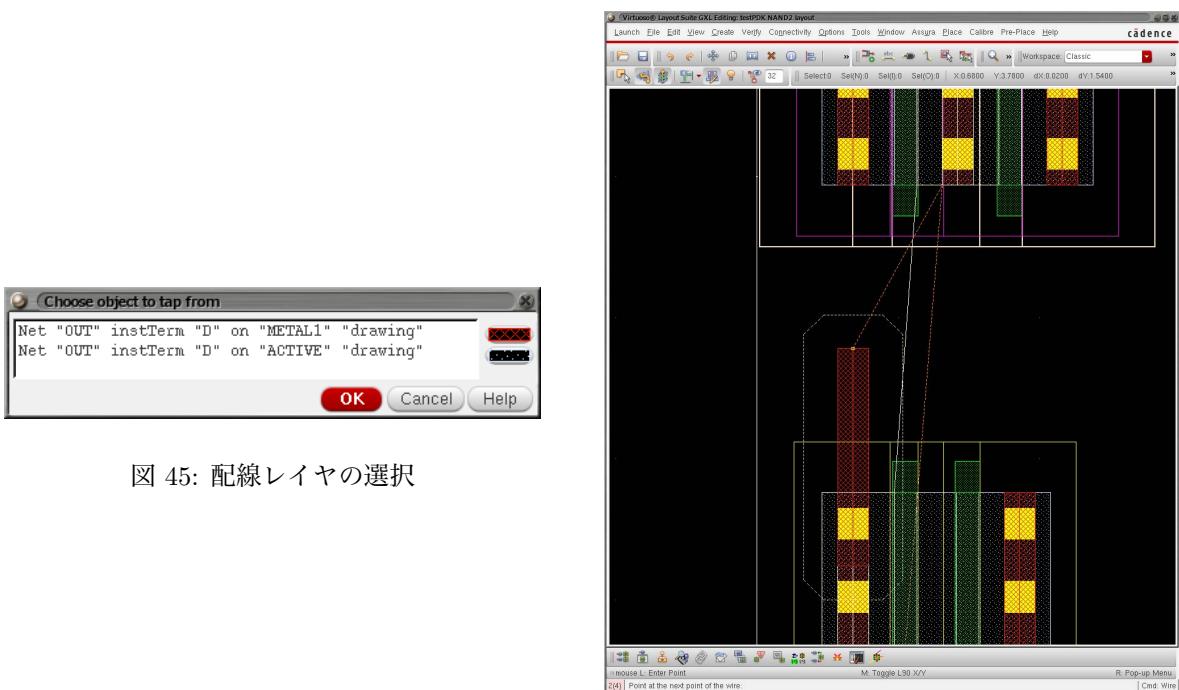


図 45: 配線レイヤの選択

図 46: Create Wire の実行

以上の手順を繰り返し、他の配線接続も行う。具体的には電源・グラウンドの接続と PMOS・NMOS トランジスタのゲートの接続を行う。ゲート端子も Create > Wire コマンドで同様の手順で配線できる（この際、X/Y Snap Spacing を 0.01 に設定しないとぴったり配線できないケースがあるので、オフグリッドエラーに注意しつつ適宜対処すること）。最終的に図 51 に全体像を示す形に配線を行う。

#### 注意:

Create > Wire で作成する配線では、デフォルトでは PathSeg というオブジェクトを生成する。この PathSeg というオブジェクトは IC61 から登場したもので、Stream Out した際に、形状によっては ROHM の提出ページで受け付けられないタイプの Path となることがある。（Path Type 4、終端が Variable になっている Path。）従って、Create > Wire を使用する際に PathSeg を使用することはおすすめできない。Create > Wire のいくつかの機能が使用できなくなるが、この PathSeg を通常の Path として Create > Wire を使用する方法があるので紹介しておく。実際に設計を行う場合には必須の設定と考えて頂きたい。

1. レイアウトエディタのメニューから Create > Wire と実行し（または Ctrl+Shift+m キー）、Wire コマンドを実行する。
2. F3 キーを押し Create Wire メニューを表示させる。
3. 一番上の Create Wire using から、Paths を選択する。

以上で Create > Wire の際のオブジェクトが Path になり、提出で失敗することがなくなる。しかしながら、上記の設定を Virtuoso 起動時に毎回やるのは面倒なので、以下の設定を各自のホームディレクトリの .cdsenv に記載しておく事をおすすめする。

```
layout createWireMode cyclic "paths"
```

以上を記述後に Virtuoso を再起動すると、次回以降は Create Wire オプションの Create Wire using が自動的に Paths に設定される。

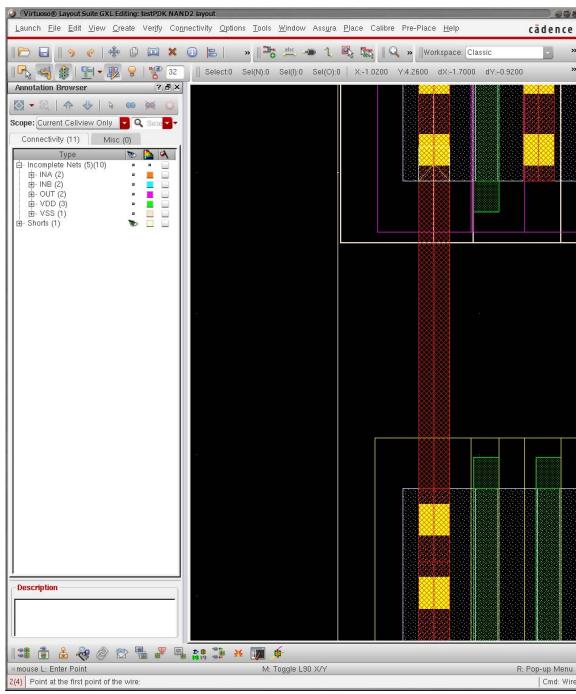


図 47: 配線のショート

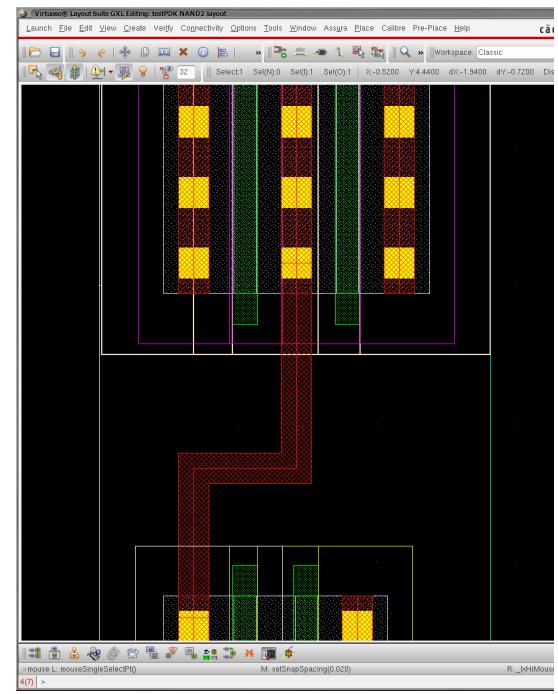


図 48: 配線接続の完了

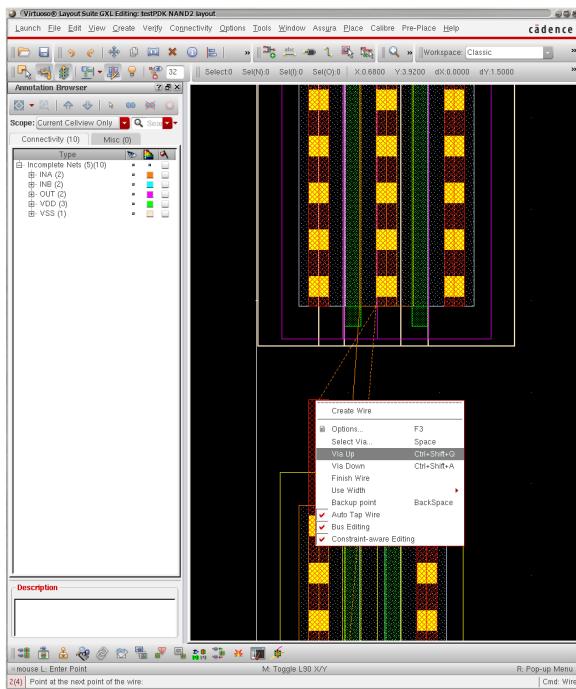


図 49: 配線レイヤの乗り換え

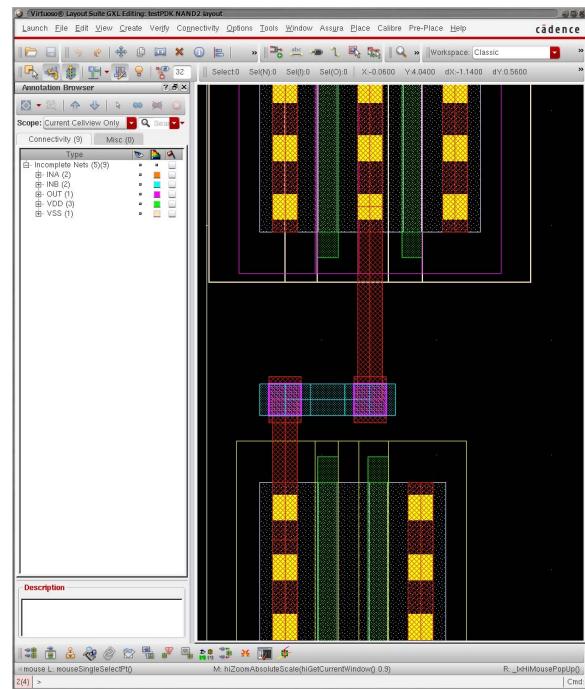


図 50: 配線レイヤの乗り換え その 2

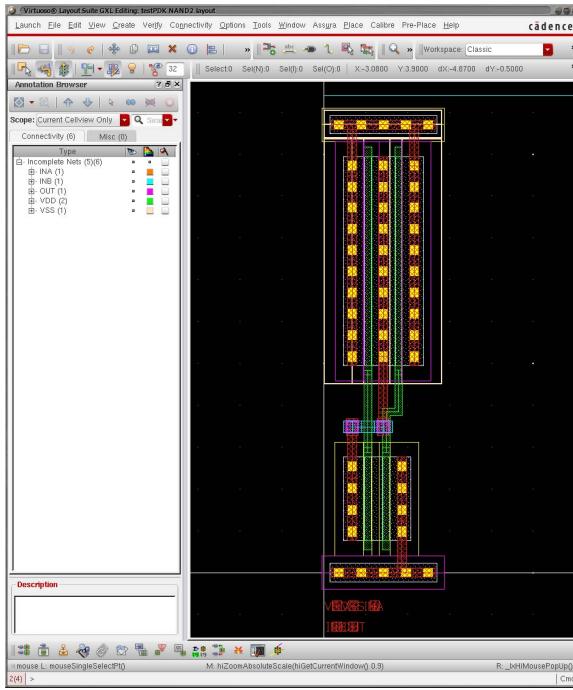


図 51: トランジスタの配線接続

## 11.7 端子の配置と Create Via コマンド

トランジスタの配線が完了したので、残りは端子を適当な位置に配置すれば 2 入力 NAND レイアウトの完成である。入力信号 INA・INB 以外の端子については既に METAL1 で配線が行われているため、Move コマンドで Pin の Rectangle を所定の配線の上に移動すれば接続が完了する。従って、VDD・VSS・OUT 端子については Move コマンド中に表示されるフライトライインに従って、結線を終了する。(この際、X/Y Snap Spacing を 0.01 にしないと、作成した配線にぴったり重ならない場合があるので、オフグリッドエラーに注意しつつ適宜対処すること。)

次に、INA・INB 入力用のゲートコンタクトを配置する。ここでも、従来までの手順では GATE-METAL1 間のコンタクトセルをあらかじめ作成しておきそれをインスタンスとして配置することが通例であったが、今回提供するテクノロジファイルを使用すると、Create > Via コマンドが使用できるのでこちらを使用する。(Create > Via コマンドは Layout GXL のみでなく Layout L でも使用可能である。)

1. レイアウトエディタのメニューから Create > Via... (または “o” キー) として、図 52 に示す Create Via ウィンドウを起動する。
2. 一番上の Mode を Single、Via Definition に M1\_GATE を選択する。
3. レイアウトエディタウィンドウに移動し、Design Rule 違反に注意しつつ (DRD を Notify に設定しているれば Design Rule 違反が表示されるはず)、それぞれのゲートに接するように、図 53 に示すように配置する。
4. ゲートコンタクトが配置できたら、ゲート入力端子を Move コマンドで移動し、図 54 に示すように対応するゲートコンタクトのメタル上に配置する。以上で端子の接続が全て終了する。Annotation Browser 上で Incomplete Nets が VDD の 1 つのみになっていることを確認する。残りの一つの VDD 端子は、PMOS の基板側の接続であり、実際は接続されているが接続関係が抽出できないもの (N-Well 経由でつながっている) である。そのため VDD の未接続が一つ残ってしまってよい。
5. トランジスタおよび端子の接続が完了したが、ゲート入力端子のメタルが最小許容面積よりも小さくなつ

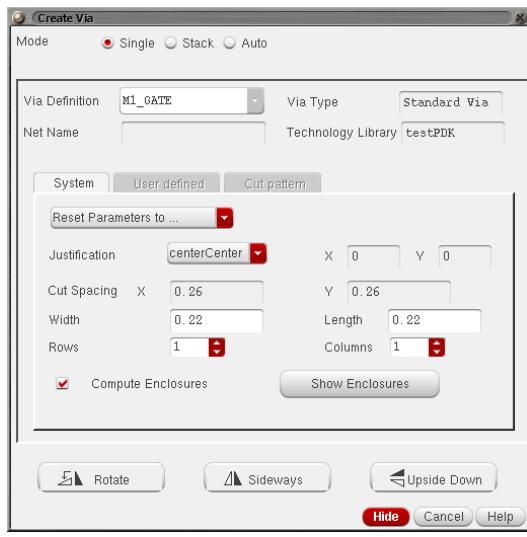


図 52: Create Via メニュー

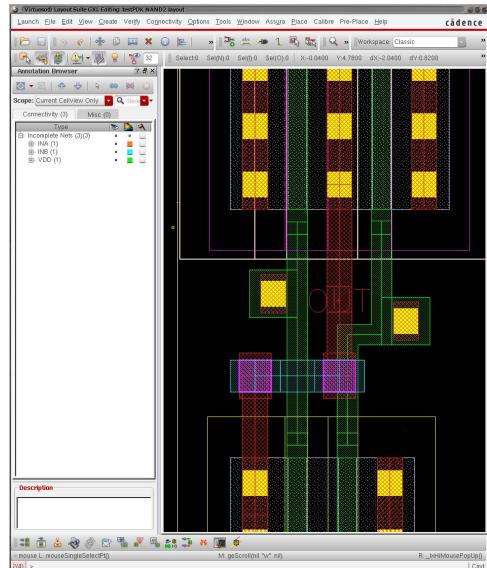


図 53: ゲート入力用コンタクトの配置

ているため、Design Rule 違反が残っている。Rectangle でメタルを描き、表示される Design Rule に従って、図 55 に示すようにメタル領域を追加する。

以上で NAND2 のレイアウトが完成した。最後に、水色の P&R Boundary が広く残ってしまっているので、このサイズを調整する。

1. LSW ウィンドウの上方の Object ボタンをクリックする
2. LSW ウィンドウの表示が変わったら中段あたりにある Boundaries: のところから P&R のところの Sel 側(右側)のチェックを入れる。
3. これでレイアウトエディタ上の水色のレイヤが選択できるようになるので、Stretch コマンド等を使用して NAND2 レイアウトのサイズに合わせる。(Delete キーで消してもよい。)

以上で NAND2 レイアウトの完成。

#### Tips:

Create Via コマンドをゲートコンタクトのみに使用したが、この機能は他にも便利に使用することができる。以下に簡単に使用方法の例を示す。

- 任意の場所に Via を打ちたい場合は、Create Via を使用できる。レイアウトエディタ上で “o” として使用したい Via を選び、アレイにしたい場合には Row/Column の値を適宜選ぶ。その後好きな位置でクリックすると Via を配置できる。
- Create Via メニューの一番上の Mode で Stack を選択すると、複数レイヤ分を Stack して打つことができる。
- Mode で Auto を選択すると、メタルの重なり部分を埋め尽くすように自動的に Via を打つことができる。図 56 (a) に示すような METAL3 と METAL4 が重なった部分をクリックすると、図 56 (b) に示すように、自動的に個数が計算され、重なり部分を埋め尽くすように所定の Via が生成される。
- METAL5 から CTM への Via は特殊なため Create Via メニューでサポートされていないので、vdecRO180PDK ライブラリに含まれている CTMVia をインスタンスとして呼び出して使用すること。

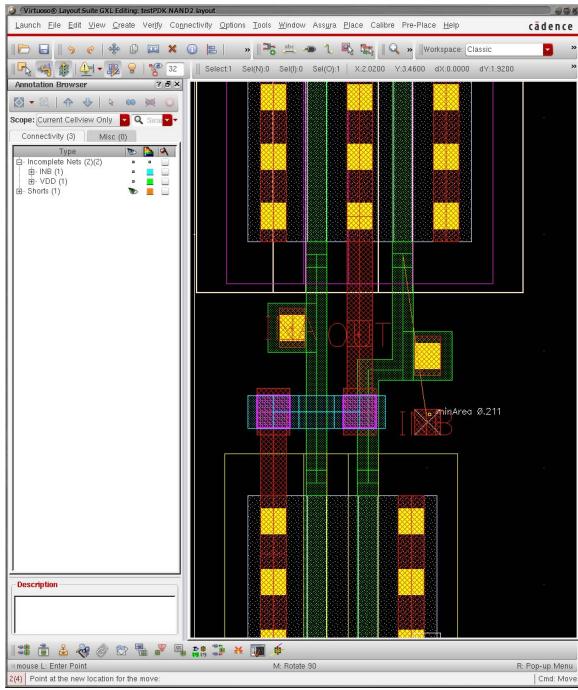


図 54: ゲート入力端子の配置

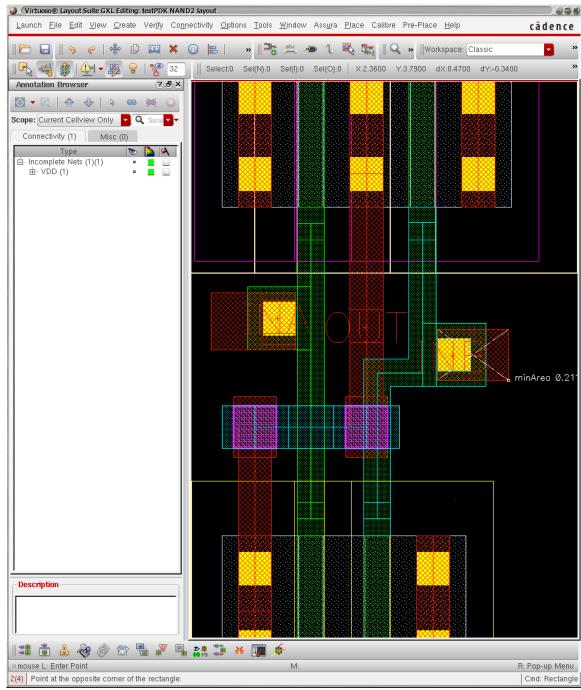


図 55: ゲート入力端子の minArea 対策

## 12 回路パラメタ変更に対するレイアウト修正

以上で、ひとまず IC61 環境で PDK を使用して回路図設計からレイアウト設計まで行うことができた。通常、回路の仕様や機能が最初からきちんと決まっている場合には設計した回路図のパラメタが変更される事は少ない。しかしながら、得てして設計途中で仕様や目的の性能が変わったり、レイアウトしてみたら寄生成分の効果で性能が出ていなかったりと言うことがあって、実際のところは設計変更をすることが非常に多い。

この節では、ここまでで作成した 2 入力 NAND 回路に対して回路図の変更を行ったとして、回路図の変更をレイアウトに反映させる手順について説明を行う。

### 12.1 準備

作成した 2 入力 NAND 回路は、さらに次の節の階層設計で用いるので直接編集してしまわないこと。以下の手順で NAND2 セルを NAND2\_MOD セルとしてコピーする。

1. Library Manager の中央の Cell の欄で NAND2 を右クリックする。
2. 一番上の Copy... を選択し、Copy Cell ウィンドウを開く。
3. 上段の To の欄の Cell を NAND2\_MOD とし、中段の Options の欄の Copy All Views にチェックが入っていることを確認して OK をクリックする。
4. Library Manager で Cell に NAND2\_MOD が追加されていることを確認する。

### 12.2 回路図の変更

コピーした NAND2\_MOD セルの schematic に変更を加える。以下では

- NMOS トランジスタのサイズを変更する。
- 出力に RC フィルタをつける（実際にはそんな回路は無いと思いますが…）。

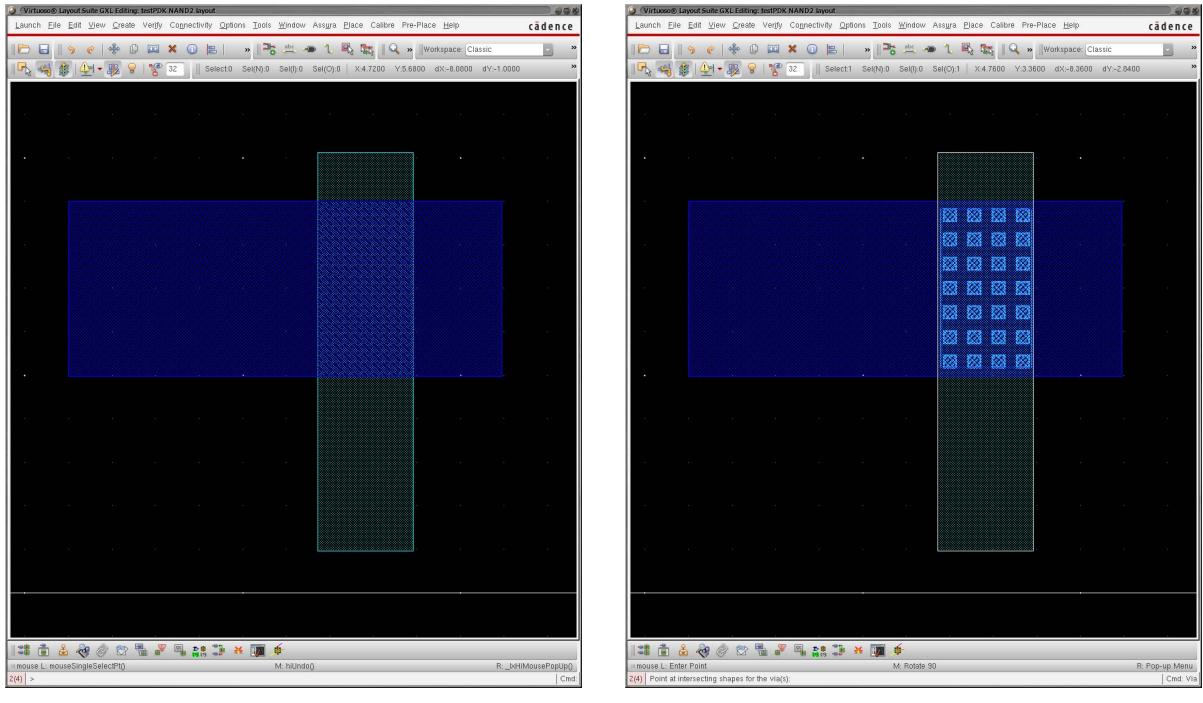


図 56: Auto Via の実行

の 2 点の変更を加える。

1. Library Manager から、NAND2\_MOD を選択し、View 欄の schematic をダブルクリックして開く。
2. NMOS トランジスタを一つ選択し、“q”キーで Properties を表示する。
3. Width を  $2\mu\text{m}$  から  $2.4\mu\text{m}$  に変更して OK をクリックする。
4. もう一方の NMOS トランジスタについても同じ変更を加える。
5. 続いて、NAND の出力に RC フィルタを挿入する。出力端子 OUT のピンを選択し、“M”キー(大文字の M)を押して、抵抗を挿入する分だけ右に移動する。
6. “i”キーとして、Add Instance ウィンドウを開き、Browse ボタンを押して vdecRO180PDK ライブラリの lpph の symbol を選択する。
7. Add Instance ウィンドウに表示されるパラメタのうち、Width や Input Res. Value を適当にいじってみる。入力される値に応じて、Length の値が自動計算されることが分かる。オフグリッドエラーを避けるため、Width の値は  $0.04\mu\text{m}$  ステップでの入力に制限されている。抵抗の PCell では、オフグリッドにならない範囲で、入力された抵抗値にもっとも近い値を計算してレイアウト形状を決定している。ここでは、Width に  $2\mu\text{m}$ 、Input Res. Value に  $2\text{K Ohms}$  の値を使用することとする。この時、Length の値は  $3.36\mu\text{m}$  と計算されるはずである。このとき、実際に Simulation で使用され、レイアウト上に実現される抵抗値は Actual Res. Value に示されている  $2.005032\text{K Ohms}$  となる。この PCell に準じてレイアウトを行えば抵抗値の Property Error は基本的には発生しないはずである。
8. 選択した lpph 抵抗を NAND の出力に直列に接続する。
9. 続いて、同様に Add Instance ウィンドウから、mimcap の symbol を選択する。抵抗と同様に、こちらも Width と Input Cap. Value が入力でき、Length が自動計算される。こちらもオフグリッドエラーを避けるために、Width 入力は  $0.04\mu\text{m}$  ステップに制限されている。適当に値を入力して挙動を確認する

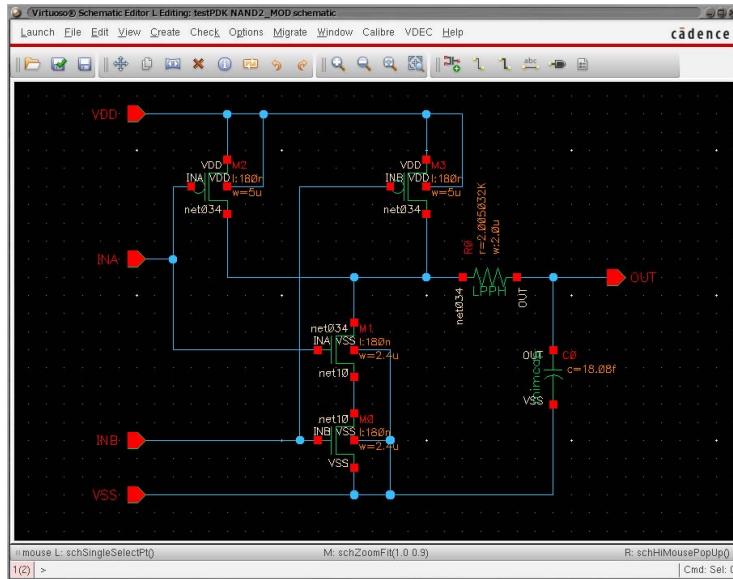


図 57: 変更を加えた 2 入力 NAND 回路図

こと。mimcap の値は Width を  $4\mu\text{m}$ 、Input Cap. Value を  $18\text{fF}$  とする。Length、Actual Cap. Value の値が自動計算され、それぞれ  $4.52\mu\text{m}$ 、 $18.08\text{fF}$  となることが分かる。こちらも抵抗と同様に Actual Cap. Value が Simulation で使用され、実際に実現される値となる。

10. mimcap の symbol を OUT 出力から VSS に接続し、図 57 に示すような回路図を作成する。

以上の作業で回路図に対する修正が終了した。本来ならここでシミュレーションによる動作確認を修正後の回路に対して行うが、本演習ではこの作業は省略する。興味があれば各自で、ネットリスト出力、HSIM によるシミュレーション実行、出力波形確認を行うこと。

### 12.3 レイアウトの変更

続いて、修正した回路のレイアウトの修正作業を行う。本作業でも Layout GXL を使用する。

1. 前節で修正を行った NAND2.MOD 回路の回路図エディタ上での左上のメニューから Launch > Layout GXL と実行する。
2. Startup Option ウィンドウが立ち上がる所以、Layout に Open Existing、Configuration に Automatic を選択し、OK をクリックする。
3. さらに立ち上がる Open File ウィンドウで、Library に testPDK、Cell に NAND2.MOD、View に layout が選択されていることを確認し、OK をクリックすると、レイアウトエディタが起動する。
4. 次に、新たに回路図に追加した要素をレイアウト上に読み込む。レイアウトエディタ下部のボタンの左から 8 番目の、びっくりマークがついた Update Components and Nets ボタンをクリックする。NAND2.MOD 回路図から情報を抽出してもいいですか?といったようなことを聞かれるので、OK をクリックする。
5. Update Components and Nets ウィンドウが立ち上がる所以、そのまま OK をクリックする。
6. 図 58 に示すように、回路図に追加した抵抗と容量のレイアウトインスタンスが自動的に配置される。
7. さらに、パラメタを変更したトランジスタの情報を更新する。情報を更新した NMOS トランジスタをレイアウトエディタ上で選択する。まず左側に配置した NMOS トランジスタを選択する。

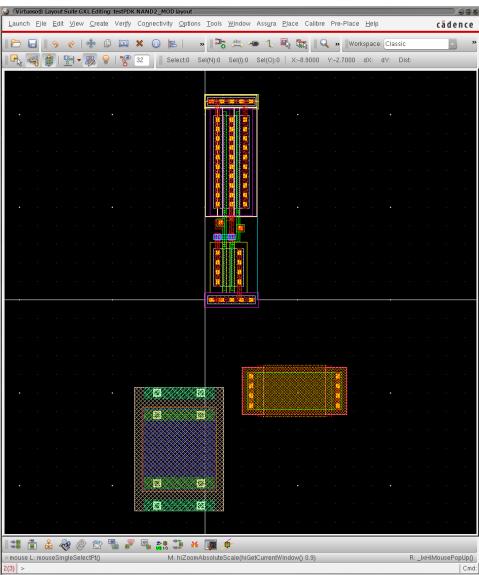
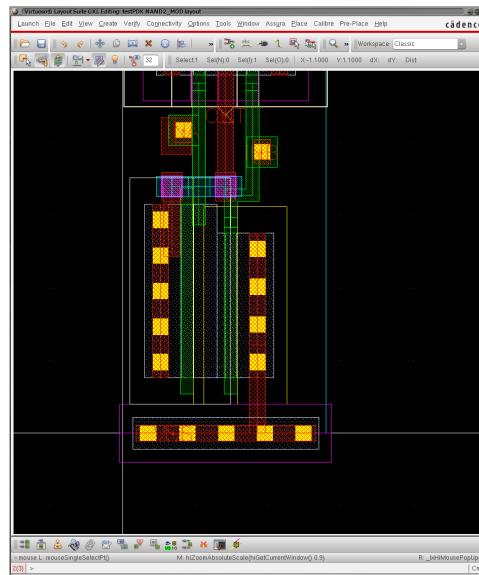


図 58: 2 入力 NAND 回路図に抵抗と容量のレイアウト  
図 59: 片方の NMOS トランジスタのパラメタが更新  
トが追加された



8. NMOS トランジスタを選択した状態で、レイアウトエディタ下部のボタンの左から 9 番目の、Update Layout Parameters ボタンをクリックすると、選択された NMOS トランジスタの変更した Width の値が更新される。
9. 続いてもう片方の NMOS トランジスタを選択し、同じボタンをクリックし、情報を更新する。(最初から二つまとめて選択して同じボタンを押すと、二つ同時に更新できる。つまり、更新したトランジスタのみを選択しなくても、Ctrl+A キー等で、レイアウトエディタ上の全てを選択し、Update ボタンを押すことで、漏れなくパラメタが変更された PCell を更新することができる。)
10. Width の変更に伴って、位置がずれる場合があるので適宜修正し、Design Rule 違反がなくなるように、ゲート等他のパターンの形状も修正すること。
11. 以上の操作で、回路図の情報がレイアウト上に更新されたので、後は残りの接続を行うのみである。これまでに説明した方法から好きな方法を使用して、抵抗、容量の接続を完了すること。
12. 注意点としては、抵抗・容量に含まれるレイヤには最小グリッドが  $0.04\mu\text{m}$  のものがあるため、抵抗・容量のインスタンスを移動させる際には X/Y Snap Spacing を 0.04 にしてから行った方がよい。(初期配置からオフグリッドになっていることもあるので注意。) また、mimcap は METAL5 で端子が出ているので、配線接続の際に Via Up/Down 等をうまく使うこと。
13. 接続作業が終了したら、Calibre Interactive を使用して DRC/LVS のチェックを行うこと。
14. 最終的に、図 60 に示すような形に接続が完了すればよい。(実際に設計・チップ試作を行う場合にはもう少しきれいにレイアウトしてください。)

## 13 階層設計

続いて、作成した元々の NAND2 回路およびレイアウトを使用して、2 入力 NAND を使用した、5 段のリングオシレータ回路を設計する。NAND2.MOD の Schematic、Layout が開いたままの場合にはそれらは閉じてしまってよい。

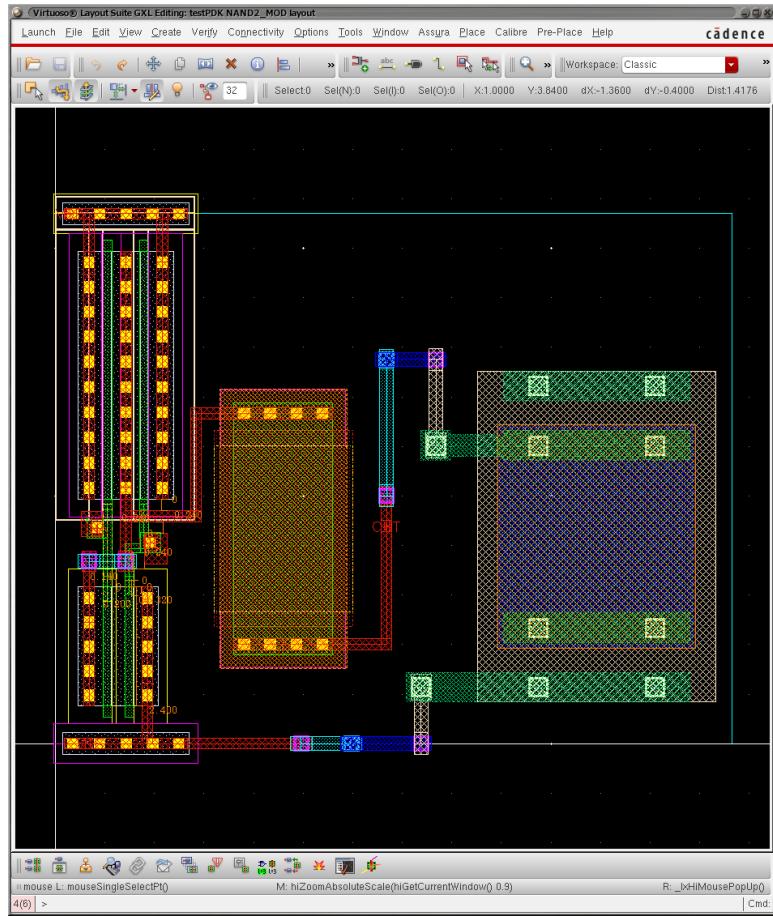


図 60: 変更後の 2 入力 NAND 回路レイアウト

### 13.1 2 入力 NAND 回路シンボルの作成

まず、作成した 2 入力 NAND 回路を階層設計で使用するための回路シンボルを以下の手順で作成する。

1. Library Manager から、testPDK ライブラリの NAND2、schematic を開く。
2. 開いた回路図エディタの上のメニューの左から 5 番目、Create > Cellview > From Cellview と実行する。
3. Cellview From Cellview というウインドウが立ち上がるので、そのまま OK とする。
4. 次に Symbol Generation Option というウインドウが立ち上がるので、Pin Specification の欄で、端子を好きな位置に指定する。デフォルトでは、Input に指定した端子は Left Pins に、Output に指定した端子は Right Pins に指定されている。自分の好きな位置に指定すること。図 61 に示すように、VDD を Top Pins に、VSS を Bottom Pins に指定するのがおすすめである。
5. OK をクリックすると、Symbol Editor が起動する。最初は指定した位置に端子がついた四角のブロックが表示されるので、そのままの形で使用してもよいが、NAND 回路であるので、図 62 に示すような一般的な NAND の回路図に編集するとよい。編集には、Delete キーで最初に表示されている緑の四角の枠を消し、Symbol Editor 上部のメニューから Create > Shape で表示される様々な形状を使用する。端子に足をつけるなど、自分の好み、各研究室ごとの流儀に合わせて作成してよい。
6. 図 62 に示すような回路図がかけたら、左上のフロッピーディスクマークをクリックして保存し、Symbol Editor を閉じる。NAND2 の Schematic Editor も閉じてしまってよい。



図 61: シンボル作成時のピン位置の指定

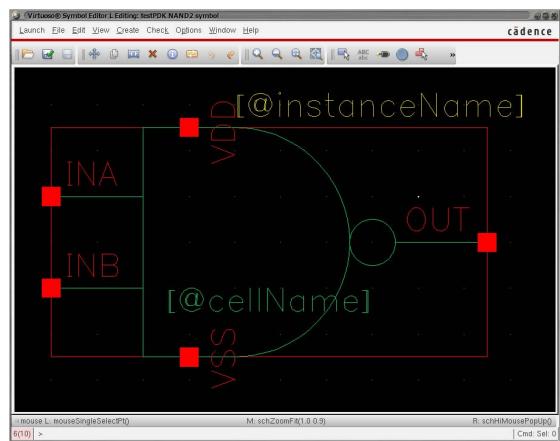


図 62: NAND2 回路のシンボル作成

#### Tips:

Symbol Editor 上でデフォルトで作成される Symbol には、緑色の @partName と言う文字が入っている。これは、作成した Symbol を Schematic Editor 上で呼び出した際に自動で回路名に置き換わる部分である。もちろんこのままにしておいてよいが、後々、例えば NAND のシンボルを別のセルにコピーして使用したい、といった場合がある。そういうときに、Symbol を別のセル名にコピーしても、@partName の箇所はコピー前のセル名のままになってしまう。後々使いまわすを考えると、これは不便である。この対策として、@partName を @cellName に変更しておく。こうするとコピー後はコピー後のセル名が表示されるようになる。

## 13.2 リングオシレータ回路図の設計

次に、作成した 2 入力 NAND 回路シンボルを使用して、本書でこれまでに説明した手順を参考にしながら 5 段のリングオシレータ回路を設計する。

1. Library Manager から、testPDK ライブラリを選択した状態で File > New > Cell View として、NAND2\_OSC というセルを作る。
2. 起動する回路図エディタ上で、“i”キーとして、Add Instance ウィンドウを開き、Browse ボタンを押して testPDK ライブラリの NAND2 の symbol を選択し、回路図上に配置する。同操作を繰り返し、各 NAND 回路を接続して、図 63 に示すような回路図を作成する。
3. 第 3.3 節の解説を参考に、VDEC メニューを使用したネットリスト出力、HSPICE/HSIM によるシミュレーション実行、wv による波形確認を行う。SPICE 入力ファイルは参考例を置いたのでシミュレーションを実行するディレクトリで以下のコマンドを実行し、コピーすること。

```
% cp /usr1/iizuka/cadence/ROHM018/forNewComer/nand2_osc.sp .
```

4. 図 64 に示すような発振波形が確認できればよい。

## 13.3 レイアウト設計

続いて、設計したリングオシレータ回路のレイアウト設計を行う。第 7 節および第 11 節で説明した手順を参考に行う。

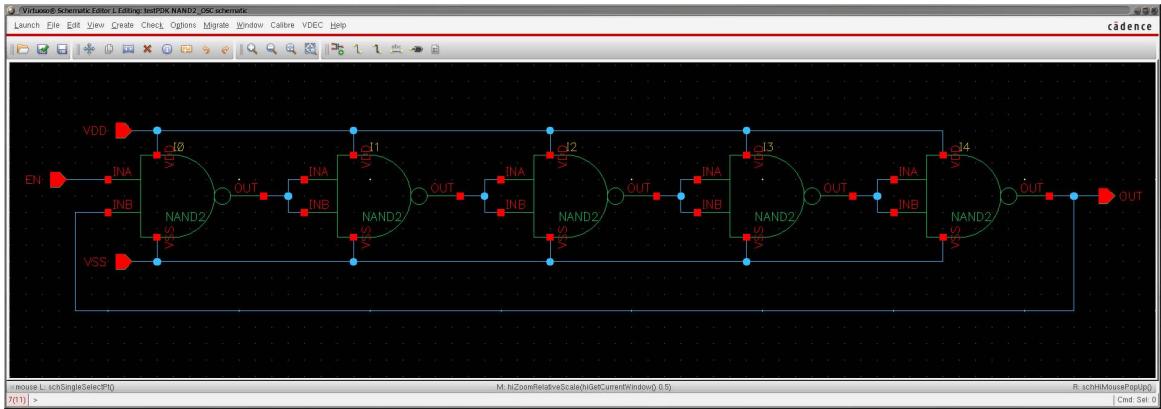


図 63: 2 入力 NAND 回路を使用した 5 段リングオシレータ

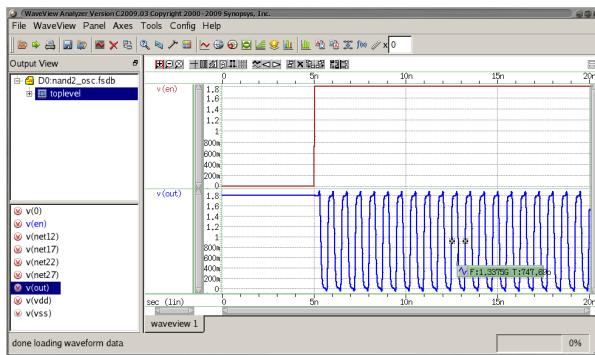


図 64: 2 入力 NAND 回路を使用した 5 段リングオシレータのシミュレーション波形

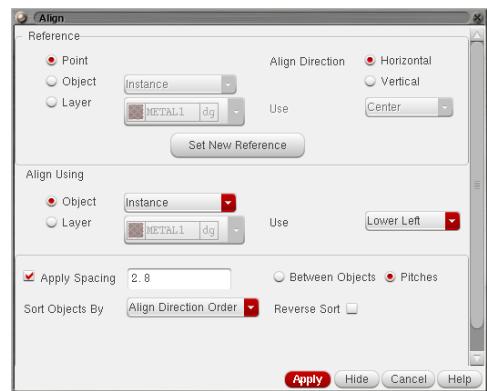


図 65: 等ピッチでの Align 設定

1. 回路図エディタの左上のメニューから Launch > Layout GXL とし、Create New、Automatic を選択して OK とする。さらに立ち上がるウィンドウでも OK とする。
2. 新たに起動したレイアウトエディタの下部の一番左側の Generate All From Source ボタンを押し、I/O Pins を全て METAL1 にし、Pin Label Shape を Label、Pin Label Options で Height を 0.2、Layer Name、Layer Purpose をどちらも Same As Pin を選択し OK。さらに Generate Layout ウィンドウも OK とする。
3. レイアウトエディタ上に NAND2 レイアウトが 5 個自動的に読み込まれる。後はこれらを必要に応じて並べ、結線すればよい。
4. 5 個の NAND2 レイアウトを規則正しく並べる際に、第 11.4 節で紹介した Align コマンドを使用するといい。この際に Pitch を指定して並べることもできる。図 65 に示すように、Reference の欄で Point を指定し、Align Direction に Horizontal を指定する。Reference 欄の Set New Reference ボタンをクリックした後、レイアウトエディタ上で原点 (0,0) をクリックする。図 66 に示すように、原点から水平にピンク色の線が表示される事を確認する。その後、Align ウィンドウに戻り、Align Using の欄では Object を指定し、Instance を選択する。Use には Lower Left を選択する。Apply Spacing に必要な数値を入力する。本例では NAND2 のレイアウト幅の 2.8 を入力する。各自の設計したレイアウトの幅に合わせて指定すること。さらにその右側で Pitches を選択しておく。レイアウトエディタウィンドウで左ドラッグによる領域選択で 5 個の NAND2 レイアウトを全て選択する。その後、Align ウィンドウで Apply をクリックすると図 67 に示すように、原点を起点に 5 個の NAND2 レイアウトが等ピッチで並ぶことが分かる。

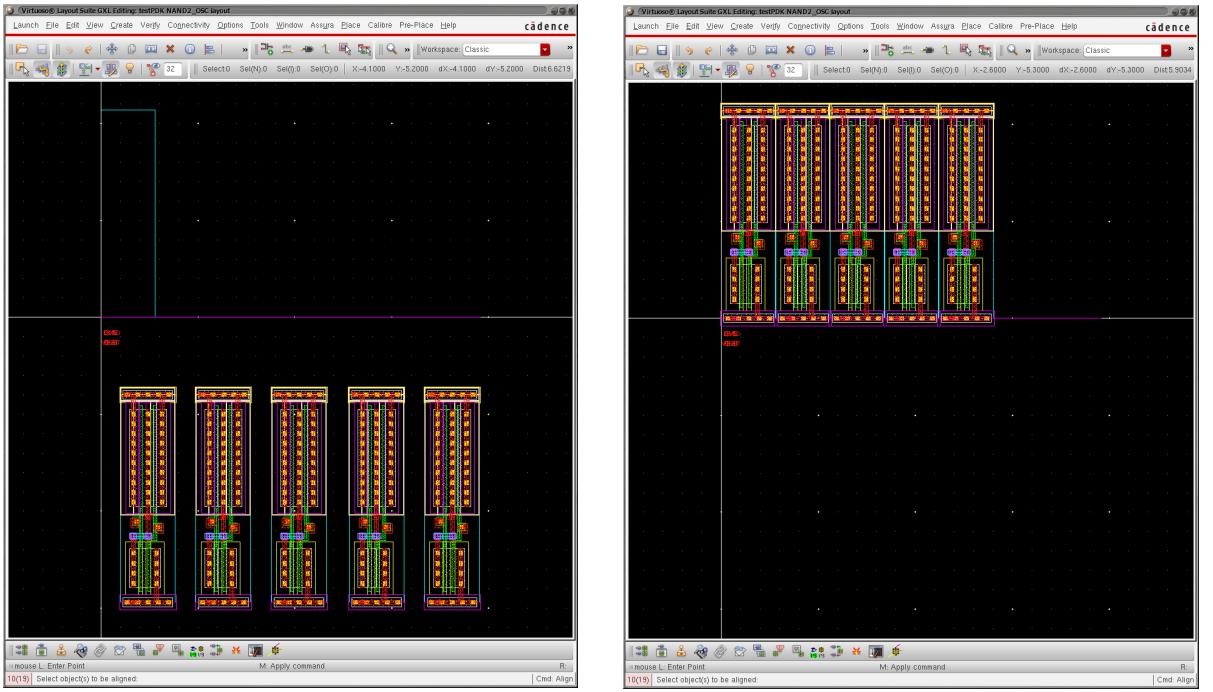


図 66: Align コマンドによる Reference Point の指定 図 67: 等ピッチで NAND2 レイアウトが並べられた

5. 配線接続を始める前に、接続関係の情報を抽出する階層の設定を行う。レイアウトエディタのメニューから Options > Layout XL... と実行し、Extraction タブの上から 2 番目の Extract connectivity to level: の欄の数字を増やす。今回は 4 度にしておけば問題ないが、もし面倒であれば 32 等の大きい数字をいれておいてもよい。OK をクリックして Options ウィンドウを閉じる。これで配線の接続関係を階層を降りてチェックしてくれるようになった。接続関係の情報はレイアウトが編集されたときにしか更新されないため、この設定が 0 の状態で接続された配線はいつまでも未接続のまま残ってしまう。配線を始める前に事前に設定すること。
6. 後は Create Wire や Create Via を駆使して必要な端子を接続すればよい。p キーや r キーを使って適宜配線しても問題無い。設計中に表示される Connectivity や Design Rule の情報をうまく利用して、間違なく結線すること。Annotation Browser を起動して未接続の配線やショートをチェックしながら行うとよい。図 68 に最終的なレイアウトの例を示す。
7. 結線が終了したら、第 8 節で説明した手順に従い、Calibre を使用して DRC/LVS を実行する。両者ともにエラーが無くなれば終了。

#### Tips:

メニューから Options > Layout XL... として接続関係を抽出する階層レベルを毎回設定するのが面倒な場合は各自の Home Directory に .cdsenv というファイルを作成し、以下を記述する(既にある場合はそこに追記する)。

```
layout extractStopLevel int 32
```

以上を記述後に Virtuoso を再起動すると、次回以降は自動的に接続関係が 32 階層下までチェックされる。

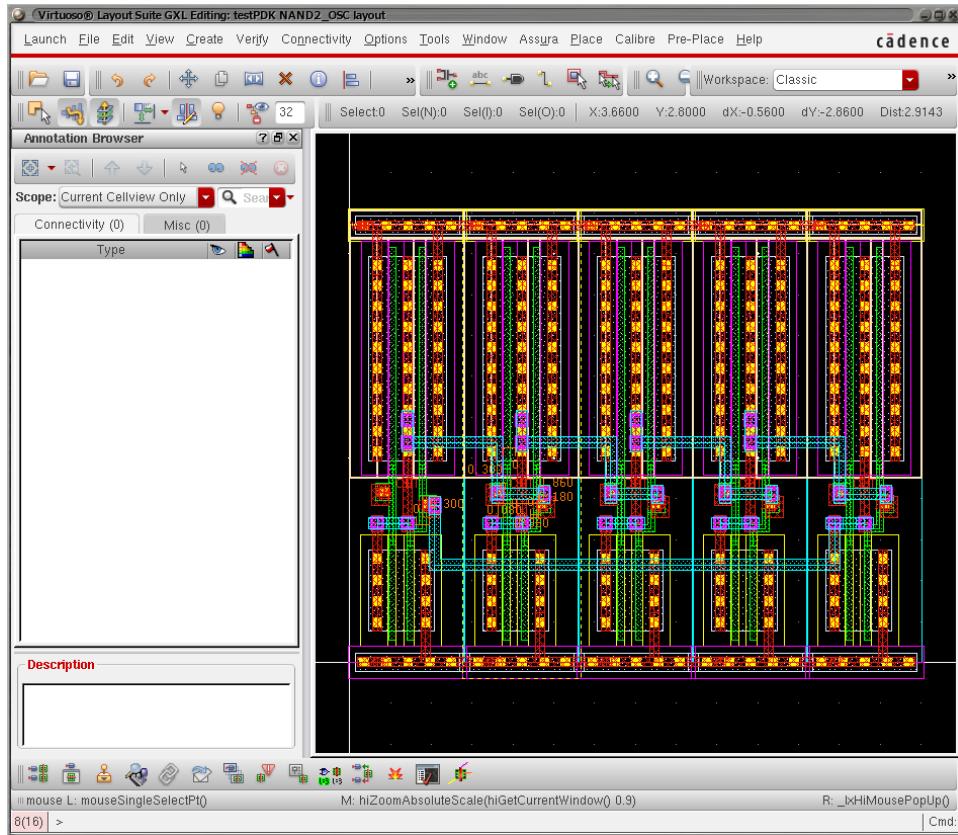


図 68: 2 入力 NAND 回路を使用した 5 段リングオシレータのレイアウト

## 14 Virtuoso Layout GXL を使用した自動配置配線

Virtuoso Layout GXL を使用してトランジスタや自分で設計した NAND 回路等を自動的に配置し配線を行うことができる<sup>5</sup>。以下では本書で設計した 2 入力 NAND 回路と、5 段のリングオシレータ回路 NAND2\_OSC 回路を再度使用して、自動配置配線の手順を体験する。

### 14.1 準備

まず、自動配置配線を行う回路を作成する。NAND2\_OSC の Schematic を NAND2\_OSC\_AUTO としてコピーし、そちらの Cellview を使用して以下の演習を行う。

1. Library Manager から、testPDK ライブラリの NAND2\_OSC を選択し、View の schematic を右クリックする。
2. 上から 4 番目の Copy... を選択し、開いた Copy View ウィンドウを開く。
3. 上段の To の欄の Cell を NAND2\_OSC\_AUTO とし、OK をクリックする。
4. Library Manager で Cell に NAND2\_OSC\_AUTO が追加されていることを確認する。

### 14.2 Layout View の作成と回路要素の読み込み

次に、コピーした NAND2\_OSC\_AUTO の schematic から layout view を作成し、回路要素をレイアウトエディタ上に読み込む作業を行う。この作業は、第 13.3 節で説明を行った 3 番までの作業と同じである。

<sup>5</sup>ただし、得られる結果は決して最適といえるものではなく、あまり有効に活用できる段階ではないのでほとんど使用機会は無いと考えてもらって良い。が、参考のため使用手順について記す。

1. Library Manager から、testPDK ライブライアリの NAND2\_OSC\_AUTO を選択し、View の schematic をダブルクリックして開く。開いた回路図エディタの左上のメニューから Launch > Layout GXL とし、Create New、Automatic を選択して OK とする。さらに立ち上がるウィンドウでも OK とする。
2. 新たに起動したレイアウトエディタの下部の一番左側の Generate All From Source ボタンを押し、I/O Pins を全て METAL1 にし、Pin Label Shape を Label、Pin Label Options で Height を 0.2、Layer Name、Layer Purpose をどちらも Same As Pin を選択し OK。さらに Generate Layout ウィンドウも OK とする。
3. レイアウトエディタ上に NAND2 レイアウトが 5 個自動的に読み込まれる。以降の節でこれらを自動的に配置し、配線を行う。

### 14.3 自動配置の実行

次に前節で読み込んだレイアウト上の回路要素を自動的に配置する。この際、本書で扱う ROHM 0.18μm プロセスでは、最小許容グリッドの異なる複数のレイヤが存在するため、最終的にオフグリッドエラーの発生しないレイアウトを生成するために最小グリッドを適宜変更する必要がある。今回の PDK 開発とともに最小グリッド変更のための GUI ユーティリティも独自に開発したので、それを使用して自動配置を行う。

1. まず初めに、Layout GXL ウィンドウ右上の Pre-Place メニューをクリックし、Change MfgGridResolution ... を選択する。このメニューは今回独自に追加されたもので、公開された PDK を設定していると自動で読み込まれる。
2. 図 69 に示すウィンドウが起動する。ここで、Library に今回設計を行っているライブラリ testPDK が選択されていることを確認し、MfgGrid の Set Value 欄で 0.02 を選択する。今回演習で使用するレイアウトでは最小グリッドが 0.02 のレイヤしか含まないためこの設定でよいが、例えば抵抗素子や MiM Cap. 等を使用している場合には最小グリッドが 0.04 のレイヤを含むため MfgGrid の欄で 0.04 を選択して以下の作業を行う必要がある。
3. Set Value を 0.02 として、Current の値が 0.02 に変更されたことを確認する。これで自動配置の最小グリッドが 0.02 に変更された。Change MfgGridResolution ウィンドウで変更したグリッド設定は、同ウィンドウを終了すると自動でデフォルトの設定に戻る。そのため、以後の自動配置・配線作業が終了するまでこのウィンドウは立ち上げたままにしておくこと。
4. レイアウトウィンドウの上部のメニューの右端、Help ボタンと cadence ロゴの間の何もないところでマウスの右クリックをし、出てくるメニューから Analog Auto Placer を選択する。
5. すると、分かりづらいが、図 70 に示す位置に新しいアイコンが表示される。アイコン横の >> マークをクリックして同図に示すメニューを表示し、その中から Analog Automatic Placement を選択する。
6. Analog Auto Place ウィンドウが新規に立ち上がるるので、Placement Effort Level に適宜値を選択し（ここでは Nominal でよい）、Update PR Boundary にチェックが入っている事を確認して OK をクリックする。
7. 以上で自動的に配置が実行され、図 71 に示すような結果が得られる。

### 14.4 自動配線の実行

自動で配置されたレイアウトに対して端子の接続を自動で行う。

1. レイアウトウィンドウの上部のメニューの右端、Help ボタンと cadence ロゴの間の何もないところでマウスの右クリックをし、出てくるメニューから Space-Based Router を選択する。

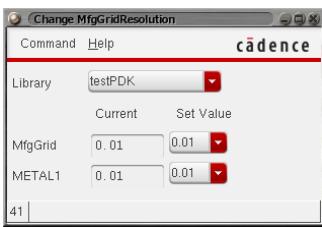


図 69: Change MfgGridResolution ウィンドウ

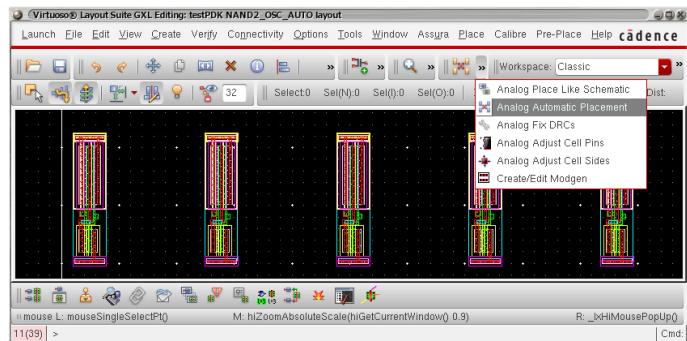


図 70: Analog Automatic Placement

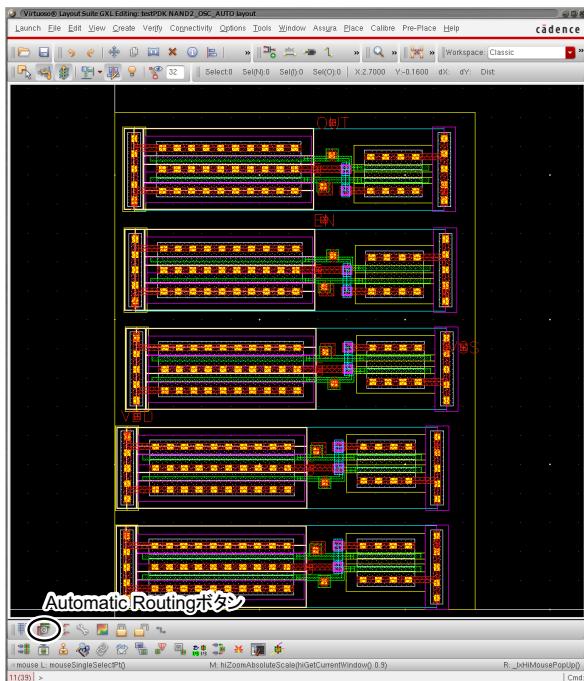


図 71: 5 個の NAND2 レイアウトが自動で配置された



図 72: Automatic Routing ウィンドウ

- すると、Layout GXL ウィンドウの下部にアイコンリストが追加される。図 71 中に示す位置の、追加されたアイコンの左から 2 番目の Automatic Routing ボタンをクリックし、図 72 に示す Automatic Routing ウィンドウを起動する。
- General タブの Operate On で Entire Cell View が選択されていることを確認する。
- 次に、同じ General タブ内の Routing Options で Use Grid に Routing を選択し、自動配線に使用するメタルレイヤを右の Top Layer および Bottom Layer から選択する。本演習では Bottom Layer に METAL1、Top Layer に METAL2 を選択すれば十分配線を完了できる。
- Automatic Routing ウィンドウで OK をクリックする。少し時間がかかった後、図 73 に示すように自動で配線が完了する。完成したレイアウトに対して第 8 節に示した手順で DRC/LVS を実行し、正しいレイアウトになっているか確認してみること。
- 4 番の手順で、Use Grid に Routing を選択したが、ここで Manufacturing を選択するとより細かい解像度で配線を実行することができ、Routing グリッドで配線を完了できない場合にも解を見つけることができる可能性がある。しかしながら通常のグリッド設定で Manufacturing を選択すると METAL1 か

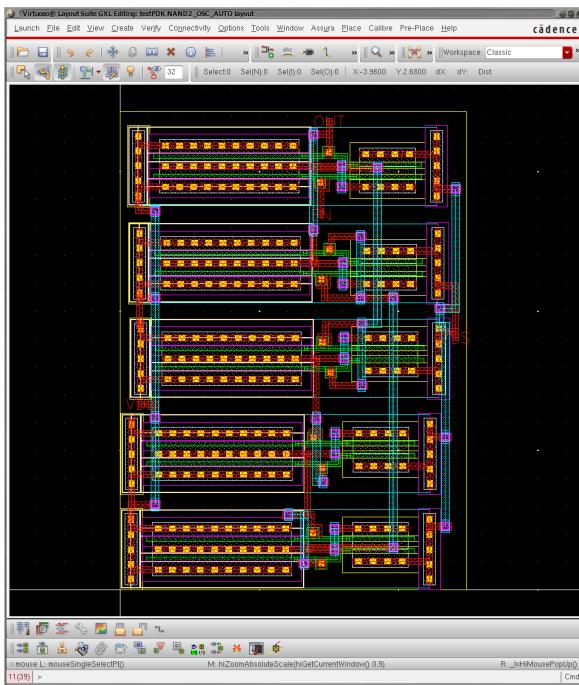


図 73: 自動配置・配線された 5 段 NAND2 リングオーバー シレータレイアウト

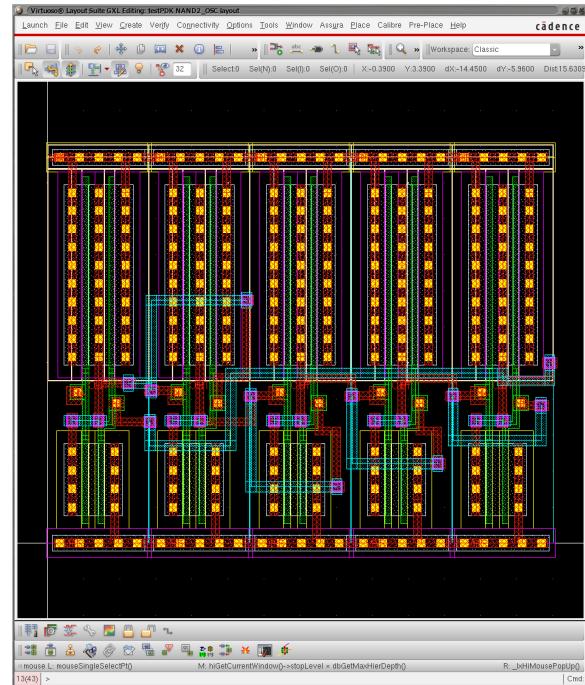


図 74: 手でインスタンスと端子を配置して自動配線を行ったリングオーバーシレータレイアウト

ら METAL2 への乗り換え箇所でオフグリッドエラーが発生する場合がある。これを避けるためには、自動配置の節で起動しておいた Change MfgGridResolution ウィンドウで METAL1 の欄の Set Value で 0.02 を選択した後に自動配線を実行すればよい。これにより METAL1 の最小グリッドが 0.02 となり、METAL1 から METAL2 への乗り換え箇所のオフグリッドエラーを抑止することができる。

7. 自動配置・配線が終了したら Change MfgGridResolution ウィンドウを×印をクリックして終了する。同ウィンドウを終了すると変更したグリッド設定は全てデフォルトの 0.01 に戻る。(CIW ウィンドウに \*INFORMATION\* としてデフォルト値に戻した旨のコメントが表示される。)

以上で自動配置・配線によるレイアウト生成フローの解説を行ったが、得られたレイアウトは一般的によく見られるような結果ではなく、少々受け入れがたい。現状推奨できる使い方は、まず配置を自分で手で確定し、配線のみ自動でやらせることである。インスタンス・端子を手で配置した後で自動配線を行った場合の結果の例を図 74 に示す。比較的受け入れられる結果が得られていることが分かる(少々手直しが必要かもしれません)。

## 15 I/O バッファ・フレームによるチップ形態への実装

ここまででの作業で回路ブロックのレイアウトは完成した。ただし、実際にチップとして試作を行い信号を入出力して測定を行うためには、もう一段階の作業が必要となる。VDEC で試作を行うチップではサイズがいくつかの選択肢で決まっており、最も小さくお手軽に試作できるサイズが 2.5mm × 2.5mm となっている。本トレーニングでは、このサイズのチップ上に、作成した NAND オシレータ回路を実装し、I/O バッファ (ESD 保護回路) を通して外部と信号を入出力する様に設計する手順を示す。

### 15.1 I/O バッファ

作成した回路を何らかの形で実装し、信号を入出力するためには I/O バッファが必ず必要である。I/O バッファ内には ESD 保護回路(実際は単なるダイオードもしくはダイオード接続されたトランジスタ)が必要で

あり、これが無いと、実装時または信号を入力するために配線を繋いだときなどに、ちょっとした静電気で内部のトランジスタのゲート酸化膜が破壊され、回路が壊れてしまう。そのため、特にゲート端子へ接続される配線を外部に接続する際には ESD 保護回路は必須となる。

VDEC では、名倉先生に作成いただいた簡易的な ESD 保護回路を使用することが多い。その理由として、研究目的では（売り物のような）それほど高い保護性能は必要ない点、I/O セルが占める面積が比較的小さく、内部の面積を有効に活用できる点、保護回路も 1.8V のトランジスタで設計されており、別途 3.3V の電源電圧を用意する必要がない点、簡単に使えるように配慮されて設計されている点などが挙げられる。

この資産を有効に活用し、設計を行う。

## 15.2 回路図設計

1. まずは先に作成した NAND オシレータ回路のシンボルを第 13.1 節を参照しながら作成する。形状については四角のままでも良いし、自由に分かり易く作れば良い。
2. File > New > Cellview から、testPDK ライブラリの中に CHIPTOP という名前の Schematic View を作る（名前は何でも良い）。
3. 先に作成した NAND オシレータ回路のシンボルを *i* キーで呼び出し配置する。
4. 入力信号には stdVGRohm ライブラリの中の IOBUFIN、出力信号には同ライブラリの中の IOBUFOUT を用いる。それぞれ同様に *i* キーで立ち上がるウィンドウの Browse ボタンを使い、対象のセルの symbol view を選択して配置し、オシレータと接続する。
5. 入出力ピンおよび電源・グラウンドピンを配置し、それぞれ接続する。このとき、内部回路は VDD/VSS、I/O 用は VDDIO/VSSIO とする。これらの I/O セルはトリプルウェル構造になっており、電源・グラウンドは内部回路と分離することができる。入出力ピン名は内部回路と同じ名前で良い。
6. I/O セルと内部回路を接続している配線に名前を付けておくとシミュレーションの時に動作確認がやりやすい。*1* キーを押し、Add Wire Name ウィンドウを立ち上げる。外部ピン名が EN の I/O 内側の配線は *enin*、外部ピン名が OUT の I/O 内側の配線は *outin*（小文字にして末尾に *in* を付ける）とするなど、ルールを決めておくとよい。付けたい配線名を入力し、Schematic Editor 上の対象の配線をクリックするとその配線に名前を付けることができる。
7. 全体を接続し、図 75 に示すような形に作成する。

## 15.3 シミュレーション実行

1. VDEC メニューからネットリスト出力を行う。
2. ネットリストが作成されたディレクトリに移動し、HSPICE/HSIM でシミュレーションを行う。このとき、先に作成した NAND オシレータ回路に VDDIO/VSSIO という電源端子が追加されているため、先にオシレータ単体のシミュレーションに使った *nand2\_osc.sp* ファイルをコピーしてきて、*vdd*、*vss* に関する記述をコピーして、*vddio*、*vssio* の電圧入力を忘れずに記述する。それ以外は全く同じ記述でシミュレーションできるはず。*vddio* の電圧は内部回路と同じ 1.8V である。
3. HSPICE/HSIM シミュレーションを実行し、内部回路のみの結果と比較する。信号が I/O セルを通過するために必要な遅延の分、発振回路の信号が出力端子に出てくるのが遅れているはずである（周波数は変わらないはず）。
4. うまく接続ができているようであればレイアウト作成に移る。

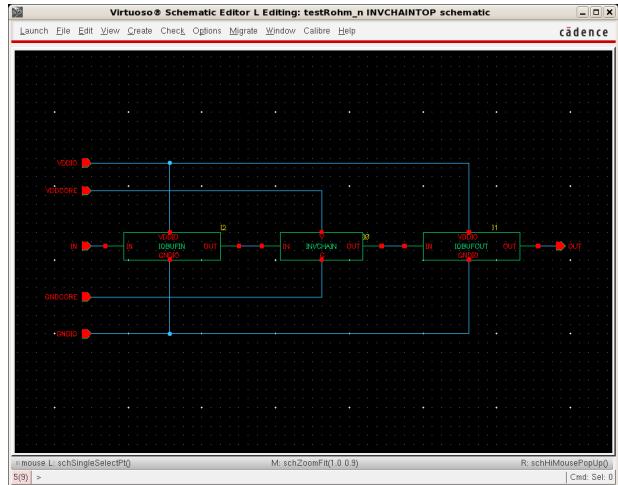


図 75: I/O セルと接続した回路図



図 76: I/O セルと接続したレイアウト

## 15.4 レイアウト設計

- まず、今回使用するフレームのレイアウトを testPDK ライブラリの中にコピーする。Library Manager で、stdVGRohm ライブラリの中の RSFRAMEWITHIO セルを選択し、さらに View 欄で layout を選択する。layout を右クリックし、開くメニューから Copy を選ぶ。二段目の To の欄で Library に testPDK を選択し、Cell 欄に先ほど作成した回路図と同じ名前 (CHIPTOP) を入力し、OK をクリックする。
- Library Manager から、testPDK の中にコピーされた CHIPTOP の layout view をダブルクリックし、開く。
- このレイアウトに NAND オシレータ回路のレイアウトを置く。*i* キーで作成したレイアウトを選択し、リング状になっている METAL5 の内側に置く。ここでは右側の外周付近に置くことにする。フレーム全体に対して、今回作成したレイアウトは非常に小さいので、適宜 Zoom-In を使って適当な位置に配置する。
- 続いて、I/O バッファを置く。今回は IOBUFIN、IOBUFOUT がそれぞれ一つずつ回路図で使用されているが、コア回路への電源・グラウンド供給用にそれぞれ一つずつ、また I/O セルへの電源・グラウンド用にさらにそれぞれ一つずつの、全体で 6 ピンが最低でも必要となる。実際のチップ試作の際には、電源・グラウンドに割り当てるピン数は多ければ多いほど、実装時のインダクタンスが（並列化により）減り、電源雑音が低減できる。今回はそれぞれ 1 ピンずつのみ配置するものとする。
- 電源リング部分にマウスカーソルを持って行くと、それぞれの PAD に繋がる部分が選択可能になっていことがある。出力バッファにしたい右側の中央付近の電源リング部分をクリックして *q* キーを押し、Cell 欄を IOBUFOUT に変更して OK とすると、この部分が出力バッファに置き換わる。
- 同様の手順で IOBUFIN を好きな位置に配置する。VDDIO/VSSIO に繋がる部分はそれぞれ IOBUFVDDIO、IOBUFGNDIO セルに置き換える。VDD/VSS に繋がる部分は IOBUFTHROUGH を使う（単純に配線が繋がっているだけのもの）。
- セルの入れ替えが完了したら内部回路との接続を行う。VDDIO/VSSIO についてはセルを置き換えた時点での接続が済んでいるのでこれ以上行う事は無い。コア回路との入出力信号および電源・グラウンドの接続を適切に行う。
- 接続が完了したら、パッドに METAL5 で Label を振る。回路図で設定したとおりのピン名を振ること。Label の Height は 50 程度、Font を roman にしておくと見やすい。図 76 に例を示す（図中のピン名やフォントが違うのは気にしないで下さい）。

- ここまで完成したら、DRC/LVS を実行し、正しくレイアウトができているか検証を行う。両者とも Pass したら、必要に応じて寄生成分の抽出を行い再度全体シミュレーション結果の確認を行う。

## 15.5 その他の作業

最終的に完成したチップレイアウトを VDEC の Website から提出することにより試作に回すことが可能となるが、その前にもういくつかの検証が必要となる。

- これまでのコア回路単位での設計ではやってこなかったが、チップ全体での密度ルールチェックが必要となる。金属配線や拡散層に密度の偏りがあると CMP (Chemical Mechanical Polishing/Planarization) や、エッチングの際に不均一性が発生して特性が変化するため、うまく製造できない。通常の DRC 実行時と同様に Calibre > nmDRC と実行し、Load Runset File で OK としたあと、左側の Rules ボタンをクリックし、最上部の DRC Rule File 欄の右側の [...] ボタンをクリックし、density.rul を選択し、OK とする。この状態で Run DRC とすると密度チェックが実行される。
- おそらくこのまま実行するといくつかの密度違反が発見されるはずである。密度チェック時には密度が足りない部分に自動的にダミーパターンが生成されているが、ACTIVE AREA に関してはダミーが発生されすぎてしまうと言う状況が発生する。これを防ぐために意図的に ACTIVE レイヤのダミー禁止レイヤ (NODMYACT) をチップ内に適当に置いておく必要がある。図 77 に例を示す。ここでは r キーで適当な大きさに四角を書いておけば良い。レイアウトを Save した後、Run DRC ボタンで再度密度チェックを行い、エラーが出なくなれば OK である。
- 次に Antenna 違反のチェックを行う。トランジスタのゲート端子に必要以上に金属配線が接続されている場合、その配線がアンテナとなり、製造プロセス中に電荷が蓄積し、トランジスタのゲート酸化膜に高電圧が印可されトランジスタが破壊されてしまう現象が起こりうる。これを防ぐため過剰な配線が接続されたトランジスタを検出しエラーとしてチェックできる。
- 既に立ち上がっている Calibre DRC ウィンドウで先と同様に、左側の Rules ボタンをクリックし、最上部の DRC Rule File 欄右側の [...] ボタンをクリックし、今度は cbu40m5\_n1c\_mod.ant ファイルを選択する。Run DRC でチェックを実行できる。今回のトレーニングでは通常はエラーは出てこないはずであるが、特に自動配置配線を行った場合などはアンテナエラーが出てくることはよくあるので、いずれにしても提出前には必ずチェックすること。
- ここまでチェックが全てパスすればチップ全体のレイアウトが完成となる。後は提出作業を行うのみである（提出作業は必要に応じて教えてもらって下さい）。
- 最後に Calibre DRC を閉じるが、このまま Runset File を上書き保存すると、次に読み込んだときの Rule File がアンテナエラー用のものとなってしまう。意図せず保存してしまったりする場合もあるので、DRC の実行時には必ずどのルールファイルを使っているか、一度チェックすること（間違ったファイルでエラーが無いと思って設計を続けると後で痛い目を見ます）。

## 16 Virtuoso ADE GXL を使用した回路最適化

Virtuoso を使用して例えばアンプのゲインを最大化したり、必要なバンド幅を満たすトランジスタサイズを自動的に探したりといった、仕様に対する回路パラメタの最適化が可能である。本節では ADE GXL を使用して、与えられた仕様に対する回路パラメタの最適化手順について示す。あまり頻繁に使う機能ではないが、今後の参考のためにここで示す。以降では、簡単な差動型のアンプ回路を例として、DC ゲインおよびバンド幅の仕様を満たすトランジスタサイズを自動で探索する手順について示す。本演習では差動アンプの設計ターゲットを DC ゲインが 20dB、Bandwidth が 1GHz を設計ターゲットとすることにする。

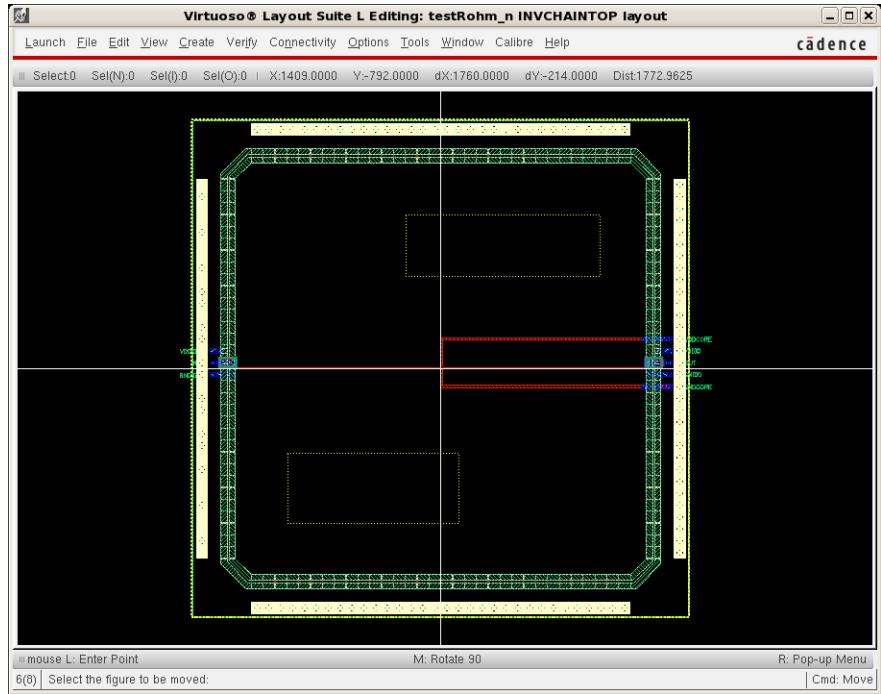


図 77: ACTIVE レイヤのダミー禁止レイヤの挿入

### 16.1 アンプ回路図の作成

まず、本節での最適化対象とするアンプの回路図を作成する。本書では図 78 に示す単純な差動アンプ回路を使用する。

1. Library Manager から、testPDK ライブラリを選択した状態で File > New > Cell View として、AMP というセルを作る。
2. 起動する回路図エディタ上で、“i” キーとして、Add Instance ウィンドウを開き、Browse ボタンを押し vdecRO180PDK ライブラリの nmos18、pmos18 の symbol を選択し、図 78 に示すような形に回路図上に配置する。この時、nmos18、pmos18 のパラメタは全て図中に示すようにデフォルトの l=180n、w=1u とする。
3. 次に “p” キーとして入出力のピンを配置する。INP、INN、NBIAS、VDD は Direction を input とし、OUT ピンは Direction を output とすること。通常は VSS のピンも配置して接続するが、今回はこの後 Virtuoso 上でシミュレーションを行うため VSS のピンは配置せず、代わりに gnd! 端子を接続する。“i” キーとして、Add Instance ウィンドウを起動し、Browse ボタンをクリックして analogLib 内の、gnd という Cell の symbol を選択する。下向きの矢印のようなインスタンスが配置できるので、図 78 に示すように VSS に当たる位置に配置すること。
4. 次に “w” キーとしてトランジスタ間の配線接続を図に示すように行う。
5. さらに、Virtuoso 上でのシミュレーション実行の際の入力電圧を設定するために電圧源インスタンスの配置を行う。図に示すように 4 つの電圧源インスタンスを配置する。今回は AC シミュレーションを行ってアンプの性能評価を行うため、電圧源として analogLib 内の vdc (DC Voltage Source) の symbol を使用する。通常のトランジスタを配置するときと同様に “i” キーとして、Add Instance ウィンドウから選択して配置することができる。
6. 各電圧源のパラメタを図中に示すように設定する。gnd!-VDD 間の vdc は DC 電圧を 1.8V、gnd!-NBIAS 間の vdc は DC 電圧を 720mV、gnd!-INP 間の vdc は DC 電圧を 900mV、AC 振幅を 1mV、gnd!-INN

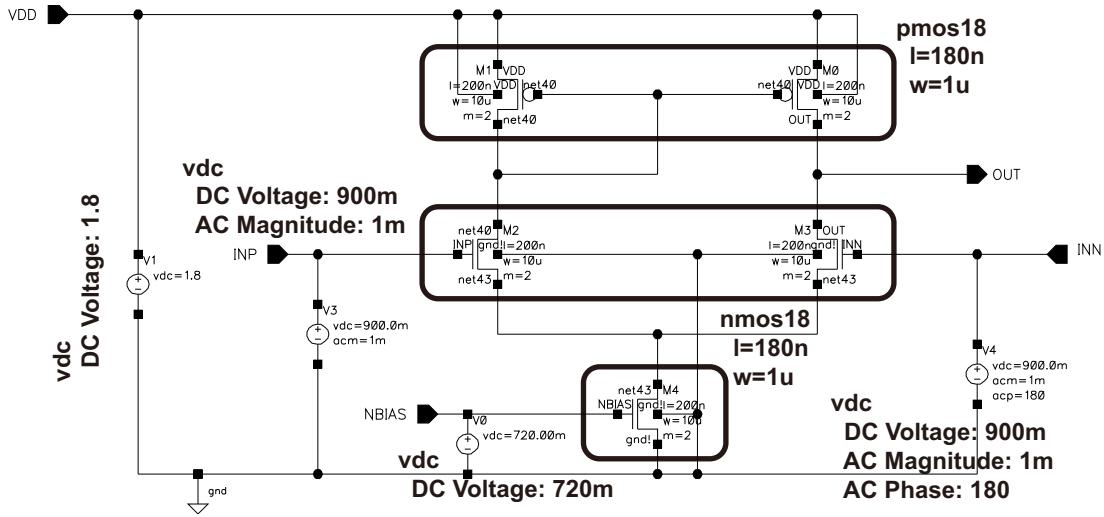


図 78: 最適化対象とするアンプの回路図



図 79: モデルファイルの設定

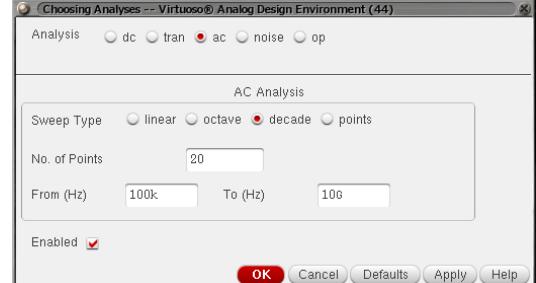


図 80: シミュレーション解析の設定

間の vdc は DC 電圧を 900mV、AC 振幅を 1mV、AC Phase を 180° となるようにそれぞれ設定する。

7. 図 78 に示す回路図が完成すれば終了。

## 16.2 ADE L によるシミュレーション環境の設定

次に、まず一度この回路パラメタの状態でのシミュレーションを実行し現状の性能を確認する。本書の前半の演習では一度 Netlist をテキストファイルで出力し、コマンドラインから hspice を実行するフローを使用したが、Virtuoso 上での自動最適化を実行するためには Virtuoso 上でシミュレーションを実行する環境を設定する必要がある。ここで設定するシミュレーション実行環境は、最適化の際にもそのまま使われる所以、この段階で正しいシミュレーション環境を作つておくことが重要である<sup>6</sup>。

- AMP 回路を作成した回路図エディタ上のメニュー左上から Launch > ADE L と実行する。
- 起動する Analog Design Environment ウィンドウのメニューから、Setup > Simulator/Directory/Host ... と実行する。
- 新たに起動するウィンドウ上で、Simulator に hspiceD を選択し、Project Directory に ~/simulation と入力されていることを確認して OK をクリックする。

<sup>6</sup>本書では ADE L を使用してシミュレーション環境を作り、それを ADE GXL で使用するフローを取るが、実際はいきなり ADE GXL でシミュレーション環境作つても構わない。今回は ADE L の使用法の練習の意味も込めてこの様なフローを取ることにする。

4. 次に、同じく Analog Design Environment ウィンドウのメニューから、Setup > Model Libraries ... と実行する。
5. 図 79 に示す、Model Library Setup ウィンドウが立ち上がるるので、hspice 用の Model File を選択する<sup>7</sup>。Click to add model file となっている箇所の右側の ... ボタンをクリックし、ROHM 提供の Model File の bu40n1.mdl を選択する。さらに下に現れる Click to add model file の欄の右側の ... ボタンを押し、ROHM 提供の Model File の bu40n1.skw を選択する。二番目に入力したモデルファイルの右側の Section の欄をクリックすると、プルダウンメニューが表示されるので、プルダウンの中から NT を選択する。さらに下に現れる Click to add model file の欄にも同様に bu40n1.skw を選択し、Section の欄にプルダウンメニューから PT を選択する。これで 1.8V 用 PMOS/NMOS のモデルファイルの設定は終わりである。Model Library Setup ウィンドウで OK をクリックして閉じる。
6. 次にシミュレーションで行う解析の種類を選択する。Analog Design Environment ウィンドウの Analyses > Choose ... と実行し、Choose Analyses ウィンドウを起動する。今回は AC シミュレーションを実行するので、図 80 に示すように ac を選択し、Sweep Type を decade、No. of Points を 20、From(Hz) を 100k、To(Hz) を 10G と設定して、OK をクリックする。Analog Design Environment ウィンドウの右上の Analyses 欄に、ac シミュレーションの設定が表示されることを確認する。
7. 次に、波形を観測する信号を選択する。Analog Design Environment ウィンドウのメニューから、Outputs > Setup ... と実行し、Setting Outputs ウィンドウを起動する。
8. 起動したウィンドウの中央付近の From Schematic ボタンをクリックすると、AMP の回路図エディタが表示されるので、波形を見たい配線を選択する。今回は OUT 端子の出力のみ観測することとするので、OUT 端子がつながっている配線部分をクリックする。すると回路図エディタ上で選択した配線に色が付きハイライトされ、Setting Outputs ウィンドウに選択した配線が順次追加されていく。他にも見たい配線があれば順次選択すること。
9. さらに、今回の設計仕様である DC ゲインと Bandwidth についても値を計算するための設定を行う。Setting Outputs ウィンドウの Calculator の欄から Open をクリックして計算式を作成する Calculator を起動する。図 81 に示すウィンドウが起動するのでまずはゲインの数式から作成する<sup>8</sup>。まず、上部の vt や vf 等の選択ができる箇所から vf を選択する。ここで vt は transient 解析の時の電圧値で、vf は AC 解析の時の電圧値を示している。vf を選択すると、回路図エディタが前面に立ち上がる所以で、まずは OUT 配線をクリックする。Calculator 上に VF("/OUT") と入力される事が分かる。これが OUT 端子の AC シミュレーション時の電圧と言う意味になる。続いて回路図エディタ上で INP 配線をクリックする。すると Calculator 上の表示が VF("/INP") に変更される。その後、右端のテンキーのボタンから割り算の "/" ボタンをクリックすると、Calculator の表示が VF("/OUT")/VF("/INP") となり、二つの信号値の割り算になる。ここまで入力できたら、次に下の様々な関数が表示されている箇所から dB20 を選択する。もし表示されていなければ関数が表示されている窓の上のプルダウンメニューから All を選択し、全ての関数を表示させて dB20 を見つけること。dB20 を選択すると Calculator 部の表示が dB20(VF("/OUT")/VF("/INP")) となり、これでゲインの周波数に対する変化の波形を示す数式ができた。
10. Setting Outputs ウィンドウに戻り、中央付近の Get Expression をクリックすると、Calculator に表示されている関数を Expression 欄に取り込むことができる。Name(opt.) 欄に Gain (任意の名前でよい) と入力し、同ウィンドウ下の Add ボタンをクリックすると Table of Outputs 欄に Gain が追加される。これで、シミュレーション実行時に Gain の波形を確認することができる。
11. 次に、ゲインがフラットなところの値を得るために数式の作成を行う。次節以降ではこの値を仕様として最適化を行っていく。再度 Calculator ウィンドウに戻り、vf をクリックした後、回路図エディタ

<sup>7</sup>本演習では、hspice 用のモデルファイルは別途 VDEC 経由で入手されていることを仮定する。

<sup>8</sup>実は私自身もこの Calculator の正しい使い方がよく分かっていません...。今後適宜追記します。

上で OUT 配線、INP 配線と順番にクリックし、Calculator ウィンドウの “/” ボタンをクリックして  $\text{VF}("/\text{OUT}"/)\text{VF}("/\text{INP}"/)$  という数式が表示されている状態にする。ここで、下の関数の欄から value 関数を選ぶと、value 関数の設定ウィンドウが表示されるので、Interpolate At の欄に 1 を入力し、その他の設定はそのままで、最下部の OK をクリックする。Calculator の表示部は  $\text{value}(\text{VF}("/\text{OUT}"/)\text{VF}("/\text{INP}"/) 1)$  となる。さらに引き続き、関数から、dB20 をクリックすると、Calculator 表示部が  $\text{dB20}(\text{value}(\text{VF}("/\text{OUT}"/)\text{VF}("/\text{INP}"/) 1))$  となる。これでゲイン値の関数作成は終了である<sup>9</sup>。Setting Output ウィンドウに戻り、中央付近の Get Expression をクリックし、新たに作った数式を取り込む。Name(opt.) の欄に DC Gain (任意の名前でよいが前回と異なる名前にすること) と入力し、同ウィンドウ下の Add ボタンをクリックして Table of Outputs に DC Gain を追加する。

12. 次に、バンド幅の数式の作成を行う。またまた Calculator ウィンドウに戻り、vf をクリックした後、回路図エディタ上で OUT 配線、INP 配線と順番にクリックし、Calculator ウィンドウの “/” ボタンをクリックして  $\text{VF}("/\text{OUT}"/)\text{VF}("/\text{INP}"/)$  という数式が表示されている状態にする。次に、関数の欄から bandwidth 関数を選択すると、bandwidth の設定ウィンドウになるので Db の欄に 3 が、Type の欄に low が入力されていることを確認し<sup>10</sup>、最下部の OK をクリックする。Calculator 表示部が  $\text{bandwidth}(\text{VF}("/\text{OUT}"/)\text{VF}("/\text{INP}"/) 3 \text{ "low"})$  となっていることを確認し、Setting Outputs ウィンドウに戻り、Get Expression ボタンをクリックして数式を取り込む。Name(opt.) 欄に Band Width (任意でよいが他のものと異なる名前にすること) と入力し、Add ボタンをクリックする。最終的に Table of Outputs 欄に、OUT、Gain、DC Gain、Band Width の 4 つが設定されていることを確認し、Setting Outputs ウィンドウの OK ボタンを押し、同ウィンドウを閉じる。一緒に Calculator ウィンドウも閉じてしまってよい。
13. ここまで設定が終わった後の Analog Design Environment ウィンドウは、図 82 に示すようになる。Analyses 欄と Outputs 欄が正しく設定されているか確認すること。
14. 設定が正しく完了したら、Analog Design Environment ウィンドウ右下にある緑色の再生ボタンをクリックしてシミュレーションを実行する。自動的にネットリスト出力が行われ、hspice が実行される。終了後、設定した OUT 出力の AC 解析結果と、Gain の周波数依存性の波形が表示される。波形の X 軸の数字のあたりで右クリックし、軸の表示を Log 表示に変更して結果を確認すると、図 83 のように表示される。また、Analog Design Environment ウィンドウの Outputs 欄で、設定した DC Gain と Band Width の値が計算されることを確認すること。Gain の周波数依存性のグラフと比較して正しい値が算出されていることも確認すること。
15. ここまでで、現状の回路でどの程度の性能が出ているのかを確認することができた。今回設計する回路の要求仕様は、DC ゲインが 20dB、Bandwidth が 1GHz となっているため現状の回路では DC ゲインは十分出ているが Bandwidth が足りない状態である。次節でトランジスタのパラメータを自動的にスイープし、回路性能を満たす値を求める手順について示す。
16. 最後に、ここでの設定を最適化環境でも使用するため、state の保存を行う。Analog Design Environment メニューの Session > Save State ... として、Saving State ウィンドウを起動し、Save As に AMP\_opt (任意の名前でよい) と入力して、最下部の What to Save 欄の全てにチェックが入っていることを確認して OK とする。これで先ほどまで設定した項目が保存された。Analog Design Environment ウィンドウは Session > Quit として閉じてしまってよい。

### 16.3 最適化対象パラメタの設定

本節では、最適化対象となるパラメタを設定する。ADE GXL での最適化では、スイープするパラメタの、範囲とステップを適宜設定する必要がある。今回はアンプを構成する全トランジスタの l および w の値をス

<sup>9</sup> 詳細な使い方の解説は VDEC のセミナーを別途受講してください。

<sup>10</sup> Low Pass 型で 3dB ダウンの点を取るという意味だと思います....。

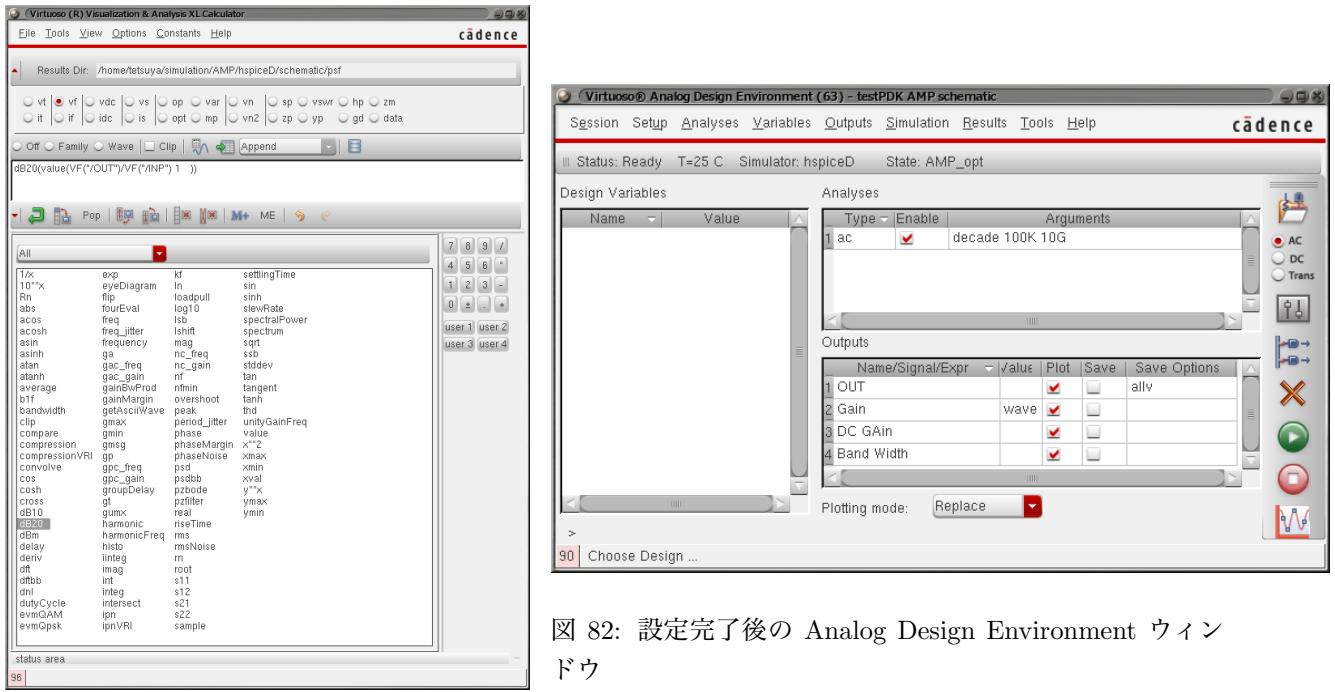


図 81: Calculator ウィンドウ

図 82: 設定完了後の Analog Design Environment ウィンドウ

イープし、使用を満たす値を探索させる。このとき、差動アンプでは差動関係にあるトランジスタペアのパラメタは同一である必要がある (Matching)。本節では二つ以上のトランジスタの Matching を設定する方法についても記載する。

1. アンプを設計した回路図エディタのメニュー左上から Launch > ADE GXL と実行する。立ち上がる小さいウィンドウで Create New View を選択し、OK とする。次に起動するウィンドウで Cell が AMP、View が adexl、Type も adexl となっていることを確認してさらに OK とすると、図 84 に示すウィンドウが起動する。このウィンドウ上で以後の作業を行う。
2. 図 84 中に示すように、左端に出ている Data View の小窓の一番上に Tests という項目があるので、その先頭についている + マークをクリックして展開し、薄字の Click to add test をクリックする。Choosing Design というウィンドウが一番上に立ち上がる所以、今最適化の対称となっている AMP を選択して OK とする。
3. もう一つ新規に起動している ADE XL Test Editor ウィンドウのメニューから、まず Setup > Simulator ... を実行し、Simulator のプルダウンメニューから hspiceD を選択して OK とする。
4. 再度 ADE XL Test Editor ウィンドウの Session > Load State ... メニューから Loading State ウィンドウを起動する。Directory Options 欄の Library に testPDK、Cell に AMP、Simulator に hspiceD が選択されていることを確認し、State Name の欄で、先ほど保存した AMP\_opt を選択して OK をクリックする。ADE XL Test Editor ウィンドウに、先ほど設定した AC シミュレーション、Outputs が表示され、シミュレーション結果の波形ウィンドウが新規に立ち上がれば設定の読み込みがうまくいっている。この状態になつたら ADE XL Test Editor ウィンドウを Session > Quit として閉じて、波形ウィンドウも×印をクリックして閉じてしまつてよい。
5. この状態で一度シミュレーションを実行し、前節までで行ったシミュレーション結果が再現できるか確認を行う。ADE GXL ウィンドウに戻り、図 85 に示すように、右上に出ている緑色の再生ボタンの左側のプルダウンメニューのところに、Single Run, Sweeps and Corners と表示されていることを確認して、再生ボタンをクリックすると、Single Run シミュレーションが実行される。再度波形ウィンドウが表示

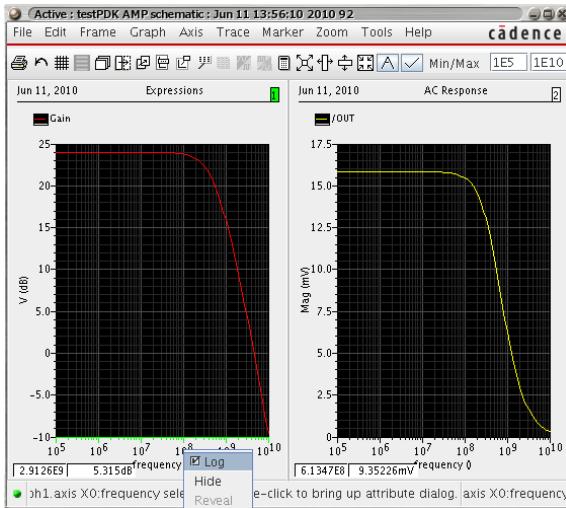


図 83: AC シミュレーション結果波形

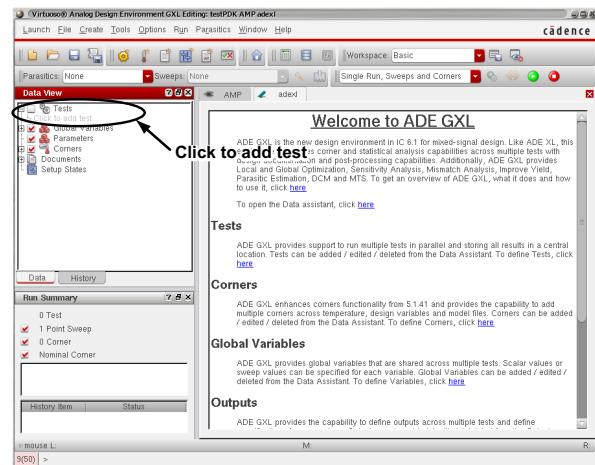


図 84: ADE GXL 新規ウィンドウ

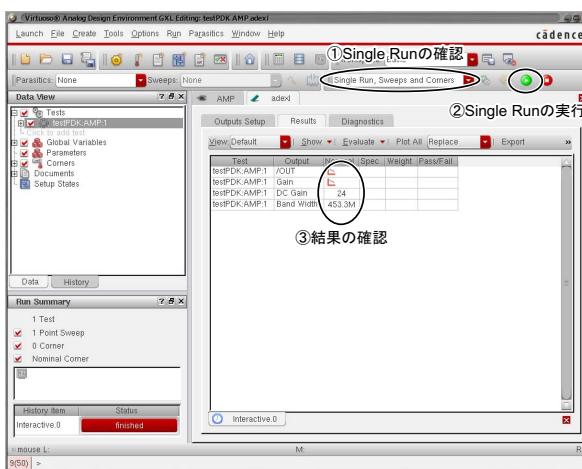


図 85: Single Run 実行後の ADE GXL ウィンドウ

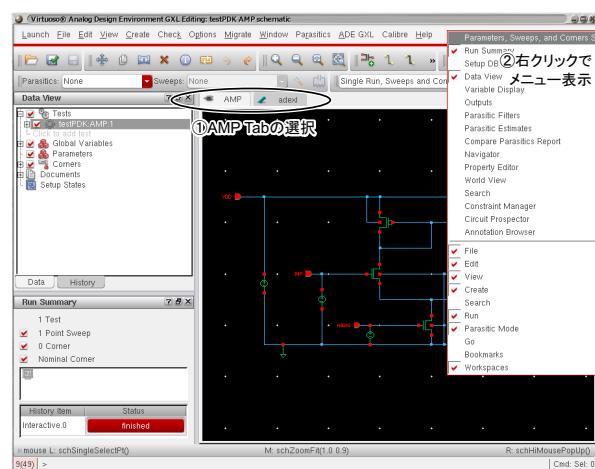


図 86: パラメータセットアップ

され、ADE GXL ウィンドウには結果が表示される。前節で行ったシミュレーションの結果と同じ値が表示されていることを確認すること。ここまでで、ADE GXL 環境でのシミュレーション設定が完了した。

6. 次に、最適化時にスイープする変数の設定を行う。図 86 に示すように、ADE GXL ウィンドウ内のタブから AMP と書いてあるタブをクリックし、回路図を表示する。その後、同ウィンドウの右上のメニューと Cadence ロゴの間の何もない所を右クリックすると表示されるメニューの一一番上の Parameters, Sweeps, and Corners Setup を選択すると、ADE GXL ウィンドウの右側にパラメタ設定用のウィンドウが表示される。このウィンドウを使用して最適化でスイープするパラメタを設定する。
7. まず、Tail の電流源になっている NMOS トランジスタに探索範囲を設定する<sup>11</sup>。Tail のトランジスタを選択すると、右上の中窓の Inst の所に選択した NMOS トランジスタのインスタンス名が表示される。インスタンス名をダブルクリックしプルダウンメニューを開くと各パラメタの値が表示されるので、1 の値をクリックして選択し、180n と表示されている箇所をもう一度クリックすると、値が入力できる状態になる。この状態で 1 の値に 180n:20n:300n とコロン区切りで入力する。この意味は 180n から 300n まで 20n 区切りで値を探索しなさいという意味である。さらに同様にして w の値には 1.0u:1.0u:20u と

<sup>11</sup> パラメタの設定順によってスイープ時の探索順が決まるため、パラメタを設定する順番が収束時間に影響する。なるべく達成したい仕様に依存性の高いパラメタから順に設定した方が収束が良さそうである。自動で最適化してくれるといつても、最低限の設計知識が必要となる。

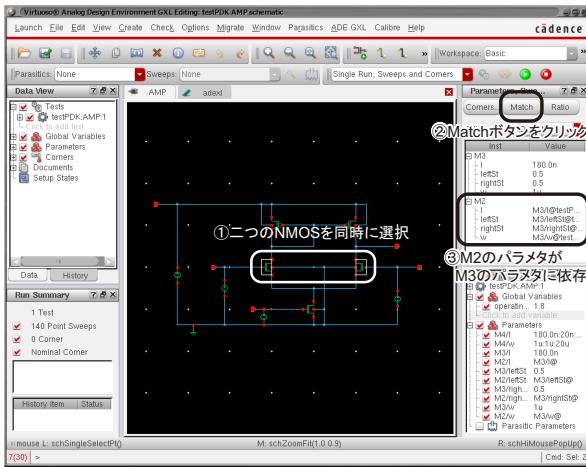


図 87: 二つのトランジスタの Match を取る

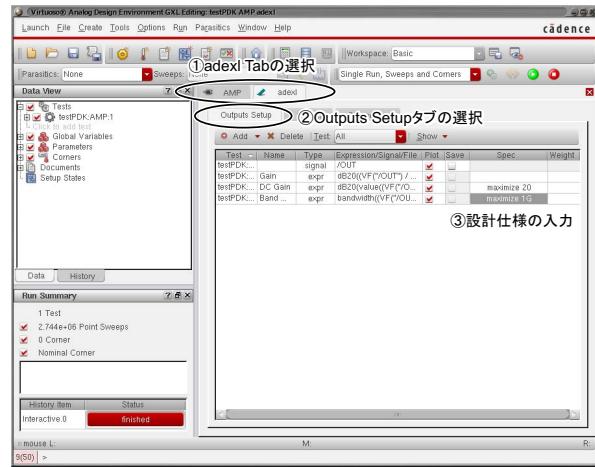


図 88: アンプの設計仕様の入力

入力する。w は 1.0u から 20u まで 1.0u 区切りで値を探索させる。実際に自分の設計に適用する際には探索範囲には適宜必要な範囲およびステップを指定すること

8. 次に、入力の NMOS トランジスタ差動対のパラメタを設定する。この二つのトランジスタは同じパラメタで Matching を取る必要がある。まず、二つのトランジスタを同時に選択する。やり方は左ドロップの領域選択で二つの NMOS を選択するか、一つの NMOS をクリックした後 Shift キーを押しながらもう一つの NMOS をクリックするかのどちらかでできる。二つの NMOS を選択すると、右上の小窓の Inst のところに選択した二つの NMOS のインスタンス名が表示される。ここでその上の Match ボタンをクリックすると、図 87 に示すように二つのトランジスタが Match される。それぞれのインスタンス名をダブルクリックしてプルダウンメニューを開き、片方のトランジスタのパラメタがもう片方のパラメタを引用していることを確認すること。図に示す例では、トランジスタ M2 の各パラメタがトランジスタ M3 のパラメタを引用する形になっている。
9. 次に、Matching を取ったパラメタのスイープ範囲とステップを設定する。Matching 後に基準となっている側のトランジスタ (パラメタに数字が入っている方。例の場合は M3。) の 1 の値に 180n:20n:300n とコロン区切りで入力する。基準側 (マスターデバイス側) でこの様に設定すると、Matching している側も同じ様に値が探索される。同様にして、w の値には 1.0u:1.0u:20u と入力する。
10. 次に、カレントミラーを構成する二つの PMOS にも Matching を設定し探索範囲を設定する。二つの PMOS トランジスタを同時に選択し、右上の Match ボタンをクリックする。マスターデバイスになっている側 (パラメタ値に数字が入っている方) の 1 の値に 180n:20n:300n と入力し、さらに w の値に 1.0u:1.0u:20u と入力する。
11. 以上で最適化の対象となるパラメタとその探索範囲の設定が終了した。値を入力し終わったら、ADE GXL ウィンドウの左上のフロッピーディスクにチェックマークがついたボタンを押し、Check and Save を行っておく。次に、最適化の目標となる性能仕様 (スペック) の設定を行う。

## 16.4 性能仕様の設定

次節の最適化実行時の目標となる性能仕様の設定を行う。今回は、初めに述べたようにアンプの性能仕様として DC ゲインが 20dB、Bandwidth が 1GHz という値を用いる。ADE GXL でこの設計仕様を入力する。

1. 図 88 中に示すように、ADE GXL ウィンドウ内のタブから、adexl タブを選択して再度シミュレーション結果が表示された状態にする。

2. adexl タブの中のタブから Outputs Setup タブを選択する。
3. 各出力パラメタがテーブルで表示されている右側に Spec という欄がある。この欄に必要な仕様を入力していく。今回は DC Gain と Band Width にスペックを設定する。
4. DC Gain の Spec の欄をクリックして選択し、もう一度クリックするとプルダウンメニューが表示されるので、その中から maximize を選択する。選択すると右側に値を入力する欄が現れるので、その中に設計仕様である 20 を入力する。この意味は DC Gain の値を 20 を越える範囲でなるべく大きくしないと言う意味である。
5. さらに同じ手順で Band Width の仕様を入力する。同様に Band Width の Spec の欄を 2 回クリックし maximize を選択する。値の欄に 1G と入力する。
6. 仕様の設定が終了すると、図 88 に示すような状態になる。

## 16.5 ADE GXL による回路最適化

ここまでで回路最適化の準備が整った。この節では実際に最適化を実行する。最適化手順には色々あるが、ここではまず Global Optimization を実行して広い範囲の探索を行い、その結果から最適化の起点となる Reference Point を設定し、その後 Local Optimizatin を実行して Reference Point の周辺での最適点の探索を行うことで回路最適化を行うフローを実行する。

1. ADE GXL ウィンドウのメニューから Run > Global Optimization ... と実行して Global Optimization ウィンドウを起動する。Evaluation に Conditional が選択され、All Specs Met にチェックが入っていることを確認する。さらに今回はシミュレーション時間の短縮のためシミュレーションを行う回数を制限する。Point Limit にチェックを入れ、回数を 20 と入力する。本来はなるべく良い Reference Point を見つけるため、時間の許す限り多くの回数の探索を行うべきである。図 89 に示す設定となったら OK をクリックすると Global Optimization が実行される。
2. Point Limit で設定した回数のシミュレーション (hspice) が実行され、ADE GXL ウィンドウには図 90 に示すように、仕様に対して性能が良い順に上から結果が表示される。結果のウィンドウで緑色で表示されている所は仕様を満たしている値、赤で表示されている所は仕様を満たしていない値、黄色で表示されている所は近い値が出ているが仕様を満たしていない値をそれぞれ示している。本例では 7 番目にシミュレーションを実行した時の結果が最も仕様に近く、DC Gain は仕様を満たしているが、Band Width がまだ仕様に達していない状態である。
3. 次にこの点を Reference Point として、Local Optimization を実行し、この点の周りの探索により仕様を満たすパラメタを見つける。図 91 に示すように、結果の表示されているウィンドウの Parameters: と表示され各トランジスタのパラメタ値が表示されている部分を右クリックし、出てくるメニューから Create Reference Point を選択する。すると、図 92 に示すようなウィンドウが立ち上がり、この時のパラメタ値が表示されるのでそのまま OK とする。これで Reference Point が設定された。
4. 設定された Reference Point の周りで Local Optimization を実行する。ADE GXL ウィンドウのメニューから、Run > Local Optimization ... と実行し、Local Optimization ウィンドウを表示させる。Effort に Fine、Evaluation に Conditional、Stopping Criteria の All Specs Met にチェックが入っていることを確認し、OK をクリックすると Local Optimization が実行される。
5. Local Optimization を実行すると、数十回のシミュレーション実行の後、図 93 に示すように、二つの設計仕様を満たすデザインが発見され、実行が終了する。

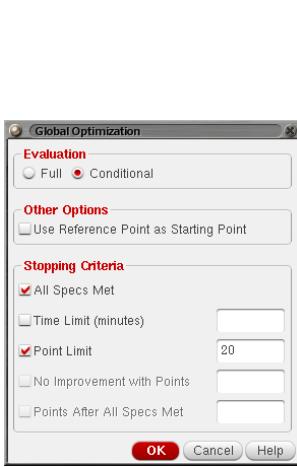


図 89: Global Optimization ウィンドウ

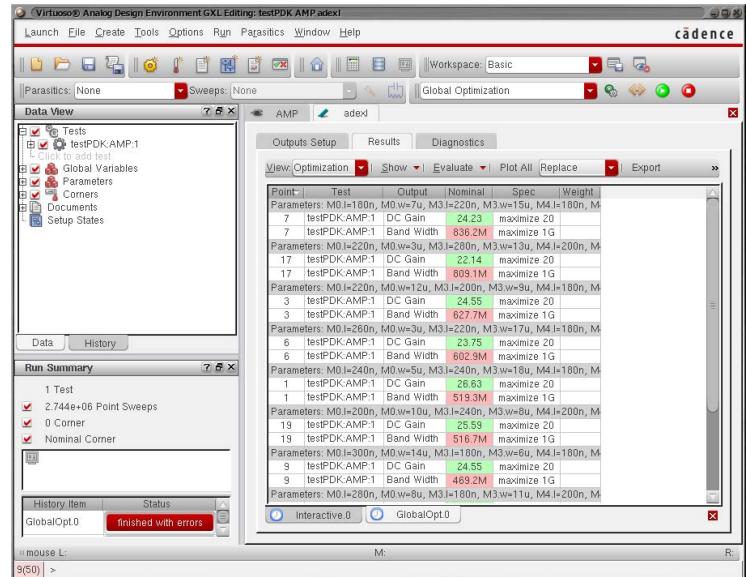


図 90: Global Optimization 実行後の ADE GXL ウィンドウ

## 16.6 最適化結果の確認および回路図への反映

以上で与えたスペックに対して、設定したスイープ範囲内の最適化が実行された。本節では簡単な結果の解析手順について示す。

1. ADE GXL ウィンドウのメニューから、Window > Assistants > Variable Display と実行すると、図 94 に示すように ADE GXL ウィンドウの下部に、最適化結果の変数の値が表示される。Results タブ内の Parameters と書いてある灰色の部分をクリックして Points をいろいろと選択し、結果の表示が変わることを確認すること。Variable Display ウィンドウでは、使用している値が指定したスイープ範囲の上限または下限に当たっている場合に数値が赤字で表示される。この場合、スイープ範囲を広げることでより最適なパラメタが見つかる可能性がある。
2. 次に、図 95 に示すように、左上の Data View の所の History タブを選択し、最後に実行した LocalOpt.0 を右クリックすると表示されるメニューから、Sensitivity Results を選択する。すると、図 96 に示されるような、各パラメタが指定した性能指標に対してどの程度の相関関係にあるかを示す結果(感度マトリックス)が表示される。傾きの方向が相関の方向(正か負か)を示し、色の濃さと四角の細さが相関の強さを示している(細くて濃いほど相関が強い)。これにより、どのパラメタが与えた設計仕様に対して強い相関性を持っているかを判断することができ、再度の最適化の際に参考にすることができる。
3. さらに、Show Sensitivity ウィンドウで、感度マトリックスのボックスでダブルクリックすると、性能指標対パラメタの散布図を確認することができる。試しに、相関関係の強い所をダブルクリックしてみる(本例では Band Width と M4/w の相関をダブルクリックしてみる)と、図 97 に示すような散布図を確認することができ、パラメタの変化に対して性能指標がどのように変化するかを確認することができる。
4. 最適化の結果が確認でき、得られたパラメタで良ければ、最後にそのパラメタを回路図に反映させる。図 98 に示すように、ADE GXL ウィンドウの結果が表示されている箇所で、パラメタを反映したい結果の Parameters: と表示されている灰色の所で右クリックし、出てくるメニューから Backannotate を選択する。これで得られたパラメタが回路図に反映される。同ウィンドウの AMP タブをクリックして回路図を表示し、正しくトランジスタのパラメタが反映されていることを確認すること。

以上で回路パラメタ最適化手順は終了である。あとは得られたパラメタを元に本書前半で書かれた手順を参考にしてレイアウトを行い、検証作業を終了すれば良い。

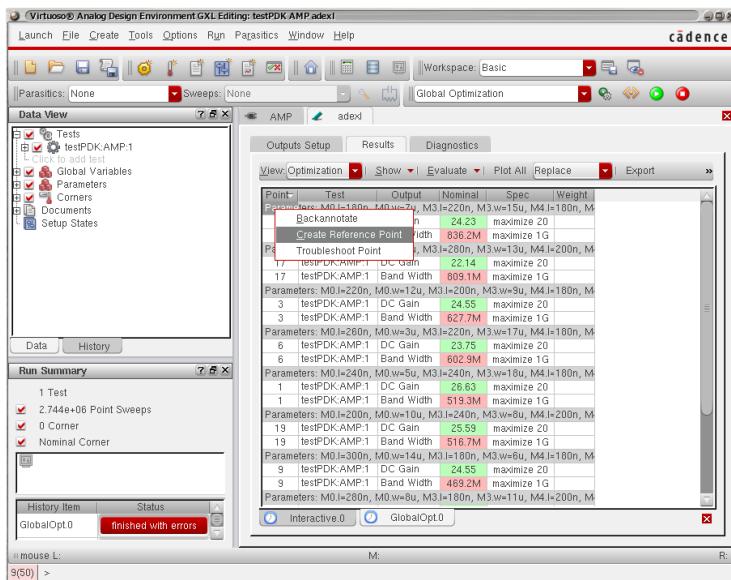


図 91: Reference Point の設定

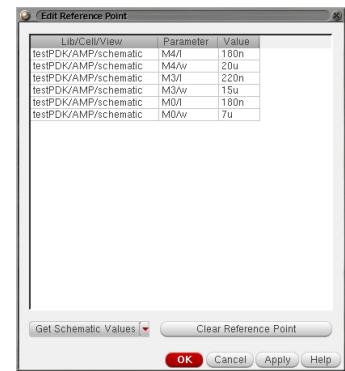


図 92: Edit Reference Point ウィンドウ

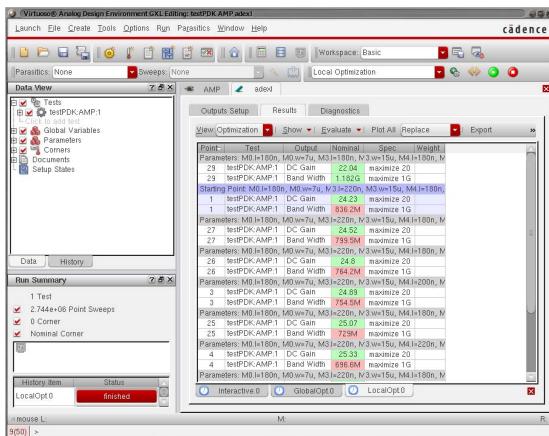


図 93: Local Optimization 後の表示

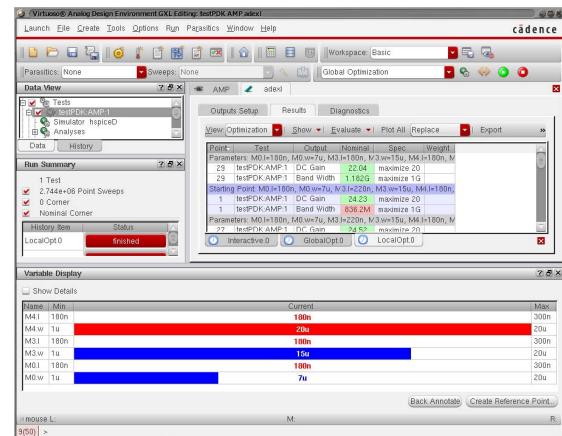


図 94: Variable Display による結果の確認

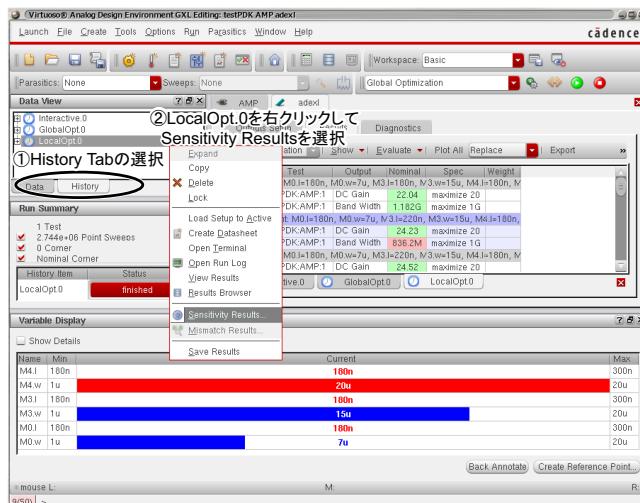


図 95: Sensitivity Results の表示

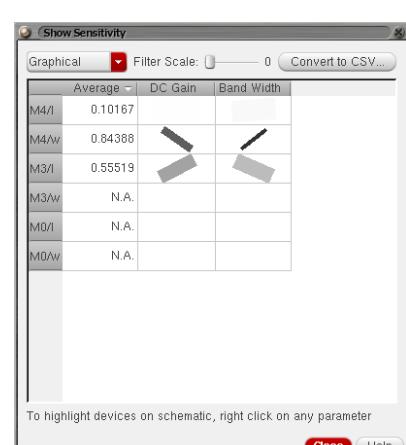


図 96: 各パラメタの Sensitivity の確認

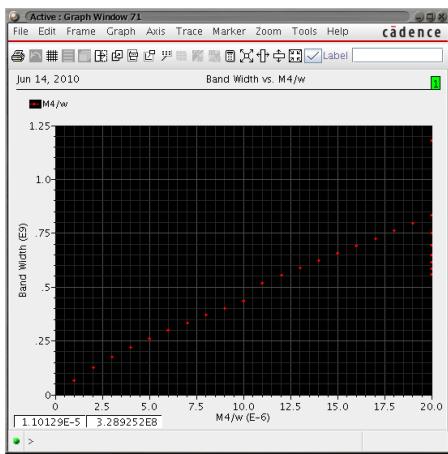


図 97: Band Width のトランジスタの  $w$  値に対する依存性

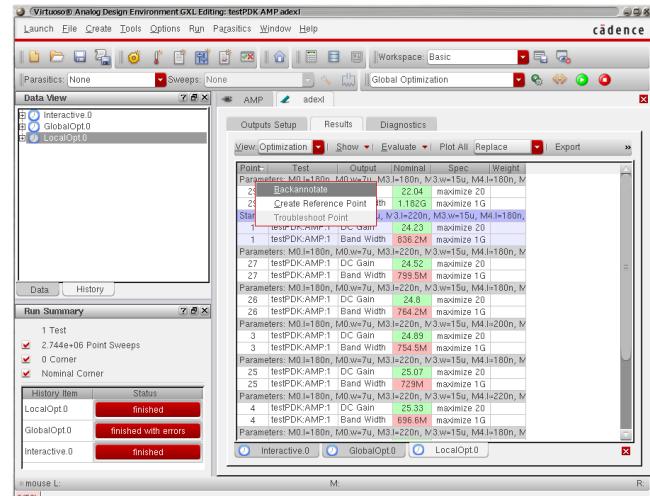


図 98: 得られたパラメタの Schematic に対する Backannotate

## 17 改変歴

### 17.1 Ver.1 平成 30 年 4 月 1 日

- 初版

### 17.2 Ver.2 平成 31 年 3 月 28 日

- HSPICE による DC/AC、Monte Carlo、雑音シミュレーションの説明を追加

## Appendix: 現バージョンの PCell の仕様

- オフグリッドエラー回避のため、トランジスタの L, W は  $0.02\mu\text{m}$  ステップでの変更に限定されている。
- オフグリッドエラー回避のため、3.3V トランジスタの PCell には DIFF2 レイヤを含めていない。そのため 3.3V トランジスタ使用の際には DIFF2 レイヤは最後に自分で描画する必要がある。(DIFF2 は Minimum Grid が 0.04 なので、自分で描画する際にも Off Grid しないように要注意。)
- 抵抗は N+ Poly w/i Silicide(npres\_s)、N+ Poly w/o Silicide(npres)、P+ Poly w/i Silicide(ppres\_s)、P+ Poly w/o Silicide(ppres)、N+ Diff w/i Silicide(ndres\_s)、N+ Diff w/o Silicide(ndres)、P+ Diff w/i Silicide(pdres\_s)、P+ Diff w/o Silicide(pdres)、LPP-H(lpjh) をサポート。
- こちらもオフグリッドエラー回避のため、抵抗の L, W は  $0.04\mu\text{m}$  ステップでの変更に限定されている。
- 抵抗の PCell の Property は、抵抗の幅 W と所望の抵抗値 R を入力すると L を自動的に計算する仕様となっている。この際、入力できる W は  $0.04\mu\text{m}$  ステップになっており、L の値が  $0.04\mu\text{m}$  ステップになるように実際の抵抗値を入力した値に近い値に自動的に設定する。Actual Res. Value に表示される抵抗値が実際に Simulation や LVS で使われる値となる。
- 設定できる抵抗値はシート抵抗値 ( $\Omega/\text{sq.}$ ) よりも小さい値にはできない。つまり、L よりも W の方が大きい抵抗はサポートしない。もし小さい抵抗値を使用したければ複数個の抵抗を作つて並列に接続してください。
- LPPH 抵抗もシート抵抗値 ( $1050\Omega$ ) よりも小さい抵抗値をサポートしない。LPPH では抵抗値の計算にコンタクト抵抗等が含まれるので、設定できる L の値は W の値よりも小さくなることがある。
- 容量は MIM Cap. をサポート。抵抗と同様にオフグリッドエラー回避のため容量の L, W は  $0.04\mu\text{m}$  ステップでの変更に限定されている。
- こちらも抵抗と同様に、容量の PCell の Property は、容量の幅 W と所望の容量値 C を入力すると L を自動的に計算する仕様となっている。この際、入力できる W は  $0.04\mu\text{m}$  ステップになっており、L の値が  $0.04\mu\text{m}$  ステップになるように実際の容量値を入力した値に近い値に自動的に設定する。Actual Cap. Value に表示される容量値が実際に Simulation や LVS で使われる値となる。
- MIM Cap. の L, W の最小値はデザインルールで  $4.00\mu\text{m}$  と規定されている。大きな W の値に対して小さな容量値を入力して L が  $4.00\mu\text{m}$  を下回ってしまうような場合には、L の値を  $4.00\mu\text{m}$  として Actual Cap. Value が計算される。
- MIMCAP の METAL4 から METAL5 に持ち上げて取り出す端子は PCell では上下に配置されているが、不要であれば Property から Terminal for Bottom Metal のチェックを外して使うことができる。その場合は適宜自分で端子を追加して接続を行うこと。