

アナログ開発ツールの使い方 (GF180MCU)

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

アジェンダ

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

お詫び

スライドの差し替えが間に合わなかったため、
一部SKY130のスライドを流用しています

アナログLSIの設計フロー

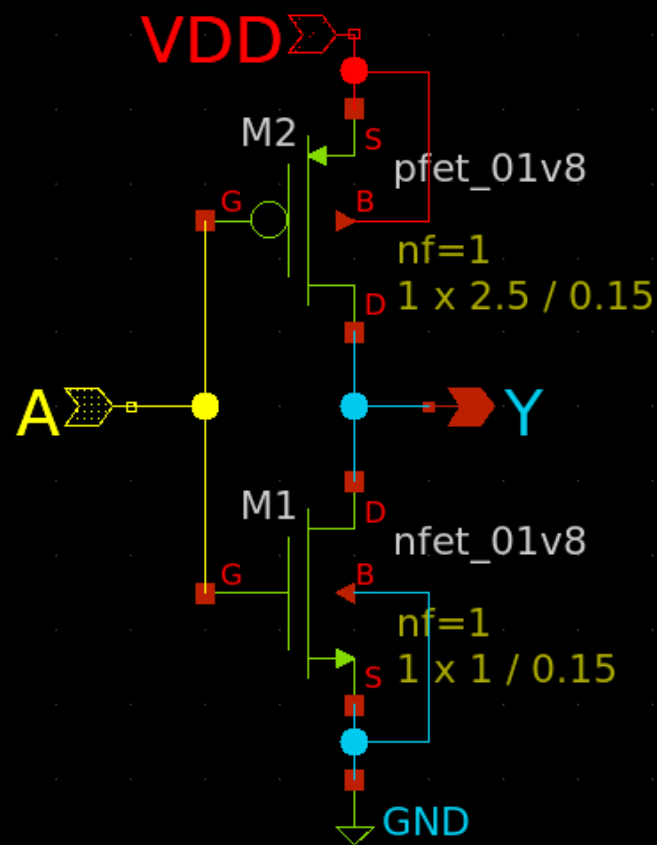
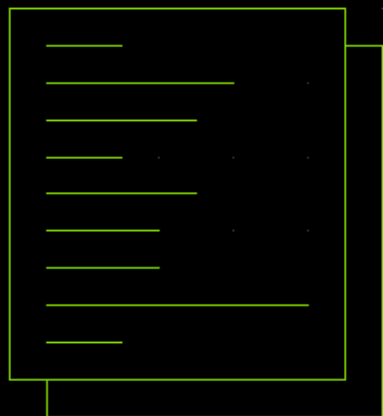
どのような設計をするか、どのような開発ツールを使用するか

アナログLSIの設計フロー

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

回路図(ベンチマーク)を描いて...

MODELS

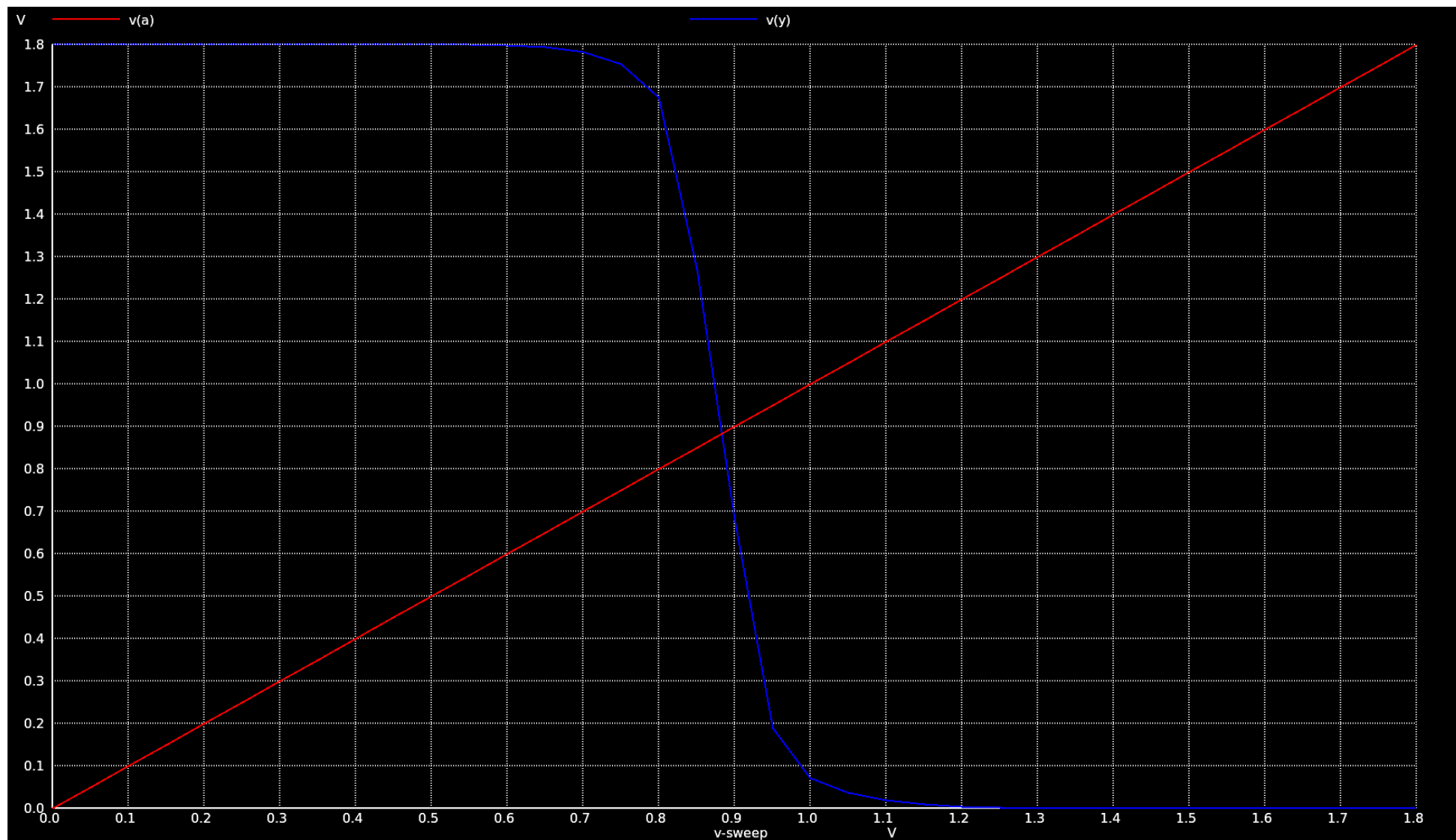


COMMANDS

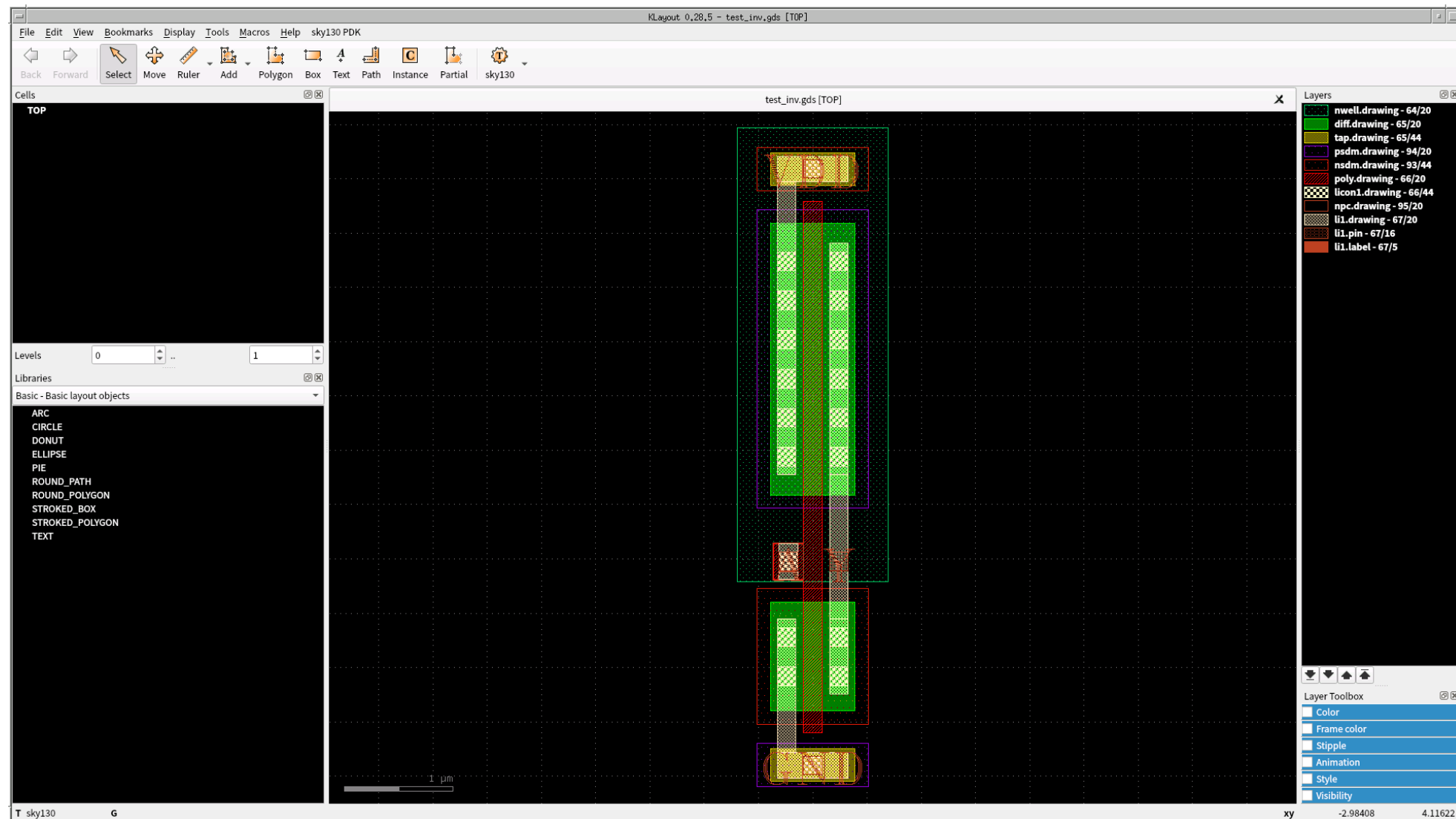
SIM=ngspice

```
VD VDD 0 dc 1.8
VA A 0 dc 0
.control
save all
dc VA 0 1.8 0.05
plot v(a) v(y)
.endc
```

論理検証をして...

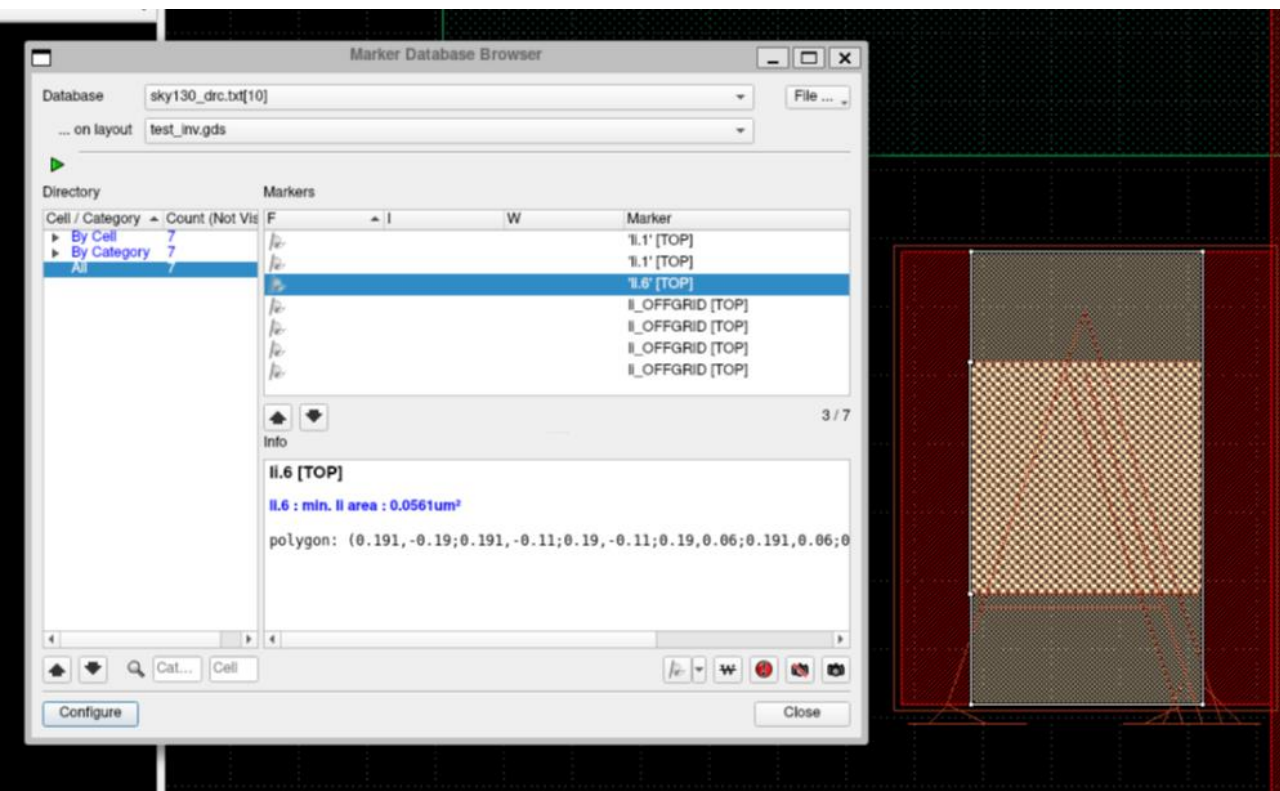


レイアウトを描いて...

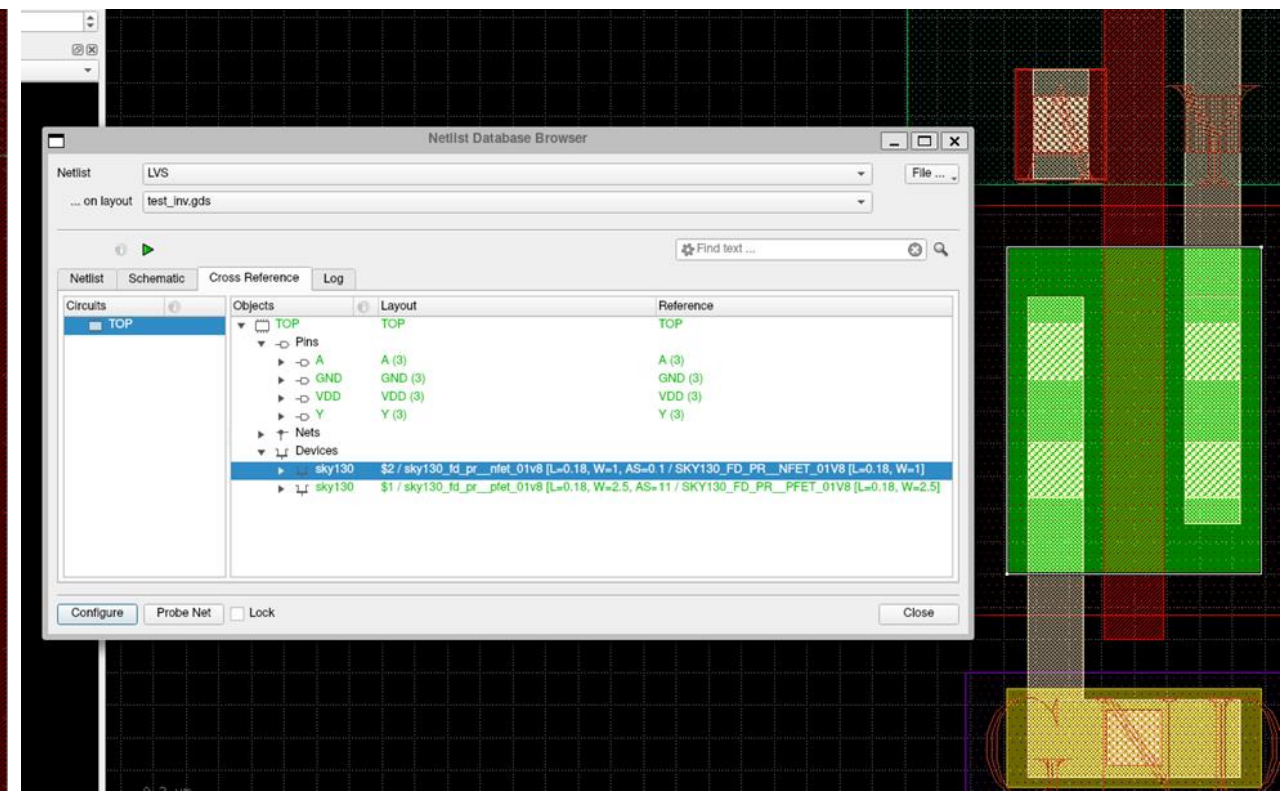


レイアウトを検証して...

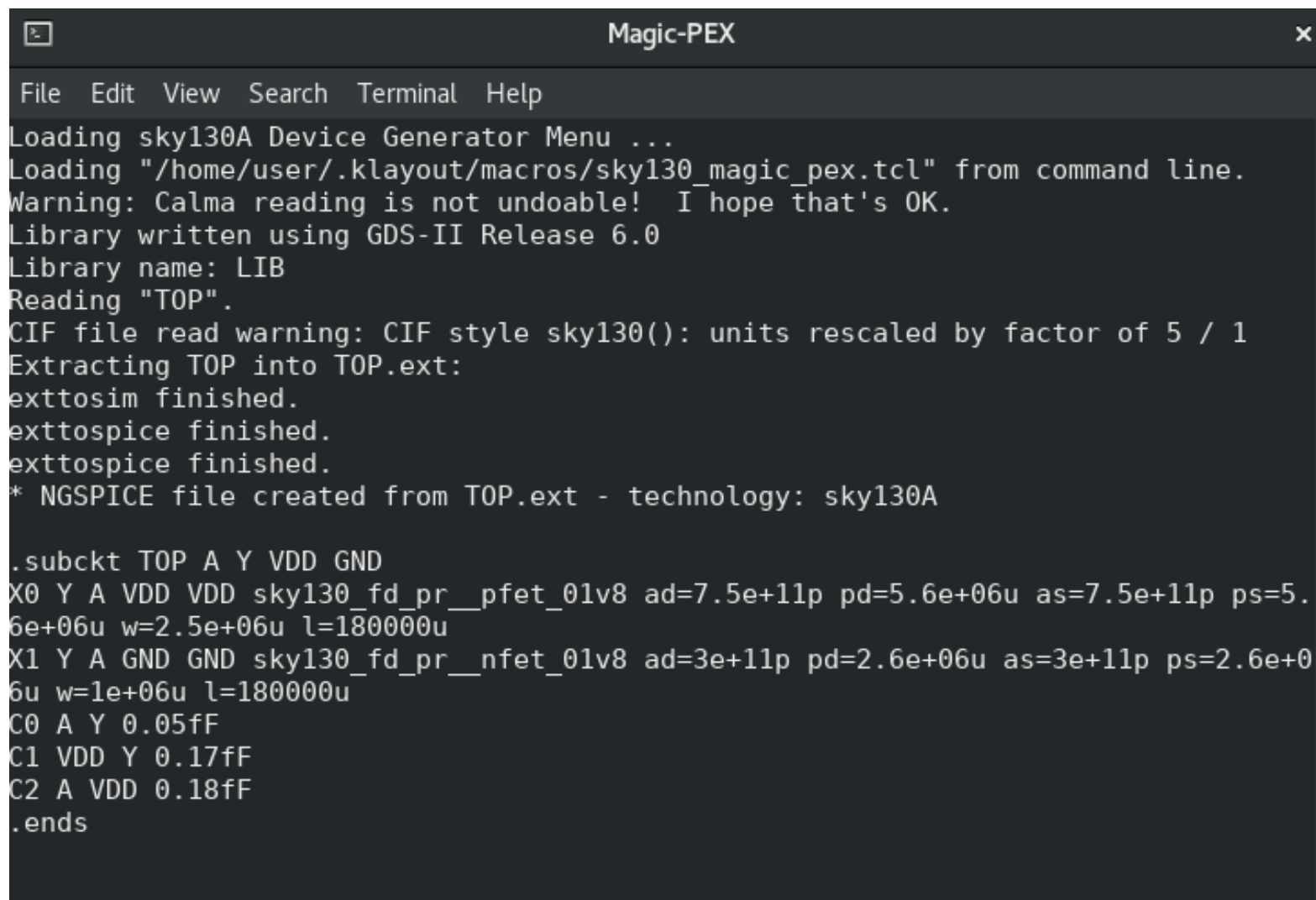
DRC



LVS



寄生成分を抽出して...



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

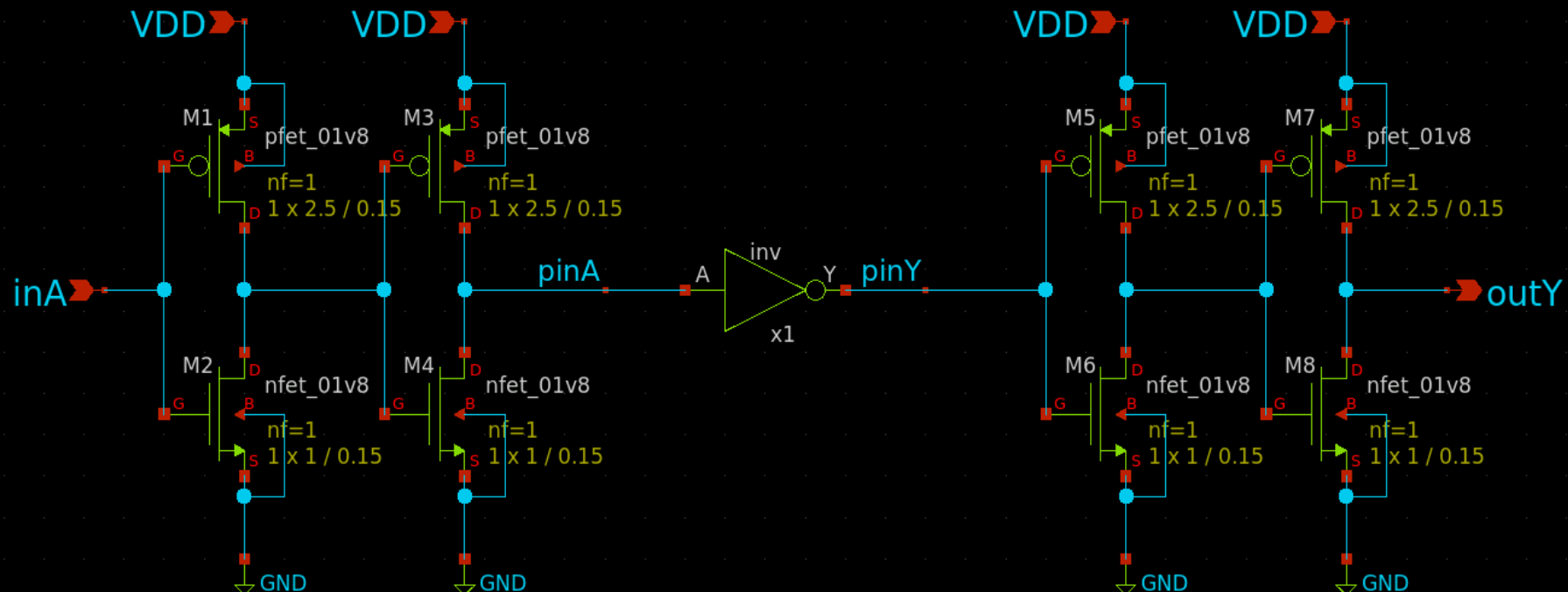
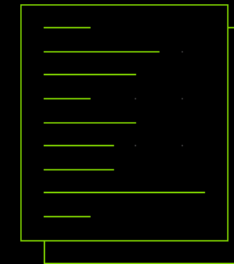
.subckt TOP A Y VDD GND
X0 Y A VDD VDD sky130_fd_pr__pfet_01v8 ad=7.5e+11p pd=5.6e+06u as=7.5e+11p ps=5.
6e+06u w=2.5e+06u l=180000u
X1 Y A GND GND sky130_fd_pr__nfet_01v8 ad=3e+11p pd=2.6e+06u as=3e+11p ps=2.6e+0
6u w=1e+06u l=180000u
C0 A Y 0.05fF
C1 VDD Y 0.17fF
C2 A VDD 0.18fF
.ends
```

ベンチマークを作成して...

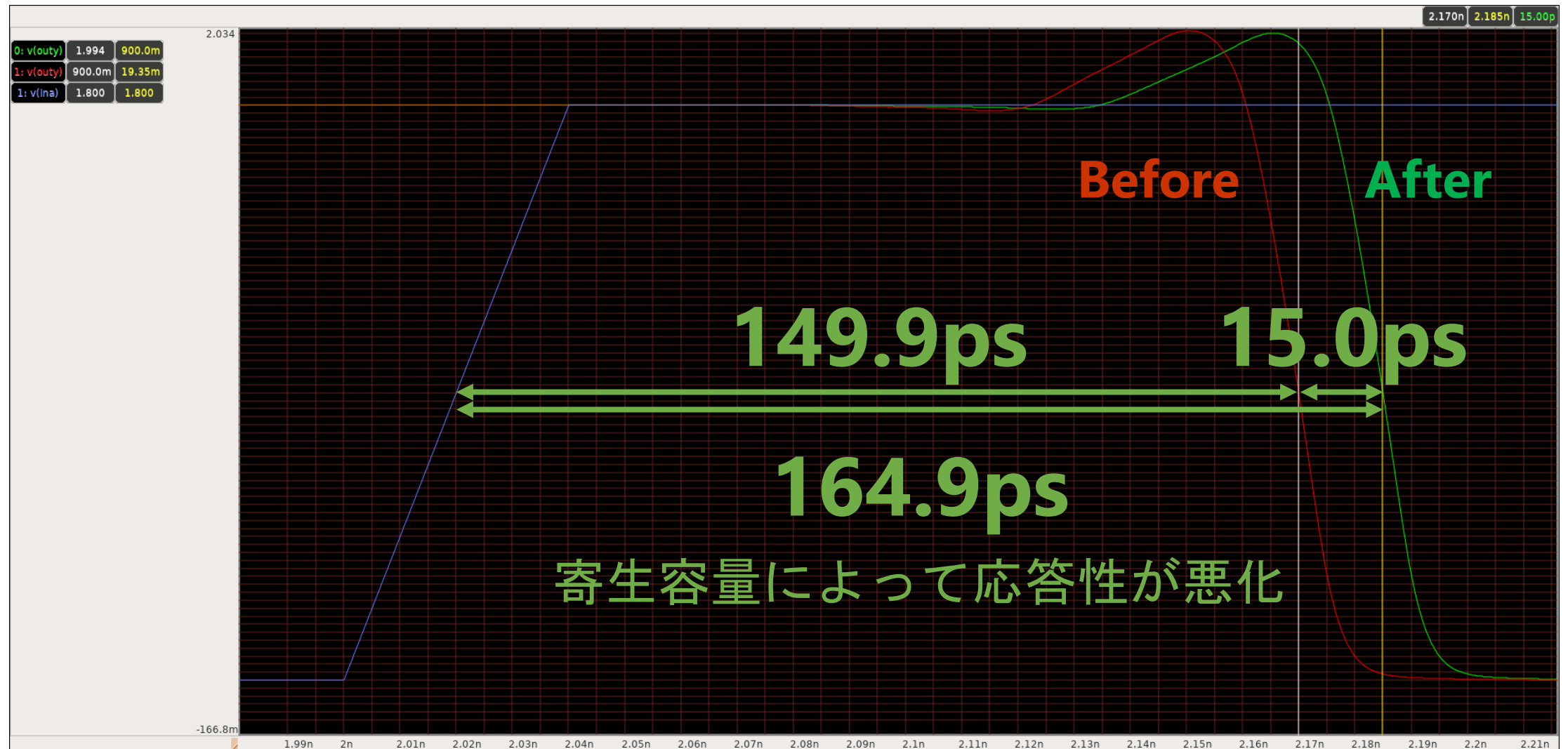
ngspice

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

MODELS



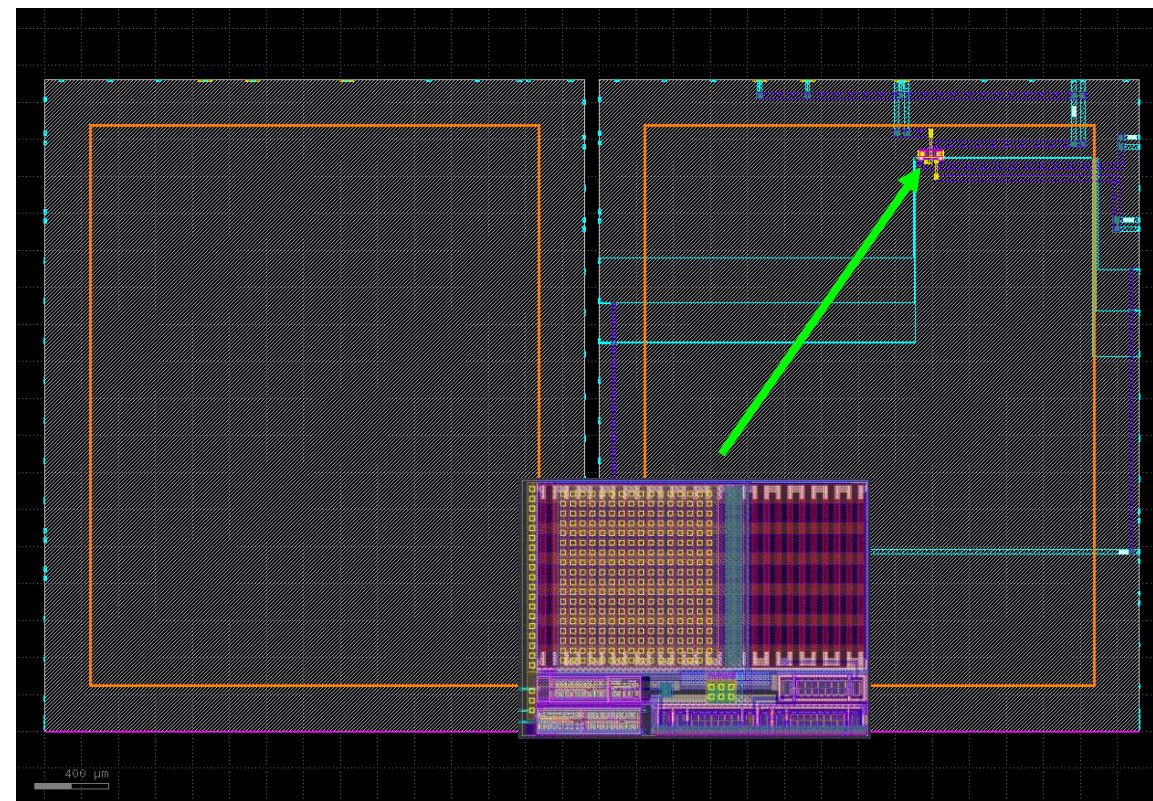
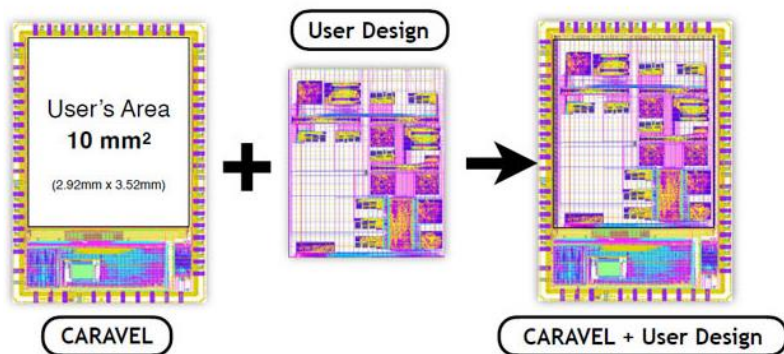
ポストレイアウトシミュレーションをする



全て完成したら指定のフレームに回路を載せる

例: Caravel harness (SKY130)

- 2.92 mm × 3.52mm user's area
- 38 IO ports
- 4 Power ports
- 128 Logic analyzer probes



そもそもLSI回路設計に必要なものは？

• プロセスデザインキット PDK

- シミュレーション用モデルライブラリ (SPICE)
- 検証ツール用ルールファイル (DRC, LVSなど)
- スタンダードセルライブラリ
 - レイアウト (GDS)
 - ネットリスト(SPICE)
 - 自動配線ルール(LEF)
 - Verilogシミュレーションライブラリ (LIB, V)
- 一部指定されたレイアウト (BJT)

• PDKで指定された各種ツール (EDAツール)

- 回路図エディタ or Verilogコンパイラ
 - 回路図エディタ : xschem
- レイアウトエディタ or 自動レイアウトツール
 - レイアウトエディタ : klayout, magic
- SPICEシミュレータ or Verilogシミュレータ
 - SPICEシミュレータ : ngspice, xyce /zīs/
- 検証ツール
 - 検証ツール : klayout, magic, netgen

ローカルでのセットアップ方法

- 各種ツールのドキュメントを読んでください
- 3zki/wsl_gf180mcu の中にセットアップスクリプトの例があるので参考にしてください
- https://github.com/3zki/wsl_gf180mcu
そのまま実行すると自分用に改造したPDKが導入されるので要注意

便利ツール

- efabless / volare
<https://github.com/efabless/volare>

コンパイル済のPDKを導入できます

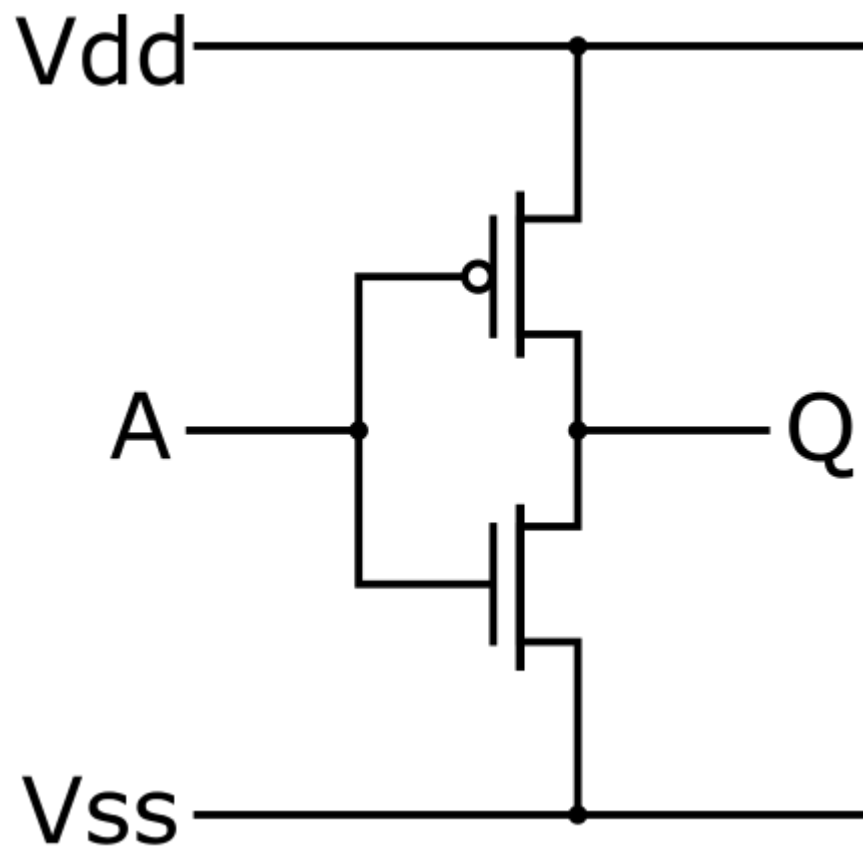
- d-m-bailey / extra_be_checks
https://github.com/d-m-bailey/extra_be_checks

LVS, ERC チェック

アナログLSI設計デモンストレーション

CMOSインバータ作成

CMOSインバータ



入力A	出力Q
L	H
H	L



入力A	出力Q
Vss	Vdd
Vdd	Vss

アナログLSIの設計フロー

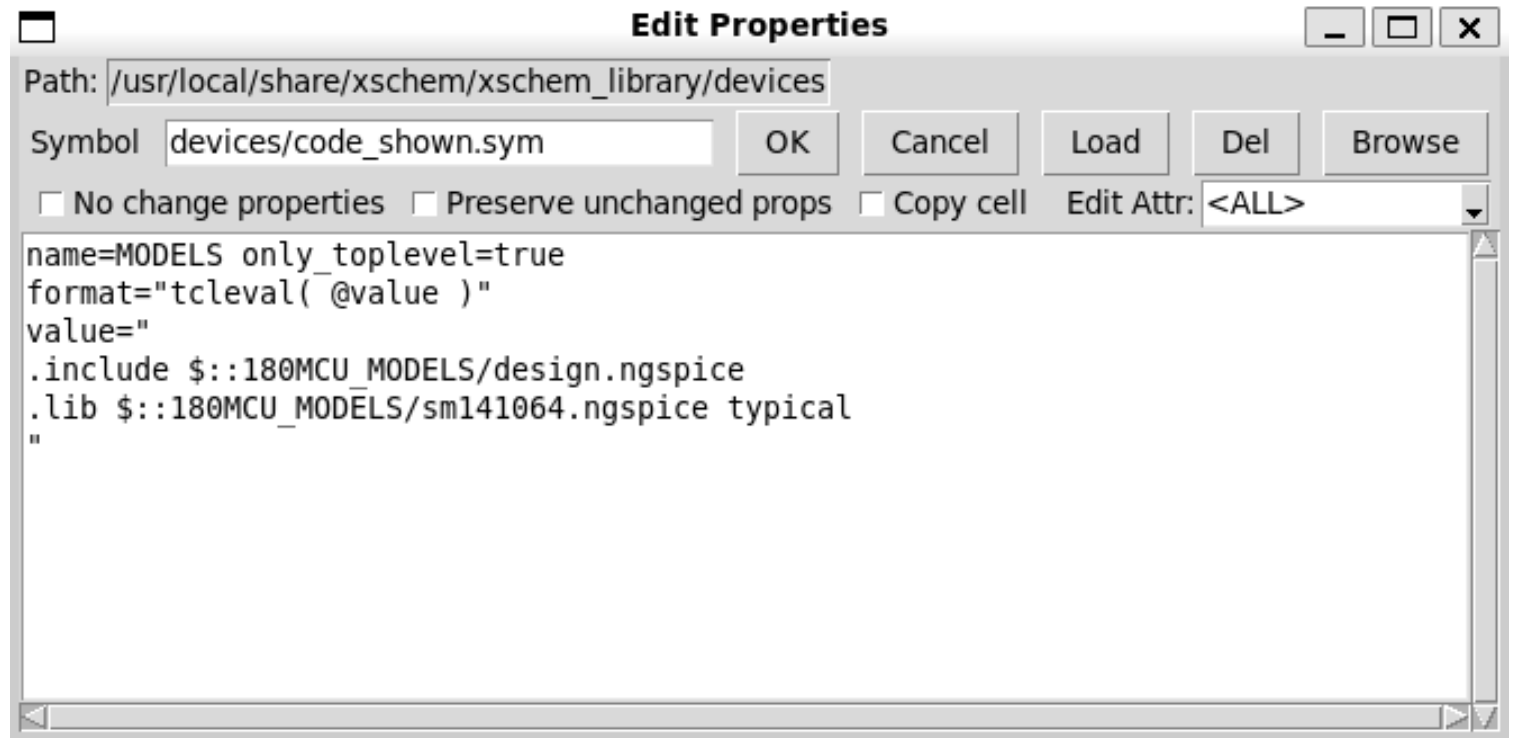
1. 回路図（ベンチマーク）を描く
2. シミュレーションをする
3. 回路図を基にレイアウトを描く
4. レイアウトを検証する
5. レイアウトを基に寄生成分を考慮したシミュレーションをする
6. （フレームに載せる）

論理検証

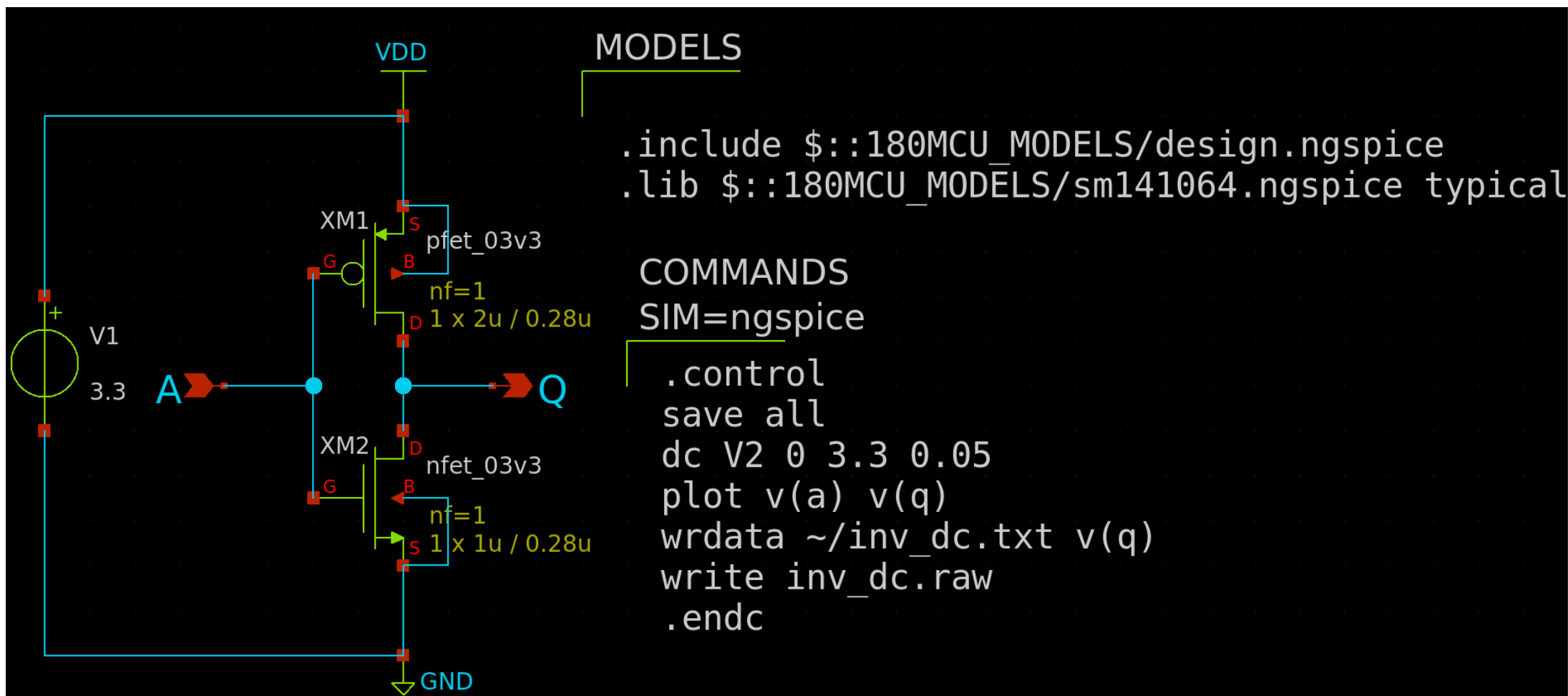
- インバータでは入力に対して反転した出力が確認できれば良い
 - 入力 0 V \rightarrow 出力 $V_{dd}\text{ V}$, 入力 $V_{dd}\text{ V}$ \rightarrow 出力 0 V
 - DC解析でCMOSインバータの動作を試みる
- DC解析 電圧を変動させたときの定常応答
- tran解析 時間を変動させたときの過渡応答
- AC解析 周波数を変動させたときの定常応答

ベンチマーク作成上の注意

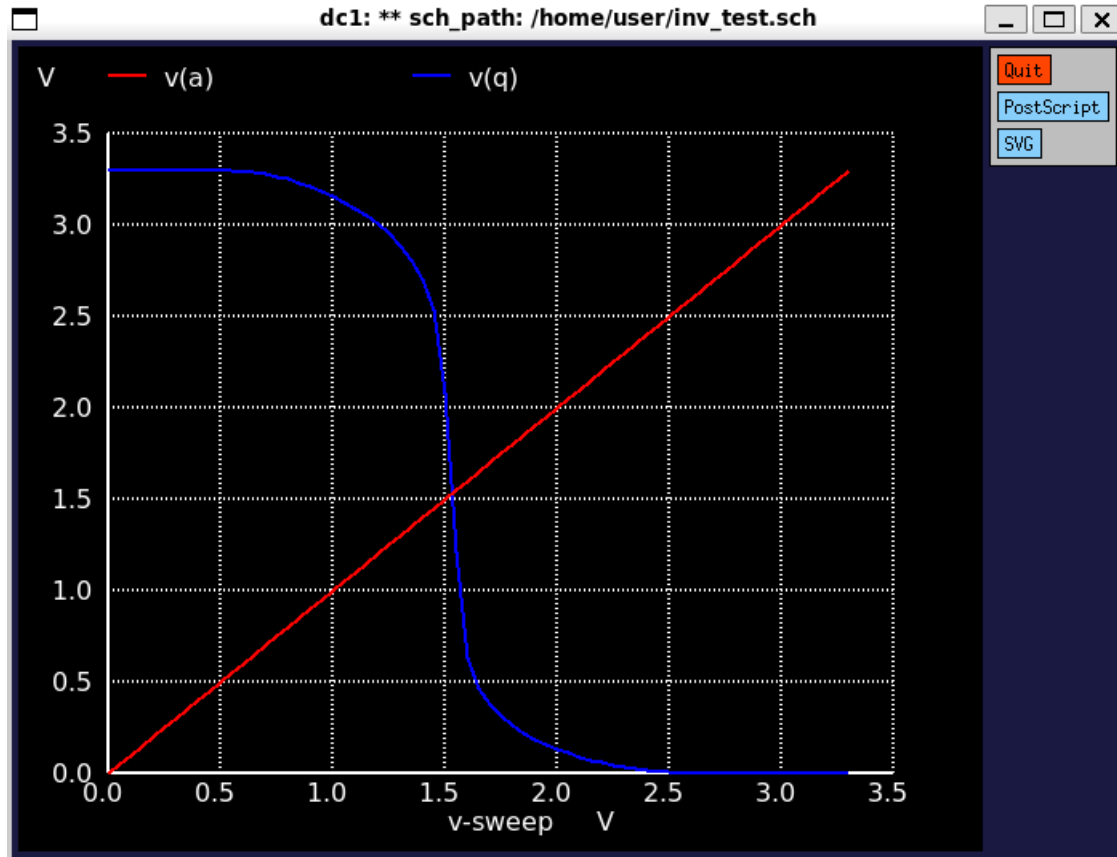
- ngspiceのシミュレーションコマンド → simulator_commands_shown.sym
- シミュレーションモデル → code_shown.sym
プロパティで format="tcleval(@value)" を追加する



ベンチマーク例



シミュレーション結果



入力A = 0 V → 出力Q = 3.3 V

入力A = 3.3 V → 出力Q = 0 V

- インバータとして機能している

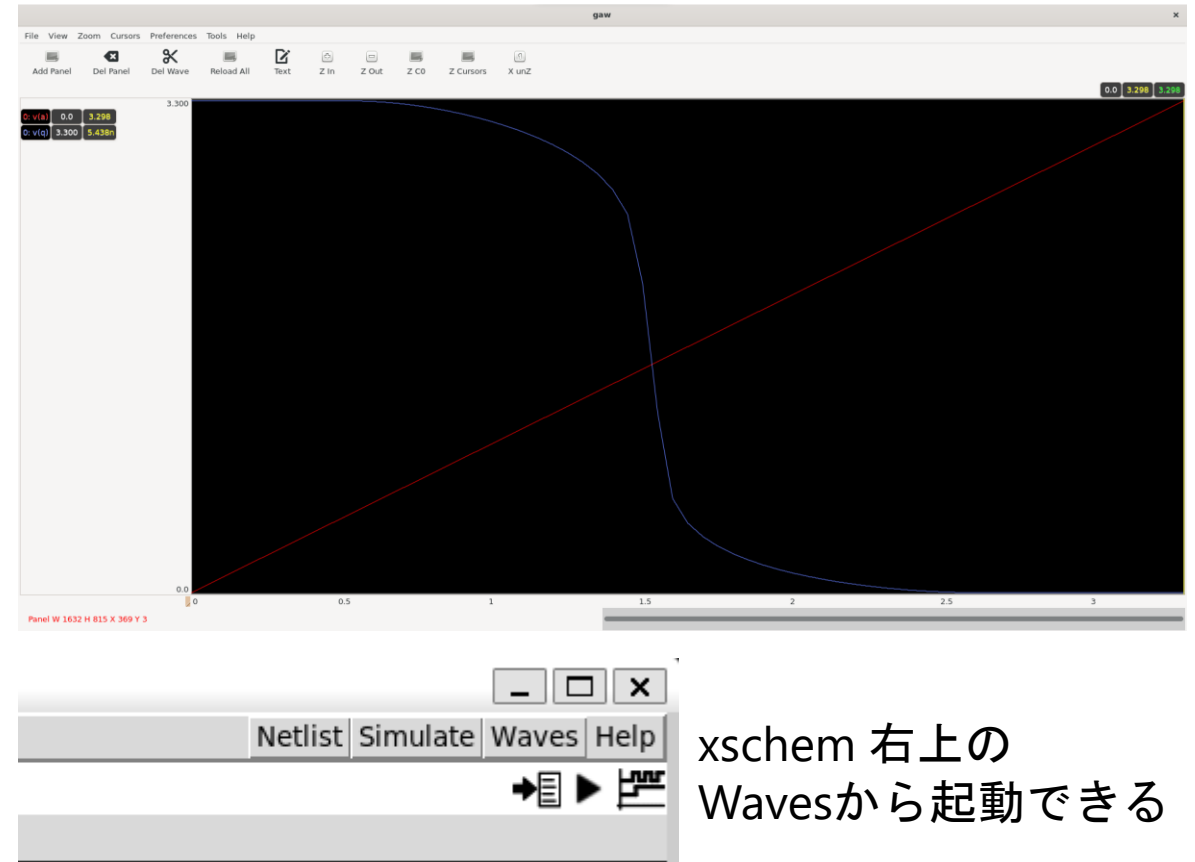
wrdata と gawによる波形表示

wrdataを使用するとテキスト出力ができる

0.00000000e+00	3.29999999e+00
5.00000000e-02	3.29999999e+00
1.00000000e-01	3.29999997e+00
1.50000000e-01	3.29999990e+00
2.00000000e-01	3.29999960e+00
2.50000000e-01	3.29999842e+00
3.00000000e-01	3.29999375e+00
3.50000000e-01	3.29997564e+00
4.00000000e-01	3.29990878e+00
4.50000000e-01	3.29968137e+00
5.00000000e-01	3.29900073e+00
5.50000000e-01	3.29728006e+00

▪
▪
▪

writeでrawファイルを出力すると gaw で波形表示ができる

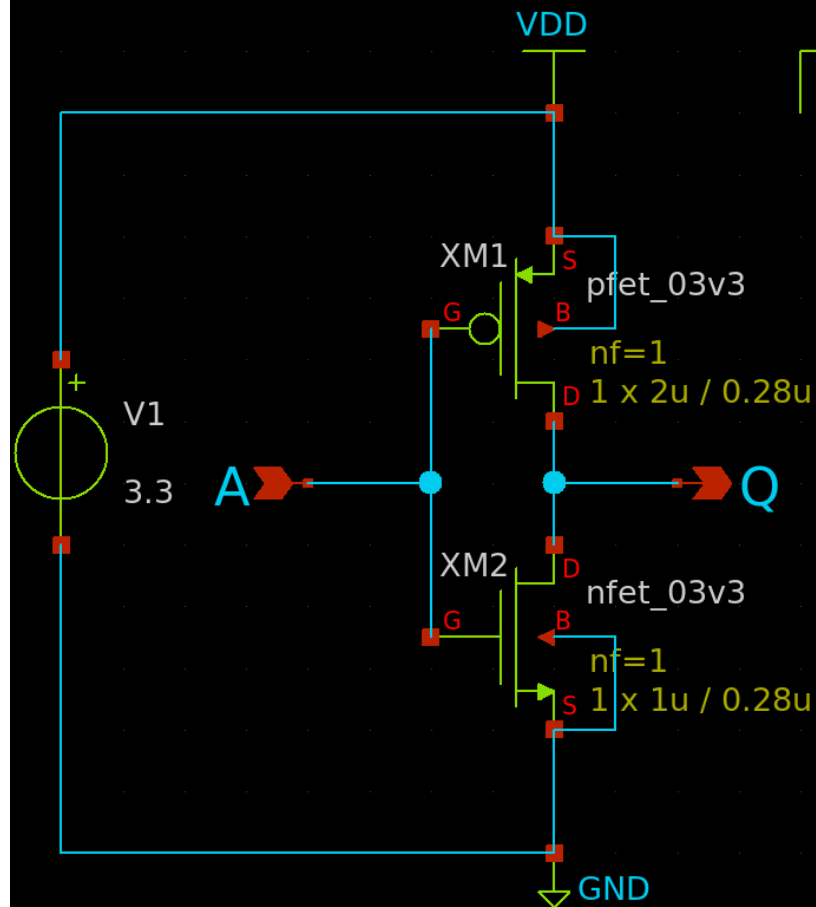


xschem 右上の
Wavesから起動できる

過渡応答、遅延時間

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

ベンチマーク例



MODELS

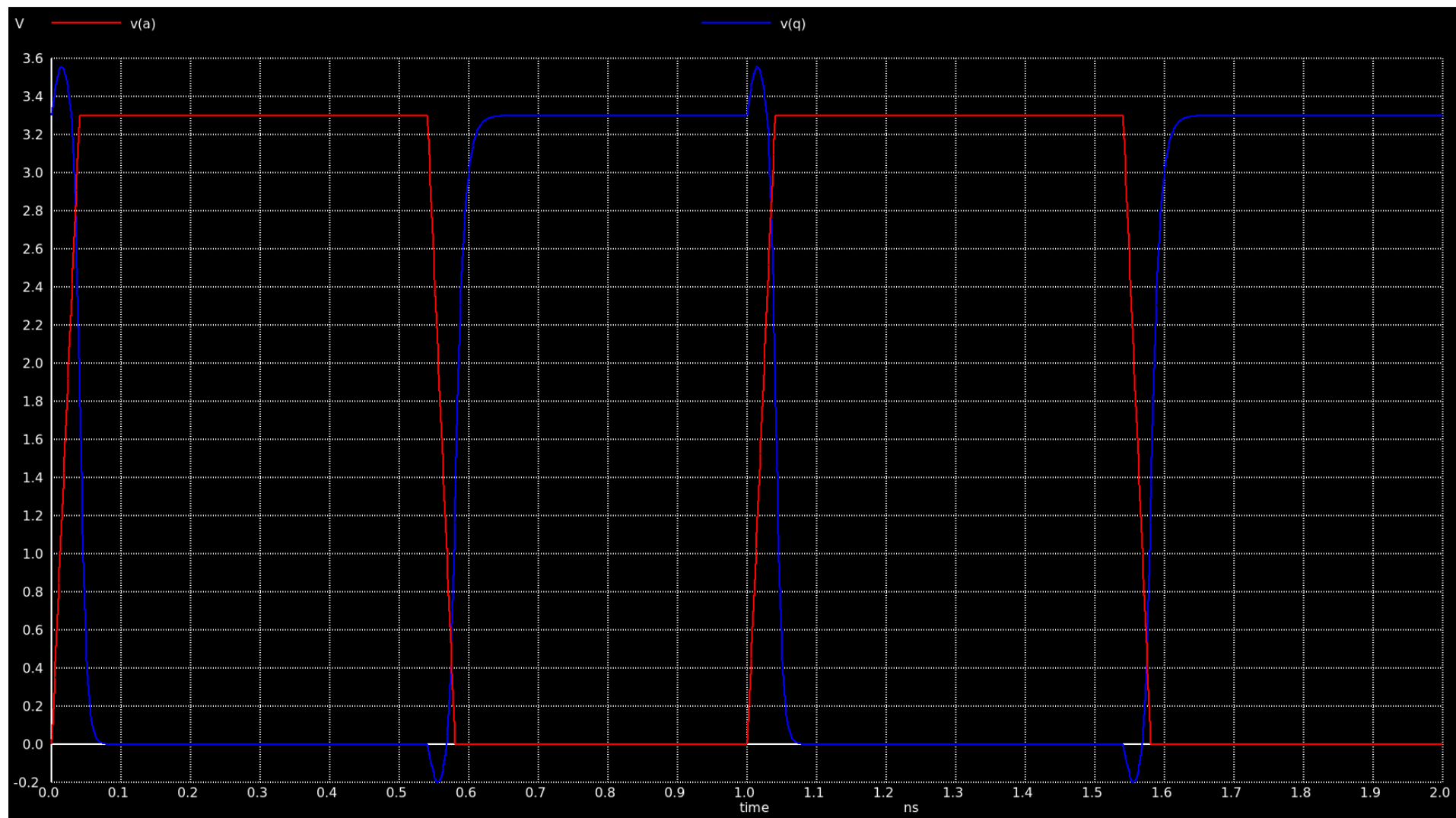
```
.include $::180MCU_MODELS/design.ngspice  
.lib $::180MCU_MODELS/sm141064.ngspice typical
```

COMMANDS

SIM=ngspice

```
VA A 0 pulse (0 3.3 0 40p 40p 0.5n 1n) dc 0  
.control  
save all  
tran 1p 2n  
plot v(a) v(q)  
wrdata ~/inv_tran.txt v(a) v(q)  
write inv_tran.raw  
.endc
```

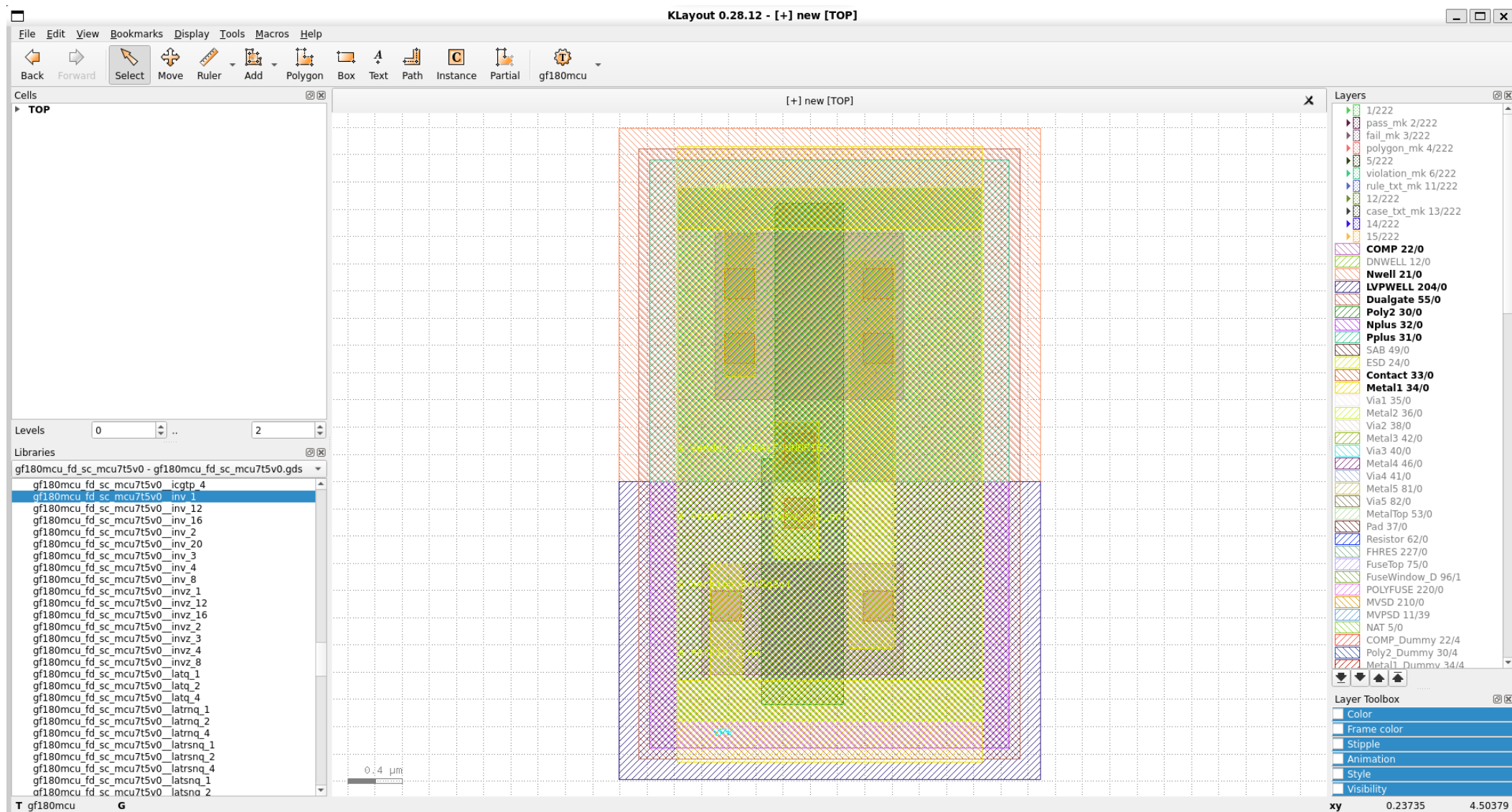
シミュレーション結果



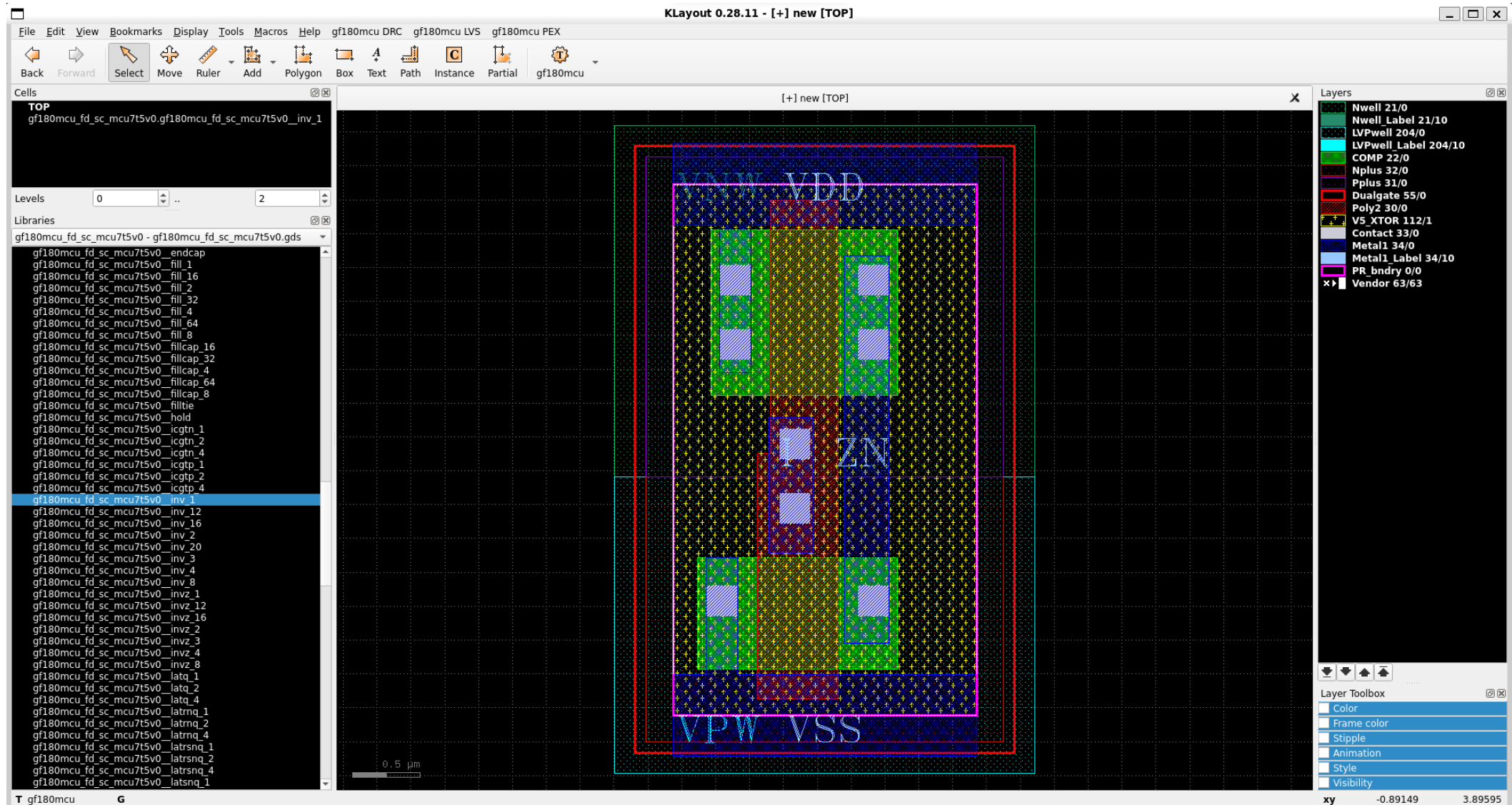
アナログLSIの設計フロー

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

公式PDKのレイアウト画面



3zki / gf180mcu のレイアウト画面

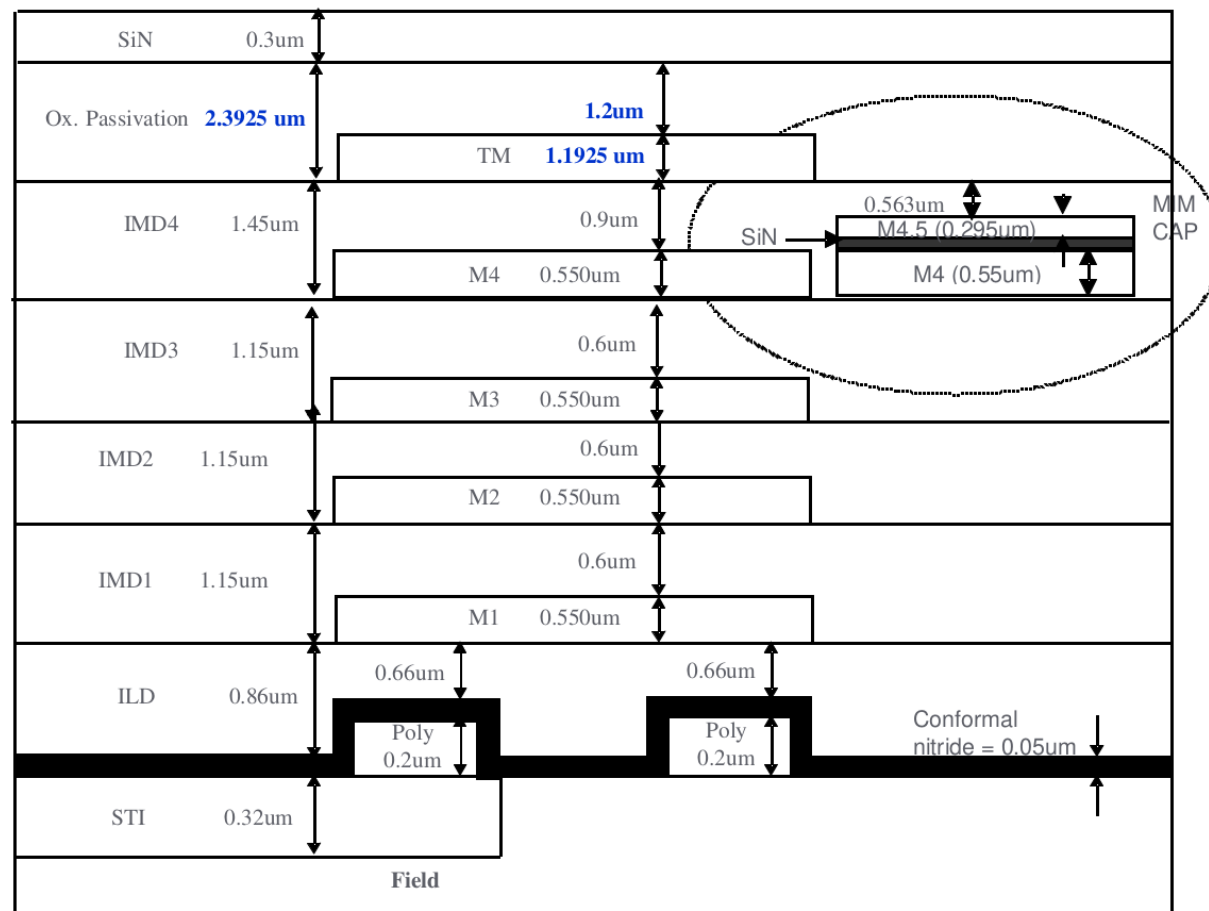


efablessが採用したオプションについて

GF-MPW-0 : gf180mcuC

GF-MPW-1 : **gf180mcuD** (TOPメタルの厚さが1.1umに変更)

- 5-metal stack, 1P5M, 5LM (metal_level=5LM)
- **1.1um thick top metal, TM 11KA (metal_top=11K)**
- MiM B option (mim_option=B)
- 2fF/um² MiM Capacitance (cap_mim_2f0fF)
メモ：キャパシタの耐圧は6Vらしい
- 1kΩ high sheet rho poly (ppolyf_u_1k)



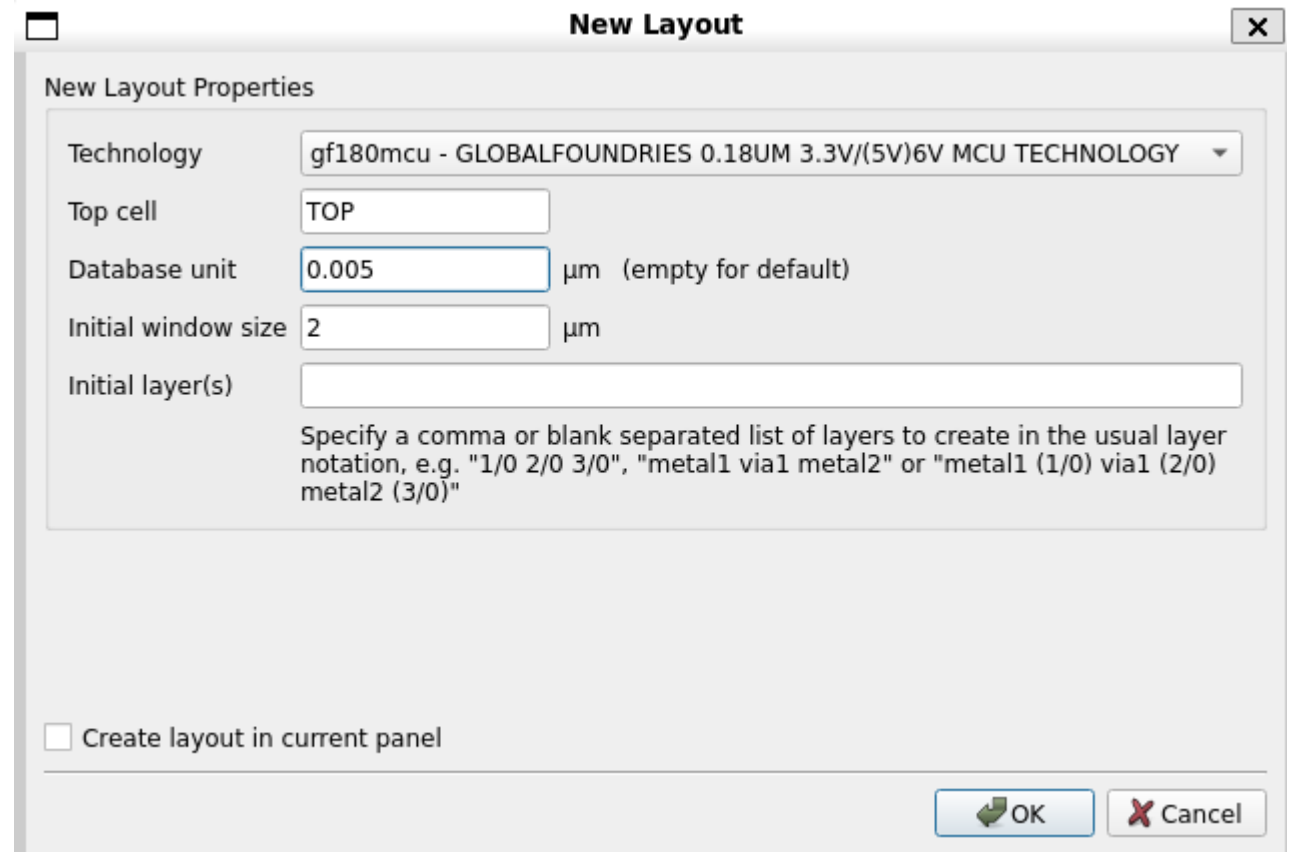
https://gf180mcu-pdk.readthedocs.io/en/latest/analog/layout/inter_specs/inter_specs_3_43.html

レイアウトを描く上での注意

- メタルは最大6層まで使えるが、今回は5層なのでVIA5, MetalTopは使用不可
- MIMキャパシタはMetal4とMetal5の間に形成する (Option=B, Level=M4-M5)
※MIMキャパシタはトップメタルの下に形成される
- 各検証の実行前にオプションを確認・設定する (特にmetal_top=11K)

レイアウトを描く上での注意

- Database unit は0.005um または 0.01um にしないとPcellがうまく動作しない



New Layout

New Layout Properties

Technology: gf180mcu - GLOBALFOUNDRIES 0.18UM 3.3V/(5V)6V MCU TECHNOLOGY

Top cell: TOP

Database unit: 0.005 μm (empty for default)

Initial window size: 2 μm

Initial layer(s):

Specify a comma or blank separated list of layers to create in the usual layer notation, e.g. "1/0 2/0 3/0", "metal1 via1 metal2" or "metal1 (1/0) via1 (2/0) metal2 (3/0)"

☐ Create layout in current panel

OK Cancel

アナログLSIの設計フロー

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

3zki / gf180mcu で利用できる レイアウト検証

Design Rules Check (DRC)

- 指定されたデザインルールから違反していないか検証

gf180mcu DRC gf180mcu LVS gf180n

Run KLayout DRC - Antenna

Run KLayout DRC - Density

Run KLayout DRC - Full DRC

→ Run KLayout DRC - Caravel DRC

Run Magic DRC

Set Options (Restart required!)

Layout Versus Schematic (LVS)

- レイアウトが回路図通りに描けているか検証

gf180mcu LVS gf180mcu PEX

→ Run Klayout LVS

Make netlist for Klayout LVS

Make netlist (MCU7T5V0) for Klayout LVS

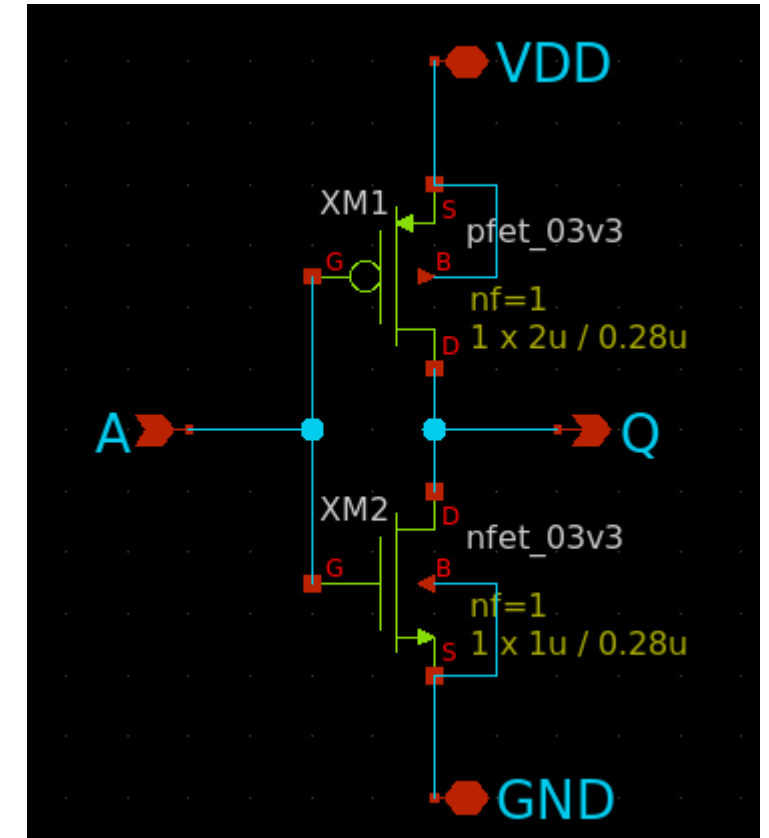
Run Magic LVS

Set Options (Restart required!)

Magic DRC, Magic LVS は未整備です
KLayout DRC, KLayout LVSを使用してください

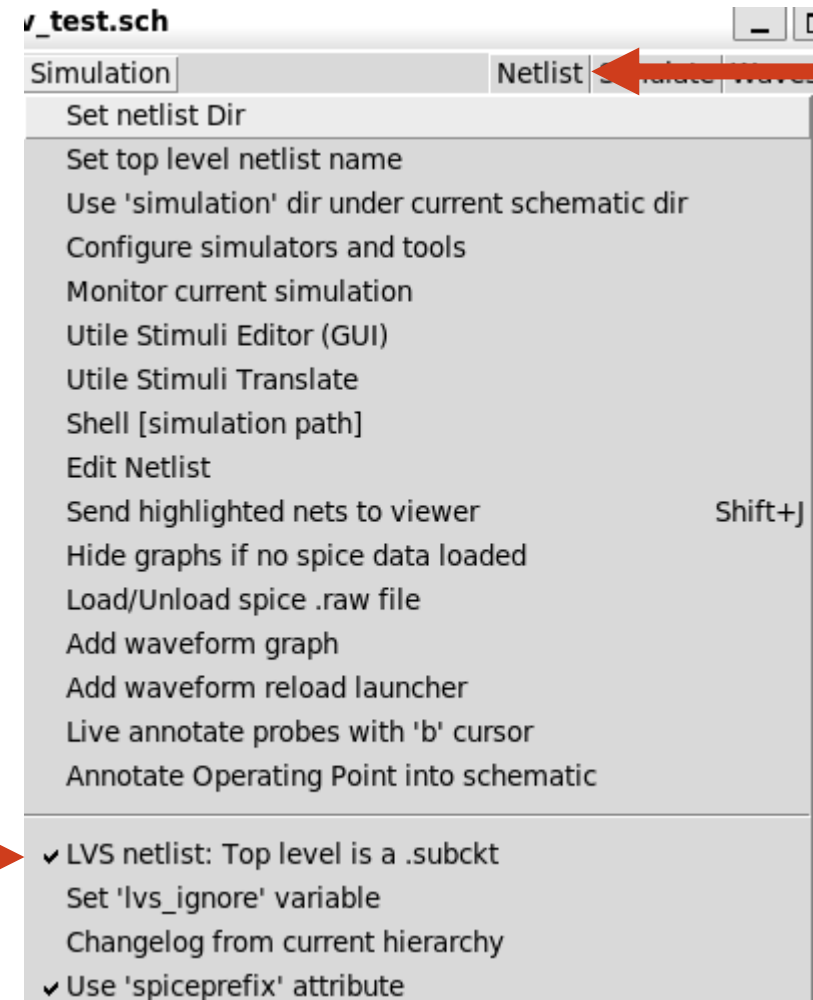
LVS用ネットリストの出力方法 (1/2)

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



LVS用ネットリストの出力方法 (2/2)

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



アナログLSIの設計フロー

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

レイアウト検証 : PEX, Parasitic Extraction

RC抽出

- 寄生抵抗と寄生容量を含むネットリストを抽出

```

Magic-PEX x
File Edit View Search Terminal Help
exttospace finished.
* NGSPICE file created from TOP.ext - technology: gf180mcuC

.subckt TOP A Q VDD GND
X0 Q A.t0 VDD.t1 VDD.t0 pfet_03v3 ad=1.3p pd=5.3u as=1.3p ps=5.3u w=2u l=0.28u
X1 Q A.t1 GND.t1 GND.t0 nfet_03v3 ad=0.61p pd=3.22u as=0.61p ps=3.22u w=1u l=0.28u
R0 A.n0 A.t1 60.6324
R1 A.n0 A.t0 50.9859
R2 A A.n0 4.00168
R3 VDD.n0 VDD.t0 909.662
R4 VDD.n0 VDD.t1 4.56542
R5 VDD VDD.n0 0.0465872
R6 Q Q.n1 6.1565
R7 Q Q.n0 4.60529
R8 GND.n0 GND.t0 2545.79
R9 GND.n0 GND.t1 6.08384
R10 GND GND.n0 0.0514016
C0 VDD Q 0.146f
C1 A Q 0.0875f
C2 A VDD 0.139f
.ends

```

C抽出

- 寄生容量のみを含むネットリストを抽出

```

Magic-PEX x
File Edit View Search Terminal Help
s fillblock lvstext obscomment
Scaled tech values by 10 / 1 to match internal grid scaling
Loading gf180mcuC Device Generator Menu ...
Loading "/home/user/.klayout/macros/gf180mcu_magic_pex.tcl" from command line.
Warning: Calma reading is not undoable! I hope that's OK.
Library written using GDS-II Release 6.0
Library name: LIB
Reading "TOP".
CIF file read warning: CIF style import: units rescaled by factor of 5 / 1
Extracting TOP into TOP.ext:
exttosim finished.
exttospace finished.
exttospace finished.
* NGSPICE file created from TOP.ext - technology: gf180mcuC

.subckt TOP A Q VDD GND
X0 Q A VDD VDD pfet_03v3 ad=1.3p pd=5.3u as=1.3p ps=5.3u w=2u l=0.28u
X1 Q A GND GND nfet_03v3 ad=0.61p pd=3.22u as=0.61p ps=3.22u w=1u l=0.28u
C0 A Q 0.0875f
C1 VDD Q 0.146f
C2 A VDD 0.139f
.ends

```

PEX前の下準備：Flatten

1. コピーを保存する

- ファイル名で大文字を使うとエラーが出てしまう（不具合）
 - 拡張子がデフォルトだと.GDSなので.gdsに直す

2. ドラッグでレイアウト全体を選択する

3. Edit > Flatten Cell > Flatten Instances を All hierarchy levels で実行

4. Cells が TOP のみになっていることを確認して Save

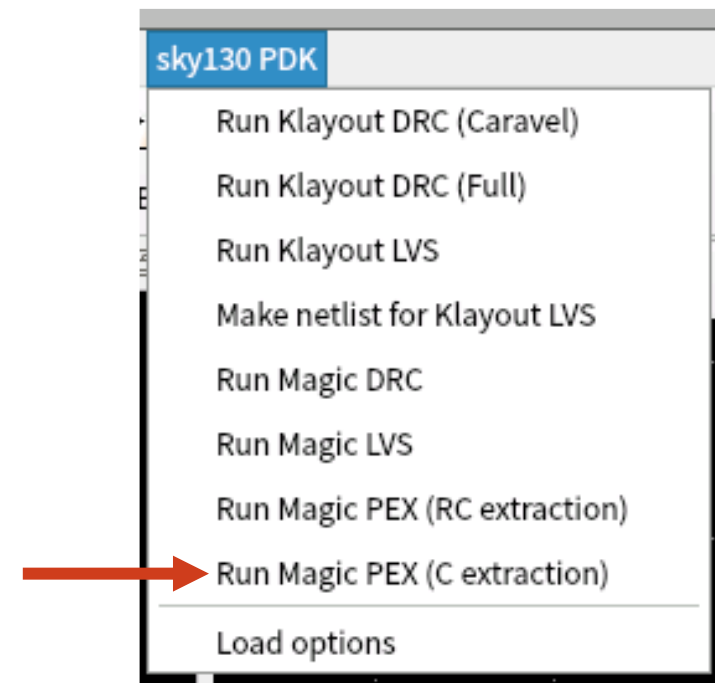
寄生抽出

Magic PEX を用いて寄生容量抽出を行う

- Run Magic PEX (C extraction) を実行

実行すると寄生容量付きネットリストが生成される

- ファイル名：{レイアウト名}_pex_extracted.spice
 - デフォルトではTOP_pex_extracted.spice



寄生容量を考慮したベンチマークに変更する

ポストシミュレーションの際は以下 2 点の変更が必要

1. 外部ネットリストへのパス
2. 外部ネットリスト用のシンボル

外部ネットリストへのパスを通す


シミュレーションスクリプトにincludeでパスを通す

- レイアウトデータと同じ階層に [レイアウト名]_pex_extracted.spice が生成される

```
COMMANDS
```

```
SIM=ngspice
```

```
VA A 0 pulse (0 3.3 0 40p 40p 0.5n 1n) dc 0
```



```
.include ~/TOP_pex_extracted.spice
```

```
.control
```

```
save all
```

```
tran 1p 2n
```

```
plot v(a) v(q) v(q2)
```

```
wrdata ~/inv_tran.txt v(a) v(q) v(q2)
```

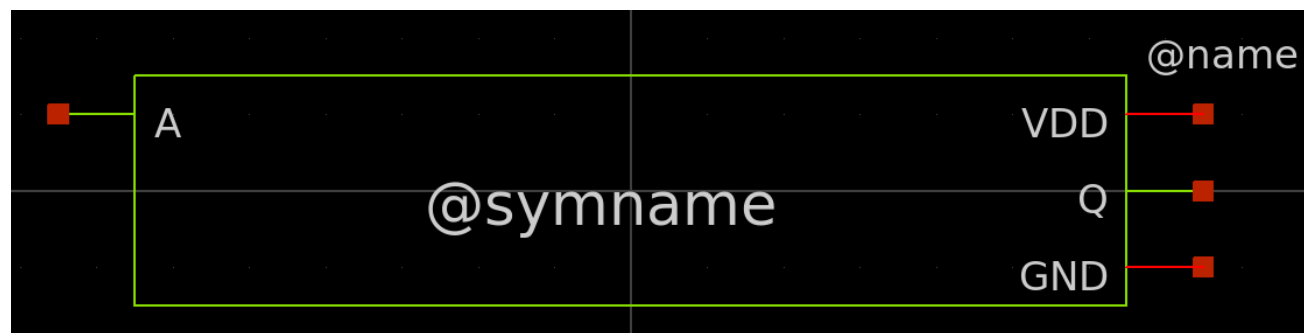
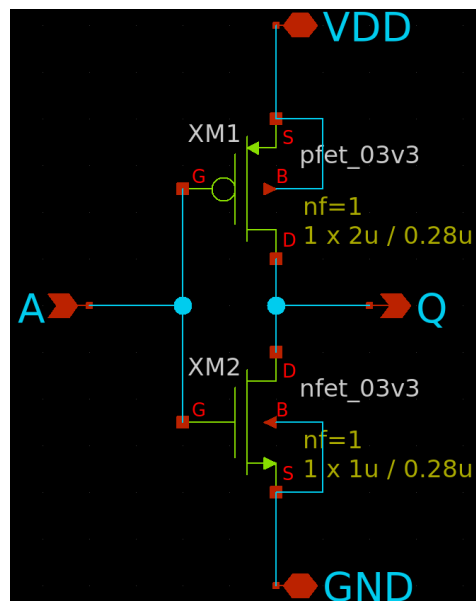
```
write inv_tran.raw
```

```
.endc
```

外部ネットリスト用のシンボル作成

LVSで使用した回路図からシンボルを作成する

- メニュー Symbol > Make symbol from schematic



このままだとシンボルはLVSで使用した回路図を参照するため
ネットリストを参照するように改変する

外部ネットリスト用のシンボル作成（基本形）

シンボルのプロパティを開いてこんな感じの中身にする（Qキーでプロパティを開く）

type=primitive

format="@name [ネットリスト内のピン] @prefix"

template="name=x1 prefix=[subcktの名前]"

extra="prefix"

highlight=true

詳しくは http://repo.hu/projects/xschem/xschem_man/symbol_property_syntax.html を参照

外部ネットリスト用のシンボル作成例 (インバータ)

type=primitive

format="@name @@A @@Y @@VDD @@GND @prefix"

template="name=x1 prefix=TOP"

extra="prefix"

highlight=true

抽出したネットリストのピンと同じ並びで記述する

```
.subckt TOP A Y VDD GND
X0 Y A VDD VDD sky130_fd_pr__pfe
X1 Y A GND GND sky130_fd_pr__nfe
C0 Y A 0.05fF
C1 Y VDD 0.17fF
C2 A VDD 0.18fF
.ends
```

外部ネットリスト用のシンボル作成例 (インバータ)

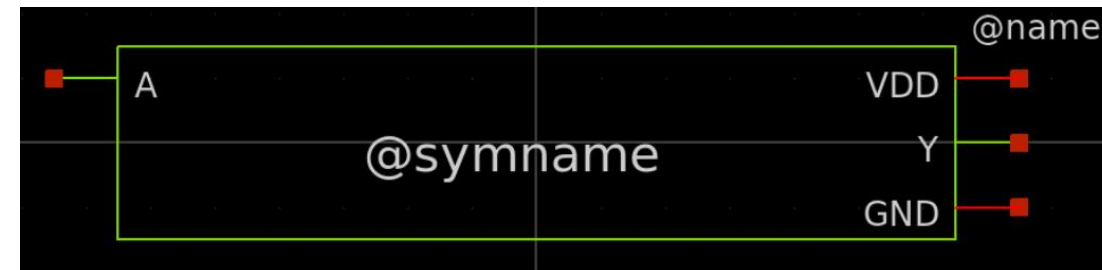
type=primitive

format="@name @@A @@Y @@VDD @@GND @prefix"

template="name=x1 prefix=TOP"

extra="prefix"

highlight=true



シンボルから出ているピンには"@@"のprefixをつける
これで一応完成

ベンチマーク例

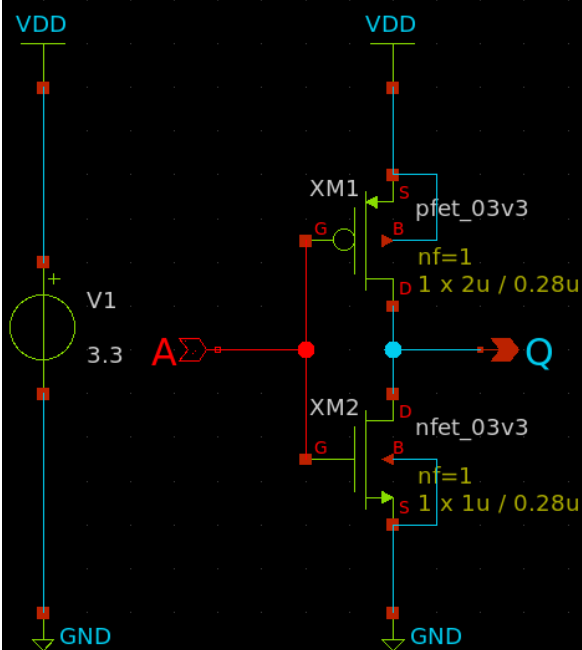
MODELS

```
.include $::180MCU_MODELS/design.ngspice  
.lib $::180MCU_MODELS/sm141064.ngspice typical
```

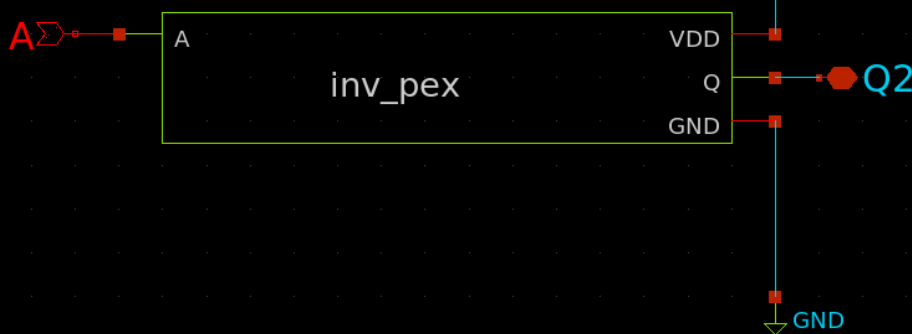
COMMANDS

```
SIM=ngspice
```

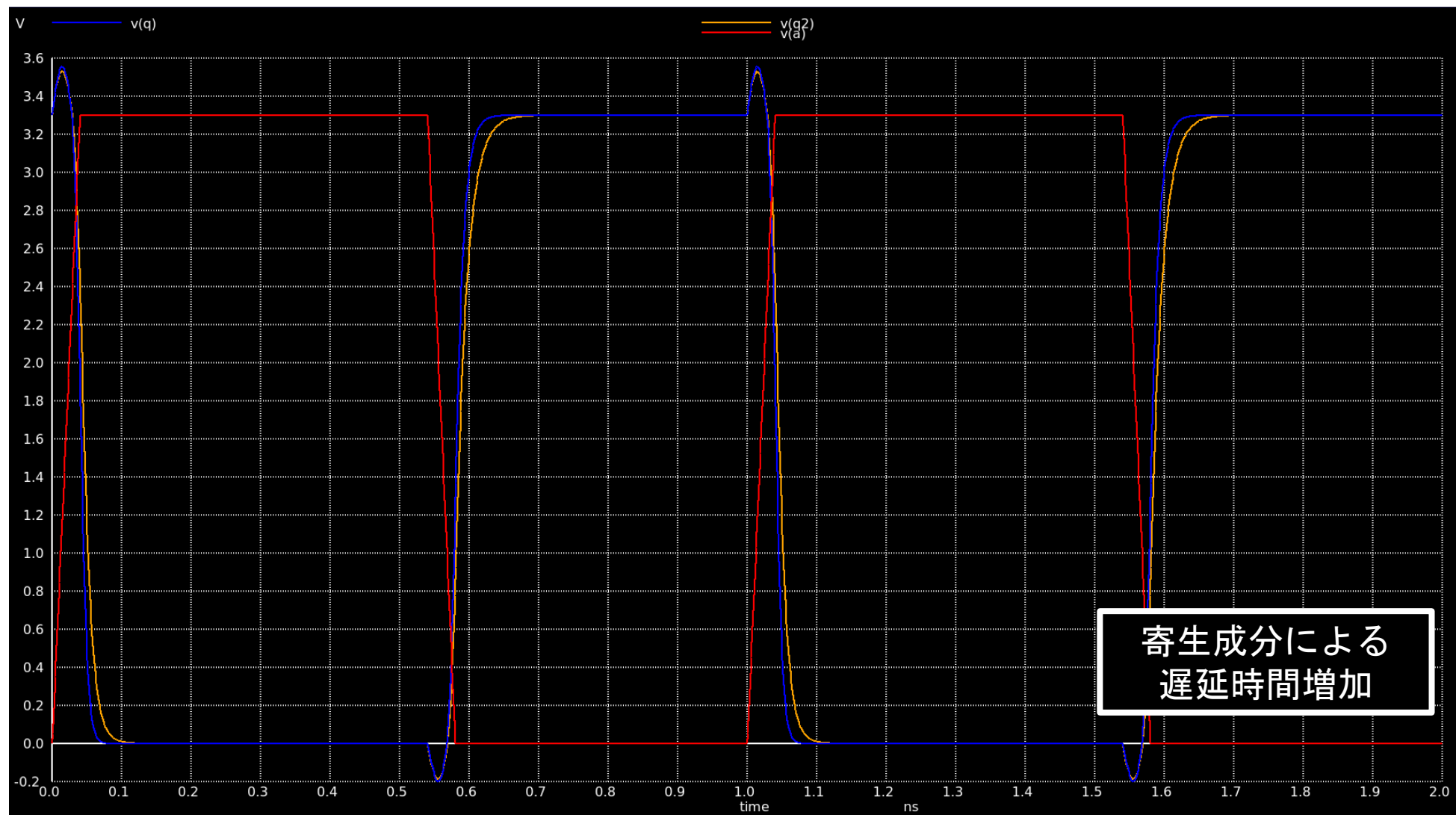
```
VA A 0 pulse (0 3.3 0 40p 40p 0.5n 1n) dc 0  
.include ~/TOP_pex_extracted.spice  
.control  
save all  
tran 1p 2n  
plot v(a) v(q) v(q2)  
wdata ~/inv_tran.txt v(a) v(q) v(q2)  
write inv_tran.raw  
.endc
```



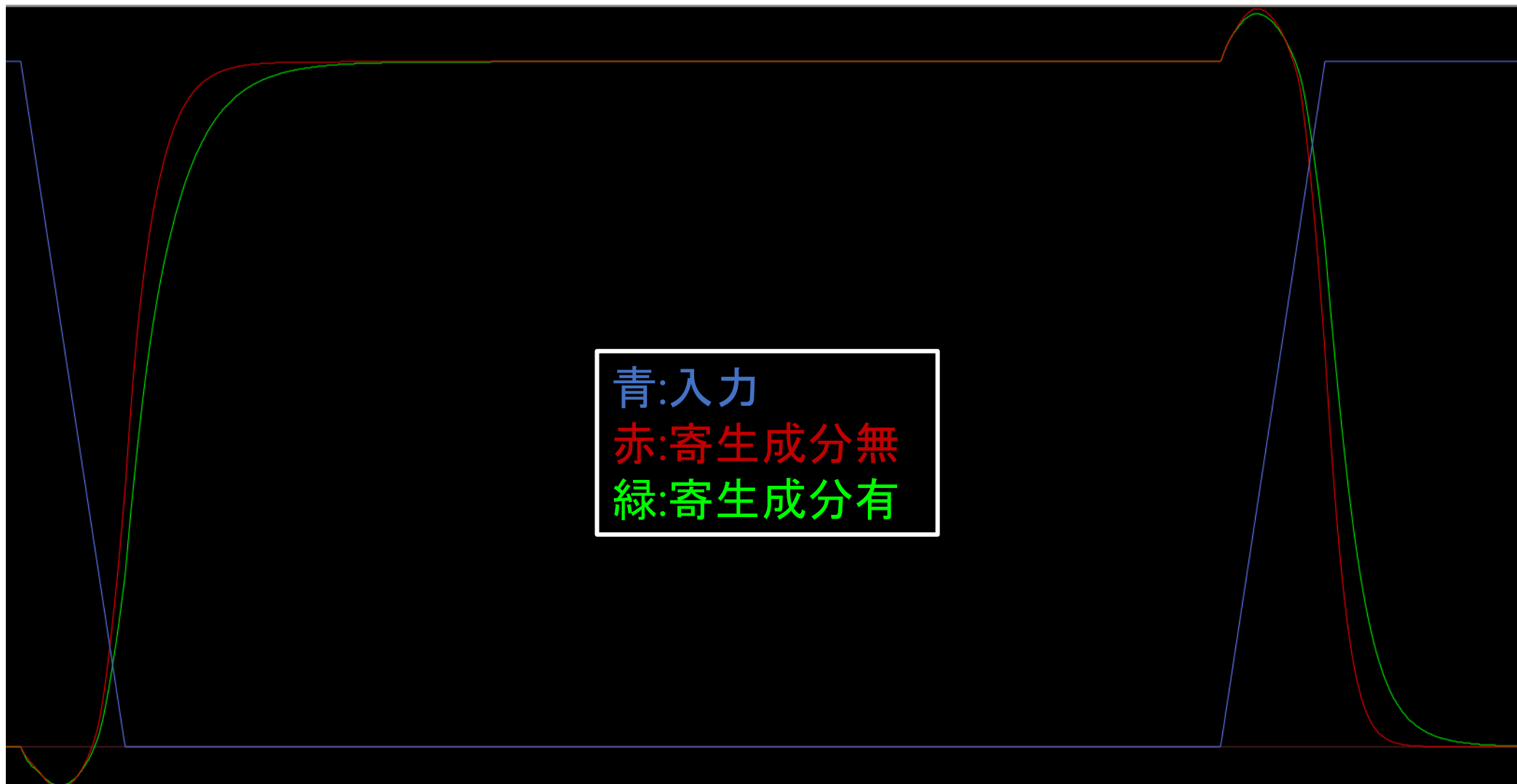
寄生成分の有無による
過渡応答の変化を見る



シミュレーション結果



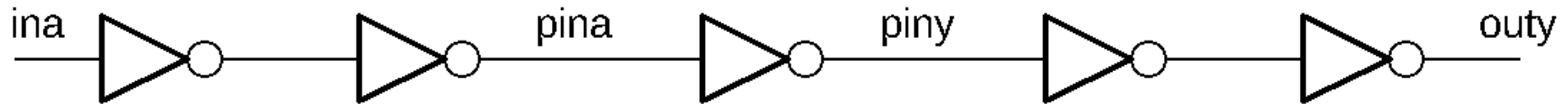
シミュレーション結果



参考：遅延時間シミュレーションの条件例

$P=2.5\mu$ / $N=1\mu$

$P=2.5\mu$ / $N=1\mu$



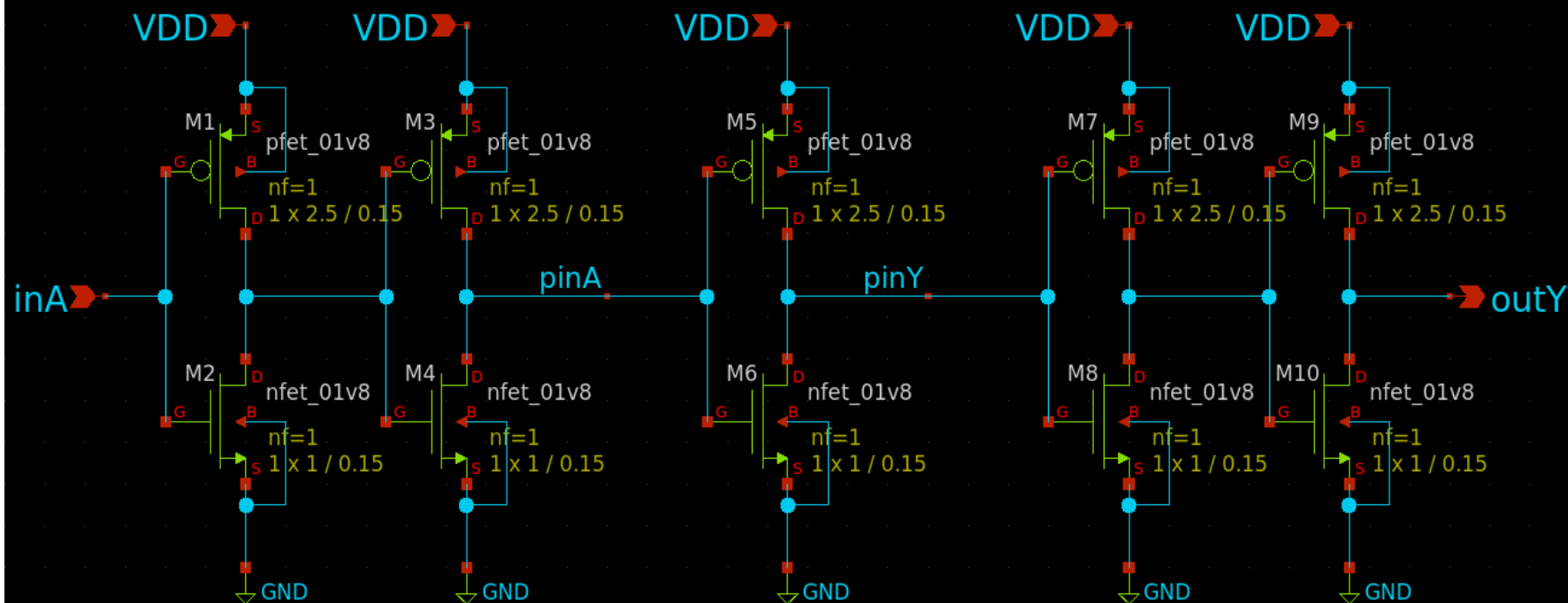
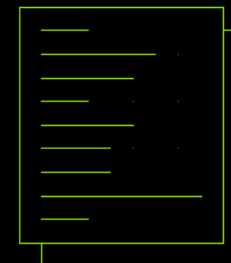
- 初段バッファサイズは $w_p = 2.5 \mu\text{m}$, $w_n = 1 \mu\text{m}$
- 出力段の負荷容量は入力容量と同じ $w_p = 2.5 \mu\text{m}$, $w_n = 1 \mu\text{m}$
- ina から outy までの遅延時間を求める

理想遅延時間を計測するベンチマークの例

ngspice

```
VA inA 0 pulse(0 1.8 0 40p 40p 1n 2n) dc 0
VD VDD 0 dc 1.8
.control
tran 1p 4n
wdata ~/inv_bench.txt v(inA) v(outY)
write ~/inv_test_pex.raw
.endc
```

MODELS



外部ネットリスト用のシンボル作成（応用形）

シンボルのプロパティを開いてこんな感じの中身にする（Qキーでプロパティを開く）

type=primitive

format="@name [ネットリスト内のピン] @prefix"

template="name=x1 [プロパティでネットを指定するピン] prefix=TOP"

extra=" [プロパティでネットを指定するピン] prefix"

highlight=true

詳しくは http://repo.hu/projects/xschem/xschem_man/symbol_property_syntax.html を参照

外部ネットリスト用のシンボル作成（応用例）

VDDとGNDのピンをシンボルから削除してプロパティで記述してみる

type=primitive

format="@name @@A @@Y @VDD @GND @prefix"

template="name=x1 VDD=VDD GND=GND prefix=TOP"

extra="VDD GND prefix"

highlight=true

プロパティでピンのネットを指定する場合は"@"のprefixを付けて、
extraにピン、templateにデフォルトの値を追加する

外部ネットリスト用のシンボル作成（応用例）

VDDとGNDのピンをシンボルから削除してプロパティで記述してみる

type=primitive

format="@name @@A @@Y @VDD @GND @prefix"

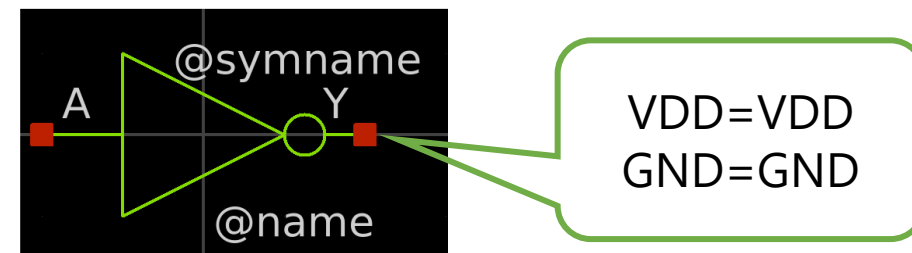
template="name=x1 VDD=VDD GND=GND prefix=TOP"

extra="VDD GND prefix"

highlight=true



ベンチマークが
見やすくなる

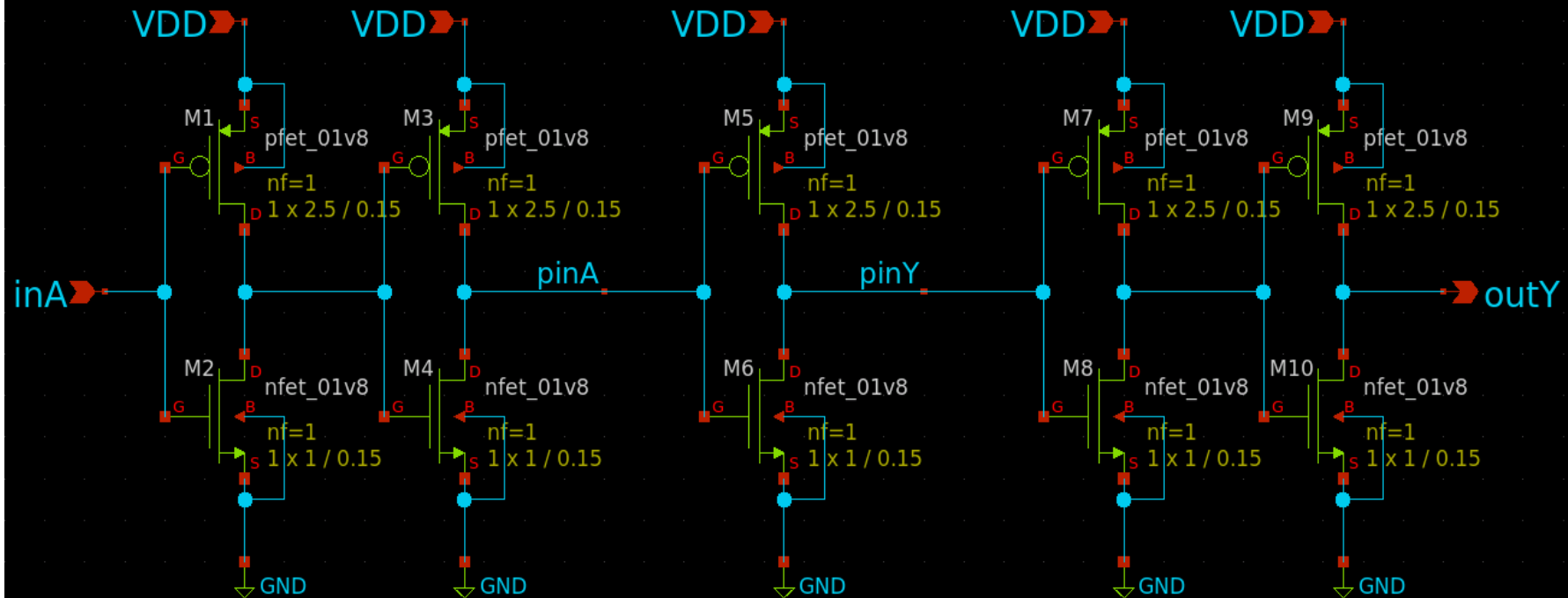


ベンチマーク改造 Before

ngspice

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

MODELS

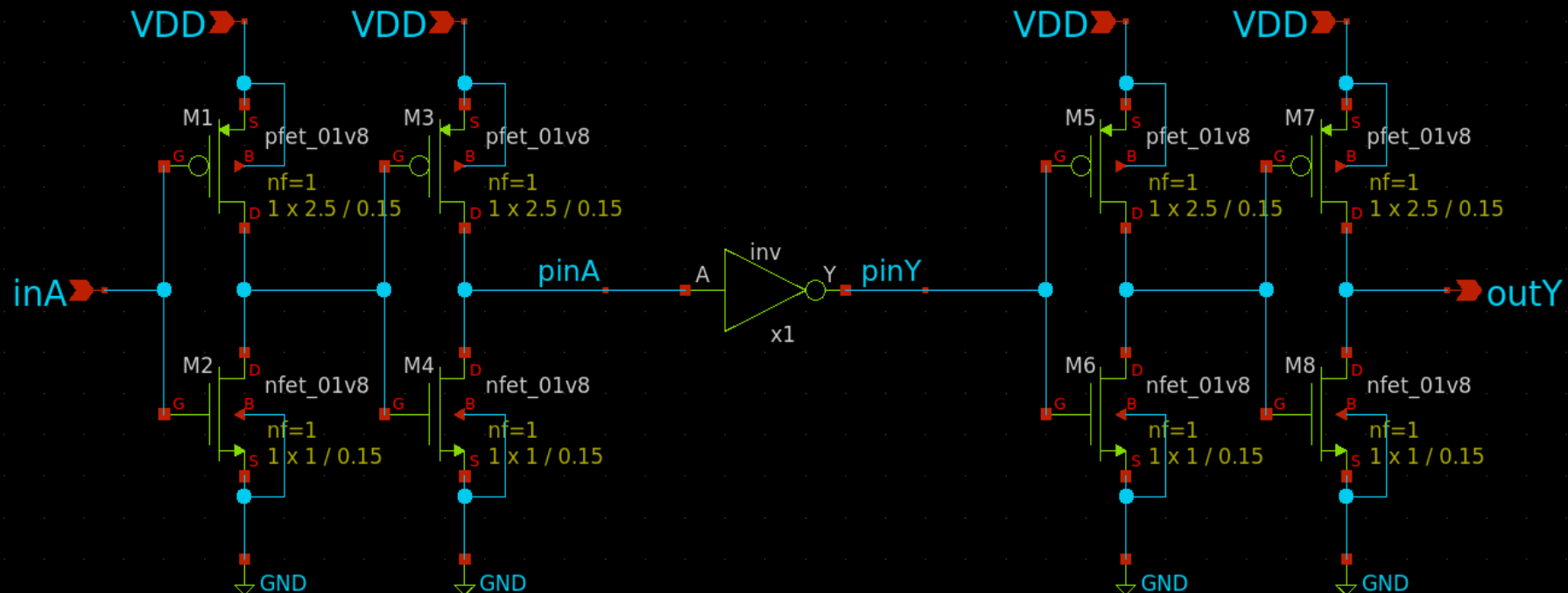


ベンチマーク改造 After

ngspice

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

MODELS



ポストレイアウトシミュレーションをする

