完成済テストベンチの配布

- https://github.com/3zki/lsi1_tutorial

- git clone コマンドで各自ダウンロードすること

```
[ub123456@remote01 ~]$ git clone https://github.com/3zki/lsi1_tutorial
```

演習

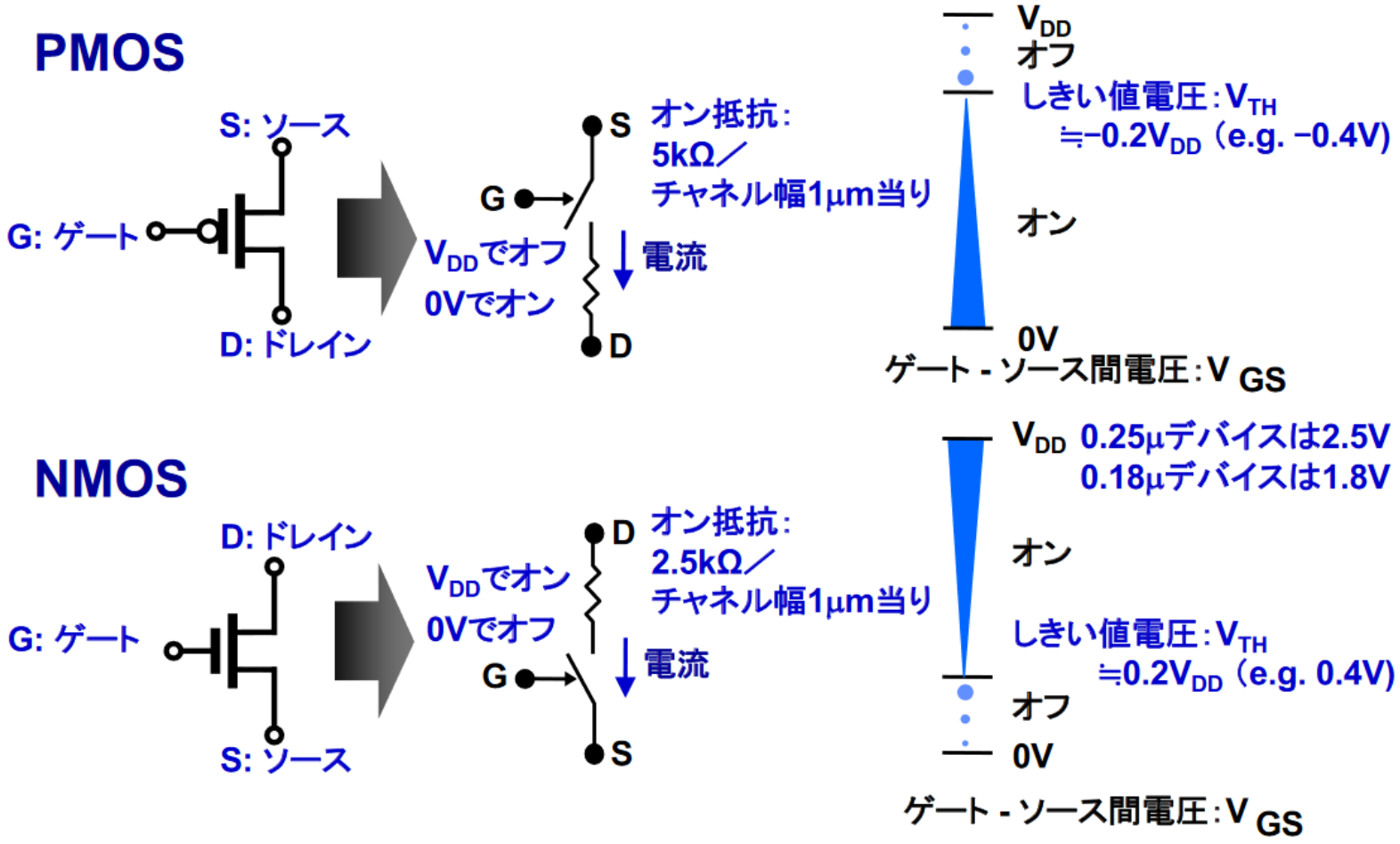
SKY130で学ぶLSI回路設計

naklab

オン抵抗測定

講義資料より

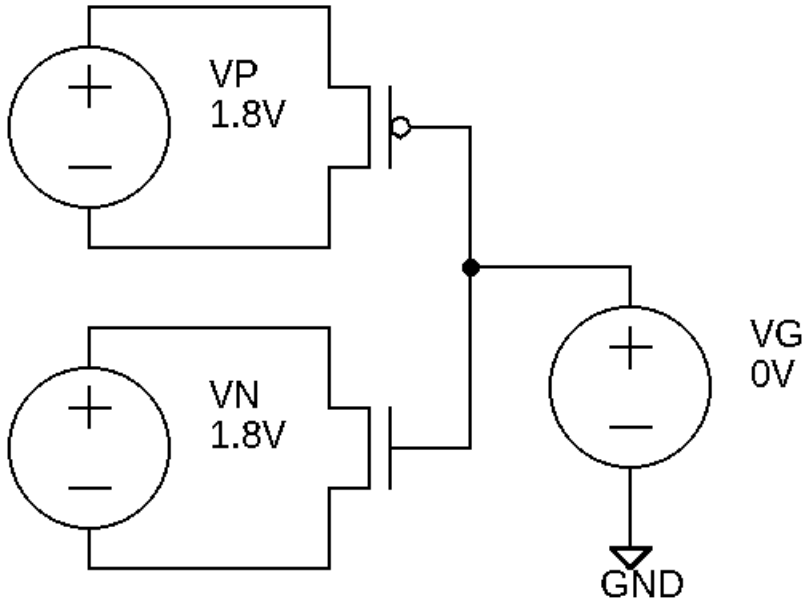
- 現実のオン抵抗はきれいに2:1になってない



オン抵抗測定

- **PMOS/NMOSのオン抵抗を測定する (L=0.15µm, W=1.0µm)**

- ドレインーソース間に電圧源vp (V=1.8), vn (V=1.8)を置く
- ゲートに電圧源vg (V=0)を置く
- DC解析でvgの電圧を0V→1.8V間で掃引
- 電圧源vp, vnの消費電流を見る



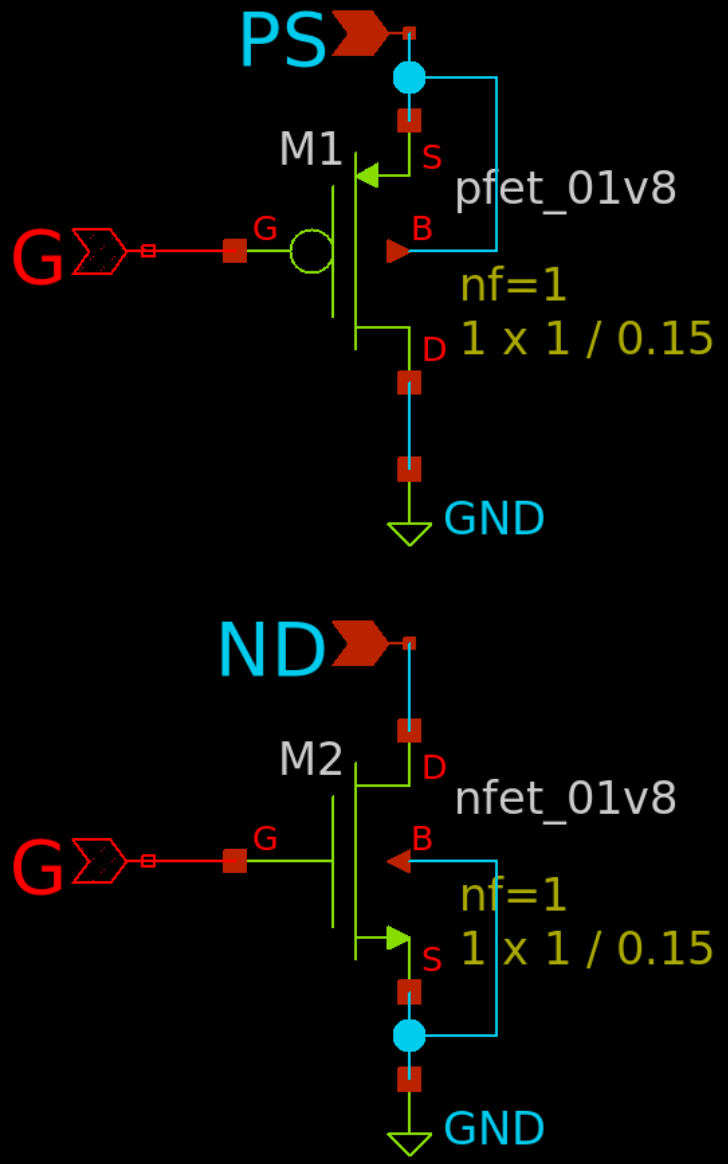
DC解析書式

dc [電圧源] [開始電圧] [終了電圧] [ステップ幅]

消費電流

ngspice: vp#branch, vn#branch または gaw: i(vp), i(vn)

Xschem 構成例 (L=0.15µm)



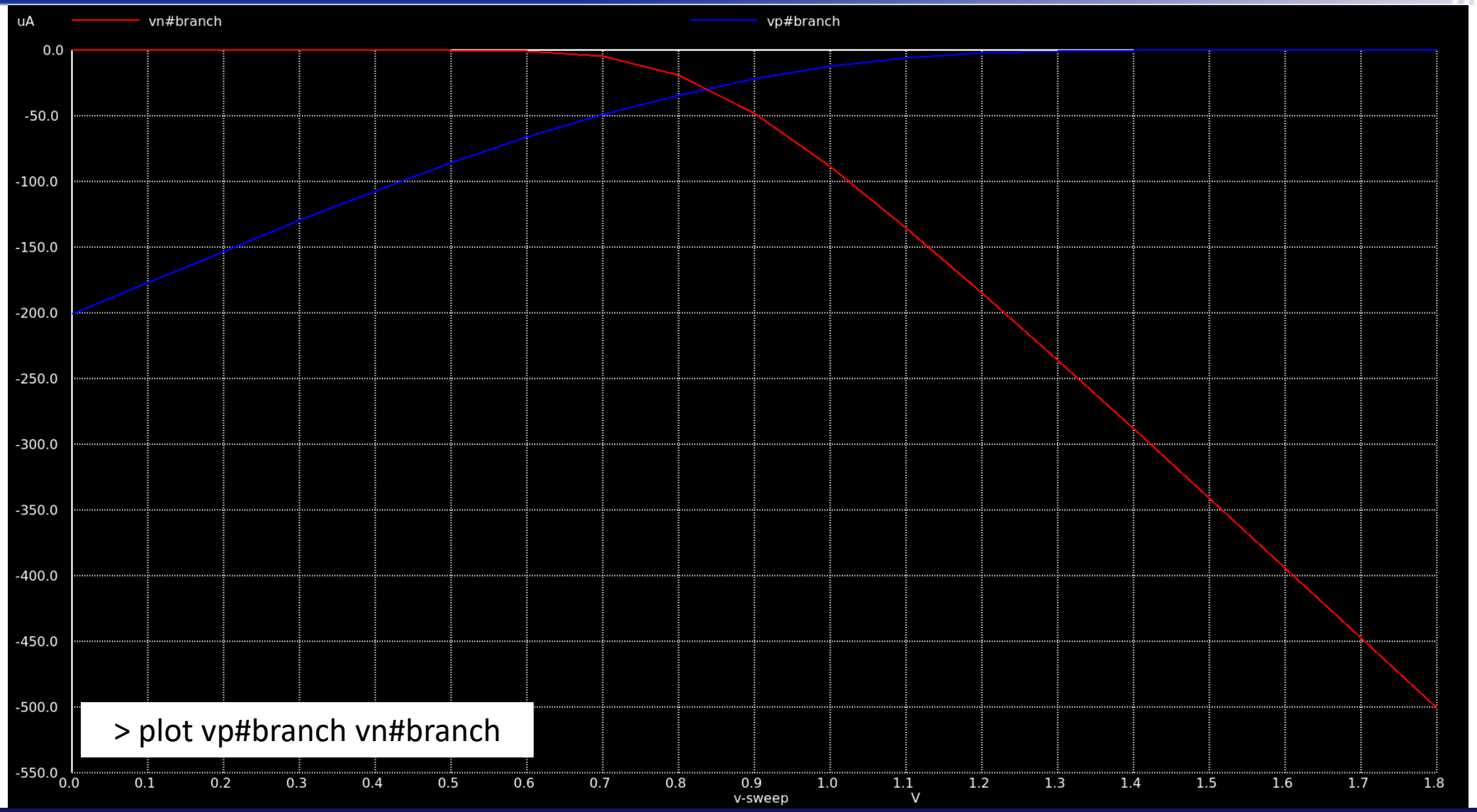
SPICEネットリストで
GNDは数字の0で書く

ngspice

```
VG G 0 dc 0
VP PS 0 dc 1.8
VN ND 0 dc 1.8
.control
save all
dc VG 0 1.8 0.1
write trbench.raw
.endc
```

ピン名は基本任意だが、
基準となるGND or 0を必ず入れる事
“VSS”は0に変換されない！

シミュレーション結果例



ngspiceの生データ出力方法

- wrdataで特定のデータを抽出可能
- ホームディレクトリ (/home/[ユーザ名]/) にvp.txt, vn.txtが生成される

vp.txt

```
0.00000000e+00 -2.00726126e-04
1.00000000e-01 -1.76783868e-04
2.00000000e-01 -1.52856174e-04
3.00000000e-01 -1.29370810e-04
4.00000000e-01 -1.06779404e-04
5.00000000e-01 -8.55349063e-05
6.00000000e-01 -6.60656564e-05
7.00000000e-01 -4.87492484e-05
8.00000000e-01 -3.38904626e-05
9.00000000e-01 -2.17083481e-05
1.00000000e+00 -1.23365952e-05
1.10000000e+00 -5.82743752e-06
1.20000000e+00 -2.07958988e-06
1.30000000e+00 -5.36969830e-07
1.40000000e+00 -1.19564554e-07
1.50000000e+00 -2.81003906e-08
1.60000000e+00 -6.92577344e-09
1.70000000e+00 -1.63614705e-09
1.80000000e+00 -3.19909099e-10
```

vn.txt

```
0.00000000e+00 -2.03659312e-12
1.00000000e-01 -4.80682161e-12
2.00000000e-01 -3.99689171e-11
3.00000000e-01 -4.84302376e-10
4.00000000e-01 -5.99945693e-09
5.00000000e-01 -7.04327370e-08
6.00000000e-01 -6.97912479e-07
7.00000000e-01 -4.74592920e-06
8.00000000e-01 -1.91451559e-05
9.00000000e-01 -4.83465908e-05
1.00000000e+00 -8.87674453e-05
1.10000000e+00 -1.35141610e-04
1.20000000e+00 -1.84593052e-04
1.30000000e+00 -2.35816015e-04
1.40000000e+00 -2.88115929e-04
1.50000000e+00 -3.41048569e-04
1.60000000e+00 -3.94303222e-04
1.70000000e+00 -4.47655558e-04
1.80000000e+00 -5.00941446e-04
```

```
ngspice
 VG G 0 dc 0
 VP PS 0 dc 1.8
 VN ND 0 dc 1.8
 .control
 dc VG 0 1.8 0.1
 wrdata ~/vp.txt vp#branch
 wrdata ~/vn.txt vn#branch
 .endc
```

オン抵抗

L=0.15µm, W=1.0µm のときの静特性

- **PMOS: 0.00000000e+00 -2.00726126e-04**
- ドレインソース間は1.8Vなので、1.8/2.01e-4 = 9.0kΩ

- **NMOS: 1.80000000e+00 -5.00941446e-04**
- ドレインソース間は1.8Vなので、1.8/5.01e-4 = 3.6kΩ

- オン抵抗とサイズは概ね逆比の関係がある　→　P : N = 2.5 : 1
  - トランジスタサイズは通常、Wを変える

演習

SKY130で学ぶLSI回路設計

naklab

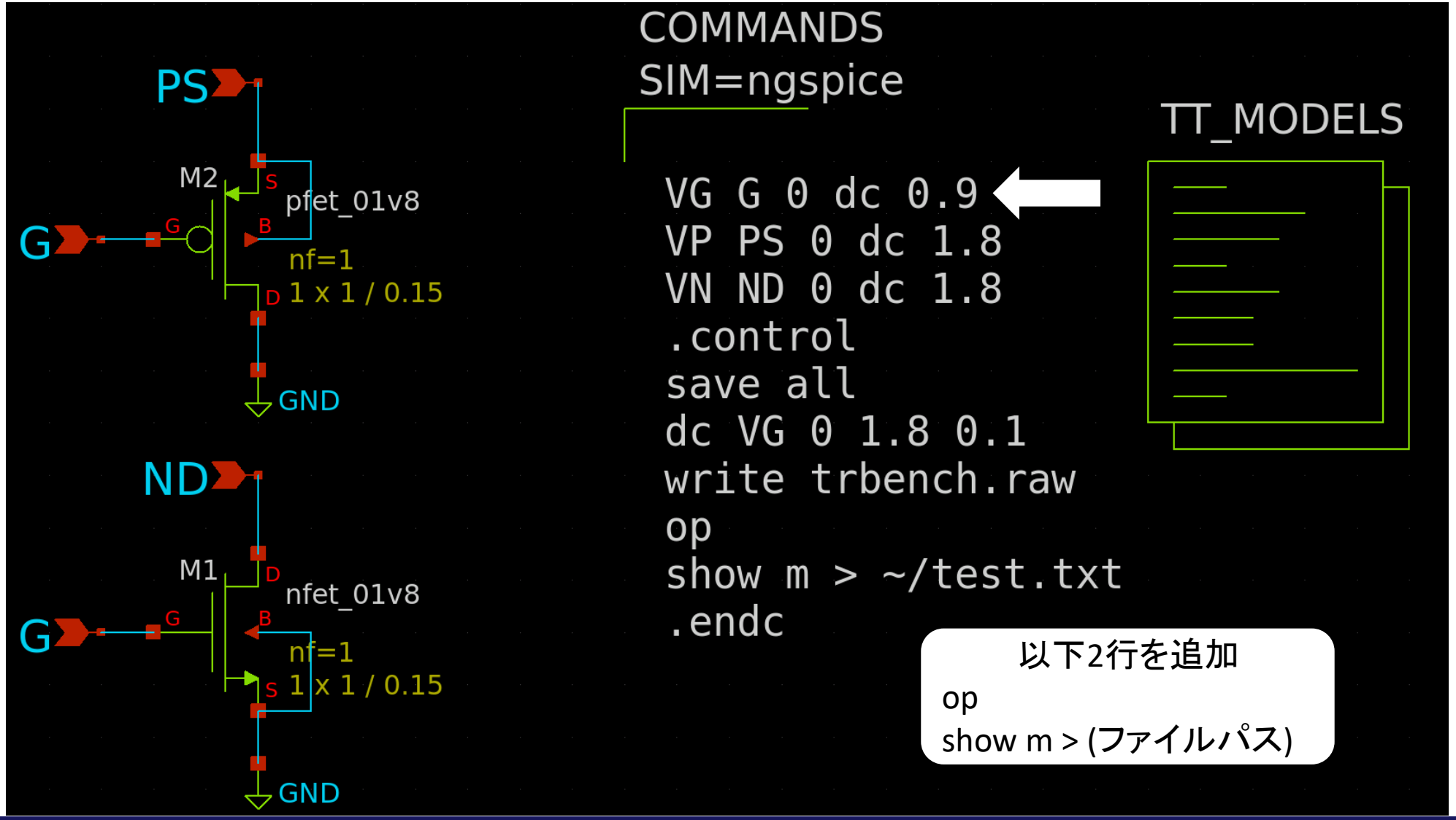
参考：入力容量の求め方

注意事項

- ネットでMOSFETの入力容量(Ciss)を検索すると下記の式がでてくる
- Ciss = Cgs + Cgd　＝　ゲートソース間容量　＋　ゲートドレイン間容量

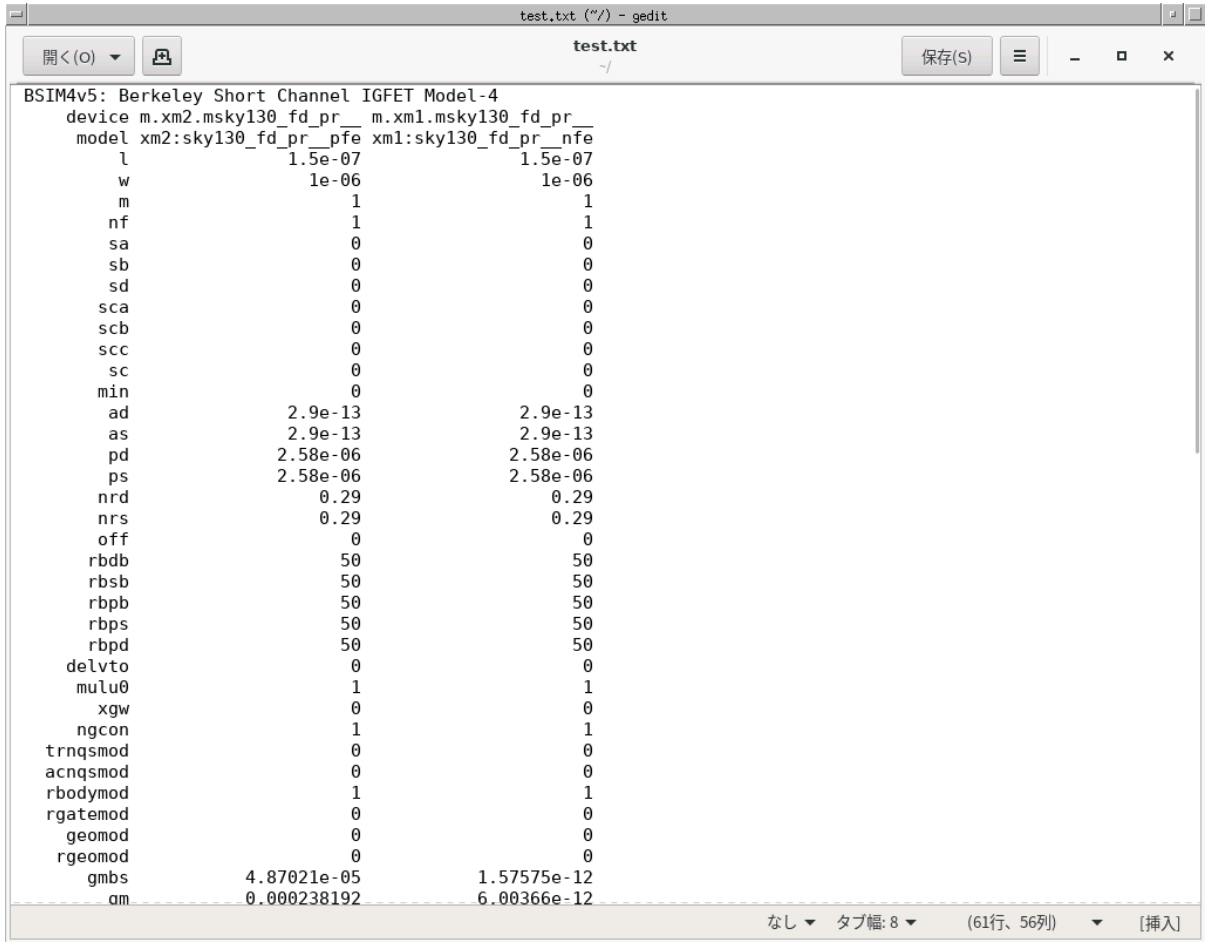
- LSIのCMOSではゲートボディ間容量も考慮する必要がある
- Cgg = |Cgs + Cgd + Cgb|
  - ゲート容量はCggというパラメータで表される
  - 出力容量=ドレイン容量 Cdd = |Cdg + Cds + Cdb|
    - Cgd、Cdgはどちらもドレイン･ゲート間の寄生容量を表しているが、
      Cdgはドレイン電圧が変化した時に影響する寄生容量である一方、
      Cgdは　ゲート電圧が変化した時に影響する寄生容量。 普通は Cdg ≠ Cgd

動作点解析(OP)


PS
M2
S
pfet_01v8
G
G
B
nf=1
D
1 x 1 / 0.15
GND
ND
M1
D
nfet_01v8
G
G
B
nf=1
S
1 x 1 / 0.15
GND
COMMANDS
SIM=ngspice
VG G 0 dc 0.9
VP PS 0 dc 1.8
VN ND 0 dc 1.8
.control
save all
dc VG 0 1.8 0.1
write trbench.raw
op
show m > ~/test.txt
.endc
TT_MODELS
以下2行を追加
op
show m > (ファイルパス)

show m

- 長いのでテキストへの保存をオススメ
  - show m > ~/test.txt　など
- 保存したテキストのみかた
  - gedit ~/test.txt など




```
BSIM4v5: Berkeley Short Channel IGFET Model-4
    device m.xm2.msky130_fd_pr__ m.xm1.msky130_fd_pr__
     model xm2:sky130_fd_pr__pfe xm1:sky130_fd_pr__nfe
         l               1.5e-07               1.5e-07
         w                 1e-06                 1e-06
         m                     1                     1
        nf                     1                     1
        sa                     0                     0
        sb                     0                     0
        sd                     0                     0
       sca                     0                     0
       scb                     0                     0
       scc                     0                     0
        sc                     0                     0
       min                     0                     0
        ad               2.9e-13               2.9e-13
        as               2.9e-13               2.9e-13
        pd              2.58e-06              2.58e-06
        ps              2.58e-06              2.58e-06
       nrd                  0.29                  0.29
       nrs                  0.29                  0.29
       off                     0                     0
      rbdb                    50                    50
      rbsb                    50                    50
      rbpb                    50                    50
      rbps                    50                    50
      rbpd                    50                    50
    delvto                     0                     0
     mulu0                     1                     1
       xgw                     0                     0
     ngcon                     1                     1
  trnqsmod                     0                     0
  acnqsmod                     0                     0
  rbodymod                     1                     1
  rgatemod                     0                     0
    geomod                     0                     0
   rgeomod                     0                     0
      gmbs           4.87021e-05           1.57575e-12
        gm          0.000238192           6.00366e-12
```

演習

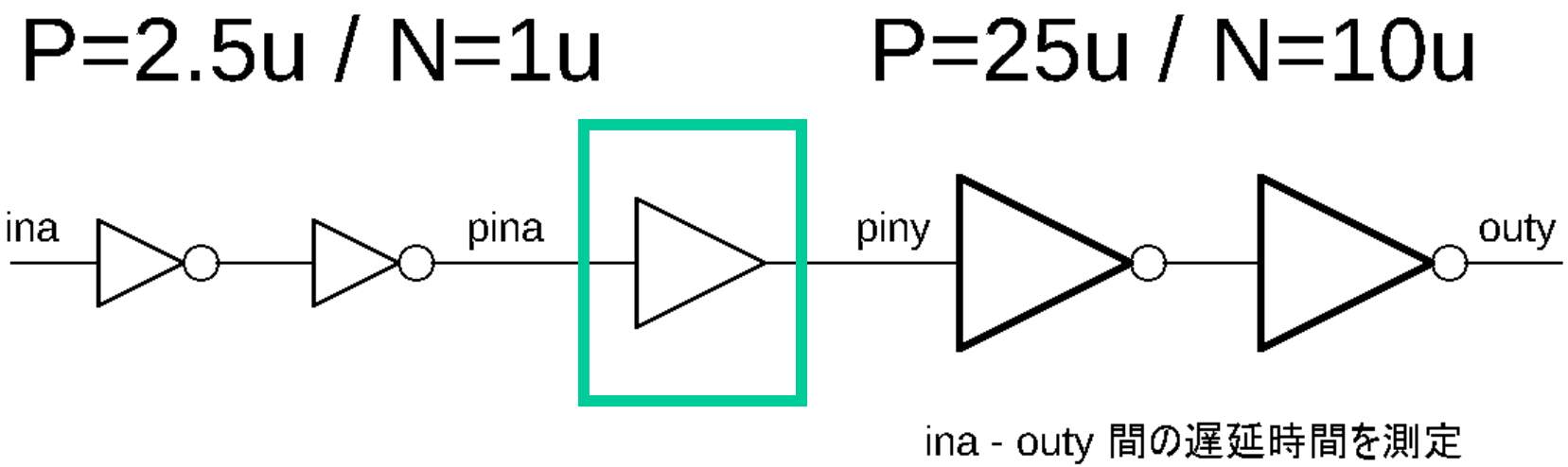
SKY130で学ぶLSI回路設計

naklab

インバータ／バッファの遅延時間

インバータ／バッファの遅延時間

- 遅延時間を検証する際は入力段・出力段のFOが必要！



- 初段インバータサイズは $w_p = 2.5\ \mu m$, $w_n = 1\ \mu m$
- 出力段の負荷容量は入力容量の10倍
- 中間にバッファ (インバータ×2)を置く　このバッファ設計する
- Ina から outy までの遅延時間の最小化をめざす

バッファ遅延時間

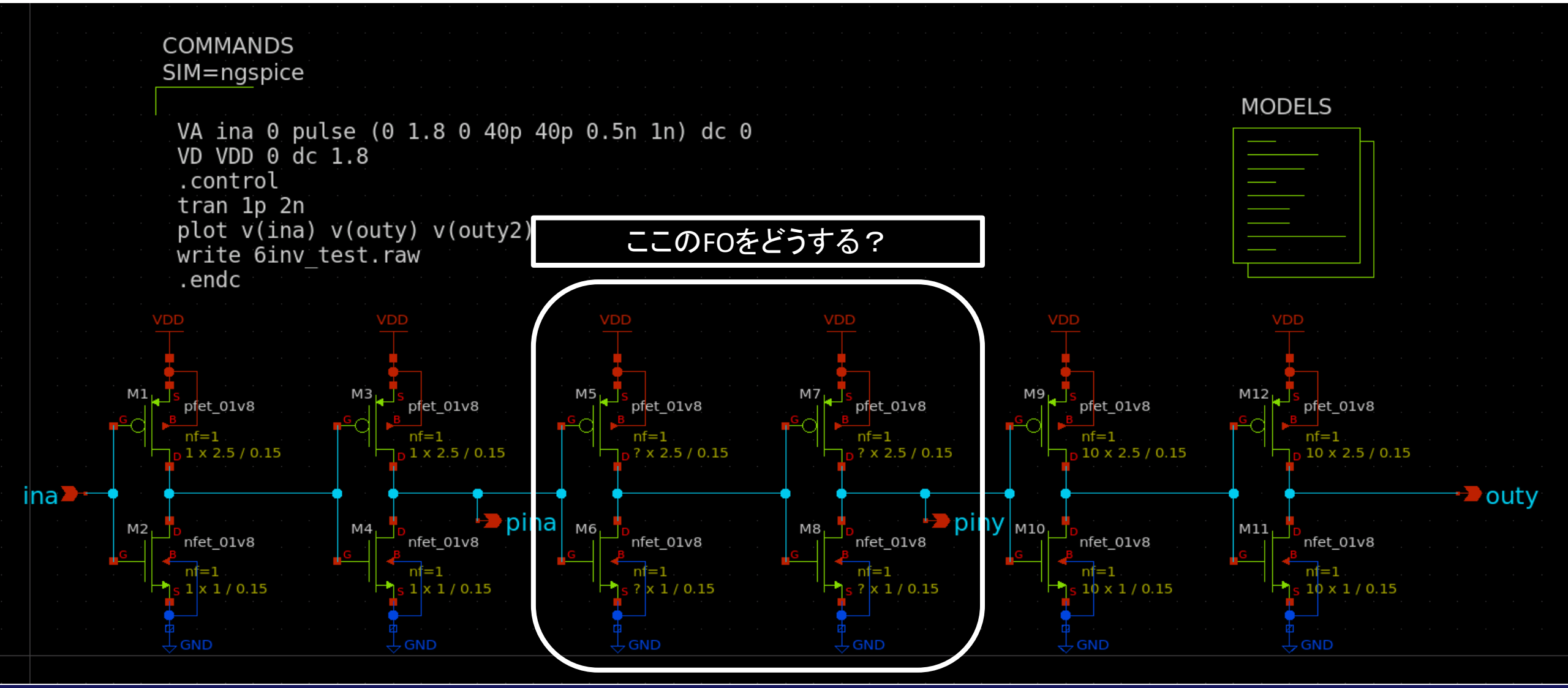
- 6段のインバータチェインをシミュレーション

- 時間対電圧の波形を見たいのでトランジェント解析を行う

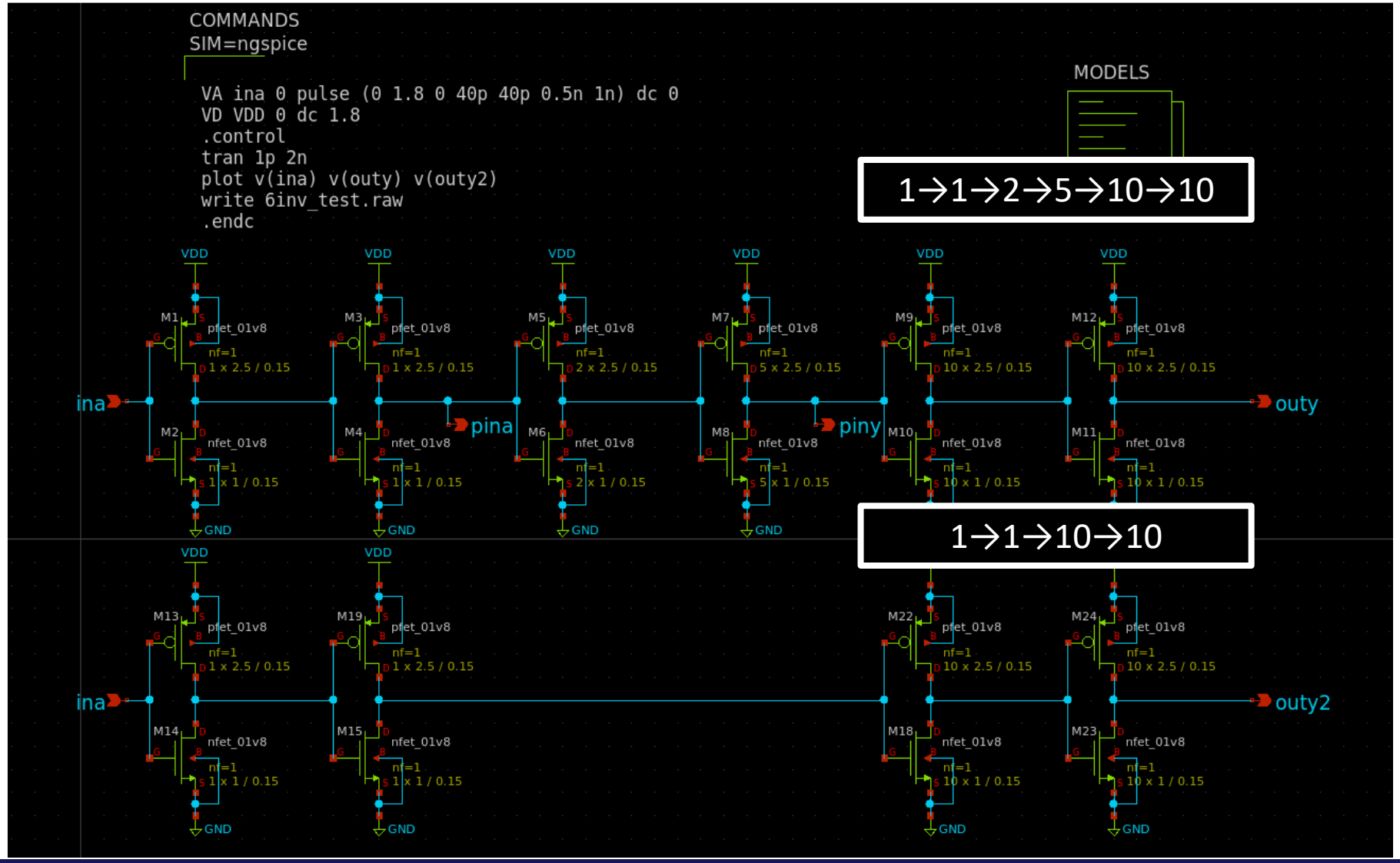
トランジェント解析書式

tran [最小ステップ幅] [解析時間]

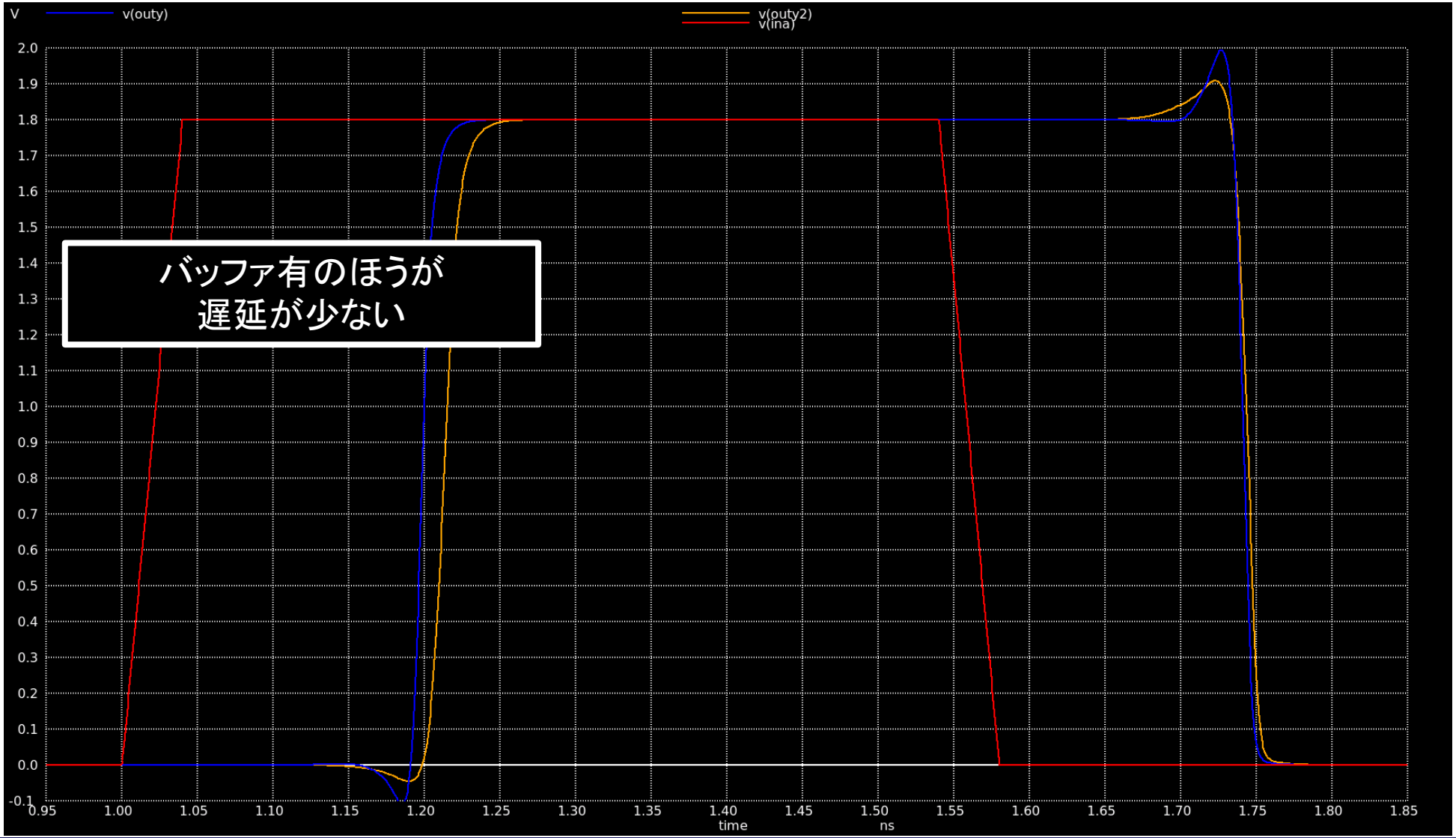
Xschem 構成例 (L=0.15µm)



Xschem 構成例 (L=0.15µm)



シミュレーション結果


V
v(outy)
v(outy2)
v(ina)
バッファ有のほうが
遅延が少ない
2.0
1.9
1.8
1.7
1.6
1.5
1.4
1.3
1.2
1.1
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
-0.1
0.95
1.00
1.05
1.10
1.15
1.20
1.25
1.30
1.35
1.40
1.45
1.50
1.55
1.60
1.65
1.70
1.75
1.80
1.85
time
ns

meas を使う

- 波形やグラフから遅延時間を都度求めるのは面倒　→　meas
  - 多機能コマンドだが、今回は遅延時間算出に絞って紹介

meas 書式

meas tran [変数] FIND time WHEN [条件] [コマンド]

- [変数] - 結果を代入する変数
- [条件] - 今回は “ v(クロック) = Vdd / 2 ”
- コマンド : RISE / FALL / CROSS のいずれかだが、CROSSは恐らく使わない。

meas を使う

- 波形やグラフから遅延時間を都度求めるのは面倒　→　meas
  - 多機能コマンドだが、今回は遅延時間算出に絞って紹介

meas 書式例

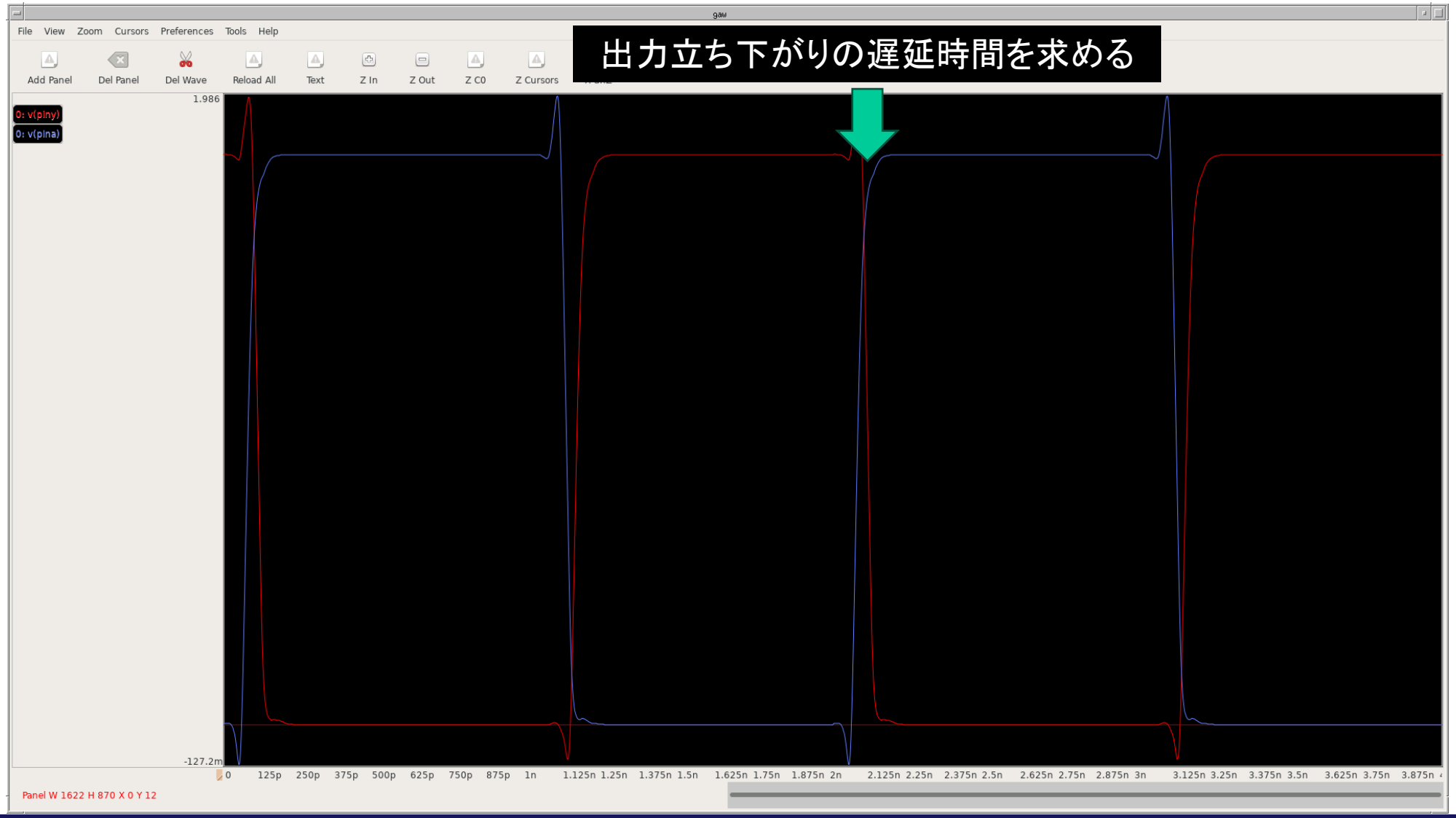
meas tran delay1 FIND time WHEN v(vin)=0.9 RISE=2

v(vin)が2回目の立ち上がりで0.9Vの時の時間をdelay1に代入

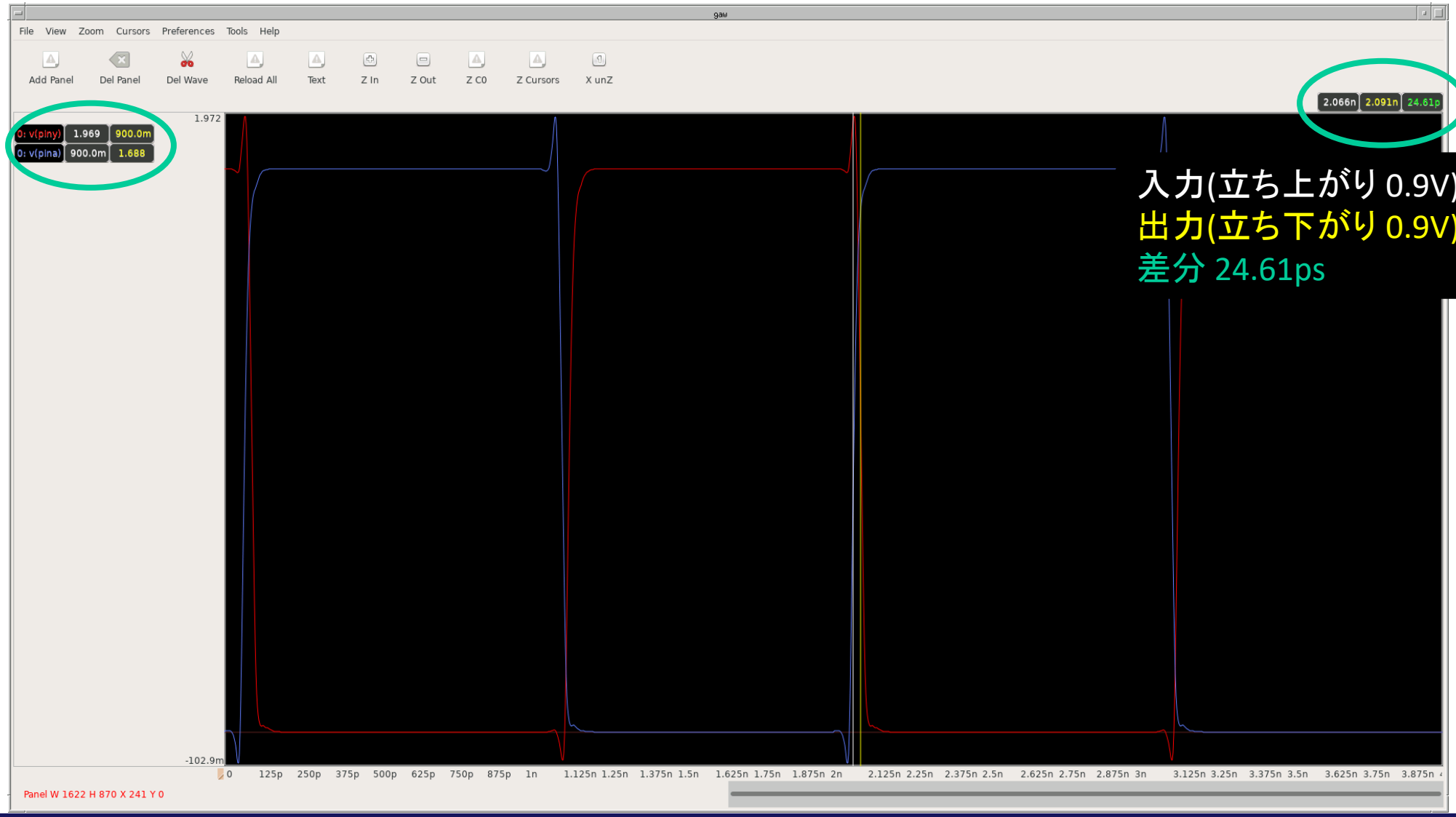
meas tran delay2 FIND time WHEN v(vout)=0.9 FALL=2

v(vout)が2回目の立ち下がりで0.9Vの時の時間をdelay2に代入

ここの遅延時間を見る

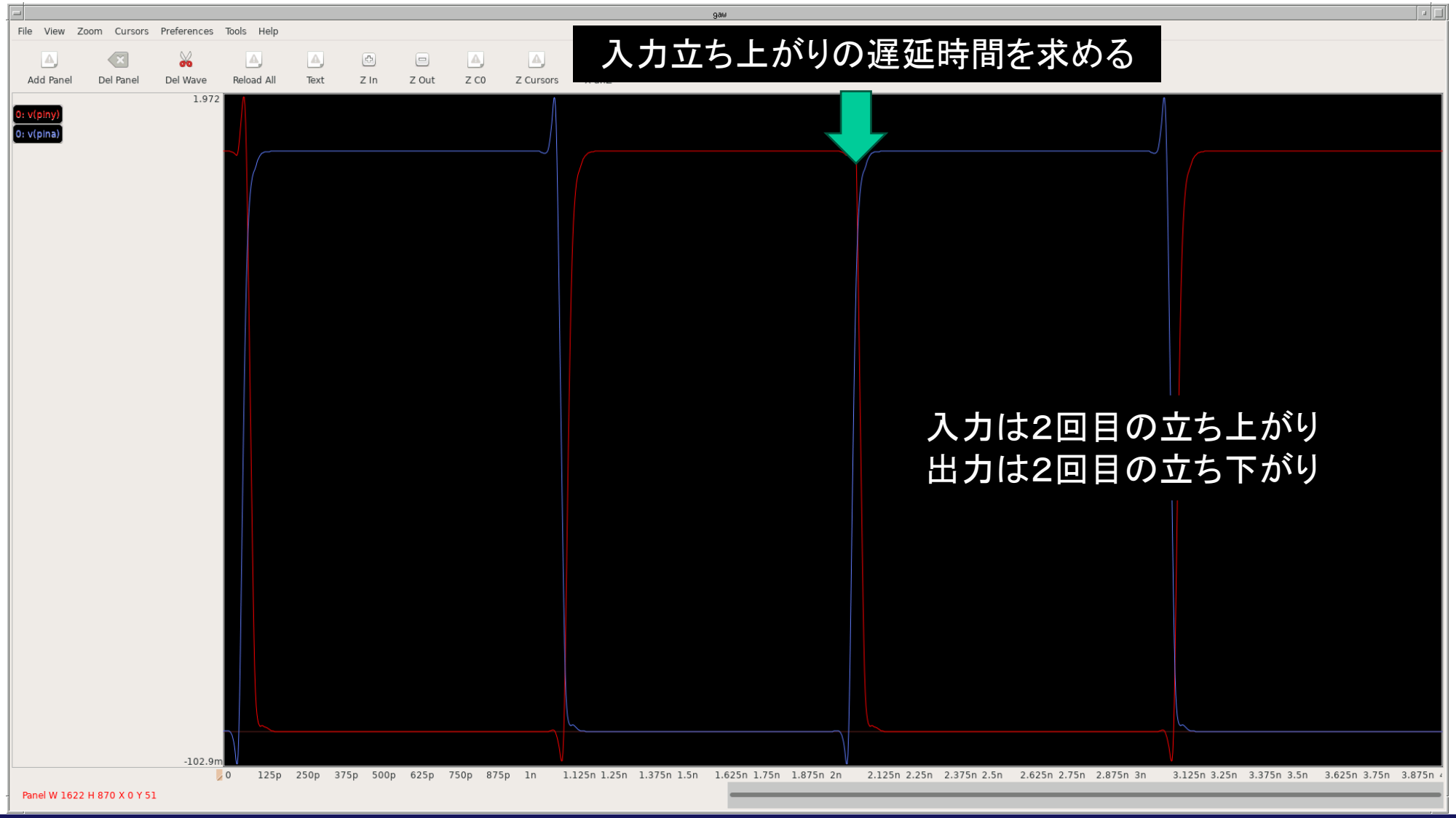

gaw
File View Zoom Cursors Preferences Tools Help
Add Panel
Del Panel
Del Wave
Reload All
Text
Z In
Z Out
Z C0
Z Cursors
出力立ち下がりの遅延時間を求める
0: v(piny)
0: v(pina)
1.986
-127.2m
0 125p 250p 375p 500p 625p 750p 875p 1n 1.125n 1.25n 1.375n 1.5n 1.625n 1.75n 1.875n 2n 2.125n 2.25n 2.375n 2.5n 2.625n 2.75n 2.875n 3n 3.125n 3.25n 3.375n 3.5n 3.625n 3.75n 3.875n
Panel W 1622 H 870 X 0 Y 12

波形ビューワで遅延時間を見ると…



入力(立ち上がり 0.9V) 2.066ns
出力(立ち下がり 0.9V) 2.091ns
差分 24.61ps

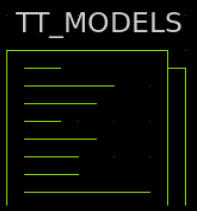
meas 遅延時間を見る



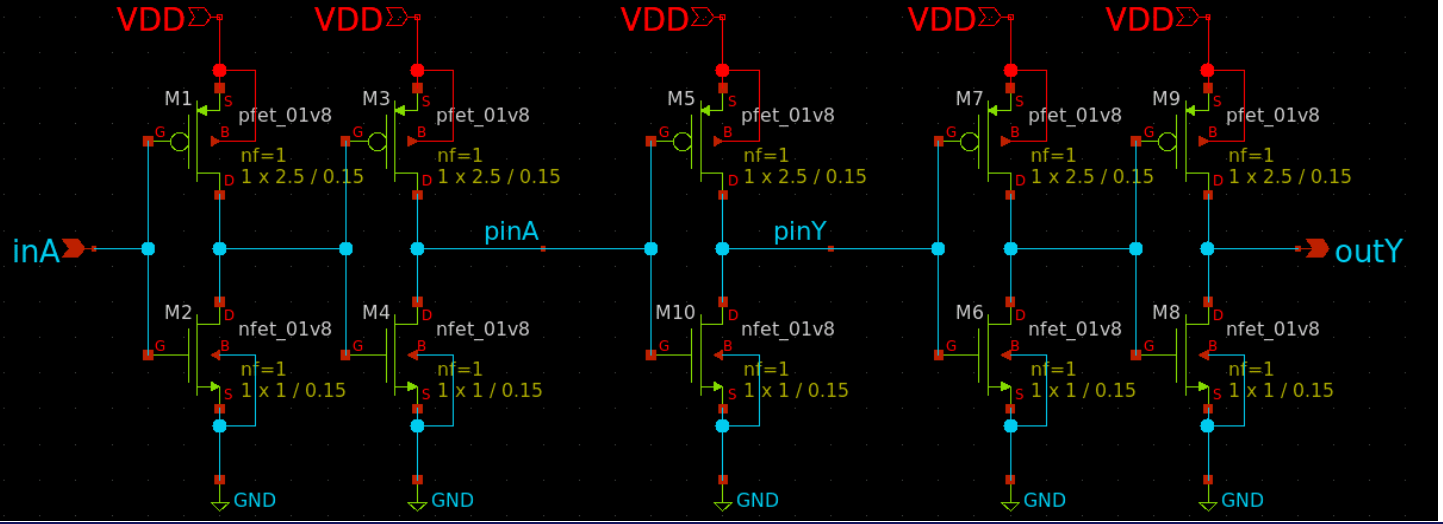
measで遅延時間を見ると…

```
ngspice
 VA inA 0 pulse(0 1.8 0 40p 40p 1n 2n) dc 0
 VD VDD 0 dc 1.8
 .control
 save all
 tran 1p 4n
 meas tran delayA1 find time WHEN v(pinA)=0.9 RISE=2
 meas tran delayY1 find time WHEN v(pinY)=0.9 FALL=2
 let delay1 = delayY1 - delayA1
 echo tpHL = $&delay1
 .endc
```

TT_MODELS



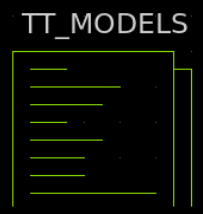
入力は2回目の立ち上がり
出力は2回目の立ち下がり



measで遅延時間を見ると…

```
ngspice
 VA inA 0 pulse(0 1.8 0 40p 40p 1n 2n) dc 0
 VD VDD 0 dc 1.8
 .control
 save all
 tran 1p 4n
 meas tran delayA1 find time WHEN v(pinA)=0.9 RISE=2
 meas tran delayY1 find time WHEN v(pinY)=0.9 FALL=2
 let delay1 = delayY1 - delayA1
 echo tpHL = $&delay1
 .endc
```

TT_MODELS



入力は2回目の立ち上がり
出力は2回目の立ち下がり

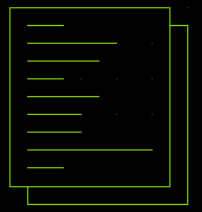
```
 Reference value :  2.34850e-09
No. of Data Rows : 4029
delaya1                =  2.066458e-09
delayy1                =  2.091065e-09
tpHL = 2.4607E-11 ←遅延時間が自動で求まる
ngspice 1 ->
```

同様にして出力立ち上がり遅延時間も見る

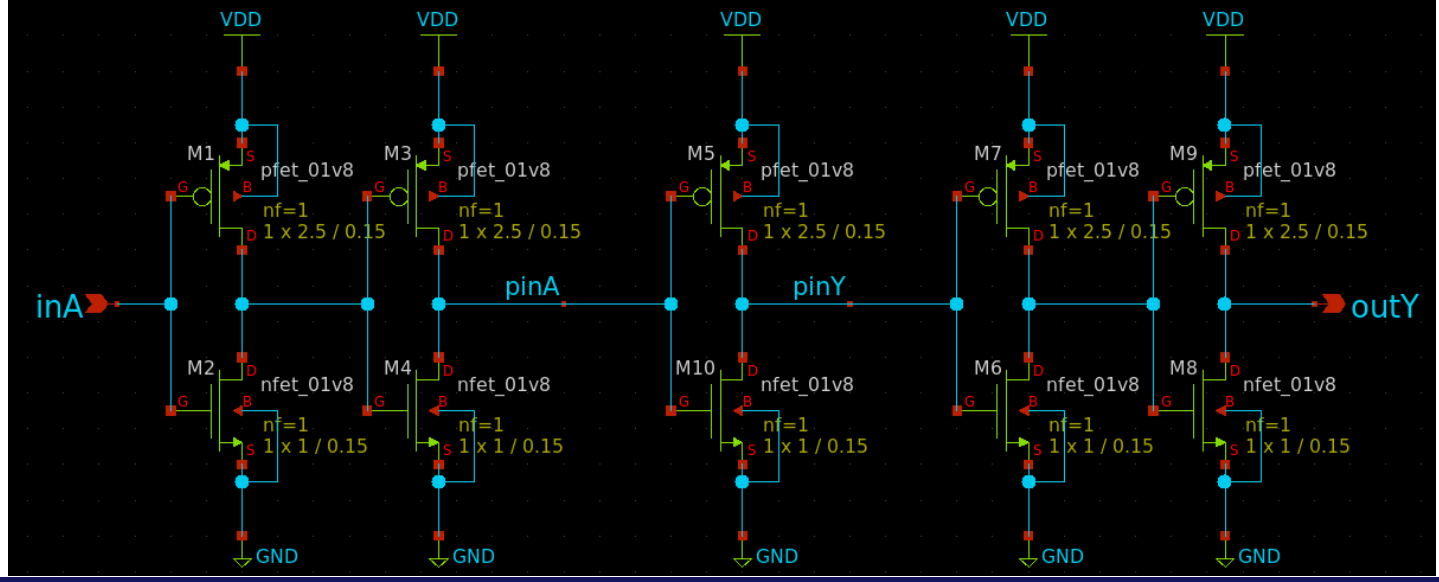
```
ngspice
VA inA 0 pulse(0 1.8 0 40p 40p 1n 2n) dc 0
VD VDD 0 dc 1.8
.control
save all
tran 1p 4n
meas tran delayA1 find time WHEN v(pinA)=0.9 RISE=2
meas tran delayY1 find time WHEN v(pinY)=0.9 FALL=2
let delay1 = delayY1 - delayA1
echo tpHL = $&delay1

meas tran delayA2 find time WHEN v(pinA)=0.9 FALL=1
meas tran delayY2 find time WHEN v(pinY)=0.9 RISE=1
let delay2 = delayY2 - delayA2
echo tpLH = $&delay2
.endc
```

inv_delay.sch

TT_MODELS



```
 Reference value :  2.37850e-09
No. of Data Rows : 4029
delaya1                =  2.066458e-09
delayy1                =  2.091065e-09
tpHL = 2.4607E-11
delaya2                =  1.109285e-09
delayy2                =  1.132613e-09
tpLH = 2.3328E-11
ngspice 1 ->
```



演習

SKY130で学ぶLSI回路設計

naklab

立ち上がり時間と立ち下がり時間

立ち上がり時間と立ち下がり時間

- オン抵抗、入力容量Cgg、出力容量Cdd のパラメータを得た

- 第4回の講義資料より、遅延時間の近似式を再定義
- $t_{PHL} = 0.75 \times (C_{dd.no} + C_{dd.po} + C_{gg.ni} + C_{gg.pi})R_n$
  - 入力立ち上がり・出力立ち下がり遅延
- $t_{PLH} = 0.75 \times (C_{dd.no} + C_{dd.po} + C_{gg.ni} + C_{gg.pi})R_p$
  - 入力立ち下がり・出力立ち上がり遅延
  - Cdd = ドレイン容量、Cgg = ゲート容量

- $R_n = R_p$なら立ち上がり遅延と立ち下がり遅延は一致する？

立ち上がり時間と立ち下がり時間

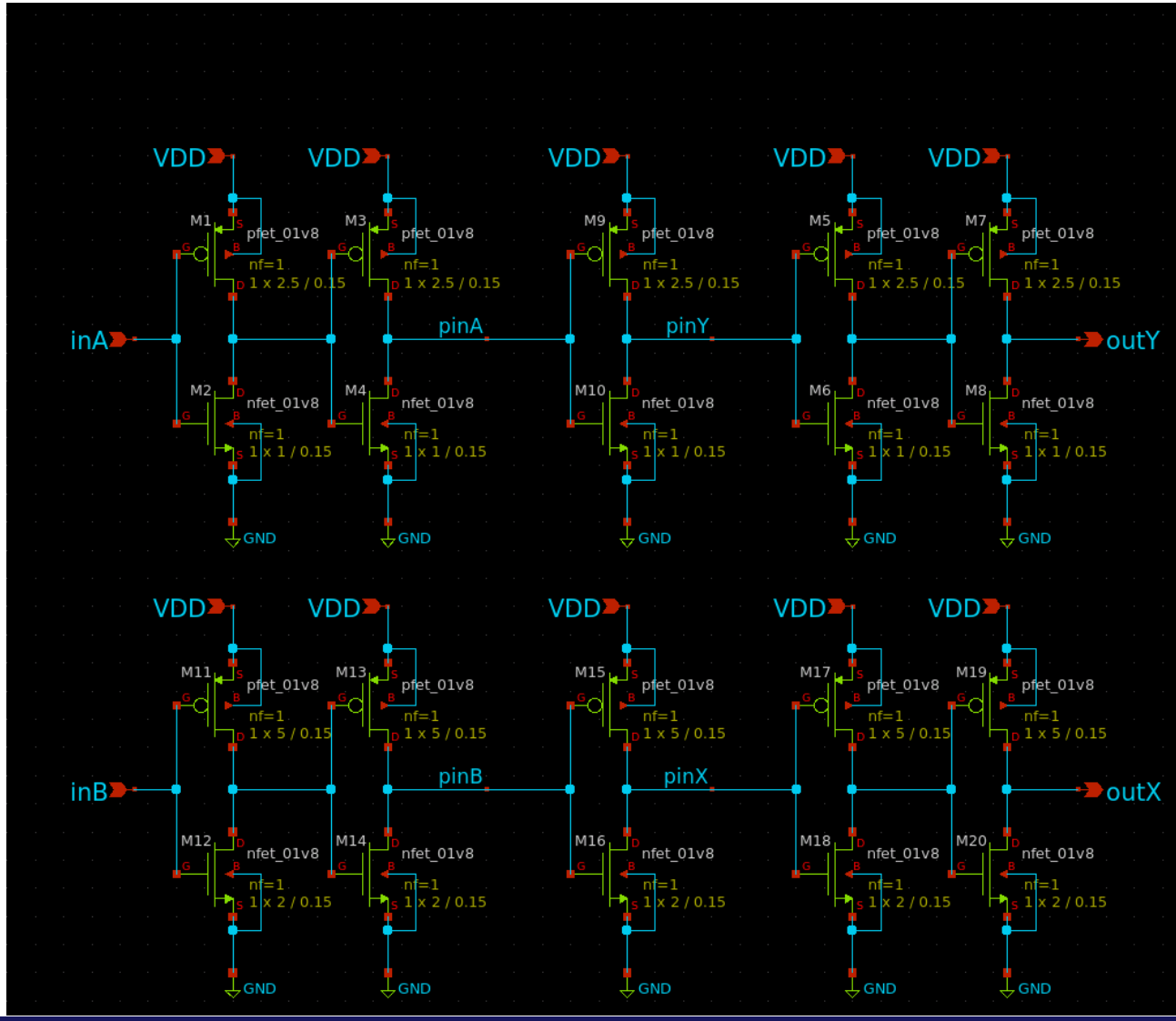
- N=1u 固定、PMOSのWを変更
  - 入力段・出力段のバッファも変更した

	2.0u	2.1u	2.2u	2.3u
t_{PHL}	23.238ps	23.496ps	23.764ps	24.035ps
t_{PLH}	23.889ps	23.725ps	23.591ps	23.482ps

  - N=1uならP=2.1u or 2.2u がよさそう

- $R_n = R_p$で立ち上がり遅延と立ち下がり遅延は一致しない
  - ちなみにDC解析では $w_p : w_n = 2.5 : 1$だった

Q.遅延時間が小さいのはどっち？



inv_delay2.sch

ngspice

```
VA inA 0 pulse(0 1.8 0 40p 40p 1n 2n) dc 0
VB inB 0 pulse(0 1.8 0 40p 40p 1n 2n) dc 0
VD VDD 0 dc 1.8
.control
tran 1p 4n
meas tran delayA1 find time WHEN v(pinA)=0.9 RISE=2
meas tran delayY1 find time WHEN v(pinY)=0.9 FALL=2
let delay1 = delayY1 - delayA1
echo tpHL = $&delay1

meas tran delayB1 find time WHEN v(pinB)=0.9 RISE=2
meas tran delayX1 find time WHEN v(pinX)=0.9 FALL=2
let delay1b = delayX1 - delayB1
echo tpHL = $&delay1b

meas tran delayA2 find time WHEN v(pinA)=0.9 FALL=1
meas tran delayY2 find time WHEN v(pinY)=0.9 RISE=1
let delay2 = delayY2 - delayA2
echo tpLH = $&delay2

meas tran delayB2 find time WHEN v(pinB)=0.9 FALL=1
meas tran delayX2 find time WHEN v(pinX)=0.9 RISE=1
let delay2b = delayX2 - delayB2
echo tpLH = $&delay2b

.endc
```

MODELS

ファンアウトについて

- PとNの比が同じであれば、大きいほうがより低遅延

- 低遅延を求めるほどレイアウトサイズは大きくなっていく
  - 実際は配線の寄生容量とVIAの寄生抵抗があるため、どこかで頭打ちになる

	t_{PHL}	t_{PLH}
P : N = 2.2 : 1.0	23.764ps	23.591ps
P : N = 4.4 : 2.0	22.384ps	22.952ps

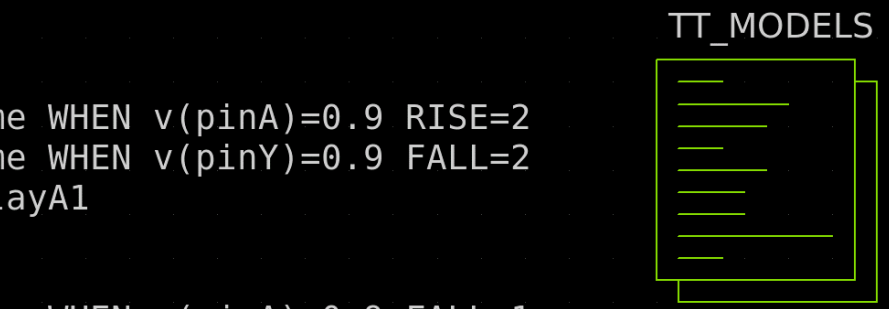
ステップその1

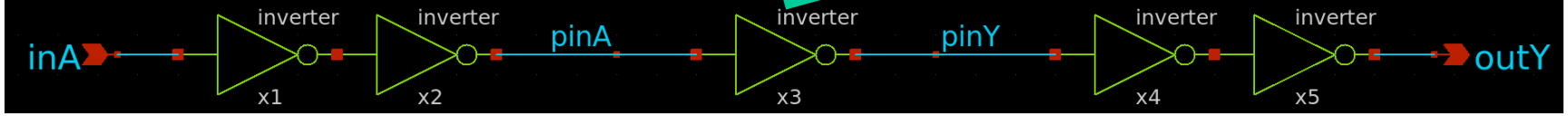
- 立ち上がり遅延と立ち下がり遅延が殆ど同じになるようなPMOSとNMOSのサイズを見つける

ngspice

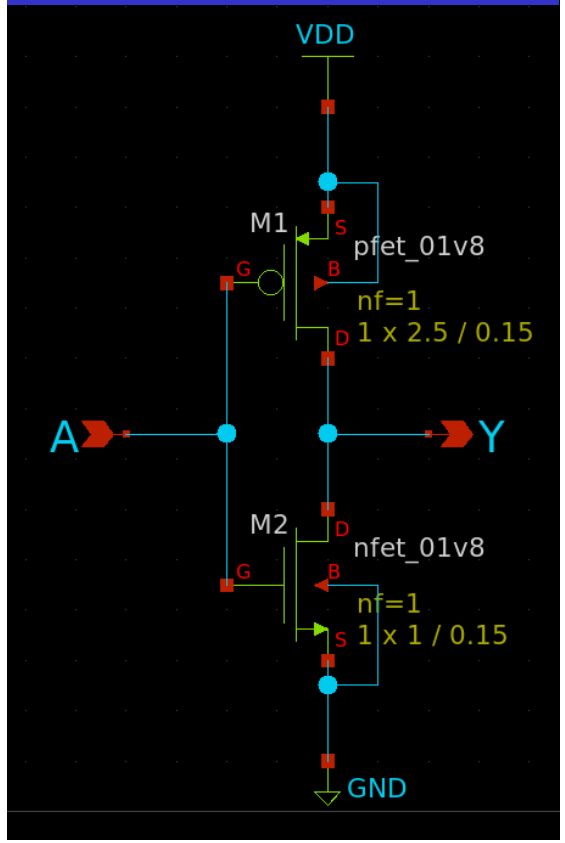
```
VA inA 0 pulse(0 1.8 0 40p 40p 1n 2n) dc 0
VD VDD 0 dc 1.8
.control
save all
tran 1p 4n
meas tran delayA1 find time WHEN v(pinA)=0.9 RISE=2
meas tran delayY1 find time WHEN v(pinY)=0.9 FALL=2
let delay1 = delayY1 - delayA1
echo tpHL = $&delay1

meas tran delayA2 find time WHEN v(pinA)=0.9 FALL=1
meas tran delayY2 find time WHEN v(pinY)=0.9 RISE=1
let delay2 = delayY2 - delayA2
echo tpLH = $&delay2
.endc
```

kadai1.sch





inverter.sch



注意事項

- Lは0.15uまたは0.18uのどちらかにする

- Wは0.05uステップで指定する
  - W=2.23uとかはダメ
  - 厳密に揃えようとしなくても良い
    - w_p：w_n=2.2：1 (L=0.15u) でも良い

- 提出課題ではありませんが、今回決定したPMOSとNMOSのサイズは次回以降デザインを設計する上での基準(FO=1)になります
  - 2で割りにくい値や極端な値は避けるべき
  - NMOSのw_nは1u – 8uの範囲がオススメ