

# オープンソース PDKの歩き方

読む・触る・作る・遊び続けるために

ほうた@SIGの遊び場

# 自己紹介

## ほうた@SIGの遊び場

### 謎のコメント

そういえば「ISHI会一周年記念イベント～大討論会～」で「北城 岳彦（ほうじょう たけひこ）

- ・1990年 総合電機メーカーに入社し、半導体事業部へ。  
デザインルール、ゲートアレイ、スタンダードセル開発部門に配属。
- ・最初の担当は、レジスタファイル(小容量メモリ)開発。
- ・その後、DRAMを搭載した ASIC(当時としては大規模メモリ混載) を開発するプロジェクトが始まり、ここで 岡村さんと出会う ことになる。
- ・



**TOSHIBA**



メモリコア部  
岡村さん

I/F部  
ほうた

新時代ASICの幕開け  
最先端システムをリードする…  
**0.14 $\mu$ mASIC**

■高集積、超低消費電力、高速度を同時に実現する東芝CMOS ASIC<TC260シリーズ>  
デジタル・モバイルシステムの高機能、高速、超低消費電力、省スペースを実現する0.14 $\mu$ mASIC<TC280シリーズ>、ゲート当たり3.4倍の高集積化、約1/5の超低消費電力、約30%の高速化を達成(当社0.25 $\mu$ mASIC・TC240シリーズ比)メカゲート時代のシステムASICをリードする新世代ASICです。

- ゲート長0.14 $\mu$ mのCMOSプロセスルールを技術、6重アルミ配線による発売機ASIC
  - ・ゲート取を途切密度(3.4倍の高集積化)
  - ・ゲート当たり1/5の超低消費電力
  - ・ゲート当たり60Rtの高集積化
- プロセスインテグレーション
  - ・CMOSはシックプロセスとDRAMプロセスの両利性を兼ねる最良プロセス使用、MPUコア、DRAMコア、アナログ回路をそれぞれの性能を壊さず一括チェック上に画成
- メモリコアの充実
  - ・基座BRAM(最大500MHz、1ポート&2ポートU)最集積BRAM(従来1/4の面積、1ポート&2ポート)を併用
  - ・TC560シリーズ向け当時の消費電力トレンチャバンク形DRAMコアをリリース

**東芝CMOS ASIC**

EDA TechnoFair '99

〒ADDRESS▶ お問い合わせは、株式会社 東京 セミコンリテター社 システムLS部 株光版 システムLS部 東芝第二ビル  
TEL35-5167-9422 ライクライン / T108-9001 東京港区赤坂1-1-1 東芝とセテイング <http://www.cemicon.toshiba.co.jp/>

1999年雑誌広告  
(白黒コピーをAIでカラー化)

こんなものを作っていました

- その後もPDK・ライブラリ（スタセル、メモリ、IOなどのいろいろな基本部品）開発に携わる事になり、、今に至る。

- そして――

四半世紀ぶりに岡村さんと再会し、

“オープンソース PDK” の世界を知ることになる。



# そもそもPDKって？（Process Design Kit）

半導体設計をするために必要なすべての技術情報をまとめたパッケージ

- トランジスタモデル（SPICE）
- デバイスパラメータ
- デザインルール
- DRC / LVS / ERCルール
- 寄生抽出ルール（LPE/PEX）
- 標準セル（デジタル設計に必要なライブラリ）
- 配置配線向け設定（自動設計環境）



ほかにもいろいろ

…などなど、半導体設計に必要なものすべてをパッケージ化されたもの

# 昔は「三種の神器」これだけで設計ができた時代

まだプロセスが比較的単純だった頃、IC設計に必要なのは以下の3つだけだった

## デバイスパラメータ

- トランジスタの閾値、特性、寸法依存性など
- データシートやアプリケーションノートで提供されていた


## デザインルール

- デバイス/配線の長さ/幅・間隔・コンタクトの最小サイズなど
- A4数枚～十数ページに収まる“単純なルール集”

## SPICEモデル

- 回路シミュレーションができれば十分だった
- 1～3種類のモデルで事足りた

昔は“この三種の神器さえあれば”チップを作れた。



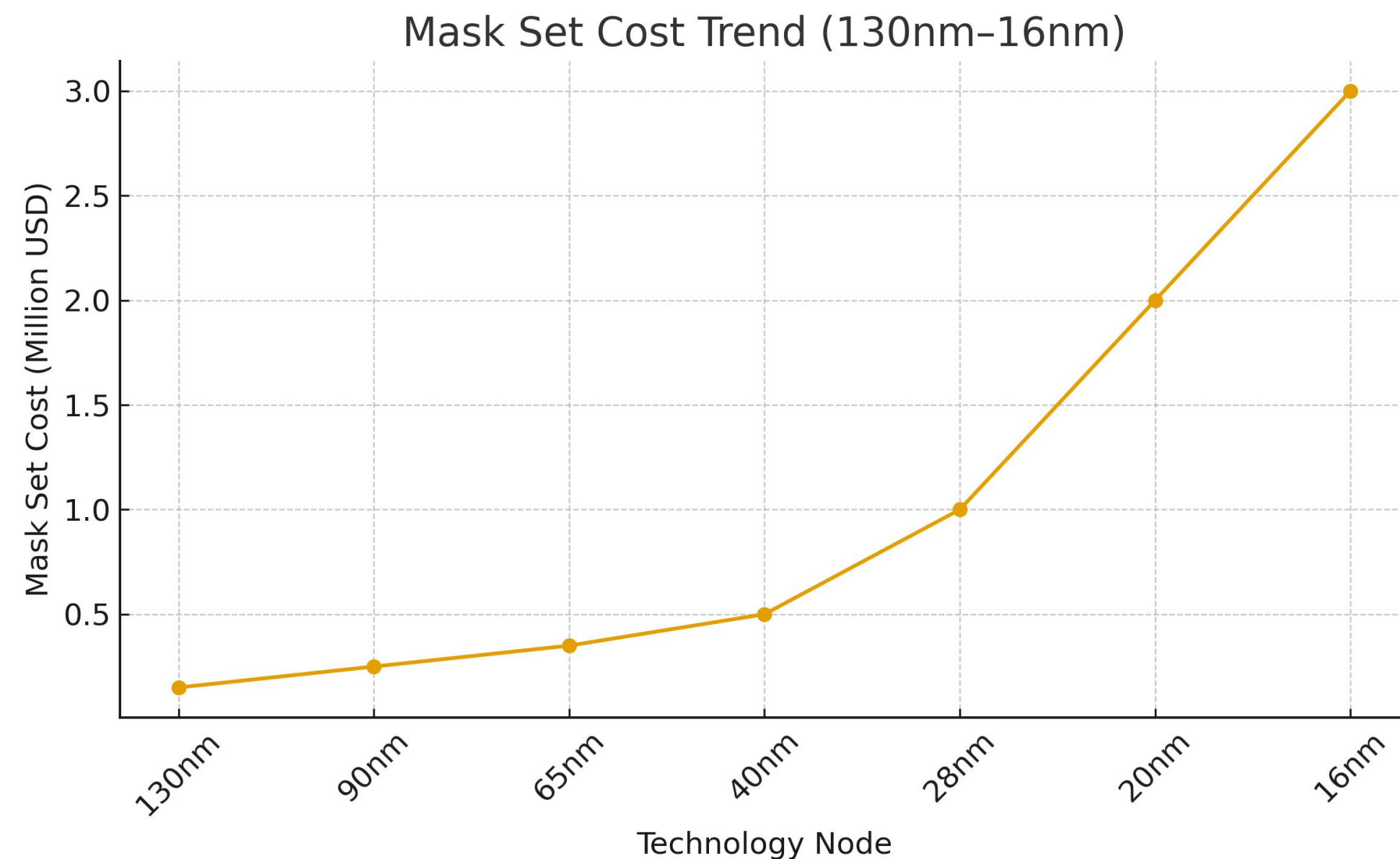
実際には設計者が、これらの情報から  
Symbol, Pcell,  
DRC, LVS  
を作っていた

# しかし微細化により激変⇒PDK充実の時代へ

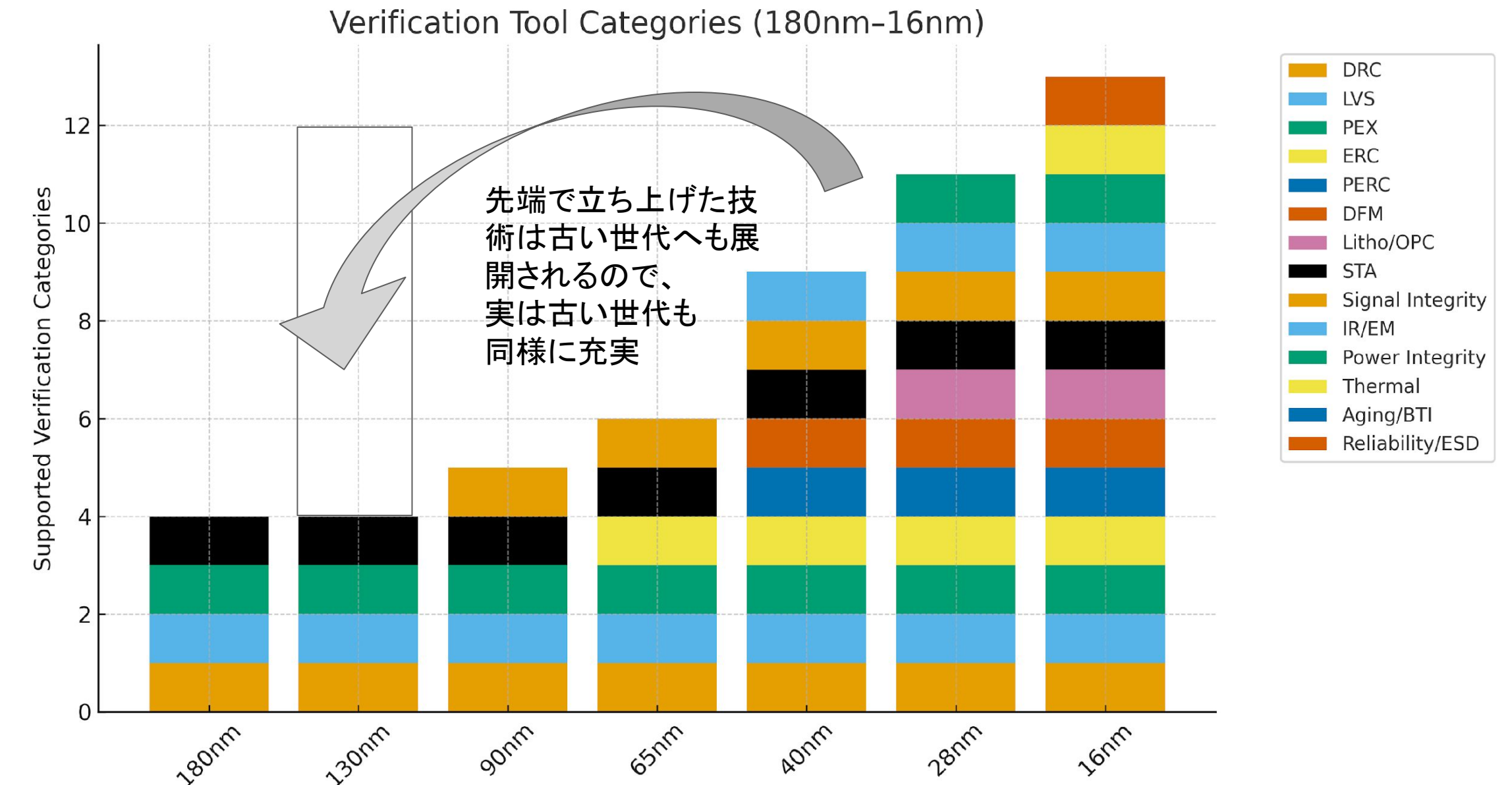
プロセスが複雑化し、レイヤ数もデバイスタイプも増加。  
さらにEDAツールが多数登場し、三種の神器だけでは対応不能に

リファインに  
なったら大変、、、

検証ツールを  
充実しなきゃ、、、



世代が進むとマスクコスト大幅にアップ (イメージ)



世代が進むと検証ツールも増加 (イメージ)

設計・検証に関わるものすべてがPDKに組み込まれるようになる

# PDKのたどった道

IDM (Integrated Device Manufacturer)

IDM全盛の「閉じた時代」

三種の神器

独自仕様のPDK

PDKは業界で主流の作法に寄り、  
形式・構造・使い方が次第に似てきた

国内協業

プロセス協業によってPDKの共通化が進む

各社バラバラの仕様のPDK

世界的に協業

新たな閉じた時代

1.0um 0.8um 0.6um 0.4um 0.25um 180nm 130nm 90nm 65nm 40nm 28nm 16nm 12nm 7nm 5nm 2nm

先端ノードでは協業とFoundryビジネスが進んだことで、PDKの形式・構造・使い方が“業界標準のスタイル”に寄っていった。

しかし、130nm 以前の世代は IDM 全盛期のため、各社が非公開の状態で独自進化 しており、  
現在オープン化が始まると **バラバラさが課題になる**。



# PDKを“読む”ならどこから？①

レイアウトを眺めるのが最も速くていろいろとわかる

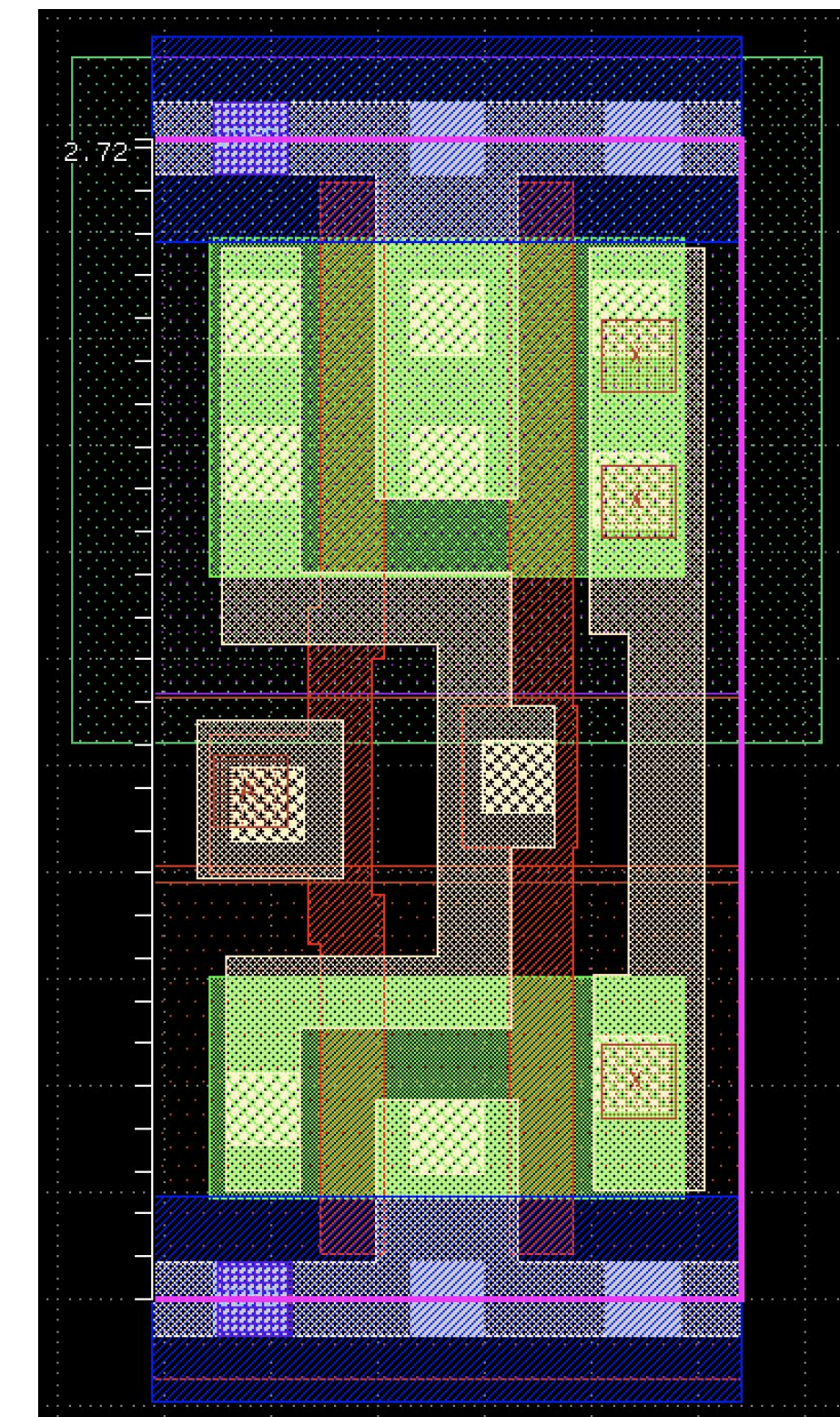
標準セル(スタンダードセル)から読み取れること

- レイヤー構成 → デザインレイヤーがわかる
- 最小ルール(長さ・幅・間隔・包含ルール)
- Metal ピッチと配線方向
- Poly 幅・ピッチ → 技術世代の見極め
- Well 構造・隔離構造

標準セルは多くが **ミニマムルールの塊** なので、読むだけでなく“**見ればわかる**”のが強み。

ただし、謎のレイヤーも出てくるので、ここは頑張って調べる必要があります。

本当はまずはデザインルールを読んで！なのですが、とにかく分かりにくい

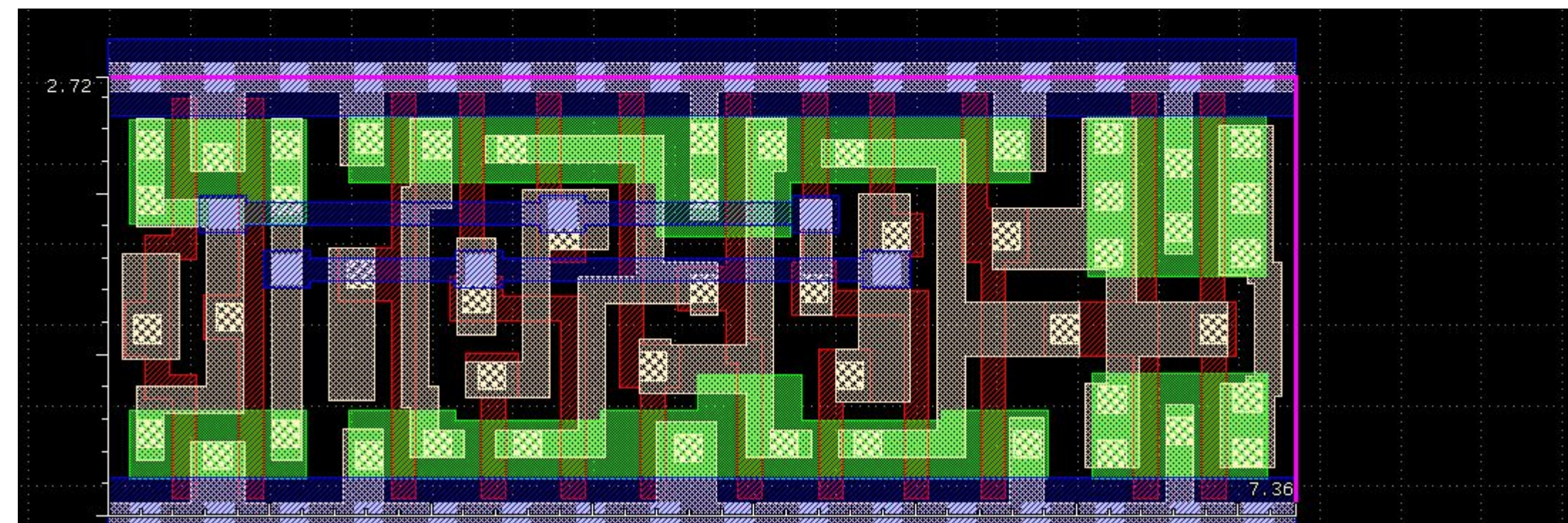




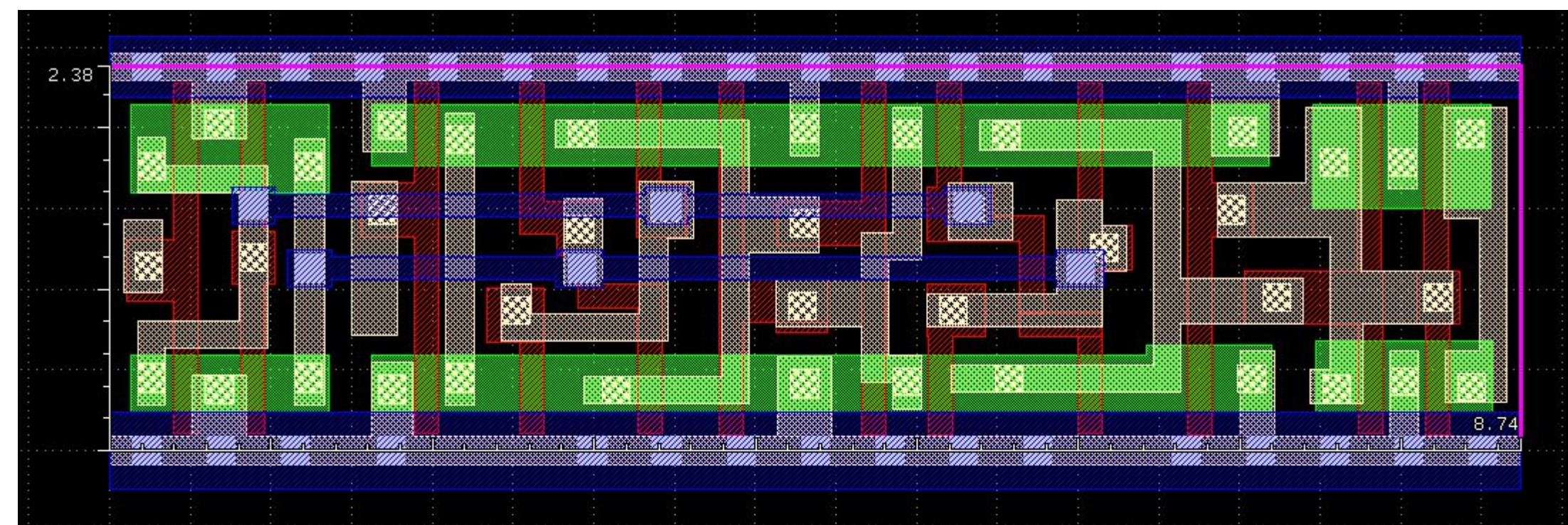
# 次に自分で描いてみる(スタンダードセル)

実際にレイアウトを描いてみると理解が飛躍的に深まります

デザインルールと睨めっこしながら描いてみる



8トラックセル



7トラックセル

こんなことやる人はいないと思いますが、、  
(やるにはDRCが必要、、)もっと追い込んでみる

例) 8T → 7T など、標準セルのトラック数を削ってみる

これにより、

- どのルールが本質的に効いているのか
- Poly／Active／Contact/Metalの本当の限界
- 配線層で“絶対に譲れない条件”
- デザインルールのクセ(描きやすさ・描きにくさ)

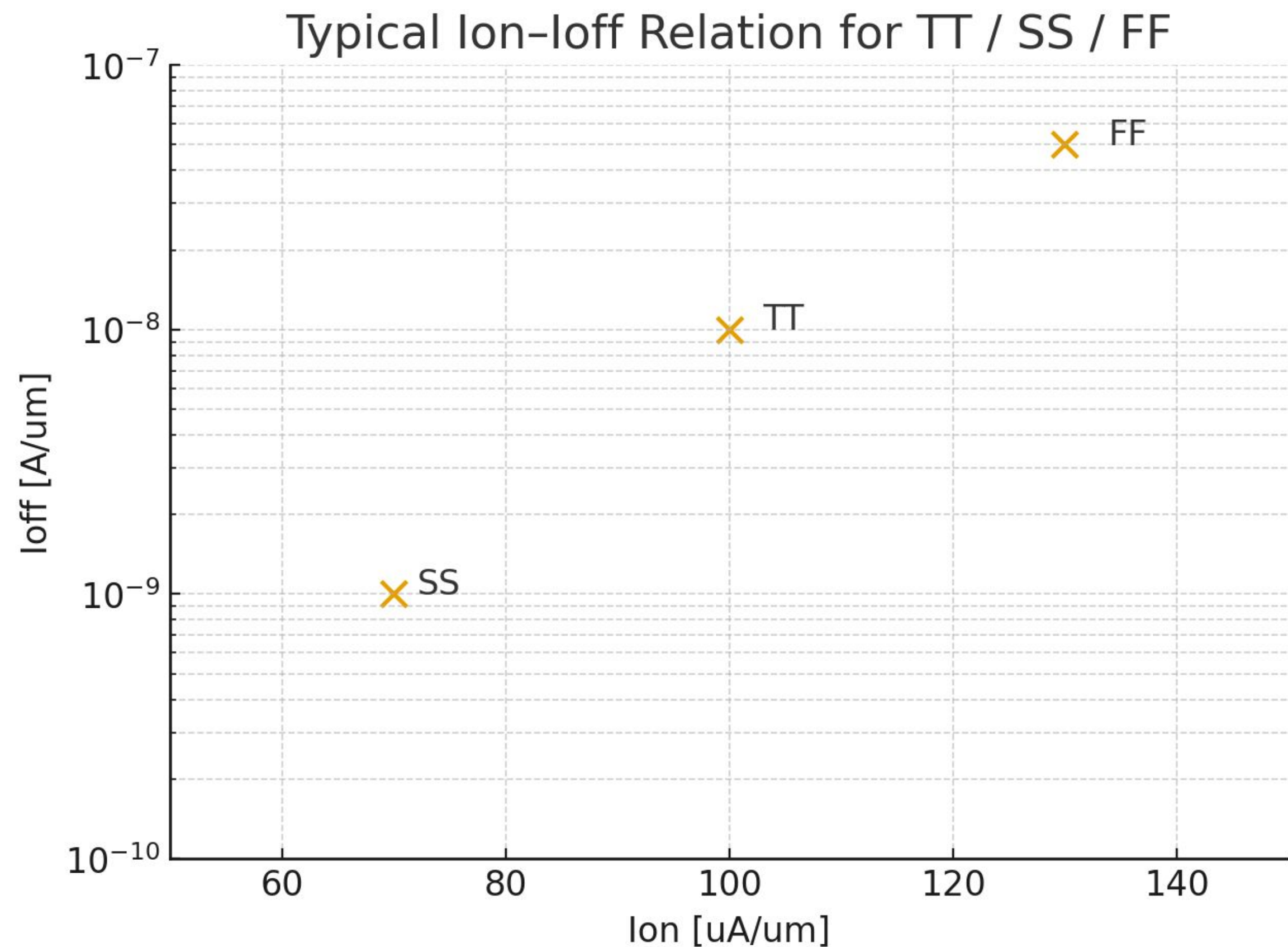
が手を動かした瞬間に見えてきます。

デザインルールの“思想”を理解するにはうってつけ。



# PDKを“読む”ならどこから？②

デバイスパラメータからデバイスの性格を読み取る



$I_{on}$  (オン電流) と  $I_{off}$  (オフ電流) の関係を見る。

このグラフから見える 3つの本質

①(デバイス) 高速性とリークのトレードオフが可視化される

FF は高速だがリーク大 → 電力課題

SS は遅いがリーク小 → 安定性と省電力重視

②(回路) 設計時の  $T_{pd}$ ・電力・動作周波数が推定できる

$I_{on}$  が大きいほど遅延は短くなる

$I_{off}$  が大きいほど動作点設計がシビアになる

③(プロセス品質)  $V_{th}$ ばらつきの窓が見えてくる

$I_{on}/I_{off}$  の差から、 $V_{th}$ のウィンドウがどれくらいか推定できる



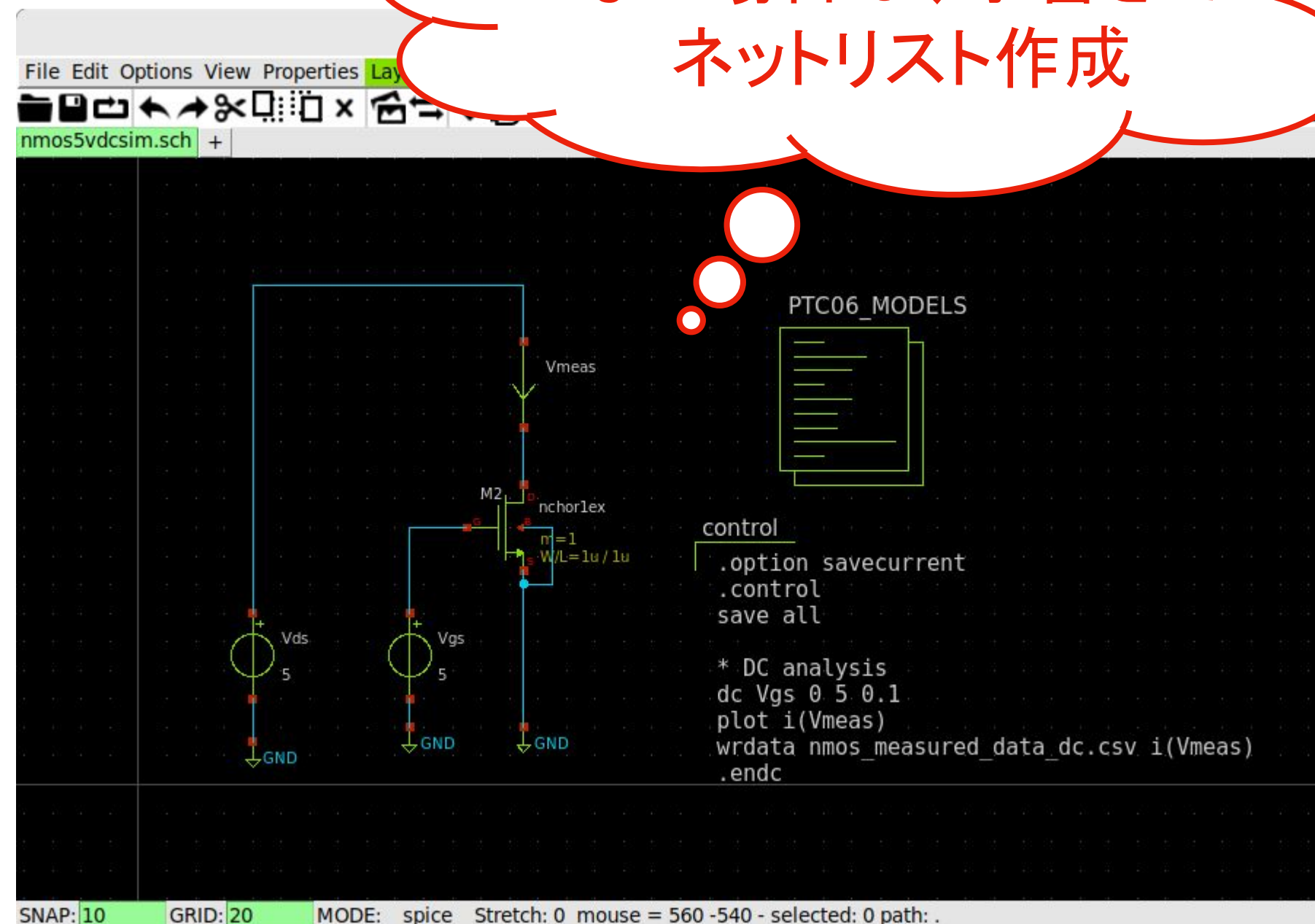
デバパラに載っていない場合は、SPICEと言う手もあるが、どこまで精度良く入れているかは??



# 次にシミュレーションしてみる

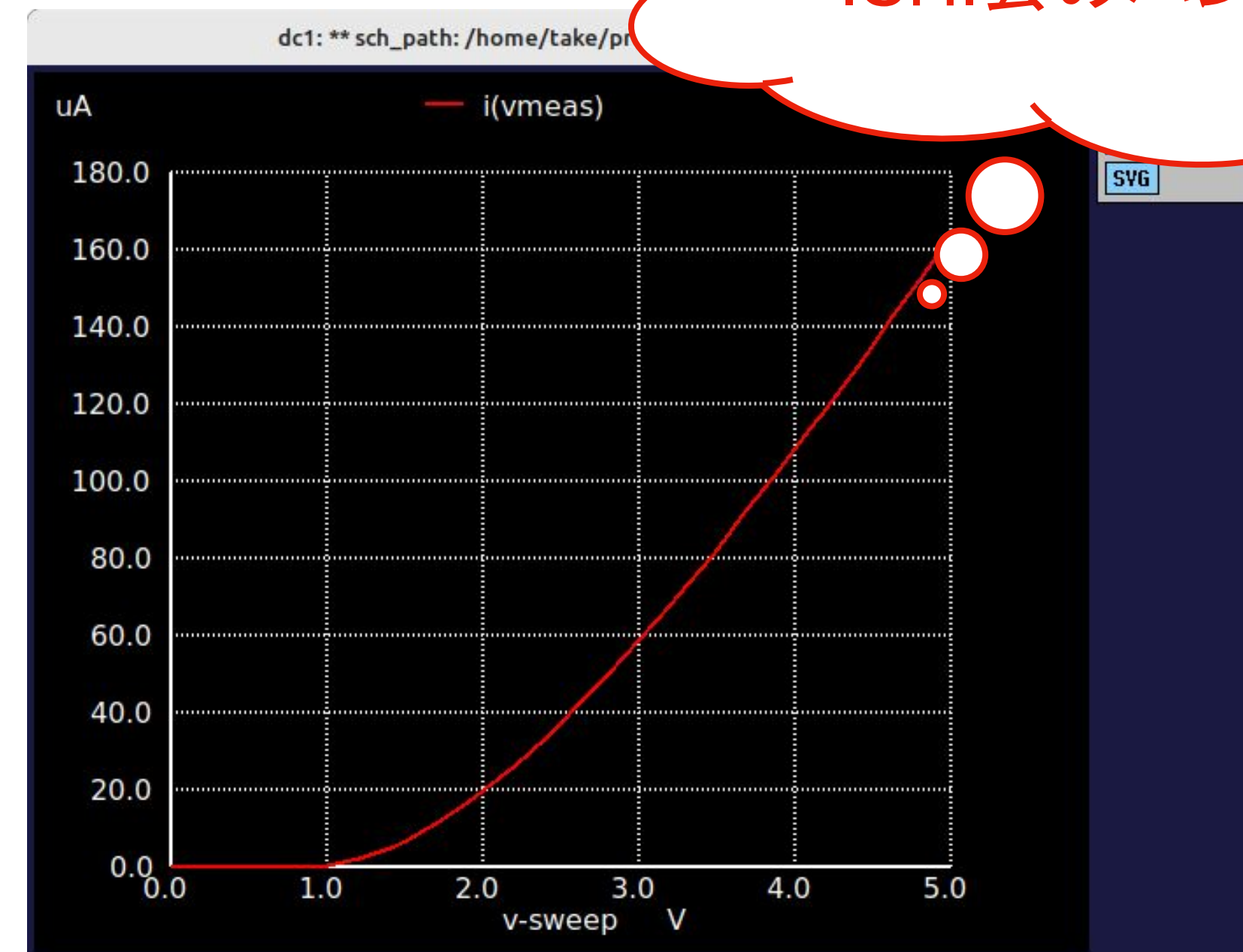
実際にSPICEシミュレーションするとデバイスの詳細が見えてくる

Symbolが準備されていない場合は、手書きで  
ネットリスト作成



I - V特性

とてもおすすめ  
ISHI会のハンズオン



シミュレーション結果

その他、スタンダードセルのスピード(Tpd)を測るのも有効(パワーvsスピード)

# このぐらいPDKを読んでおけばPDKの自作はしやすい

三種の神器(デバパラ・デザインルール・SPICE)があるなら、あとは

**DRC と LVS を整えて“ツールが使える PDK”に仕上げるだけ \*\***

- レイアウトが描ける → DRC が必要
- 抽出して比較できる → LVS が必要
- Symbol / Pcell は後からでよい

→ まずは設計ツールが解釈できる最小PDKを作る



# PDKを“自作”するなら何から？

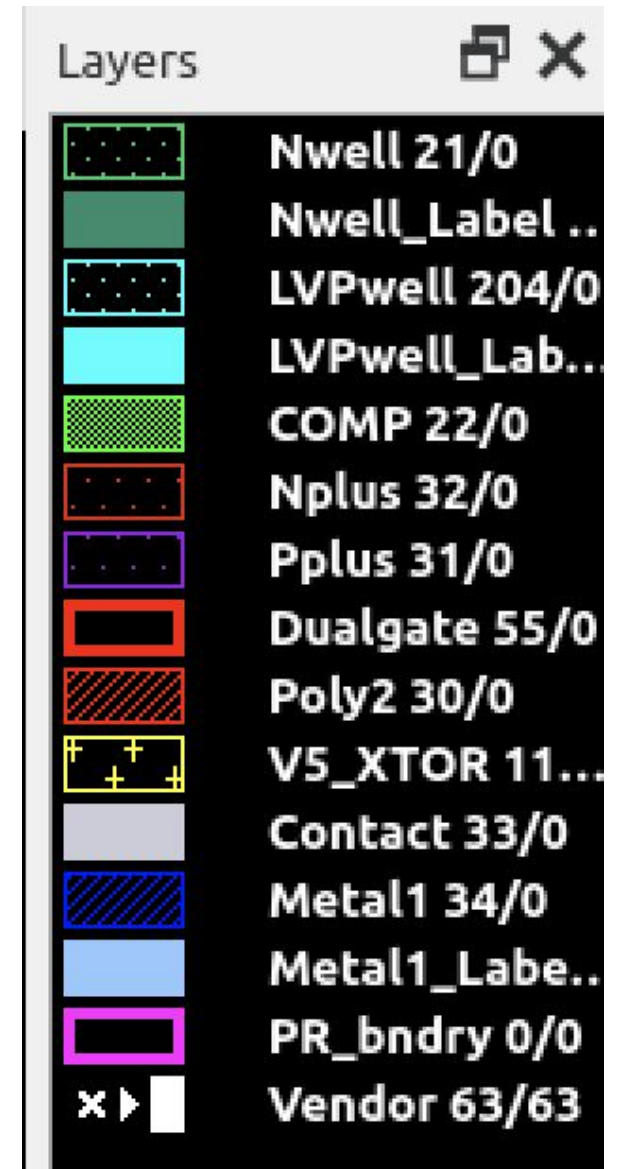
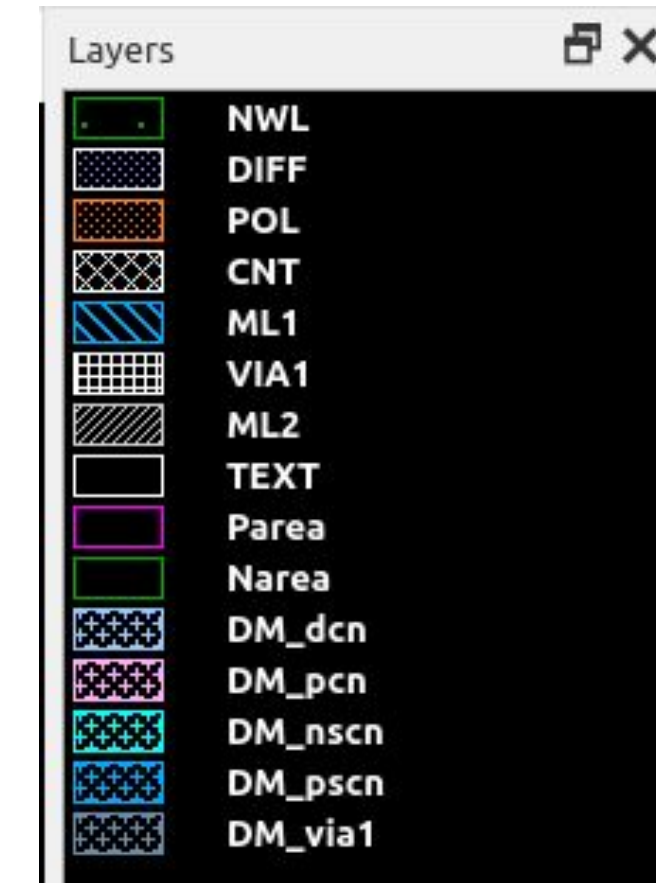
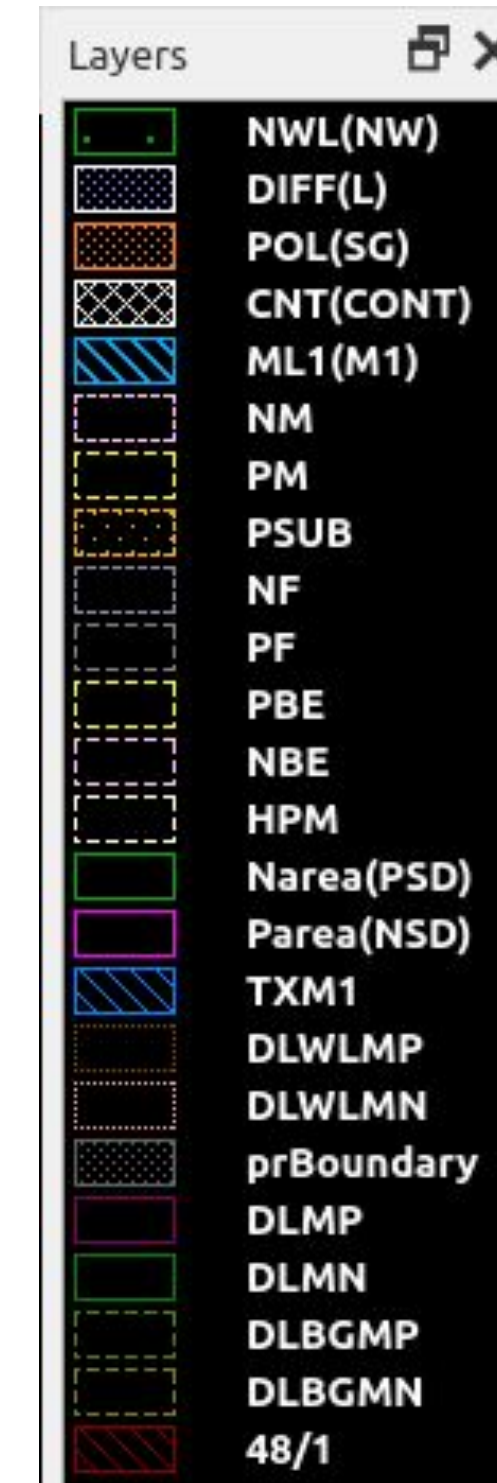
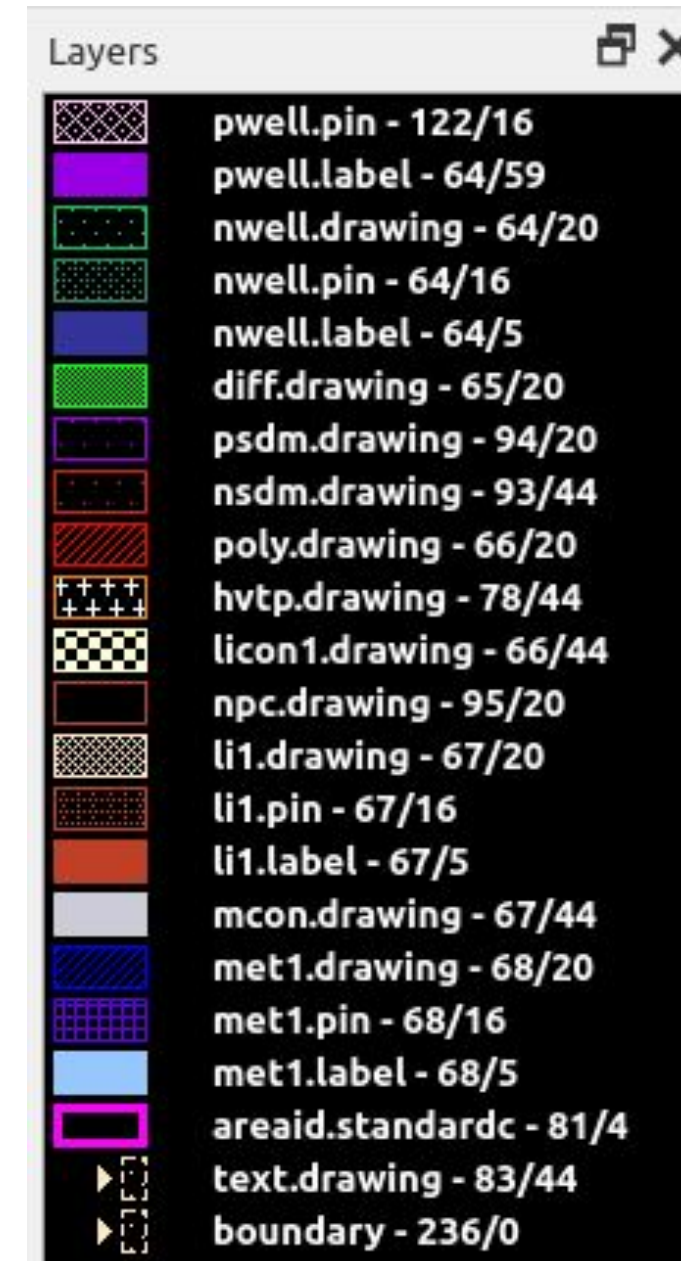
## レイヤ構造の最小セットを決める

### ① レイヤ定義 (Layer Map)

- レイヤ名・番号・色・GDSマップを決める
- デザインレイヤとマスクレイヤの対応を定義

まず必要なのは次の“最小構成”:

- 拡散層 (Active Area)
- N-WELL / P-WELL
- ゲート (POLY)
- N-/P- LDD implant
- N+/P+ S/D implant
- コンタクト
- 配線メタル (Metal1～)
- VIA (Via1～)



各社独自仕様

PDKの“世界観”がここで決まる(こだわりたいところ)

# PDKを“自作”するなら何から？

この項目はパラって  
作る事もできるフェーズ

続いて、DRC/LVSと検証用にスタンダードセル

## ② 最小DRCルール(WIDTH / SPACE / ENCLOSURE)

- 一般的なコマンドで構成する
- INV / NAND / DFFが描ければO.K
- 描けるPDKであるかの最初のチェックポイント

## ③ LVS抽出ルール(DEVICE / CONNECT)

- MOSデバイスの抽出(W/L取得)
- 接続(METAL / CONTACT)の抽出
- SPICEモデルと構造の整合が必要

## ④ 基本スタンダードセルのレイアウト

- レイヤ定義＋最小DRC＋LVSが機能するか  
実際に描いて確認
- INV / NAND → ミニマムDRCの妥当性
- DFF → 実用レベルのセルが描けるかの最終確認

## ● ⑤ Symbol / Pcell(拡張・利便性向上)

PDKを“使いやすく”するための要素

- Symbol: 回路図入力を容易にする
- Pcell: MOSなどのデバイスをサイズ指定で生成

ここまで出来れば以降は、誰もがどんどん拡張できる



# オープンソース PDKで遊ぼう

## 仕事じゃないPDKの楽しみ方

- ・ 半導体会社のPDK開発メンバー(SIGの遊び場)が、“仕事じゃない時間”にオープンソースPDKに触って遊んでいます。

### 仕事のPDK開発 vs SIGの遊び場

- ・ 納期とスケジュール → 思いついたときに動く
- ・ 正確さ・再現性が命 → 試して壊して笑う
- ・ 社内ルールに縛られる → 自分ルールでOK
- ・ 成果報告と検証 → 雑談と共有



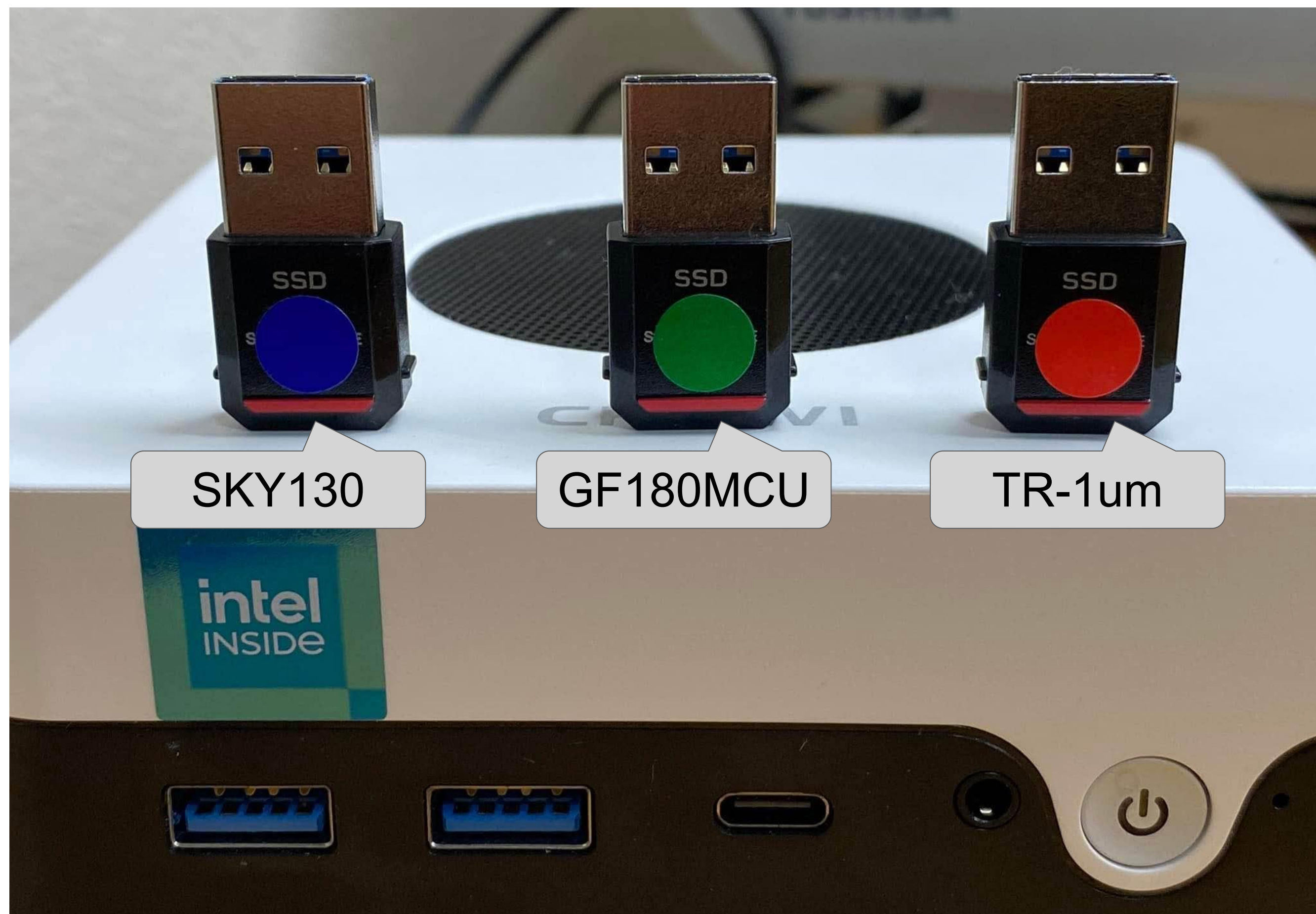
“ちゃんと動かす”より、“とりあえずやってみる”をモットー

オープンソースPDKに触ることで、仕事以上にPDKの理解が深まる（??）



# これまでオープンソース PDKで遊んできた事

ミニPCとミニSSDで設計環境を作って遊ぶ



ミニPC+ミニSSD

格安ミニPCを買って、ミニSSDにLinux (Mint) と各設計環境 (PDK) を入れて、差し替えて楽しむ。

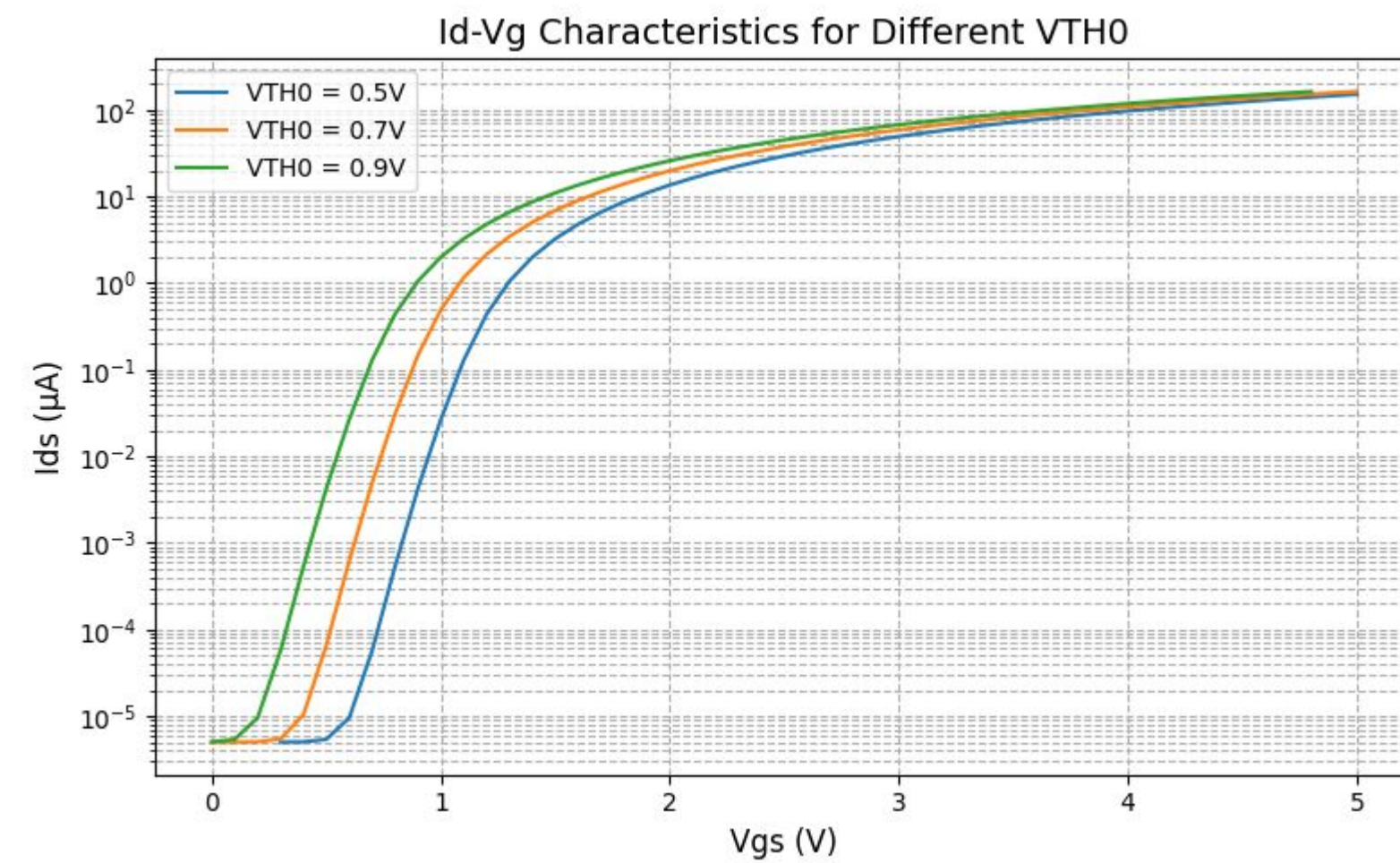


ノートに挿せば外出先でも



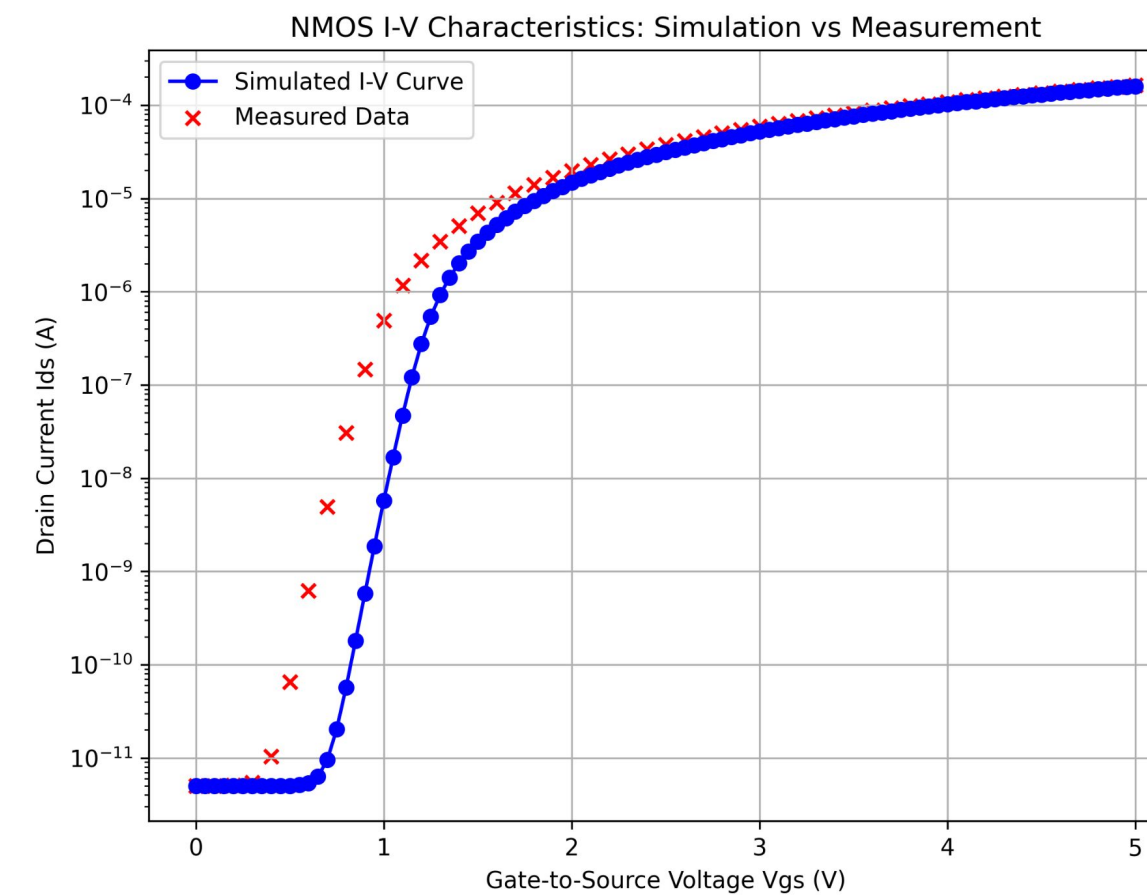
# これまでオープンソース PDKで遊んできた事

## SPICEモデルパラメータをいじって遊ぶ

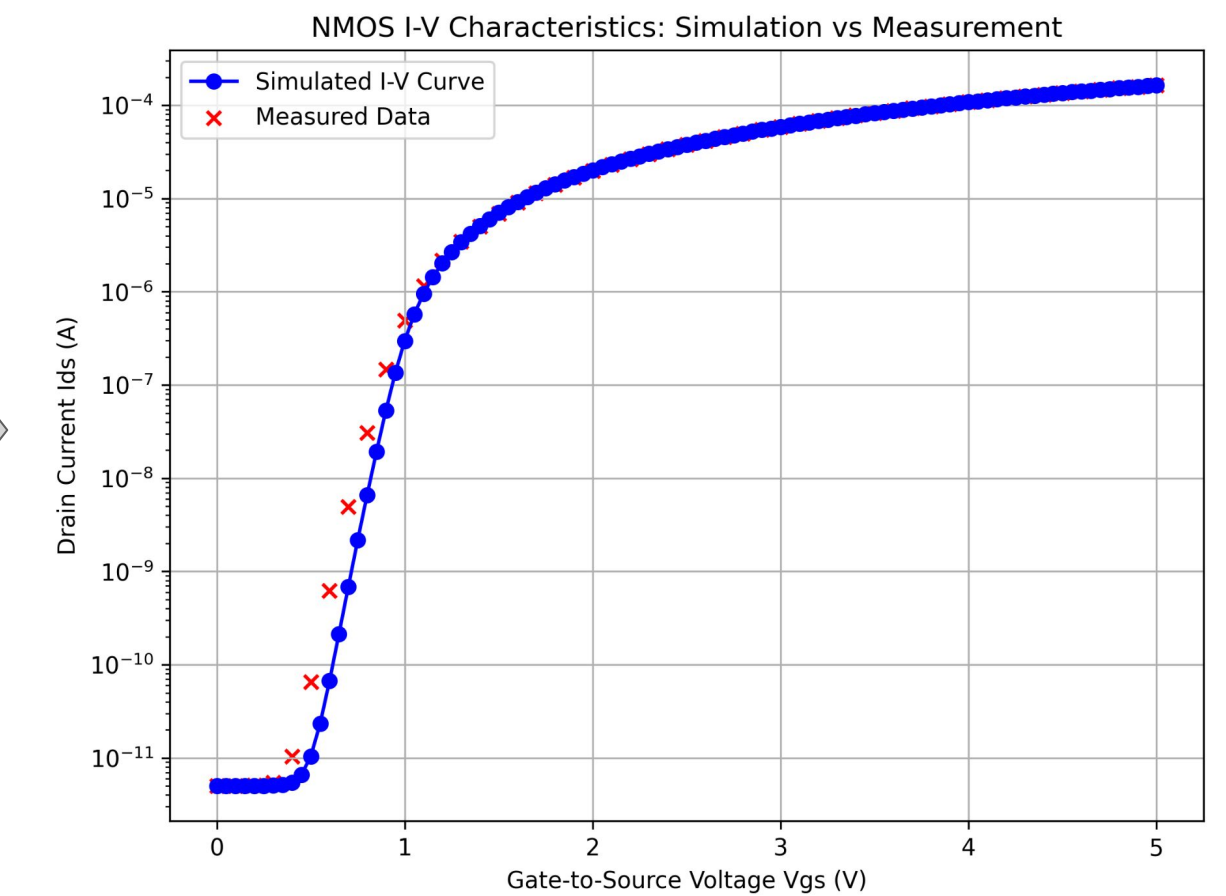
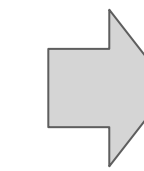


Vth0を変えてみる

いろいろとモデルパラメータを  
変えるとどうなるか シミュ  
レーションしてみた



イニシャル

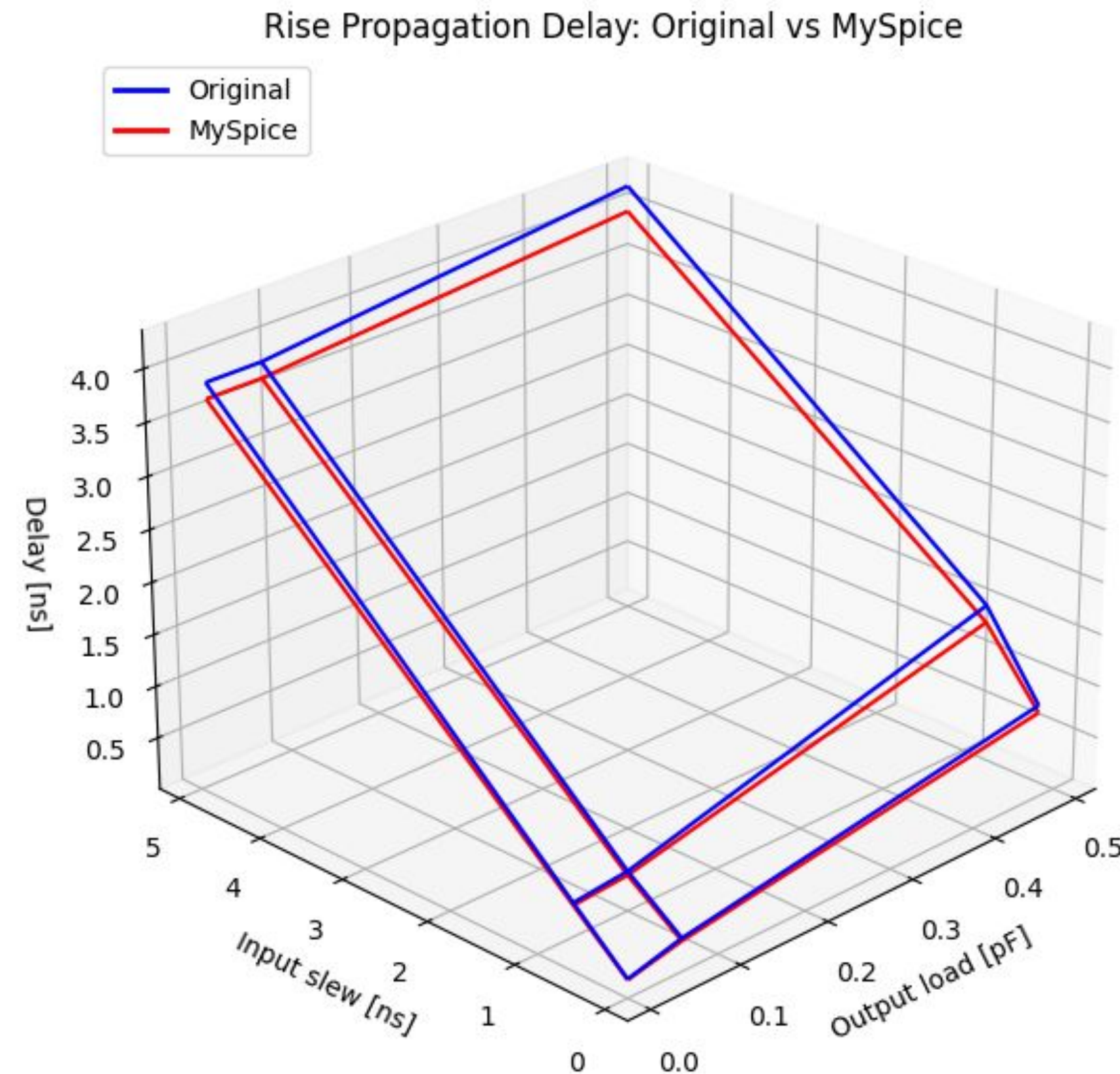


フィッティング

仮想測定値とSPICEシミュレーション結果のズレ  
が最適になる $V_{TH0}$ と $U_0$ を探索させてみた(自作モ  
デル? 作成)

# これまでオープンソース PDKで遊んできた事

## キャラクタライズツールLibrettoで遊ぶ



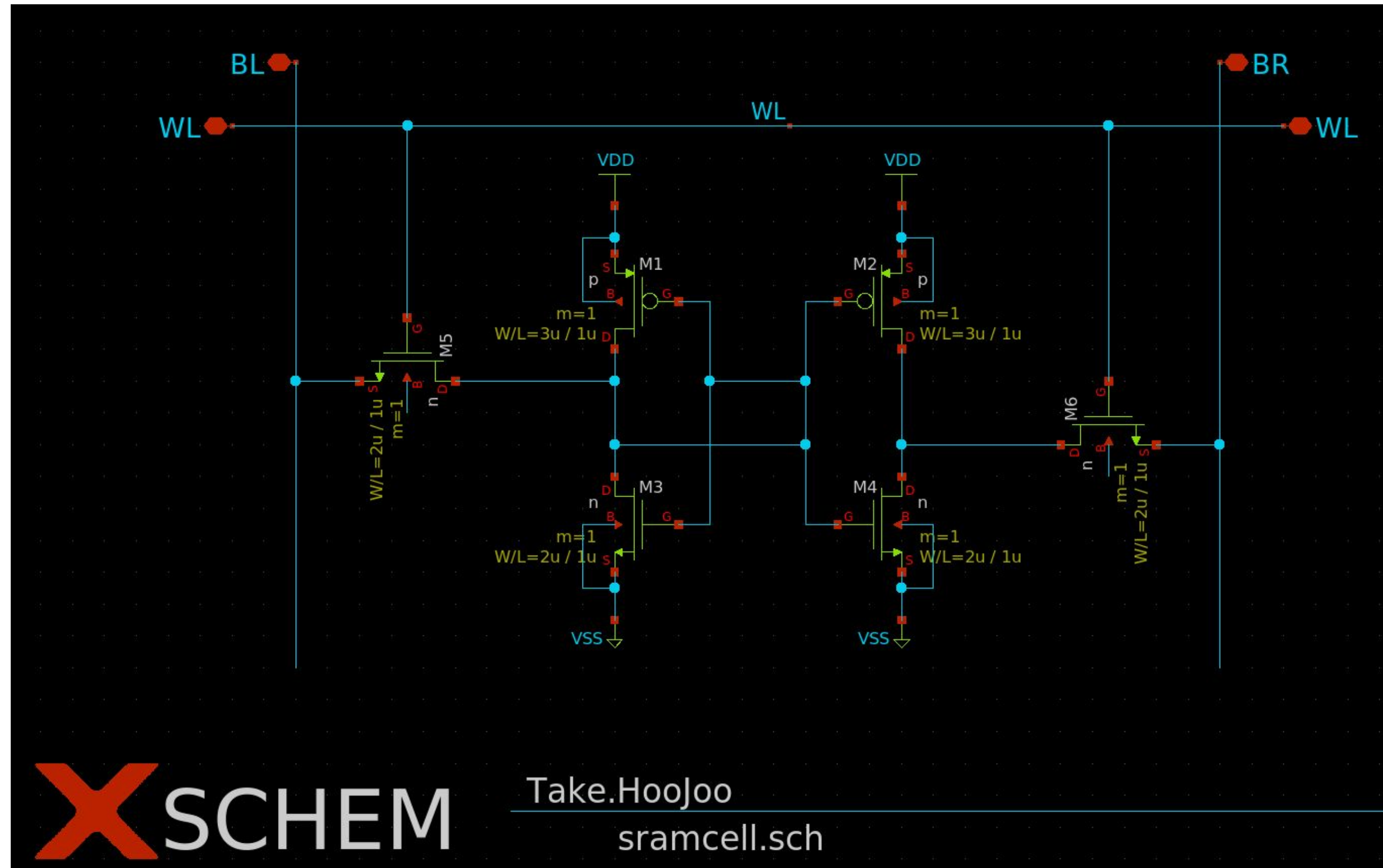
キャラクタライズツールLibrettoを使い自作モデルを使ってキャラクタライズしてみた

キャラクタライズ結果をオリジナルモデルと自作モデルで比較してみた

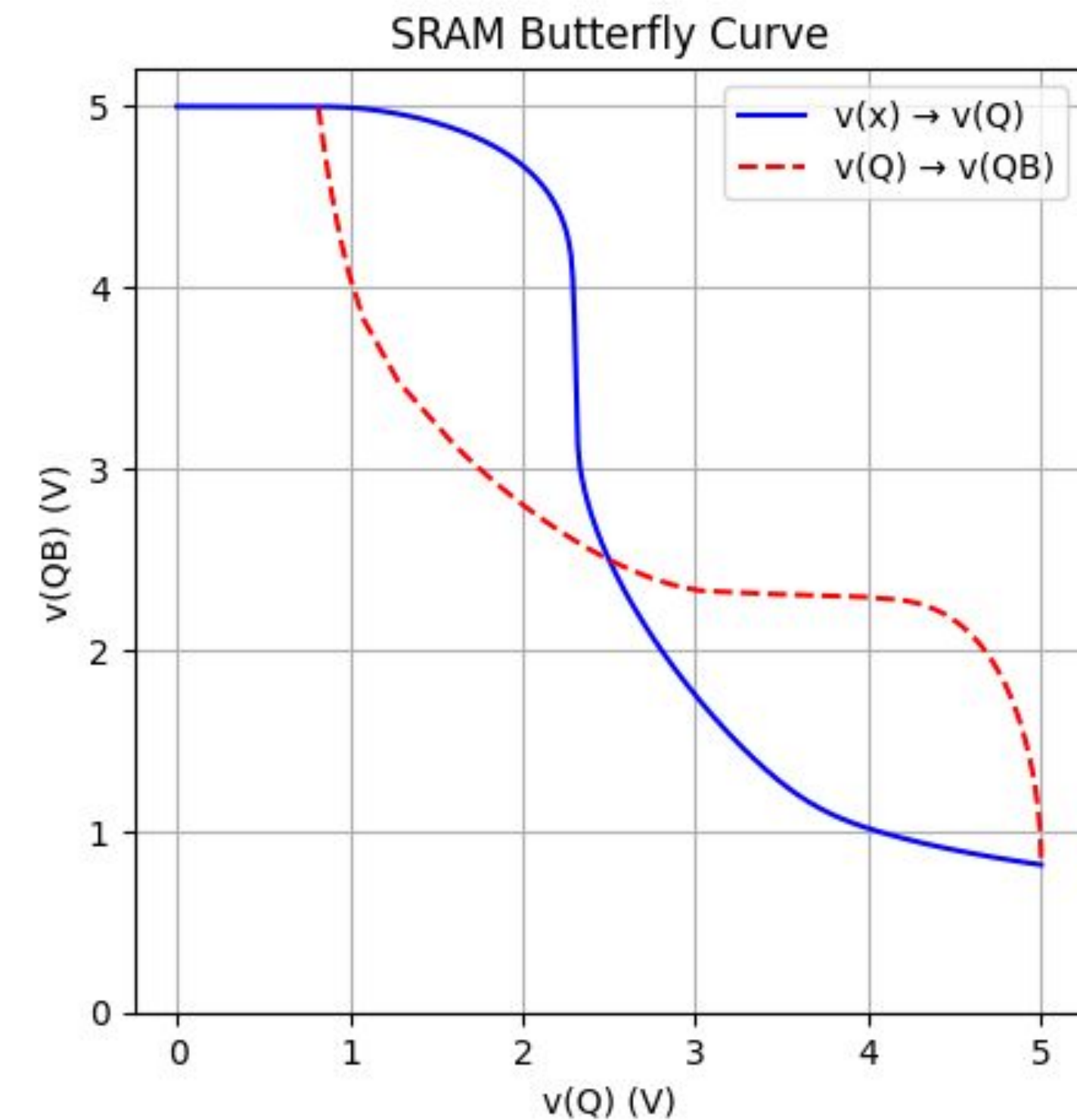


# これまでオープンソース PDKで遊んできた事

SRAMのビットセルでメガネ特性を描いて遊ぶ



SRAM 6T ビットセル



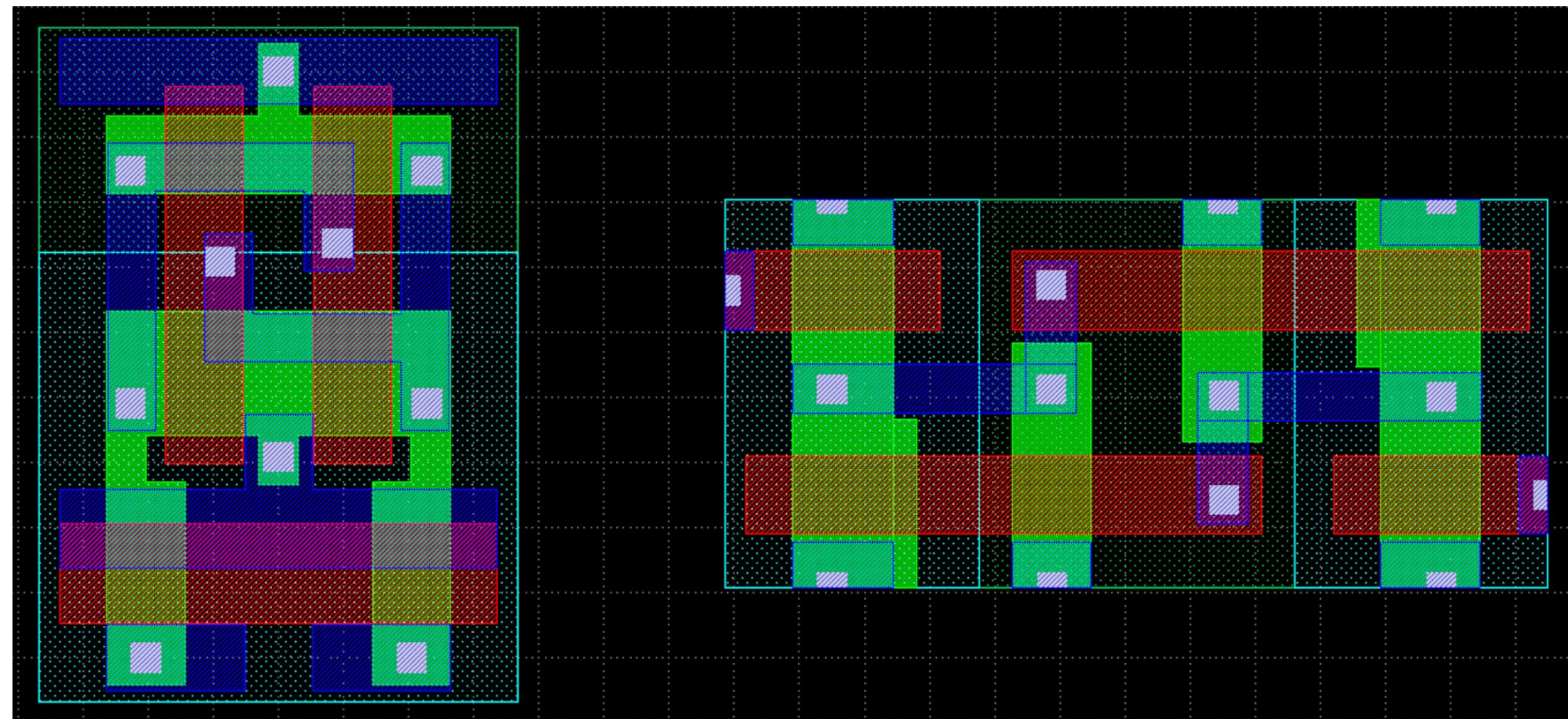
メガネ特性

SRAMビットセルの安定性SNM (Static Noise Margin) の評価してみた



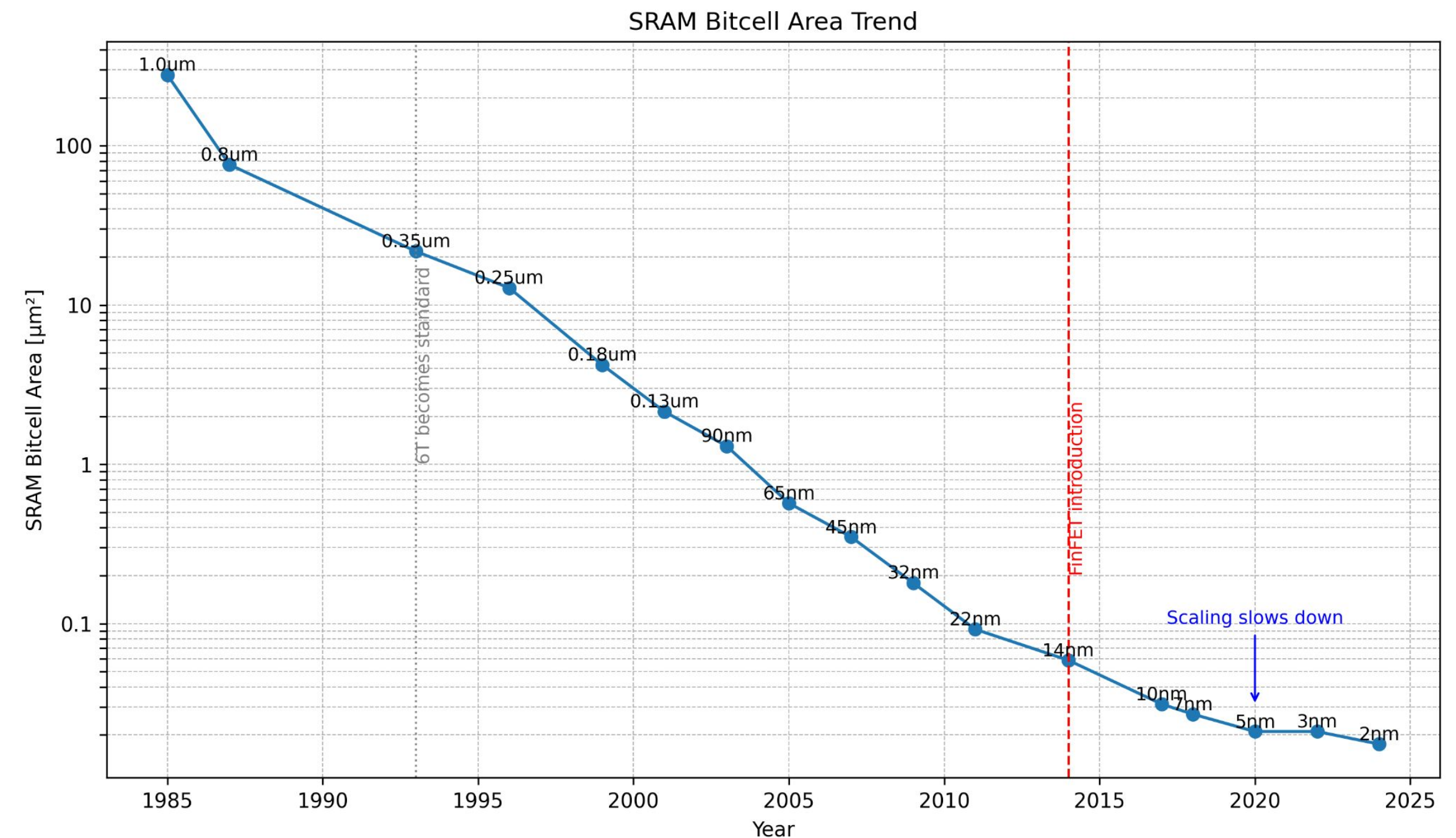
# これまでオープンソース PDKで遊んできた事

SRAMのビットセルをレイアウトして遊ぶ



縦型セル

点対称型セル



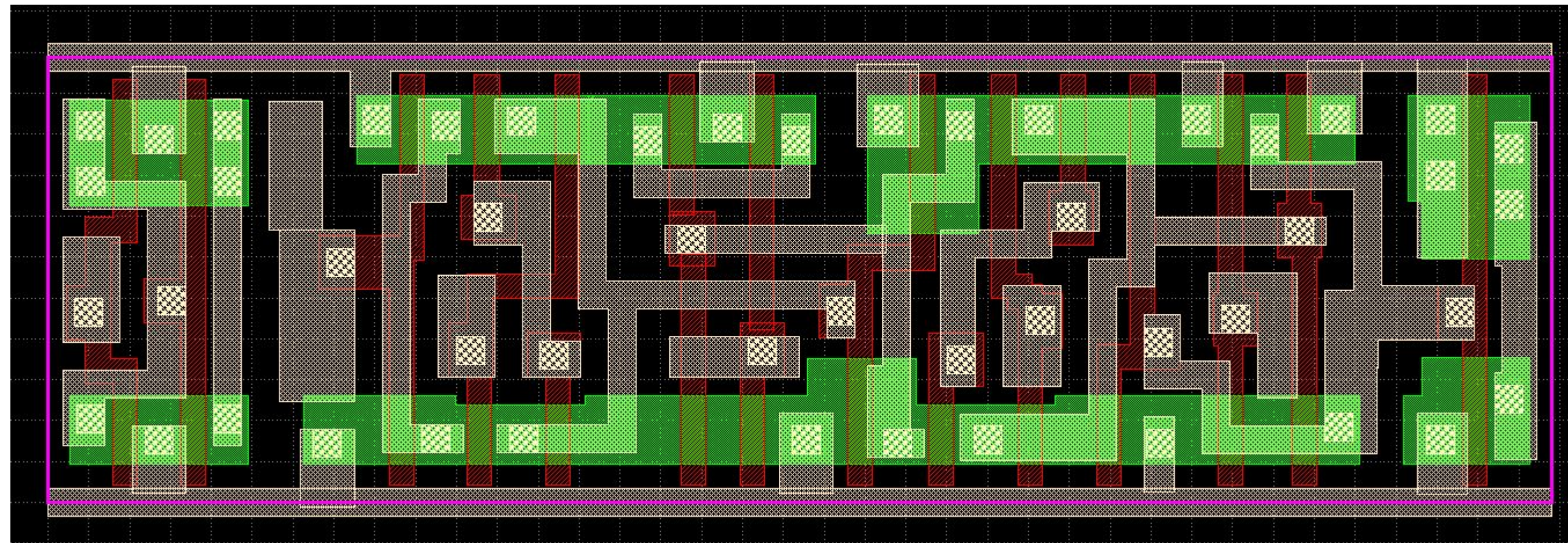
ビットセルトレンド

SRAMビットセルを縦型セルから点対称型セルにしてみた

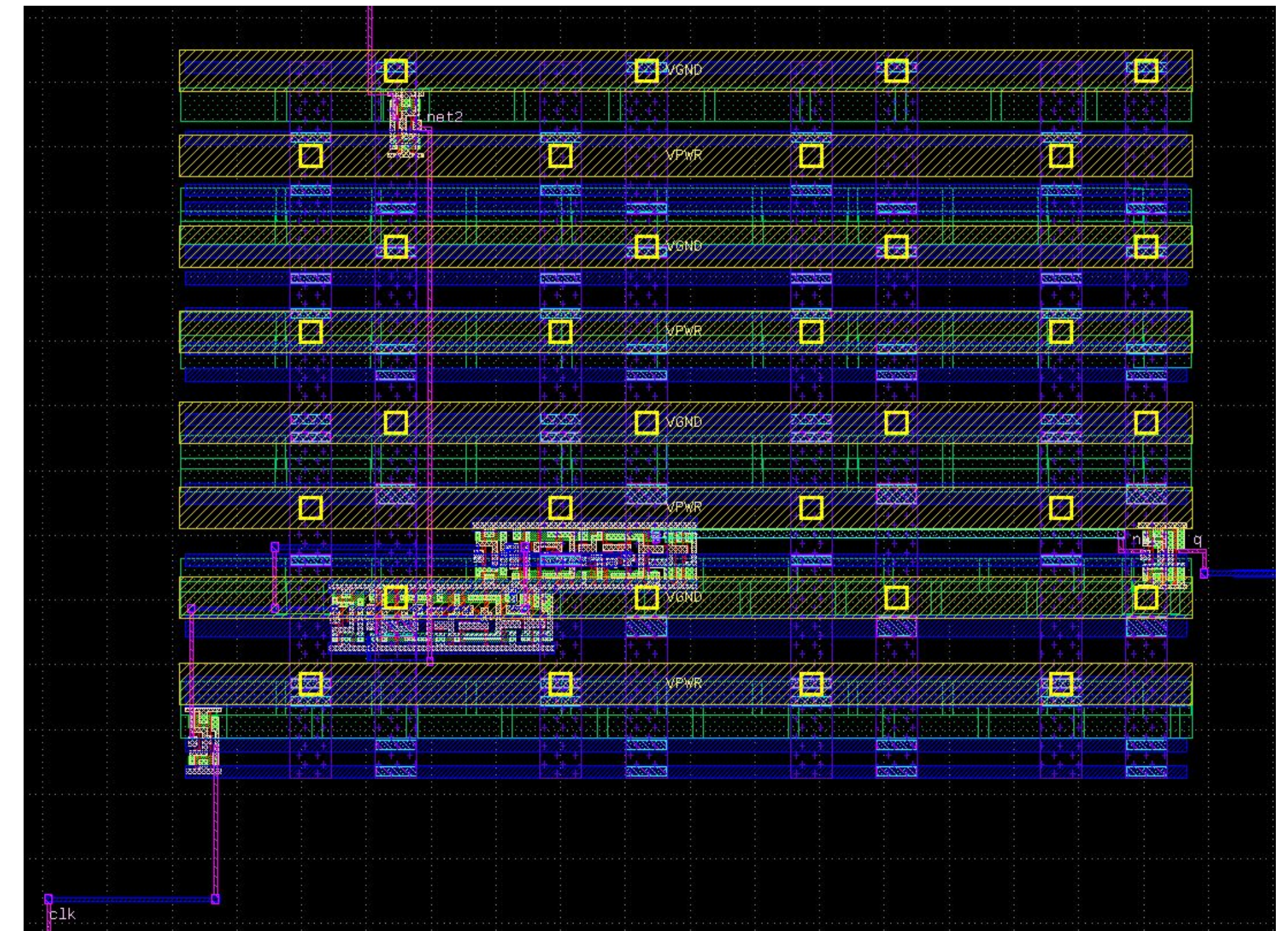


# これまでオープンソース PDKで遊んできた事

自作7トラックセルでOpenLaneのフローを流して遊ぶ



自作7トラックセル



自作7Tセルを使った配置配線結果

7Tセルのgds・LEF・.v・.libを作ってOpenLaneフローを流してみた



# 遊び続けるために

ここからは、後のPDK対談ネタかもしれませんが

オープンソースPDKでいろいろなことができるようになり、  
とても楽しくなってきました。

ただ、この先も 長く遊び続けていくためには、

あらかじめ 考えておいた方がよさそうな点 もいくつかあります。



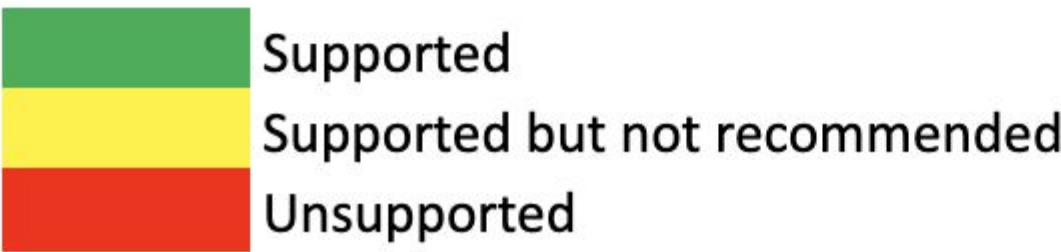
# PDKは意外と短命

PDKはメンテナンスなしでは生き残れない

PDKはプロセスの更新だけでなく、

- ・OS
- ・EDAツール
- ・検証フローの変化にも追従が必要。

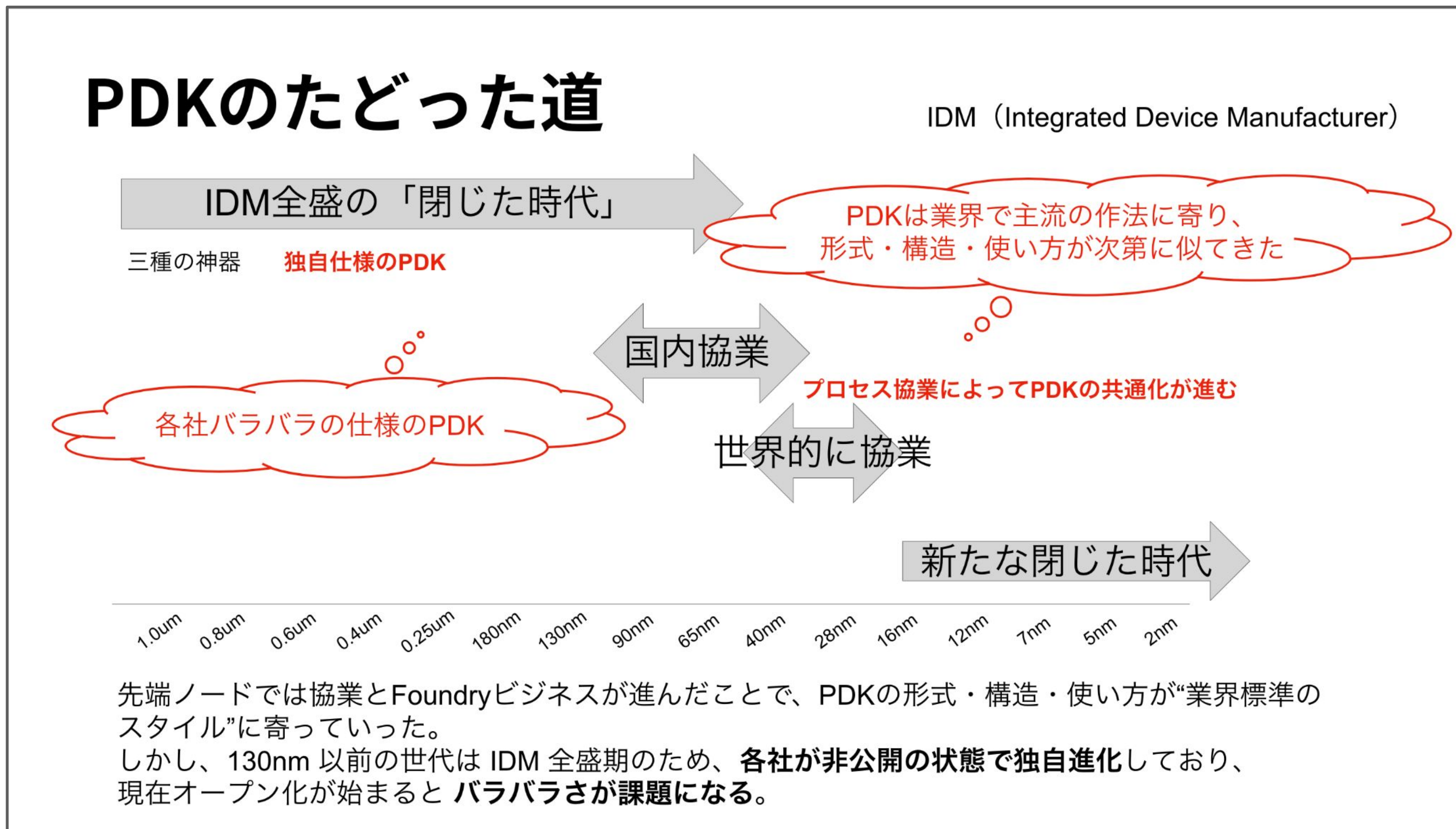
	RHEL 6.5	RHEL 6.1	RHEL 7.2	RHEL 7.6	RHEL 7.9	RHEL 8.2	RHEL 8.6
2015							
2016							
2017							
2018							
2019							
2020							
2021							
2022							
2023							
2024							
2025							



そうだとすると、メンテナンスしやすいPDKにしておく必要がある

# 独自仕様の PDK がオープン化

いろいろあると、ユーザ視点からは使いこなすのが大変。(メンテナンスも)



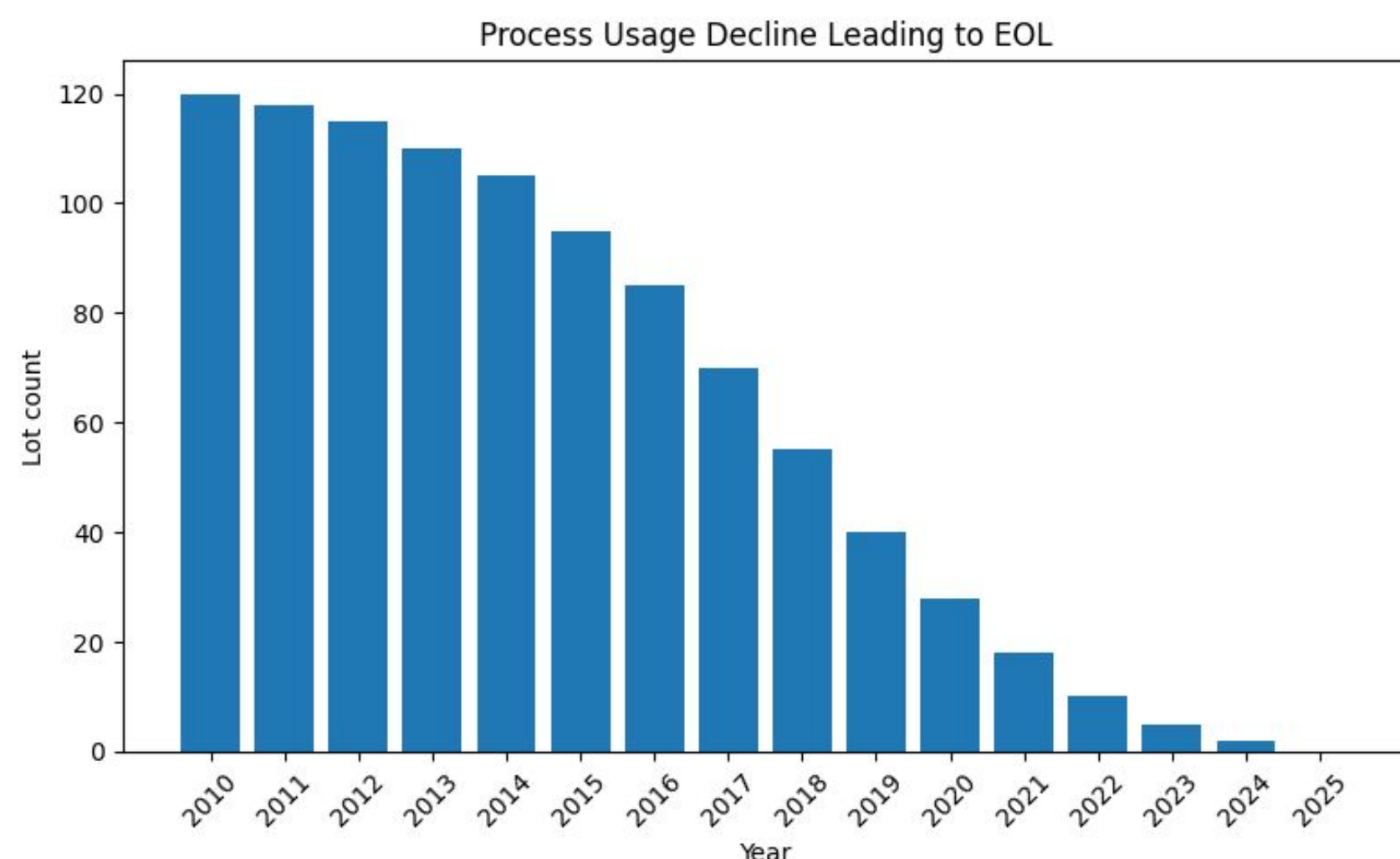
標準化を考える必要は？

このままでは、オープンソースPDKは“使える人が限られる”世界になる。



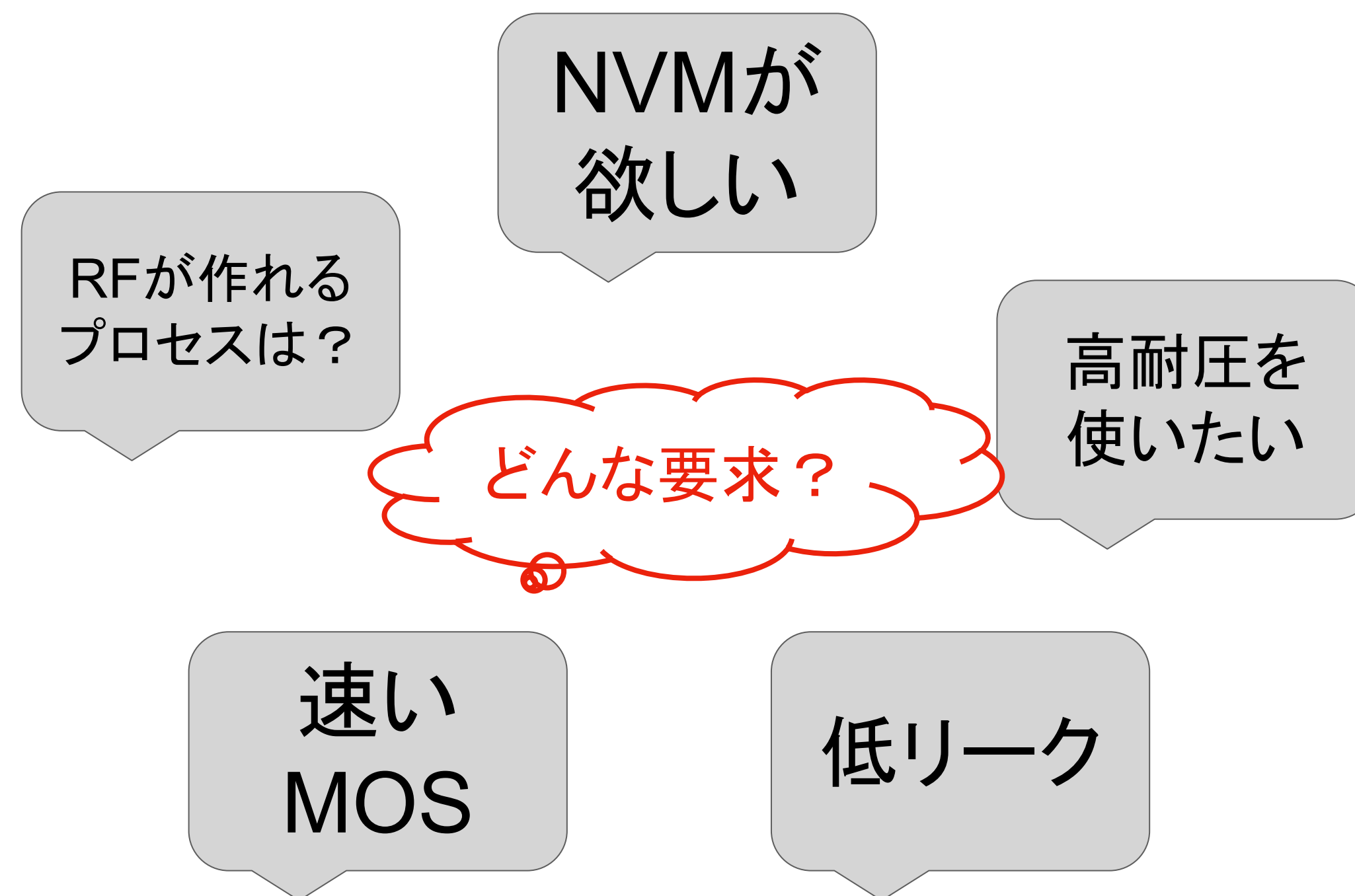
# そのプロセスの寿命は？魅力は？

プロセスは使われなくなるとEOL。オープンソースで使うプロセスは？



## 図が示していること

- プロセスは **量産ロットが多い間は維持・改善される**
- 年数とともにロット数が減少
- 一定水準を下回ると  
→ **設備維持・人材・PDKメンテナンスが成り立たなくなる**  
結果として **EOL (End of Life)**を迎える



魅力あれば、使われる。  
どんなプロセス？

# まとめ

## ① 三種の神器があれば、PDKは作れる

デバイスパラメータ・デザインルール・SPICEモデルが揃っていれば、DRC / LVS を中心に、設計ツールで使える**PDKを構築することは可能**。

## ② PDKは“作って終わり”ではない

PDKは時間とともに変化するため、継続的なメンテナンスが不可欠。オープンソースPDKが増えるほど、**メンテナンスしやすい構成・作り方を考えることが重要** になる。

## ③ この楽しみを続けるには、プロセスも重要

PDKを長く使い続けるためには、現在も量産され、今後も改善・拡張されていく **“魅力的で成長するプロセス”**であることが必要。

こんな事も考えつつ

オープンソース PDKで誰もが半導体設計を楽しめる世界へ