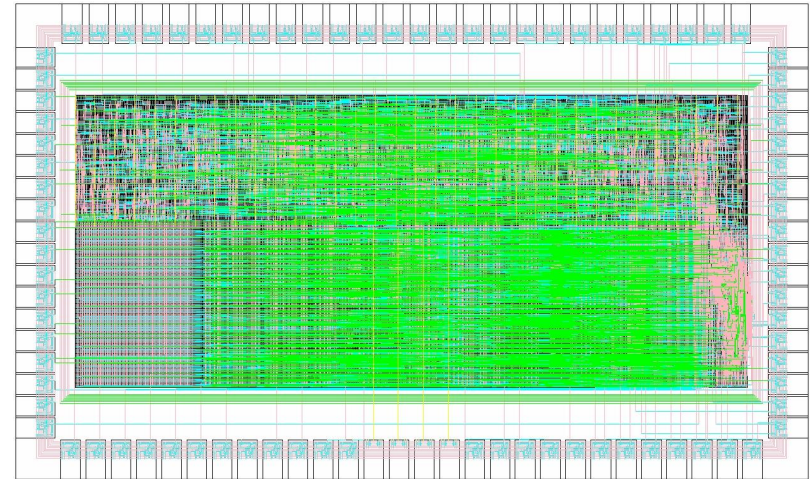


# MOCCA

## circuits numériques

Flot de conception  
Dessin des masques  
Dessin de cellules  
Outils Alliance

## Exemple : Processeur MIPS 32 bits



Master SESI / MOCCA / Dessin de cellules

Franck Wajsbürt 3

## Avertissement

On ne s'intéresse qu'à une classe particulière de circuits,  
Les circuits intégrés CMOS numériques synchrones :

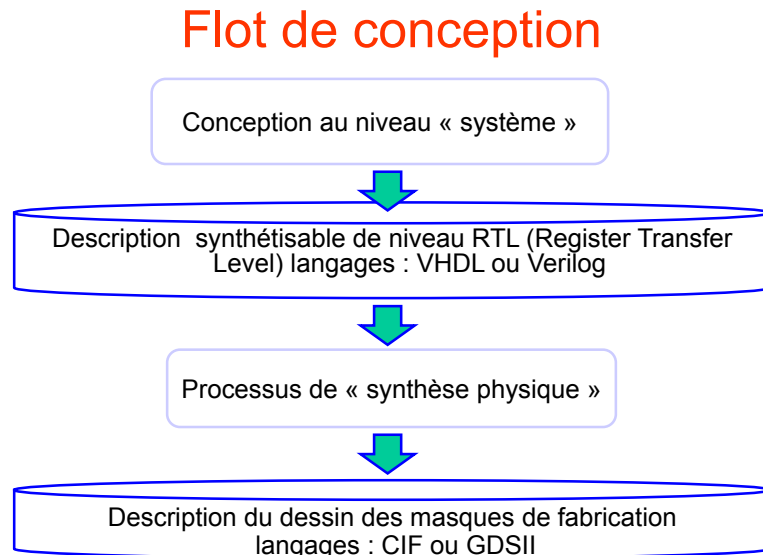
### – Numériques

Les signaux d'entrée et de sortie, ainsi que les variables internes sont représentées par des variables Booléennes ne pouvant prendre que des valeurs discrètes : 0 ou 1 (U, X, ...).

### – Synchrones

Les valeurs stockées dans les registres internes du circuit ne peuvent être modifiées qu'à certains instants, définis par les fronts d'un signal particulier, appelé signal d'horloge.

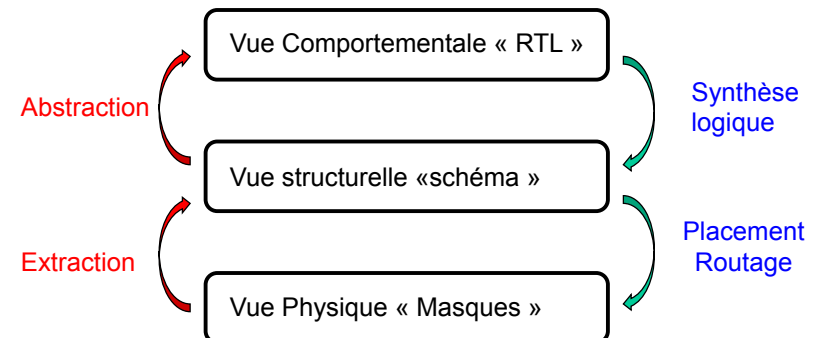
# Flot de conception



# Les trois vues d'un circuit intégré

- Dans la phase de synthèse physique, on peut décrire le composant à trois niveaux d'abstraction :
  - vue comportementale (« behaviour ») : Permet de simuler et d'analyser le comportement, mais ne décrit pas la structure interne du composant modélisé.
  - vue structurelle (« net-list ») : Décrit la structure interne, c'est à dire la façon dont le composant peut se décomposer en une interconnexion de composants plus simples.
  - vue physique (« layout ») : Décrit le dessin des masques de fabrication utilisés pour graver le silicium.
- Le processus de conception consiste à transformer progressivement la description comportementale en une description physique (utilisable par le fabricant de circuits).
- Les outils de synthèse physique permettent d'automatiser cette transformation.

# Les trois vues



## Dessin des masques

technologies  
règles de dessin

- Principe du procédé de fabrication CMOS
- Dessin réel
- Dessin symbolique sur grille lambda
- Les règles symboliques Alliance

## Quel est le problème ?

- Un circuit numérique est constitué principalement de transistors reliés par des fils conducteurs gravés sur un substrat.
- Tous les éléments doivent respecter des règles de positionnement (taille, distance, recouvrement, densité...)
- Chaque technologie (process) impose ses propres règles.
- Un nouveau process est créé tous les 18 mois.
- Le dessin de circuit est une activité très délicate et chacun essaie de pérenniser le résultat pour ne pas avoir à tout refaire à chaque fois.

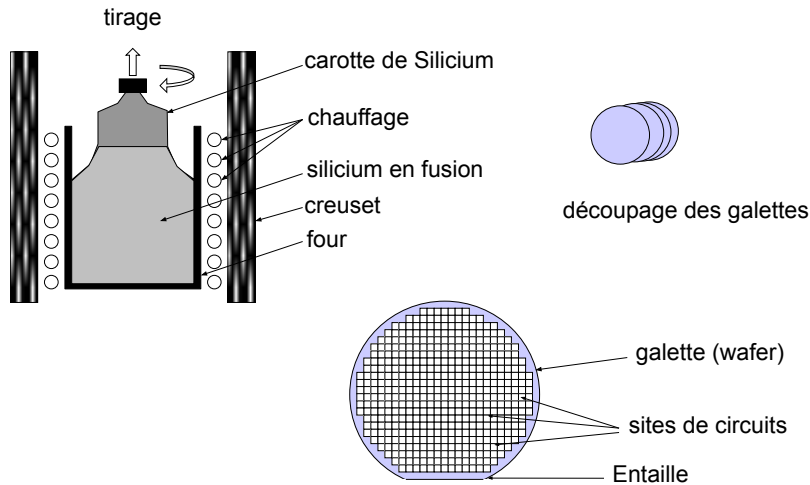
## Procédé de fabrication CMOS

### Structure en couches

- Un circuit intégré est composé d'un assemblage de couches, qui peuvent être :
  - semi-conductrices : pour former les transistors ;
  - métalliques : pour relier les transistors entre eux ;
  - isolantes pour séparer les couches semi-conductrices ou les couches métalliques entres-elles.
- Les couches sont déposées suivant un ordre précis grâce à un procédé photolithographique.
- Le substrat peut-être semi-conducteur ou isolant.

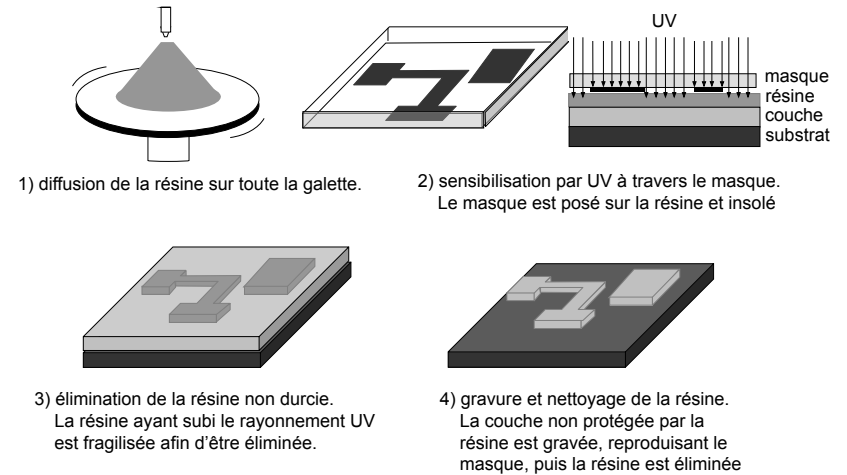
# Procédé de fabrication CMOS

Une carotte pure à 99.999999 %



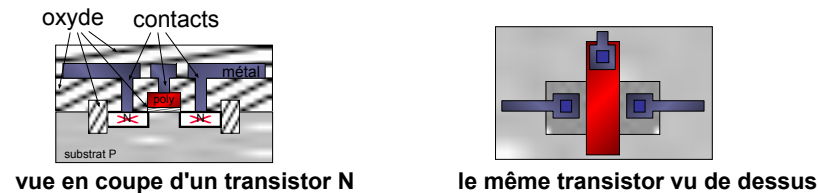
# Procédé de fabrication CMOS

Séquence photolithographique à résine positive



# Procédé de fabrication CMOS

Une construction en 3D

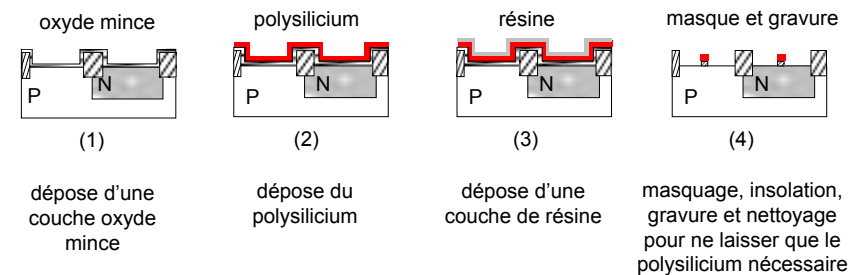


On remarque les zones d'oxyde qui isole les transistors entre eux, qui isole la grille du substrat, qui isole le métal du silicium, l'oxyde est percé pour réaliser les contacts.

C'est ainsi que le dessine le concepteur en superposant des rectangles de couleurs, chaque couleur représentant une couche différente.

# Procédé de fabrication CMOS

Création du polysilicium



La création des grilles avant celle des drains et sources permet l'auto-alignement du transistor !

# Procédé de fabrication CMOS

## Liste des niveaux

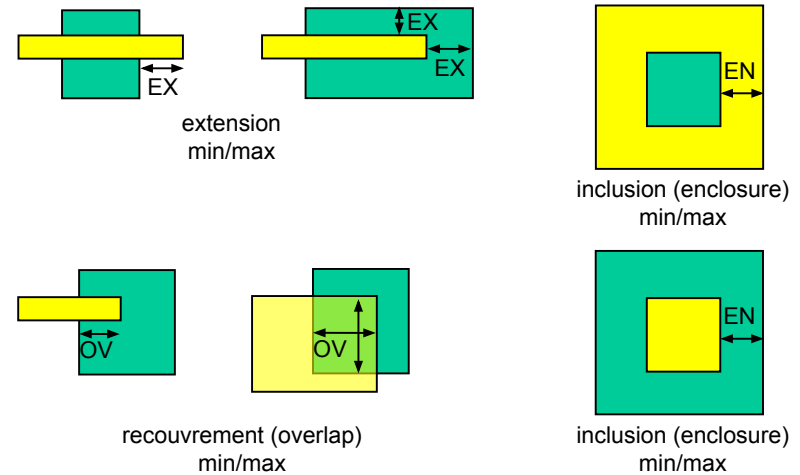
- Le fondeur attend une série de masques permettant la création de chaque couche :
  - caisson N (éventuellement caisson P)
  - zone active
  - polysilicium
  - implantation N
  - implantation P
  - cuts metal1 (trous metal1 vers caissons, polysilicium ou implantations)
  - metal1
  - cuts metal2
  - metal2
  - passivation

idem metal3, 4, 5, 6....

En tout une vingtaine de couches.

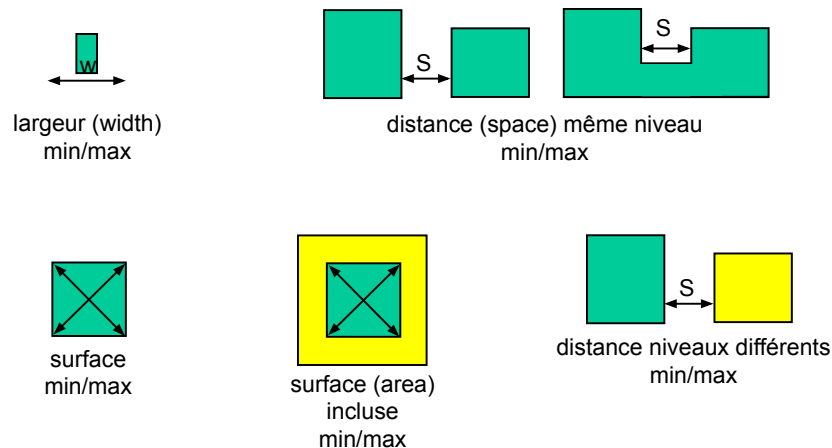
# Procédé de fabrication CMOS

## Type de règles



# Procédé de fabrication CMOS

## Type de règles



# Le dessin aux règles fondeur

- Chaque technologie a ses propres règles
  - Techno ES2 1 $\mu$ m  $\approx$  90 règles
  - Techno ST 90nm  $\approx$  400 règles
- Tous les masques sont sur une grille minimale ( $\approx$  1/10 la longueur minimale d'un transistor).
- Si on dessine en suivant les règles du fondeur, on dessine tout :
  - il faut apprendre les centaines de règles !
  - De fait réservé aux fondeurs ou aux circuits analogiques.
- Pour pérenniser son travail, chaque fondeur essaie d'avoir des règles homothétiques entre ses technologies pour limiter le re-dessin.

# Le dessin symbolique lambda

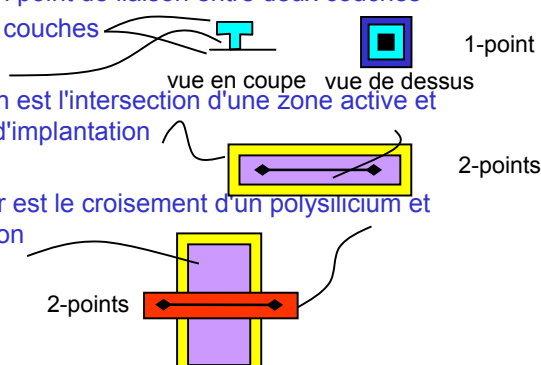
## Définition

- Le dessin symbolique utilise une représentation stick.
  - Seuls les axes des fils sont représentés.
  - Un fil représente un symbole : segment (2-points) ou via (1-point)
  - Le nombre et le type des symboles est réduit :
    - metal run, metal contact, polysilicium, contact, via12, via23, etc,
    - diffusions (*intersection d'une zone active et d'une implantation*),
    - transistors (*croisement d'un polysilicium et d'une diffusion*), ...
  - Les symboles peuvent être annotés par une largeur.
- Le dessin symbolique lambda se fait sur une grille régulière au pas de 1 lambda.

# Le dessin symbolique lambda

## Macro génération des motifs

- Les transistors, les diffusions et les vias sont des symboles. Ils représentent toutes les couches nécessaires.
- Un via est un point de liaison entre deux couches
  - les deux couches
  - et le cut
- Une diffusion est l'intersection d'une zone active et d'une zone d'implantation
- Un transistor est le croisement d'un polysilicium et d'une diffusion



# Le dessin symbolique lambda

## Définition

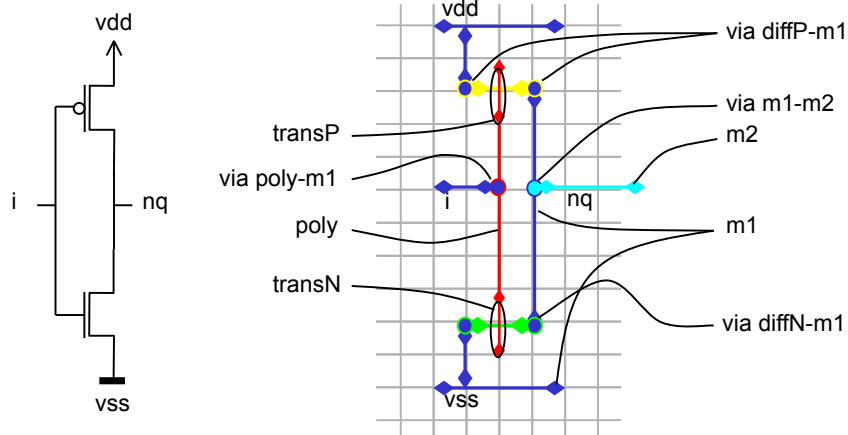


Schéma d'un inverseur.

La grille a un pas de 1 lambda.

# Le dessin symbolique lambda

## Fondement

Une étude statistique (*Greiner90*) sur une vingtaine de technologies de  $2\mu\text{m}$  à  $0.6\mu\text{m}$  a montré que les paramètres géométriques les plus homogènes sont les distances centre-à-centre des fils de largeur minimale.

En d'autres termes :

entre 2 technologies équivalentes (même longueur de grille),

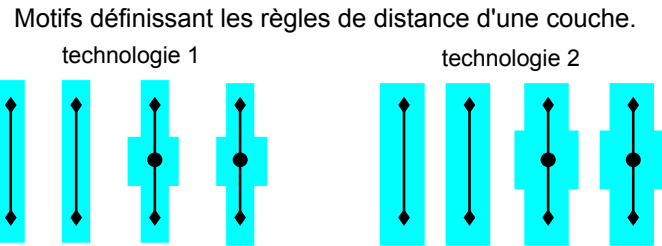
- la largeur minimale des fils change,
- la distance minimale entre les fils change,

mais

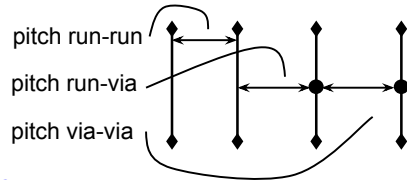
- la distance centre-à-centre ne change pas ou peu.

# Le dessin symbolique lambda

## Fondement (illustration)



Les règles de largeur et de distance sont différentes, mais les règles entre les axes sont les mêmes au changement d'échelle près.



# Le dessin symbolique lambda

## Passage du symbolique au réel

- Un fil de largeur symbolique minimale sera traduit en un ou plusieurs rectangles réels de largeur minimale.  
Pour un type de fil donné les largeurs minimales réelles et symboliques sont telles que :

$$W_{\text{réel}}_{\text{min}} = W_{\text{symb}}_{\text{min}} * \lambda + \Delta W_{\text{réel}}$$

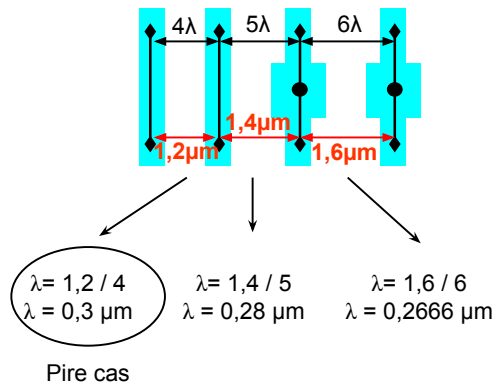
et ceci pour chaque rectangle constituant le fil.

- Pour un type de via donné Les dimensions de chaque rectangle réel le constituant sont imposées et fixes.
- Pour passer d'un dessin symbolique à un dessin réel, il faut connaître la valeur du lambda, le  $\Delta W$  de chaque type de fil et la définition de chaque via.

# Le dessin symbolique lambda

## Calcul de la valeur du lambda

On prend toutes les règles symboliques, on les compare aux règles réelles correspondantes et on garde le pire cas

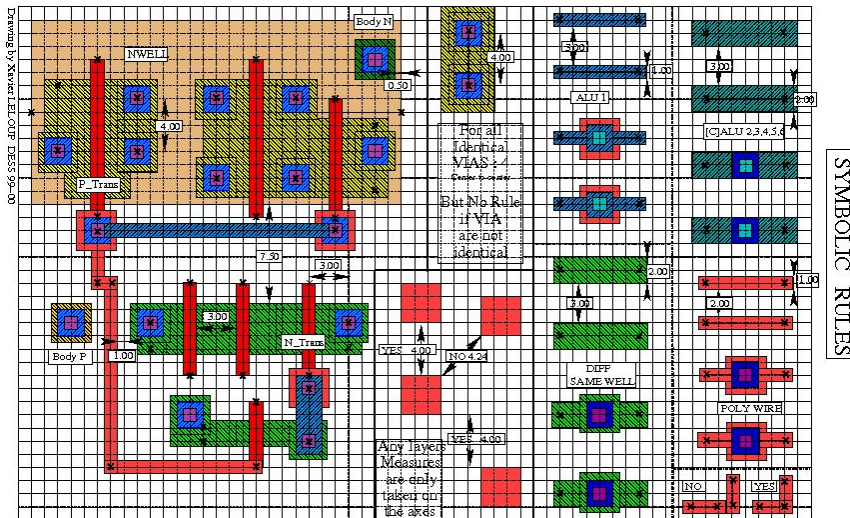


# Les règles de dessin Alliance

- Le dessin se fait sur une grille au pas de 1 lambda.
- Le dessin utilise des objets mono-point ou bi-points
  - mono-point : via (type), référence (nom), ...
  - bi-points : segment (type, largeur) horizontaux ou verticaux  
big-via (type, largeur) surface définie par une diagonale  
boite d'aboutement. idem
- Les segments ont une largeur minimale qui s'étend de part et d'autre du centre et une extension aux extrémités.
  - On peut augmenter la largeur par pas de 1 lambda,
  - l'extension est fixe.
- Les objets sont constitués de plusieurs couches.
- Les règles de dessin Alliance peuvent être assimilées aux règles d'une technologie 1μm.



# Les règles de dessin Alliance



# Dessin des cellules

## Problèmes ...

- La technologie symbolique a été définie en supposant que les seules règles étaient des valeurs minimales pour les distances, largeurs, recouvrement, inclusion et extension.
- Les technologies actuelles ont des règles plus complexes, ex:
  - densité minimale et maximale, pour certains niveaux
  - largeur maximale, par ex. pour les métaux
  - longueur maximale, pour réduire la résistance
  - distance maximale, par ex. pour les caissons
  - certains alignements interdits, pour les transistors
  - etc....

Ces règles ne sont pas vérifiées par le vérificateur de règles symboliques d'Alliance, mais le sont par construction des cellules ou par le routeur.

## Quel est le problème ?

En raison de la dimension des circuits numériques, il n'est pas possible de les réaliser à plat.

- On définit donc des bibliothèques
    1. de cellules de base (opérations booléennes, drivers, mémoire)
    2. d'opérateurs complexes paramétrables (multiplieur, ram, rom...)
    3. de composants (processeur, dma, ...)
  - Les cellules sont souvent précaractérisées, c'est-à-dire
    - qu'elles sont valides du point de vue des règles de dessin,
    - que leurs caractéristiques électriques sont connues.
  - L'investissement en temps au dessin des masques est considérable
    - ~ 100 cellules dans une bibliothèque de cellules)
    - plusieurs centaines pour les générateurs
- du layout portable indépendant du process,  
 □ une circuiterie robuste indépendante du process.



# Plan

- gabarit des cellules
- méthodologie de dessin de cellule
- dimensionnement des transistors

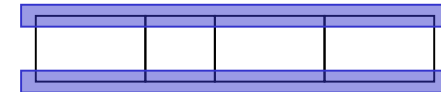
# Gabarit des cellules

## Contraintes

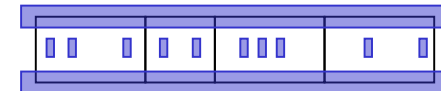
- Les cellules sont faites pour être aboutées : hauteur commune et largeur variable



- Les cellules se partagent les rails d'alimentation.



- Les cellules n'utilisent que le polysilicium et le metal1 pour router
- Les connecteurs sont uniquement en metal1, le routage est fait au dessus des cellules en metal2 et plus.



# Gabarit des cellules

## Hypothèses

- Une bibliothèque de cellules précaractérisées contient
  - des fonctions booléennes élémentaires : and, or, xor, mux, adder, ...
  - des éléments mémorisant et des barrières : basculeD, latches, drivers trois-état, ...
  - des éléments divers utiles pour le placeur et le routeur : cellules de bourrage, rappel d'alim, pull-up, pull-down, ...
- Dans tous les cas, moins de 20 transistors (environ)
- Une cellule est toujours « self-consistent »
  - une fonction booléenne complète
  - sans violation de règles de dessin (sauf la polarisation)
  - sans contrainte d'utilisation (hormis la sortance)
  - avec une circuiterie robuste (pas d'hypothèse sur la techno)

# Gabarit des cellules

## Contraintes

- La hauteur des cellules est un compromis :
  - Elle doit être assez grande pour permettre le routage des transistors avec des fils de metal1.
  - Elle doit être assez petite pour que ce ne soit pas l'aboutement des cellules qui impose la taille du circuit.
- La taille des cellules et la position des connecteurs sont des contraintes du routeur.
  - Le routeur utilise une grille de routage égale au pitch de routage
  - la taille d'une cellule et la position de ses connecteurs doivent être de multiple de ce pitch.

# Gabarit des cellules

## Contraintes

Les cellules sont faites pour être aboutés

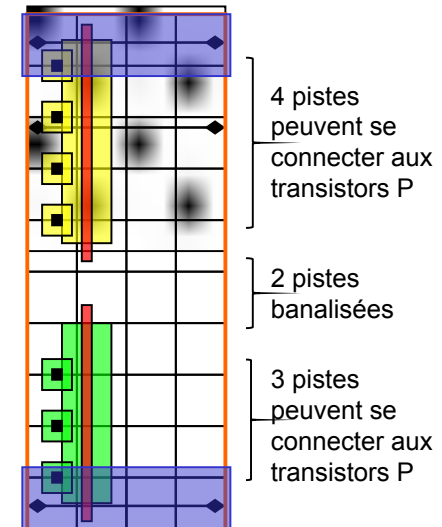
Il ne faut qu'il y ait de violation de règles entre les masques de cellules différentes.

no-mask-land :

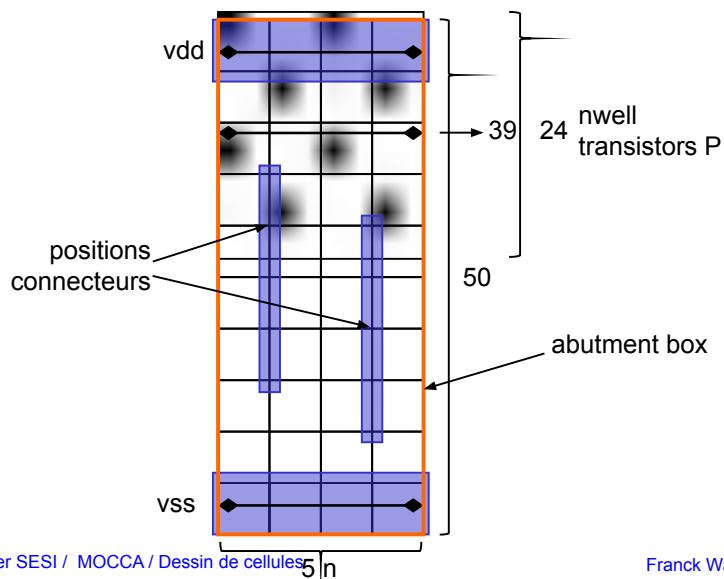
Tout symbole doit être au moins à une demi-distance minimum de l'aboutment box (sauf le NWELL et les rails VDD et VSS).

(exemple : distance poly-poly = 2  
le bord de tout poly est distant d'au moins 1)

# Gabarit : Position des transistors



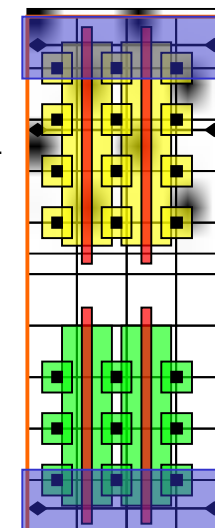
# Gabarit : Format choisi



# Gabarit : Pitch de routage

Sur cette figure, les transistors ont été mis au plus près. On note que le pitch entre les grilles est de 6.

Le positionnement des connecteurs va poser un problème délicat !



**pitch**  
**transistor = 6**  
**routage = 5**

Attention, cette cellule incomplète comporte des erreurs de dessin !

# Règles de gabarit

Les règles de gabarit définissent les contraintes sur la forme des cellules pour une bibliothèque, ici :

- Une hauteur de 50 lambda pour l'aboutement
- Une largeur multiple de 5 lambda pour le Place & Route
- Une position et une largeur pour les alimentations et le caisson pour assurer un routage des alimentations par aboutement
- Des coordonnées de connecteurs multiples de 5 lambda par rapport à l'aboutment-box pour le Place & Route
- Un « no-mask-land » pour ne pas avoir de contrainte de placement des cellules entre elles

# Dessin sur papier

## concept de dessin symbolique sur papier

- L'objectif du dessin sur papier est de gagner du temps est s'assurant, avant de dessiner sur l'éditeur, que le dessin sera réalisable.
- Les transistors et tous les segments sont représentés par des fils et les vias par des ronds (ou des croix).
- Le routage des transistors tient compte des possibilités offertes par le gabarit.
- Le principe consiste à ne pas introduire simultanément toutes les contraintes (placement des transistors de taille bien déterminée, routage des transistor, placement des connecteurs, etc...)

# méthode de dessin de cellule CMOS

## Principe général

1. Faire un schéma en transistors non dimensionnés
2. Déterminer un placement des transistors permettant de minimiser la taille en maximisant le nombre de sources/drains communs.
3. Placer les transistors dans le gabarit symbolique.
4. Router symboliquement la cellule sur la grille.
5. Placer les segments connecteurs.
6. Dessiner la cellule sous graal et retourner en 3 si échec.
7. Dimensionner la taille des transistors.

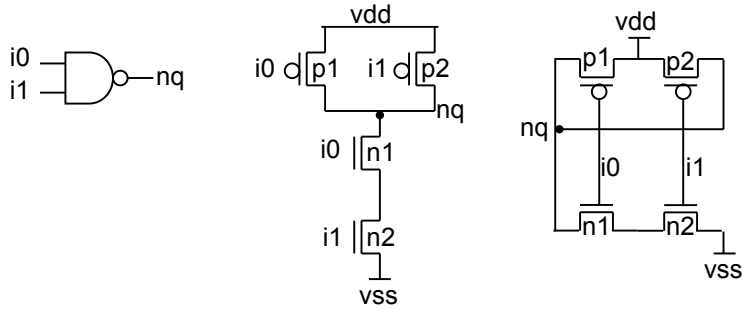
Règles  
simplifiées

# Règles de dessin simplifiées

## Distances centre-à-centre

- Trans-Trans = **6 avec contact au milieu  
2 bout-à-bout (à 1 de l'AB)**
- Diff-Diff = **6 dans tous les cas (à 3 de l'AB)**
- Poly-Poly = **3 fil-fil  
4 fil-cont  
5 cont-cont**
- Metal1-Metal1 = **5 fil-cont ou cont-cont  
5 fil-fil (en principe c'est 4)**

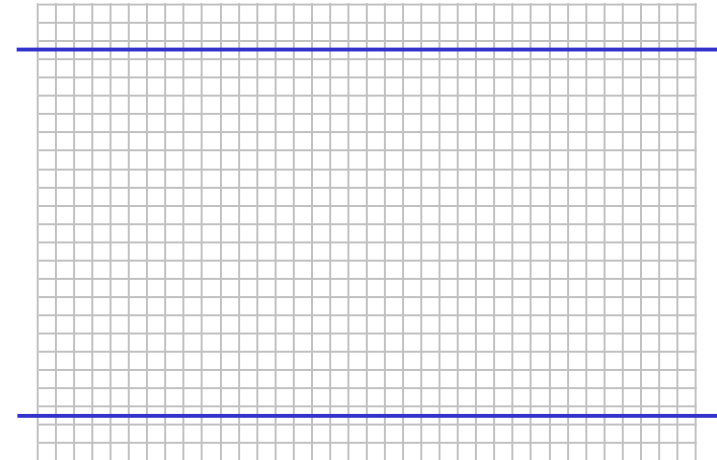
## Dessin d'un nand2



## nand2

### Les alimentations

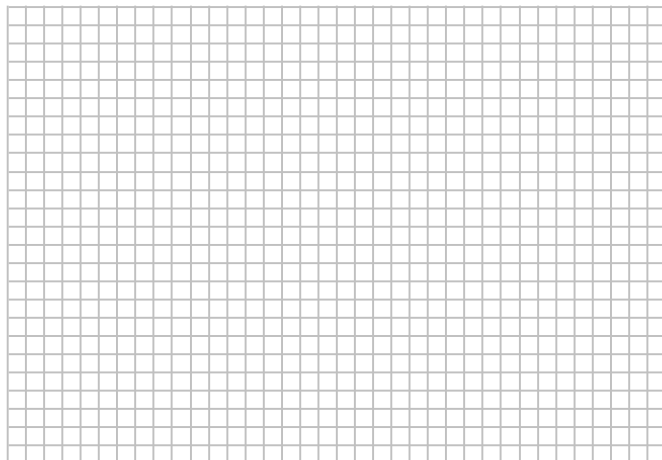
Tracer deux traits (bleu) correspondant à vdd et vss distant de 40 lamdbas. C'est à ces coordonnées que les transistors P et N seront polarisés.



## nand2

### Le terrain de jeu

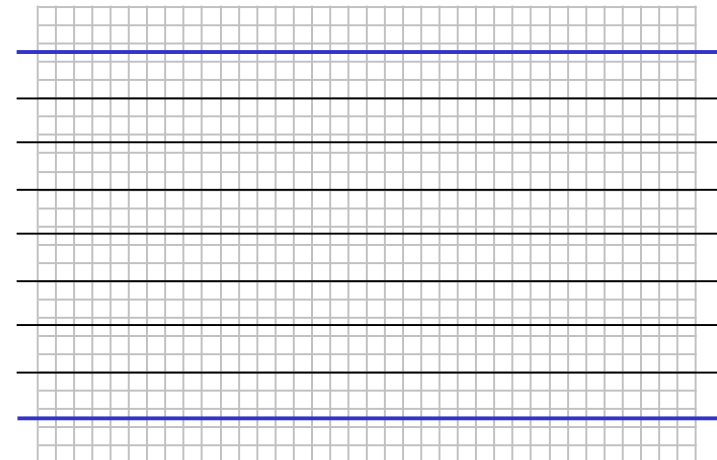
Prendre une feuille de papier quadrillé 5x5 : 1 carreau pour 2 lamdbas



## nand2

### Les pistes de routage

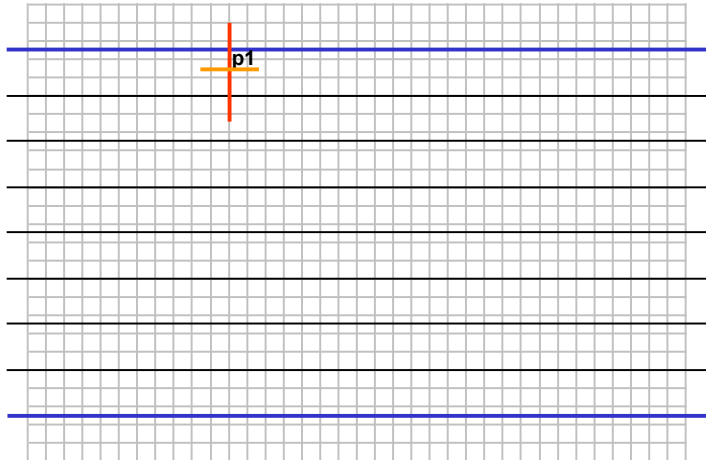
Tracer les pitchs horizontaux tous les 5 lamdbas. C'est à ces coordonnées que l'on fera le routage en metal1



# nand2

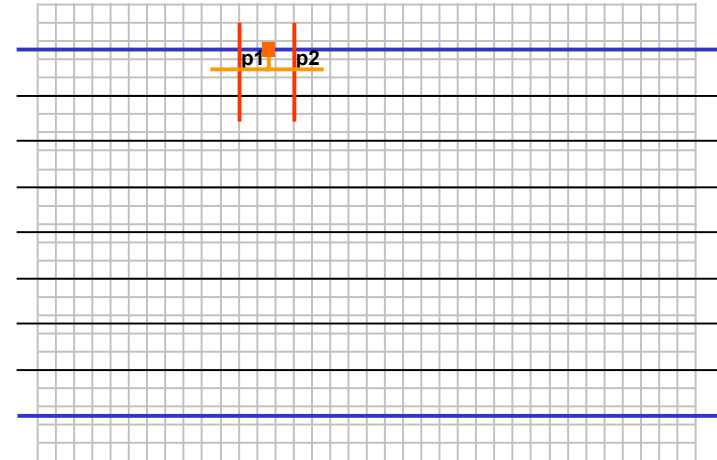
## Un premier transistor

Tracer les transistors comme sur le schéma horizontal le plus haut possible en les nommant.



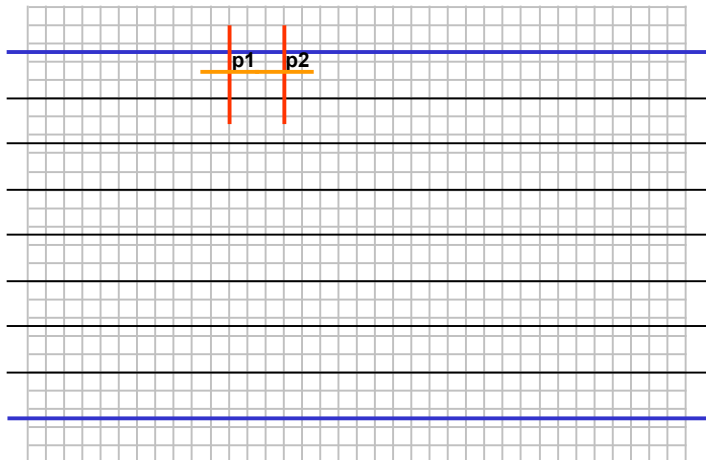
# nand2

## connexion à vdd



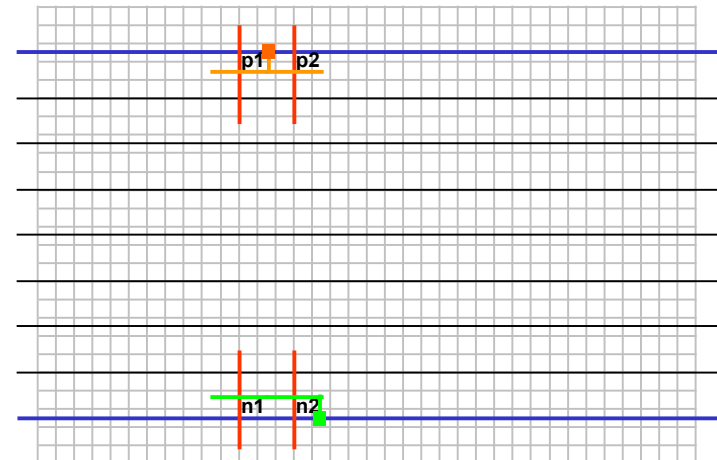
# nand2

## Son voisin

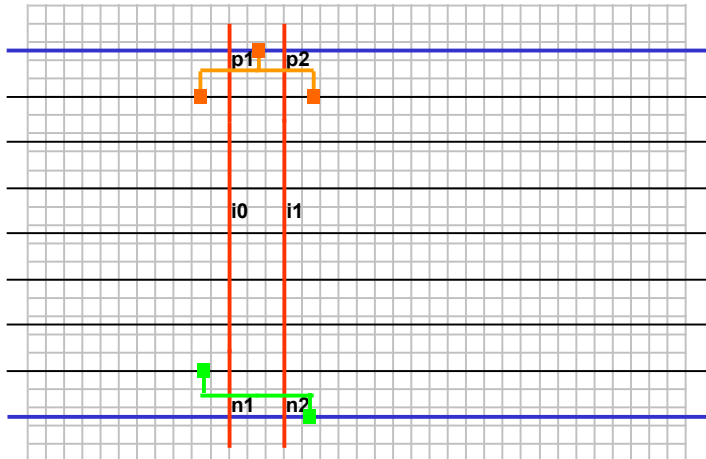


# nand2

## Les transistors N



## nand2 Connexion sur la première piste libre

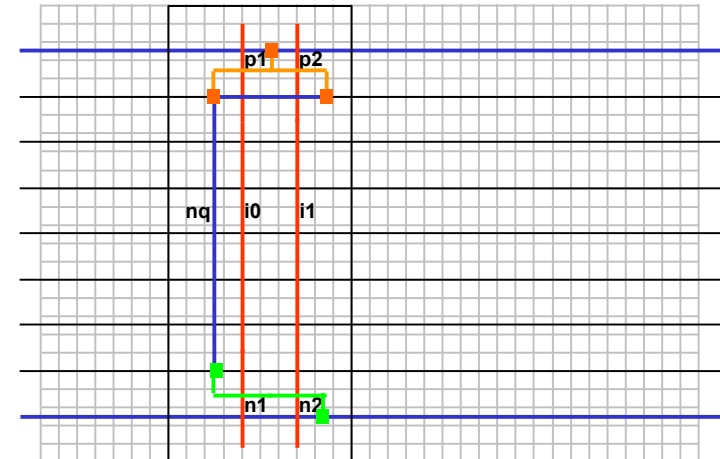


Master SESI / MOCCA / Dessin de cellules

Franck Wajsbürt 52

## nand2 On connaît la taille de la cellule

Tracer l'abutment box en agrandissant au pitch le plus proche et sachant qu'il va falloir positionner les connecteurs i0, i1, et nq.

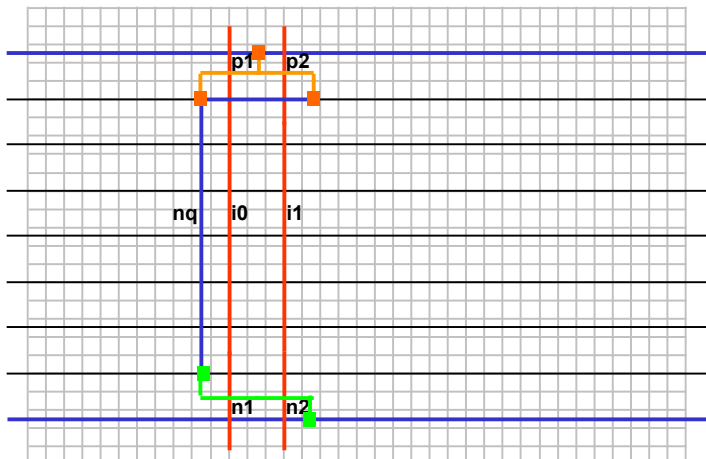


Master SESI / MOCCA / Dessin de cellules

Franck Wajsbürt 54

## nand2 routage du metal1

Tracer les transistors comme sur le schéma horizontal en les nommant.

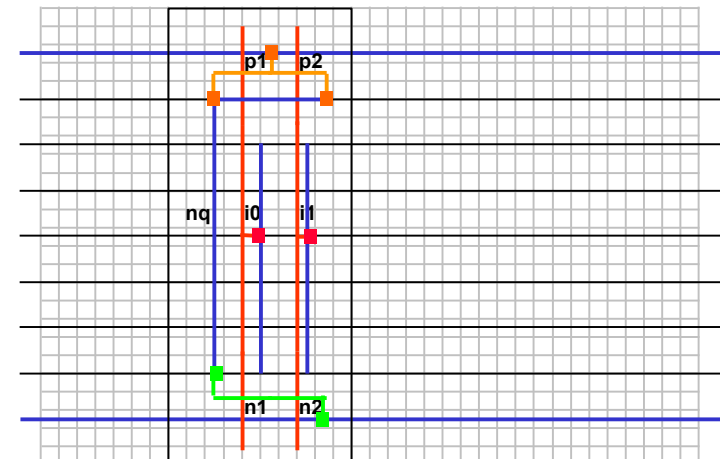


Master SESI / MOCCA / Dessin de cellules

Franck Wajsbürt 53

## nand2 On sait qu'elle est faisable

Tracer les connecteurs sur le pitch de routage vertical.

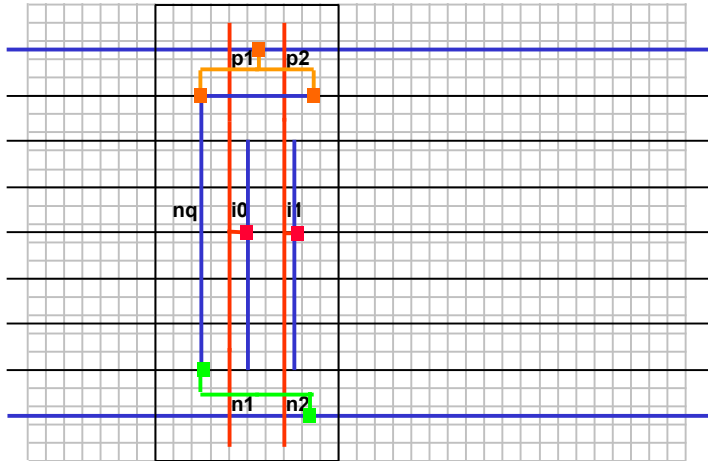


Master SESI / MOCCA / Dessin de cellules

Franck Wajsbürt 55

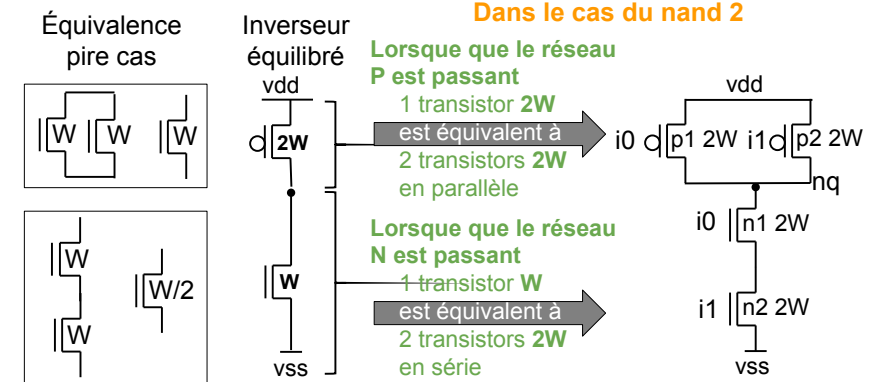
## nand2 dimensionnement max des transistors

La cellule est routable, on peut dimensionner les transistors. Ici le routage n'impose pas de limite. N et P peuvent aller jusqu'à 17 et 23  $\lambda$ .



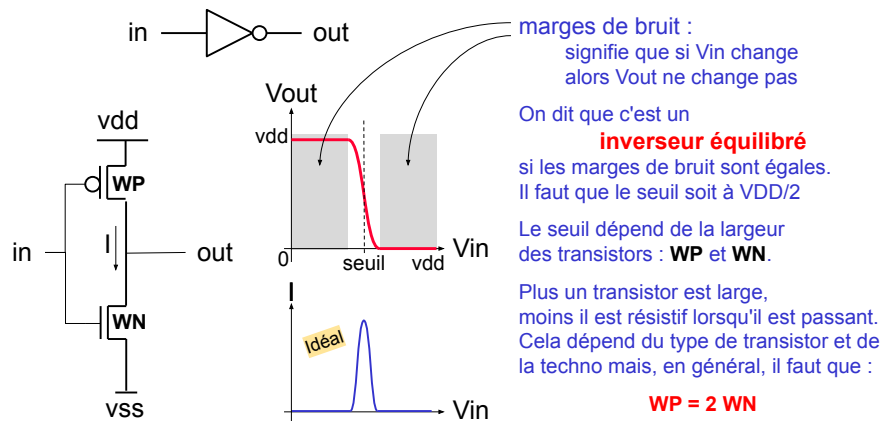
## Dimensionnement des transistors

Pour une porte quelconque on ne peut pas équilibrer les marges de bruit car il y a plusieurs conditions de changement d'état en fonction de la valeur des entrées. L'idée est de trouver la largeur des transistors de la porte complexe en partant de l'**inverseur équivalent**, en supposant le pire cas.

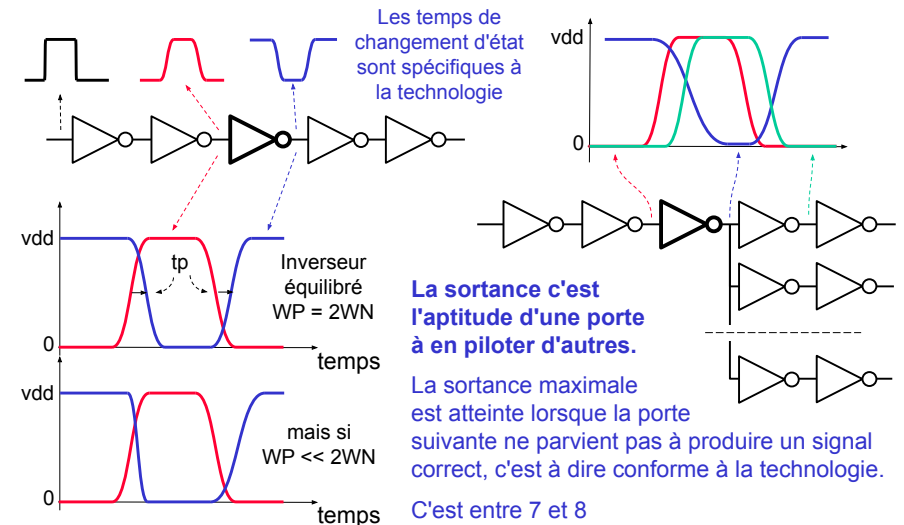


## Dimensionnement des transistors

Objectif : déterminer la taille relative des transistors N et P



## Caractérisation dynamique et sortance





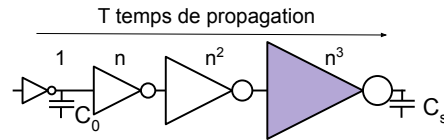
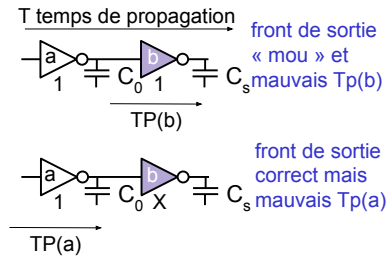
# Adaptation de charge

Pour obtenir une sortance importante et (dé)charger rapidement une grande capacité ( $C_s$ ), il faut des transistors plus larges pour la porte qui commande  $C_s$  (nommée porte d'adaptation) mais sans dégrader le temps de propagation total et en conservant des fronts corrects.

Sur le schéma, la porte violette est la porte d'adaptation

L'idée est de faire une chaîne d'inverseurs de taille de plus en plus grandes.

On peut montrer (je ne fais pas) qu'avec  $n$  entre 3 et 4 on n'a le meilleur temps de propagation

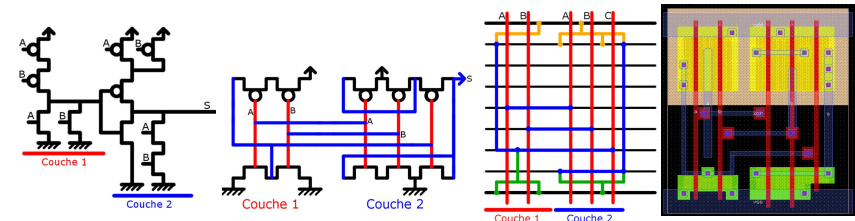
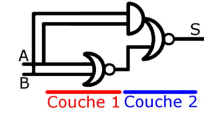


# En résumé

## pour dessiner une cellule

- concevoir un schéma de transistors
- dessiner le schéma avec les transistors « à plat »
- dessiner le schéma dans le gabarit avec des transistors minimaux
- placer la boîte d'aboutement et les E/S
- déterminer la taille des transistors
- dessiner le schéma sur Graal en respectant les règles de dessins

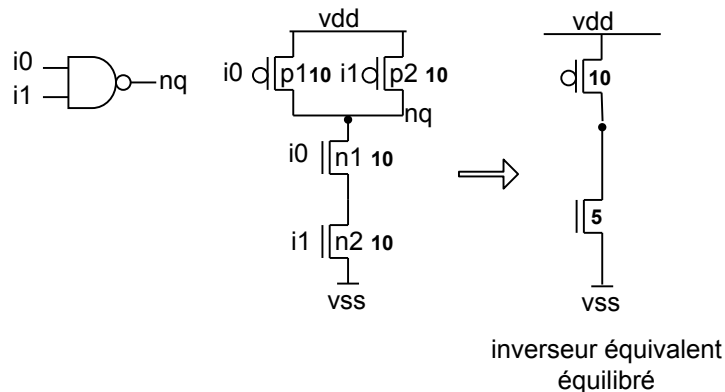
$$S \leq A \text{ xor } B$$



# nand2

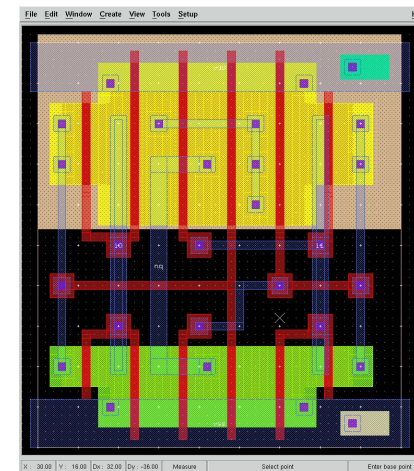
## dimensionnement choisi

schéma d'un nand 2 dimensionné



# Sinon...

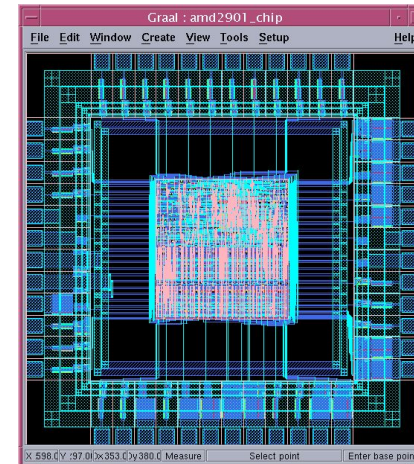
... on peut aussi router en polysilicium !



## Outils Alliance

graal / dreal  
s2r / druc  
yagle / proof

## graal Editeur de dessin symbolique



## graal

### Editeur de dessin symbolique

- graal est un éditeur de dessin symbolique lambda.
  - édition de cellule,
  - placement d'instances pour un dessin hiérarchique,
  - navigation à travers la hiérarchie,
  - tracé d'équipotentielle,
  - appel du vérificateur de règles de dessin et affichage séquentiel des erreurs,
  - copie d'écran.

graal [-l <filename>]

- Environnement
  - MBK\_IN\_PH, MBK\_OUT\_PH
  - RDS\_IN, RDS\_OUT
  - RDS\_TECHNO\_NAME

## druc

### vérificateur des règles de dessin

- druc permet de vérifier les règles de dessin définies dans le fichier \$RDS\_TECHNO\_NAME.
- druc peut être appelé par graal. Quand il est utilisé seul, il génère un fichier d'erreur avec les erreurs et les rectangles concernés et un fichier .cif chargeable par graal.

druc <file>

- Environnement
  - MBK\_IN\_PH, MBK\_OUT\_PH
  - RDS\_IN, RDS\_OUT
  - RDS\_TECHNO\_NAME

## s2r

### passage du symbolique au réel

- s2r permet de traduire un dessin symbolique en dessin réel respectant les règles d'un fondeur
- en entrée
  - le layout symbolique (possiblement hiérarchique)
  - les règles de transformation de chaque symbole :  
1 symbole à l'échelle  $\lambda \rightarrow n$  rectangles à l'échelle  $\mu\text{m}$
- son travail
  - traduire individuellement chaque symbole
  - éliminer les notchs créés par la traduction
  - remplacer certaines cellules symboliques par leur équivalent réel
  - sauver la hiérarchie complète dans un format fondeur (.cif/.gdsII)

## s2r

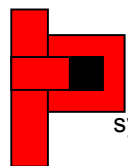
### passage du symbolique au réel

s2r -tv <file>

- Option
  - t pas de gestion de notch
  - v mode verbeux
- Environnement
  - MBK\_IN\_PH, MBK\_OUT\_PH
  - RDS\_IN, RDS\_OUT
  - RDS\_TECHNO\_NAME

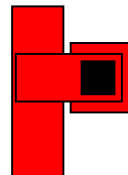
## s2r

### problème du notch



symbolique

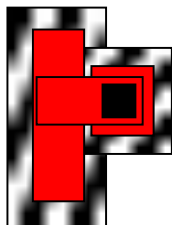
réel



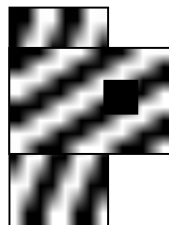
**le problème vient de ce que le via a subit un rétrécissement**

- La solution consiste à faire grossir tous les masques de la demi-distance min – 1 pas de grille physique
- puis d'unifier les rectangles qui se touchent
- puis de réduire tous les masques de la demi-distance – 1 pas de grille

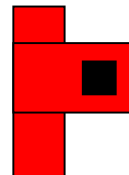
blot



unify



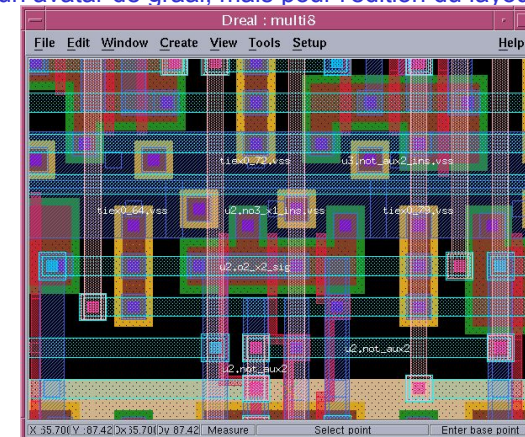
shrink



## dreal

### Éditeur de dessin réel

dreal est un avatar de graal, mais pour l'édition du layout réel.

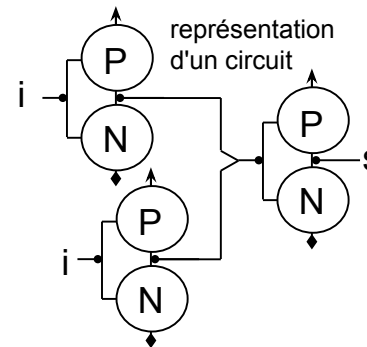


# validation

- Valider une cellule ou un bloc signifie
  - que le dessin est sans erreur de dessin  
**druc**
  - qu'elle se comporte électriquement et temporellement bien  
**cougar avec les règles de dessin du fondeur**  
+ **spice (eldo)**  
+ **tas**
  - que son comportement correspond à celui attendu  
**cougar + spice** donne des informations  
**cougar + yagle + proof**

# yagle abstracteur

- yagle prend un dessin netlist de transistor et en abstrait le comportement pour produire du vhdl.
- principe général :
  - S est à 0 si le réseau N est passant
  - S est à 1 si le réseau P est passant



en CMOS :  
le réseau P est le dual du réseau N  
on peut ne regarder que le réseau N

# cougar extracteur de netlist

- cougar prend un dessin de masque à plat ou hiérarchique et produit une netlist de cellule ou de transistors.
- grâce au fichier CATAL on peut décider de jusqu'où se fait la mise à plat

ATTENTION au fichier techno défini par  
RDS\_TECHNO\_NAME

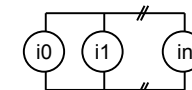
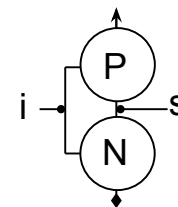
- si on veut utiliser la netlist pour une simulation temporelle, il faut utiliser une technologie réelle

# yagle

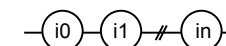
en CMOS :

- Le réseau P est le dual du réseau N.
- On peut ne regarder que le réseau N.
- On regarde les conditions qui le rendent passant
- Le réseau N est constitué de transistors en série et en parallèle.
  - Pour que des transistors en série soient passants, il faut qu'il soit tous passants → ET
  - Pour que des transistors en parallèle soient passant il faut au moins un de passant → OU

représentation d'une couche logique



→ i0 OU i1 OU ... OU in



→ i0 ET i1 ET ... ET in

# TME

En TME, vous allez redessiner la cellule nand2 en utilisant les outils d'Alliance et vous allez la valider.

Il n'y a pas de difficulté, hormis le respect de la méthode présentée et l'apprentissage de l'usage des outils d'Alliance