Modélisation Comportementale

version 1.0

Avertissement

On ne s'intéresse dans ce cours qu'a 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.

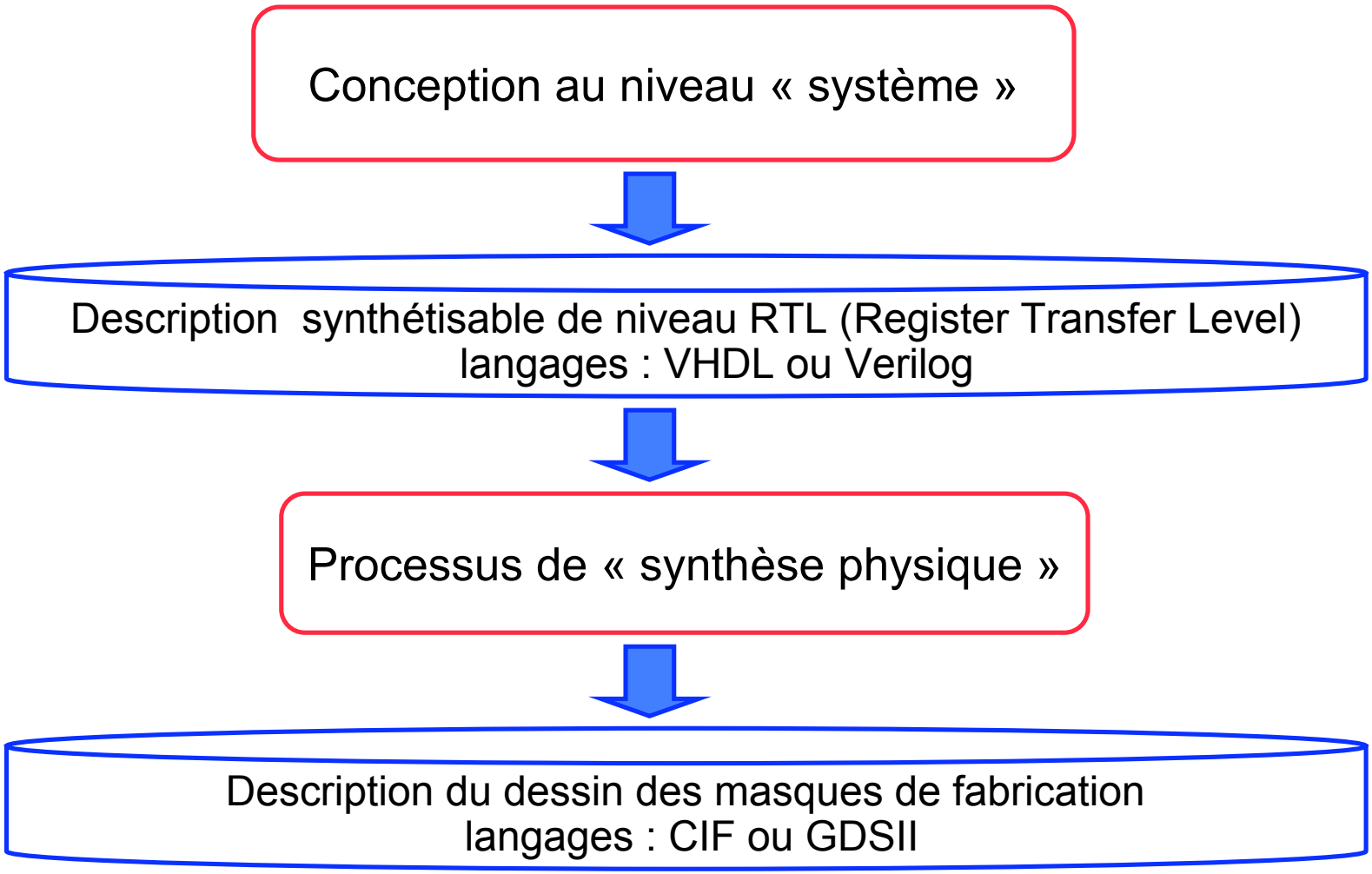
- Synchrones

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

Plan

- **Les trois « vues »**
- **Conception descendante**
- **Modélisation comportementale**
- **Exemple du composant « addaccu »**

Flot de conception

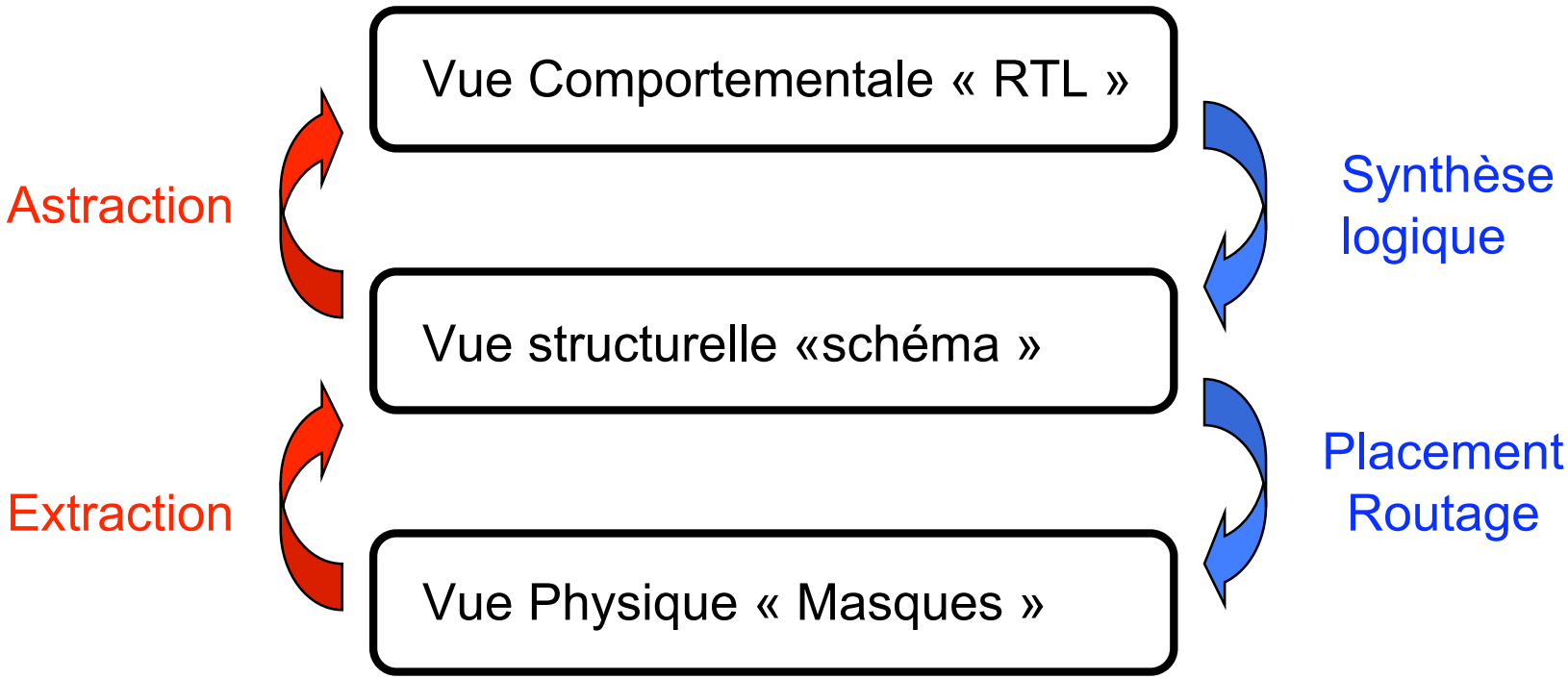


Les trois vues d'un circuit intégré

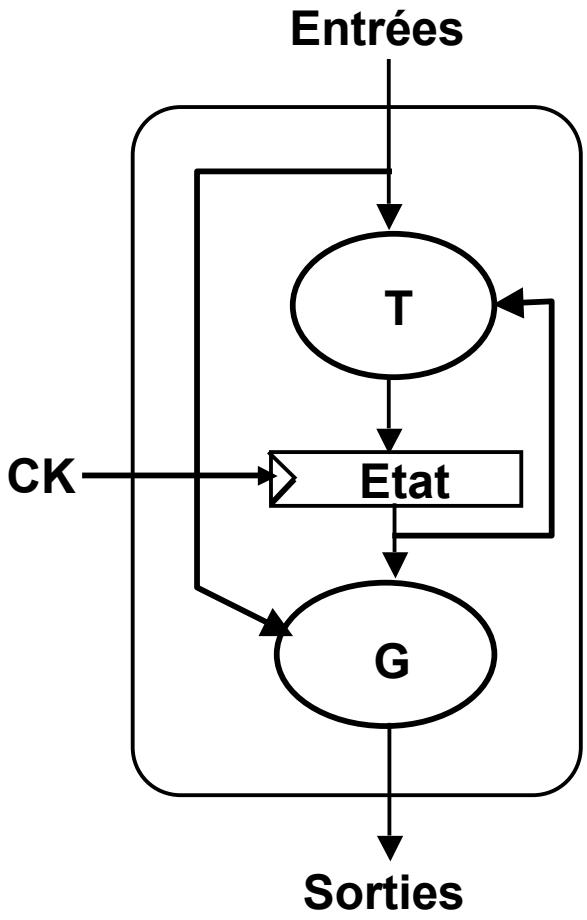
- Dans la phase de synthèse physique, on peut décrire le composant à trois niveaux d'abstraction :
  - vue conportementale (« behaviour ») : Elle permet de simuler et donc d'analyser le comportement, mais ne décrit pas la structure interne du composant modélisé.
  - vue structurelle (« net-list ») : elle 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 ») : elle décrit le dessin des masques de fabrication qui sont utilisés pour graver le silicium.

- Le processus de conception consiste à transformer progressivement la description comportementale en une description physique (utilisable par le fabriquant de circuits). Les outils CAO de synthèse physique permettent d'automatiser cette transformation.

Les trois vues



La vue comportementale RTL



Modèle de référence :
Automades d ’états synchrones

Fonction de transition :

NextEtat <= T(Etat, Entrées)

Fonction de génération :

Sorties <= G(Etat, Entrées)

La vue structurelle

C ’est une description hiérarchique du schéma d ’interconnexion.
Les éléments terminaux de cette description sont

- soit des cellules logiques
- soit des transistors

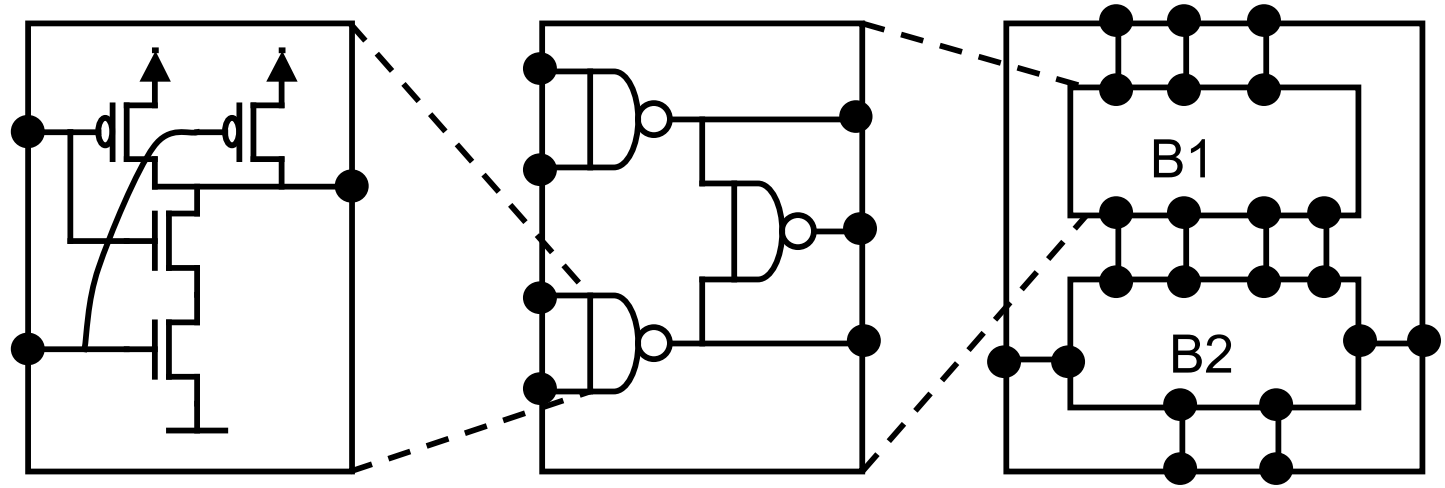


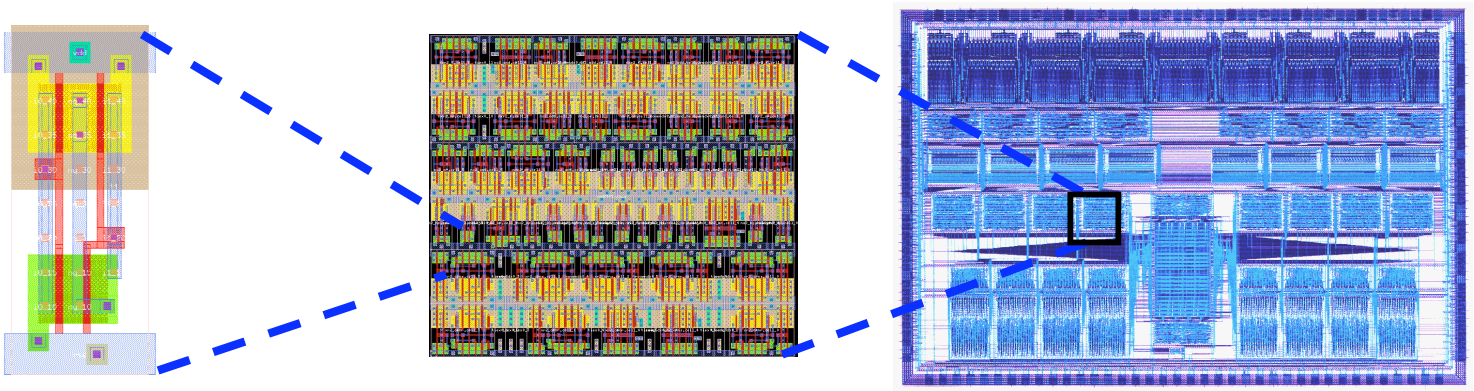
Schéma « transistors »

Schéma « portes logiques »

Schéma « blocs »

La vue physique

Elle représente le dessin des masques de fabrication.
C 'est une description modulaire et hiérarchique

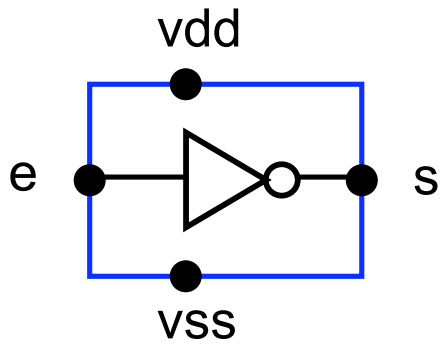


Une cellule élémentaire

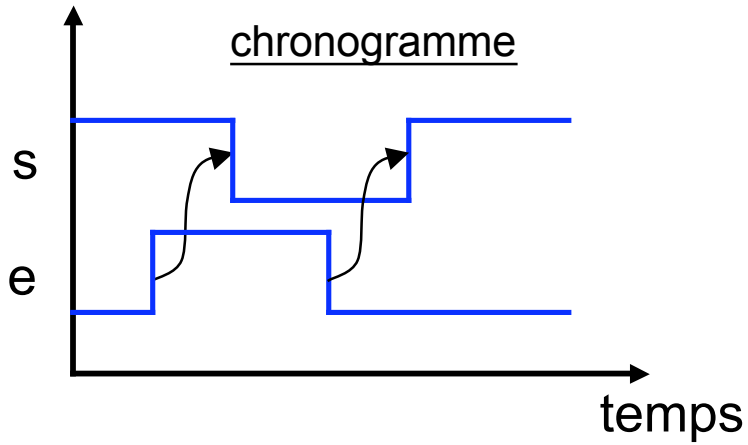
Un bloc fonctionnel

Un circuit complet

Vue comportementale inverseur CMOS



Ce composant matériel est décrit comme une « boîte noire », dont on ne connaît que l'interface (ports d'entrée/sortie), et le comportement logico-temporel :



LANGAGE VHDL

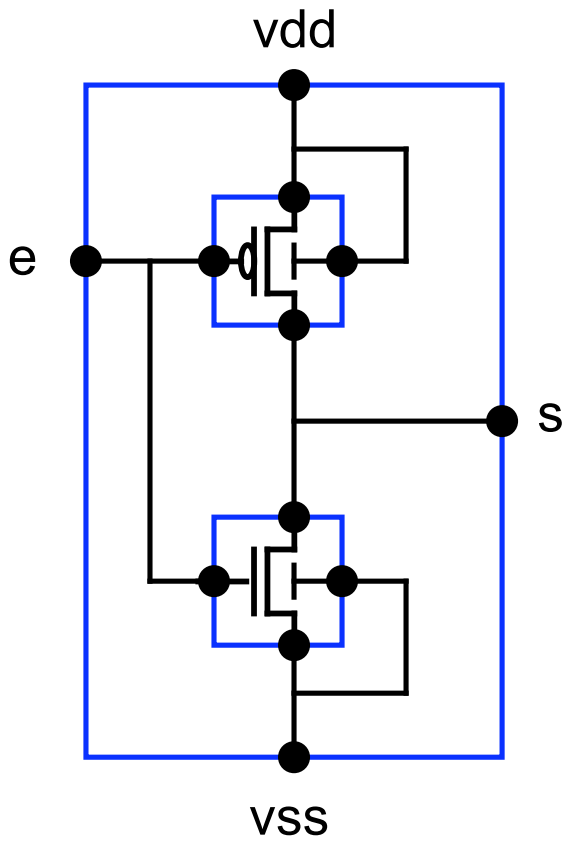
```
entity inverter is
-- liste des ports d'entrée/sortie
port (
      e : in bit ;
      s : out bit ;
      vss : in bit ;
      vdd : in bit
      ) ;
end inverter ;

architecture vbe of inverter is
-- comportement logico-temporel
begin
      s <= not e after 200 ps ;
end vbe ;
```

Vue structurelle inverseur CMOS

La structure interne du composant est décrite comme une interconnexion de composants matériels plus simples :
Dans le cas de l'inverseur :
- un transistor N
- un transistor P

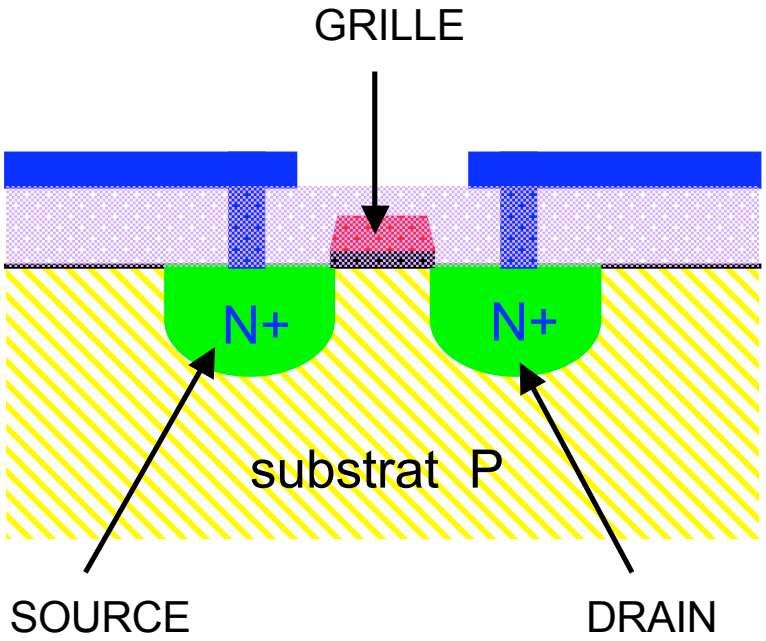
Les deux transistors N et P peuvent être considérés comme des interrupteurs contrôlés par la valeur du signal appliqué sur la grille :

Grille	0	1
Transistor N	bloqué	passant
Transistor P	passant	bloqué

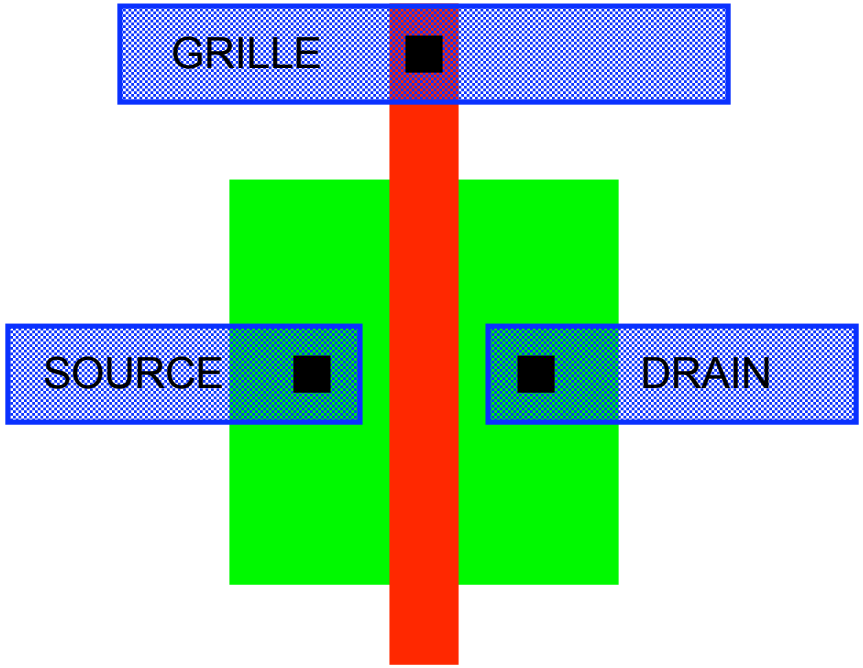


Vue physique inverseur CMOS / 1

Le transistor NMOS



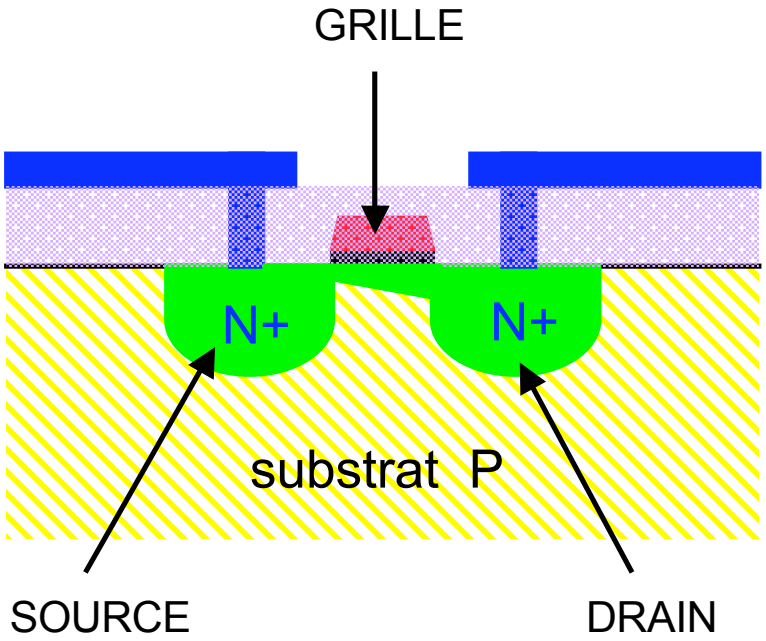
Vue en coupe



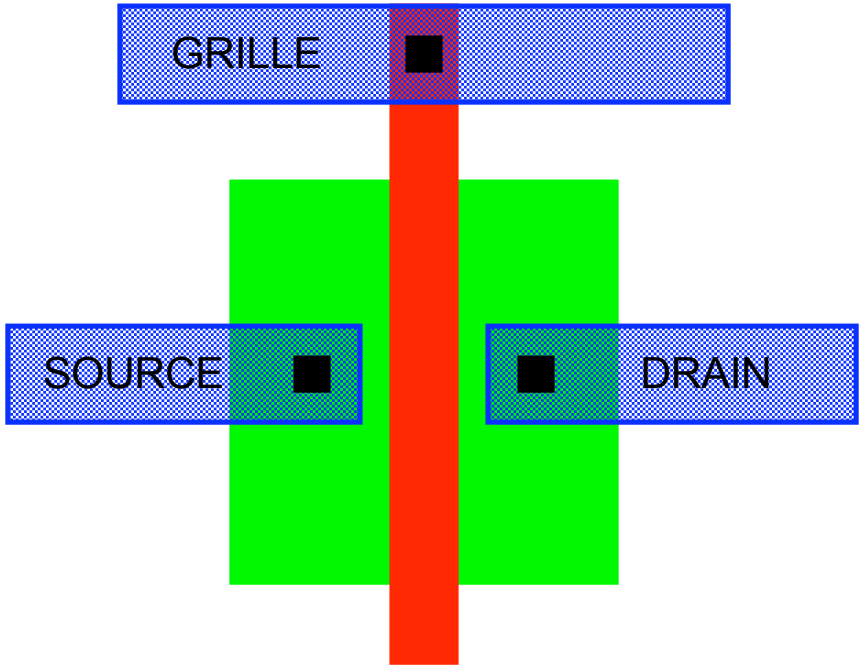
Dessin des masques

Vue physique inverseur CMOS / 1

Le transistor NMOS



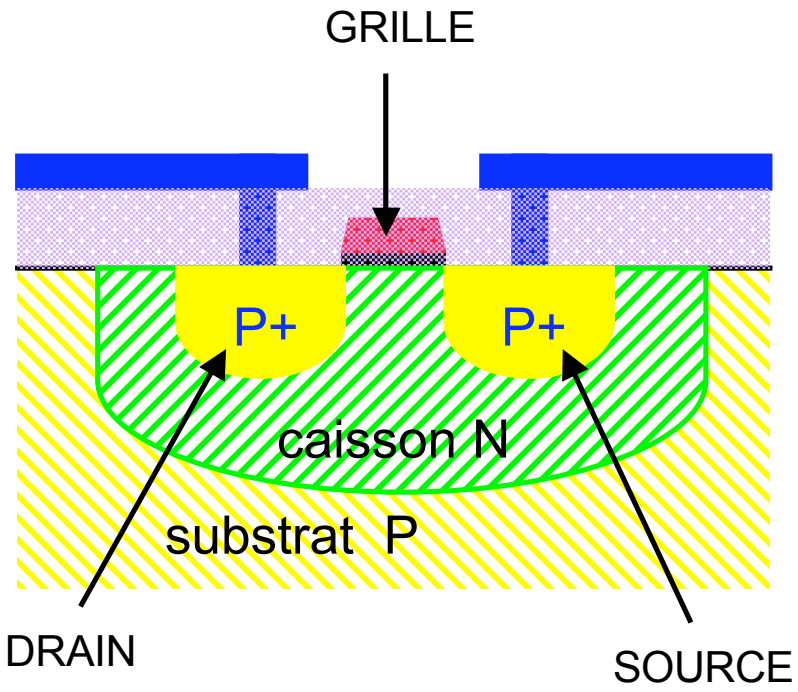
Vue en coupe



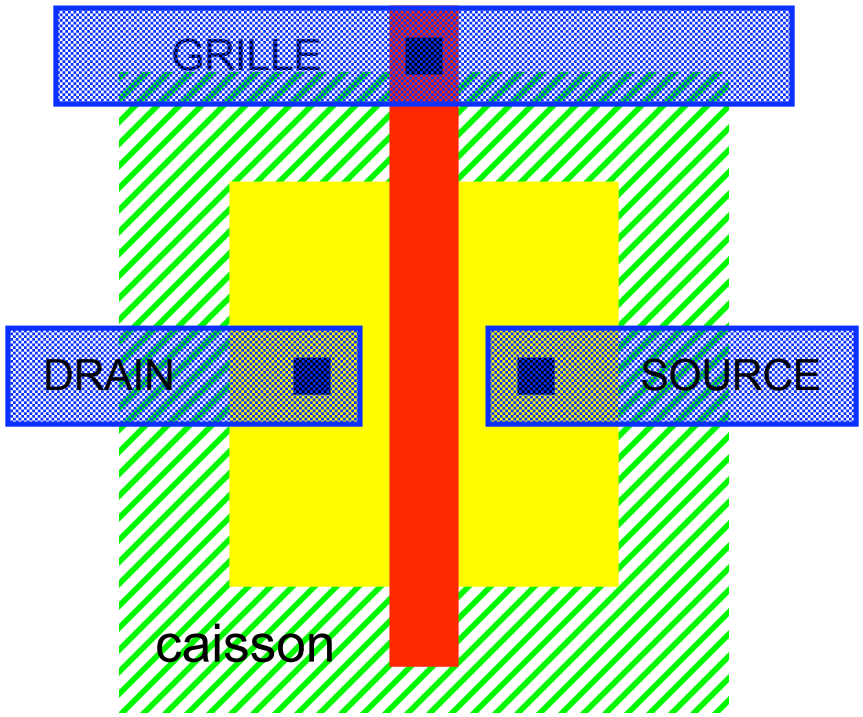
Dessin des masques

Vue physique inverseur CMOS / 2

Le transistor PMOS

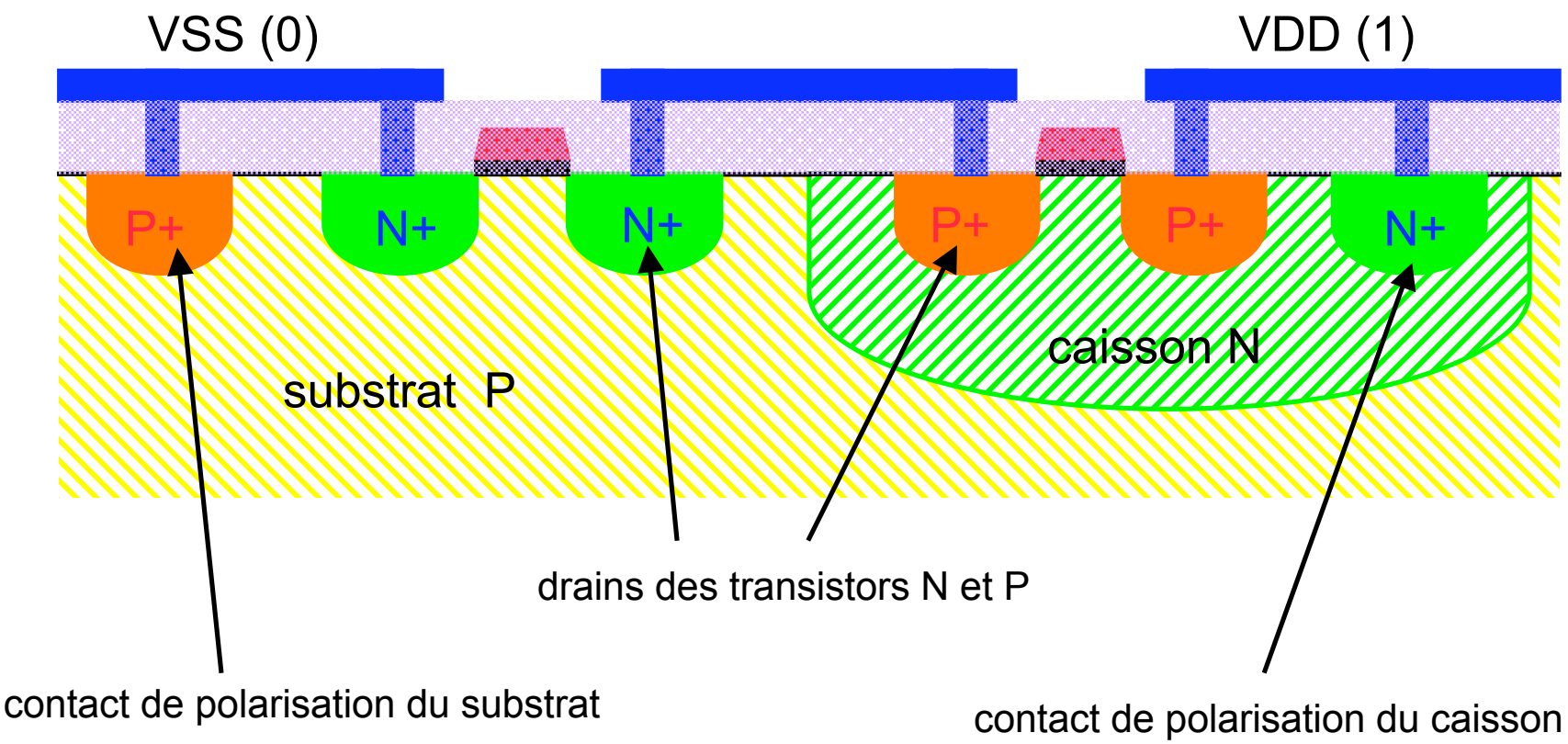


Vue en coupe



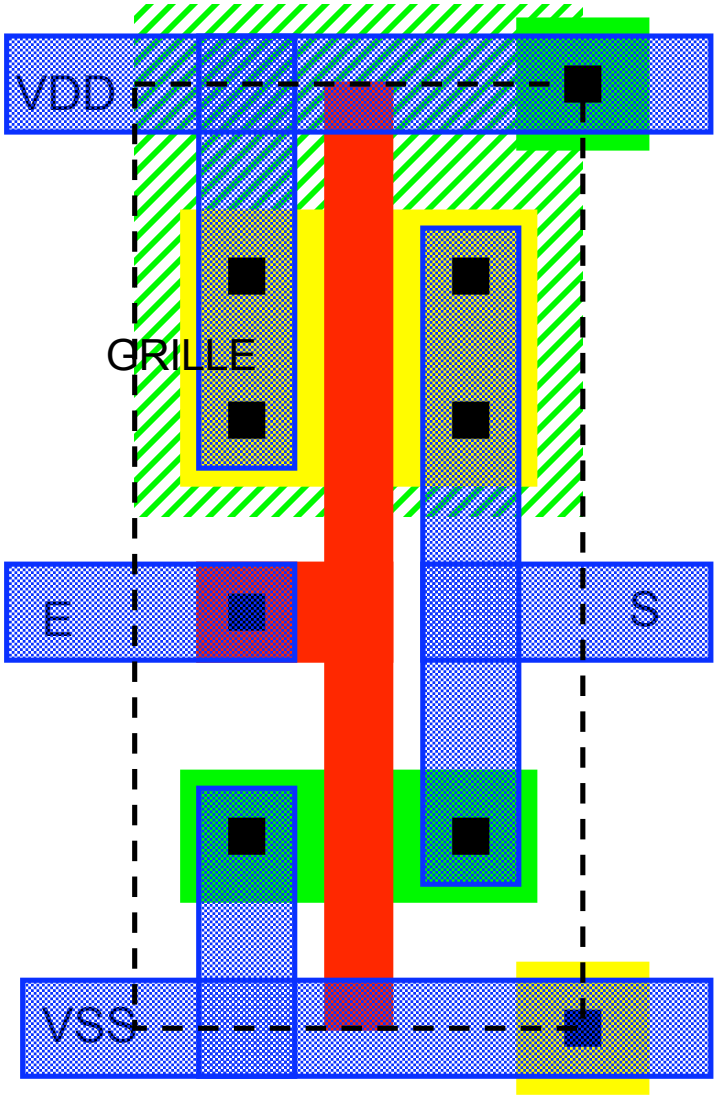
Dessin des masques

Vue physique inverseur CMOS / 3



vue en coupe de l'inverseur CMOS (2 transistors)

Vue Physique inverseur CMOS / 4



Dessin des masques

- vert : diffusion N
- jaune : diffusion P
- vert hachuré : caisson N
- rouge : Polysilicium
- bleu : métal 1
- noir : contact (trou dans l'oxyde de 1er niveau)

Les 4 ports d'entrée/sortie de l'inverseur (e,s,vdd,vss) sont en metal 1.

La « boîte d'aboutement » définit l'encombrement de la cellule, mais ne correspond pas à une couche physique.

Plan

- **Les trois « vues »**
- **Conception descendante**
- **Description comportementale**
- **Exemple du composant « addaccu »**

Top-down ou Bottom-up ?

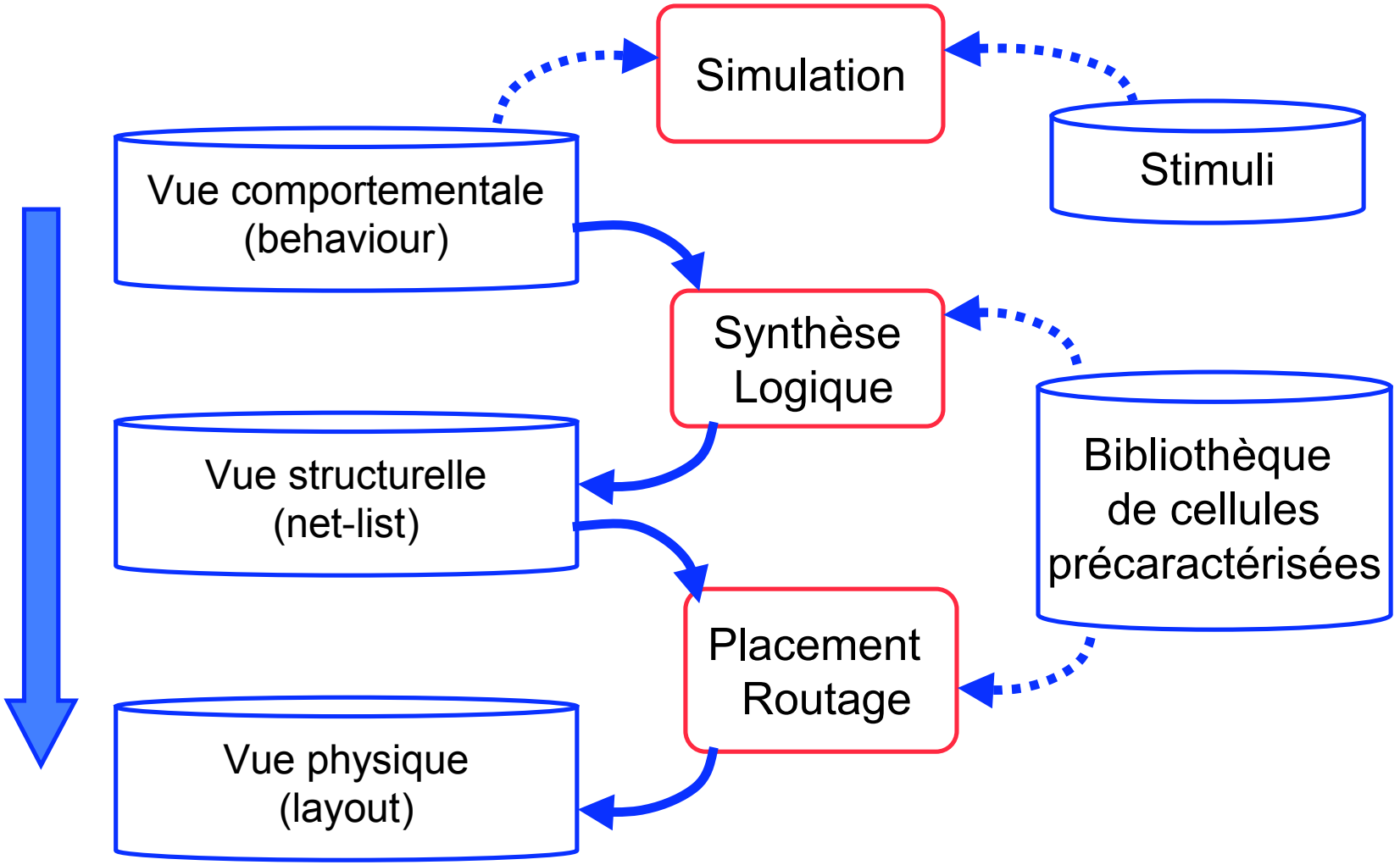
La méthode générale de conception est une méthode descendante (*top-down*).
On met en oeuvre des techniques de synthèse logique et de placement/routage,
en partant de la spécification comportementale pour aboutir au dessin des masques.

Cependant, pour vérifier que l'implémentation est conforme à la spécification,
il faut des étapes d'extraction et/ou d'abstraction qui s'inscrivent dans une
stratégie ascendante (*bottom up*).

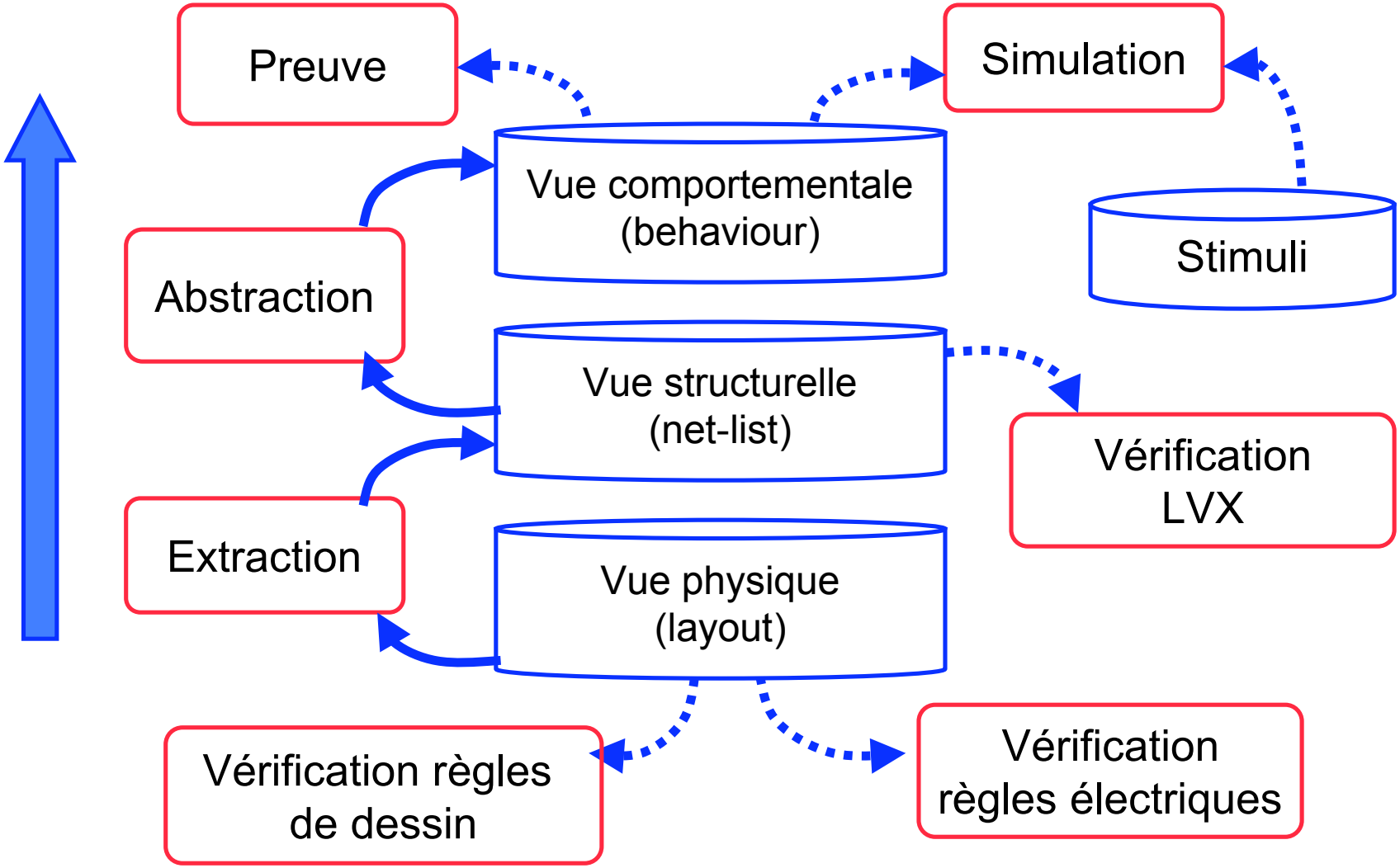
La définition du schéma par raffinement successif est un processus descendant,
mais les éléments terminaux du schéma (portes logiques) doivent nécessairement
appartenir à une (ou plusieurs) bibliothèques de cellules précaractérisées.

La méthode de conception est donc à la fois descendante et ascendante.

Conception descendante



Vérification « ascendante »



Spécification...

Règle absolue : Il faut spécifier avant de développer...

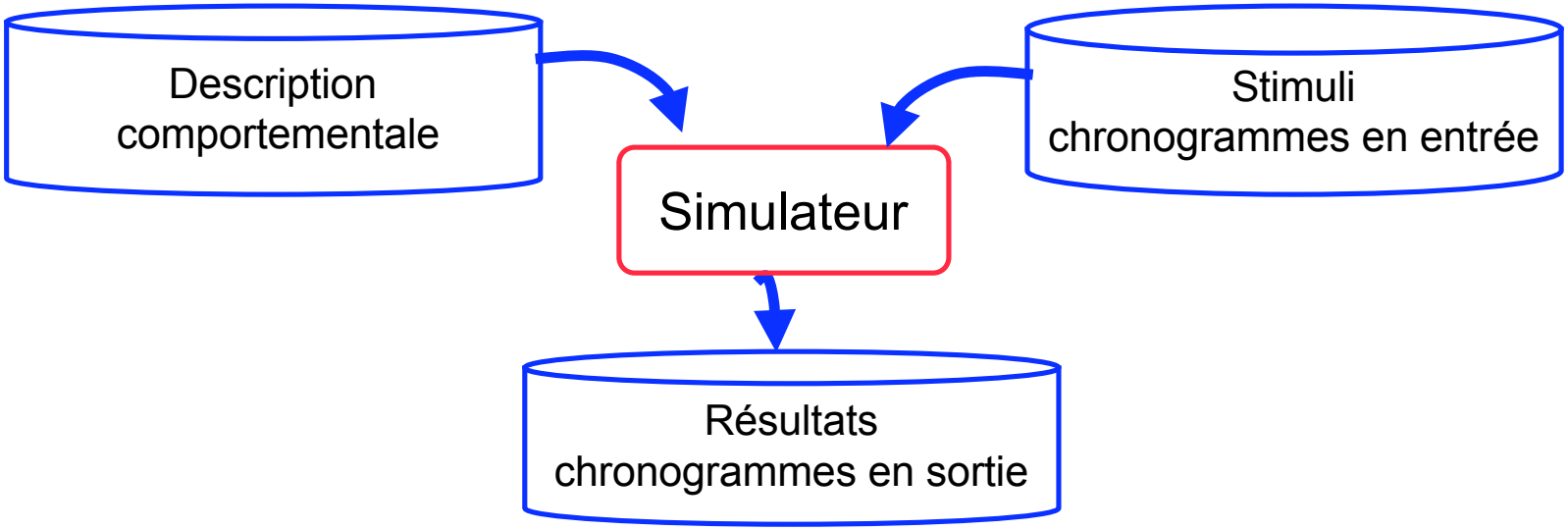
- Le point de départ de la synthèse physique est donc toujours la description comportementale, qui joue le rôle de spécification, pour la synthèse physique, et qui sera utilisée comme référence à toutes les étapes de conception ultérieures.

- Cette description comportementale peut être elle-même générée par des outils de synthèse de plus haut niveau (synthèse d'architecture, compilateurs de silicium), à partir de descriptions plus abstraites.

- Dans ce cours (et dans les TPs associés), on écrira la description comportementale « à la main ».

Validation / Simulation...

Le principal outil de validation de la description comportementale est la simulation, ce qui nécessite donc de développer les stimuli qui seront appliqués sur le modèle du composant à valider.



La qualité de la validation dépendant entièrement de la qualité des stimuli, la génération des stimuli peut prendre autant de temps que l'écriture du modèle du composant...

Simulation « Zéro-délay »

Au début du processus de conception, on ne connaît pas les temps de propagation des signaux (ces temps dépendent du schéma en portes qui sera généré par la synthèse, ainsi que du placement/routage).

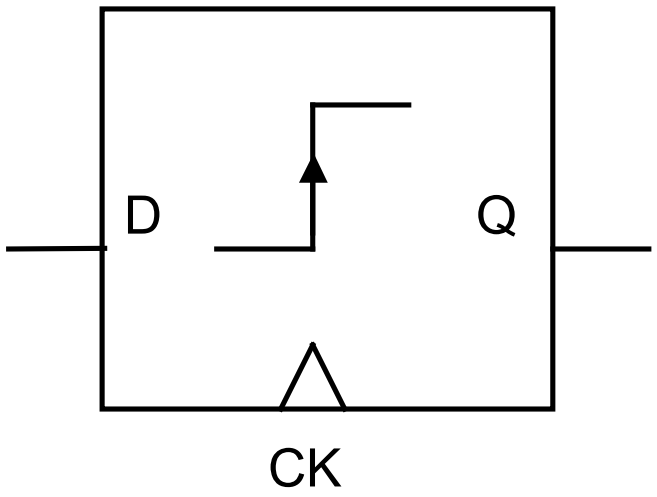
On écrit donc un modèle « zéro délay », où les seules informations temporelles sont apportées par les fronts successifs du signal d'horloge, car les temps de propagation dans la logique combinatoire sont supposés infiniment petits par rapport au temps de cycle.

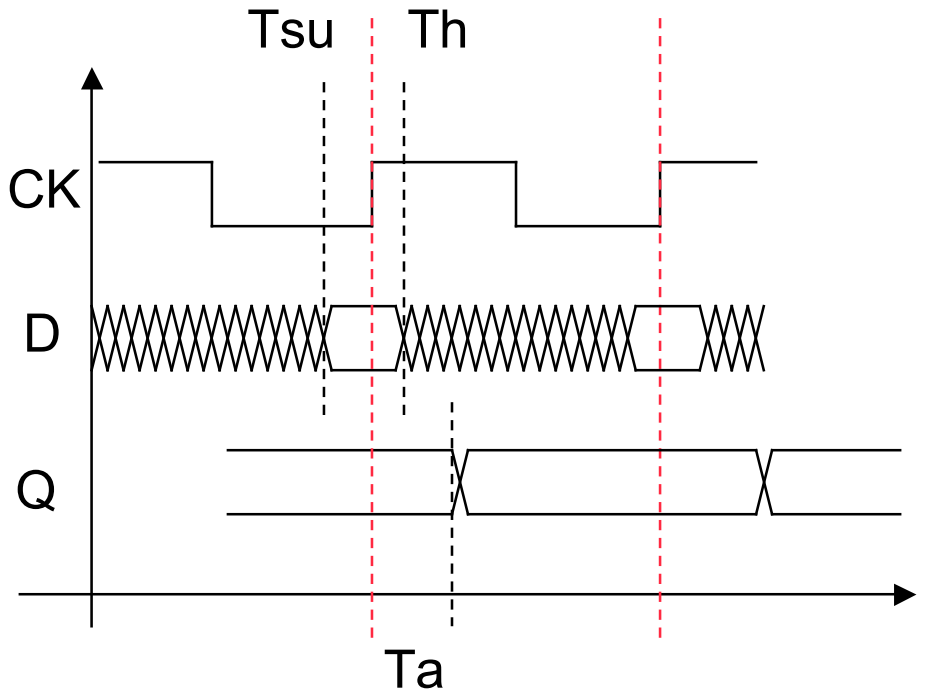
Les stimuli (c'est à dire les chronogrammes des signaux d'entrée) doivent donc être positionnés par rapport aux fronts du signal d'horloge.

Bascule D (D flip-flop)

Une bascule D permet de « mémoriser » 1 bit

- L'écriture d'une nouvelle valeur a lieu lors du front montant du signal CK
- L'entrée D doit être stable un peu avant (Tsu) et un peu après (Th) le front
- La sortie Q change de valeur au plus une fois par cycle après le front (Ta)

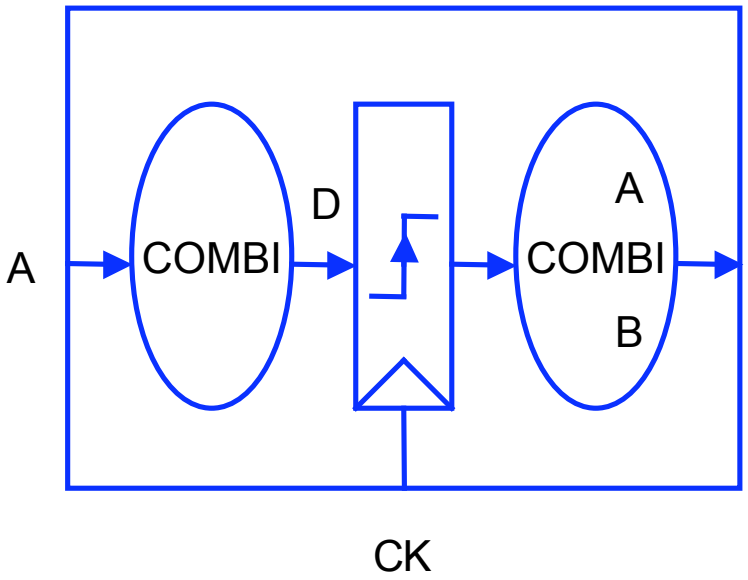




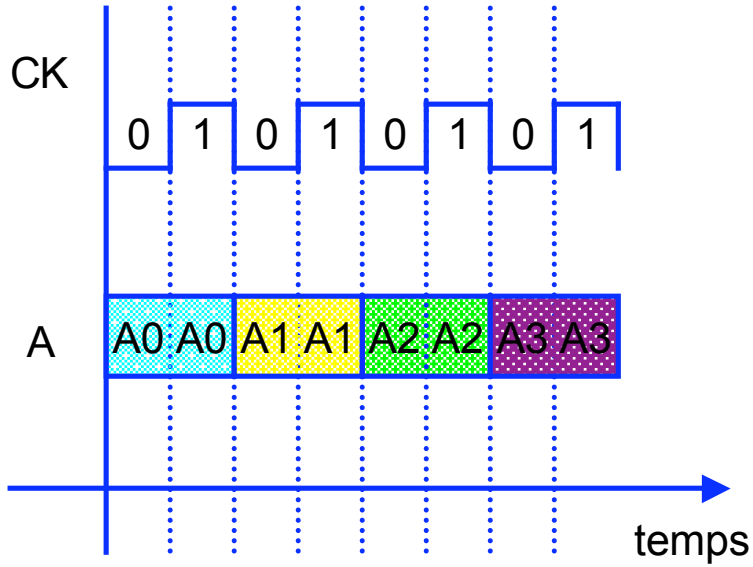
Structure temporelle des stimuli

A chaque port d'entrée est associé un un chronogramme.
Si le composant simulé ne contient que des registres à échantillonnage sur front (bascules D), on utilise un signal d'horloge CK à deux phases.
Les signaux d'entrée ne doivent pas changer de valeur au moment du front actif du signal CK.

Composant à simuler

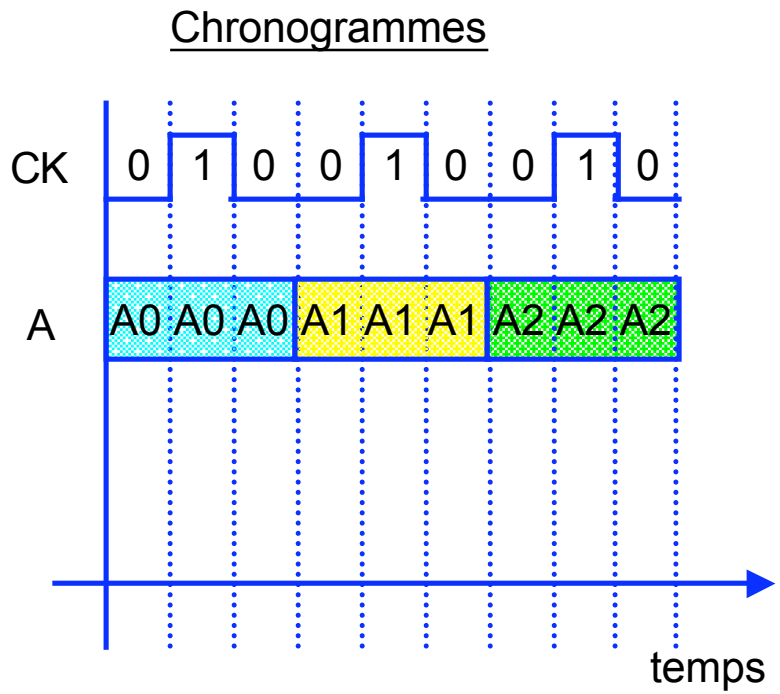


Chronogrammes



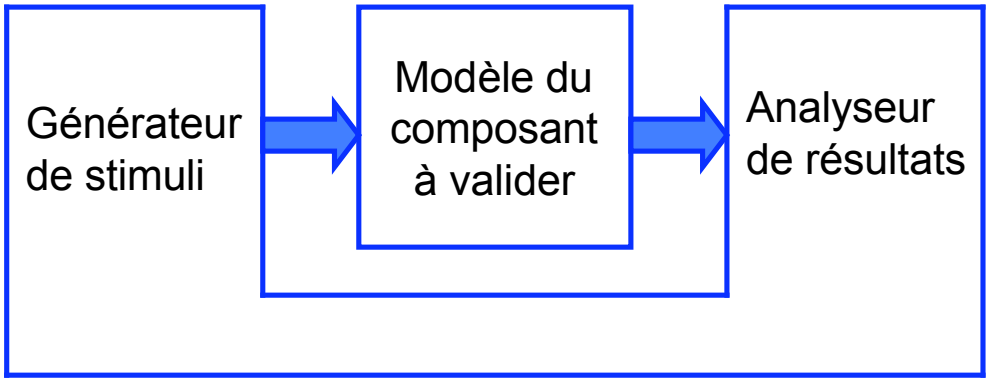
Cas des latchs

Si le composant simulé contient des latchs à échantillonnage sur état,
il faut utiliser un signal d'horloge CK à trois phases, car certains signaux
d'autorisation d'écriture doivent être stables AVANT le front montant de CK,
et rester stables jusqu' APRES le front descendant de CK.



Simulation : Banc de test

Un banc de test (test-bench) est un modèle comportemental de l'environnement dans lequel sera plongé le composant à valider.
La technique du banc de test permet d'éviter l'écriture - fastidieuse - des séquences de stimuli, et les - inévitables - erreurs humaines dans la phase d'analyse des résultats de simulation.



Attention : L'écriture du test-bench peut prendre autant de temps que l'écriture du modèle du composant à valider.
Suggestion : Le modèle du test-bench et le modèle du composant peuvent être développés en parallèle par deux personnes différentes.

Plan

- **Les trois « vues »**
- **Conception descendante**
- **Modélisation comportementale**
- **Exemple du composant « addaccu »**

Modélisation comportementale...

1. **Niveau « algorithmique »**
   - interface défini « au bit près » par une listes de signaux
   - pas de signal d'horloge explicite
   - pas d'identification des registres
   - sémantique séquentielle : un (ou plusieurs) processus
2. **Niveau « automate abstrait »**
   - interface défini « au bit près » par une liste de signaux
   - le signal d'horloge est explicite
   - les registres sont identifiés, mais les valeurs stockées (états) ne sont pas représentées au bit près.
   - sémantique séquentielle : un (ou plusieurs) processus
3. **Niveau « data-flow »**
   - interface défini « au bit près » par une liste de signaux
   - le signal d'horloge est explicite
   - les registres sont identifiés, et les valeurs stockées sont représentées au bit près.
   - sémantique « data-flow » : assignations concurrentes

Le niveau « RTL »

- Les descriptions comportementales de type 1 servent d'entrée aux outils de synthèse d'architecture, et ne seront pas considérées ici.

- Les descriptions comportementales de type 2 et 3 sont dites de niveau «RTL » (Register Transfer Level). Une description RTL est assez proche de la réalisation matérielle, puisqu'on peut décrire explicitement - cycle par cycle - la succession des états internes du composant (c'est à dire les valeurs stockées dans les registres).

- Le niveau d'abstraction RTL est fondamental, car il sert d'entrée aux outils de « synthèse physique », qui permettent d'automatiser la génération du dessin des masques (vue physique). Ce sont les différentes étapes de cette « synthèse physique », qui sont analysées dans ce cours.

Langages de description de matériel

On utilise généralement des langages spécialisés, appelés HDL (Hardware Description Language) pour décrire le comportement du matériel :

Différents objectifs :

- simulation logico-temporelle
- synthèse logique
- preuve (?)

Des caractéristiques communes :

- Expression du parallélisme du matériel
- Descriptions mixtes comportementale et structurelles
- support de types spécialisés (vecteurs de bits)
- Représentation explicite du temps

Principaux HDLs :

VHDL, VERILOG, SYSTEMC

Langage VHDL

Un modèle VHDL d'un composant matériel comporte deux parties :

- La partie « Entity » décrit l'interface du composant.
- La partie « Architecture » décrit son comportement, ou sa structure interne.

Il peut exister plusieurs architectures pour un même composant, correspondant à différents niveaux d'abstraction :

- description comportementale « algorithmique »
- description comportementale « automate abstrait »
- description comportementale « data-flow »
- description structurelle

Attention : Le langage VHDL permettant de représenter différents niveaux d'abstraction d'un circuit, l'expression « modèle VHDL » sans qualificatif est extrêmement ambigüe, et doit être évitée...

VHDL : assignations concurrentes

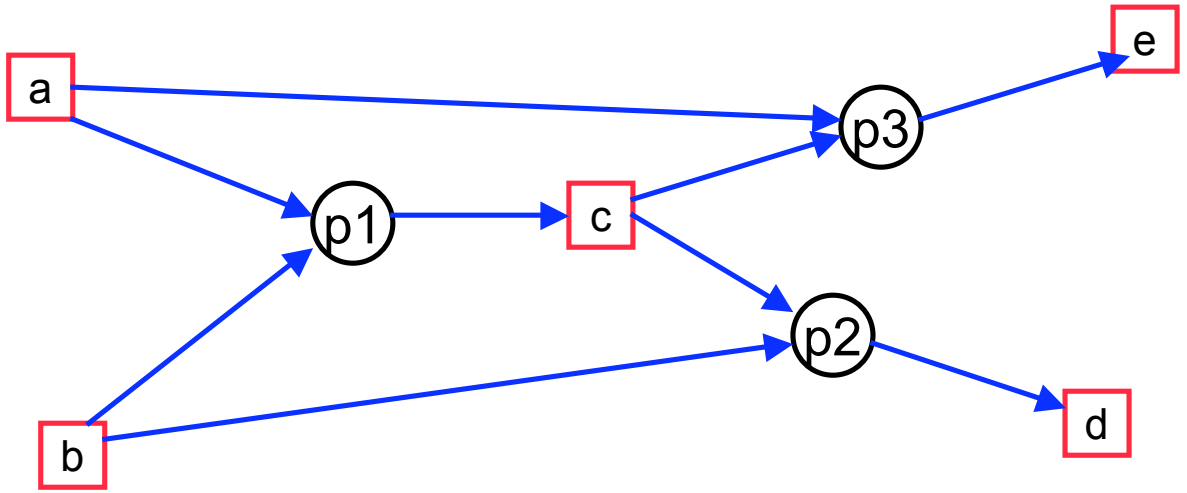
Un modèle VHDL est dit « data-flow » s'il ne comporte que des assignations concurrentes : Le circuit est décrit comme un ensemble de « processus » interconnectés par des signaux, qui s'exécutent en parallèle.

- Chaque assignation correspond à un processus.
- L'ordre d'écriture des assignations concurrentes est sans signification.
- un signal ne peut être assigné qu'une seule fois (assignation unique).
- Une description comportementale de type « data-flow » permet de décrire le parallélisme intrinsèque d'un réseau Booléen.

Réseau Booléen

Un réseau Booléen est un graphe orienté biparti (deux types de noeuds):

- Le premier type de noeud (carré rouge) représente les « signaux ».
- Le deuxième type de noeud (rond noir) représente les « processus ».
- Une variable booléenne est associée à chaque signal.
- Une Fonction Booléenne permettant de calculer la valeur du signal de sortie en fonction de la valeur des signaux d'entrée est associée à chaque processus.



Exemple

description VHDL de type « assignations concurrentes »

```
z <= x or y ;
nc <= not c ;
x <= a and c ;
y <= b and nc ;
```

réseau Booléen

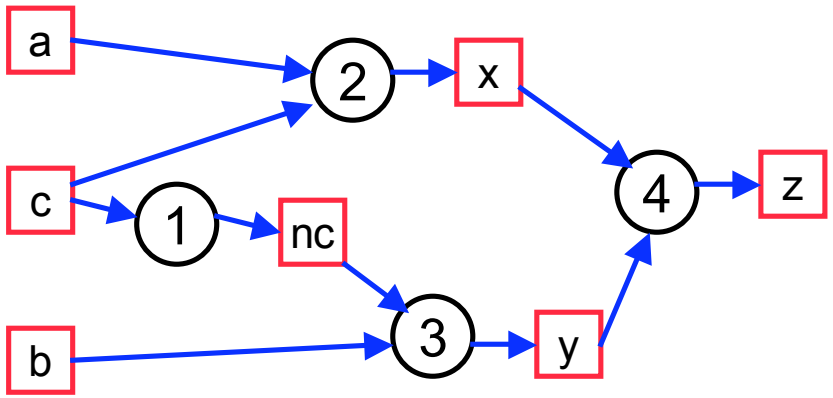
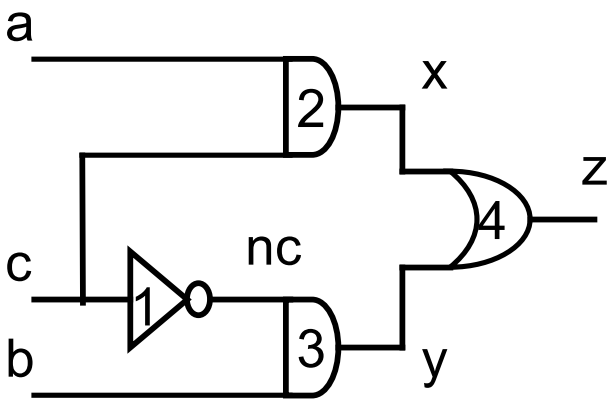


schéma en portes

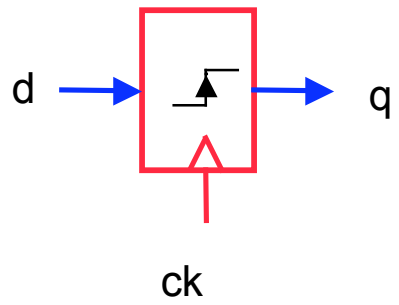


Représentation des registres

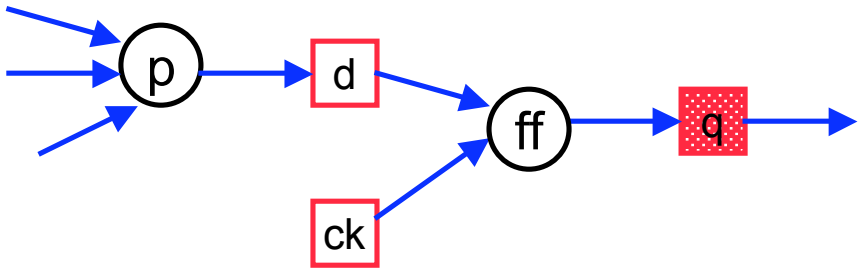
Le réseau Booléen associé à un opérateur combinatoire est un DAG (graphe orienté acyclique), car on ne doit pas trouver de dépendances cycliques dans un opérateur combinatoire.

Si le circuit modélisé contient des registres, les variables Booléennes stockées dans les registres sont représentées par des signaux d'un type particulier, puisque - dans un circuit synchrone - l'écriture dans les registres est conditionnée par le front du signal d'horloge.

bascule D



réseau Booléen

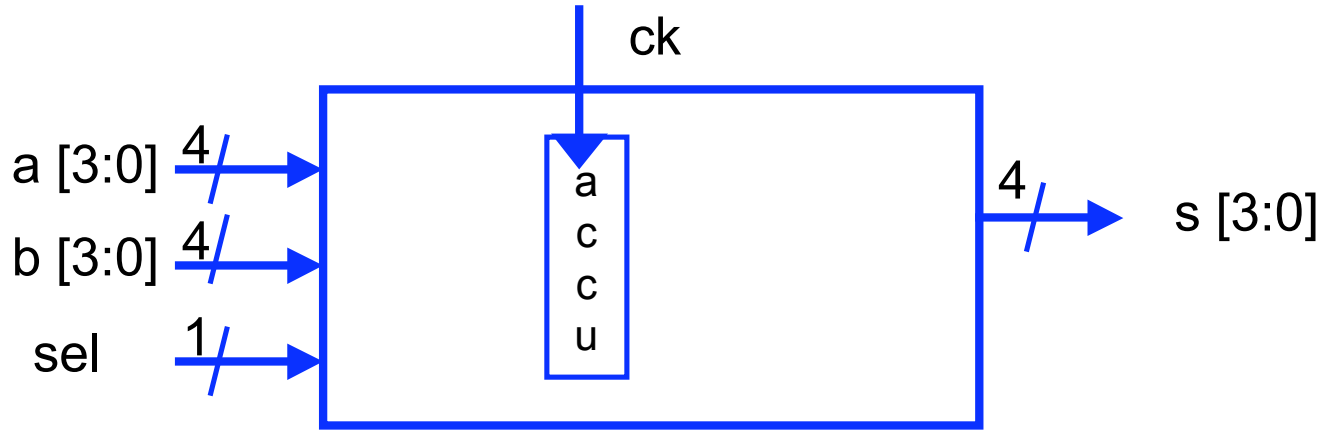


Plan

- **Les trois « vues »**
- **Conception descendante**
- **modélisation comportementale**
- **Exemple du composant « addaccu »**

Exemple : addaccu

Le composant addaccu est décrit comme une boîte « grise » possédant 4 signaux d'entrée : a, b, sel, ck (10 bits au total) et un seul signal de sortie : s (4 bits).



On souhaite le comportement suivant :

- **Si (sel == 0)** **sum <= a + b**
  Sinon **sum <= accu + b**
- **s <= sum**
- **Quand (ck passe de 0 à 1)** **accu <= sum**

Modélisation VHDL addaccu : Entity

```
entity addaccu is
-- liste des ports d'entrée/sortie
port (
     a : in bit_vector ( 3 downto 0 ) ;
     b : in bit_vector ( 3 downto 0 ) ;
     sel : in  bit ;
     ck : in bit ;
     s : out bit_vector (3 downto 0 ) ;
     vss : in  bit ;
     vdd : in bit
     )
end addaccu ;
```

VHDL ne fait pas de distinction entre majuscules et minuscules.
Les commentaires commencent par « -- ».
Dans l'exemple proposé ici, les mots-clés et les séparateurs du langage VHDL sont en rouge.

Modélisation VHDL addaccu : Architecture vbe

```
Architecture vbe of addaccu is

-- déclaration des signaux
    signal accu     : reg_vector ( 3 downto 0 ) ;
    signal x        : bit_vector ( 3 downto 0 ) ;
    signal sum      : bit_vector ( 3 downto 0 ) ;
    signal r        : bit_vector ( 3 downto 0 ) ;

-- description du comportement
    begin
    -- selection du deuxième opérande
    with sel select
          x[3 downto 0] <=   a[3 downto 0] when '0' ,
                             accu[3 downto 0] when '1' ;
    ...
```

Modélisation VHDL addaccu : Architecture vbe

```
    -- calcul de la somme ... et du report
    sum[3 downto 0] <= b[3 downto 0] xor x[3 downto 0] xor r[3 downto 0] ;
    r[3 downto 1]     <=      ( b[2 downto 0] and x[2 downto 0] )  or
                              ( b[2 downto 0] and r[2 downto 0] ) or
                              ( x[2 downto 0] and r[2 downto 0] ) ;
    r[0] <= '0' ;

    -- écriture dans le registre
    reg :  block ( ck = '1' and not ck'stable )
           begin
           accu[3 downto 0] <= guarded sum[3 downto 0] ;
           end block ;

    -- affectation signal de sortie
    s[3 downto 0] <= sum[3 downto 0] ;

end vbe ;
```

Justification du calcul de la somme