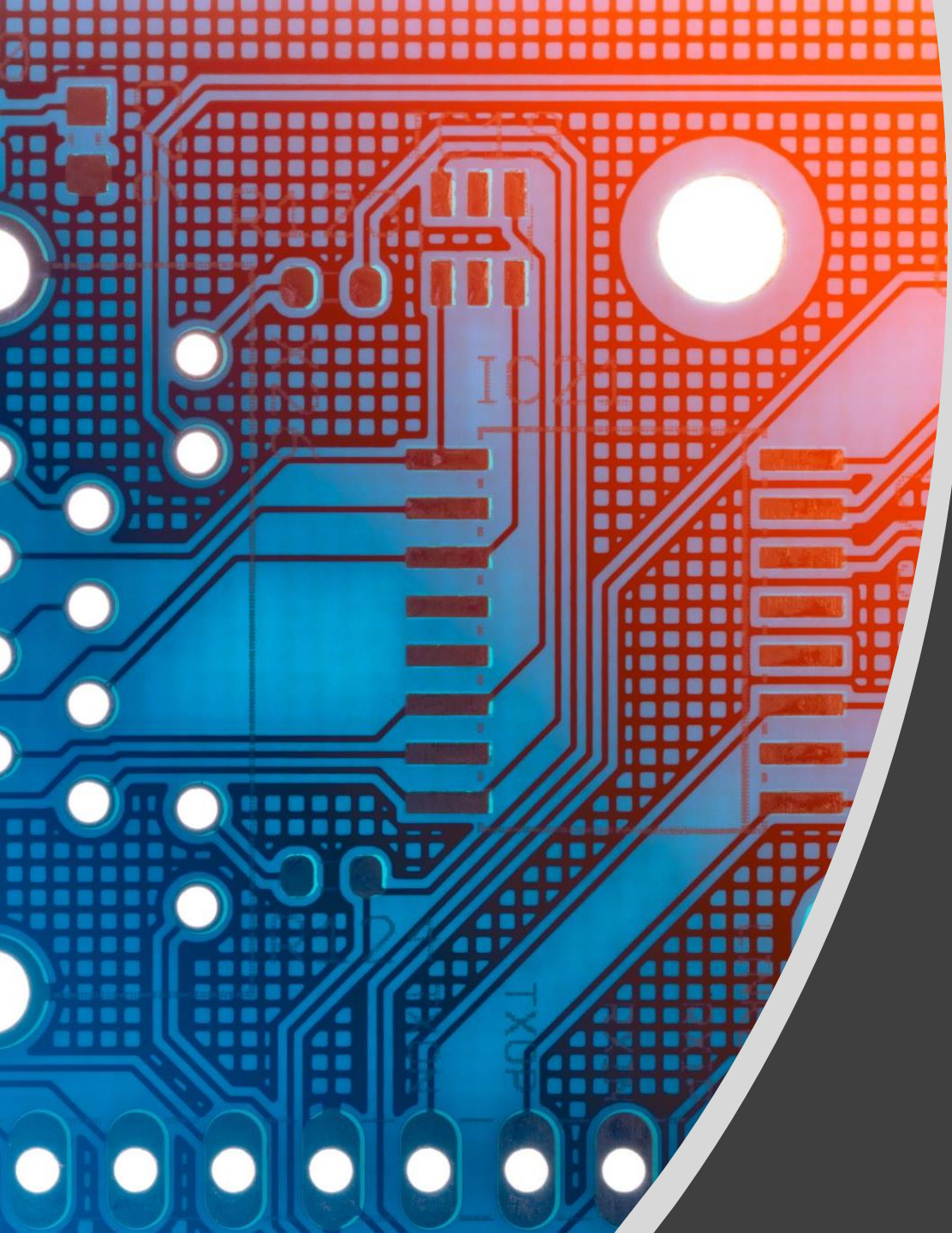


La couche circuits logiques

Architecture des ordinateurs

Couche logique numérique

- Les objets considérés à ce niveau sont les portes logiques, chacune construite à partir de quelques transistors
- Chaque porte prend en entrée des signaux numériques (0 ou 1) et calcule en sortie une fonction logique simple (ET, OU, NON)
- De petits assemblages de portes peuvent servir à réaliser des fonctions logiques telles que mémoire, additionneur, ainsi que la logique de contrôle de l'ordinateur

Circuits logiques Combinatoires

Circuit logique

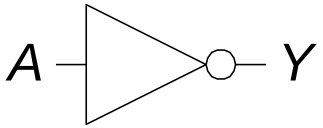
- Un circuit logique est un circuit dans lequel seules 2 valeurs logiques sont possibles
  - 0 ou 1
- En pratique : circuit électrique (transistors) dans lequel une faible tension représente le signal 0 alors qu'une tension élevée correspond au signal 1
- Composants de base : les portes logiques qui permettent de combiner ces signaux binaires

Portes Logiques

- Une porte logique est un composant qui reçoit en entrée une ou plusieurs valeurs binaires et renvoie en sortie une unique valeur binaire en implémentant une fonction logique:
  - inversion (NOT), AND, OR, NAND, NOR, etc.
- Une valeur en entrée:
  - Porte NOT, tampon
- 2-entrées:
  - AND, OR, XOR, NAND, NOR, XNOR
- Plusieurs entrées

Portes logiques d'une entrée

NOT

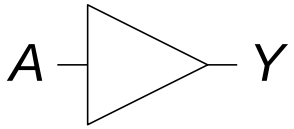


$Y = \bar{A}$

A	Y
0	1
1	0

Tampon

BUF

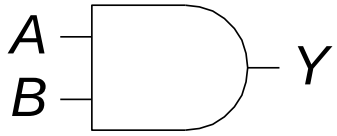


$Y = A$

A	Y
0	0
1	1

Portes logiques de deux entrées

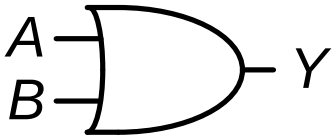
AND



$Y = AB$

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

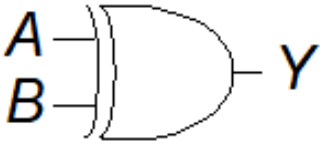
OR



$Y = A + B$

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

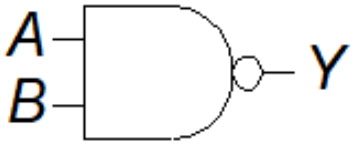
XOR



$Y = A \oplus B$

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

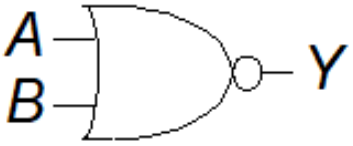
NAND



$Y = \overline{AB}$

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

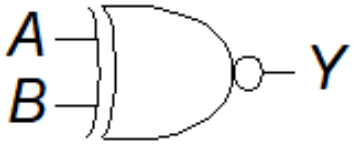
NOR



$Y = \overline{A + B}$

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

XNOR



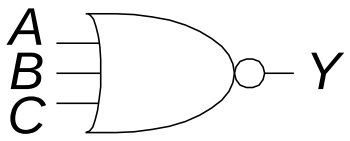
$Y = \overline{A \oplus B}$

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1

Portes logiques de deux entrées

Portes logiques à plusieurs entrées

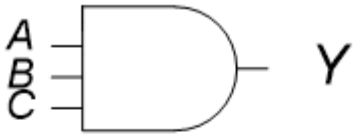
NOR3



$Y = \overline{A+B+C}$

A	B	C	Y
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

AND3



$Y = ABC$

A	B	C	Y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

- XOR multi-entrées : Parité impaire

Niveaux de tension

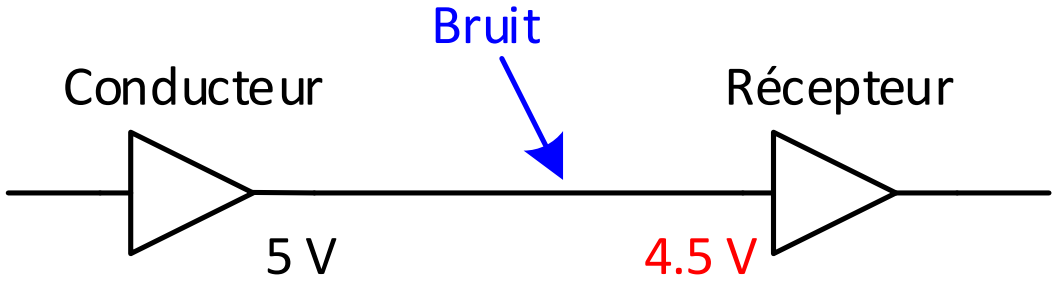
- Définir les tensions discrètes (voltages) pour représenter 1 et 0
  - 0 logique/booléen = 0 binaire (nombre) = absence de courant électrique (ou une tension -V)
  - 1 logique/booléen = 1 binaire (nombre) = tension +V
- Par exemple, on pourrait associer:
  - 0 binaire avec la masse (ground) ou 0 volts
  - 1 binaire avec V_{DD} (source électrique, « power supply ») ou 5 volts
- Comment interpréter 4.99 volts? Comme 0 ou 1?
- Et 3.2 volts?

Niveaux de tension

- Définir un intervalle de voltage pour représenter 1 et 0
- Définir les intervalles de voltage différents pour les entrées et les sorties pour prendre en compte de bruit existant dans tout système
- Qui-est ce qu'un bruit?

Qui-est ce qu'un bruit?

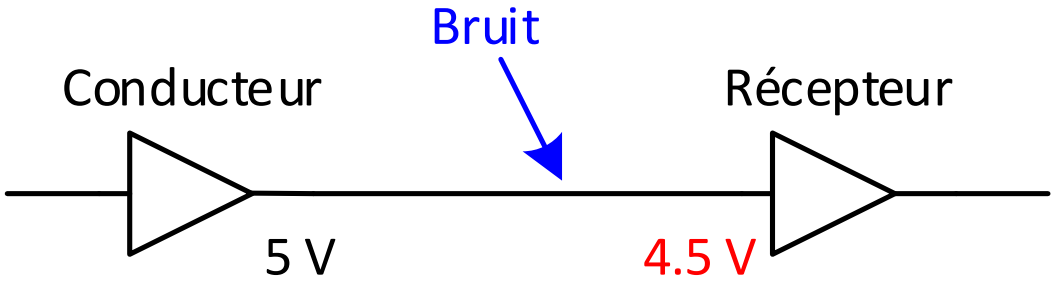
- Un signal est toujours affecté par de petites fluctuations plus ou moins importantes
- **Un bruit - Quelque chose qui affaiblit le signal**
  - E.g., résistance, bruit de la source électrique, couplage de fils voisins, etc.
- **Exemple:** une porte(conducteur) pourrait transmettre un signal de 5 volts mais à cause de la résistance de long fil, le signal arriverait dégradé - 4.5 volts

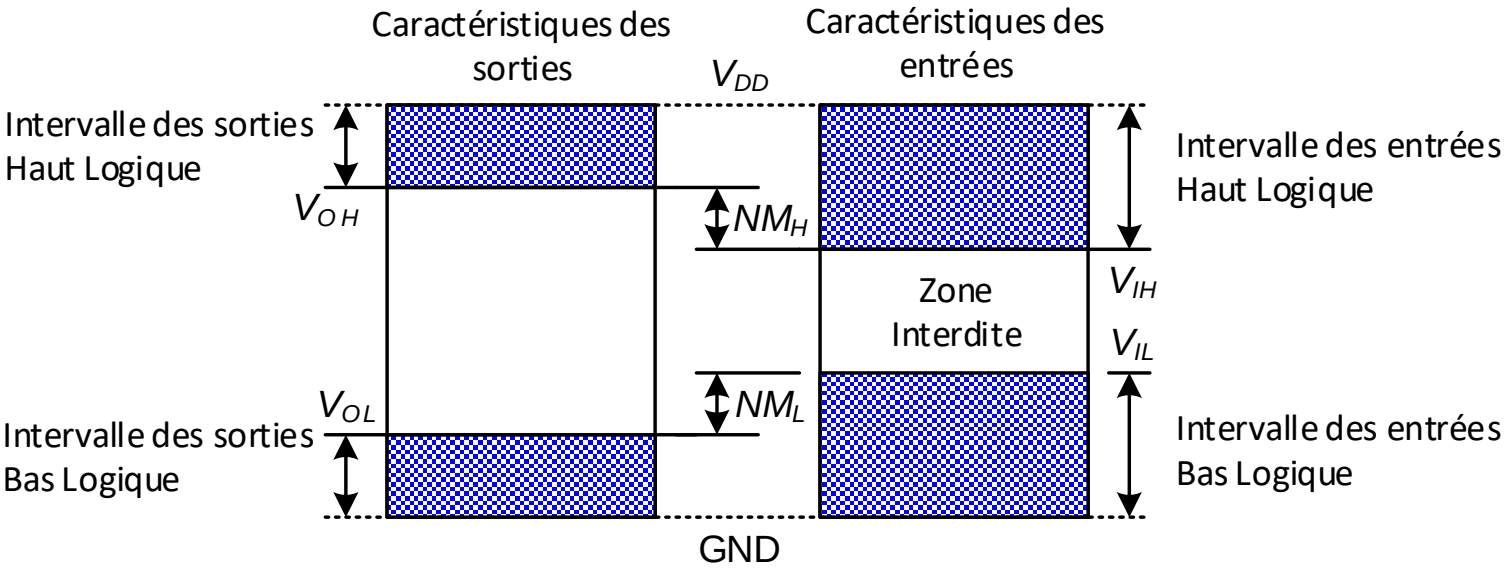


Exigences

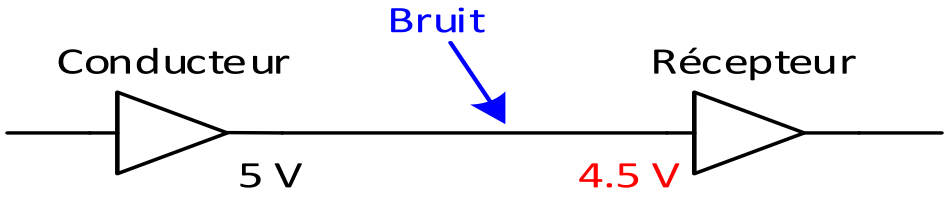
- Entrées logiques valides de tous les éléments de circuit doivent produire les sorties valides
- Pour représenter les valeurs discrètes il faut utiliser les intervalles de voltages spécifiés et bornés

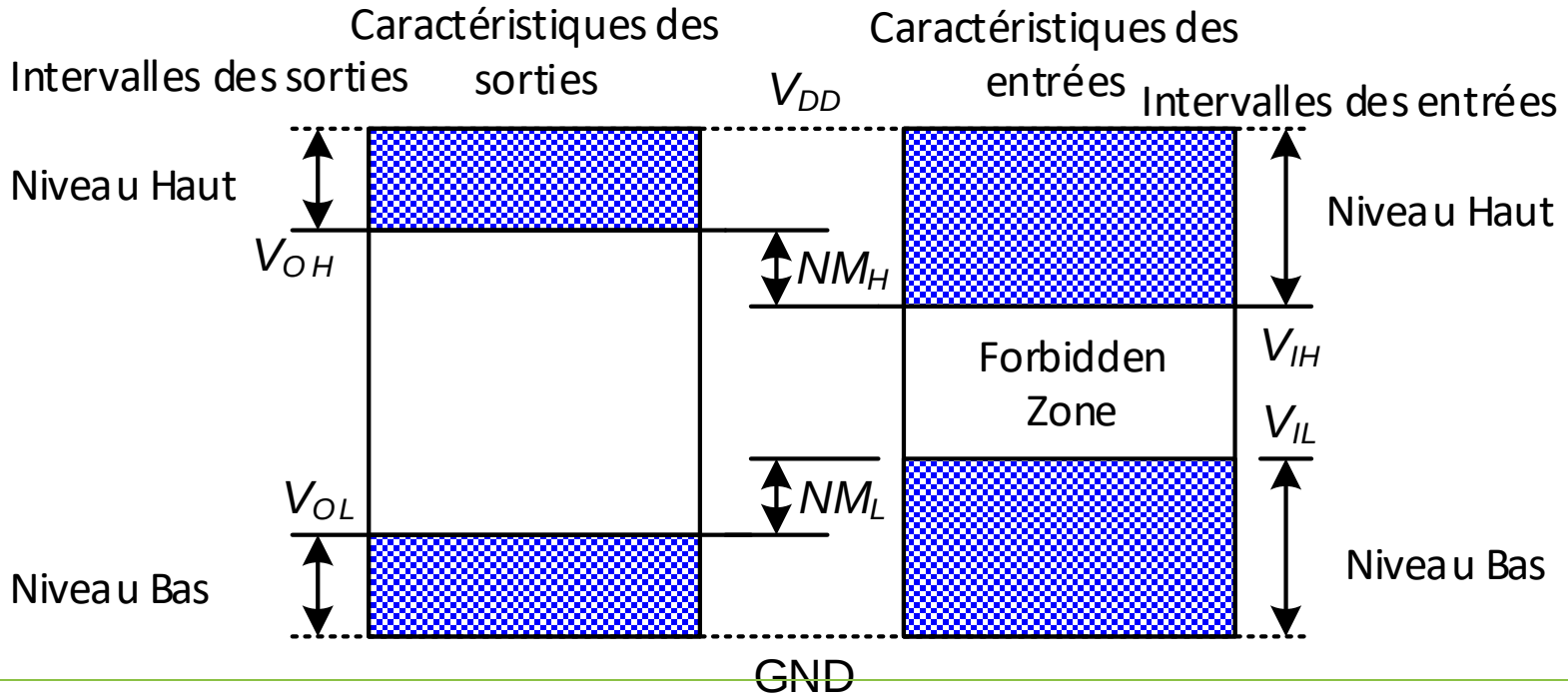
Niveaux logiques





Marges de bruit





Marge de bruit ("Noise Margin") $NM_H = V_{OH} - V_{IH}$

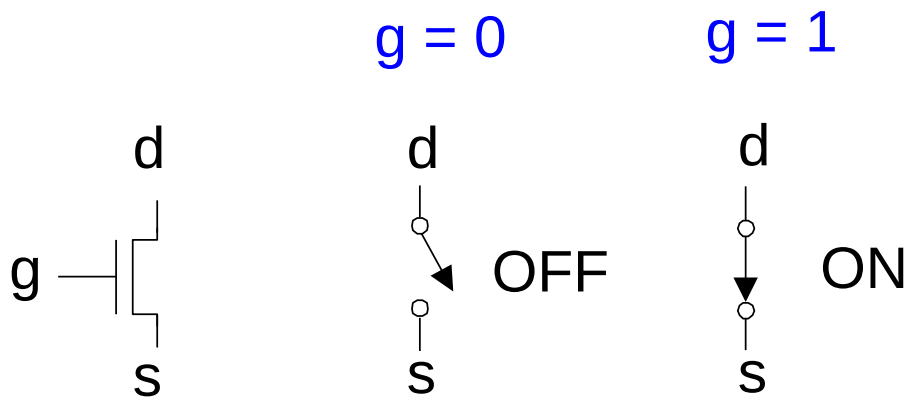
$NM_L = V_{IL} - V_{OL}$

Tension de fonctionnement

- La tension de fonctionnement des circuits a été abaissée
- De 5V pour les premières générations à 0.9V maintenant
- 5V …, 3.3 V, 2.5 V, 1.8 V, 1.5 V, 1.2 V, 1.0 V, 0.9 V
- Il n’est plus possible de diminuer la tension avec les technologies actuelles
  - Bruit thermique causerait trop d’erreurs

Transistors

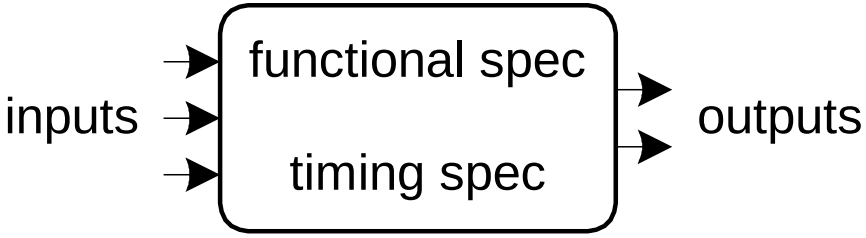
- Transistor est la brique avec laquelle sont construits les circuits électroniques
  - possède trois broches appelées drain, grille et source
  - se comporte comme un interrupteur électrique entre la source et le drain qui serait commandé par la grille.



Introduction

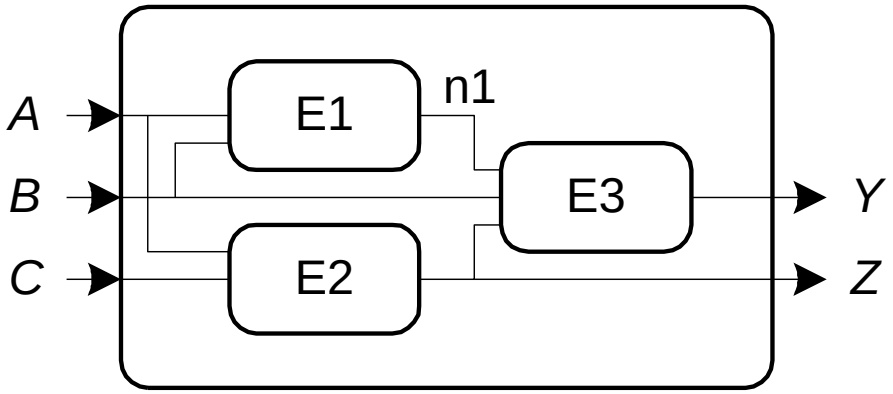
Un circuit logique est composé:

- Entrées
- Sorties
- Spécification fonctionnelle
- Spécification temporelle



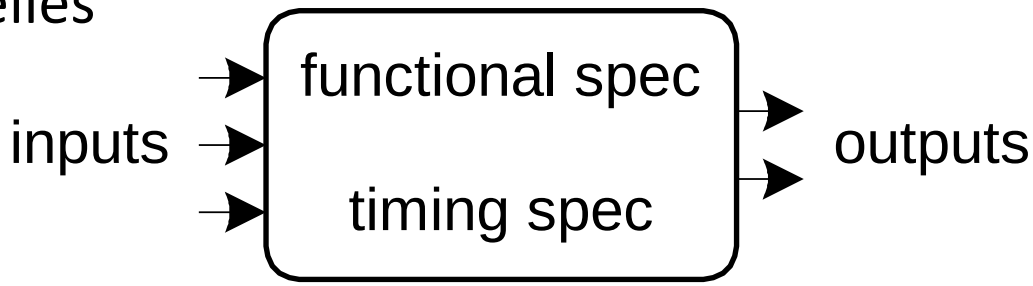
Circuits

- Nœuds
  - Entrées: A, B, C
  - Sorties: Y, Z
  - Internes: n1
- Éléments circuit
  - E1, E2, E3
  - Chacun un circuit



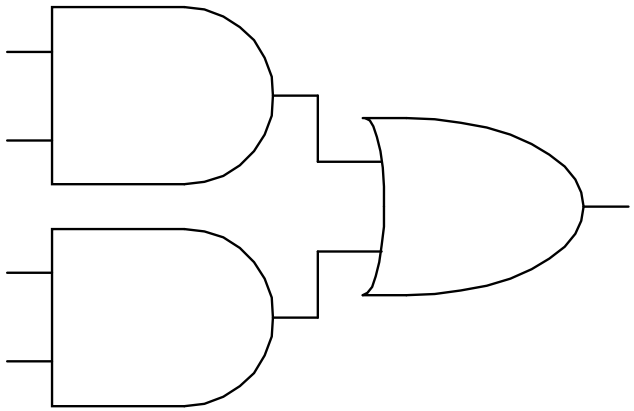
Types des circuits logiques

- Logique combinatoires
  - Sans mémoire
  - Sorties sont définies par les valeurs des entées actuelles
- Logique séquentielle
  - Possède une mémoire
  - Sorties sont définies par les entrées antérieures et actuelles



Règles de la composition combinatoire

- Chaque élément d'un circuit est un circuit combinatoire
- Chaque nœud du circuit est soit une entrée, soit est connecté à une seule sortie d'élément
- Le circuit ne contienne aucun chemin cyclique
- Exemple:

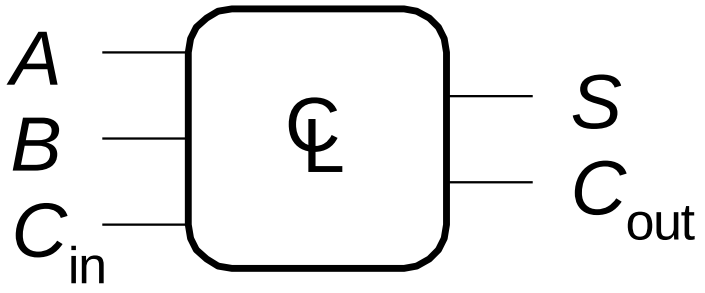


Équations Booléennes

- Pour décrire les circuits réalisables en combinant des portes logiques, on a besoin d'une algèbre opérant sur les variables 0 et 1
- Une fonction booléenne (logique) à une ou plusieurs variables est une fonction qui renvoie une valeur ne dépendant que de ces variables.
- Exemple:

$S = F(A, B, C_{in})$

$C_{out} = F(A, B, C_{in})$



$$S = A \oplus B \oplus C_{in}$$

$$C_{out} = AB + AC_{in} + BC_{in}$$

Table de vérité

- Une fonction logique à n variables possède seulement 2^n combinaisons d'entrées possibles
- On peut définir une fonction logique par une table à 2^n lignes donnant la valeur de la fonction pour chaque combinaison d'entrée
  - *Table de vérité de la fonction*

Définitions

- Complément: variable avec une barre au-dessus
  - $A, \bar{A}, B, \bar{B}, C, \bar{C}$
- Impliquant: produit de littéraux
  - $ABC, B\bar{C}, A\bar{B}$
- Minterm: produit qui inclue toutes les variables entrées
  - $F(A, B, C) = ABC, AB\bar{C}, CA\bar{B}\ldots$.
- Maxterm : ou logique qui inclue toutes les variables entrées
  - F(A, B, C) = A + B + C, $A + \bar{B} + C$, etc.

Somme-de-Produits (SOP)

- Toute fonction peut être décrite en spécifiant lesquelles des combinaisons d'entrée donnent 1
- Avec chaque ligne on peut associer un minterme unique égale à 1
- Un minterm est un produit (ET) de littéraux
- On peut, donc, représenter une fonction logique comme le OU logique de mintermes sur les combinaisons d'entrée donnant 1

	A	B	Y	minterm
0_{10} ←	0	0	0	$\overline{A}\,\overline{B}$
1_{10} ←	0	1	1	$\overline{A}\,B$
2_{10} ←	1	0	0	$A\,\overline{B}$
3_{10} ←	1	1	1	$A\,B$

$$Y(A,B) = \bar{A}B + AB \Leftrightarrow Y(A,B) = \sum(1.3)$$

Produits-de-sommes (POS)

- Toutes les équations booléennes peuvent être décrites en forme POS
- Un maxterm est une somme (OU) de littéraux
- Avec chaque ligne on peut associer un maxterme unique égale à 0
- On peut, donc, représenter une fonction logique comme le ET logique de maxtermes sur les combinaisons d'entrée donnant 0

A	*B*	*Y*	maxterm
0	0	0	$A + B$
0	1	1	$A + \overline{B}$
1	0	0	$\overline{A} + B$
1	1	1	$\overline{A} + \overline{B}$

$$Y(A, B) = (A + B)(A + \overline{B})$$

Équations Booléennes, Exemple

- Les formulations SOP et POS fournissent une méthode directe pour implanter n'importe quelle fonction logique
- Vous allez à la cafétéria pour un diner
  - Vous ne mangez pas (La variable M = Zéro)
    - Si la cafétéria n'est pas ouverte (O = Zéro) ou
    - Si on vous offre des saucisses sur bâtonnet (P = 1 "corndogs")
- Écrire une table de vérité de la fonction de prédiction si vous mangez ou pas (E).

O	*P*	*M*
0	0	
0	1	
1	0	
1	1	

.

Équations Booléennes, Exemple

- Vous allez à la cafétéria pour un diner
  - Vous ne mangez pas (M = 0)
    - Si la cafétéria n'est pas ouverte (O = 0) ou
    - Si on vous offre des saucisses sur bâtonnet (P = 1 "corndogs")
- Écrire une table de vérité de la fonction de prédiction si vous mangez ou pas (M).

O	*P*	*M*
0	0	0
0	1	0
1	0	1
1	1	0

Formes SOP & POS

- SOP – somme-de-produits

O	*P*	*M*	minterm
0	0		$\overline{O}\,\overline{P}$
0	1		$\overline{O}\,P$
1	0		$O\,\overline{P}$
1	1		$O\,P$

- POS – produit-de-sommes

O	*P*	*M*	maxterm
0	0		$O + P$
0	1		$O + \overline{P}$
1	0		$\overline{O} + P$
1	1		$\overline{O} + \overline{P}$

Formes SOP & POS

- SOP – somme-de-produits

O	*P*	*M*	minterm
0	0	0	$\overline{O}\,\overline{P}$
0	1	0	$\overline{O}\,P$
1	0	1	$O\,\overline{P}$
1	1	0	$O\,P$

$$M = O\overline{P}$$

- POS – produit-de-sommes

O	*P*	*M*	maxterm
0	0	0	$O + P$
0	1	0	$O + \overline{P}$
1	0	1	$\overline{O} + P$
1	1	0	$\overline{O} + \overline{P}$

$$M = (O + P)(O + \overline{P})(\overline{O} + \overline{P})$$

Simplification de l'implantation

- Deux fonctions sont équivalentes si et seulement si leurs tables de vérité sont identiques
- Il est intéressant d'implanter une fonction avec le moins de portes possibles
  - Économie de place sur le processor
  - Réduction de la consommation électrique
  - Réduction du temps de parcours du signal

Simplification de l'implantation

- Il est possible de simplifier les formes canoniques SOP (ou POS)
- Nous verrons trois méthodes
  - Réduction algébrique
  - Tables de Karnaugh
  - Méthode tabulaire de Quine-McCluskey

Algèbre booléenne

- Pour réduire la complexité des fonctions booléennes, on essaye d'appliquer des identités simplificatrices à la fonction initiale
- Un ensemble d'identités de l'algèbre booléenne
- Chaque loi a deux formes, qui sont duales si on échange les rôles respectifs de ET, OU et de 0 et 1

Identités booléennes

	Axiom		Dual	Name
A1	$B = 0$ if $B \neq 1$	A1′	$B = 1$ if $B \neq 0$	Binary field
A2	$\overline{0} = 1$	A2′	$\overline{1} = 0$	NOT
A3	$0 \bullet 0 = 0$	A3′	$1 + 1 = 1$	AND/OR
A4	$1 \bullet 1 = 1$	A4′	$0 + 0 = 0$	AND/OR
A5	$0 \bullet 1 = 1 \bullet 0 = 0$	A5′	$1 + 0 = 0 + 1 = 1$	AND/OR

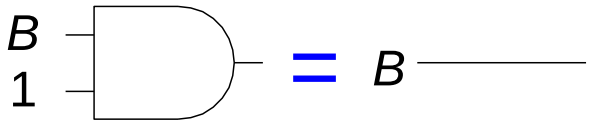
	Theorem		Dual	Name
T1	$B \bullet 1 = B$	T1′	$B + 0 = B$	Identity
T2	$B \bullet 0 = 0$	T2′	$B + 1 = 1$	Null Element
T3	$B \bullet B = B$	T3′	$B + B = B$	Idempotency
T4		$\overline{\overline{B}} = B$		Involution
T5	$B \bullet \overline{B} = 0$	T5′	$B + \overline{B} = 1$	Complements

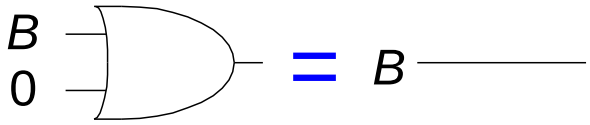
T1: Identité

- $B \cdot 1 =$
- $B + 0 =$

T1: Identité

- $B \bullet 1 = B$
- $B + 0 = B$



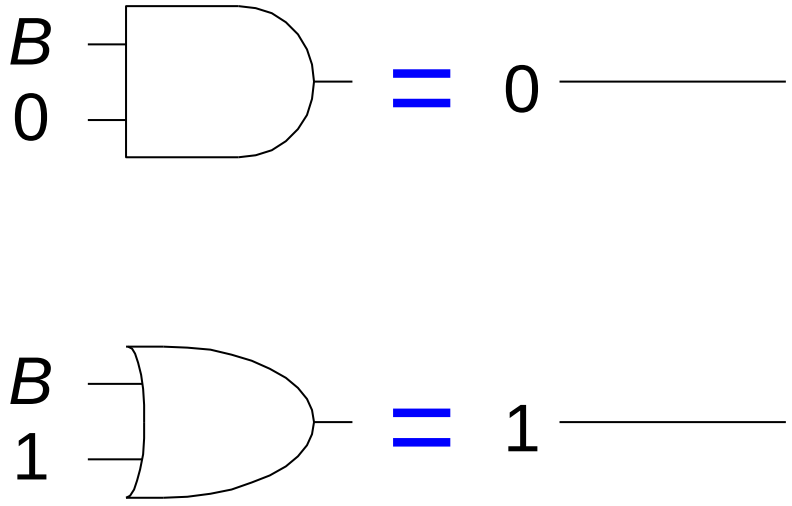


T2: Nul

- $B \bullet 0 =$
- $B + 1 =$

T2: Nul

- $B \bullet 0 = 0$
- $B + 1 = 1$

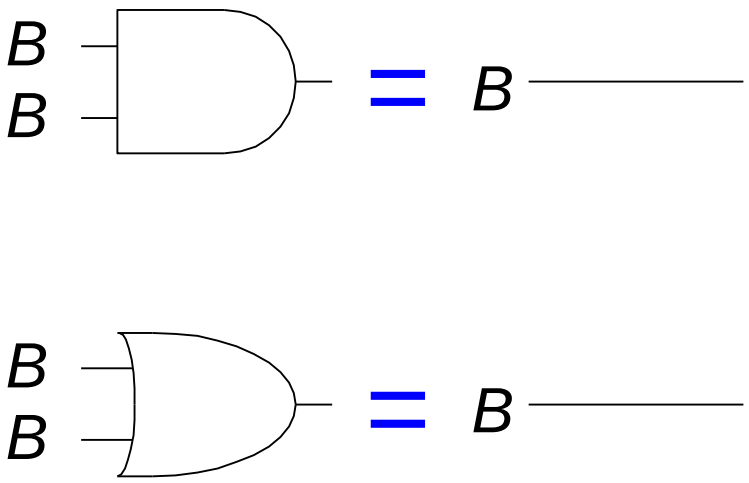


T3: Idempotence

- $B \bullet B =$
- $B + B =$

T3: Idempotence

- $B \bullet B = B$
- $B + B = B$

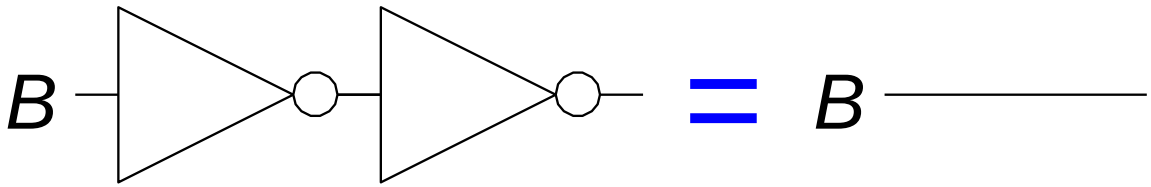


T4: Involution

- $\overline{\overline{B}} =$

T4: Involution

- $\overline{\overline{B}} = B$

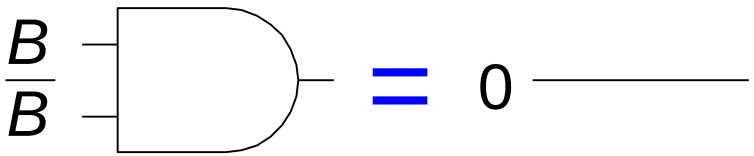


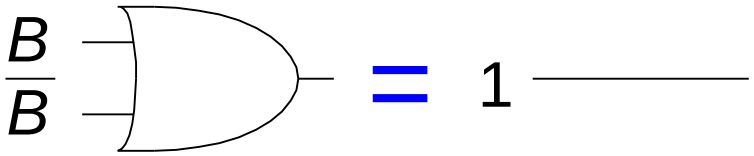
T5: Complémentation

- $B \bullet \overline{B} =$
- $B + \overline{B} =$

T5: Complémentation

- $B \cdot \overline{B} = 0$
- $B + \overline{B} = 1$





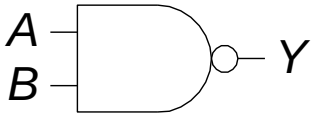
Réduction algébrique: Exemples

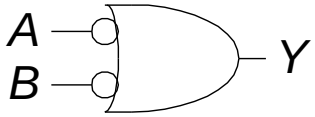
- $Y = \bar{A}B + AB = B(\bar{A} + A) = B \cdot 1 = B$
- $Y = A(AB + ABC) = A(AB(1 + C)) = A\big(AB(1)\big) =$

$$= A(AB) = AB$$

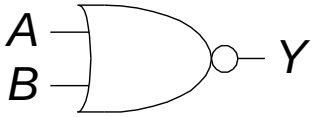
Lois de De Morgan

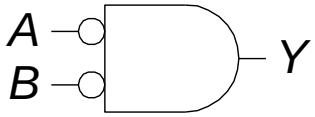
- $Y = \overline{AB} = \overline{A} + \overline{B}$





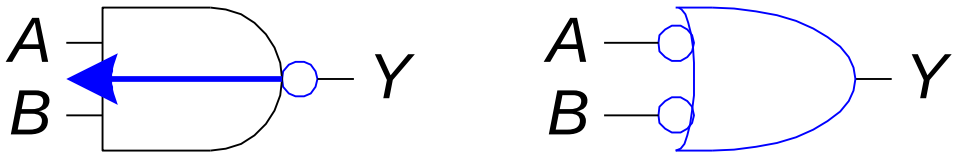
- $Y = \overline{A + B} = \overline{A} \bullet \overline{B}$



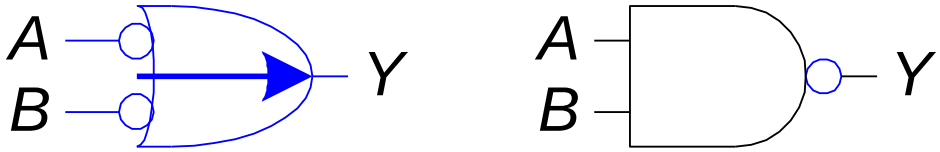


Propagation de bulles

- Propagation de bulles entrées-sorties ou sorties-entrées
  - change la forme de la porte OU vers la porte ET et vice versa
  - Déplace les bulles de sorties vers les entrées lors de la propagation sortie-entrées

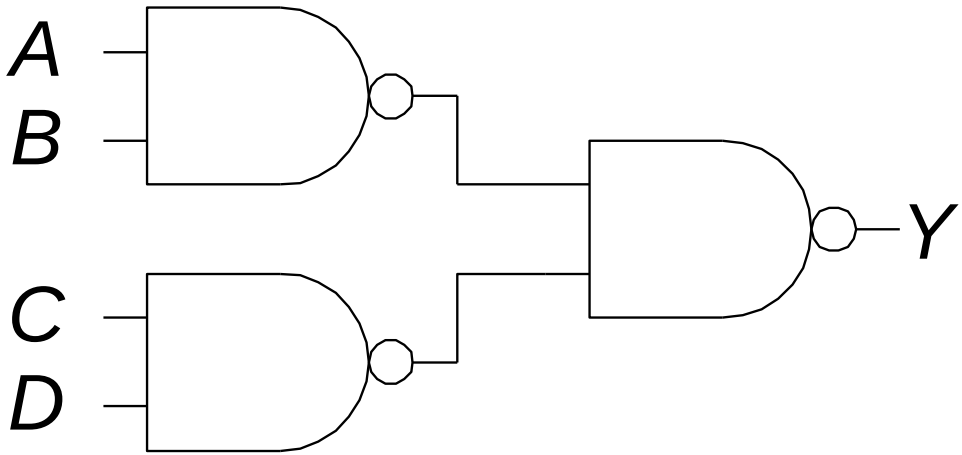


  - Déplace les bulles des entrées en sortie lors de la propagation entrées-sorties



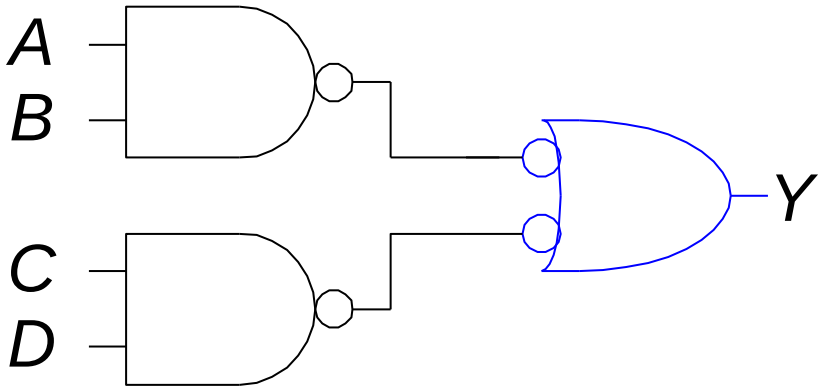
Propagation de bulles

- Quelle est l'expression de ce circuit ?



Propagation de bulles

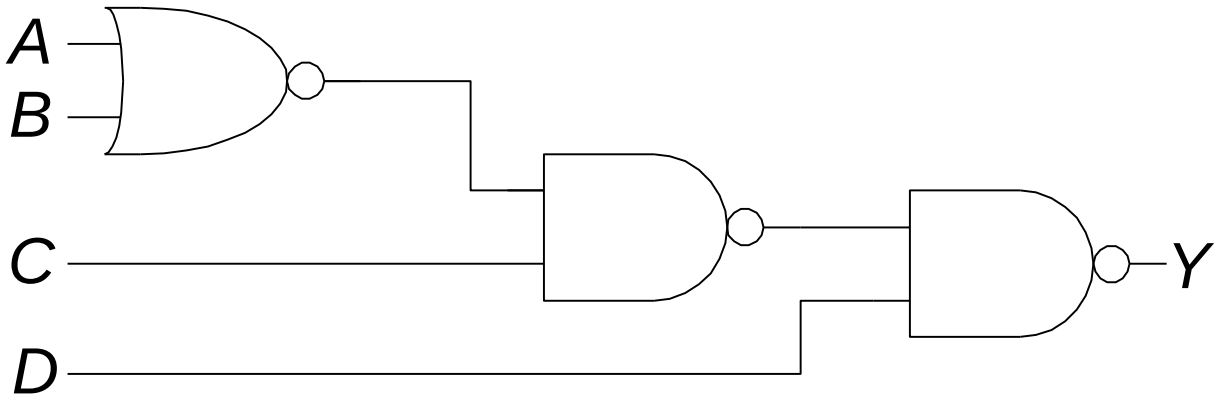
- Quelle est l'expression de ce circuit ?



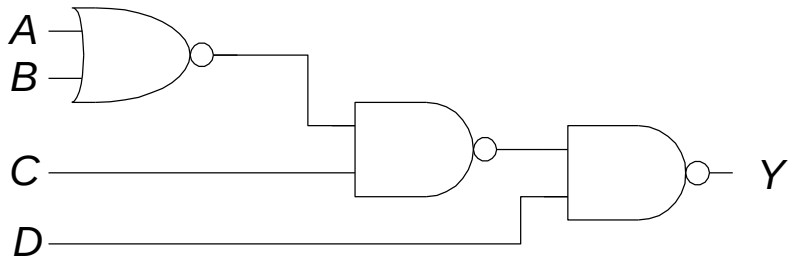
$$Y = AB + CD$$

Propagation de bulles

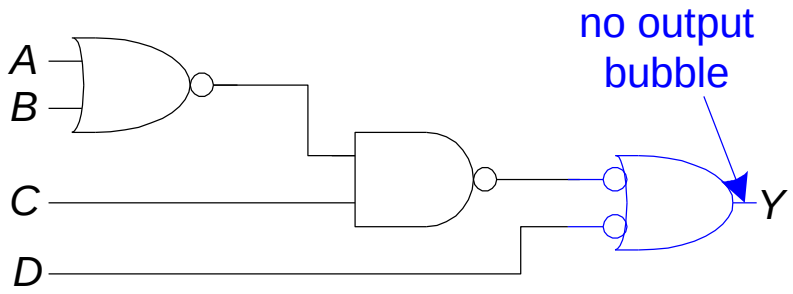
- On commence par le fin et on avance vers les entrées
- Dessiner chaque porte pour que les bulles s'annulent



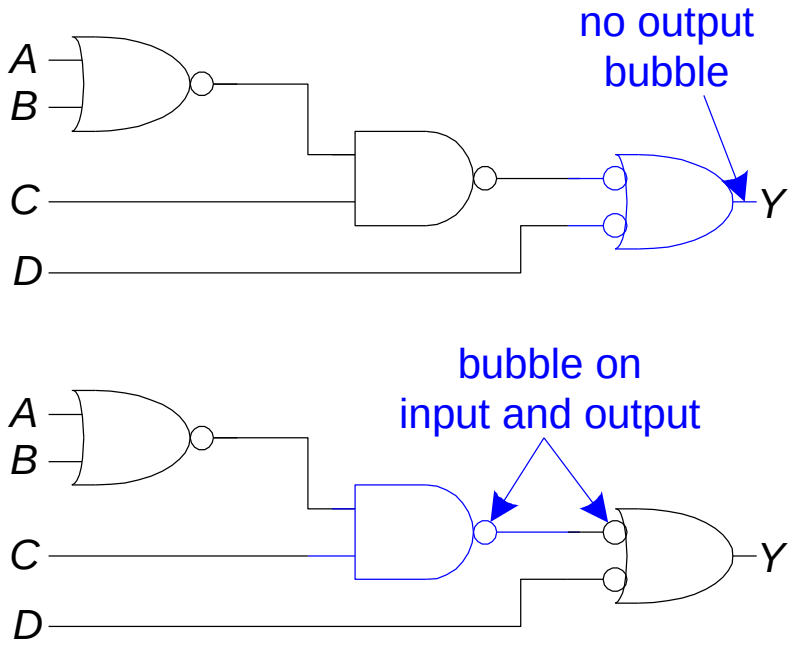
Propagation de bulles



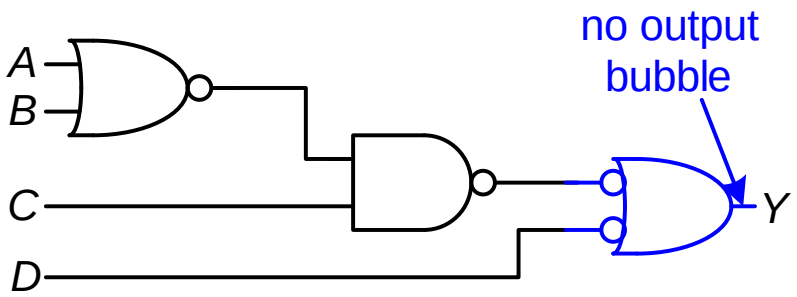
Propagation de bulles

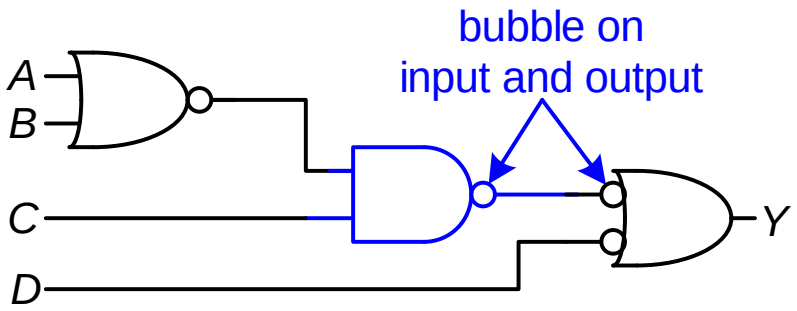


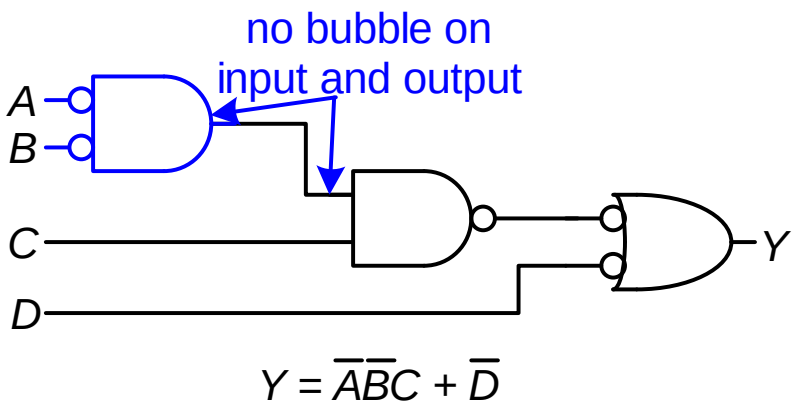
Propagation de bulles



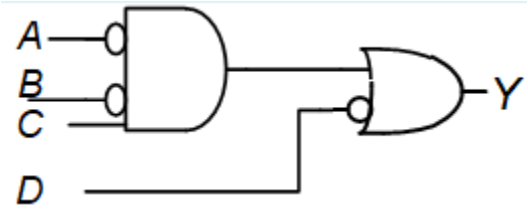
Propagation de bulles





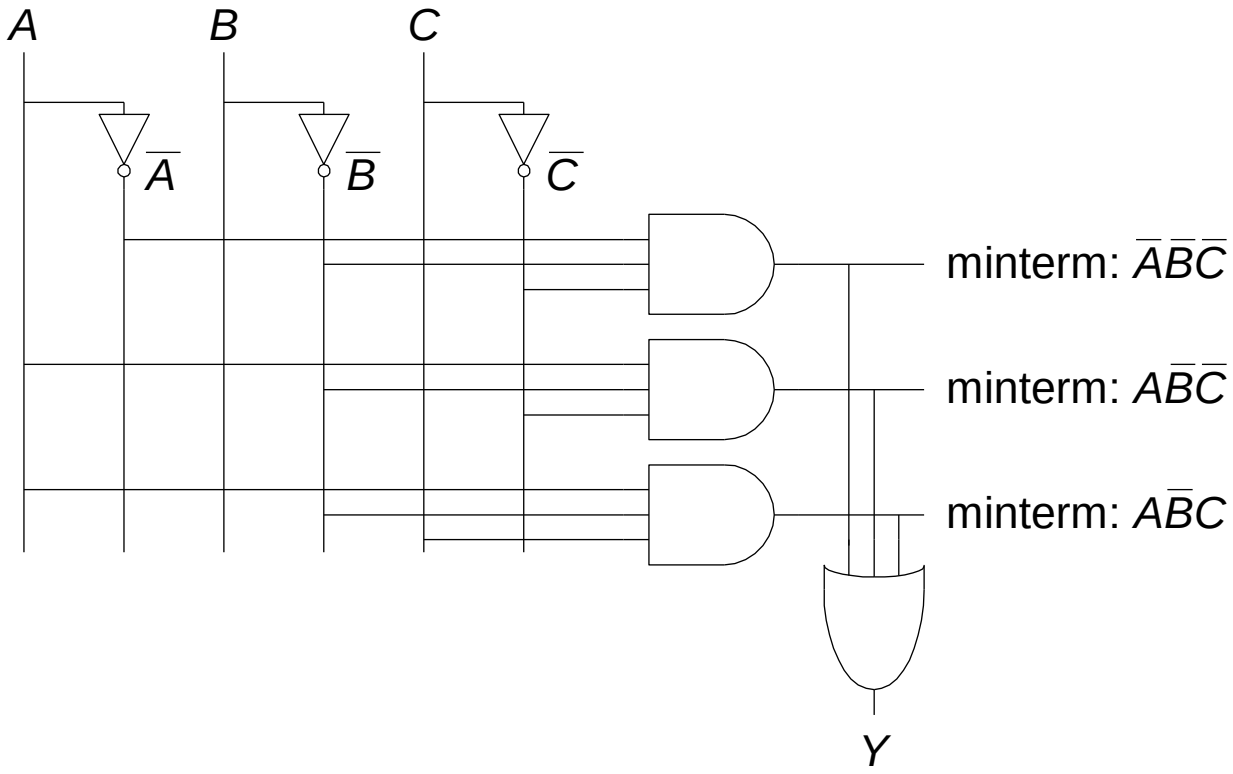


$Y = \overline{A}\overline{B}C + \overline{D}$



Circuit combinatoire

- Logique de deux niveaux: ET suivis de OU
- Exemple: $Y = A\overline{B}\overline{C} + \overline{A}BC + \overline{A}BC$

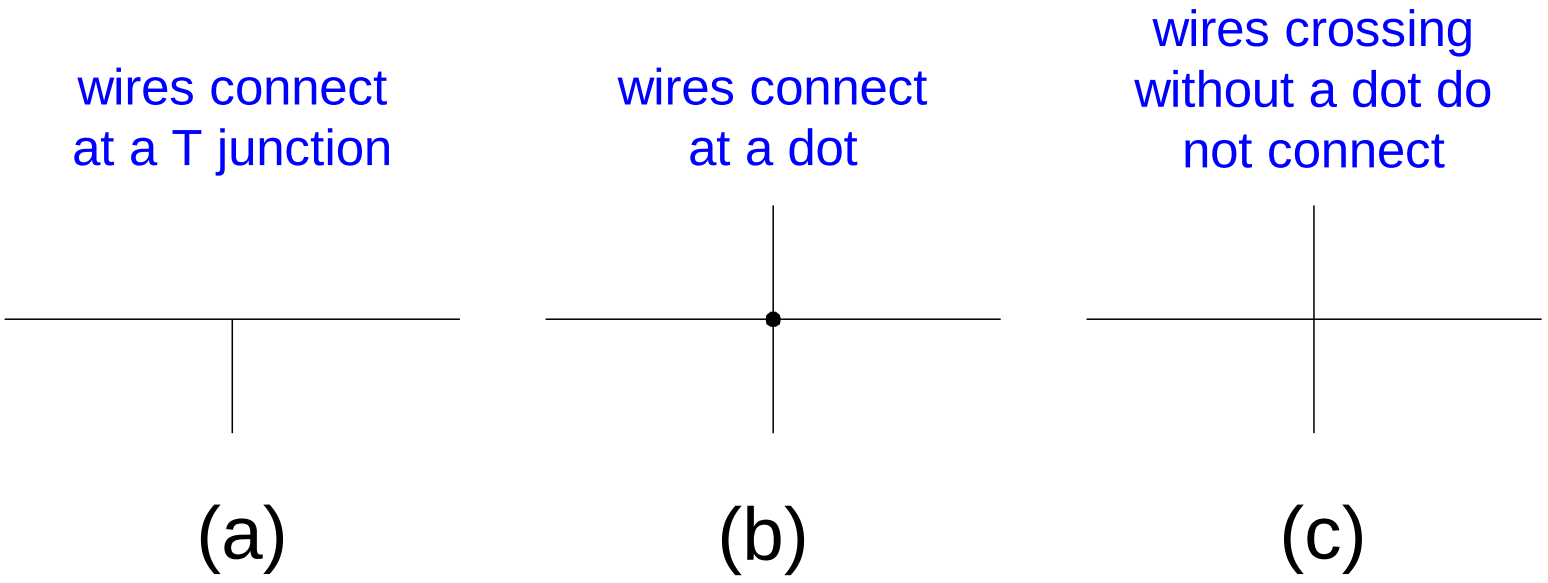


Schématiques avec le style

- Entrées – en haut, à gauche
- Sorties – à droite ou en bas
- Lorsque possible, les portes sont dessinées de gauche à droite
- Fils droits sont préférables

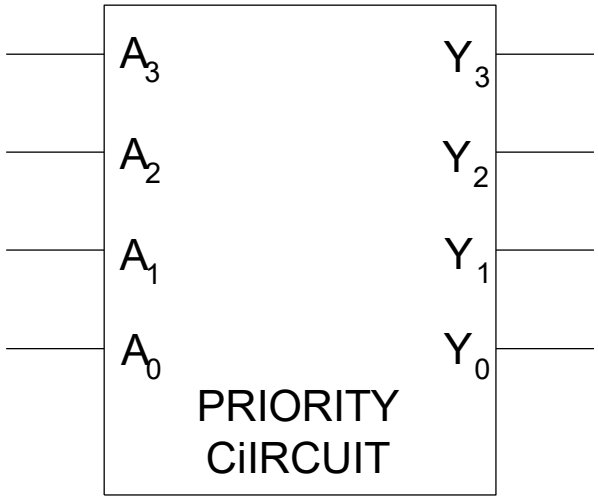
Conventions dans les schémas

- Jonction T – toujours connexion
- Si deux fils se croisent, il n'y a pas de connexion de deux fils sauf si l'intersection est matérialisée par un gros point



Circuits à plusieurs entrées

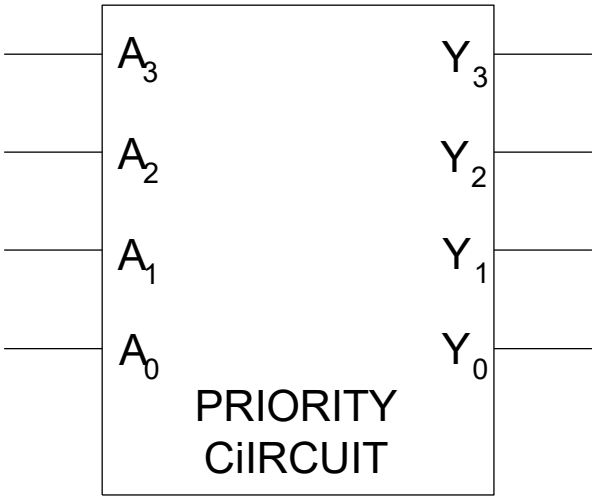
- 1 dans le bit le plus significatif est copié sur la sortie correspondante



A_3	A_2	A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0				
0	0	0	1				
0	0	1	0				
0	0	1	1				
0	1	0	0				
0	1	0	1				
0	1	1	0				
0	1	1	1				
1	0	0	0				
1	0	0	1				
1	0	1	0				
1	0	1	1				
1	1	0	0				
1	1	0	1				
1	1	1	0				
1	1	1	1				

Circuits à plusieurs entrées

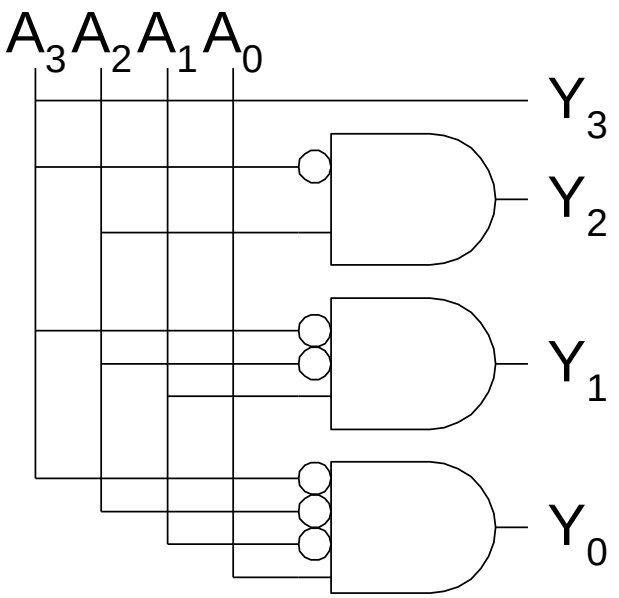
- 1 dans le bit le plus significatif est copié sur la sortie correspondante



A_3	A_2	A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	0
0	0	1	1	0	0	1	0
0	1	0	0	0	1	0	0
0	1	0	1	0	1	0	0
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	0
1	0	0	0	1	0	0	0
1	0	0	1	1	0	0	0
1	0	1	0	1	0	0	0
1	0	1	1	1	0	0	0
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	0
1	1	1	0	1	0	0	0
1	1	1	1	1	0	0	0

Encodeur de priorité

A_3	A_2	A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	0
0	0	1	1	0	0	1	0
0	1	0	0	0	1	0	0
0	1	0	1	0	1	0	0
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	0
1	0	0	0	1	0	0	0
1	0	0	1	1	0	0	0
1	0	1	0	1	0	0	0
1	0	1	1	1	0	0	0
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	0
1	1	1	0	1	0	0	0
1	1	1	1	1	0	0	0



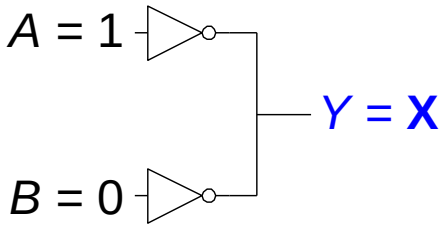
“Don’t Cares” – d ou X

A_3	A_2	A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	0
0	0	1	1	0	0	1	0
0	1	0	0	0	1	0	0
0	1	0	1	0	1	0	0
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	0
1	0	0	0	1	0	0	0
1	0	0	1	1	0	0	0
1	0	1	0	1	0	0	0
1	0	1	1	1	0	0	0
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	0
1	1	1	0	1	0	0	0
1	1	1	1	1	0	0	0

A_3	A_2	A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	d	0	0	1	0
0	1	d	d	0	1	0	0
1	d	d	d	1	0	0	0

Valeur du signal X – Conflit (« contention »)

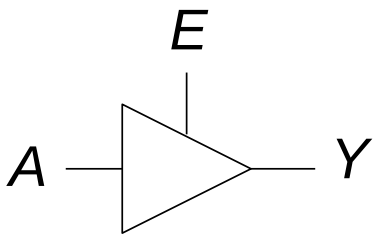
- Conflit: circuit essaye de propager les deux valeurs opposées sur le même fil
  - Valeur actuelle pourrait être une valeur intermédiaire entre 1 et 0
  - Pourrat être 0, 1 ou une valeur de la zone interdite
- Valeur sensible aux tous les changement (tension, température, temps, bruits)
- Souvent cause une dissipation de puissance excessive



- Conflit = erreur
- Attention: symbole X – Contexte!
  - Table de vérité – “don’t care”
  - Signaux – conflit (contention)

Valeur du signal Z – Flottante

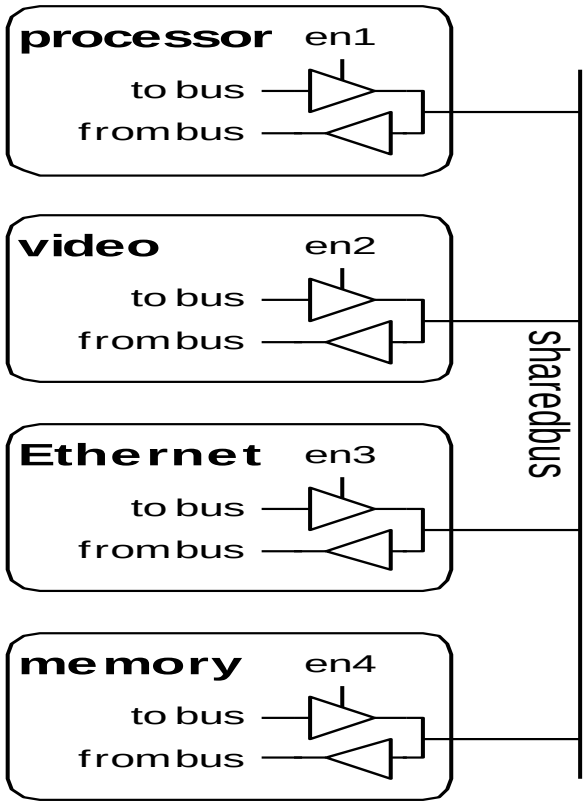
- Valeur flottante, haut impédance
- Tampon à trois états



E	A	Y
0	0	Z
0	1	Z
1	0	0
1	1	1

Bus à trois états

- Nœuds flottants sont utilisés dans les bus à trois états
  - Plusieurs conducteurs différents
  - Exactement un actif à la fois



2. Méthode de Karnaugh: K-Maps

- Méthode graphique (visuelle) pour simplifier au maximum les formules et/ou les circuits
- Chaque case – un minterme
- Les valeurs des entrées ne diffèrent que d'un seul bit entre chaque case.
- $PA + P\overline{A} = P$

A	B	C	Y
0	0	0	**1**
0	0	1	**1**
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

Y \ AB / C	00	01	11	10
0	**1**	0	0	0
1	**1**	0	0	0

Y \ AB / C	00	01	11	10
0	$\bar{A}\bar{B}\bar{C}$	$\bar{A}B\bar{C}$	$AB\bar{C}$	$A\bar{B}\bar{C}$
1	$\bar{A}\bar{B}C$	$\bar{A}BC$	ABC	$A\bar{B}C$

K-map

Entourer tous les 1 dans des cases adjacents:
les plus grands possibles,
tels que leur taille est une puissance de 2,
éventuellement sur les bords

A	B	C	Y
0	0	0	**1**
0	0	1	**1**
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

Expression simplifiée contient les variables qui ne changent pas leurs valeurs entre les cases groupées

- $Y = \bar{A}\bar{B}$

Y: C \ AB	00	01	11	10
0	1	0	0	0
1	1	0	0	0

K-map de 3 entrées

Y: C \ AB	00	01	11	10
0	$\bar{A}\bar{B}\bar{C}$	$\bar{A}B\bar{C}$	$AB\bar{C}$	$A\bar{B}\bar{C}$
1	$\bar{A}\bar{B}C$	$\bar{A}BC$	ABC	$A\bar{B}C$

Truth Table

A	B	C	Y
0	0	0	0
0	0	1	0
0	1	0	**1**
0	1	1	**1**
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	**0**

K-Map

Y: C \ AB	00	01	11	10
0				
1				

K-map de 3 entrées

Y: C \ AB	00	01	11	10
0	$\bar{A}\bar{B}\bar{C}$	$\bar{A}B\bar{C}$	$AB\bar{C}$	$A\bar{B}\bar{C}$
1	$\bar{A}\bar{B}C$	$\bar{A}BC$	ABC	$A\bar{B}C$

Truth Table

A	B	C	Y
0	0	0	0
0	0	1	0
0	1	0	**1**
0	1	1	**1**
1	0	0	0
1	0	1	0
1	1	0	**1**
1	1	1	0

K-Map

Y: C \ AB	00	01	11	10
0	0	1	1	0
1	0	1	0	0

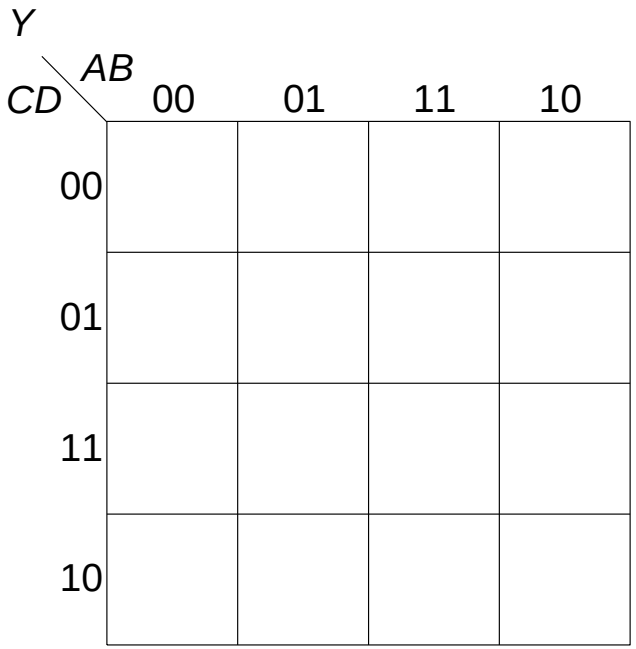
$$Y = \bar{A}B + B\bar{C}$$

Définitions

- **Impliquant premier:** impliquant correspondant au plus grand cercle dans un K-map

K-map de 4 entrées

A	*B*	*C*	*D*	*Y*
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0



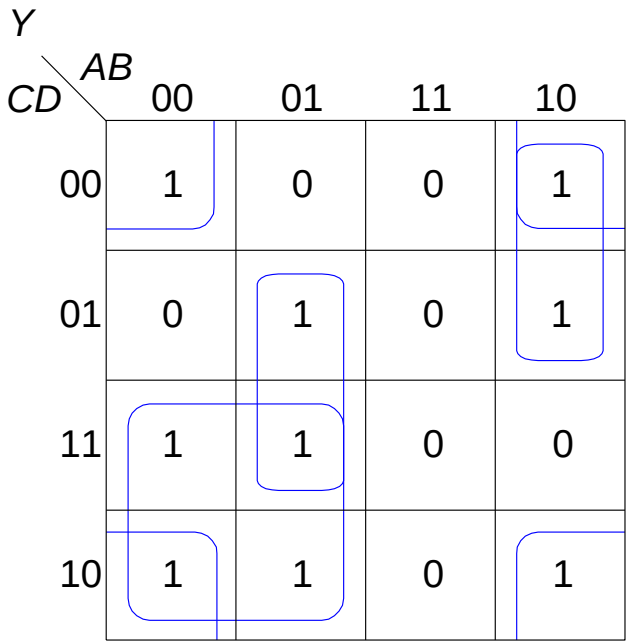
K-map de 4 entrées

A	*B*	*C*	*D*	*Y*
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

Y	*AB* 00	01	11	10
CD 00	1	0	0	1
01	0	1	0	1
11	1	1	0	0
10	1	1	0	1

K-map de 4 entrées

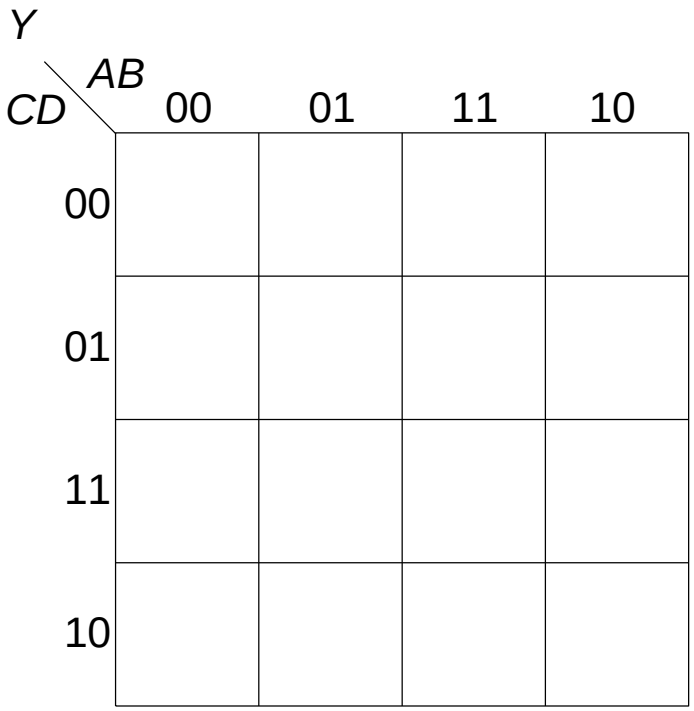
A	B	C	D	Y
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0



$$Y = \bar{A}C + \bar{A}BD + A\bar{B}\bar{C} + \bar{B}\bar{D}$$

K-maps avec Don’t Cares

A	*B*	*C*	*D*	*Y*
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	X
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	X
1	0	1	1	X
1	1	0	0	X
1	1	0	1	X
1	1	1	0	X
1	1	1	1	X



K-maps avec Don’t Cares

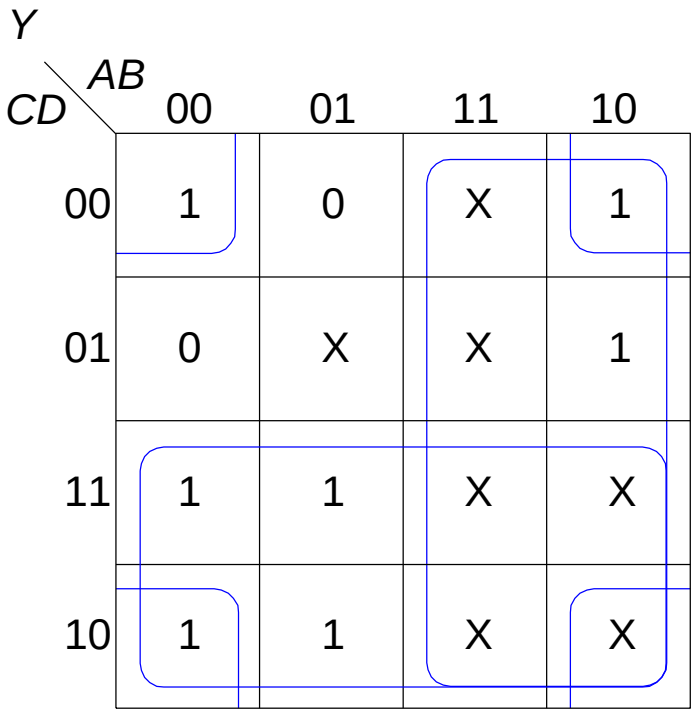
A	*B*	*C*	*D*	*Y*
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	X
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	X
1	0	1	1	X
1	1	0	0	X
1	1	0	1	X
1	1	1	0	X
1	1	1	1	X

Y

CD \ *AB*	00	01	11	10
00	1	0	X	1
01	0	X	X	1
11	1	1	X	X
10	1	1	X	X

K-maps avec Don’t Cares

A	B	C	D	Y
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	X
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	X
1	0	1	1	X
1	1	0	0	X
1	1	0	1	X
1	1	1	0	X
1	1	1	1	X

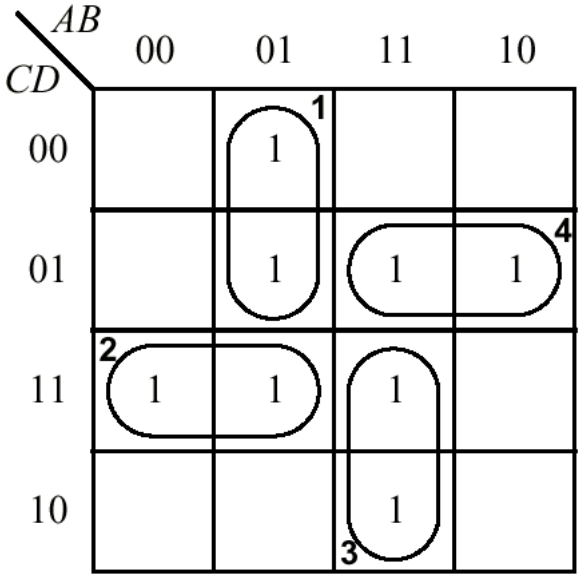


$Y = A + \overline{B}\,\overline{D} + C$

Groupements

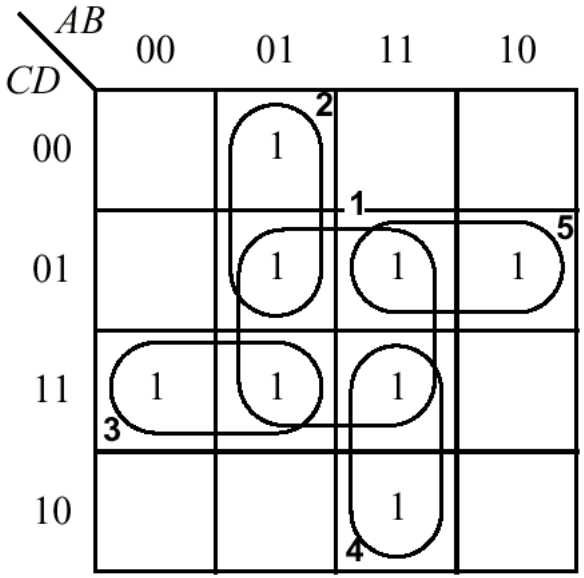
- Groupement minimal

minimal



$$F = \overline{A}\,B\,\overline{C} + \overline{A}\,C\,D + A\,B\,C + A\,\overline{C}\,D$$

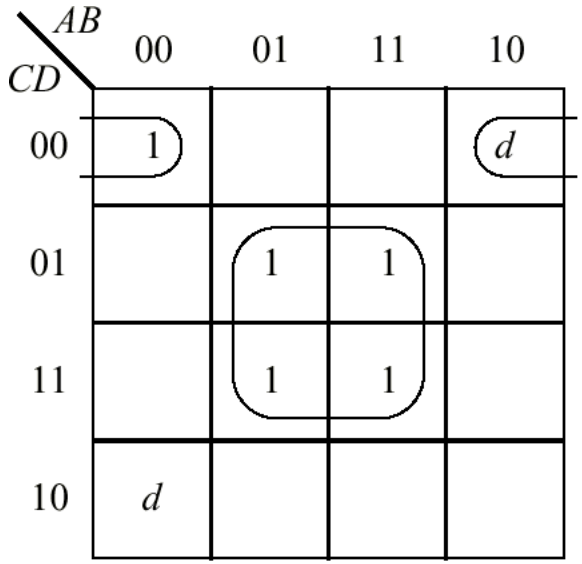
Groupement non



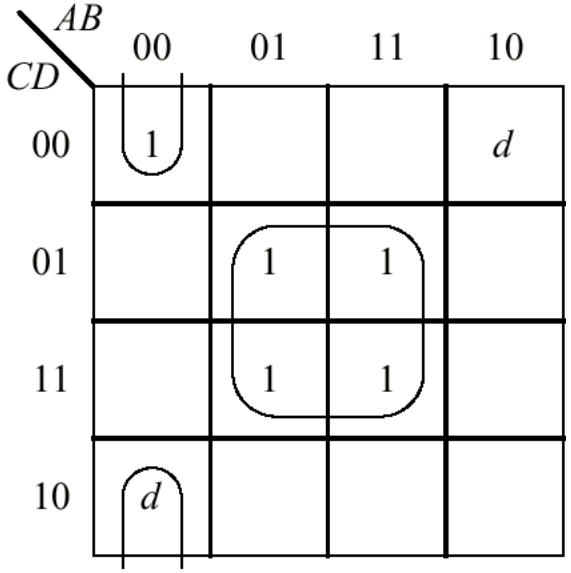
$$F = B\,D + \overline{A}\,B\,\overline{C} + \overline{A}\,C\,D + A\,B\,C + A\,\overline{C}\,D$$

K-Maps et Don’t Cares

- Plusieurs groupements minimaux.



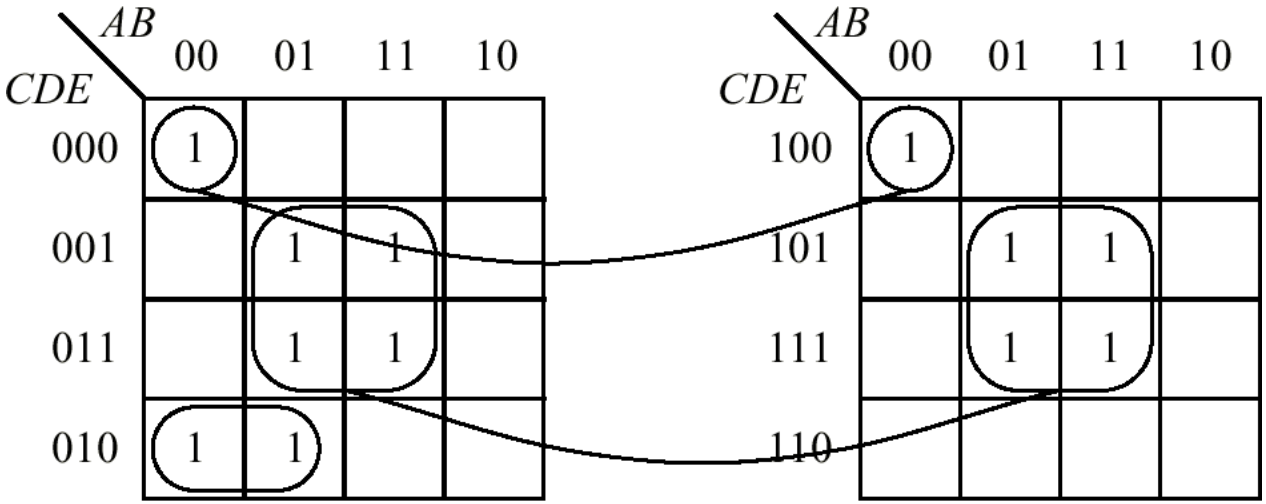
$F = \overline{B}\,\overline{C}\,\overline{D} + B\,D$



$F = \overline{A}\,\overline{B}\,\overline{D} + B\,D$

K-map de 5 entrées

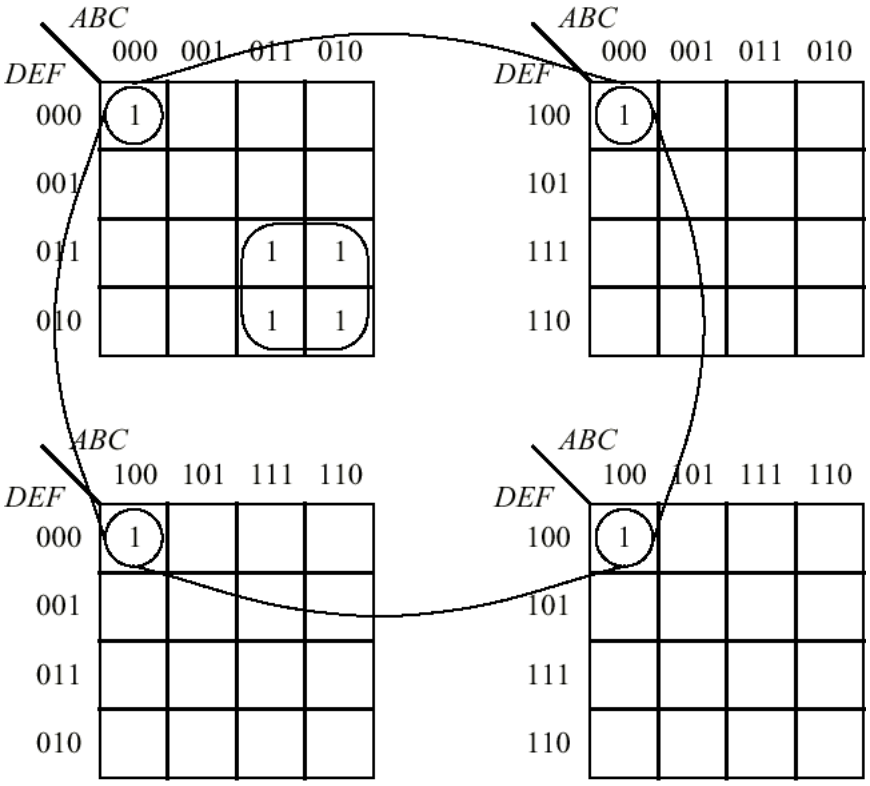
- • Visualiser deux K-maps de 4 variables positionner l'une au-dessus de l'autre
- Groupements sont effectués dans un cube tridimensionnel



$$F = \overline{A}\,\overline{C}\,D\overline{E} + \overline{A}\,\overline{B}\,\overline{D}\,\overline{E} + B\,E$$

K-map de 6 entrées

- • Visualiser 4 K-maps de 4 variables positionner l'une au-dessus de l'autre. Groupements sont effectués dans un cube tridimensionnel



$$G = \overline{B}\,\overline{C}\,\overline{E}\,\overline{F} + \overline{A}\,B\,\overline{D}\,E$$

Réduction Tabulaire (Quine-McCluskey)

- Table de vérité avec d(X) – une fonction non complètement définie, “dont care” dans la partie Sorties
- Notation équivalente d’une table de vérité:
- $F(A, B, C, D) = \sum(1, 3, 5, 6, 7, 10, 13) + \sum_d(0, 11, 15)$

	A	B	C	D	F
0	0	0	0	0	d
1	0	0	0	1	1
2	0	0	1	0	0
3	0	0	1	1	1
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	1
7	0	1	1	1	1
8	1	0	0	0	0
9	1	0	0	1	0
10	1	0	1	0	1
11	1	0	1	1	d
12	1	1	0	0	0
13	1	1	0	1	1
14	1	1	1	0	0
15	1	1	1	1	d

Réduction Tabulaire (Quine-McCluskey)

	A	B	C	D	F
0	0	0	0	0	d
1	0	0	0	1	1
2	0	0	1	0	0
3	0	0	1	1	1
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	1
7	0	1	1	1	1
8	1	0	0	0	0
9	1	0	0	1	0
10	1	0	1	0	1
11	1	0	1	1	d
12	1	1	0	0	0
13	1	1	0	1	1
14	1	1	1	0	0
15	1	1	1	1	d

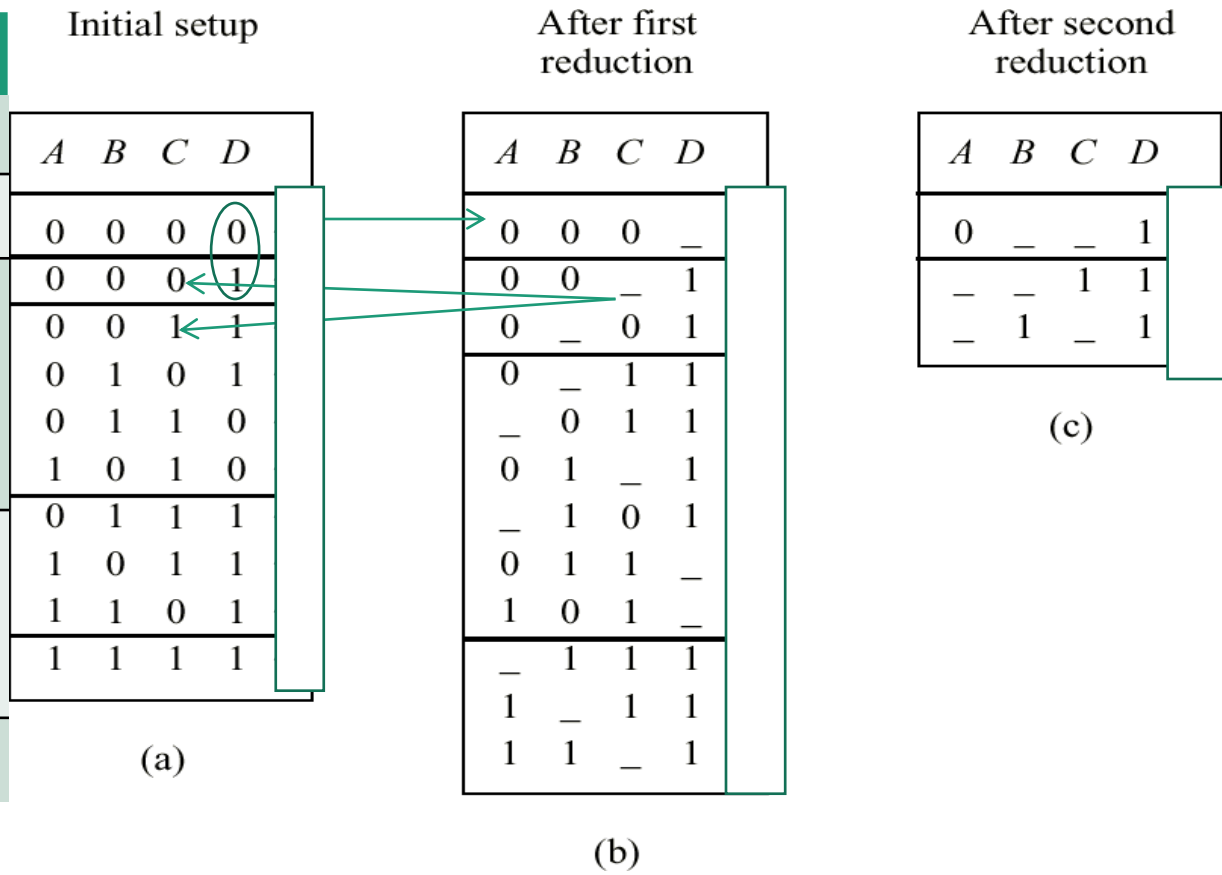
- Groupement des mintermes pour lesquels F ≠ 0 selon le nombre de 1 dans chaque minterme.
- Don't cares – sont considérés comme non zéros

A	B	C	D
0	0	0	0
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	1	0
0	1	1	1
1	0	1	1
1	1	0	1
1	1	1	1

Réduction Tabulaire (Quine-McCluskey)

- Former un consensus (produit logique) entre chaque paire des groupes adjacents pour tous les termes qui se diffèrent d'un seul bit

A	B	C	D
0	0	0	0
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	1	0
0	1	1	1
1	0	1	1
1	1	0	1
1	1	1	1



Réduction Tabulaire (Quine-McCluskey)

- Impliquants premiers forment un ensemble qui couvre complètement une fonction, mais pas obligatoirement minimal
- Une table de choix est utilisée pour obtenir un ensemble de couverture minimal

A	B	C	D	
0	0	0	_	*
0	0	_	1	√
0	_	0	1	√
0	_	1	1	√
_	0	1	1	√
0	1	_	1	√
_	1	0	1	√
0	1	1	_	*
1	0	1	_	*
_	1	1	1	√
1	_	1	1	√
1	1	_	1	√

A	B	C	D	
0	_	_	1	*
_	_	1	1	*
_	1	_	1	*

Prime Implicants	Minterms						
	0001	0011	0101	0110	0111	1010	1101
0 0 0 _	√						
0 1 1 _				√	√		
1 0 1 _						√	
0 _ _ 1	√	√	√		√		
_ _ 1 1		√			√		
_ 1 _ 1			√		√		√

Réduction Tabulaire (Quine-McCluskey)

- Impliquants premiers essentiels, 011_, 101_ et _1_1, sont retirés de la table de choix en formant une table réduite de choix. La procédure est répétée.
- Résultat:

0 1 1 _ + 1 0 1_ +_ 1 _1 + 0_ _1

$$F(A,B,C,D) = \bar{A}\,B\,C + A\,\bar{B}\,C + B\,D + \bar{A}\,D$$

Eligible Set	Minterms	
	0001	0011
0 0 0 _	√	
0 _ _ 1	√	√
_ _ 1 1		√

Set 1	Set 2
0 0 0 _	0 _ _ 1
_ _ 1 1	

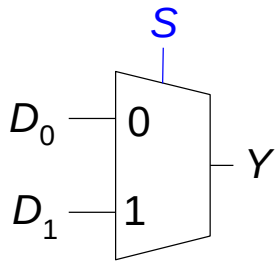
Blocs combinatoires

- Multiplexeurs
- Décodeurs

Multiplexeur (Mux)

2:1 Mux

- Entrées:
  - 2^n *lignes d'entrée (données):* $D_0, D_1, \ldots, D_{2^n}$
  - *N lignes de sélection: a; b; c; ...*
  - *Sortie: une seule sortie Y*
- Rôle: Aiguiller la valeur de l'une des 2^n lignes d'entrée vers la sortie Y
- La ligne d'entrée choisie est désignée grâce aux bits de sélection

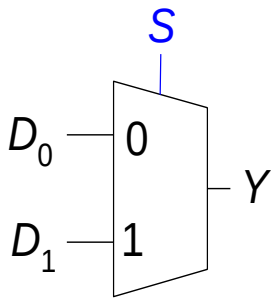


S	D_1	D_0	Y
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Multiplexeur (Mux)

- Entrées:
  - 2^n *lignes d'entrée (données):* $D_0, D_1, \ldots, D_{2^n}$
  - *N lignes de sélection: a; b; c; ...*
  - *Sortie: une seule sortie Y*
- Rôle: Aiguiller la valeur de l'une des 2^n lignes d'entrée vers la sortie Y
- La ligne d'entrée choisie est désignée grâce aux bits de sélection

2:1 Mux

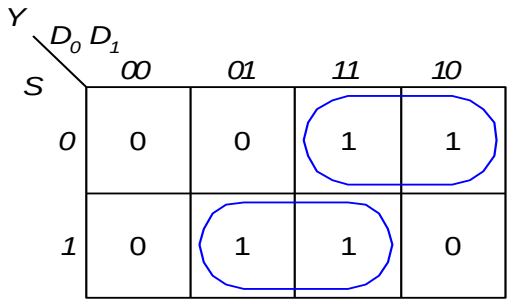


S	D_1	D_0	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

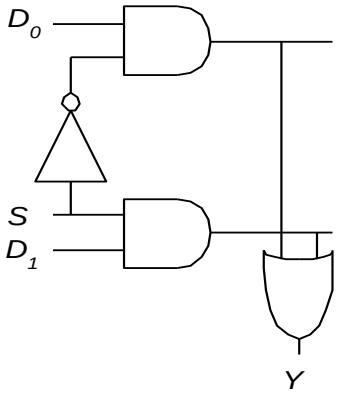
S	Y
0	D_0
1	D_1

Implémentations équivalentes d'un Multiplexeur

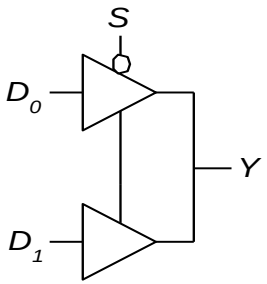
- Implémentation 1:
  - Portes logiques (SOP)



$$Y = D_0\overline{S} + D_1S$$



- Implémentation 2:
  - Tampons à trois états



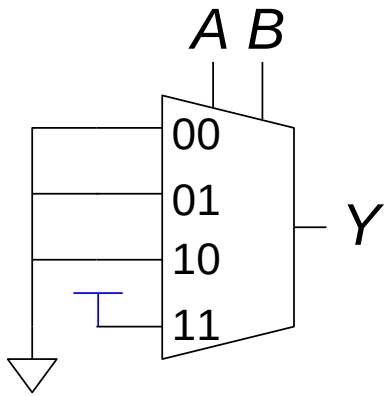
– Pour un MUX de N entrées
  - N tampons
– Un seul tampon est dans l'état de propagation

Logique en utilisant les MUXs

- Une **table de correspondance** (*Look-Up Table* (*LUT*)) est un terme informatique et électronique désignant une liste d'association de valeurs
- Utilisation de MUX comme LUT

A	*B*	*Y*
0	0	0
0	1	0
1	0	0
1	1	1

$Y = AB$



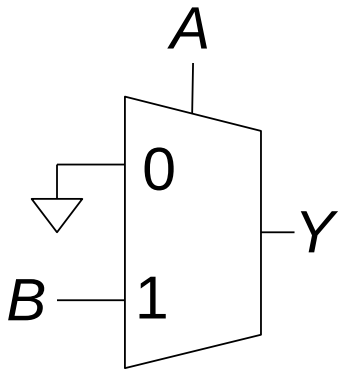
Logique en utilisant les MUXs

- Réduisons la taille de MUX

$Y = AB$

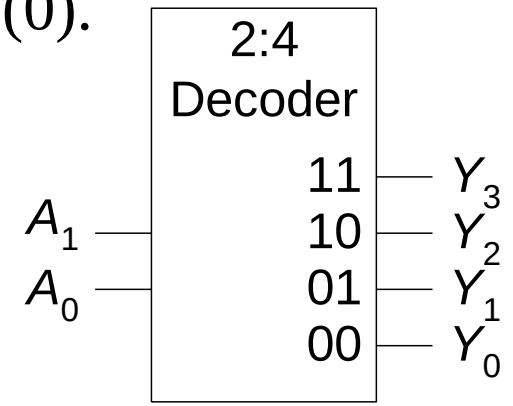
A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

A	Y
0	0
1	B



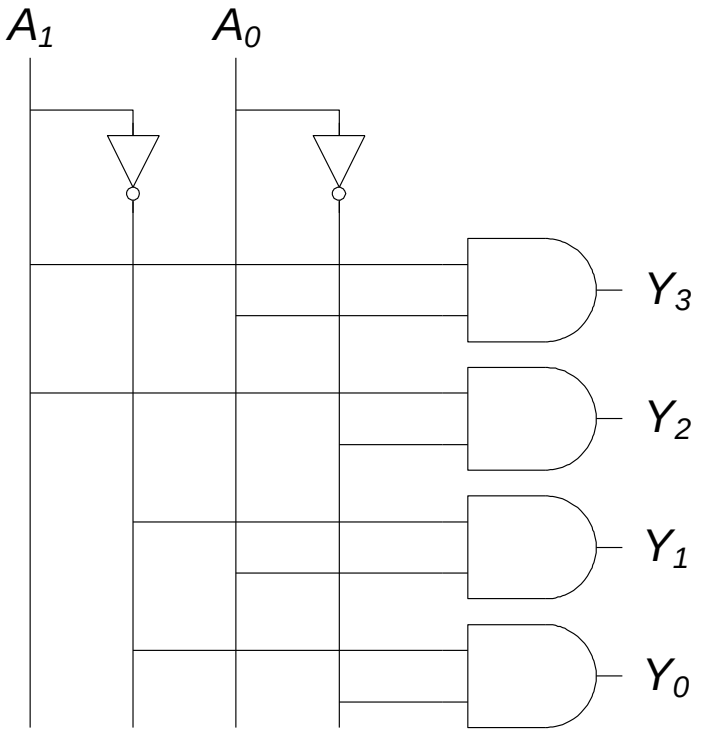
Décodeurs

- N entrées, 2^N sorties
- Sorties “One-hot” : seulement une sortie est active (1) et les autres – non (0).



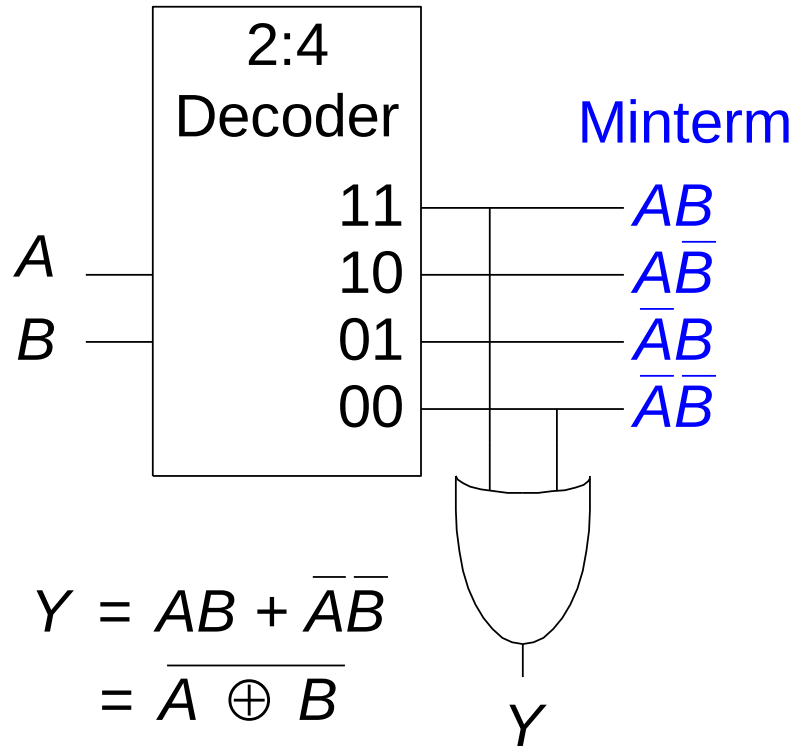
A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

Implémentation d'un décodeur



Logique en utilisant des décodeurs

- OU de minterms



$$Y = AB + \bar{A}\bar{B}$$
$$= \overline{A \oplus B}$$

$$Y = \overline{A \oplus B} = \overline{\bar{A}B + A\bar{B}} = \overline{\bar{A}B}\ \overline{A\bar{B}} =$$
$$\left(\bar{\bar{A}} + \bar{B}\right)\left(\bar{A} + \bar{\bar{B}}\right) = (A + \bar{B})(\bar{A} + B) =$$
$$A\bar{A} + \bar{B}\bar{A} + AB + B\bar{B} = AB + \bar{A}\bar{B}$$

Caractéristiques électriques et temporelles

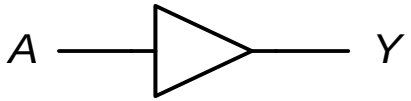
L'ordre d'apparition des variables - important
Changement de valeurs (0 → 1 ou 1 → 0)

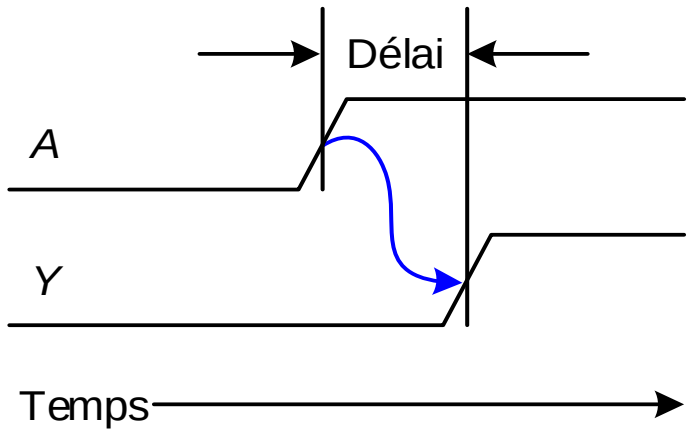
- Idéalement: instantané
- En pratique: prend un certain temps, délai de montée 0 → 1 et de descente 1 → 0

Caractéristiques électriques et temporelles

Passage d'un signal au travers d'une porte

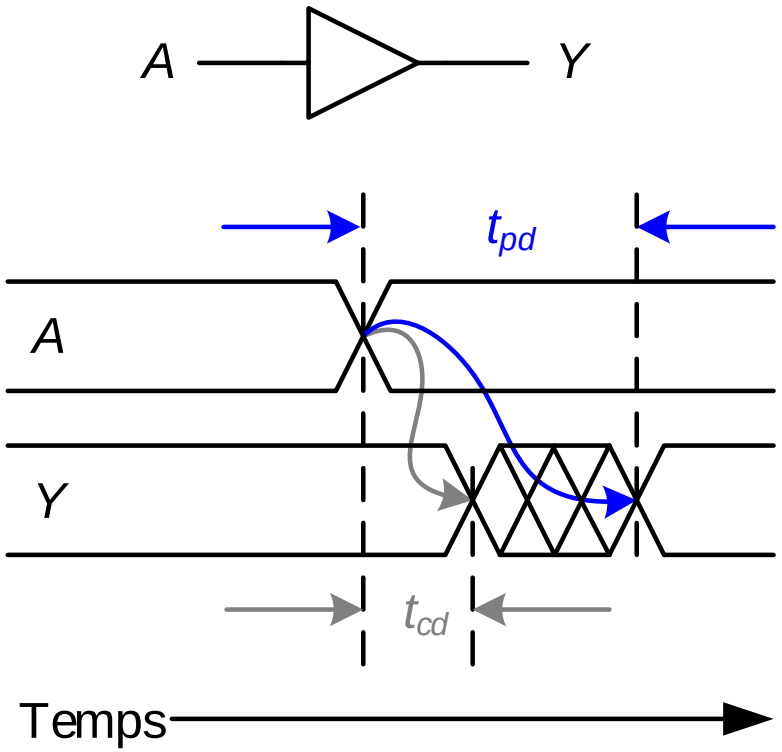
- Idéalement: instantané
- En pratique: prend un certain temps, délai de propagation





Délai de Propagation & Contamination

- Délai de propagation : t_{pd} = délai max du premier changement de l'entrée et jusqu'à la stabilisation complète du signal de sortie
- Délai de contamination : t_{cd} = délai min du premier changement de l'entrée et jusqu'au premier changement du signal de sortie

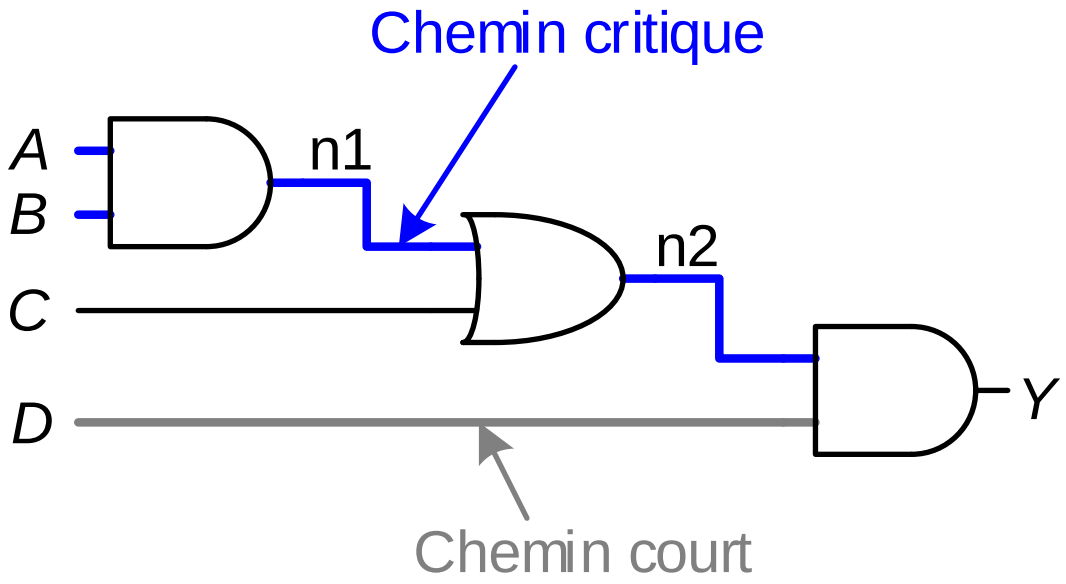


Contraintes temporelles des circuits

Un circuit est formé de plusieurs portes

- Chemin critique : chemin le « plus long » pour la propagation des signaux à travers le circuit
- Détermine le temps total de propagation des signaux à travers tout le circuit
- Temps minimal à attendre pour avoir une sortie valide
  - Intuitivement : chemin passant par le plus grand nombre de portes
  - Mais dépend aussi du temps de propagation de chaque type de porte

Chemin critique (longue) et chemin court



Chemin Critique (Longue): $t_{pd} = 2t_{pd\_\mathrm{AND}} + t_{pd\_\mathrm{OR}}$

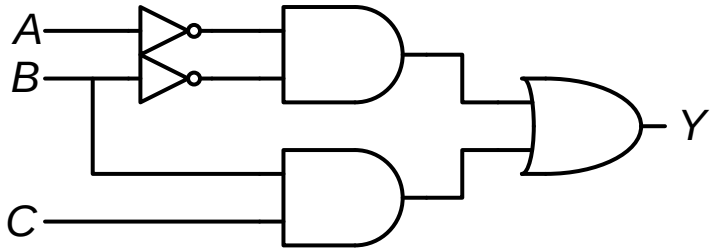
Chemin court $t_{cd} = t_{cd\_\mathrm{AND}}$

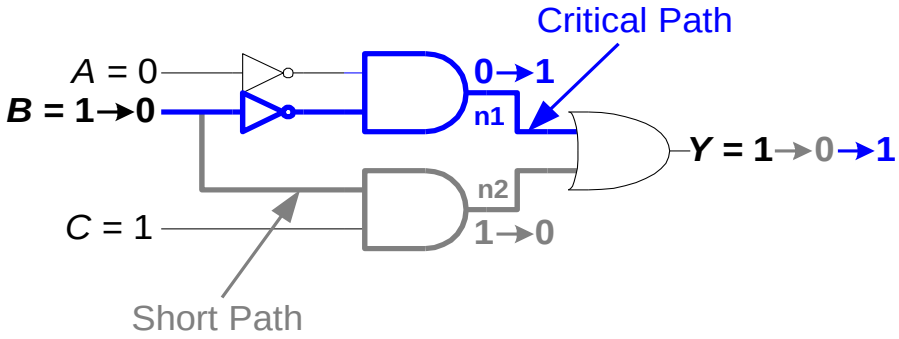
Aléas (Glitch)

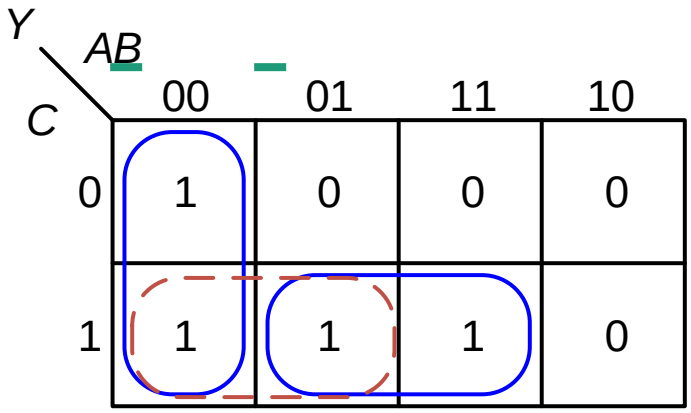
- Glitch (Aléas): Un nœud présente des aléas si, suite à une variation particulière du vecteur d'entrée des transitions non prévues par la fonction booléenne apparaissent
- Les aléas sont dus à l'existence de chemins de natures différentes et de longueurs différentes actifs simultanément

Glitch, Exemple

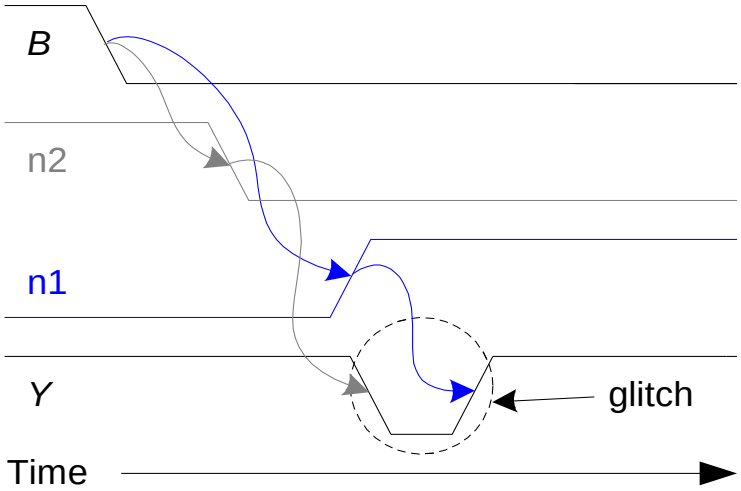
- A = 0, C = 1, B passe de 1 à 0?







$Y = \overline{A}\overline{B} + BC$



Horloge

- À cause de tous les délais (montée, descente, propagation) un signal n'est pas dans un état valide en permanence
- Idée : on ne lit ses valeurs qu'à des instants précis et à des intervalles réguliers
  - Instants donnés par une horloge
- Horloge
  - Système logique qui émet régulièrement une suite d'impulsions calibrées
  - L'intervalle de temps entre 2 impulsions représente le temps de cycle ou la période de l'horloge

Horloge

- Signal périodique
  - un demi période à 0, l'autre à 1
- Début d'une nouvelle période : instant t_i
- Exemple
  - Instant *t1* : E = 1, S = 0
  - Instant *t2* : E = 0, S = 1
  - CLK = Clock = signal d'horloge