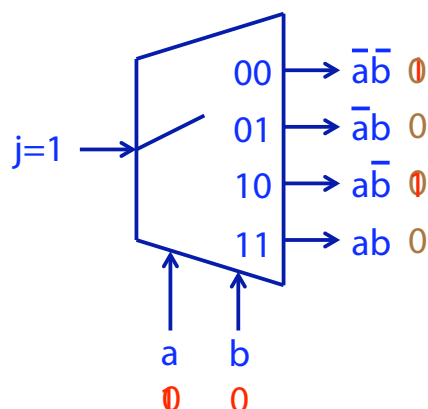
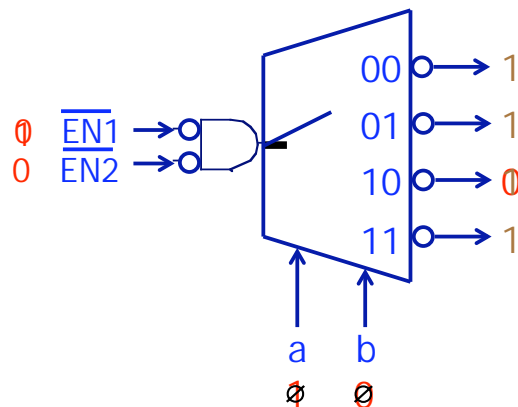


Le démultiplexeur

- ◆ Un démultiplexeur est un système combinatoire réalisant tous les 2^n mintermes des n variables d'entrée
- ◆ Une seule sortie, c'est-à-dire un seul minterme, est égale à 1 à un moment donné



- ◆ L'entrée j peut être utilisée comme un signal de "permission" (*enable*) de la fonction du démultiplexeur

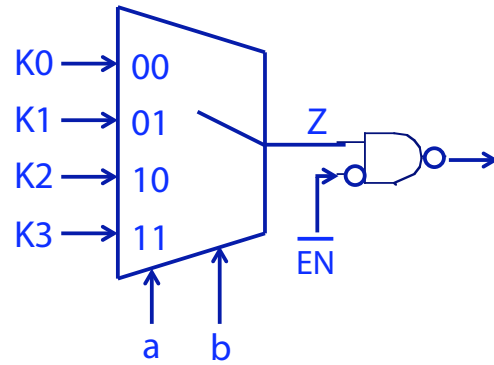
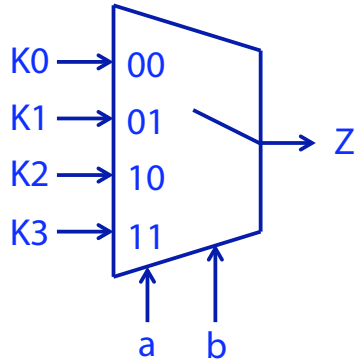


◆ Applications:

- réalisation d'une fonction combinatoire quelconque: une porte OU fait la somme logique des mintermes qui composent la forme canonique
- activation d'un élément parmi 2^n : chaque signal de sortie est connecté à un élément à activer

Le multiplexeur

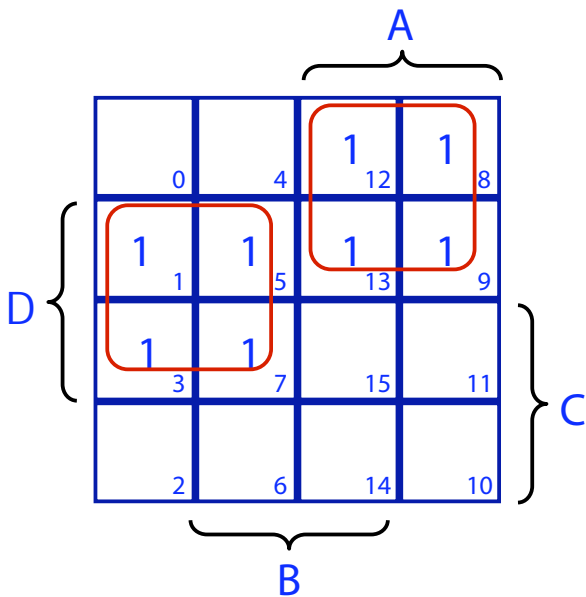
- ◆ Un multiplexeur est un système combinatoire réalisant la fonction universelle
- ◆ Applications:
 - réalisation d'une fonction combinatoire
 - choix d'un signal parmi 2^n
- ◆ Un démultiplexeur permet la réalisation de plusieurs fonctions en même temps, par le simple ajout d'une porte OU par fonction. Un multiplexeur permet la réalisation d'une seule fonction à un moment donné: la réalisation d'une autre fonction implique le changement des valeurs des paramètres K_i



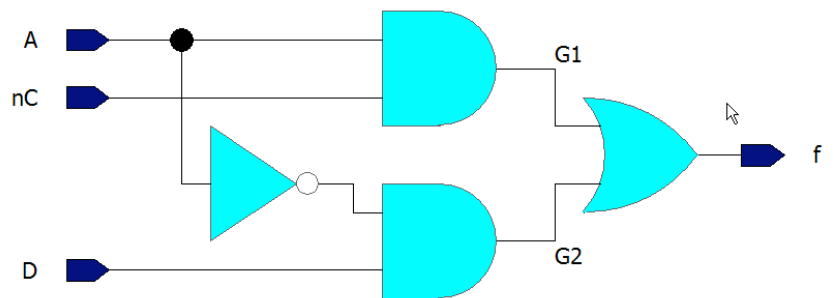
Hasards

◆ Exemple:

$$f(A,B,C,D) = \sum 1,3,5,7,8,9,12,13$$

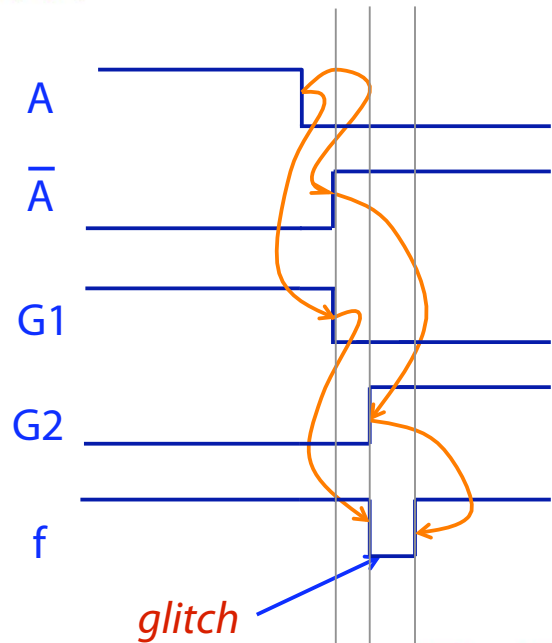
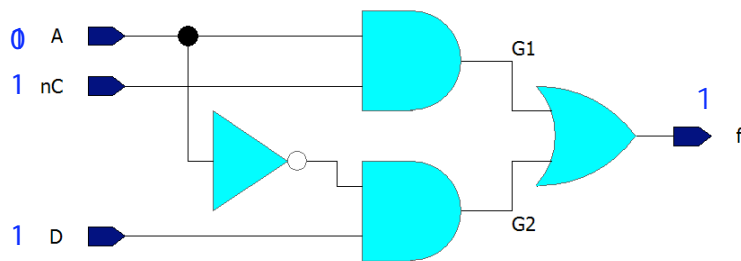


$$f = A\bar{C} + \bar{A}D$$



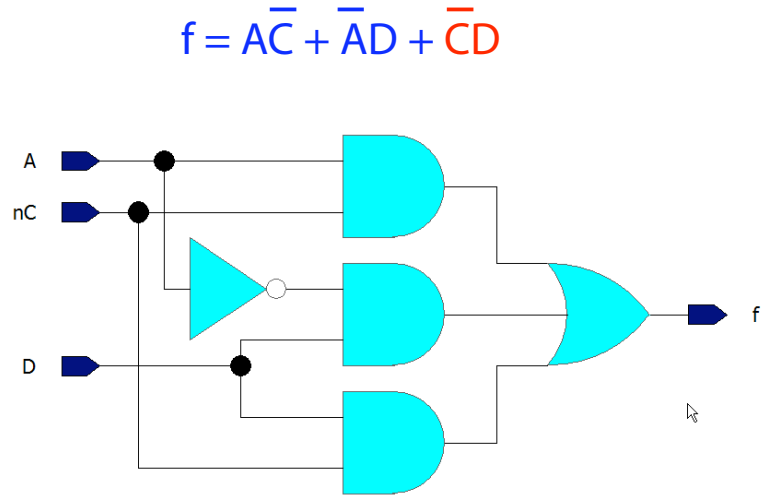
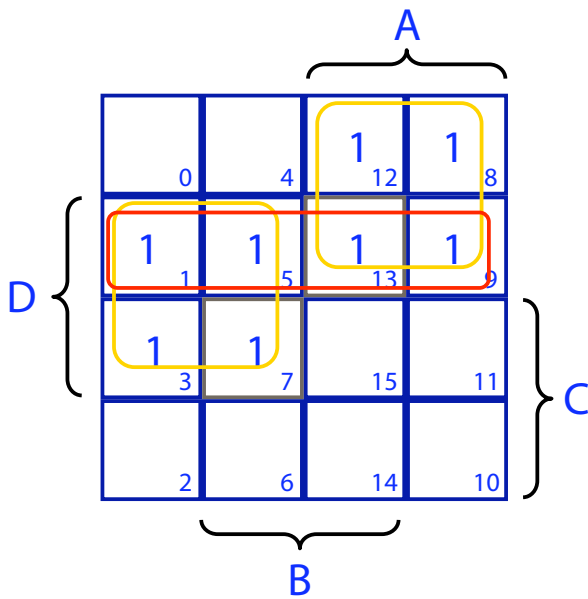
Supposons un changement des entrées:

ABCD → ABCD
1 1 0 1 0 1 0 1



- ◆ La possibilité d'un *glitch à zéro* existe si:
 - 1) deux combinaisons d'entrées se suivent avec le changement d'une seule entrée
 - 2) les deux combinaisons d'entrées produisent une sortie égale à 1
- ◆ La possibilité d'un *glitch à un* existe si:
 - 1) deux combinaisons d'entrées se suivent avec le changement d'une seule entrée
 - 2) les deux combinaisons d'entrées produisent une sortie égale à 0
- ◆ Le *glitch* apparaît si la porte qui couvre une combinaison d'entrées passe à 0 avant que celle qui couvre l'autre combinaison passe à 1. Le *glitch* n'apparaît pas si les deux combinaisons sont couvertes par la même porte.
Bien entendu, une réalisation avec tous les implicants premiers est libre de hasards...

- ◆ Pour éliminer les *glitches*, il faut couvrir avec un même implicant premier les minterms en cause.
Pour notre exemple:



A lire dans Wakerly

- ◆ Chapitre 4
 - 4.5: hasards
- ◆ Chapitre 5
 - 5.7: multiplexeurs et démultiplexeurs