I22 — Architecture des ordinateurs et système d'exploitation

J. Razik

Dpt. Informatique — UTLN

2021–2022

Logique Combinatoire

Fonction booléenne à plusieurs variables

- $f : \{0,1\}^n \rightarrow \{0,1\}$
- $y = f(x_1, x_2, \ldots, x_n), \quad y \in \{0,1\}$
- Table de vérité

x	$f(x)$
0	$f(0)$
1	$f(1)$

x_1	x_0	$f(x_0, x_1)$
0	0	$f(0,0)$
0	1	$f(1,0)$
1	0	$f(0,1)$
1	1	$f(1,1)$

Fonction booléenne à plusieurs variables

- Fonctions à 1 variable : exactement 4 fonctions possibles

x	$f(x)$	
0	0	f_1, fonction *Faux* ou *False*
1	0	
0	0	f_2, fonction *Identité* ou *Id*
1	1	
0	1	f_3, fonction *Non* ou *Not*
1	0	
0	1	f_4, fonction *Vrai* ou *True*
1	1	

Fonction booléenne à plusieurs variables

- Fonctions à 2 variables booléennes : $2^4 = 16$ fonctions différentes
  - « ET » (*AND*),
  - « OU » (*OR*),
  - « OU EXCLUSIF » (*XOR*),
  - « NON-ET » (*NAND*),
  - « NON-OU » (*NOR*),
  - etc ...
- Fonctions à n variables booléennes
  - On peut se ramener aux fonctions de bases à 1 ou 2 variables.

Algèbre de Boole

- $B : (\{0,1\}, OU, ET, NON)$, ou noté $(\{0,1\}, +, ., \overline{\ })$
- Propriétés ensemblistes
  - Lois de composition internes : ET et OU sont internes, c'est-à-dire à valeur dans $\{0,1\}$;
  - La loi ET est commutative et associative ;
  - La loi OU est commutative et associative ;
  - Distributivité du ET sur le OU et du OU sur le ET ;
  - Idempotence : $A\ ET\ A = A$, $A\ OU\ A = A$;
  - Élément neutre : 1 pour ET, 0 pour OU ;
  - Élément absorbant : 0 pour ET, 1 pour OU ;
  - $\forall A,\ A + \overline{A} = 1$;
  - $\forall A,\ A.\overline{A} = 0$.
- Lois de De Morgan :
  - $\forall (A,B),\ \overline{A+B} = \overline{A}.\overline{B}$;
  - $\forall (A,B),\ \overline{A.B} = \overline{A} + \overline{B}$;
  - $\overline{A_1 + \cdots + A_n} = \overline{A_1}.\cdots.\overline{A_n}$;
  - $\overline{A_1.\cdots.A_n} = \overline{A_1} + \cdots + \overline{A_n}$;

Décomposition en *OU*

- Soit $f(x, y, z)$ telle que $f(0, 1, 0) = 1$, $f(1, 0, 0) = 1$ et $f = 0$ ailleurs
  - Soit $f_{010}(x, y, z)$ telle que $f_{010}(x, y, z) = 1$ pour $(x, y, z) = (0, 1, 0)$ et $f_{010}(x, y, z) = 0$ pour toutes les autres combinaisons ;
  - Soit $f_{100}(x, y, z)$ telle que $f_{100}(x, y, z) = 1$ pour $(x, y, z) = (1, 0, 0)$ et $f_{100}(x, y, z) = 0$ pour toutes les autres combinaisons.
  - Soit $g(x, y, z)$ définie par $g(x, y, z) = f_{010}(x, y, z) + f_{100}(x, y, z)$
  - Alors $f = g$
- Soit $f(x_1, x_2, \ldots, x_N)$ une fonction booléenne à N variables booléennes x_i
  - $\{\underline{a}_i\}$, q multiplets de N bits
  - $f(\underline{x}) = 1 \Leftrightarrow \underline{x} \in \{\underline{a}_1, \ldots, \underline{a}_q\}$
  - On peut créer q fonctions $f_i(\underline{x})$ telles que $f_i(\underline{x}) = 1$ si et seulement si $\underline{x} = \underline{a_i}$
  - $f(\underline{x}) = f_1(\underline{x}) + \cdots + f_q(\underline{x})$

Monôme booléen

Produit booléen par l'opérateur ET de N variables distinctes, complémentées ou non

- Exemple : $A.B.C$
  - $A.B.C = 1$ si et seulement si A, B et C valent 1, 0 sinon
- Exemple : $A.\overline{B}.C$
  - $A.\overline{B}.C = 1$ si et seulement si A et C valent 1 et que B vaut 0

Si $f(x_1, \ldots, x_N) = 1$ pour $(a_1, \ldots, a_N)$, $a_i \in \{0, 1\}$, alors

- $f(\underline{x}) = u_1. \cdots . u_N$, tel que

$$\forall i \in [1, N], \begin{cases} u_i & = x_i \ si \ a_i = 1 \\ u_i & = \overline{x}_i \ si \ a_i = 0 \end{cases}$$

Forme normale disjonctive (*OU*)

- Forme normale disjonctive
  - disjonction de conjonctions
  - définition des monômes booléens où f vaut 1
  - disjonction entre ces monômes
- Exemple
  - $f(\underline{x}) = 1$ si et seulement si $\underline{x} = (0,1,0)$ ou $\underline{x} = (1,0,0)$

$$\begin{cases} f_{010}(x_1,x_2,x_3) = \overline{x}_1.x_2.\overline{x}_3 \\ f_{100}(x_1,x_2,x_3) = x_1.\overline{x}_2.\overline{x}_3 \end{cases}$$

  - $f = f_{010} + f_{100}$

$$f(x_1,x_2,x_3) = f_{010}(x_1,x_2,x_3) + f_{100}(x_1,x_2,x_3) = \overline{x}_1.x_2.\overline{x}_3 + x_1.\overline{x}_2.\overline{x}_3$$

- Forme normale conjonctive (*ET*)
  - Autre expression du polynôme booléen : conjonction de disjonctions

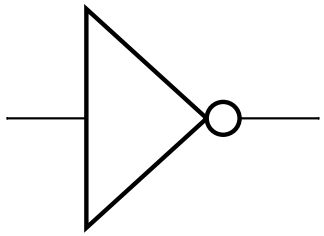
Polynôme en *NON-ET* (*NAND*) et en *NON-OU* (*NOR*)

- 3 opérateurs logiques générateurs
  - *ET*, *OU*, *NON*
- 1 opérateur pour les générer tous : *NON-ET* (NAND)
  - $NAND(x, y) = \overline{x.y}$
  - $NAND(x, x) = \overline{x}.$
  - $NAND(NAND(x, y), NAND(x, y)) = \overline{\overline{x.y}} = x.y$
  - $NAND(NAND(x, x), NAND(y, y)) = \overline{\overline{x}.\overline{y}} = \overline{\overline{x}} + \overline{\overline{y}} = x + y$
- Même conclusion avec l'opérateur *NON-OU* (NOR)

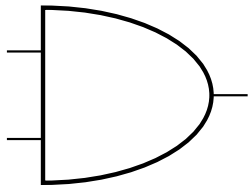
$\Rightarrow$ On peut exprimer un polynôme booléen avec uniquement des opérateurs *NON-ET* (respectivement *NON-OU*)

Réseau logique

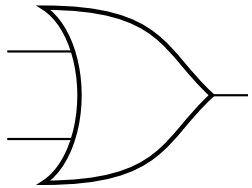
Portes logiques



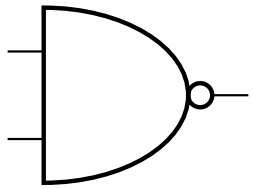
(a) Porte *NON/NOT*



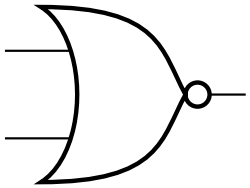
(b) Porte *ET/AND*



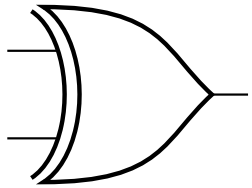
(c) Porte *OU/OR*



(d) Porte *NON-ET/NAND*

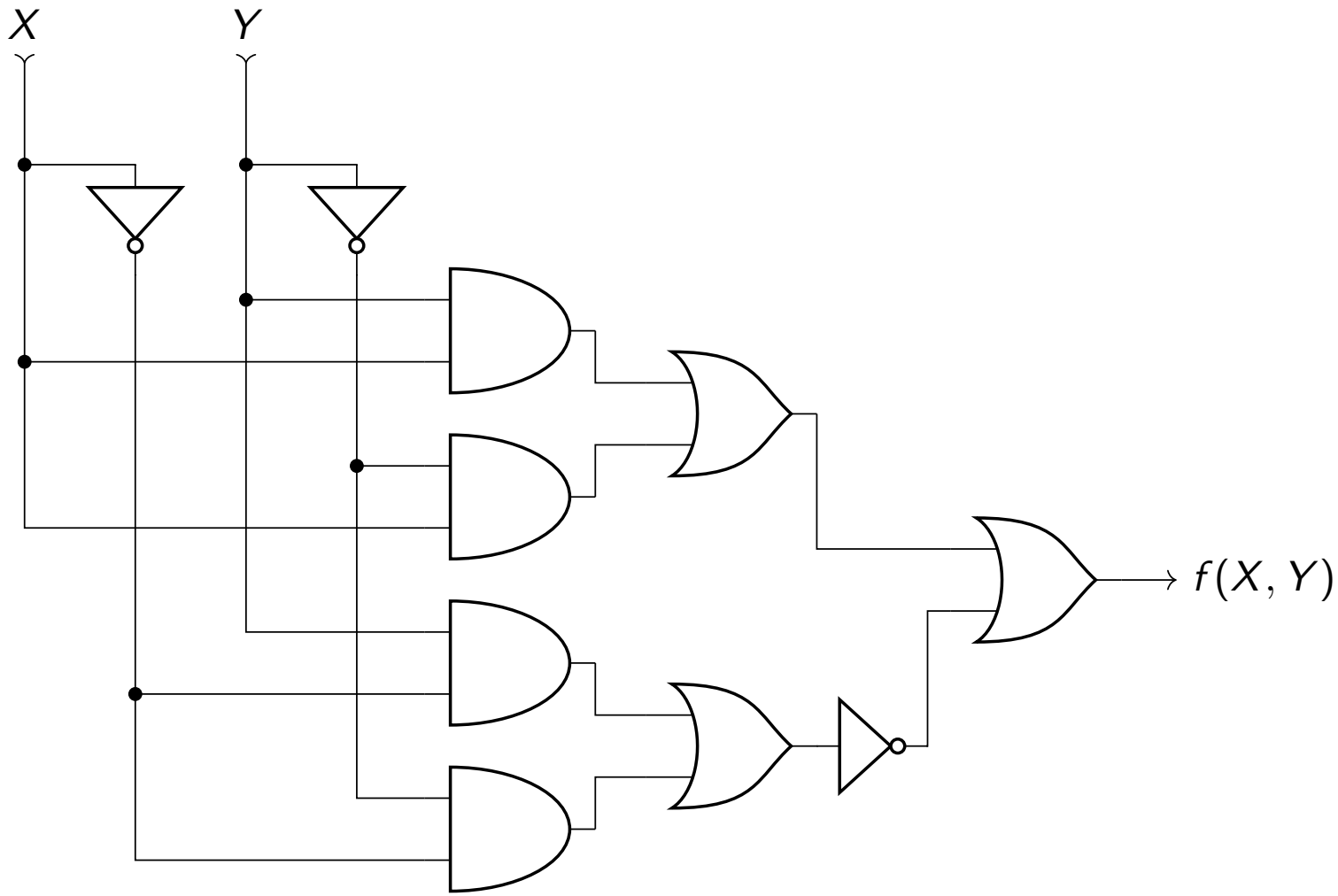


(e) Porte *NON-OU/NOR*

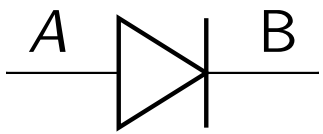


(f) Porte *XOR*

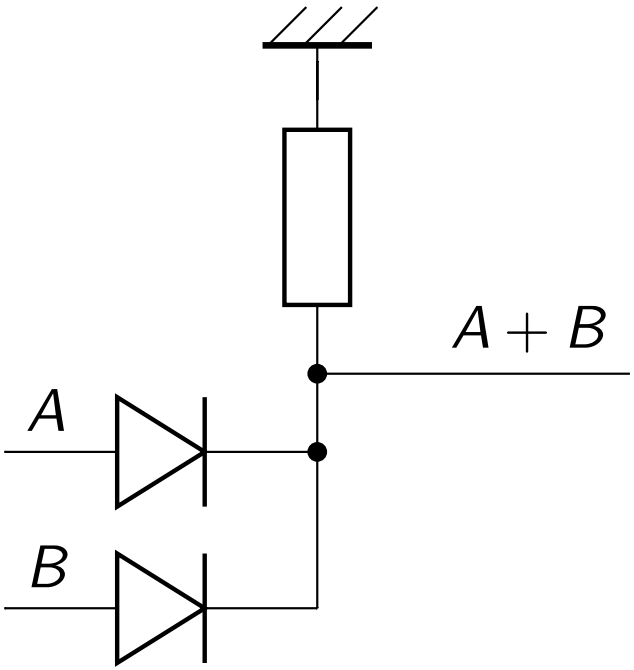
Réseau logique : exemple



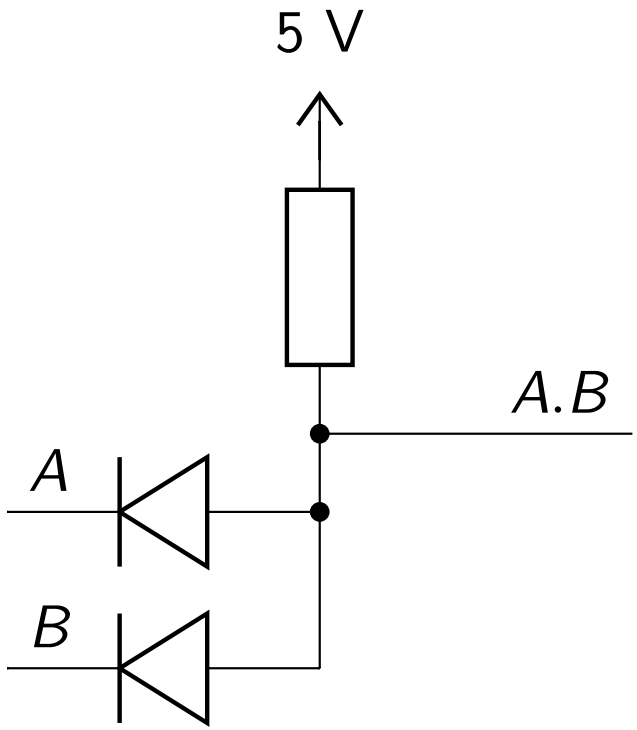
Réalisation physique des opérateurs logiques



Une diode

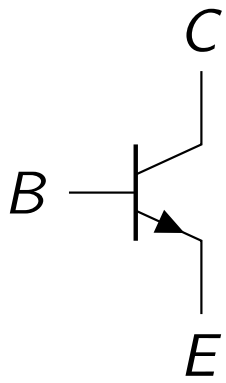


(g) Réalisation d'une porte *OU*

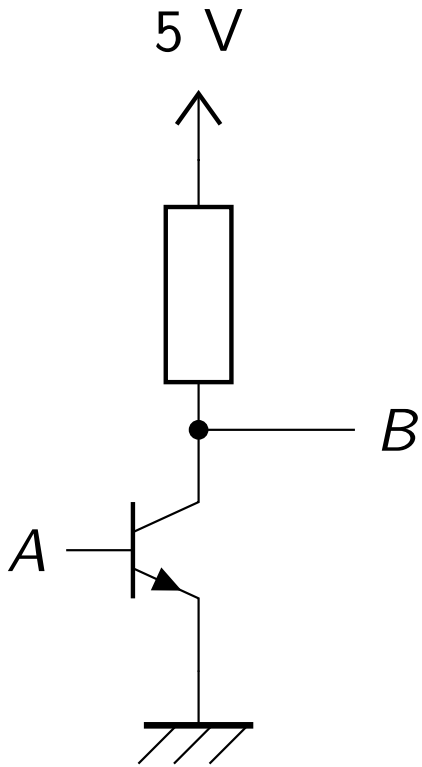


(h) Réalisation d'une porte *ET*

Réalisation physique des opérateurs logiques



(i) Un transistor *NPN*



(j) Réalisation d'une porte *NON* avec un transistor *NPN*

Réduction d'un polynôme booléen

- Objectifs
  - Simplifier l'équation
  - Diminuer le nombre de monômes et leur complexité
  - Diminuer la complexité et le coût de réalisation
- 2 méthodes
  - Algébrique
    - ★ Utilisation des propriétés ensemblistes
  - Graphique
    - ★ Utilisation des tables de Karnaugh

Réduction d'un polynôme booléen

- Comment ?
  - Fusion de monômes avec un facteur et son inverse
    - ★ Exemple : $\overline{A}.B.\overline{C}$ et $A.B.\overline{C}$
    - ★ $(A + \overline{A}).B.\overline{C} = 1.B.\overline{C} = B.\overline{C}$
- Détection de ces monômes candidats
  - Réordonner la table de vérité
  - Code binaire réfléchi ou *code de Gray*
  - 1 seul 1 bit change entre deux valeurs successives
  - Exemple sur 3 bits : 000, 001, 011, 010, 110, 111, 101, 100.

Code binaire réfléchi

Exemple : $f(A, B, C) = A + B.C$

A	B	C	A+B.C
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

(k) Ordre binaire habituel

A	B	C	A+B.C
0	0	0	0
0	0	1	0
0	1	1	1
0	1	0	0
1	*1*	0	1
1	1	1	1
1	0	**1**	1
1	*0*	**0**	1

(l) Ordre binaire réfléchi

Table de Karnaugh

Représentation réordonnée de la table de vérité sur 2 dimensions

A	B	C	A+B.C
0	0	0	0
0	0	1	0
0	1	1	1
0	1	0	0
1	*1*	0	1
1	1	1	1
1	0	**1**	1
1	*0*	**0**	1

A \ BC	00	01	11	10
0	0	0	1	0
1	1	1	1	1

Table de Karnaugh

Exemple de tables de Karnaugh à 2, 3 et 4 variables.

A \ B	0	1
0	0	1
1	0	1

$$f(A,B) = B$$

A \ BC	00	01	11	10
0	0	0	1	0
1	1	1	1	1

$$f(A,B,C) = A + B.C$$

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

$$f(A,B,C,D) = CD + AB\overline{D}$$

Table de Karnaugh

Exemple d'une table de Karnaugh.

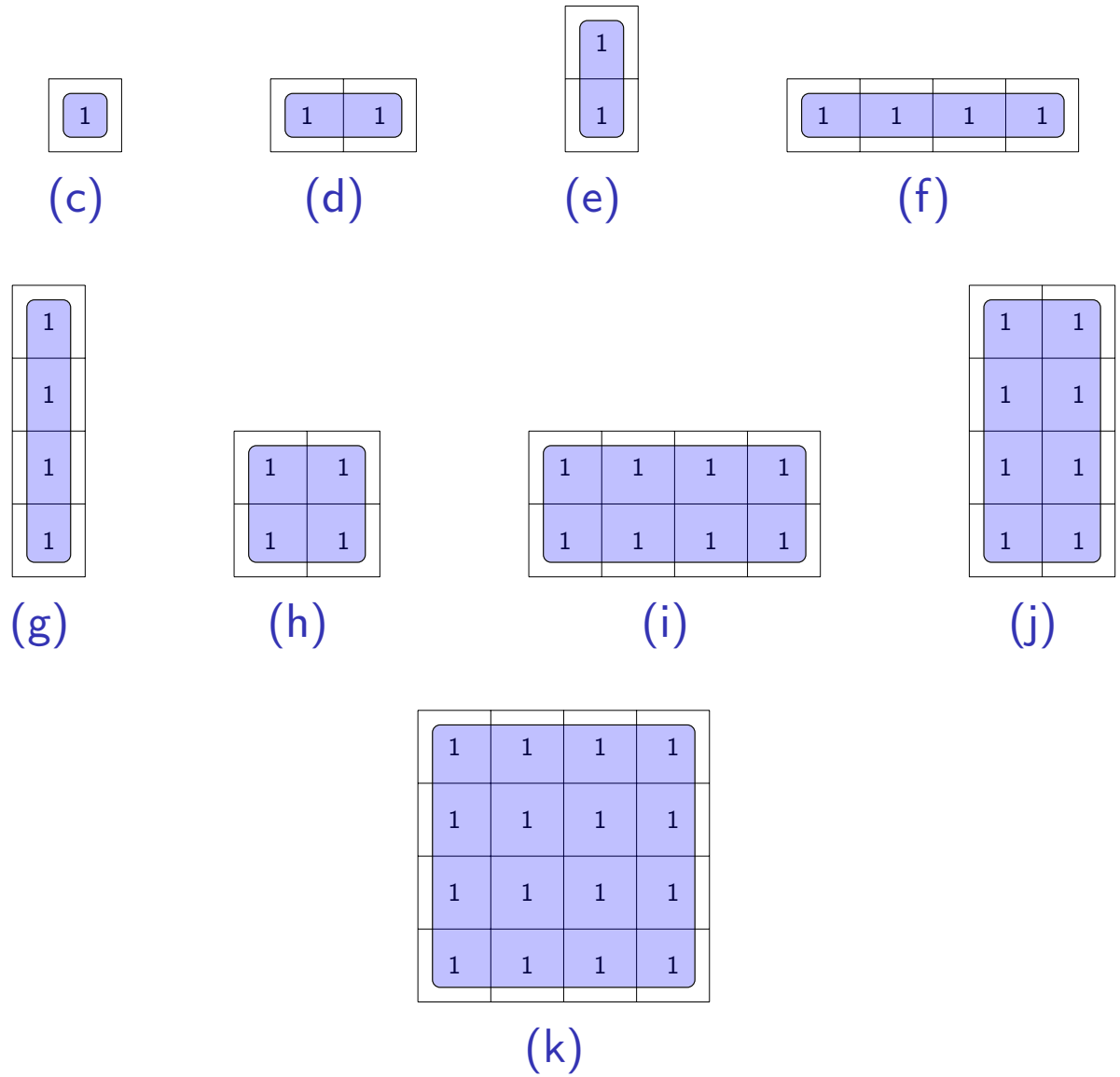
A \ BC	00	01	11	10
0	1	1	0	0
1	1	1	0	0

(a)

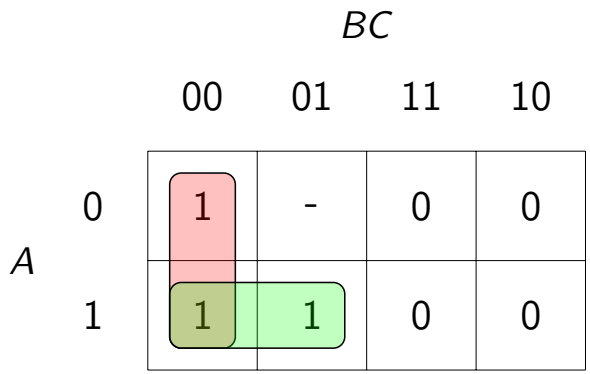
A \ BC	00	01	11	10
0	1	1	0	0
1	1	1	0	0

Table de Karnaugh

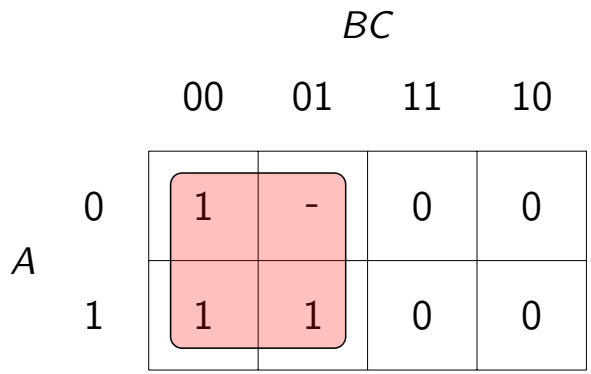
Les différents regroupements valides pour une table de Karnaugh d'au plus 4 variables



Les cas indéterminés



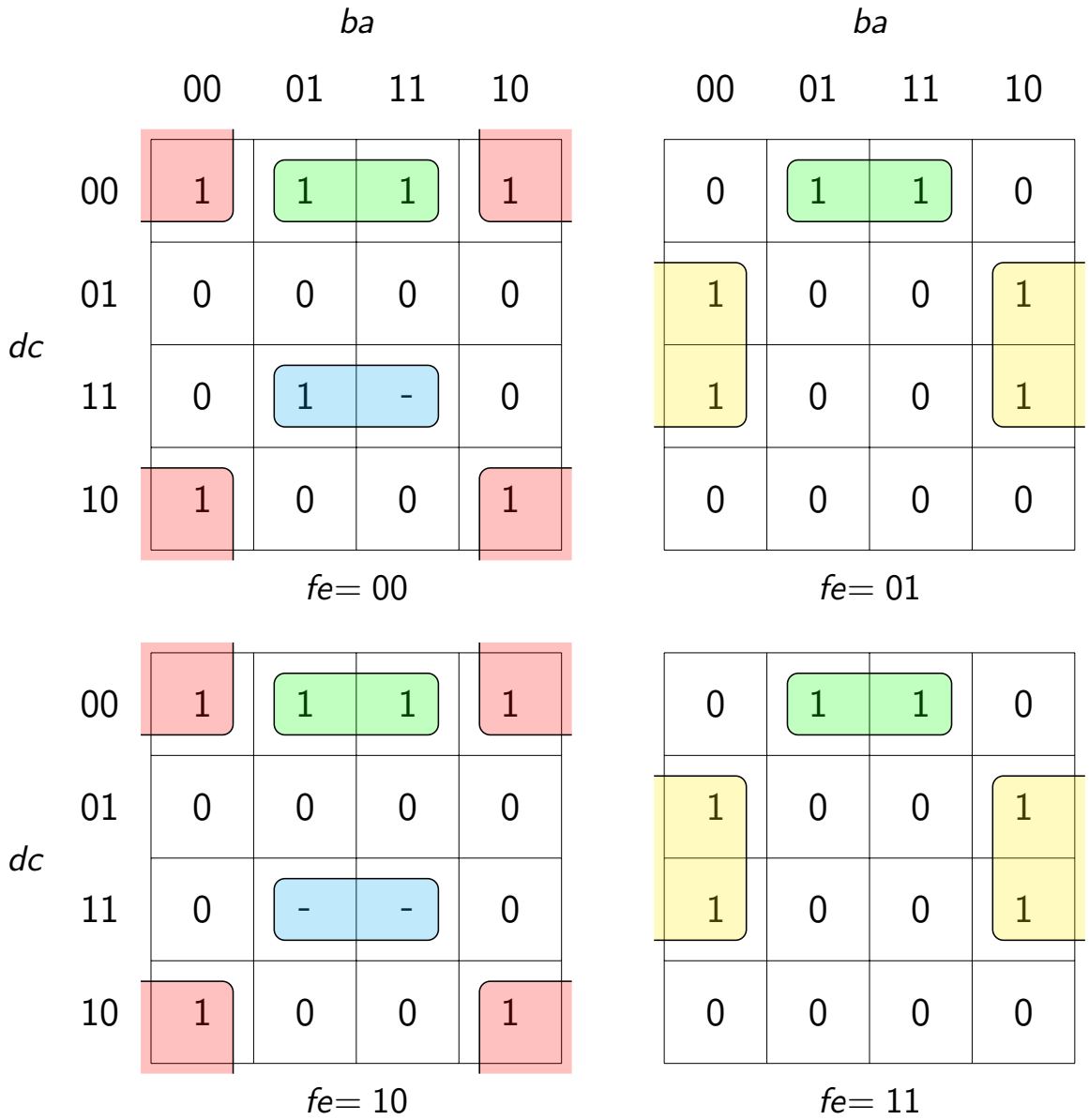
(m) Table sans tenir compte de la valeur indéterminée.



(n) Table en tenant compte de la valeur indéterminée.

Table de Karnaugh à plus de 4 variables

Exemple de table de Karnaugh à 6 variables



Fonctions booléennes et multiplets (sur n bits)

Fonction $f : \{0,1\}^N \rightarrow \{0,1\}^M$

- $\underline{y} = f(\underline{x})$
- Introduction de fonctions à valeur sur $\{0,1\}$ telles que
  - $f_0, \ldots, f_{M-1}$
  - $y_0 = f_0(\underline{x}) = f_0(x_0, \ldots, x_{N-1})$
  - $\vdots$
  - $y_{M-1} = f_{M-1}(\underline{x})$

Fonctions booléennes usuelles

Comparaison CMP sur N bits

Tranche i sur 1 bit

X_i	Y_i	S_i
0	0	1
0	1	0
1	0	0
1	1	1

Figure – Test d'égalité - table de vérité.

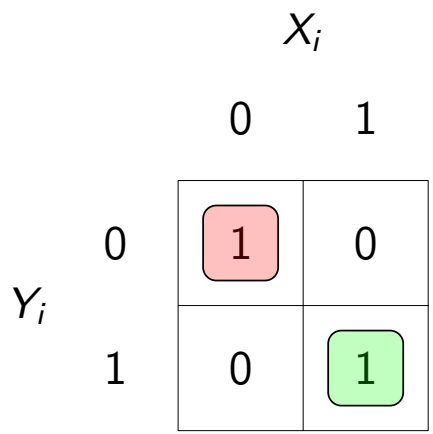
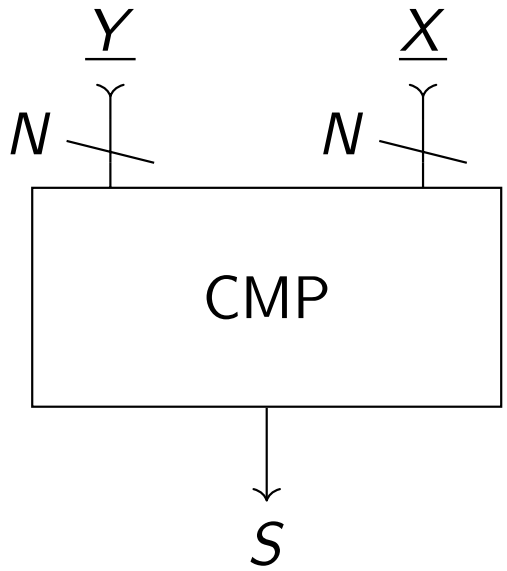


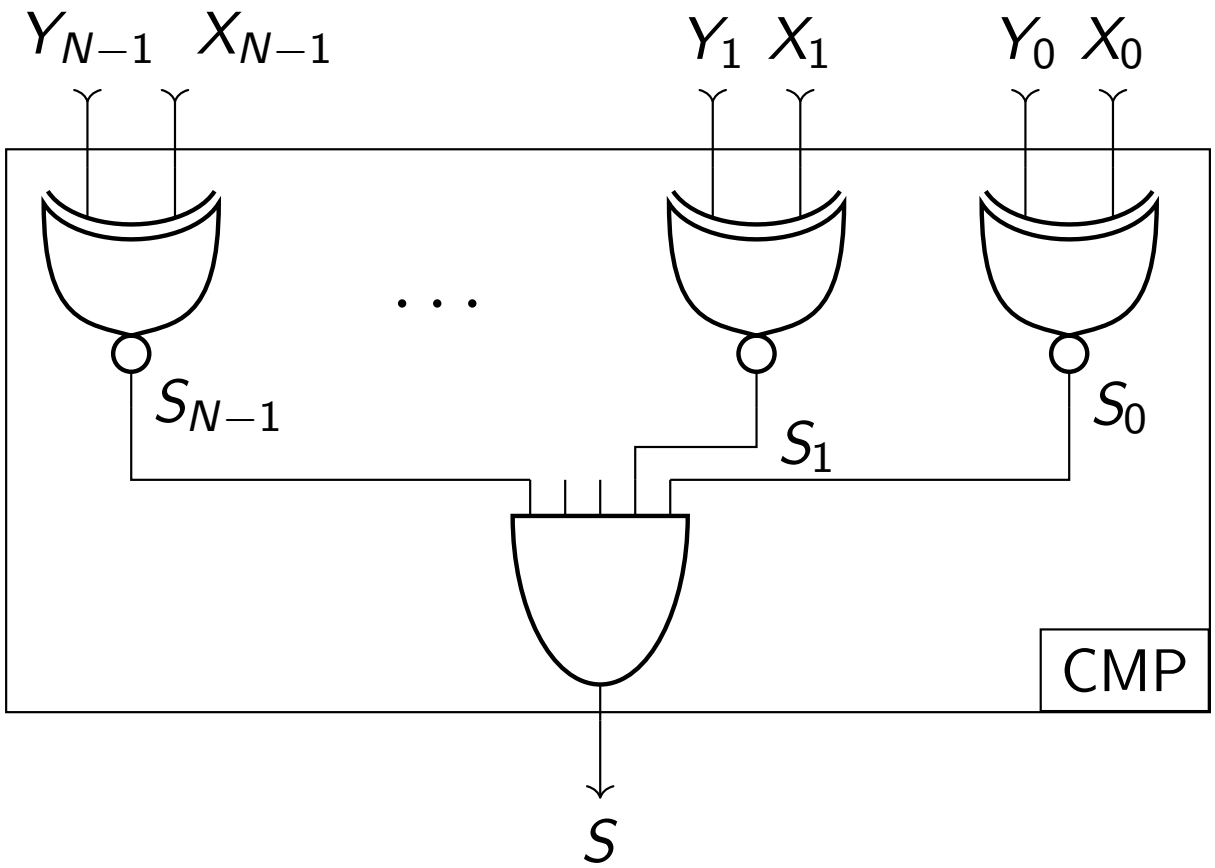
Figure – Test d'égalité - table de Karnaugh

Comparaison CMP sur N bits



Comparaison CMP sur N bits

Resynthèse des tranches



Additionneur ADD sur N bits

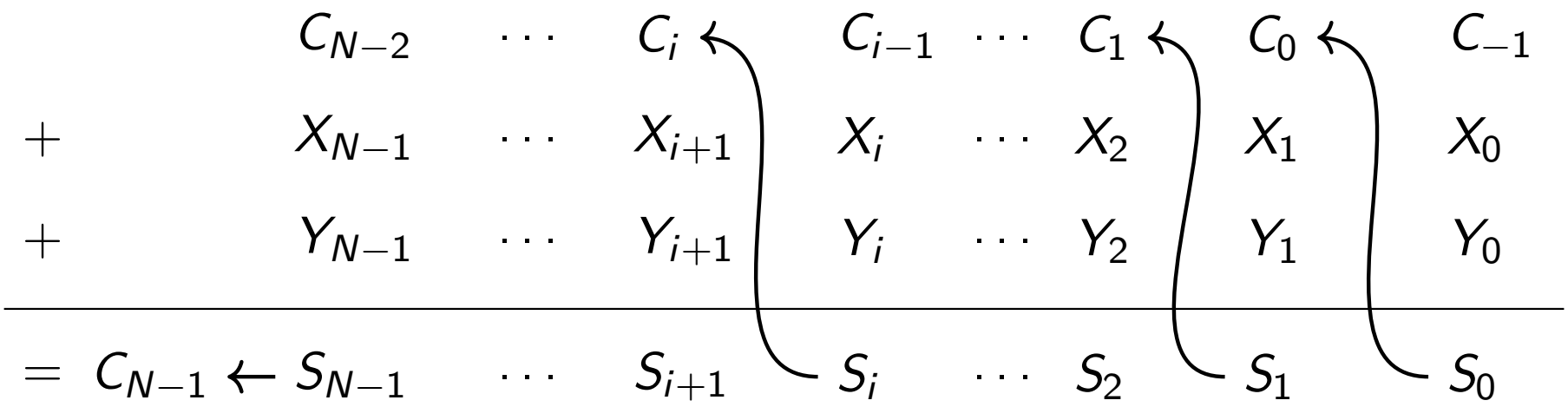


Figure – Algorithme d'addition sur N bits avec retenues amont et aval.

Additionneur ADD sur N bits

Tranche sur 1 bit

C_{i-1} \ X_iY_i	00	01	11	10
0	0	1	0	1
1	1	0	1	0

(l) La fonction de sortie S_i.

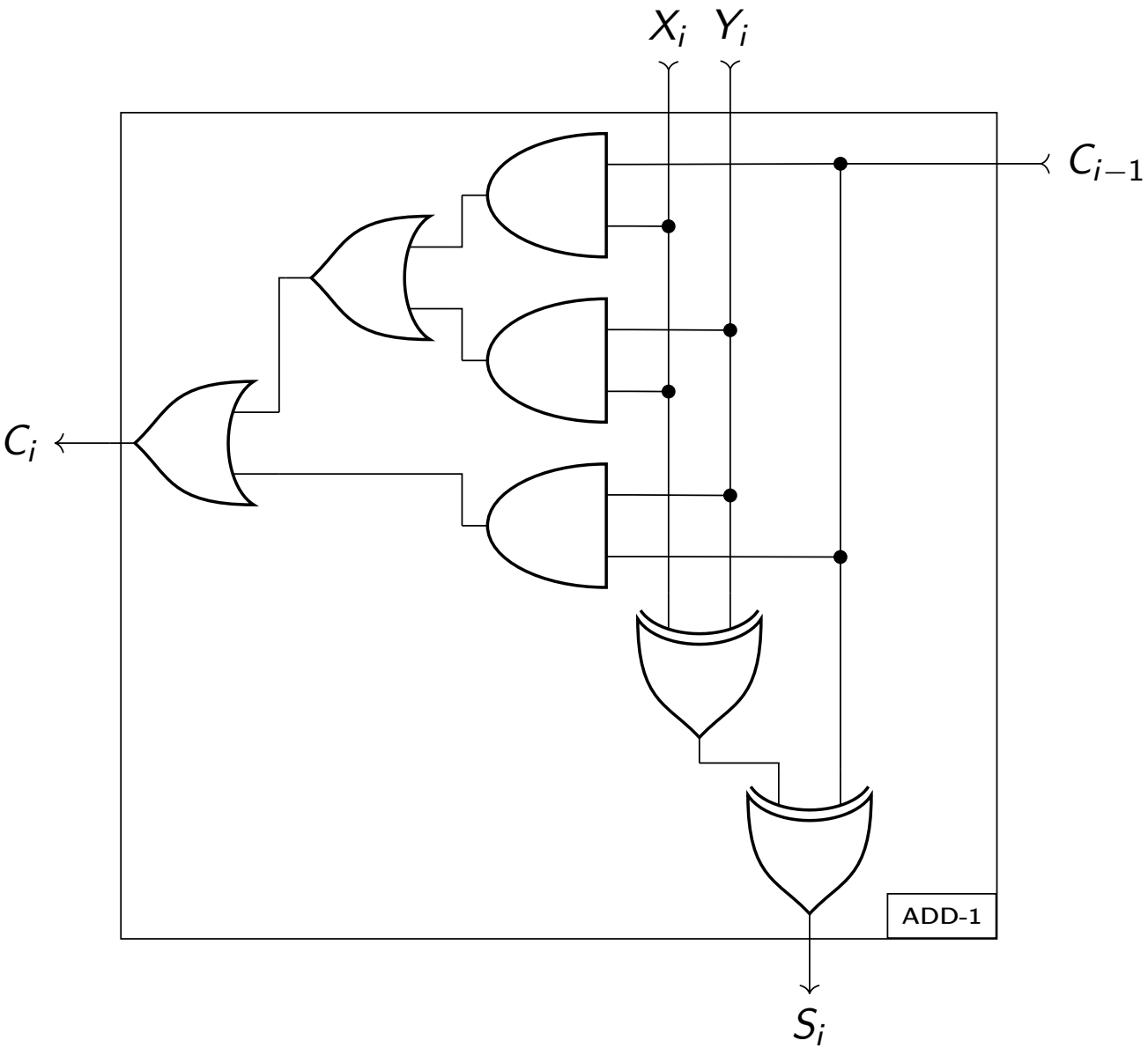
C_{i-1} \ X_iY_i	00	01	11	10
0	0	0	1	0
1	0	1	1	1

(m) La fonction de retenue C_i.

Table – Tables de Karnaugh d'un additionneur 1 bit avec retenues amont et aval.

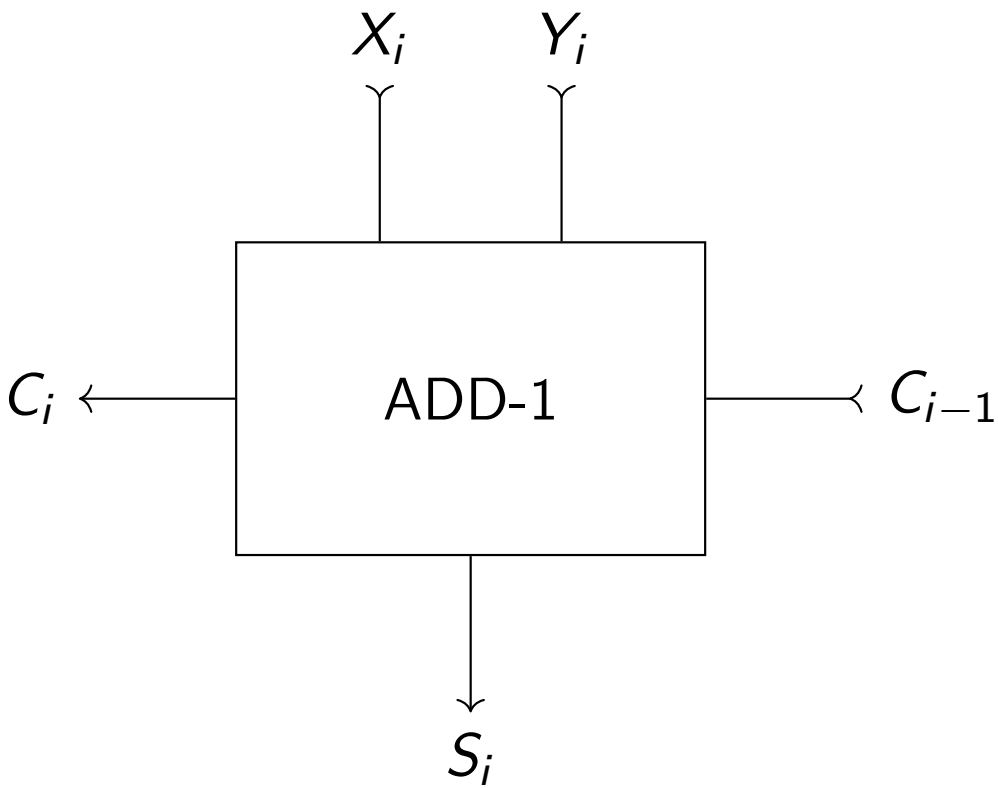
Additionneur ADD sur N bits

Tranche sur 1 bit

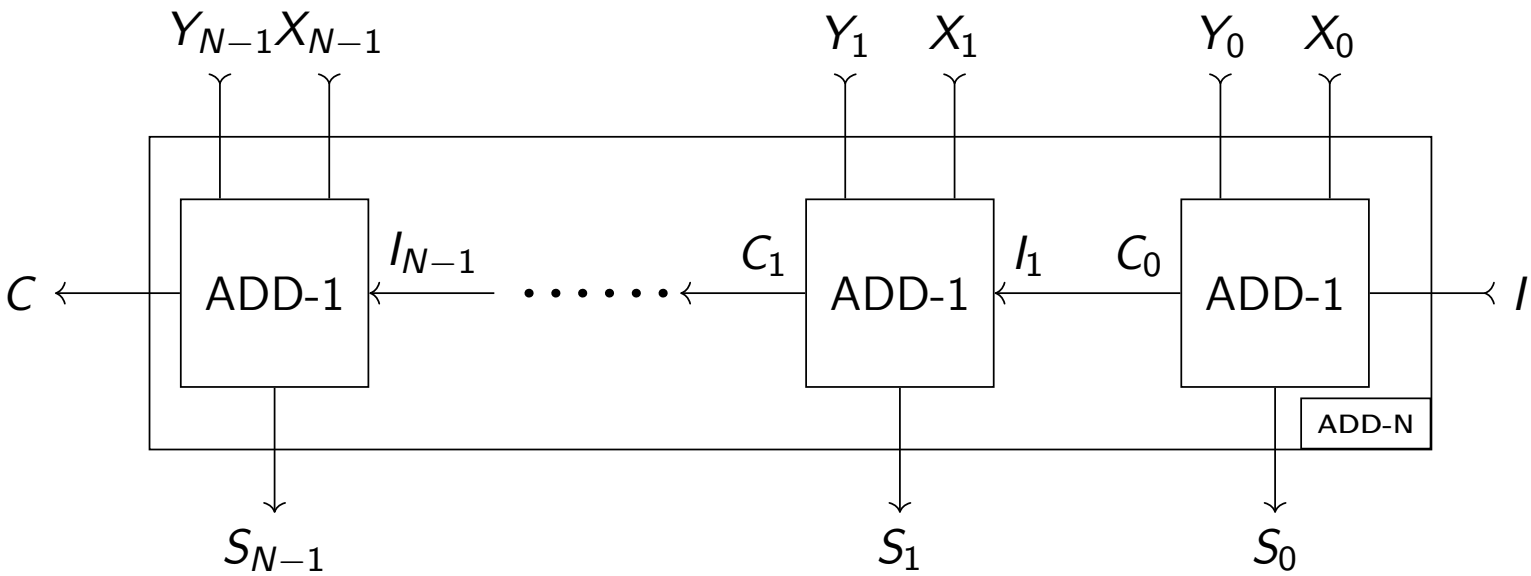


Additionneur ADD sur N bits

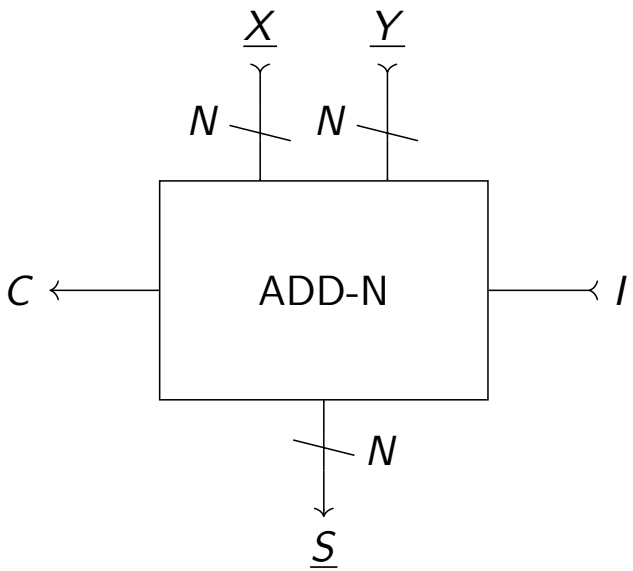
Tranche sur 1 bit



Additionneur ADD sur N bits



Additionneur ADD sur N bits



Inverseur commandé *CINV* sur *N* bits

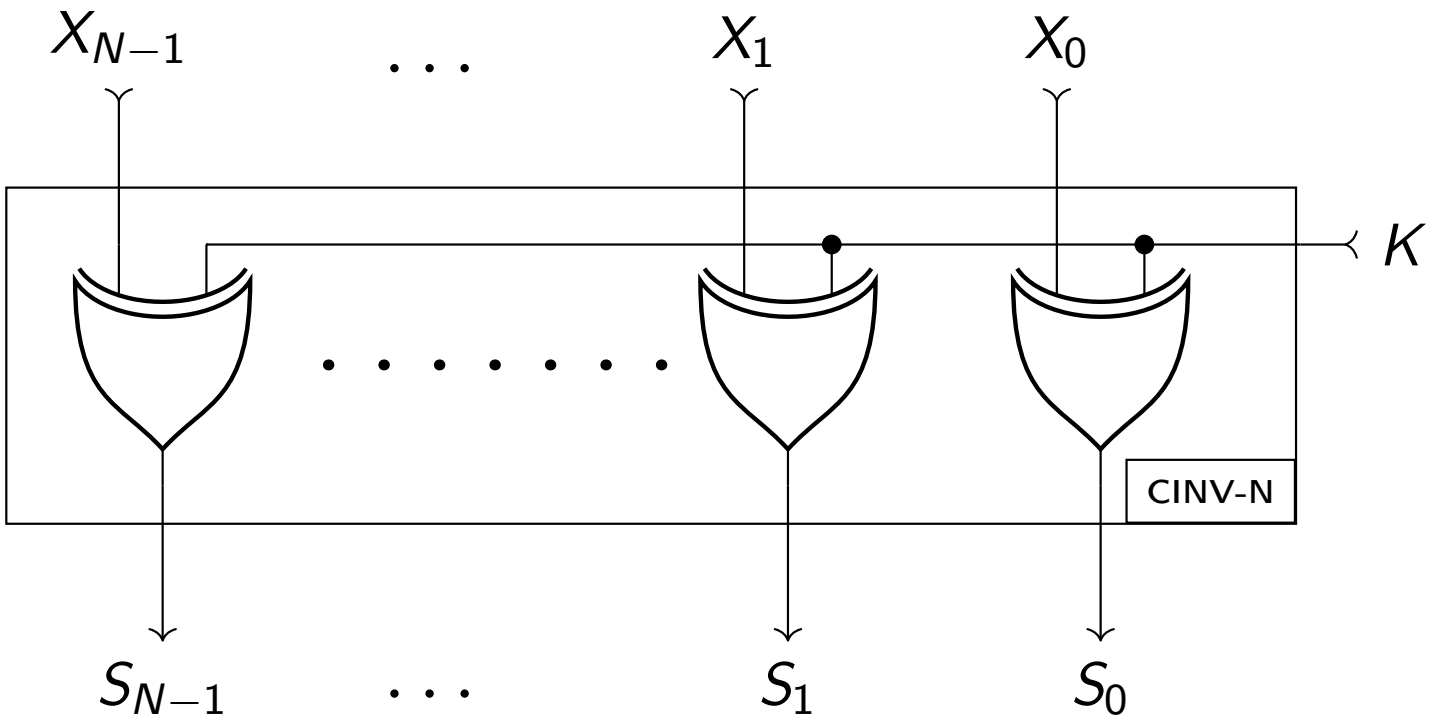
- Si $K = 0$, alors la sortie $\underline{S} = \underline{X}$;
- Si $K = 1$, alors la sortie $\underline{S} = \overline{\underline{X}}$.

Analyse par tranche sur 1 bit

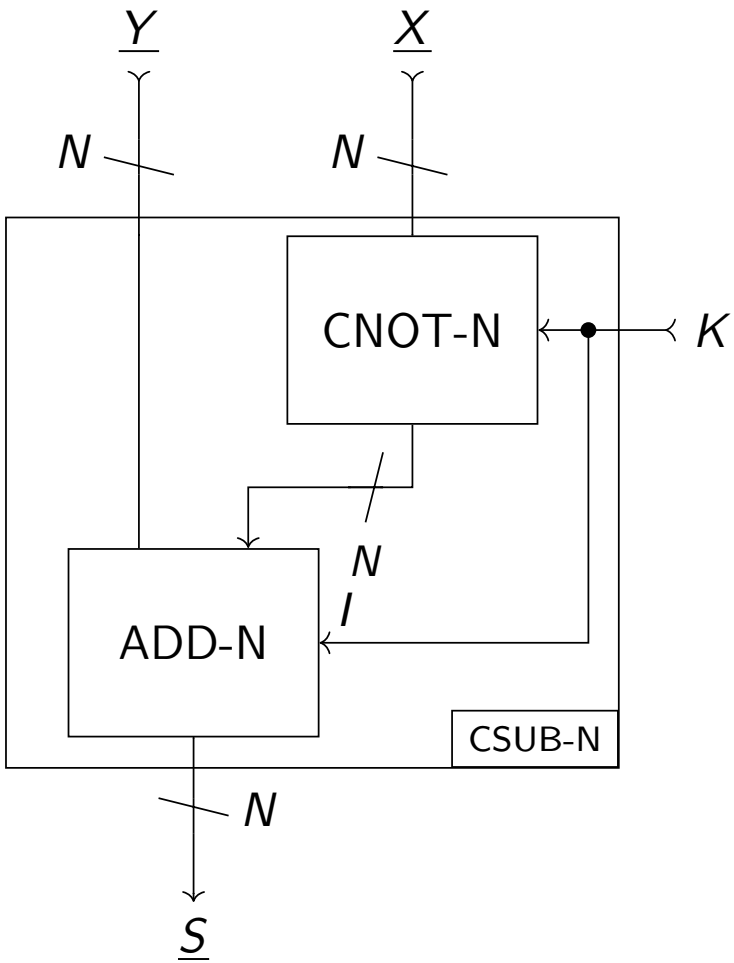
K \ X_i	0	1
0	0	1
1	1	0

Table – Table de Karnaugh d'un inverseur commandé sur 1 bit.

Inverseur commandé *CINV* sur *N* bits

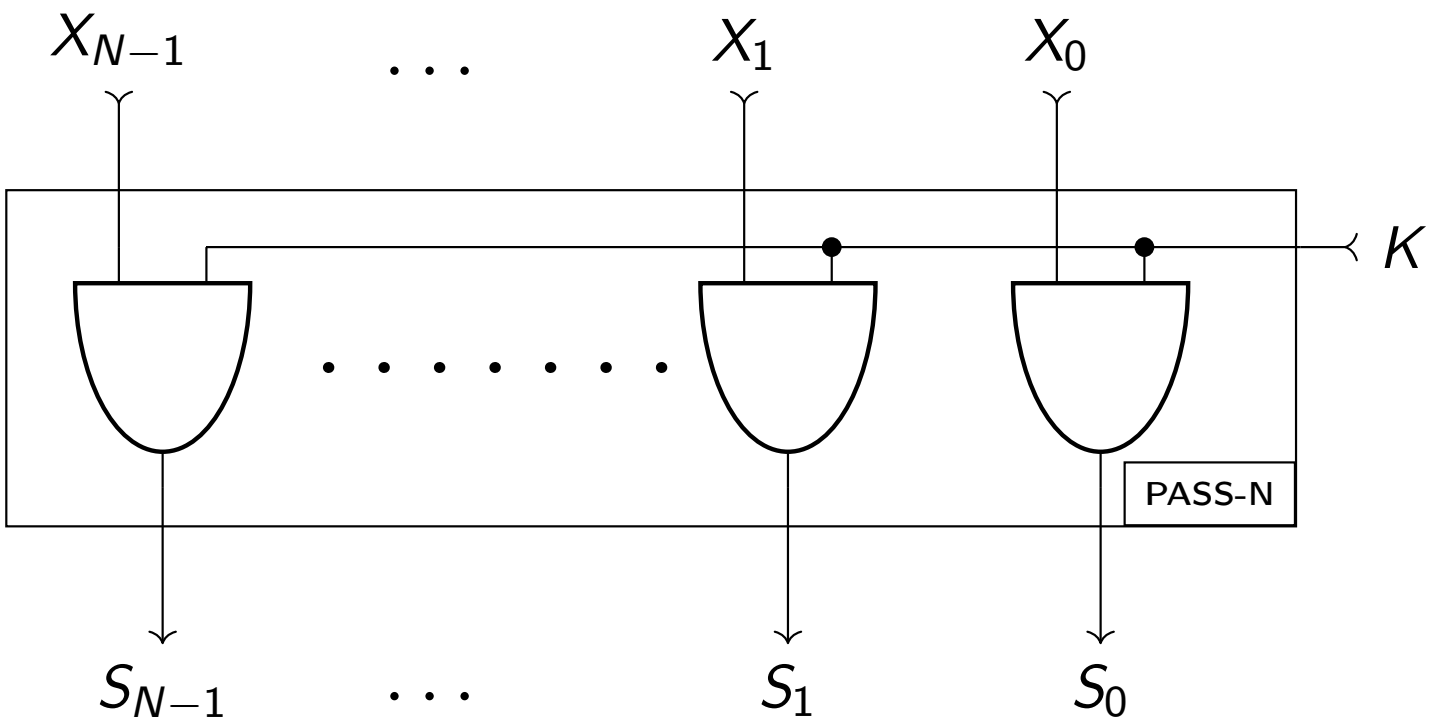


Soustracteur commandé *CSUB* sur *N* bits



Passeur *PASS* sur N bits

- Si $K = 0$, alors la sortie $\underline{S} = \underline{0}$;
- Si $K = 1$, alors la sortie $\underline{S} = \underline{X}$.



Codeur/Décodeur ENC/DEC sur N bits

Codeur 2^N vers N

X_3	X_2	X_1	X_0	S_1	S_0
0	0	0	0	0	0
0	0	0	1	0	0
0	0	1	-	0	1
0	1	-	-	1	0
1	-	-	-	1	1

(a) Codeur 4 vers 2.

X_3	X_2	X_1	X_0	S_1	S_0	G
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	-	0	1	1
0	1	-	-	1	0	1
1	-	-	-	1	1	1

(b) Codeur 4 vers 2 corrigé.

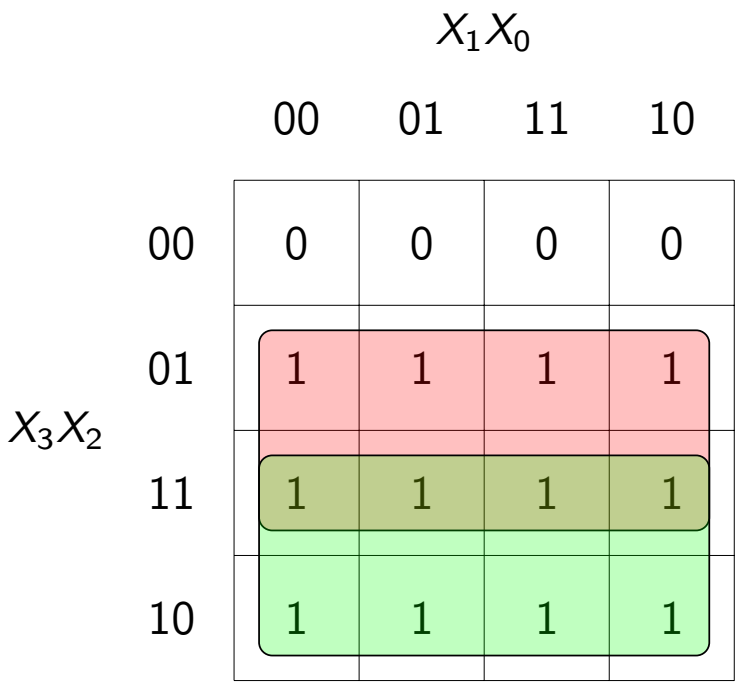
Table – Codeur 4 vers 2.

On en déduit les équations de ceux-ci :

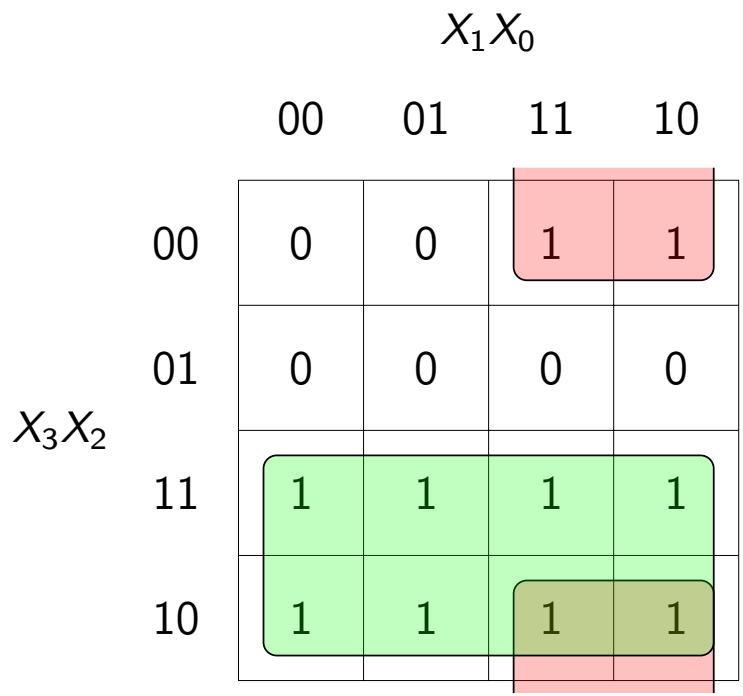
- $S_1 = X_2 + X_3$,
- $S_0 = X_3 + \overline{X_2}.X_1$,
- et par déduction logique $G = X_0 + X_1 + X_2 + X_3$.

Codeur/Décodeur ENC/DEC sur N bits

Codeur 2^N vers N



(a) Table de Karnaugh du bit S_1.

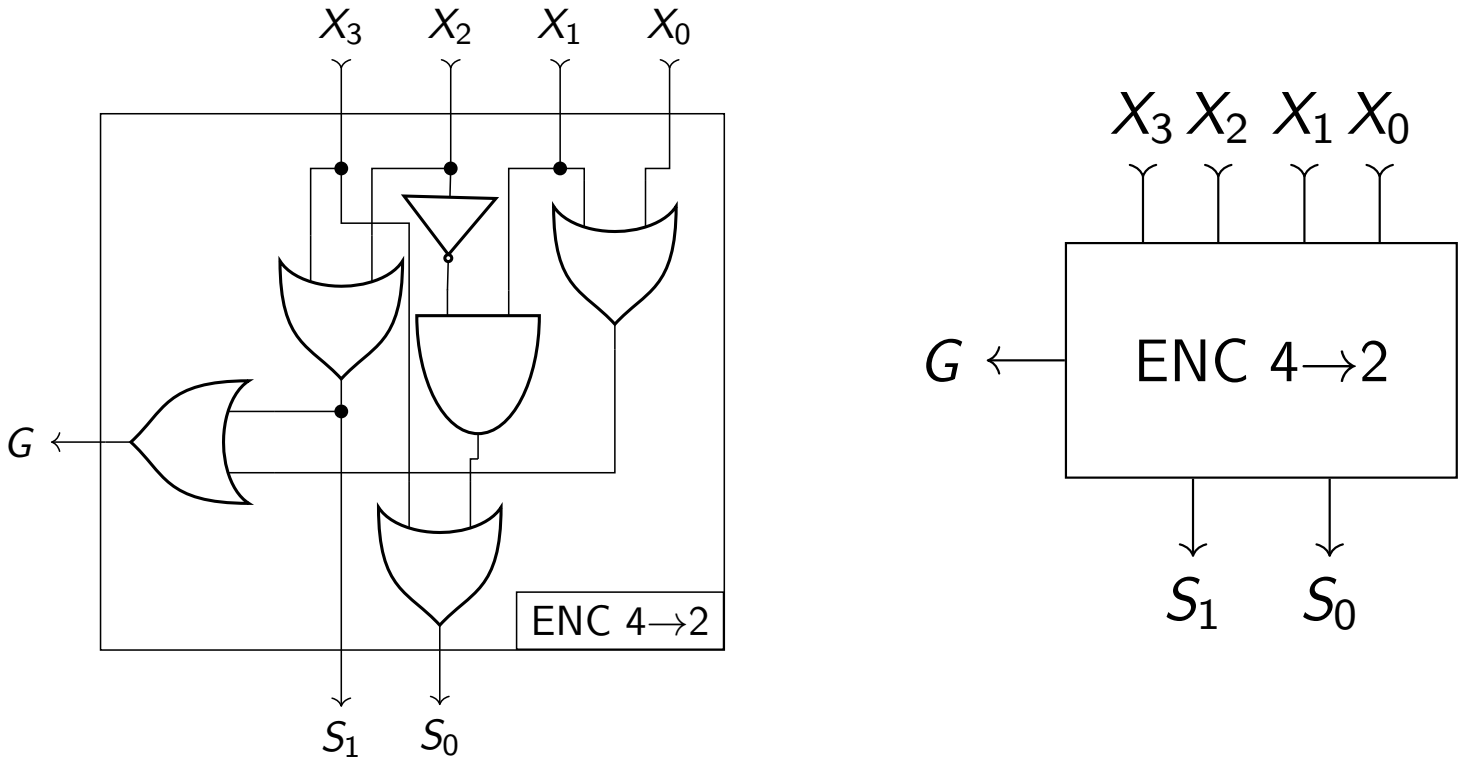


(b) Table de Karnaugh du bit S_0.

Table – Tables de Karnaugh des sorties d'un codeur 4 vers 2.

Codeur/Décodeur ENC/DEC sur N bits

Codeur 2^N vers N



(a) Schéma de réalisation.

(b) Représentation simplifiée.

Figure – Encodeur 4 vers 2.

Codeur/Décodeur ENC/DEC sur N bits

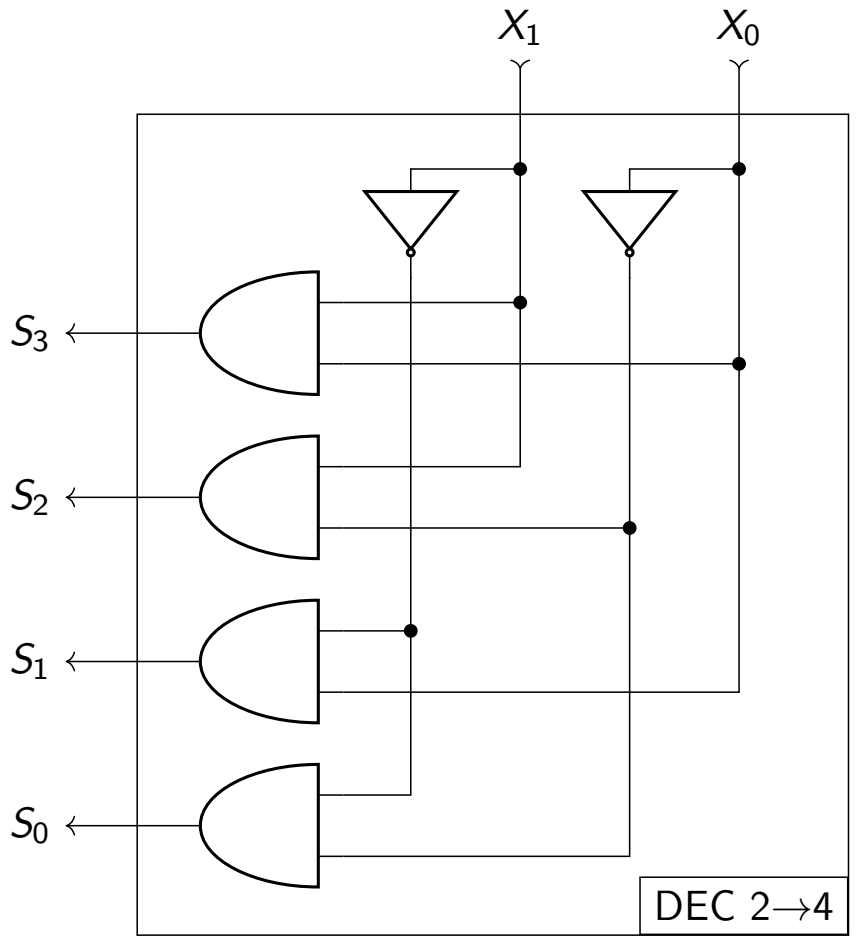
Décodeur N vers 2^N

X_1	X_0	S_3	S_2	S_1	S_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

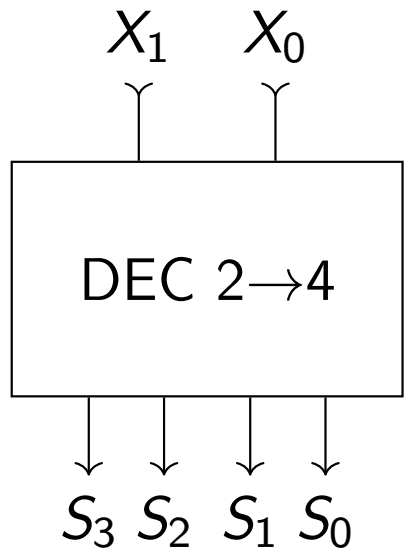
Table – Table de vérité d'une décodeur 2 vers 4.

Codeur/Décodeur ENC/DEC sur N bits

Décodeur N vers 2^N



(a) Schéma de réalisation.



(b) Représentation simplifiée.

Figure – Décodeur 2 vers 4.

Multiplexeur/Démultiplexeur sur N bits

Multiplexeur M vers 1

S	X_1	X_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

(a) Table de vérité.

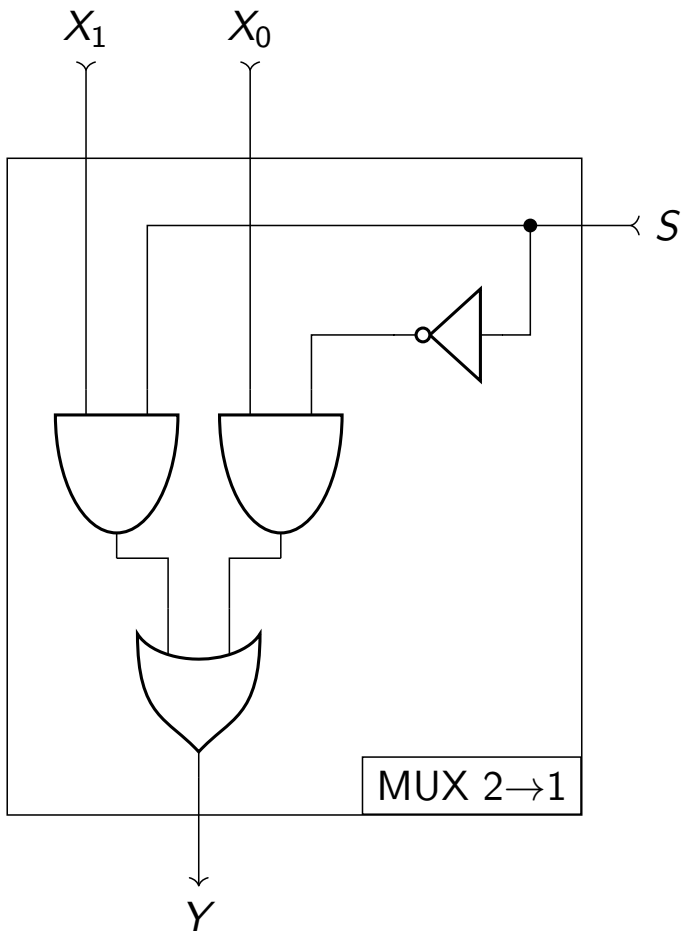
S \ X_1X_0	00	01	11	10
0	0	1	1	0
1	0	0	1	1

(b) Table de Karnaugh.

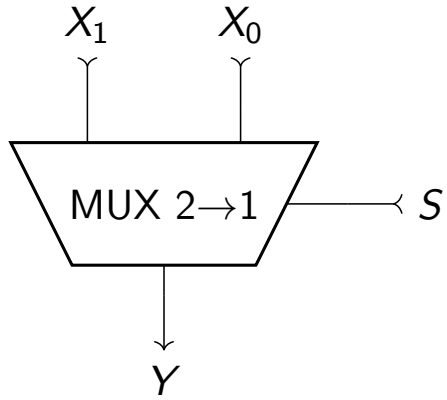
Table – Multiplexeur 2 vers 1 sur 1 bit.

Multiplexeur/Démultiplexeur sur N bits

Multiplexeur M vers 1



(a) Schéma de réalisation.



(b) Représentation simplifiée.

Figure – Multiplexeur 2 vers 1 sur 1 bit.

Multiplexeur/Démultiplexeur sur N bits

Multiplexeur M vers 1

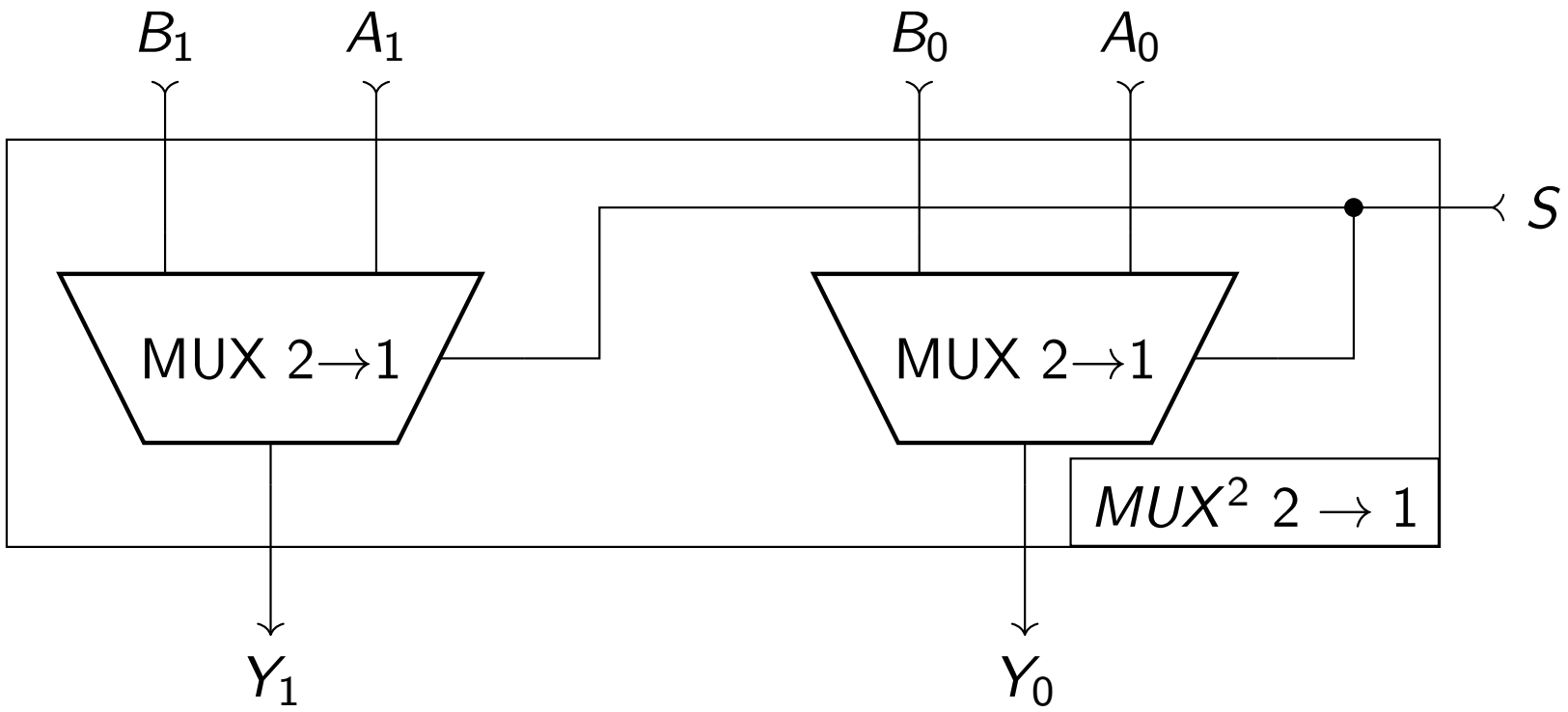


Figure – Multiplexeur 2 vers 1 sur 2 bits.

Multiplexeur/Démultiplexeur sur N bits

Multiplexeur M vers 1

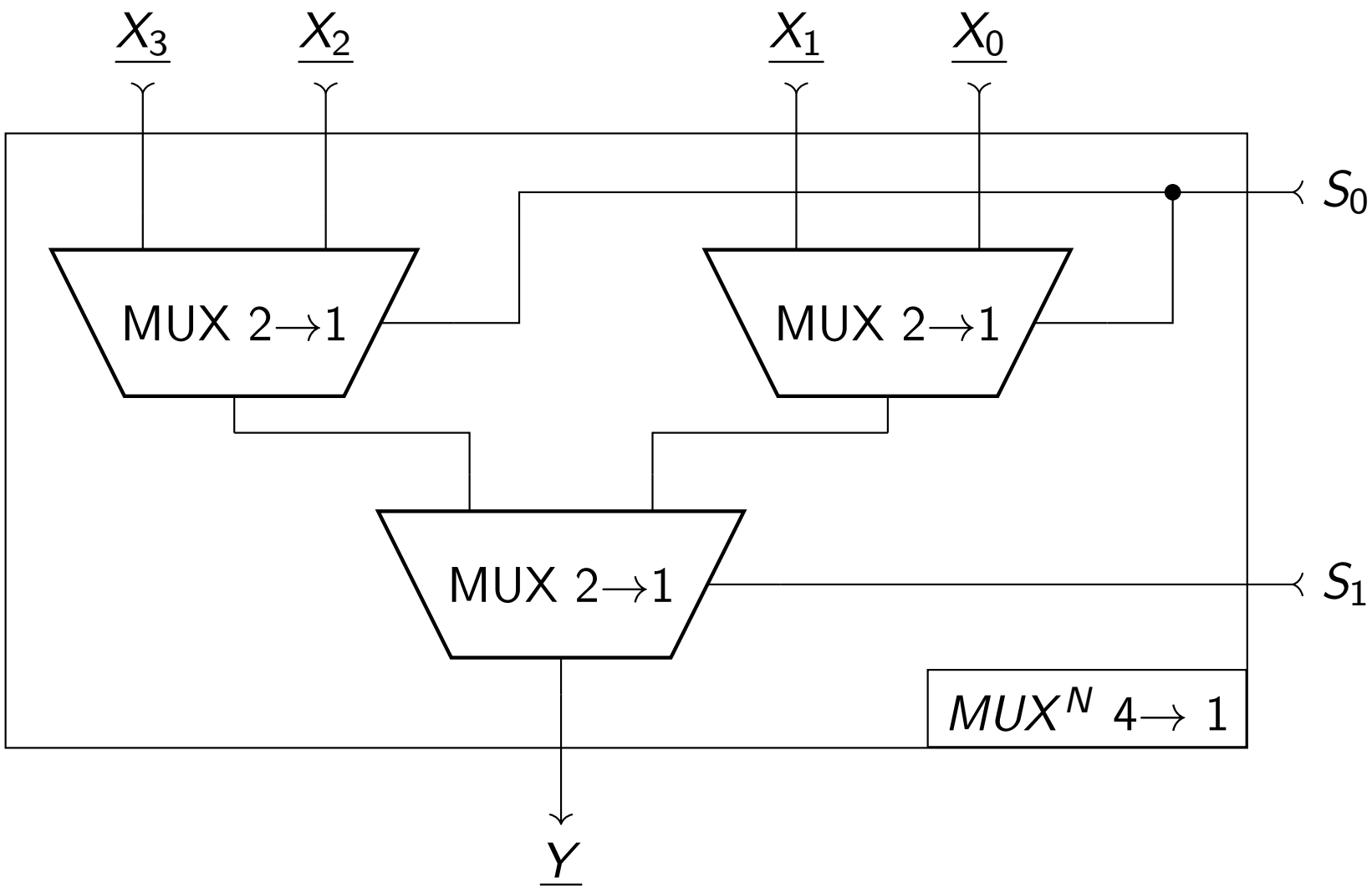
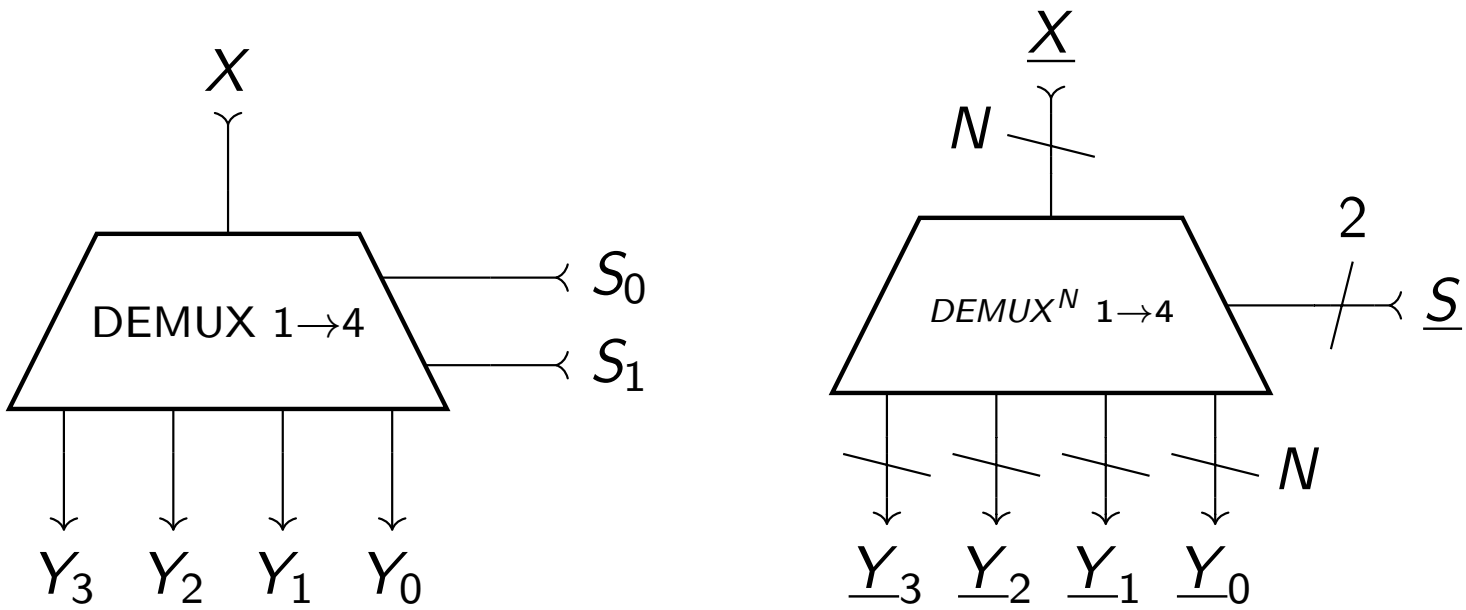


Figure – Multiplexeur 4 vers 1 sur N bits, réalisé à partir d'un arbre de MUX2→1.

Multiplexeur/Démultiplexeur sur N bits

Démultiplexeur 1 vers M



(a) Version 1 bit. (b) Version N bits.

Figure – Représentation d'un démultiplexeur 1 vers 4.

Décaleur sur N bits

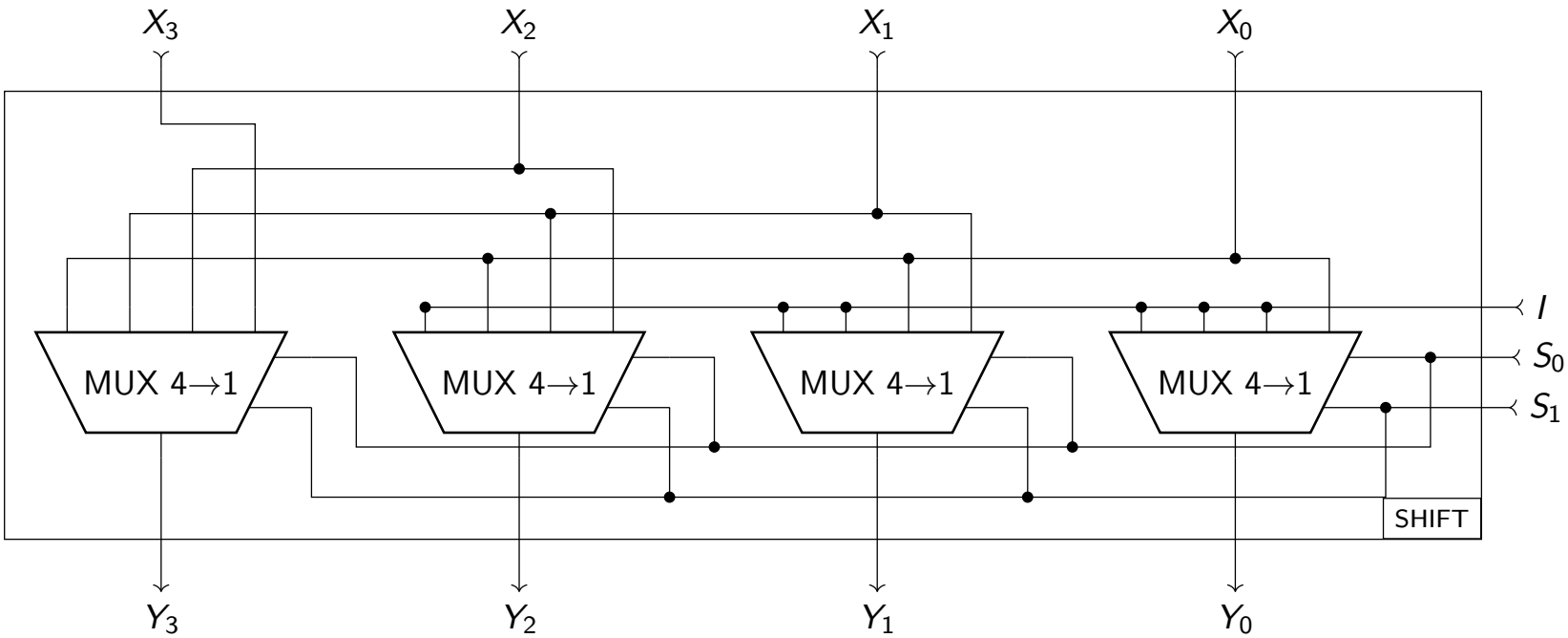


Figure – Décaleur à gauche de 0 à 3 bits sur un mot de 4 bits.

Décaleur sur N bits

- Si $\underline{S} = 00$ alors on en fait rien, $\underline{Y} = \underline{X}$;
- Si $\underline{S} = 01$ alors on décale à droite d'un bit, et $Y_3 = I_L$;
- Si $\underline{S} = 10$ alors on décale à gauche d'un bit, et $Y_0 = I_R$.

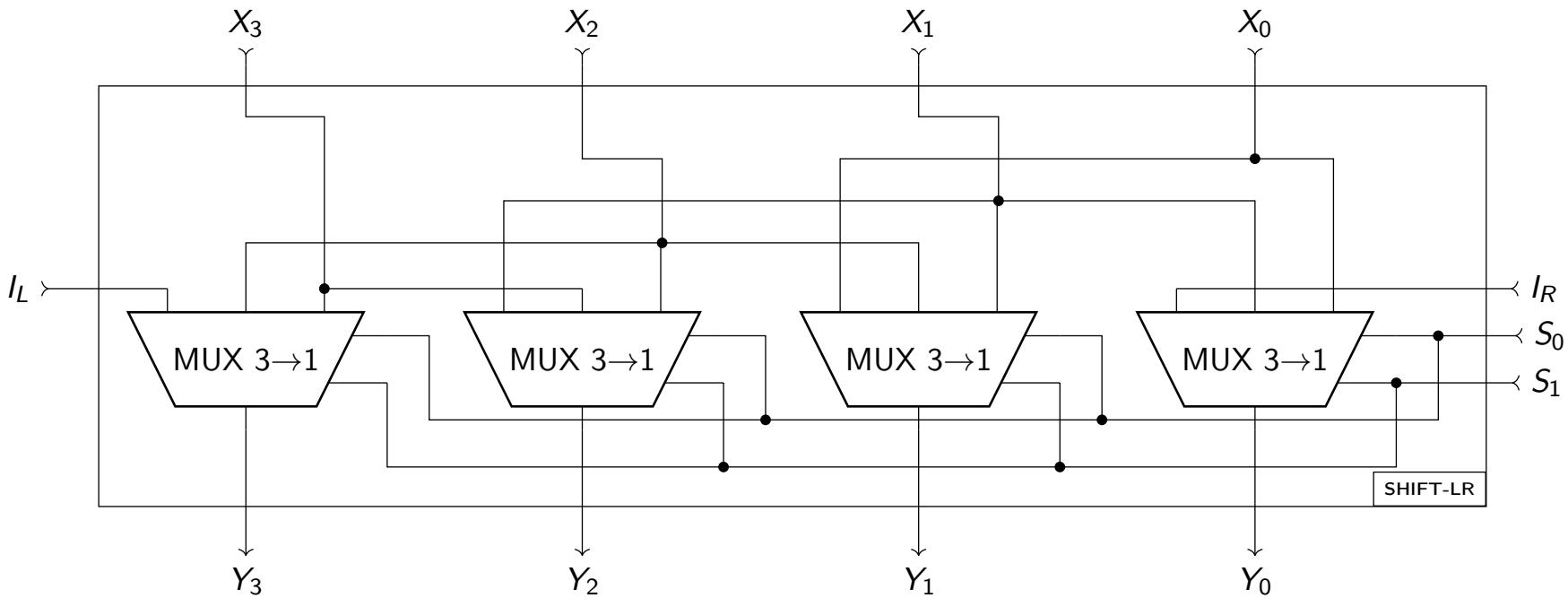


Figure – Décaleur gauche/droite d'un bit.

Unité Arithmétique et Logique UAL sur N bits

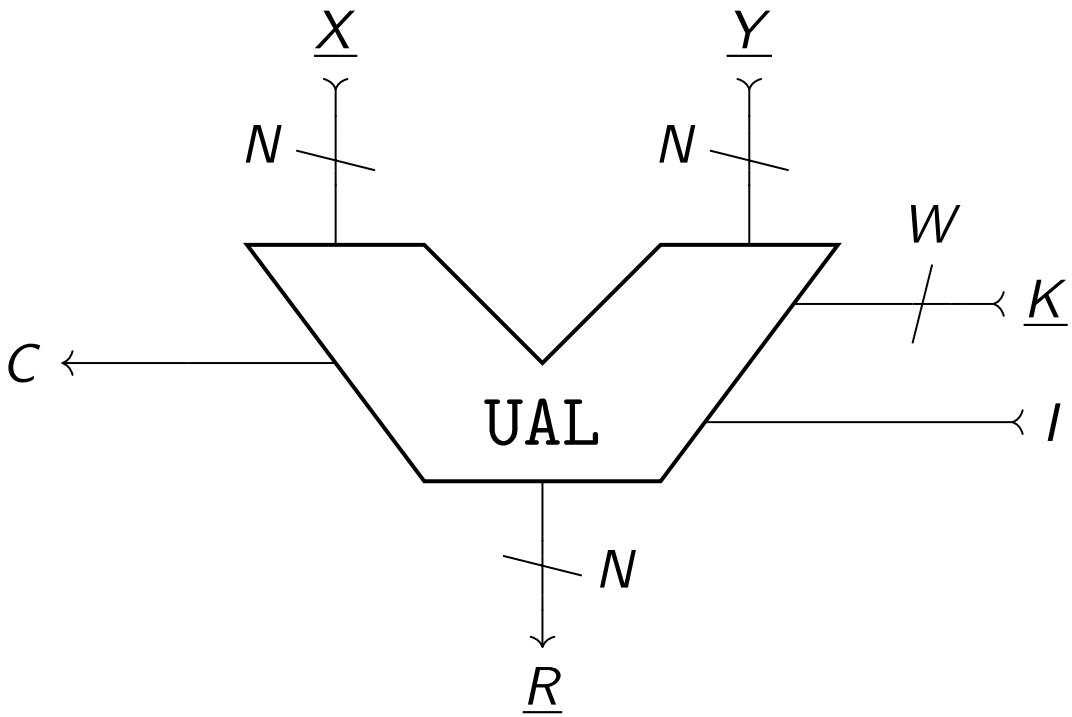


Figure – Représentation d'une Unité Arithmétique et Logique sur N bits.

Unité Arithmétique et Logique UAL sur N bits

- *PASS* : un passeur, utilisé par exemple pour l'instruction *CLA* ;
- *CSUB* : un soustracteur commandé de deux mots sur N bits ;
- *SHIFLR* : un décaleur commandé gauche-droit sur 1 bit ;
- *CMP* : un comparateur d'égalité entre les deux entrées.

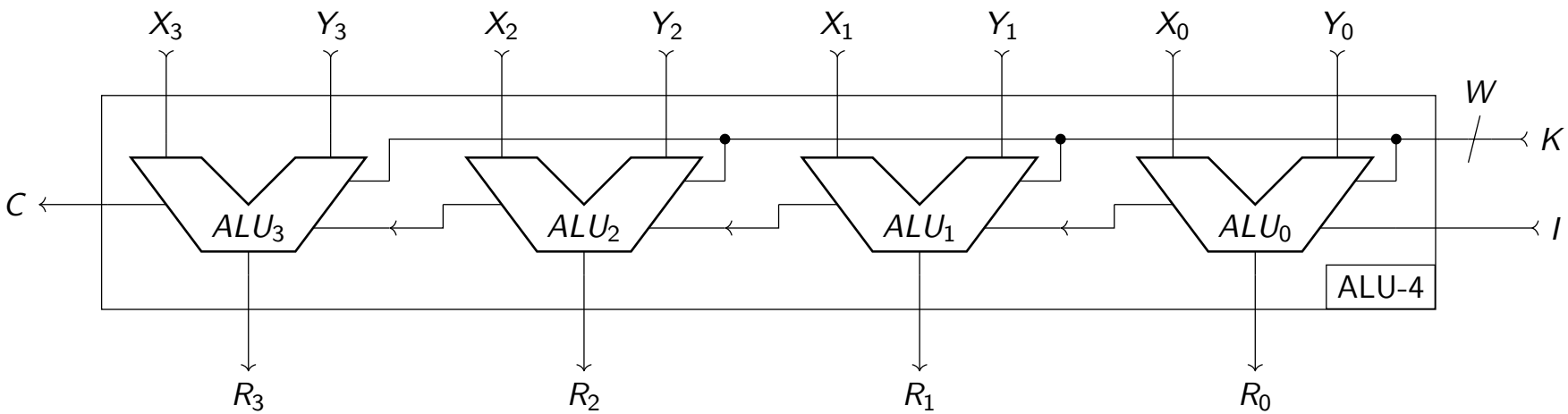


Figure – Décomposition d'une *UAL* 4 bits en tranches.

Temps de calcul — temps de propagation

- la sortie ne change sûrement pas d'état pendant T_{pmin} après un événement en entrée ;
- la sortie a sûrement pris sa nouvelle valeur de manière stable à T_{pmax} après un événement en entrée.

Deux portes A et B en cascade

- $T_p(Y|X)$ le temps de propagation pour le sortie Y par rapport à un changement sur l'entrée X pour la porte logique A ;
- $T_p(Z|Y)$ le temps de propagation pour le sortie Z par rapport à un changement sur l'entrée Y pour la porte logique B ;

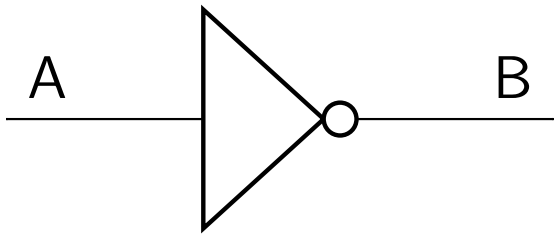
Alors

$$T_p(Z|X) = T_p(Z|Y) + T_p(Y|X)$$

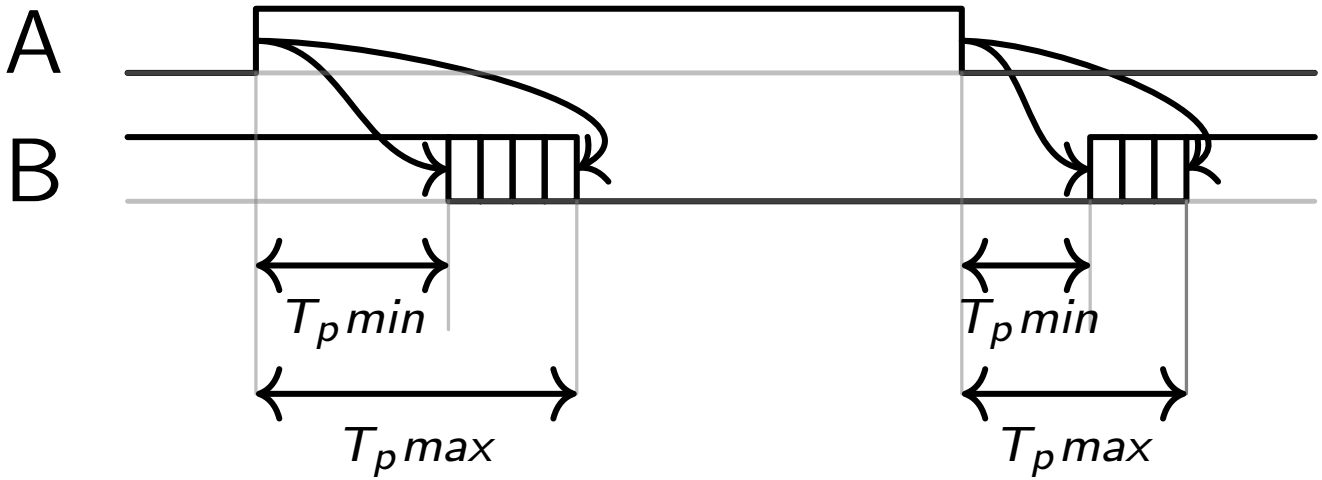
Avec la même conséquence pour la fourchette des temps de propagation :

$$T_pmin(Z|X) = T_pmin(Y|X) + T_pmin(Z|Y)$$
$$T_pmax(Z|X) = T_pmax(Y|X) + T_pmax(Z|Y)$$

Temps de calcul — temps de propagation

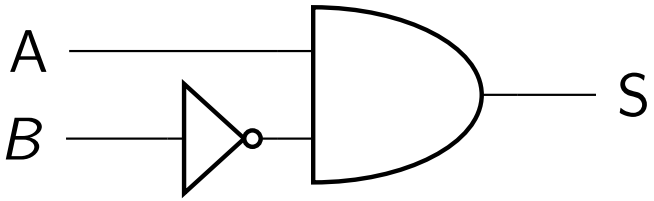


(a) Une porte *Not*.



(b) Chronogramme avec inversion de l'entrée.

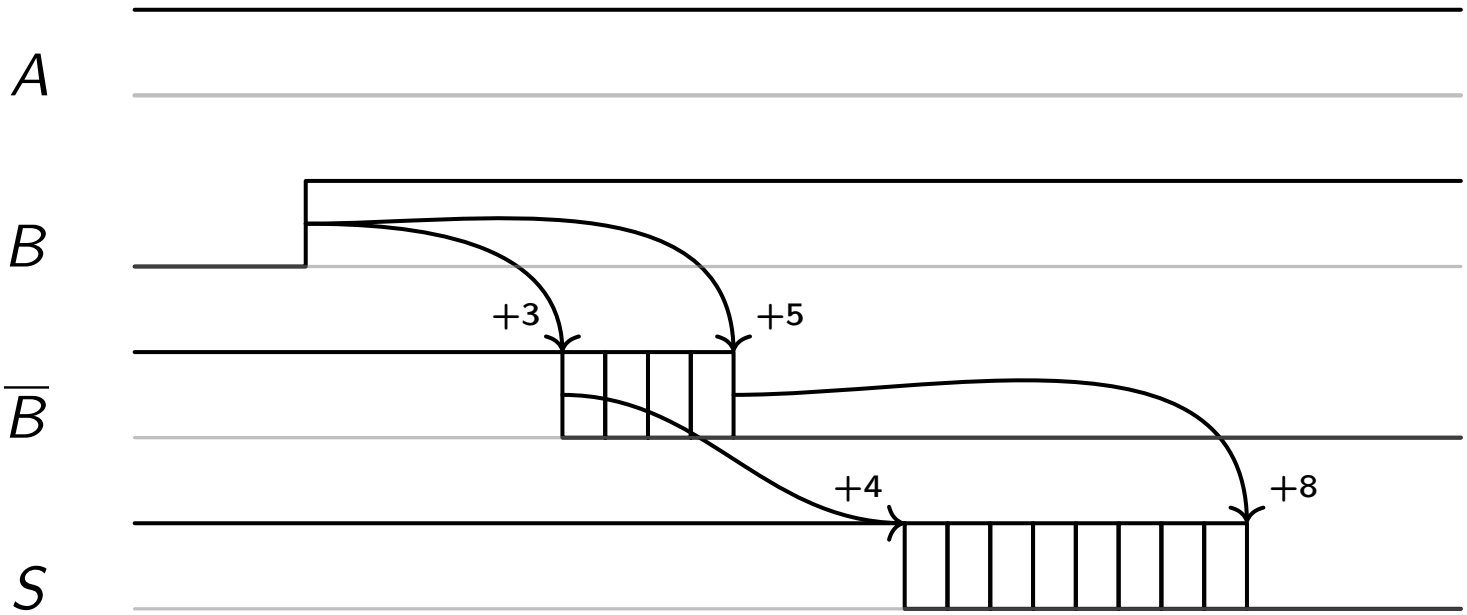
Temps de calcul — temps de propagation



(c) Circuit à 2 portes $S = A.\overline{B}$.

Porte	T_{pmin}	T_{pmax}
Not	3	5
And	4	8

(d) Temps de propagation (en ns).



(e) Chronogramme du circuit.

Aléas et correction

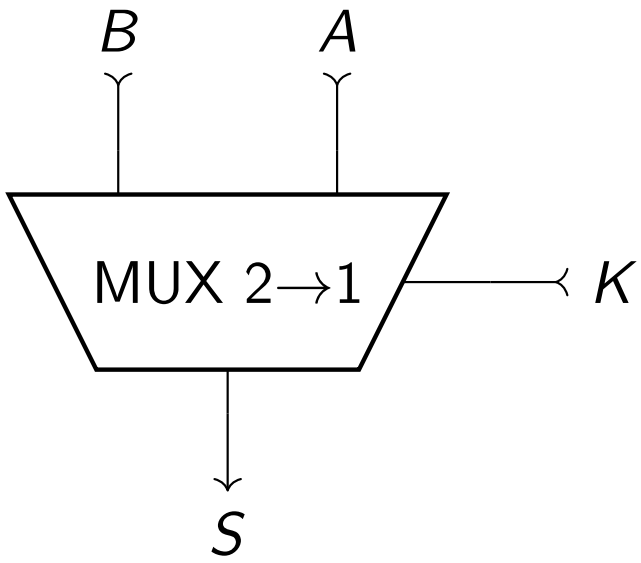
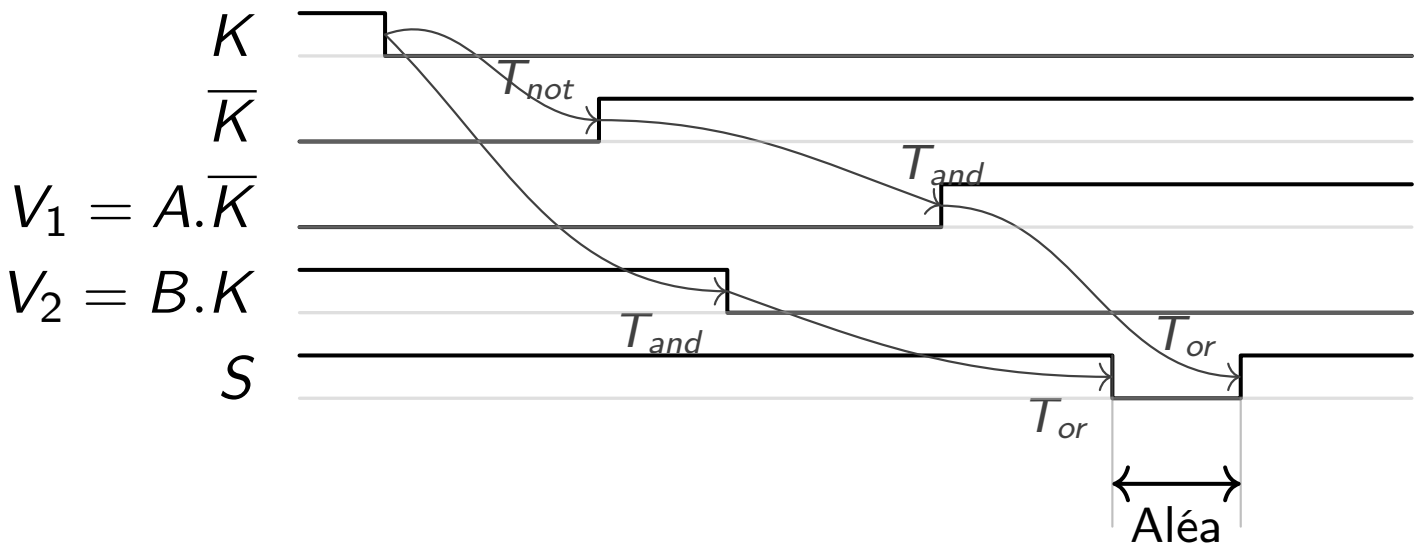


Figure – Multiplexeur 2 vers 1.

Aléas et correction

Temps t	V_1	V_2	$S = V_1 + V_2$
$< T_{and}$	1	0	1
$T_{and} \leq \cdots \leq T_{and} + T_{not}$	0	0	0
$> T_{and} + T_{not}$	0	1	1

Table – Table des états selon le temps depuis le changement de K.



Aléas et correction

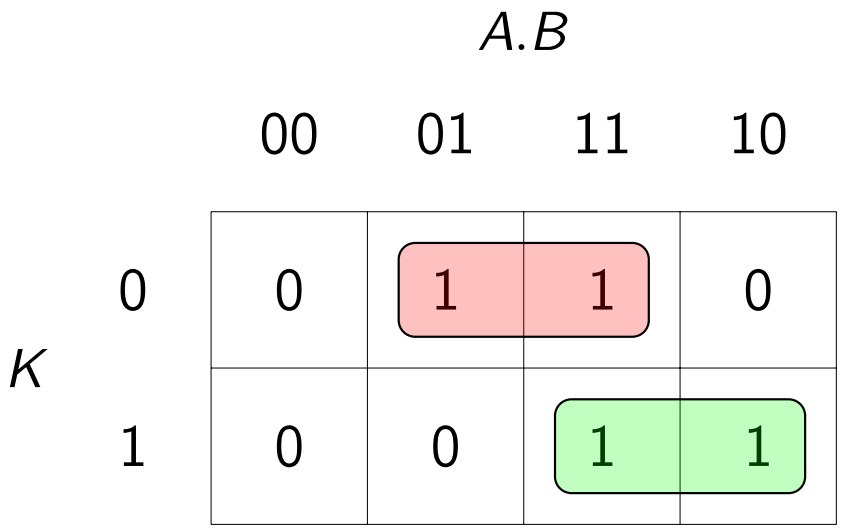


Table – Table de Karnaugh du multiplexeur avec aléa.

Aléas et correction

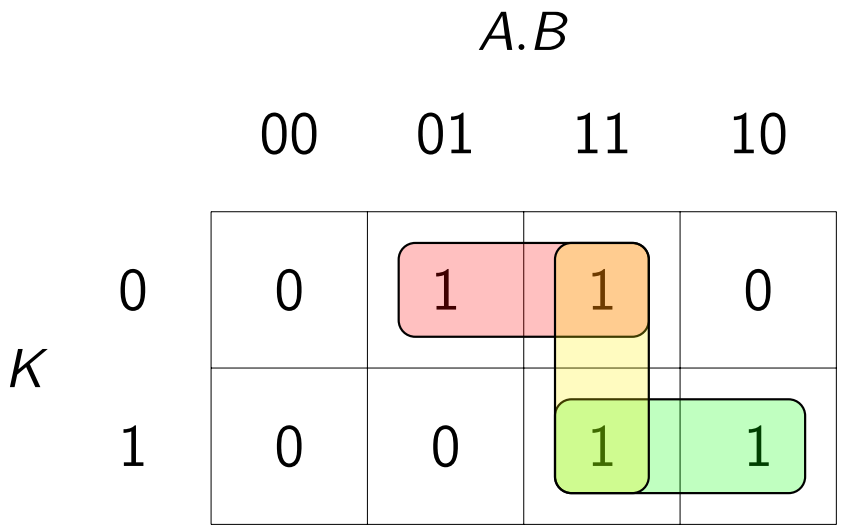


Table – Table de Karnaugh du multiplexeur sans aléa.