Logique combinatoire

Dans ce chapitre nous nous intéressons à une famille de circuits logiques pour lesquels la sortie dépend uniquement des états des entrées.

III.1 Addition binaire

III.1.a Demi-additionneur

Addition et soustraction sont deux opérations arithmétiques de base. Commençons par l'addition de deux nombres binaires, la soustraction sera étudiée dans le prochain paragraphe. En base 2 l'addition de deux bits s'écrit :

$$\begin{cases} 0+0 = 00 \\ 0+1 = 01 \\ 1+0 = 01 \\ 1+1 = 10 \end{cases}$$

Comme en décimal, nous devons donc tenir compte d'une éventuelle retenue (carry). La figure 1 montre la décomposition de l'addition de deux nombres binaires de quatre bits.

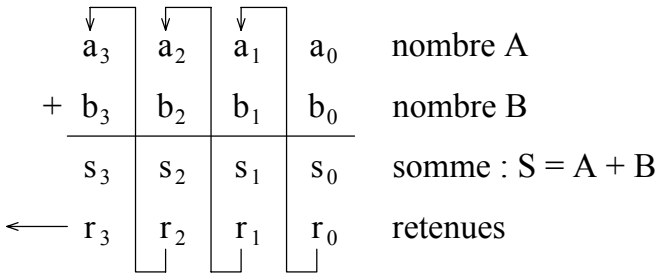


Figure 1

L'addition des deux bits de plus bas poids (LSB : Least Significant Bit) a_0 et b_0, donne un résultat partiel s_0 et une retenue r_0. On forme ensuite la somme des deux bits a_1 et b_1 et de la retenue r_0.

Nous obtenons un résultat partiel s_1 et une retenue r_1. Et ainsi de suite, nous obtenons un résultat sur quatre bits S et une retenue r_3.

Considérons la cellule symbolisée sur la figure 2, comptant deux entrées A et B, les deux bits à sommer, et deux sorties D le résultat de la somme et C la retenue.

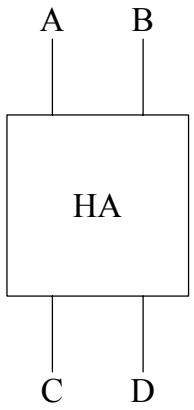


Figure 2

Ce circuit, qui permettrait d'effectuer l'addition des deux bits de plus bas poids est appelé demi-additionneur (Half-Adder). Ecrivons la table de vérité de celui-ci :

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

Table 1

Si nous écrivons ces deux fonctions sous leur forme canonique il vient :

$$\begin{cases} D = \overline{A}\ B + A\ \overline{B} \\ C = A\ B \end{cases}$$

Nous reconnaissons pour la sortie D une fonction OU exclusif, donc :

$$\begin{cases} D = A \oplus B \\ C = A\ B \end{cases}$$

Ce qui peut être réalisé par le circuit schématisé sur le logigramme de la figure 3.

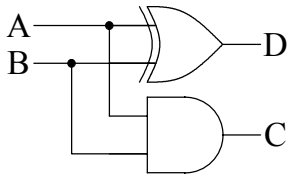


Figure 3

III.1.b Additionneur

Il faut en fait tenir compte de la retenue des bits de poids inférieurs, un circuit additionneur doit donc comporter trois entrées et deux sorties, comme représenté sur la figure 4.

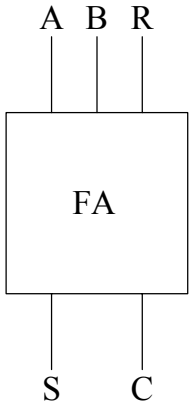


Figure 4

Ce serait possible en combinant deux demi-additionneurs comme présenté par la figure 5. En pratique pour minimiser le nombre de composants, ou de portes dans un circuit intégré, un tel additionneur est réalisé directement.

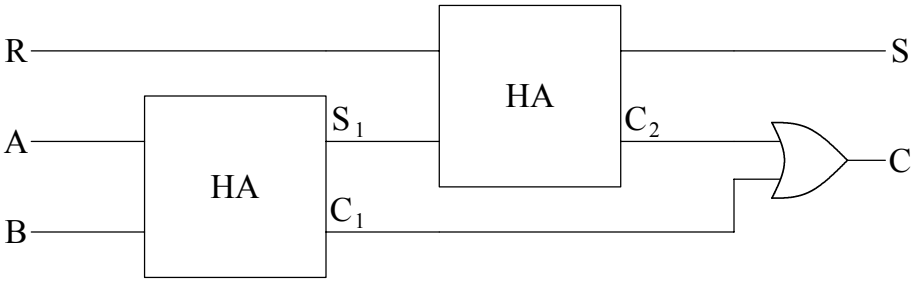


Figure 5

Les entrées A et B représentent les bits à additionner et R le report de la retenue de l'addition des bits de poids inférieurs. La sortie S représente le résultat de la somme et C la retenue. La table de vérité de ce circuit est la suivante :

A	B	R	S	C
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Table 2

A partir de cette table nous pouvons écrire pour S et C les expressions booléennes suivantes :

$$\begin{cases} S = \overline{A}\,\overline{B}\,R + \overline{A}\,B\,\overline{R} + A\,\overline{B}\,\overline{R} + A\,B\,R \\ C = \overline{A}\,B\,R + A\,\overline{B}\,R + A\,B\,\overline{R} + A\,B\,R \end{cases}$$

Nous pouvons simplifier l'expression de C en utilisant un tableau de Karnaugh :

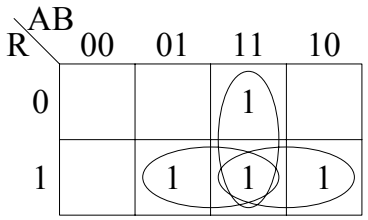


Figure 6

Nous en déduisons :

$$C = A\,B + A\,R + B\,R$$

Le bit de carry est égal à 1 si au moins deux des entrées sont à 1. D'autre part, nous pouvons remarquer qu'intervertir les 0 et les 1 dans la table 2 revient à permuter les lignes 1 et 8, 2 et 7, 3 et 6, 4 et 5. La table de vérité reste globalement invariante par inversion des entrées et des sorties, nous avons donc :

$$\overline{C} = \overline{A}\,\overline{B} + \overline{A}\,\overline{R} + \overline{B}\,\overline{R}$$

A partir de cette relation, qui peut également être démontrée en appliquant l'algèbre de Boole, nous pouvons écrire :

$$\begin{cases} A\,\overline{C} = A\,\overline{B}\,\overline{R} \\ B\,\overline{C} = \overline{A}\,B\,\overline{R} \\ R\,\overline{C} = \overline{A}\,\overline{B}\,R \end{cases} \Rightarrow \quad (A+B+R)\,\overline{C} = A\,\overline{B}\,\overline{R} + \overline{A}\,B\,\overline{R} + \overline{A}\,\overline{B}\,R$$

Ce qui nous permet de réécrire l'expression de S :

$$S = (A+B+R)\,\overline{C} + A\,B\,R$$

La figure 7 donne un exemple de réalisation d'un additionneur 1 bit basé sur deux portes AOI (AND OR INVERT), c'est-à-dire un ensemble de portes ET suivies d'une porte NON-OU.

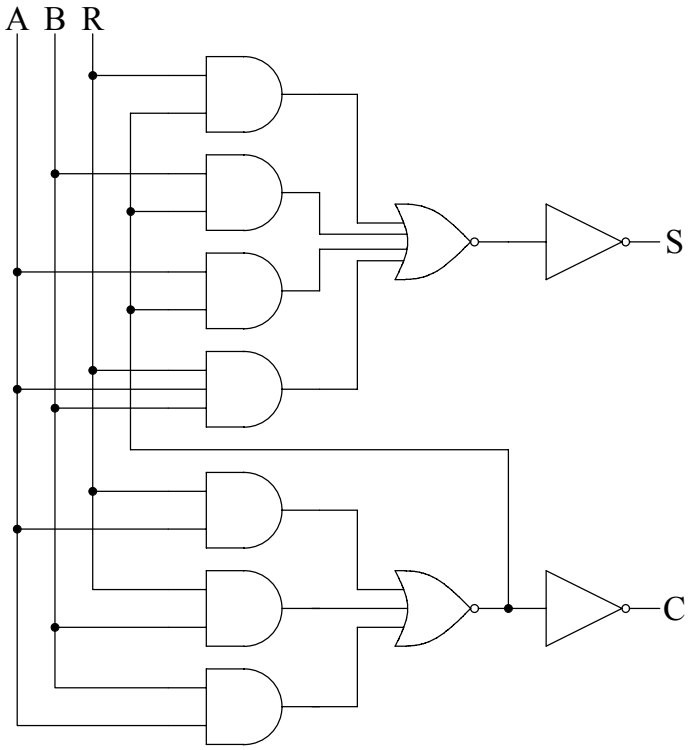


Figure 7

III.1.c Addition en parallèle

L'addition de nombres comptant plusieurs bits peut se faire en série (bit après bit) ou en parallèle (tous les bits simultanément). La figure 8 montre l'exemple d'un additionneur 4 bits comptant quatre "Full Adders", comparables à celui schématisé figure 7, montés en parallèle ou en cascade. Chaque additionneur FA_i est affecté à l'addition des bits de poids i. L'entrée correspondant au report de retenue pour FA_0 est imposée à 0 (en logique positive). La retenue

finale C indique un dépassement de capacité si elle est égale à 1. Le temps d'établissement du résultat correspondant au temps de propagation des retenues au travers des diverses cellules. Si δt est le temps réponse d'une cellule, la sortie S_0 et la retenue R_0 sont valables après un retard δt, la sortie S_1 et la retenue R_1 ne sont correctes qu'après un retard 2 δt, et ainsi de suite. La figure 9 présente un exemple de réalisation logique d'un additionneur de deux mots de 2 bits.

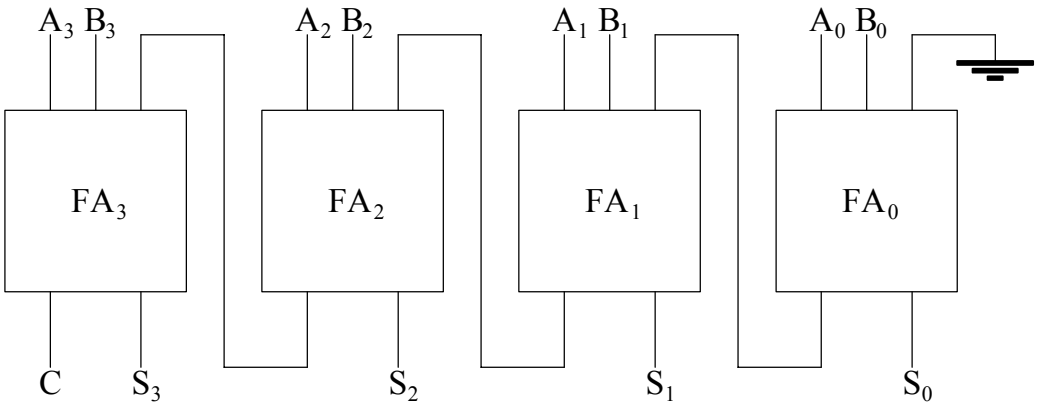


Figure 8

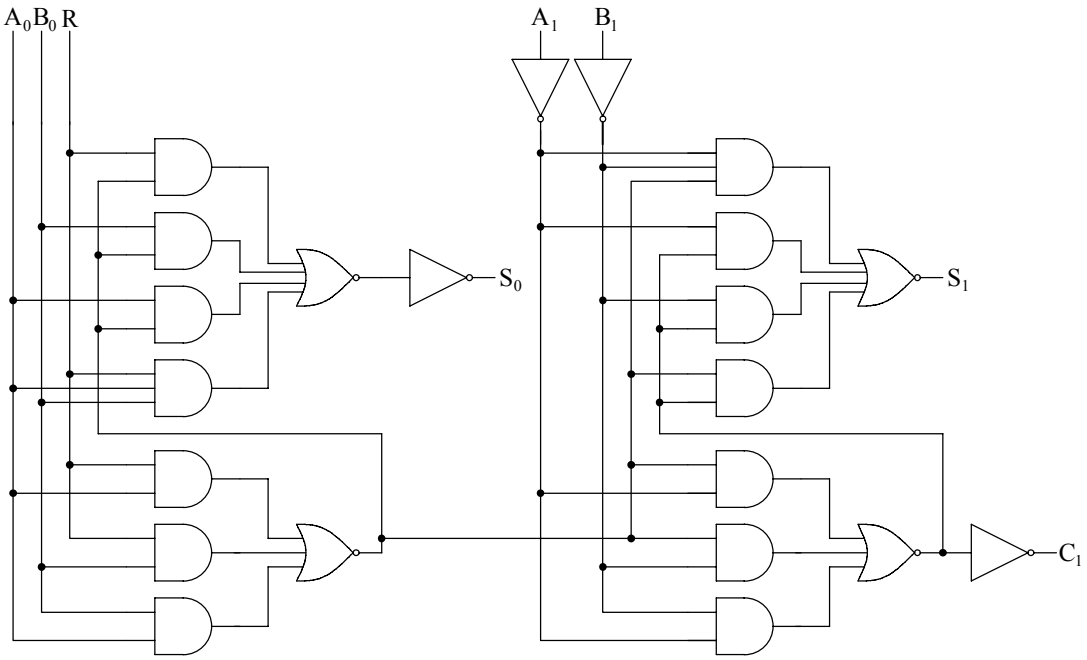


Figure 9

III.1.d Addition séquentielle

Dans un additionneur séquentiel chacun des nombres A et B est représenté par un train d'impulsions (figure 10) synchrones par rapport à un signal d'horloge. L'ordre chronologique d'arrivée des impulsions correspond à l'ordre croissant des poids : le bit le moins significatif se présentant le premier. Ces impulsions sont injectées sur les deux lignes d'entrée d'un additionneur (figure 11). A chaque cycle d'horloge, la retenue provenant des bits de poids inférieurs doit être mémorisée (par exemple, à l'aide d'une bascule D qui sera étudiée dans le chapitre suivant).

Un additionneur parallèle est plus rapide mais nécessite plus de composants.

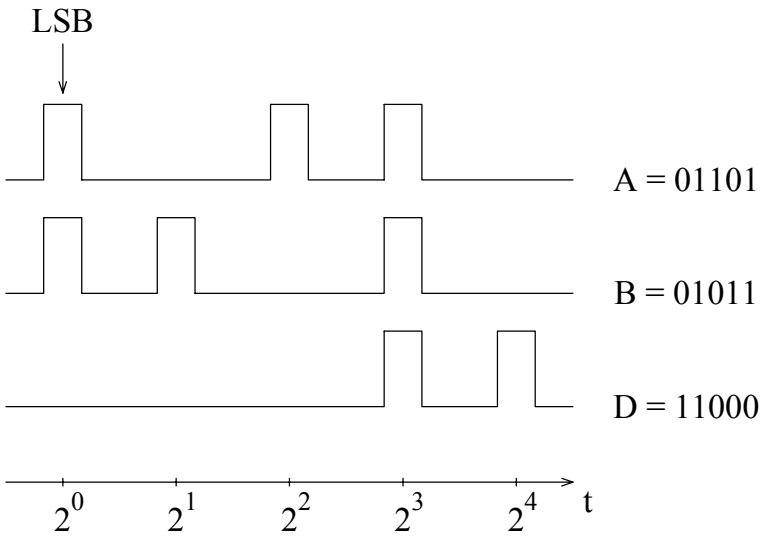


Figure 10

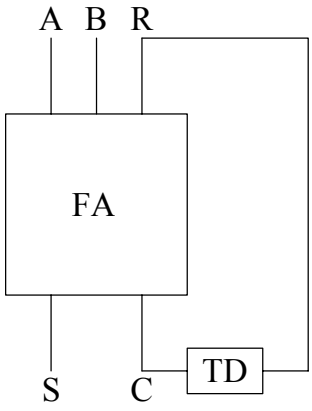


Figure 11

III.2 Soustraction

III.2.a Demi-soustracteur

La table de vérité pour un demi-soustracteur (ne tenant pas compte d'une éventuelle retenue provenant des bits de poids inférieurs) est la suivante :

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

Table 3

Où D représente le résultat de la soustraction A – B et C la retenue. Nous en déduisons les expressions logiques définissant D et C :

$$\begin{cases} D = \overline{A}\,B + A\,\overline{B} = A \oplus B \\ C = \overline{A}\,B \end{cases}$$

et le schéma correspondant :

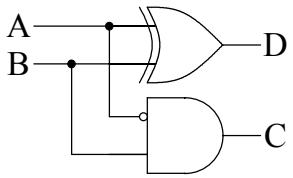


Figure 12

Nous pourrions maintenant étudier un soustracteur prenant en compte la retenue. Nous allons plutôt tirer parti de certaines propriétés de la numération binaire pour traiter de la même manière l'addition et la soustraction.

III.2.b Additionneur-soustracteur

Nous savons qu'avec un mot de n bits nous pouvons représenter un entier positif dont la valeur est comprise entre 0 et $2^n - 1$. Le complémentaire d'un mot de n bits est obtenu entre prenant le complément de chacun des n bits. Ainsi, si nous sommons un nombre et son complément nous obtenons un mot dont tous les bits sont à 1. C'est-à-dire :

$$A + \overline{A} = 2^n - 1$$

Attention : dans ce paragraphe le signe + représente l'opération addition et non la fonction logique OU. Nous pouvons encore écrire :

$$-A = \overline{A} + 1 - 2^n$$

Mais sur n bits l'entier 2^n est identique à 0 :

$$2^n \equiv 0 \quad (\text{n bits})$$

C'est-à-dire qu'il est possible d'écrire un nombre entier négatif comme le "complément à 2" de sa valeur absolue :

$$-A = \overline{A} + 1$$

Nous reviendrons sur les divers codages des entiers signés plus tard. Nous pouvons utiliser cette propriété pour écrire la soustraction de deux mots de n bits sous la forme suivante :

$$A - B = A + \overline{B} + 1 - 2^n \equiv A + \overline{B} + 1 \quad (\text{n bits})$$

Ce résultat conduit au schéma de principe présenté sur la figure 13 combinant les fonctions addition et soustraction. Celui-ci est basé sur l'emploi d'un additionneur n bits et d'un multiplexeur à deux lignes d'entrée. Nous étudierons ce type de circuit un peu plus loin dans ce chapitre. Selon le code opération O (0 pour une addition et 1 pour une soustraction) ce multiplexeur permet de sélectionner une des deux entrées, B ou son complémentaire. Le code opération est également injecté sur l'entrée report de retenue de l'additionneur. Pour simplifier le schéma et éviter de représenter n lignes de connexion parallèles, on ne matérialise qu'une seule ligne. Celle-ci est barrée et accompagnée d'une valeur qui indique le nombre réel de connexions.

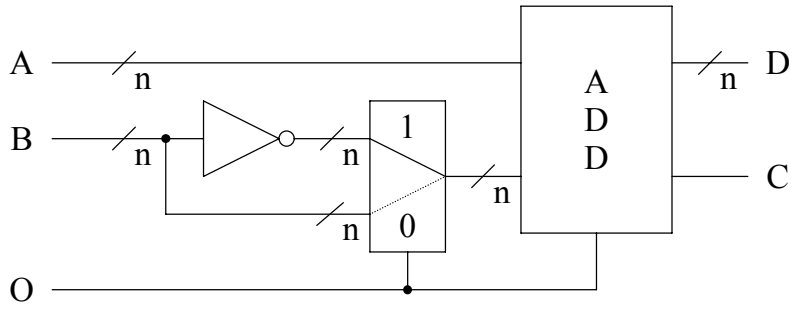


Figure 13

III.3 Comparaison

On rencontre très souvent la nécessité de comparer deux entiers (A = B, A > B ou A < B). Ecrivons la table de vérité correspondant à ces trois fonctions de comparaison de 2 bits. La fonction C doit être égale à 1 si et seulement si A > B, la fonction D si et seulement si A < B et la fonction E si et seulement si A = B. Ce qui nous donne :

A	B	C (A > B)	D (A < B)	E (A = B)
0	0	0	0	1
0	1	0	1	0
1	0	1	0	0
1	1	0	0	1

Table 5

Nous en déduisons les expressions logiques de C, D et E :

$$\begin{cases} C = A\,\overline{B} \\ D = \overline{A}\,B \\ E = \overline{A \oplus B} = \overline{A\,\overline{B} + \overline{A}\,B} = \overline{C + D} \end{cases}$$

La figure 14 présente le diagramme d'un bloc logique comparant deux bits A et B.

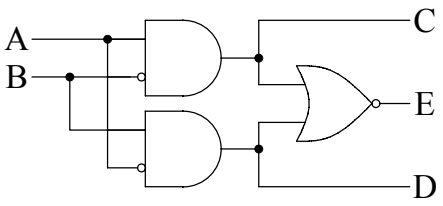


Figure 14

III.4 Contrôle de parité

La parité d'un mot binaire est définie comme la parité de la somme des bits, soit encore :

- parité paire (ou 0) : nombre pair de 1 dans le mot;
- parité impaire (ou 1) : nombre impair de 1 dans le mot.

La fonction OU-exclusif donne la parité d'un sous-ensemble de deux bits. Le contrôle de parité est basé sur la constatation que le mot de n+1 bits formé en adjoignant à un mot de n bits son bit de parité est toujours de parité 0. La figure 15 représente le diagramme logique d'un générateur-

contrôleur de parité pour 4 bits. Si l'entrée P' est imposée à 0 ce circuit fonctionne comme générateur de parité : la sortie P représente la parité du mot composé par les bits A, B, C et D.

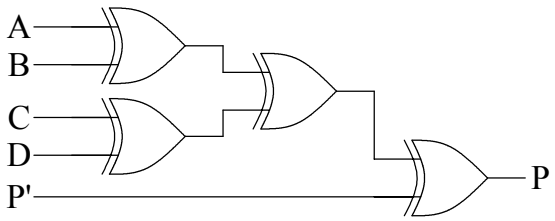


Figure 15

Le contrôle de la parité est utilisé, par exemple, pour augmenter la fiabilité d'un système de transmission ou de stockage de données. La figure 16 montre l'utilisation du circuit précédent en générateur de parité du côté de l'émission et contrôleur de parité du côté de la réception. La sortie P_2 doit être à 0 pour chaque mot transmis, sinon cela indique un problème de transmission.

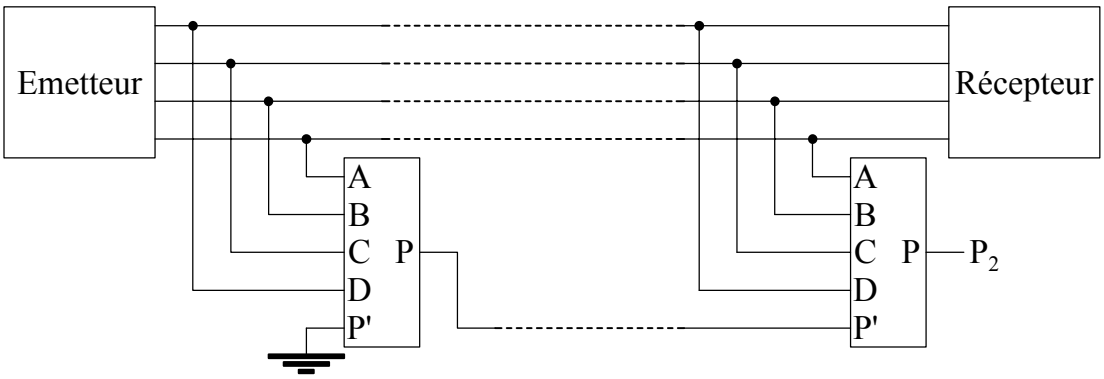


Figure 16

Remarquons cependant que la parité ne permet de détecter qu'un nombre impair de bits en erreur dans un mot. Par ailleurs il ne permet pas corriger les erreurs détectées. Pour ce faire il faut utiliser des codes correcteurs d'erreur qui nécessitent plusieurs bits supplémentaires.

III.5 Décodage

Dans un système numérique les instructions, tout comme les nombres, sont transportées sous forme de mots binaires. Par exemple un mot de 4 bits peut permettre d'identifier 16 instructions différentes : l'information est codée. Très souvent l'équivalent d'un commutateur à 16 positions permet de sélectionner l'instruction correspondant à un code. Ce processus est appelé décodage. La fonction de décodage consiste à faire correspondre à un code présent en entrée sur n lignes une seule sortie active parmi les $N = 2^n$ sorties possibles. A titre d'exemple, nous allons étudier le décodage de la représentation DCB des nombres.

III.5.a Représentation DCB (Décimale Codée Binaire)

Le code DCB (ou en anglais BCD : Binary Coded Decimal) transforme les nombres décimaux en remplaçant chacun des chiffres décimaux par 4 chiffres binaires. Cette représentation conserve donc la structure décimale : unités, dizaines, centaines, milliers, etc… Chaque chiffre est codé sur 4 bits selon le code de la table 6 :

Décimal	DCB
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Table 6

Par exemple le nombre décimal 294 sera codé en DCB : 0010 1001 0100. Ce type de codage permet, par exemple, de faciliter l'affichage en décimal du contenu d'un compteur. Pour ce faire on peut utiliser des tubes de Nixie, contenant 10 cathodes ayant chacune la forme d'un chiffre (fig. 17) ou des afficheurs lumineux à sept segments (fig. 18).

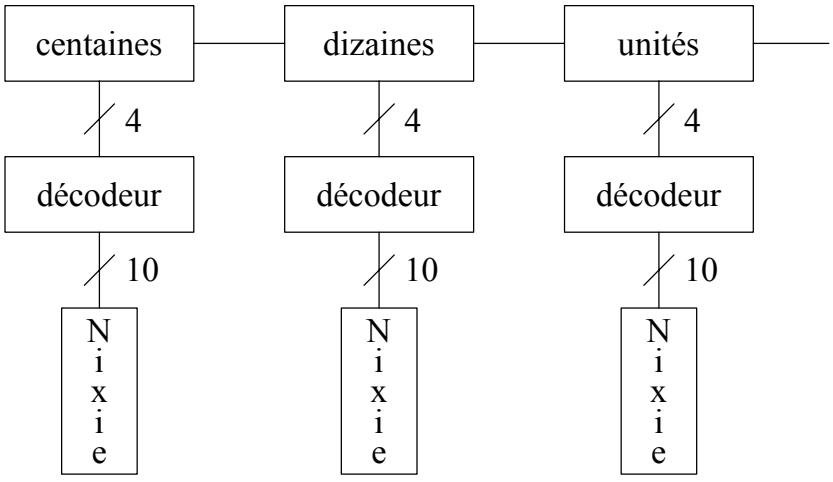


Figure 17

La fonction de chaque décodeur est d'activer une des dix lignes en sortie et une seule en fonction du code présent sur les quatre entrées. Par exemple, si ce code est égal à 5, la 6ème ligne de sortie est mise dans l'état 1 et le chiffre 5 est affiché par le tube de Nixie.

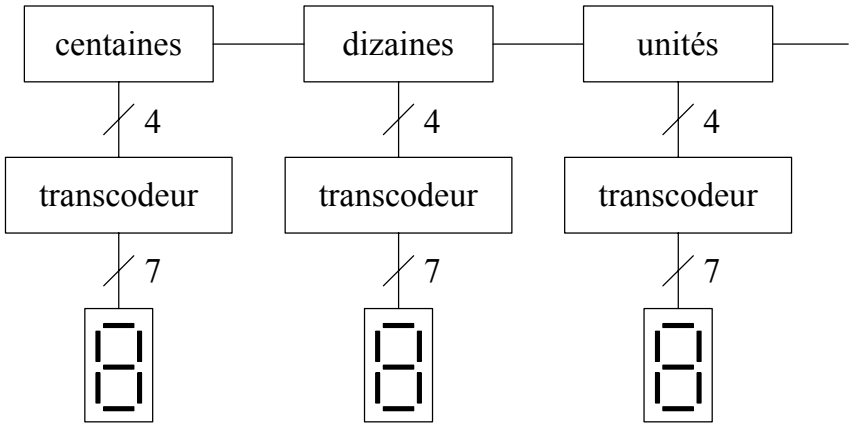


Figure 18

La fonction de chacun des transcodeurs est de positionner à 1 les lignes de sortie correspondant aux segments à allumer selon de code porté par les quatre lignes d'entrée. De manière générale, un transcodeur fait correspondre à un code A en entrée sur n lignes, un code B en sortie sur m lignes.

III.5.b Décodeur DCB-décimal

Nous allons étudier l'exemple d'un décodeur DCB-décimal. La table de vérité de ce décodeur est très simple :

D	C	B	A	L_0	L_1	L_2	L_3	L_4	L_5	L_6	L_7	L_8	L_9
0	0	0	0	1									
0	0	0	1		1								
0	0	1	0			1							
0	0	1	1				1						
0	1	0	0					1					
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

Table 7

A chacune des lignes de sortie nous pouvons associer un produit prenant en compte chacune des quatre entrées ou leur complément. Ainsi la ligne 5 correspond à :

$$A \overline{B} C \overline{D}$$

D'autre part, on souhaite souvent n'activer les lignes de sortie qu'en présence d'un signal de commande global (strobe ou enable). Ce signal S est mis en coïncidence sur chacune des dix portes de sortie. Dans l'exemple suivant, si S est dans l'état 0 le décodeur est bloqué et tous les sorties sont également dans l'état 0.

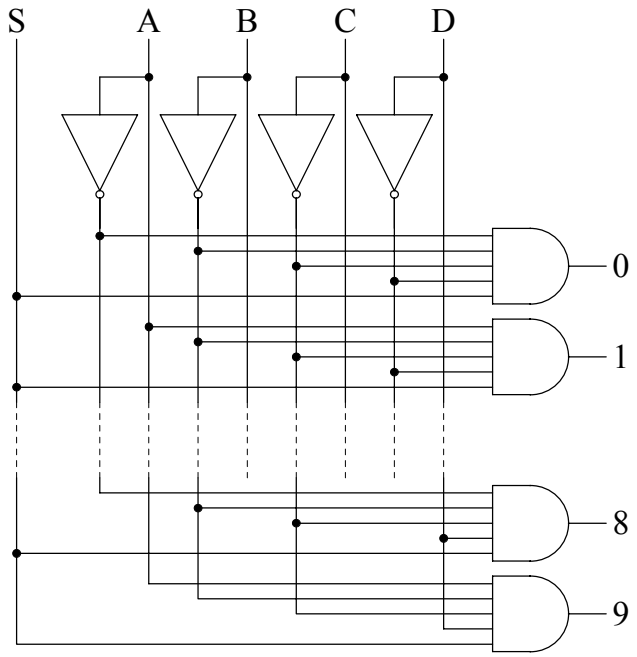


Figure 19

III.6 Multiplexage

Le multiplexage est un dispositif qui permet de transmettre sur une seule ligne des informations en provenance de plusieurs sources ou à destination de plusieurs cibles. La figure 20 en présente une analogie mécanique avec deux commutateurs à plusieurs positions. Choisir une ligne revient à définir l'angle du levier ou une adresse.

Figure 20

III.6.a Démultiplexeur

Un démultiplexeur est un circuit comptant une entrée et N sorties et qui met en relation cette entrée avec une sortie et une seule. Pour pouvoir sélectionner cette sortie il faut également des lignes d'adressage : le code porté par ces lignes identifie la ligne de sortie à utiliser. Ce circuit est très proche d'un décodeur. Considérons un démultiplexeur avec quatre lignes de sortie. Il faut deux lignes d'adresse. Supposons que nous souhaitons également valider les données avec un signal de contrôle E (pour laisser par exemple le temps aux niveaux d'entrée de se stabiliser). Par convention nous choisissons de prendre en compte les données pour E = 0.

E	B	A	Y_0	Y_1	Y_2	Y_3	Produit
0	0	0	D	0	0	0	$\overline{A}\,\overline{B}\,\overline{E}\,D$
0	0	1	0	D	0	0	$A\,\overline{B}\,\overline{E}\,D$
0	1	0	0	0	D	0	$\overline{A}\,B\,\overline{E}\,D$
0	1	1	0	0	0	D	$A\,B\,\overline{E}\,D$
1	0	0	0	0	0	0	
1	0	1	0	0	0	0	
1	1	0	0	0	0	0	
1	1	1	0	0	0	0	

Table 8

De cette table nous déduisons le logigramme suivant :

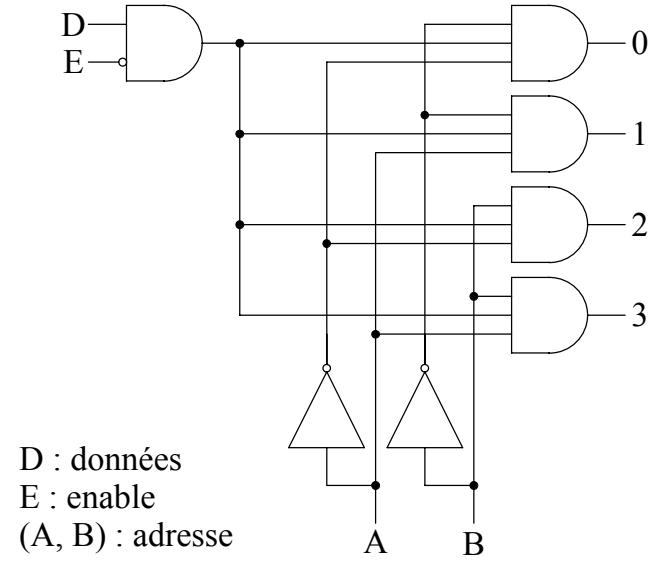


Figure 21

Il existe sous forme de circuits intégrés des démultiplexeurs avec 2, 4 ou 16 lignes de sortie. Pour constituer des démultiplexeurs d'ordre supérieur on peut être amené à cascader des démultiplexeurs. Par exemple un démultiplexeur avec 32 sorties peut être réalisé avec un "tronc" de 4 sorties et 4 "branches" de 8 sorties :

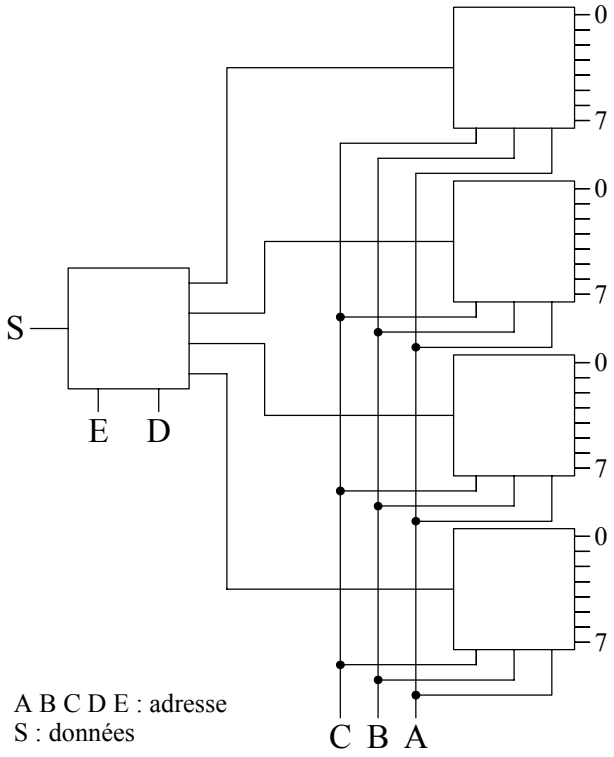


Figure 22

III.6.b Multiplexeur

Un multiplexeur, réalise l'opération inverse. Il sélectionne une entrée parmi N et transmet l'information portée par cette ligne à un seul canal de sortie. Considérons un multiplexeur à quatre entrées, donc deux lignes d'adressage, et une ligne de validation. La table de vérité de ce circuit est donnée par la table 9. De cette table nous déduisons une expression logique pour la sortie :

$$Y = \overline{A}\,\overline{B}\,\overline{E}\,X_0 + A\,\overline{B}\,\overline{E}\,X_1 + \overline{A}\,B\,\overline{E}\,X_2 + A\,B\,\overline{E}\,X_3$$

Cette expression correspond au schéma présenté sur la figure 23.

E	B	A	Y
0	0	0	X_0
0	0	1	X_1
0	1	0	X_2
0	1	1	X_3
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

Table 9

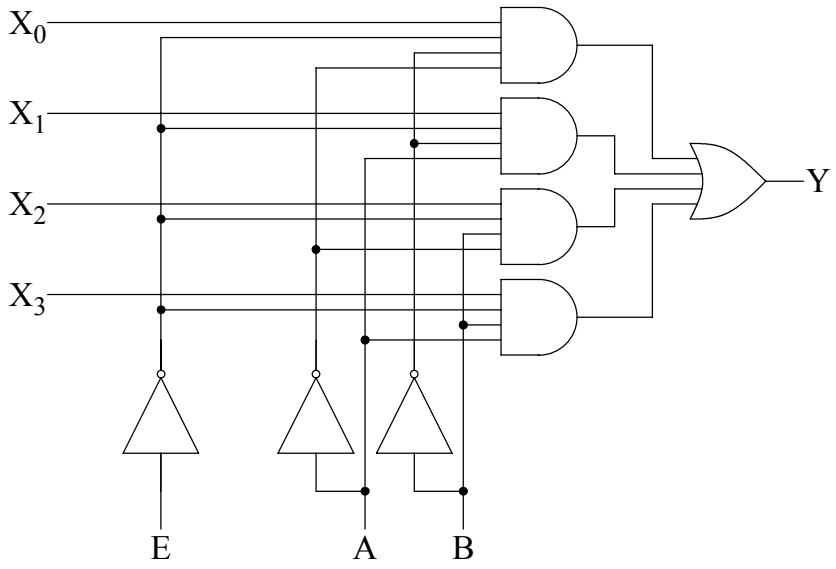


Figure 23

Tout comme pour les démultiplexeurs on peut cascader plusieurs multiplexeurs pour obtenir un multiplexeur d'ordre supérieur. La figure 24 montre comment un multiplexeur à 32 entrées peut être réalisé à partir de quatre multiplexeurs à 8 entrées et d'un multiplexeur à 4 entrées.

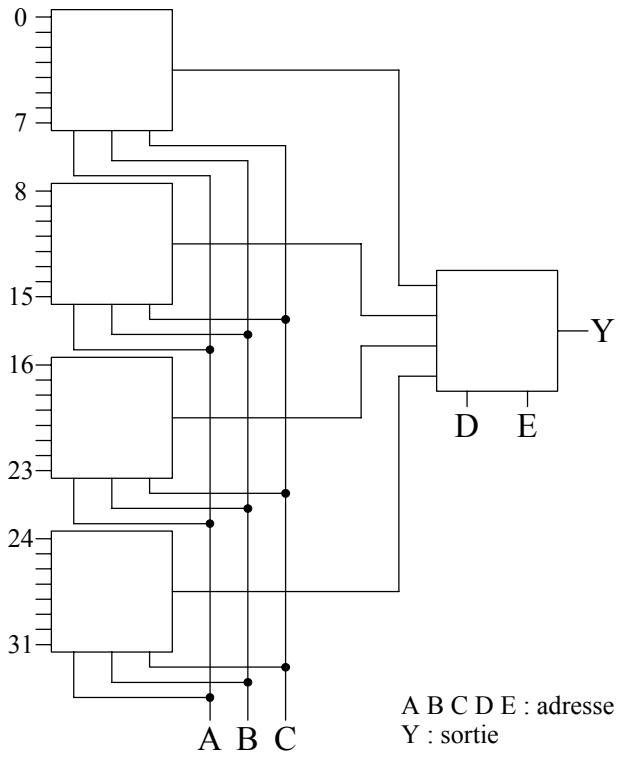


Figure 24

III.6.c Conversion parallèle-série

Considérons un mot de n bits (par exemple 4) présent en parallèle sur les entrées d'un multiplexeur :

- $X_0 \equiv$ bit correspondant à 2^0;
- $X_1 \equiv$ bit correspondant à 2^1;
- $X_2 \equiv$ bit correspondant à 2^2;
- $X_3 \equiv$ bit correspondant à 2^3.

Supposons que les lignes d'adresse A et B soient connectées aux sorties d'un compteur de période T, nous aurons en fonction du temps :

t	B	A	Y
[0,T]	0	0	X_0
[T,2T]	0	1	X_1
[2T,3T]	1	0	X_2
[3T,4T]	1	1	X_3
[4T,5T]	0	0	X_0

Table 10

Les bits X_0, X_1, X_2 et X_3 se retrouvent en série dans le temps sur la sortie Y du multiplexeur.

III.6.d Réalisation d'une fonction logique

Un multiplexeur peut être utilisé pour réaliser une fonction logique. Il permet en effet une transcription directe de la table de vérité. Considérons par exemple la fonction de quatre variables logiques F(x, y, z, t) définie par la table suivante :

x	y	z	t	F	Entrée du multiplexeur
0	0	0	0	1	X_0
0	0	0	1	0	X_0
0	0	1	0	1	X_2
0	0	1	1	0	X_3
0	1	0	0	0	X_4
0	1	0	1	1	X_5
0	1	1	0	0	X_6
0	1	1	1	0	X_7
1	0	0	0	1	X_8
1	0	0	1	1	X_9
1	0	1	0	1	X_{10}
1	0	1	1	1	X_{11}
1	1	0	0	0	X_{12}
1	1	0	1	1	X_{13}
1	1	1	0	0	X_{14}
1	1	1	1	0	X_{15}

Table 11

Il est possible d'utiliser un multiplexeur à 16 entrées. Il suffit de connecter les variables logiques x, y, z et t sur les entrées d'adresse et de mettre chacune des entrées X_k à 0 ou 1 selon la table de vérité.

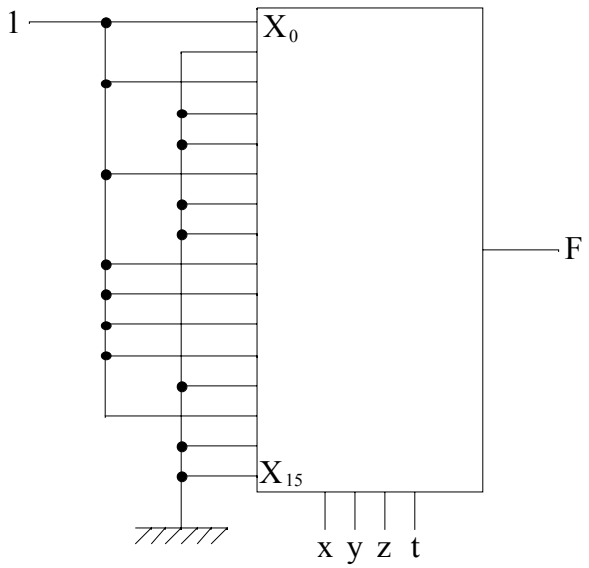


Figure 25

III.7 Encodage

Nous venons d'étudier le principe du décodage, passons à l'opération inverse ou encodage. Un encodeur est un système qui comporte N lignes d'entrée et n lignes de sortie. Lorsqu'une des lignes d'entrée est activée l'encodeur fournit en sortie un mot de n bits correspondant au codage de l'information identifiée par la ligne activée.

Considérons un encodeur transformant un nombre décimal en son équivalent en code DCB. Il comportera donc 10 entrées (0 à 9) et 4 sorties. Nous pouvons par exemple imaginer que chacune des dix lignes d'entrée peut être reliée à une touche d'un clavier. La table 12 correspond à la table de vérité de cet encodeur. A partir de cette table nous pouvons écrire les expressions logiques définissant les sorties à partir des entrées.

W_0	W_1	W_2	W_3	W_4	W_5	W_6	W_7	W_8	W_9	Y_3	Y_2	Y_1	Y_0
1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	0	0	0	1	0	0
0	0	0	0	0	1	0	0	0	0	0	1	0	1
0	0	0	0	0	0	1	0	0	0	0	1	1	0
0	0	0	0	0	0	0	1	0	0	0	1	1	1
0	0	0	0	0	0	0	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	0	1	1	0	0	1

Table 12

$$\begin{cases} Y_0 = W_1 + W_3 + W_5 + W_7 + W_9 \\ Y_1 = W_2 + W_3 + W_6 + W_7 \\ Y_2 = W_4 + W_5 + W_6 + W_7 \\ Y_3 = W_8 + W_9 \end{cases}$$

En effet Y_0 est égal à 1 quand la ligne W_1 est dans l'état 1, ou la ligne W_3, ou la ligne W_5, ou la ligne W_7, ou la ligne W_9. La ligne Y_0 est nulle dans tous les autres cas. Il est possible de réaliser ces fonctions OU avec des diodes selon le montage de la figure suivante :

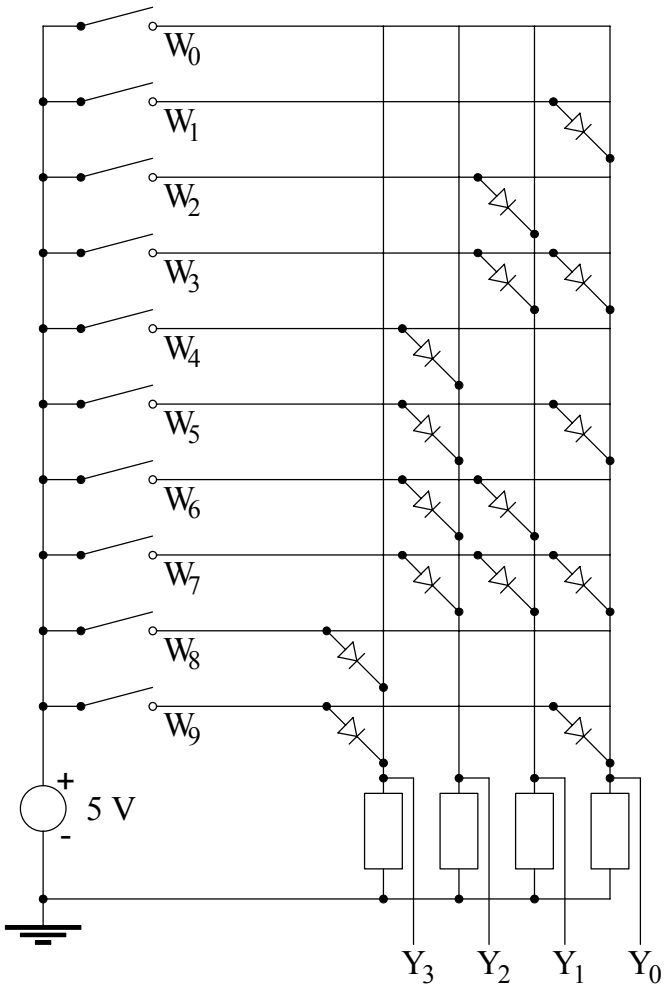
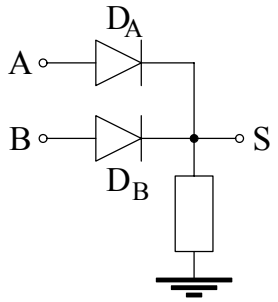


Figure 26

En effet considérons le circuit de la figure 27 :



A	B		S
0	0	D_A et D_B bloquées	0
+V	0	D_A passante/D_B bloquée	+V
0	+V	D_A bloquée/D_B passante	+V
+V	+V	D_A et D_B passantes	+V

Figure 27

Si nous traduisons la signification logique des niveaux haut et bas en logique positive, au circuit de la figure 27 correspond la table de vérité 13. La fonction réalisée est donc un OU inclusif.

A	B	S
0	0	0
0	1	1
1	0	1
1	1	1

Table 13

La figure 26 représente un exemple de réalisation d'un encodeur DCB réalisé avec des diodes. Le bon fonctionnement de ce codeur suppose qu'une seule ligne d'entrée peut être dans l'état 1.

Par contre, si plusieurs entrées sont actives simultanément le résultat pourra ne pas avoir de signification. Par exemple, si les deux lignes W_7 et W_8 sont dans l'état 1 (frappe simultanée des deux touches), il en sera de même pour les quatre sorties. Pour éviter ce problème on utilise un encodeur prioritaire. Pour ce type de circuit si plusieurs lignes d'entrée sont actives simultanément le résultat correspondant à une seule parmi celles-ci est affiché en sortie. La règle peut être, par exemple, de mettre en sortie le code correspondant à la ligne d'entrée d'indice le plus élevé. Par exemple, si W_7 et W_8 sont dans l'état 1 l'encodeur prioritaire donne en sortie le code correspondant à W_8. La table de vérité correspondant à ce choix est donnée par la table 14. Chaque croix indique que le code en sortie doit être indépendant de l'état de l'entrée concernée.

W_0	W_1	W_2	W_3	W_4	W_5	W_6	W_7	W_8	W_9	Y_3	Y_2	Y_1	Y_0
1	0	0	0	0	0	0	0	0	0	0	0	0	0
X	1	0	0	0	0	0	0	0	0	0	0	0	1
X	X	1	0	0	0	0	0	0	0	0	0	1	0
X	X	X	1	0	0	0	0	0	0	0	0	1	1
X	X	X	X	1	0	0	0	0	0	0	1	0	0
X	X	X	X	X	1	0	0	0	0	0	1	0	1
X	X	X	X	X	X	1	0	0	0	0	1	1	0
X	X	X	X	X	X	X	1	0	0	0	1	1	1
X	X	X	X	X	X	X	X	1	0	1	0	0	0
X	X	X	X	X	X	X	X	X	1	1	0	0	1

Table 14

Alors que les expressions logiques définissant les lignes de sortie Y_i ne dépendaient que des 1 dans la table 11, il faut ici tenir compte des 0. Par exemple pour Y_0 nous avons :

$$Y_0 = W_1\,\overline{W}_2\,\overline{W}_3\,\overline{W}_4\,\overline{W}_5\,\overline{W}_6\,\overline{W}_7\,\overline{W}_8\,\overline{W}_9 + W_3\,\overline{W}_4\,\overline{W}_5\,\overline{W}_6\,\overline{W}_7\,\overline{W}_8\,\overline{W}_9 + W_5\,\overline{W}_6\,\overline{W}_7\,\overline{W}_8\,\overline{W}_9 + W_7\,\overline{W}_8\,\overline{W}_9 + W_9$$

Nous pouvons mettre le complémentaire de W_9 en facteur dans les quatre premiers termes, puis en utilisant l'identité :

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

il vient, après factorisation du complément de W_8 :

$$Y_0 = \overline{W}_8\,(W_1\,\overline{W}_2\,\overline{W}_3\,\overline{W}_4\,\overline{W}_5\,\overline{W}_6\,\overline{W}_7 + W_3\,\overline{W}_4\,\overline{W}_5\,\overline{W}_6\,\overline{W}_7 + W_5\,\overline{W}_6\,\overline{W}_7 + W_7) + W_9$$

Soit encore :

$$Y_0 = \overline{W}_8\,(W_1\,\overline{W}_2\,\overline{W}_3\,\overline{W}_4\,\overline{W}_5\,\overline{W}_6 + W_3\,\overline{W}_4\,\overline{W}_5\,\overline{W}_6 + W_5\,\overline{W}_6 + W_7) + W_9$$

$$Y_0 = \overline{W}_8\,((W_1\,\overline{W}_2\,\overline{W}_3\,\overline{W}_4\,\overline{W}_5 + W_3\,\overline{W}_4\,\overline{W}_5 + W_5)\,\overline{W}_6 + W_7) + W_9$$

$$Y_0 = \overline{W}_8\,((W_1\,\overline{W}_2\,\overline{W}_3\,\overline{W}_4 + W_3\,\overline{W}_4 + W_5)\,\overline{W}_6 + W_7) + W_9$$

$$Y_0 = \overline{W}_8\,(((W_1\,\overline{W}_2\,\overline{W}_3 + W_3)\,\overline{W}_4 + W_5)\,\overline{W}_6 + W_7) + W_9$$

$$Y_0 = \overline{W}_8\,(((W_1\,\overline{W}_2 + W_3)\,\overline{W}_4 + W_5)\,\overline{W}_6 + W_7) + W_9$$

Soit en réorganisant l'ordre des termes :

$$Y_0 = W_9 + \overline{W}_8\ (W_7 + \overline{W}_6\ (W_5 + \overline{W}_4\ (W_3 + W_1\ \overline{W}_2)))$$

Pour la ligne Y_1 nous avons :

$$\begin{aligned} Y_1 = {} & W_2\ \overline{W}_3\ \overline{W}_4\ \overline{W}_5\ \overline{W}_6\ \overline{W}_7\ \overline{W}_8\ \overline{W}_9 + W_3\ \overline{W}_4\ \overline{W}_5\ \overline{W}_6\ \overline{W}_7\ \overline{W}_8\ \overline{W}_9 \\ & + W_6\ \overline{W}_7\ \overline{W}_8\ \overline{W}_9 + W_7\ \overline{W}_8\ \overline{W}_9 \end{aligned}$$

Soit en factorisant :

$$Y_1 = \overline{W}_8\ \overline{W}_9\ (W_7 + \overline{W}_7\ (W_6 + \overline{W}_6\ (W_3 + W_2\ \overline{W}_3)\ \overline{W}_4\ \overline{W}_5))$$

En utilisant toujours la même identité nous pouvons simplifier cette expression, il vient en réordonnant les termes :

$$Y_1 = \overline{W}_9\ \overline{W}_8\ (W_7 + W_6 + \overline{W}_5\ \overline{W}_4\ (W_3 + W_2))$$

Pour Y_2 nous devons écrire :

$$Y_2 = W_4\ \overline{W}_5\ \overline{W}_6\ \overline{W}_7\ \overline{W}_8\ \overline{W}_9 + W_5\ \overline{W}_6\ \overline{W}_7\ \overline{W}_8\ \overline{W}_9 + W_6\ \overline{W}_7\ \overline{W}_8\ \overline{W}_9 + W_7\ \overline{W}_8\ \overline{W}_9$$

Soit encore :

$$Y_2 = \overline{W}_9\ \overline{W}_8\ (W_7 + \overline{W}_7\ (W_6 + \overline{W}_6\ (W_5 + \overline{W}_5\ W_4)))$$

En utilisant toujours la même identité il vient :

$$Y_2 = \overline{W}_9\ \overline{W}_8\ (W_7 + W_6 + W_5 + W_4)$$

Enfin pour Y_3 nous avons :

$$Y_3 = W_8\ \overline{W}_9 + W_9 = W_9 + W_8$$