Chapitre

12 Systèmes à événements discrets

On définit les systèmes à temps discret souvent en opposition par rapport aux systèmes à temps continus.

Dans les systèmes à temps continus (les systèmes linéaires et asservis), les variables évoluent continûment et peuvent, en général, être décrites par un ensemble d'équations différentielles. Nous avons vu dans le chapitre sur les systèmes linéaires quelques outils mathématiques permettant d'étudier ces systèmes.

Les systèmes à événements discrets permettent eux de décrire le comportement de systèmes qui évoluent lorsqu'un événement est présent.

Considérons un ascenseur, le déplacement de l'ascenseur entre deux étages peut être décrit par un ensemble d'équations différentielles (P.F.D. pour obtenir l'équation différentielle du mouvement, les équations électriques du moteur...) ; à l'aide de ces équations on peut par exemple réaliser un asservissement de vitesse de la cabine. Par contre la gestion des appels à chaque étage, la demande de déplacement d'un passager (les multiples demandes), l'arrivée à un étage, le départ d'un étage,...correspond elle, à un traitement d'événements discrets (appui sur un bouton d'étage, sélection d'un étage, ouverture/fermeture de la porte, ...).

12.1 Typologie des systèmes

12.1.1 Les systèmes continus

Le modèle d'un système continu ne comprend que des variables continues, de plus, le temps est également une variable continue (figure 12.2). Le temps et les variables prennent leurs valeurs dans $\mathbb{R}$. C'est le grand domaine des systèmes représentés par des équations différentielles. Les systèmes linéaires continus invariants que nous avons étudié au premier semestre sont une partie des systèmes continus.

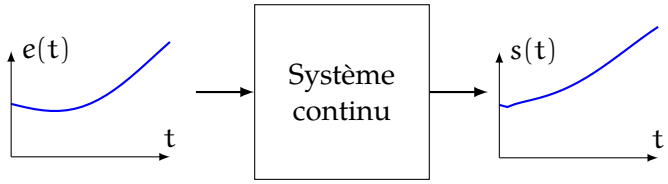


FIGURE 12.1 – Système continu - Information continue

— Les variables continues sont des variables qui prennent leurs valeurs sur le domaine des réels $\mathbb{R}$.

12.1.2 Les systèmes numériques échantillonnés

Dans les systèmes numériques échantillonnés, le temps n'est plus une variable continue mais une variable discrète. Le temps est en effet représenté comme une suite infinie d'instants repérés par des entiers naturels $\mathbb{N}$.

Les systèmes informatiques ne peuvent traiter que des systèmes échantillonnés en effet :

— le temps de calcul ne pouvant être nul, le système informatique ne peut prélever l'information qu'à des instants particuliers, cela nécessite un échantillonnage du temps ;

— une conversion analogique/numérique pour ramener les variables dans un domaine (codage sur 8, 16, 32,...bits) que peut traiter le système numérique ;

— la fréquence d'échantillonnage doit respecter le critère de Shannon sans être trop importante pour ne pas saturer les capacités du système de traitement.

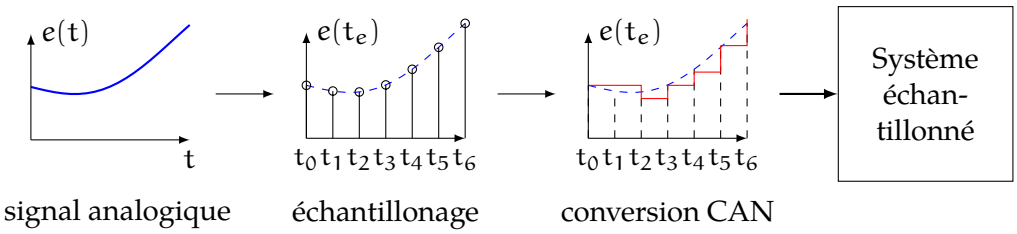


FIGURE 12.2 – Système échantillonné

Les systèmes échantillonnés sont la représentation des systèmes analogiques dans l'espace numérique.

12.1.3 Les systèmes discrets

Un système discret est un système dans lequel, les variables sont toutes discrétisées, soit par nature (lampe éclairée / éteinte), soit parce que l'on ne prend en compte que l'état de la variable (four en marche ou température atteinte, sans se préoccuper de l'évolution de la température).

Dans les systèmes discrets, le temps est en général continu ou discrétisé.

Les systèmes discrets ne manipulent que des variables logiques.

12.1.4 Variable logique

Une variable logique est une variable, à laquelle on associe deux états. On peut ainsi associer à un grand nombre de phénomènes physiques un état logique (porte ouverte/fermée, vrai/faux, voyant éclairé/éteint, 0/1, 0 V/5 V,...).

Un signal réel est une grandeur physique en général continue qui peut donc prendre une infinité de valeurs. Pour associer à ce signal, un signal binaire (0,1), il est nécessaire de fixer des seuils. Le passage d'un seuil caractérise le passage de l'état 1 à 0 et réciproquement - figure 12.3).

Le seuil pour passer de l'état bas (0) à l'état haut (1) peut être identique ou différent de celui pour passer de l'état haut à l'état bas.

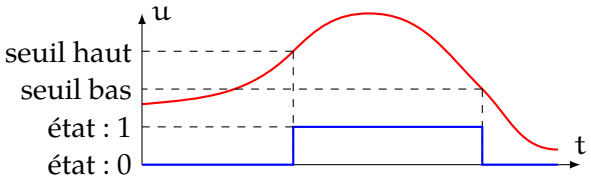


FIGURE 12.3 – Variable et état logique

a) Convention

Il est nécessaire de distinguer :

— la grandeur physique représentée,

— le support de l'information logique,

— le nom attribué,

— l'état logique qu'elle peut prendre.

Par convention on établit une correspondance entre l'état physique de la grandeur et sa valeur logique correspondante (logique positive ou logique négative).

L'état logique peut être noté « 0 » ou « 1 » ou encore « L » (Low) ou « H » (High). On parle aussi de niveau logique haut ou niveau logique bas, état haut ou état bas.

On parle de logique positive lorsqu'on associe à l'état logique 1 l'état actif de la variable et de logique négative dans le cas contraire (figure 12.4).

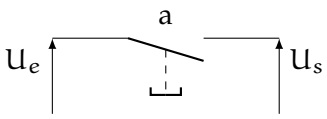
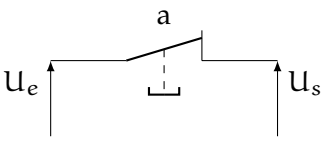
	Contact établissement de circuit		Contact coupure de circuit	
état physique	Actionné	Non actionné	Actionné	Non actionné
état électrique	Passant	Non passant	Non passant	Passant
état logique	1	0	1	0
	interrupteur  ne conduit pas au repos, le courant passe s'il est actionné		poussoir de réfrigérateur  conduit au repos, le courant passe que s'il n'est pas actionné	

Figure 12.4 – Modélisation d'un contact électrique

12.1.5 Les systèmes à événements discrets - S.E.D.

Pour les systèmes à événements discrets, les variables sont discrètes et le temps n'est connu que lors des changements d'états de ces variables.

L'évolution temporelle n'est prise en compte que lors des changement d'état, on ne se préoccupe pas de l'écoulement du temps.

Ainsi, si on s'intéresse à une barrière de péage, les événements qui font évoluer le système sont : la présence d'un véhicule, la validation du paiement, le cycle d'ouverture, le départ du véhicule, le cycle de fermeture, et cela quel que soit le temps qui s'écoule entre deux véhicules ou le temps mis par l'automobiliste à payer.

12.2 Typologie des S.E.D.

12.2.1 S.E.D. combinatoires

Un système est dit combinatoire, lorsque la ou les sorties ne dépendent que de la combinaison des entrées. La même cause (même combinaison des entrées) produit toujours le même effet (même état des sorties).

L'effet disparaît lorsque la cause disparaît.

Chaque sortie est une fonction des entrées :

$$s_j = f(e_1, e_2, \ldots, e_i, \ldots, e_n)$$

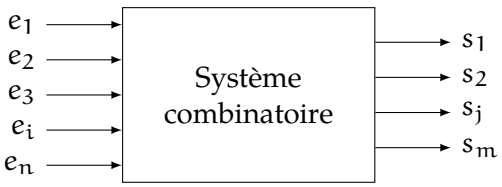


Figure 12.5 – Système combinatoire

12.2.2 S.E.D. séquentiels

Un système est dit séquentiel, lorsque la ou les sorties dépendent de la combinaison des entrées et de l'état précédent des sorties. Une même cause (même combinaison des entrées) peut produire des effets différents.

Le temps peut aussi être une des causes d'évolution des sorties. L'effet peut persister après la disparition de la combinaison l'ayant provoqué.

Chaque sortie, à chaque instant, est le résultat d'une combinaison des entrées et de l'état précédent des sorties ou des variables internes.

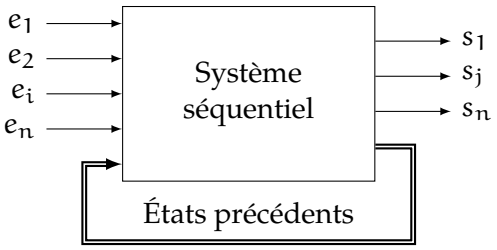


FIGURE 12.6 – Système séquentiel

$$s_j = f(e_1, e_2, \ldots, e_i, \ldots, e_n, E_1, E_i)$$

12.3 Systèmes combinatoires

12.3.1 Algèbre de Boole

L'algèbre de Boole ou algèbre logique est l'algèbre définie pour des variables ne pouvant prendre que deux états. On appelle variable booléenne, une variable ne prenant que deux états, pour une variable a, les deux états sont a et NON a (noté $\overline{a}$).

a) Opérateurs logiques fondamentaux

On distingue 4 opérateurs fondamentaux.

Opérateur OUI : opérateur identité, le comportement est décrit par la table de vérité suivante : si S est la sortie et a la variable d'entrée :

Table de vérité

a	S
0	0
1	1

Équation logique : $S = a$

Opérateur NON : opérateur négation

a	S
0	1
1	0

Équation logique : $S = \overline{a}$
lire **NON a** (ou a barre)
Codage en Python

```
S=not a
```

Propriétés

$$\overline{1} = 0 \quad \text{et} \quad \overline{0} = 1$$
$$\text{involution} \quad \overline{\overline{a}} = a$$

Opérateur ET : produit logique

a	b	S
0	0	0
1	0	0
0	1	0
1	1	1

Équation logique : $S = a \cdot b$
lire **a ET b**
Codage en Python

```
S=a and b
```

Propriétés

$$\begin{cases} 0 \cdot 0 = 0 \\ 0 \cdot 1 = 0 \\ 1 \cdot 0 = 0 \\ 1 \cdot 1 = 1 \end{cases} \quad \text{et} \quad \begin{cases} a \cdot 0 = 0 \\ a \cdot 1 = a \\ a \cdot a = a \\ a \cdot \overline{a} = 0 \end{cases}$$

— Le produit logique est commutatif, comme dans l'algèbre classique : $a \cdot b = b \cdot a$.

— 1 est l'élément neutre pour le produit logique : $a \cdot 1 = a$.

— 0 est l'élément absorbant pour le produit logique : $a \cdot 0 = 0$.

Opérateur OU : somme logique

a	b	S
0	0	0
1	0	1
0	1	1
1	1	1

Équation logique : $S = a + b$
lire **a OU b**
Codage en Python

```
S=a or b
```

Propriétés

$$\begin{cases} 0+0=0 \\ 0+1=1 \\ 1+0=1 \\ 1+1=1 \end{cases} \quad \text{et} \quad \begin{cases} a+0=a \\ a+1=1 \\ a+a=a \\ a+\overline{a}=1 \end{cases}$$

— La somme logique est commutative : $a + b = b + a$.

— 0 est l'élément neutre pour le produit logique : $a + 0 = a$.

— 1 est l'élément absorbant pour le produit logique : $a + 1 = 1$.

Dans l'algèbre usuelle, l'addition n'a pas d'élément absorbant.

b) Propriétés des opérateurs logiques

— Commutativité, associativité :
Le produit et la somme logique sont commutatifs et associatifs.

— Propriétés combinées de la somme et du produit

— Distributivité du produit logique par rapport à la somme logique

$$a \cdot (b + c) = a \cdot b + a \cdot c$$

— Distributivité de la somme logique par rapport au produit logique (cette propriété n'existe pas dans l'algèbre usuelle).

$$a + b \cdot c = (a + b) \cdot (a + c)$$

— Absorption

$$a + a \cdot b = a$$
$$a + \overline{a} \cdot b = a + b$$

c) Théorèmes de De Morgan

— Le complément d'un produit est égal à la somme des compléments des termes du produit.

$$\overline{a \cdot b} = \overline{a} + \overline{b}$$

— Le complément d'une somme est égal au produit des compléments des termes de la somme.

$$\overline{a + b} = \overline{a} \cdot \overline{b}$$

d) Règle du consensus

La règle du consensus est une règle de simplification des fonctions logiques, elle prend deux formes, celle d'une somme de produit, celle d'un produit de somme.

$$a \cdot c + b \cdot \overline{c} + a \cdot b = a \cdot c + b \cdot \overline{c}$$
$$(a + c) \cdot (b + \overline{c}) \cdot (a + b) = (a + c) \cdot (b + \overline{c})$$

Remarque : on pourra utiliser la description par une table de vérité pour vérifier ces assertions, puis retrouver le résultat par des simplifications à partir de l'algèbre de Boole.

12.3.2 Table de vérité

Une table de vérité est un tableau qui permet de décrire le fonctionnement d'un système combinatoire. L'état de chaque entrée et de chaque sortie est représenté par sa valeur logique. Toutes les combinaisons possibles des entrées sont étudiées pour déterminer l'état des sorties. Pour décrire complètement un système combinatoire comportant n entrées, il faut étudier les combinaisons possibles des entrées.

Pour un système combinatoire comportant n entrées, la table de vérité comporte 2^n combinaisons des variables d'entrée, donc 2^n lignes.

Il est possible de déterminer l'équation de fonctionnement en recherchant toutes les valeurs pour lesquelles la sortie est vraie (=1). L'équation de fonctionnement est égale à la somme logique de toutes les combinaisons pour lesquelles la sortie est vraie.

Exemple guidé : Va et vient

L'éclairage d'un couloir est commandé par deux interrupteurs, un à chaque extrémité (d à droite, g à gauche), la lampe est notée L.

— Une personne traverse le couloir de droite à gauche, en entrant, elle bascule l'interrupteur d pour éclairer la lampe, à l'autre extrémité, elle appuie sur g pour éteindre.

— Une seconde traverse à sa suite dans l'autre sens, elle bascule g pour éclairer, elle éteint à l'autre extrémité (d).

— Une troisième arrive à sa suite, éclaire le couloir en appuyant sur g, elle éteint en sortant.

On supposera qu'au début, les deux interrupteurs sont dans la même position. À cet état repos est associée la valeur logique 0. La figure 12.7 décrit chronologiquement les différents états des deux interrupteurs et de la lampe.

	Action	d	g	L	Commentaires
	État repos	0	0	0	La lampe est éteinte.
Première personne	Entrée en d	1	0	1	Appuie sur d (d=1), la lampe s'éclaire (L=1).
	traversée	1	0	1	La lampe reste éclairée.
	Sortie en g	1	1	0	Appuie sur g (g=1), la lampe s'éteint.
Deuxième personne	Entrée en g	1	0	1	(g=0) la lampe s'éteint.
	traversée	1	0	1	
	Sortie en d	0	0	0	(d=0) la lampe s'éteint.
Troisième personne	Entrée en g	0	1	1	(g=1), la lampe s'éclaire.
	Traversée	0	1	1	
	Sortie en d	1	1	0	(d=1) la lampe s'éteint.

FIGURE 12.7 – Fonctionnement d'un va et vient

On peut traduire de fonctionnement à l'aide d'une table de vérité.

	d	g	L	
les deux boutons au repos	0	0	0	
un bouton appuyé	1	0	1	$L = d \cdot \overline{g} = 1$
les deux boutons appuyés	1	1	0	$L = \overline{d} \cdot g = 1$
l'autre bouton relâché	0	1	1	

En recherchant les combinaisons, pour lesquelles la sortie est vraie, on peut déterminer l'équation logique décrivant le fonctionnement.

La lampe s'éclaire donc pour la combinaison suivante des variables d'entrée :

$$L = d \cdot \overline{g} + \overline{d} \cdot g$$

12.4 Fonctions logiques de base à 2 variables

12.4.1 Fonctions de base

À chaque opérateur de base, on associe une fonction logique représentée par un symbole graphique que l'on retrouve dans le tableau 12.1. À l'aide de ces symboles, on peut construire des schémas logiques (logigramme) traduisant l'équation logique.

Les fonctions logiques élémentaires permettent de décrire tous les fonctionnements logiques (voir le schéma logique du va et vient figure 12.8a).

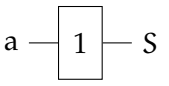
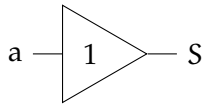
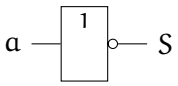
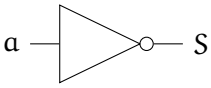
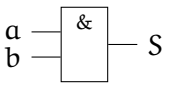
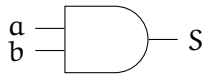
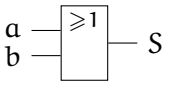
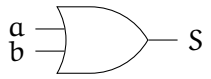
Fonction	table de vérité	symbole EU	symbole US	équation
OUI	a | S 0 | 0 1 | 1			$S = a$
NON	a | S 0 | 1 1 | 0			$S = \overline{a}$
ET	a b | S 0 0 | 0 1 0 | 0 0 1 | 0 1 1 | 1			$S = a \cdot b$
OU	a b | S 0 0 | 0 1 0 | 1 0 1 | 1 1 1 | 1			$S = a + b$

TABLE 12.1 – Fonctions logiques de base

12.4.2 Fonctions complexes

Il existe aussi d'autres fonctions[1], souvent utilisées en électronique pour construire des schémas plus complexes, ou plus compacts (tableau 12.2).

Les fonctions NON-OU (NOR) et NON-ET (NAND) sont des fonctions universelles, qui permettent de réaliser toutes les autres fonctions logiques.

Les fonctions NON-OU et NON-ET sont principalement utilisées en technologie électronique cablée (circuits intégrés) pour optimiser les circuits (1 seul type de composant pour toutes les fonctions).

1. Ces fonctions ne sont pas explicitement au programme.

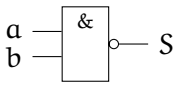
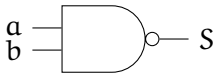
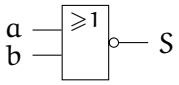
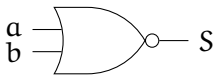
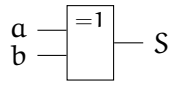
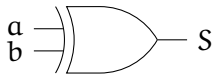
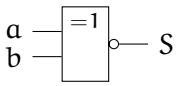
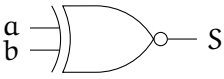
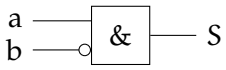
Fonction	table de vérité	symbole EU	symbole US	équation
NAND (NON-ET)	a b S 0 0 1 1 0 1 0 1 1 1 1 0			$S = \overline{a \cdot b} = \overline{a} + \overline{b}$
NOR (NON-OU)	a b S 0 0 1 1 0 0 0 1 0 1 1 0			$S = \overline{a + b} = \overline{a} \cdot \overline{b}$
XOR OUX ou exclusif	a b S 0 0 0 1 0 1 0 1 1 1 1 0			$S = \overline{a} \cdot b + a \cdot \overline{b}$ $S = a \oplus b$
XNOR NON- OUX non ou exclusif	a b S 0 0 1 1 0 0 0 1 0 1 1 1			$S = \overline{a} \cdot \overline{b} + a \cdot b$ $S = \overline{a \oplus b}$
INH inhibition	a b S 0 0 0 1 0 1 0 1 0 1 1 0			$S = a \cdot \overline{b}$

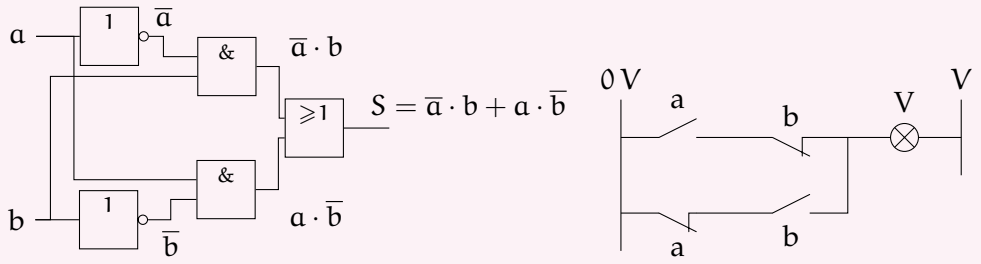
TABLE 12.2 – Tableau des fonctions logiques - suite

12.4.3 Logigramme

Un logigramme ou schéma logique est la représentation schématique d'une fonction logique obtenue en utilisant les symboles graphiques associés à chaque fonction. La figure 12.8a représente le schéma logique ou logigramme du va et vient.

Exemple guidé : Va et vient (suite)

On complète l'exemple du va et vient avec le circuit logique et le schéma électrique.



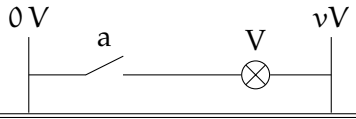
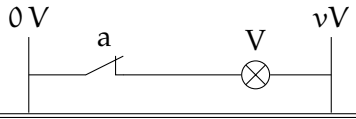
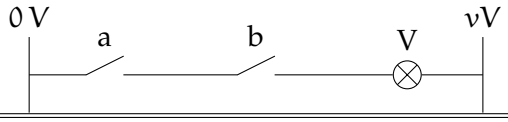
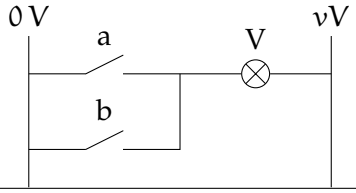
(a) Schéma logique (b) Technologie électrique

FIGURE 12.8 – Va et vient

12.4.4 Réalisation des fonctions logiques en technologie électrique câblée

a) Fonctions de base

Les fonctions logiques de base peuvent être réalisées à partir du câblage des contacts électriques en série et/ou en parallèle.

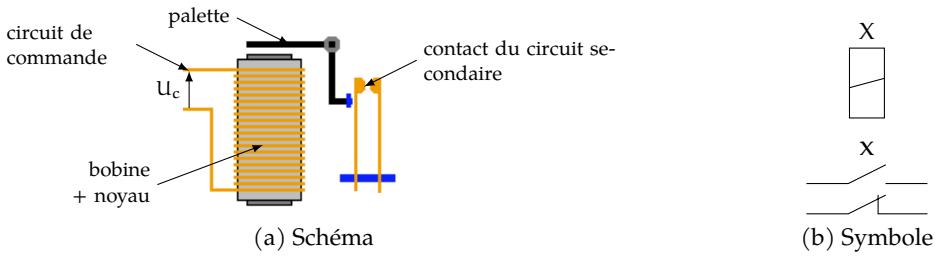
Fonction	Schéma
OUI	
NON	
ET	
OU	

La figure 12.8b correspond à la réalisation d'un circuit va et vient (ou exclusif) en technologie électrique. On remarque les fonctions $\overline{a}$ et $\overline{b}$ sont réalisées par des contacts à ouverture.

b) Utilisation d'un relais

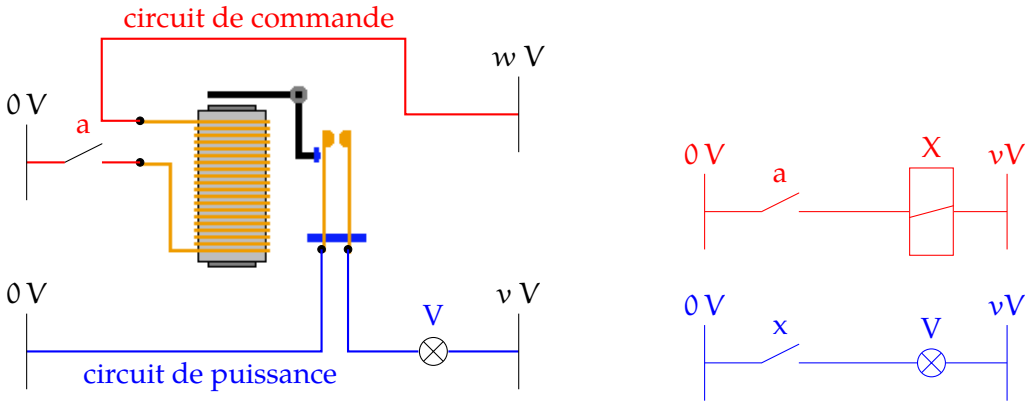
La réalisation de fonctions complexes ou de mémoire ne peut se faire en technologie électrique qu'en utilisant des relais électromagnétiques.

Un relais électromagnétique (figure 12.9) est constitué d'un électroaimant (une bobine et un noyau en fer doux). La bobine est alimentée par le circuit de commande, le champ électromagnétique créé permet de fermer un ou plusieurs contacts dans un circuit secondaire. Ce circuit secondaire peut être alimenté avec la même tension que le circuit de commande ou une tension propre au circuit de puissance.



(a) Schéma (b) Symbole

FIGURE 12.9 – Relais électromagnétique

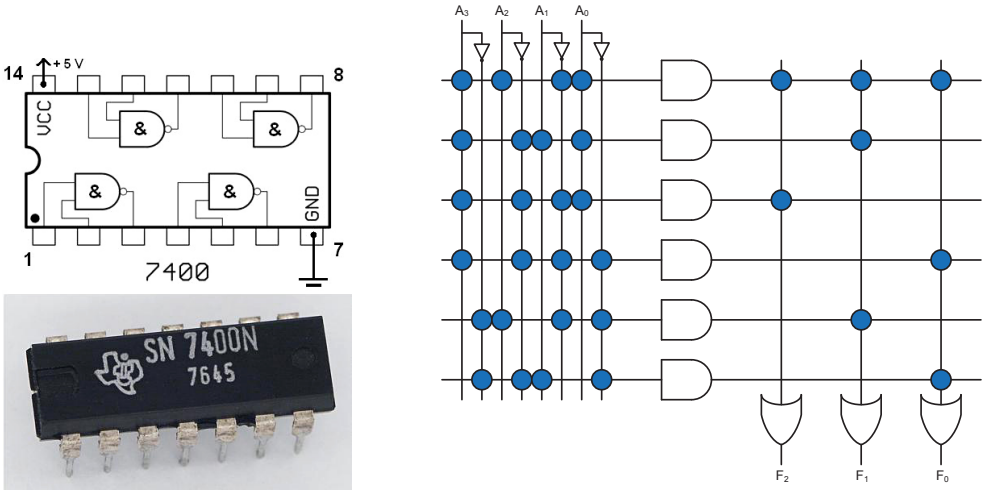


(a) Schéma de principe (b) Schéma normalisé

FIGURE 12.10 – Circuit élémentaire de commande d'un relais

12.4.5 Réalisation des fonctions logiques en technologie électronique

Les différentes fonctions logiques sont directement réalisées à partir de circuit intégré à base de semi-conducteur.



(a) Circuit intégré Nand 7400 (b) Schéma de principe d'un PAL

FIGURE 12.11 – Technologie électronique

Ces fonctions existent aussi bien sous la forme de composant discret que de circuit programmable.

Les fonctions discrètes sont la transposition des fonctions logiques, il suffit de les connecter comme sur le schéma.

Il existe aussi des composants de type P.A.L. (Programmable Array Logic) qui permettent de réaliser l'ensemble des fonctions possibles en combinant les opérateurs logiques OU, ET et NON. La figure 12.11b montre la structure interne d'un circuit P.A.L. à 4 entrées et à 3 sorties. L'équation souhaitée est obtenue en réalisant par programmation la connexion entre les lignes et les colonnes du réseau.

La sortie F_2 du schéma est ainsi :

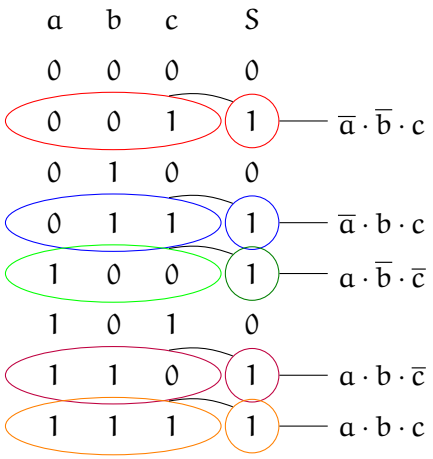
$$F_2 = A_0 \cdot \overline{A_1} \cdot A_2 \cdot A_3 + A_0 \cdot \overline{A_1} \cdot \overline{A_2}$$
$$F_2 = A_0 \cdot \overline{A_1} \cdot A_3$$

12.5 Détermination et simplification des fonctions logiques

12.5.1 Simplification à partir des relations de l'algèbre

Soit un comportement combinatoire décrit par une table de vérité, commençons par rechercher toutes les combinaisons pour lesquelles la sortie est vraie.

a	b	c	S
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1



La sortie est vraie pour la somme logique de toutes les combinaisons vraies.

$$S = \overline{a} \cdot \overline{b} \cdot c + \overline{a} \cdot b \cdot c + a \cdot \overline{b} \cdot \overline{c} + a \cdot b \cdot \overline{c} + a \cdot b \cdot c$$

Simplification à partir des règles de l'algèbre de Boole.

$$S = \overline{a} \cdot \overline{b} \cdot c + \overline{a} \cdot b \cdot c + a \cdot \overline{b} \cdot \overline{c} + a \cdot b \cdot \overline{c} + a \cdot b \cdot c \qquad \text{(factorisation)}$$
$$S = \overline{a} \cdot c \cdot (\overline{b} + b) + a \cdot \overline{b} \cdot \overline{c} + a \cdot b \cdot (\overline{c} + c) \qquad (b + \overline{b} = 1)$$
$$S = \overline{a} \cdot c + a \cdot \overline{b} \cdot \overline{c} + a \cdot b$$
$$S = \overline{a} \cdot c + a \cdot (\overline{b} \cdot \overline{c} + b) \qquad \text{(absorbtion : } \overline{b} \cdot \overline{c} + b = b + \overline{c})$$
$$S = \overline{a} \cdot c + a \cdot (\overline{c} + b)$$
$$S = \overline{a} \cdot c + a \cdot \overline{c} + a \cdot b$$
$$S = a \oplus c + a \cdot b \qquad \text{(ou exclusif : } \overline{a} \cdot c + a \cdot \overline{c} = a \oplus c)$$

Finalement

$$S = a \oplus c + a \cdot b$$

12.5.2 Tableau de Karnaugh

Les tableaux de Karnaugh sont une autre représentation d'une table de vérité. Un tableau de Karnaugh à n est un tableau de 2^n cases. À chaque case correspond une ligne (une combinaison) de la table de vérité.

Les lignes et les colonnes sont « numérotées » en fonction d'une combinaison des variables d'entrées de la table de vérité suivant un ordre qui présente la particularité suivante : d'une ligne ou d'une colonne aux suivantes et précédentes, seule une variable change d'état.

La figure 12.12 montre les tableaux de Karnaugh de 2 (a,b), 3 (a,b,c) et 4 (a,b,c,d) variables la combinaison correspondante est notée dans chaque case. On remarque bien que d'une case à l'autre, en ligne ou en colonne, il n'y a qu'une et une seule variable qui change et que la dernière colonne est adjacente à la première et la dernière ligne à la première.

a \ b	0	1
0	00	01
1	10	11

(a) Tableau de Karnaugh à 2 variables

a \ b c	00	01	11	10
0	000	001	011	010
1	100	101	111	110

(b) Tableau de Karnaugh à 3 variables

a b \ c d	00	01	11	10
00	0000	0001	0011	0010
01	0100	0101	0111	0110
11	1100	1101	1111	1110
10	1000	1001	1011	1010

(c) Tableau de Karnaugh à 4 variables

FIGURE 12.12 – Tableaux de Karnaugh

Simplification à partir des tableaux de Karnaugh

On reprend l'exemple précédent.

– On commence par reporter dans le tableau les valeurs de la fonction pour chacune des combinaisons des entrées (figure 12.13a).

– On réalise ensuite des groupes de 2^i cases adjacentes (figure 12.13b).

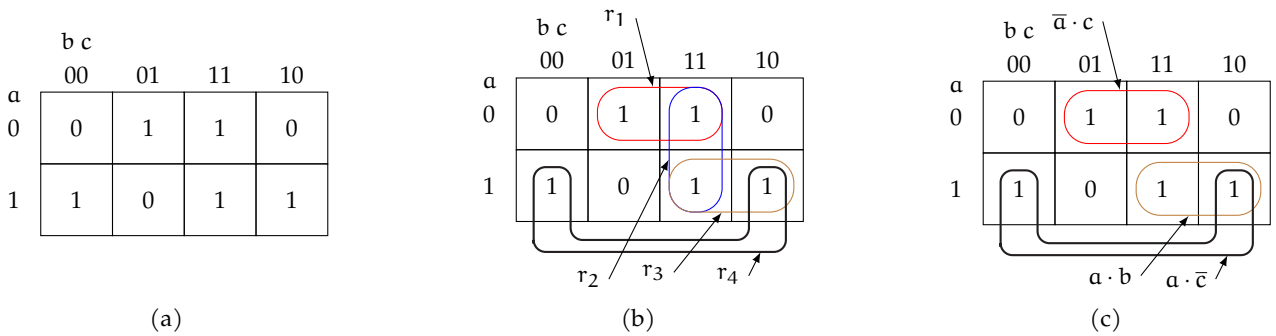
– On note le regroupement des deux cases extrêmes de la dernière ligne. Le groupe r_2 n'est pas utile, les deux 1 ayant déjà été associés à d'autres groupement.

– Pour chaque groupement, on ne conserve pour l'équation logique que les variables qui ne changent pas d'état (figure 12.13c).

– Finalement, on déduit l'équation de la sortie S en sommant les différentes combinaisons :
$S = r_1 + r_3 + r_4 = \overline{a} \cdot c + a \cdot b + a \cdot \overline{c}$.

– On continue si nécessaire les simplifications : $S = a \oplus c + a \cdot b$.

– On aurait pu aussi choisir les groupement r_1, r_2 et r_3, on obtient alors $S = r_1 + r_2 + r_4 = \overline{a} \cdot c + b \cdot c + a \cdot \overline{c}$.



(a) (b) (c)

FIGURE 12.13 – Simplification d'une fonction logique avec les tableaux de Karnaugh

12.6 Feuille de travaux dirigés n°12-a

Exercice 1 - Code barre entrelacé

X MP 2001 *Corrigé page 18*

Présentation

Le calculateur est équipé d'un lecteur optique de codes à barres capable de lire le code 2/5 INTERLEA VED (”2 parmi 5 entrelacé”) permettant d'identifier automatiquement les pièces à souder. Chaque pièce est identifiée par un nombre de quatre chiffres décimaux C3, C2, C1 et C0 ?

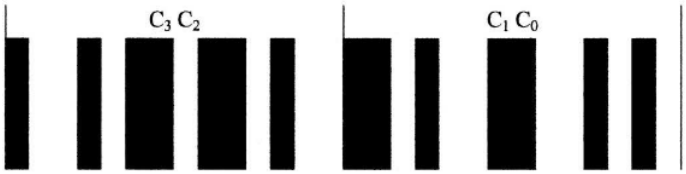


FIGURE 12.14 – Code entrelacé

Le code 2/5 INTERLEAVED utilise 5 bits (2 valent 1 et 3 valent 0) pour coder un chiffre décimal. Les chiffres de rang impair (C3 et C1) sont codés sur les barres noires, les chiffres de rang pair (C2 et C0) sont codés sur les espaces entre les barres noires. Les 1 sont codés par les barres ou espaces ”larges” (utilisant deux largeurs de base), les 0 sont codés par les barres ou espaces ”étroits” (utilisant une largeur de base).

Chaque chiffre de 0 à 9 est codé sur 4 bits a, b, c et d de poids respectifs 1, 2, 4 et 7. Le code est complété par un bit de contrôle de parité e.

	a	b	c	d	e
Poids	1	2	4	7	0
0	...				
9	...	...			

Q1. Déterminer les codes des chiffres de 1 à 9 en présentant le résultat sous forme de tableau. En déduire le code du chiffre 0 en justifiant son unicité. Déterminer le nombre correspondant au code de la figure 12.14.

Le calculateur traduit chaque chiffre de ce code à barres en un nombre binaire codé sur les quatre bits s_3, s_2, s_1 et s_0 (le poids du bit s_i vaut 2^i).

Q2. Établir la table de vérité des sorties s_i en fonction des entrées a, b, c, d et e. En déduire l'équation simplifiée de la sortie s_3.

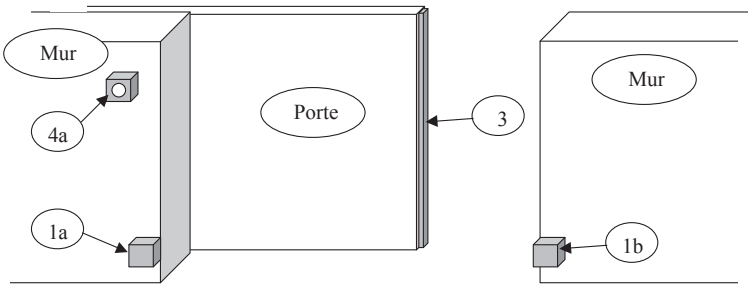
Exercice 2 - Porte coulissante

ATS 2004 *Corrigé page 18*

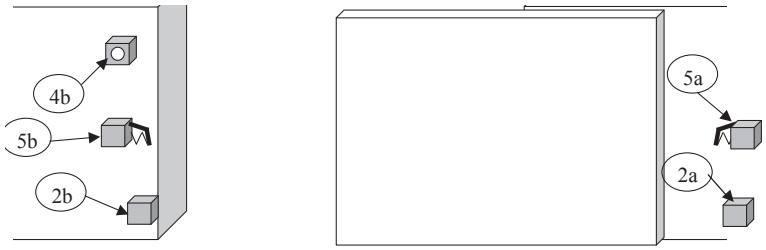
Pour la sécurité des personnes, la porte comporte deux barrières infra-rouges : l'une à l'intérieur du garage et l'autre à l'extérieur, ainsi qu'un palpeur qui est un capteur de contact localisé sur toute la tranche de la porte.

- 1a et 1b : barrière infra-rouge extérieure
- 2a et 2b : barrière infra-rouge intérieure
- 3 : palpeur
- 4a et 4b : capteurs de télécommande
- 5a et 5b : capteurs de fin de course

Les signaux logiques issus de ces dispositifs de sécurité sont notés I_1, I_2 et P_P. Ils valent 1 lorsque le système est au repos et 0 en cas d'activation de la sécurité.



(a) vue intérieure



(b) vue extérieure

R_X	S_O	F_C	S_C	M_O	A
0	0	0	0	1	1
0	0	0	1	0	1
0	0	1	0	1	1
0	0	1	1	0	1
0	1	0	0	0	1
0	1	0	1	0	1
0	1	1	0	0	1
0	1	1	1	0	1
1	0	0	0	1	0
1	0	0	1	1	0
1	0	1	0	1	1
1	0	1	1	1	1
1	1	0	0	1	0
1	1	0	1	1	0
1	1	1	0	1	1
1	1	1	1	1	1

(c) table de vérité

L'activation d'une sécurité doit être sans effet sur le système si la porte est en train de s'ouvrir, par contre si la porte est en train de se refermer elle doit bien sûr se rouvrir.

Les variables logiques qui commandent la porte sont notées M_O, M_F et A pour « marche ouverture », « marche fermeture » et « arrêt ».

Ces variables de commande sont liées à quatre variables de contrôle notées R_X, F_C, S_O et S_C qui sont décrites ci-dessous :

— R_X vaut 1 si l'ordre d'ouverture est donné par une télécommande

— S_O vaut 1 si la porte est en train de s'ouvrir

— F_C vaut 1 si la porte touche un capteur de fin de course

— S_C vaut 0 si une sécurité est activée.

Q1. Pourquoi le palpeur est-il nécessaire en plus des deux barrières infra-rouges ?
Q2. Donner l'expression de la variable S_C en fonction de I_1, I_2 et P_P.
Q3. À l'aide de la table de vérité donnée ci-dessus, déterminer l'expression la plus simple de A.
Q4. Représenter le schéma logique permettant d'obtenir A.
Q5. À l'aide de la table de vérité, déterminer l'expression la plus simple de M_O.
Q6. Représenter le schéma logique permettant d'obtenir M_O.

Exercice 3 - Codage Data Matrix

adapté du concours ICNA 2011 *Corrigé page 19*

Le code DataMatrix est un code bidimensionnel à haute densité, permettant de représenter une quantité importante d'informations sur une surface réduite.

Il se compose d'un motif extérieur permettant le repérage des lignes et des colonnes et d'une matrice de données située à l'intérieur.

Grâce à l'écriture redondante des informations, il offre un niveau de sécurité maximal : lecture possible d'un symbole partiellement effacé jusqu'à environ 20% de la surface totale, au contraire d'un code unidimensionnel qui n'offre aucune sécurité si le symbole est dégradé.

(a) Zoom (b) Taille réelle

FIGURE 12.15 – Code DataMatrix

La taille du DataMatrix dépend du nombre de caractères numériques à coder. Pour un message de 6 chiffres, la matrice intérieure sera constituée d'un carré de 10 lignes et 10 colonnes. La surface totale du DataMatrix pour un message de 6 chiffres sera alors de 10 mm^2, alors que le même message codé avec le code barre linéaire 39 occuperait plus de 30 mm^2.

A. Codage du DataMatrix

En étudiant l'exemple simple du message de 6 chiffres « 1 2 3 4 5 6 », le message de données sera constitué de 3 octets, chaque octet codant une paire de caractères sera codé selon la règle suivante :

Octet = (valeur numérique de la paire) +130

Ainsi :

« 12 »=12+130=142, « 34 »=34+130=164 et « 56 »=56+130=186

La séquence de données après encodage sera 142 164 186 en décimal.

Cette séquence sera complétée par 5 octets de correction d'erreurs, calculés par un algorithme de Reed-Solomon. Ces 5 octets représentent des informations redondantes par rapport aux données initiales. La séquence complète sera constituée des 8 octets suivants, donnés ici en décimal :

142 164 186 114 25 5 88 102

La séquence complète sera placée dans la matrice binaire du DataMatrix suivant les emplacements définis figure 12.16 :

2.1	2.2	3.6	3.7	3.8	4.3	4.4	4.5
2.3	2.4	2.5	5.1	5.2	4.6	4.7	4.8
2.6	2.7	2.8	5.3	5.4	5.5	1.1	1.2
1.5	6.1	6.2	5.6	5.7	5.8	1.3	1.4
1.8	6.3	6.4	6.5	8.1	8.2	1.6	1.7
7.2	6.6	6.7	6.8	8.3	8.4	8.5	7.1
7.4	7.5	3.1	3.2	8.6	8.7	8.8	7.3
7.7	7.8	3.3	3.4	3.5	4.1	4.2	7.6

FIGURE 12.16 – Grille de codage

1.1 correspond au premier bit du premier octet

1.2 au 2è bit du premier octet

Et ainsi de suite...

Un « 1 » correspond à une cellule noire, un « 0 » à une cellule blanche.

Q1.

Q1a. Donner l'écriture binaire codée sur 8 bits de chacun des 3 premiers octets de données seulement.

Q1b. Griser les cellules élémentaires correspondantes du DataMatrix sur le document réponse.

Q1c. Donner l'écriture hexadécimale (base 16) des 3 premiers octets de données étudiés.

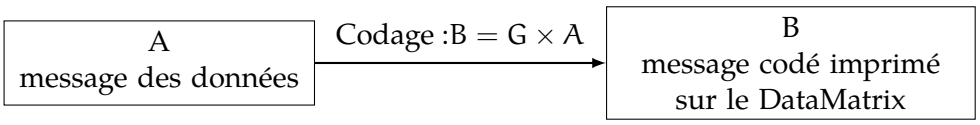
B. Correction d'erreurs

L'algorithme de correction d'erreurs utilisé dans le code DataMatrix est un algorithme très complexe. L'algorithme étudié ici, beaucoup plus simple, permet néanmoins de comprendre le principe de fonctionnement.

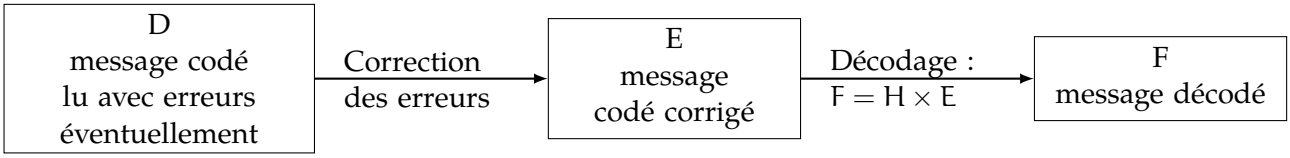
Les différentes étapes du processus codage/correction des erreurs / décodage sont schématisées sur la figure 12.17.

Les opérations de codage /décodage sont réalisées par des matrices génératrices G et H.

Soit $A = (a_0, a_1)$ le message de données de dimension 2 à coder.



(a) Processus de codage et écriture



(b) Processus de lecture et décodage

FIGURE 12.17 – Processus de lecture/écriture du DataMatrix

Soit $B = (b_0, b_1, b_2, b_3)$ le message codé qui sera imprimé.

On détermine B en fonction du message A, à partir de la matrice 4 lignes 2 colonnes $G = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \\ 0 & 1 \end{pmatrix}$

où l'opération d'addition correspond au OU exclusif et la multiplication au ET. Ainsi par exemple : $b_0 = 1 \cdot a_0 \oplus 0 \cdot a_1$ et $b_1 = 0 \cdot a_0 \oplus 1 \cdot a_1$ ...

à partir du calcul suivant $B = G \times A$ soit $\begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \\ 0 & 1 \end{pmatrix} \times \begin{pmatrix} a_0 \\ \\ a_1 \end{pmatrix}$

Q2. Donner l'expression simplifiée des quatre composantes b_i du message B en fonction des a_i en utilisant uniquement les fonctions logiques ET, OU et NON.

La lecture du code barre fournit le segment de message D de même dimension que le message B, mais qui ne correspond pas forcément exactement au message B si des erreurs d'impression ou de lecture apparaissent ou si le code barre a été partiellement effacé.

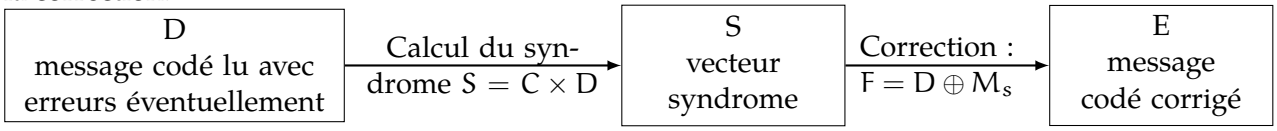
Q3. En considérant toutes les combinaisons possibles du message A sur 2 bits.

Q3a. Donner la liste des messages A différents à coder.

Q3b. Donner la liste des messages B correspondant à imprimer.

Q3c. Comparer avec le nombre de messages B possibles.

L'algorithme de correction est basé sur le calcul d'un vecteur S, appelé syndrome, qui va permettre la correction.



Soit $D = (d_0, d_1, d_2, d_3)$ le message codé lu et S le vecteur syndrome, noté $S = \begin{pmatrix} s_0 \\ s_1 \end{pmatrix}$ tel que :

$\begin{pmatrix} s_0 \\ \\ s_1 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 0 \\ \\ 0 & 1 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{pmatrix}$ avec C la matrice 2 lignes 4 colonnes : $C = \begin{pmatrix} 1 & 1 & 1 & 0 \\ \\ 0 & 1 & 0 & 1 \end{pmatrix}$.

Q4. Déterminer les équations du syndrome $S = \begin{pmatrix} s_0 \\ s_1 \end{pmatrix}$ et donner la valeur de S lorsque le message lu est correct.

Suivant la valeur du syndrome S, la correction est effectuée en comparant le mot reçu avec un mot M_S, caractéristique du syndrome : $E = D \oplus M_S$ obtenu par le calcul suivant : $\begin{pmatrix} e_0 \\ e_1 \\ e_2 \\ e_3 \end{pmatrix} = \begin{pmatrix} d_0 \oplus m_0 \\ d_1 \oplus m_1 \\ d_2 \oplus m_2 \\ d_3 \oplus m_3 \end{pmatrix}$.

La table des valeurs de M_S pour les différentes valeurs du vecteur syndrome est donnée ci-dessous :

$S = \begin{pmatrix} s_0 \\ s_1 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 1 \end{pmatrix}$	$\begin{pmatrix} 1 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 1 \\ 1 \end{pmatrix}$
$M_s = \begin{pmatrix} m_0 \\ m_1 \\ m_2 \\ m_3 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$

On considère que le message codé et imprimé est $B = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 1 \end{pmatrix}$.

Après lecture, le message lu est $D = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \end{pmatrix}$ présentant une seule erreur par effacement (c'est-à-dire remplacement d'un bit « 1 » par un bit « 0 »).

Q5. Appliquer l'algorithme de correction au message D et vérifier son efficacité.

On considère toujours que le message codé et imprimé est $B = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 1 \end{pmatrix}$.

Après lecture, le message lu est maintenant $D = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$ présentant cette fois deux erreurs par effacement sur 2 bits différents.

Q6. Appliquer l'algorithme de correction au message D et conclure sur l'efficacité de l'algorithme en cas d'erreur de transmission sur 2 bits.

Comme pour le codage, le décodage est donné par l'équation : $F = H \times E$.

Q7. Donner l'expression de la matrice de décodage H. À noter que, puisque le codage est redondant, il peut exister plusieurs expressions possibles pour la matrice H.