Circuits Numériques

MAYORQUIM

E.I.S.T.I - 2004

Sommaire

Liste de Figures

Liste de Tableaux

Partie I

Informations Générales

Avant Propos

Même si un domaine n'est pas complexe, il y a toujours quelque chose à en apprendre.
Lorsqu'il l'est, il est toujours possible d'en comprendre quand même quelque chose.
Antônio Carlos Lucena - Florianôpolis - Brésil - 1988

Je vous informe que ce polycopié du cours d'électronique est une actualisation de la documentation original du Prof. Guy ALmouzni de l'*EISTI*. La nouvelle version sera disponible en septembre de 2007 qui sera un résumé du livre *VHDL : Analyse et Synthèse de Circuits Numériques*.

Naturellement, les étudiants en informatique posent la question suivante : Pourquoi devons nous étudier la partie électronique si notre objectif est l'informatique. D'abord, il ne faut pas oublier que derrière un clavier, nous avons un système qui est composé d'un ensemble de composants électroniques. Pour profiter au maximum du potentiel de l'ordinateur, nous avons besoin d'une connaissance la plus large possible et encore de façon formelle. Bientôt, le microprocesseur sera *front-end* pour machine parallèle programmable au niveau du matériel, et que nous avons besoin de définir l'architecture du système informatique plus adaptée au problème et ensuite définir les instructions spécifiques qui seront utilisées par les microprocesseurs. Donc, dans un futur proche la connaissance en électronique sera plus utilisée par l'informaticien.

L'informatique n'est pas seulement déplacer la souris et faire un click.

MAYORQUIM
LASYN - Laboratoire de Systèmes Numériques et VHDL
LAPI - Laboratoire en Processus Intelligents
EISTI - École Internationale des Sciences du Traitement de l'Information

Remerciements

Ce travail pourra sembler peu dense pour un certain nombre d'étudiants. C'est grâce à Dieu, à la richesse de l'environnement humain de l'École Internationale des Sciences du Traitement de l'Information ainsi que le soutien des étudiants des promotions 2000, 2001 et 2002 que j'ai pu mener à bien ces travaux.

Je tient à remercier Mme SERGENT et les étudiants SCHNEIDER THOMAS, EDDEGDAG MAJDA, SCHWENDER THOMAS, HAMADAN FADI et COCQUEREZ BENJAMIN qui m'ont apporté une aide précieuse pendant la phase de correction de ce polycopié.

MAYORQUIM

Cours Numériques

Résumé

Informations pour le cours d'électronique de la promotion 2004. L'objectif est de faire apprendre les concepts de circuits numériques des ordinateurs et leurs liaisons avec le domaine analogique. La stratégie pédagogique sera divisée pour conquérir, i.e., le cours est composé de deux modules : - circuit numérique, langage *VHDL*. Pour profiter de tout le potentiel d'un ordinateur, il faut connaître un peu plus le matériel.

Localisation

Première Année - Département Physique - 1^{er} et $2^{\text{ème}}$ Semestre.

Objectif

Maîtriser l'électronique numérique pour préparer l'étude des différentes architectures d'ordinateurs.

Pré-requis

Connaissances de base en électronique analogique et numérique de mathématiques spéciales.

Moyens et Méthodes Pédagogiques

Cours magistraux, travaux dirigés, travaux pratiques et projet. L'école dispose des logiciels pour faire la simulation analogique (*CircuitMaker*) et simulation numérique (*VHDL - MaxPlus et Quartus II*). Les ressources pour le cours sont les suivantes : Laboratoire de Systèmes Numériques et VHDL, Tableau, barco (*projection*).

Contrôle de Connaissances

Chaque semestre aura deux devoirs surveillés de 2 (deux) heures pour évaluer les connaissances circuit numérique ou VHDL, et aussi les instruments du laboratoire. Nous avons aussi déterminé que pour le premier et deuxième semestre se dérouleront les examens, les travaux pratiques et le projet qui seront notés, comme l'illustre la Table 1. Si le résultat final de l'étudiant à la fin du semestre est inférieur à 8, il devra faire une rattrapage qui sera composé d'une partie théorique et d'une autre partie que sera en laboratoire.

Semestre	*Examen*	*TP N^o 1*	*TP N^o 2*	*TP N^o 3*	*TP N^o 4*	*TP N^o 5*
1^{er}	*50%*	*10%*	*10%*	*10%*	*10%*	*10%*
$2^{ème}$	*50%*	*10%*	*10%*	*10%*	*10%*	*10%*

Table 1: *Poids pour le contrôle des connaissances.*

Composition des Examens et Rattrapages

Les examens de 2 (deux) heures seront présentés comme suit :

⇒ La première question sera composée de quatre item. Chaque item vaudra deux points.

⇒ La deuxième question sera de synthèse et elle vaudra six points.

⇒ La troisième question de six points sera d'analyse.

⇒ Pour le deuxième examen il y aura une question sur le projet.

La rattrapage sera composé de cinq questions et la durée sera de trois heures. Les deux premières questions seront théoriques et la valeur sera de deux points et demi chacune. Les trois dernières seront de synthèse et le simulation avec un logiciel en laboratoire. Chaque question de synthèse aura la valeur de cinq points.

Documents du Cours

Polycopié du cours et des travaux dirigés. Les références bibliographiques suivantes sont disponibles à la bibliothèque de l'École :

⇒ a. Roth, C., "Advanced Digital Logic with VHDL," PWS, ISBN : 053495099X, Janvier, 1998.

⇒ b. Ashenden, P.J., "Designer's Guide to VHDL," Morgan Kaufmann Publishers, ISBN : 1558602704, Decembre, 1995.

⇒ c. Johns, D. & Martin K. W., "Analog Integrated Circuits," John Wiley & Sons Canada, ISBN : 0471144487, Decembre, 1996.

$\Rightarrow d.$ Gray, P.R. & Meyer, R. G., "Analysis and Design of Analog Integrated Circuits, " John Wiley & Sons, ISBN : 0471574953, Janvier, 1993.

$\Rightarrow e.$ Mano, M. M. & Kime, C. R., "Logic and Computer Design Fundamentals," Prentice Hall, ISBN : 0130124680, Janvier, 2000.

$\Rightarrow f.$ Rabaey, J. M., "Digital Integrated Circuits: A Design Perspective," Prentice Hall, ISBN: 0131786091, Decembre, 1995.

Travail Pratique et Projet

Chaque TP est effectué en binôme, 1 (un) binôme par poste de travail. Chaque TP fait l'objet d'une préparation et d'un compte-rendu (1 (un) compte-rendu par binôme). Pour la préparation, le binôme devra utiliser *Latex* et conformément au modèle lasyn.tex ou lasyn.ps (disponible sur réseau $O : \backslash corriges \setminus jma \setminus lasyn$) avant d'entrer au laboratoire. Si le TP n'a pas été préparé par le binôme, le résultat sera noté zéro. Pour le projet, l'évaluation aura lieu à l'examen.

Recommandations pour le Rapport

Pour écrire un rapport il y a une règle de base :

Penser de façon claire et écrire de façon organisée.

Si vous n'arrivez pas à avoir une pensée claire, alors vous ne maîtrisez pas le domaine étudié. Revenir aux phases de copéhension et formalisation.

Si vous n'arrivez pas à organiser le contenu de votre rapport, alors le déroulement du TP vous a echappé. Revenir sur les phases application et discussion.

Nous donnons dans la suite un bref aperçu d'un présentation-type du rapport des TP.

Introduction

Écrire un rapport de TP est une opportunité pour l'élève de penser de façon complète et analytique la totalité de l'expérience qui vient d'accomplir et de se poser la question sur la meilleure manière pour communiquer ce qu'il appris et les résultats auxquels il a abouti.

Le rapport peut être considéré comme une description succincte de votre travail en TP et des résultats que vous avez obtenus. Il peut servir comme référence par vous ou par d'autres personnes qu'elles l'ont lu et compris. La réalisation que vous avez développé, peut être réutiliser par vous ou par d'autres personnes. Nous voyons ainsi qu'un bon rapport de TP doit insister sur la compréhension qu'ont eu ses auteurs des concepts, de la théorie et des techniques utilisés. De plus, présenter seulement les résultats n'est pas suffisant. Il faut aussi expliquer en quoi ces résultats seraient différents si les entrées étaient différentes, examiner des cas particuliers et montrer ainsi, à travers la discussion sur les résultats, la connaissance que vous avez du domaine étudié.

Contenu

Dans un rapport on doit retrouver cinq sections fondamentales, à savoir :

- *Introduction* : Son objectif est de présenter le problème et la ou les méthodes utilisées. Nous devons aussi relier la méthode utilisée avec d'autres méthodes qui existent et expliquer les raisons du choix.

- *Métodes et Réalisation* : Présentation formelle, c'est-à-dire mathématique, de la méthode et des ses principales propriétés. Présentation aussi du diagramme réalisé. Il faut surtout insister sur la façon dont ce projet peut être utilisé par d'autres personnes et sur ses éventuelles limitations. Il faut aussi indiquer le domaine de variation admissible par la réalisation de toutes les entrées. Le diagramme du projet et la simulation sera mis en annexe.

- *Résultats* : Présentation claire en utilisant, si besoin, des tables te des graphiques. En particulier pour chaque résultat présenté il faut noter les valeurs de différents paramètres utilisés.

  Les graphiques et les tables utilisés doivent être bien étiquetés (par exemple Fig.1 ou Table 1).

- *Discussion* : Il s'agit de la partie la plus importante du rapport. Il faut :

  – Expliquer (pourquoi ces résultats? comparaison des résultats obtenus avec les résultats attendus, est-ce que la théorie est bien illustrée avec les résultats obtenus?),

  – Analyser (quelle est la signification des résultats, comment pouvons-nous les justifier, quelles conclusions peut-on tirer en se fondant sur ces résultats?, quelles sont les forces et les faiblesses de votre approche?),

  – Interpréter (les ambiguïtés qui restent, les questions auxquelles nous avons répondu et les questions ouvertes).

- *Conclusion* : Une ou deux phrases qui rappellent le problème et les grandes lignes des résultats obtenus. Nous pouvons aussi suugérer, s'il y a lieu, des études complémentaires.

Nous pouvons compléter un rapport par une section références qui doit faire mention à tous les documents que vous avez utilisé pour réaliser le TP et écrire le rapport. Nous pouvons aussi avoir une section d'annexes dans laquelle nous regroupons le diagramme du projet et, éventuellemnt, les signaux utilisées.

Style

Votre rapport doit être écrit en utilisant des phrases complètes et facilement compréhensible. Même si le style d'écriture est une caractéristique personnelle, un rapport doit être clair et grammaticalement et syntaxiquement correct.

Dans un rapport scientifique nous utilisons des verbes au passé, parce qu'il s'agit de présenter des expériences déjà effectuées. D'autre part on évite de s'autoréférencer, c'est-à-dire d'utiliser des pronoms personnels de la 1e personne, e.g. *mon rapport...* ou *j'ai effectué ...* .

Présentation

Tous les rapports doivent être préparés en utilisant **Latex (package pstricks)** et écrits avec des caractères de taille 10pt. Ils doivent faire entre deux pages minimum et cinq pages maximum.

Travail Dirigé

Le TD sera composé de deux parties :

$\Rightarrow$ 1. Un ensemble d'exercices seront résolus en groupe (étudiants et moi même) ;

$\Rightarrow$ 2. Un ensemble d'exercices nommé *Entraînez-vous !* sera résolu par les étudiants.

Planning des Enseignements

Premier Semestre

Théorie

$\Rightarrow$ 1. Algèbre de Boole : calcul booléen, fonctions booléennes et minimisation.

$\Rightarrow$ 2. Circuits combinatoires : portes, codeurs, décodeurs, démultiplexeurs.

$\Rightarrow$ 3. Familles logiques : RTL, TTL, CMOS, ECL, logique programmable (ROM et PLA).

$\Rightarrow$ 4. Arithmétiques binaires, circuits itératifs et cellulaires.

$\Rightarrow$ 5. Circuits séquentiels : bascules, registres, compteurs, automates et méthodes de synthèse.

$\Rightarrow$ 6. Synthèse de circuits complexes, comportant un contrôleur et une partie opérative.

Travaux Pratiques

⇒ 1. Introduction aux instruments, montage et simulation d'un circuit combinatoire.

⇒ 2. Addition, Multiplexeur et troisième état.

⇒ 3. Réalisation de Compteur et de registre.

⇒ 4. Première partie du processeur.

⇒ 5. Deuxième partie du processeur.

Deuxième Semestre

Théorie

⇒ 1. Positionnement des problèmes : Analyse jusqu'à la réalisation.

⇒ 2. Les supports de réalisation : ASIC , FPGA, circuits standards.

⇒ 3. Méthodes de conception : Fonctionnelle et structurée.

⇒ 4. Concepts et syntaxe du langage VHDL.

⇒ 5. Exemples d'applications modélisées en VHDL.

⇒ 6. Synthèse de circuits complexes, comportant un contrôleur et une partie opérative.

Travaux Pratiques

⇒ 1. Introduction au logiciel MaxPlus et Implementation d'un code sur la carte Altera..

⇒ 2. Codification et simulation par la description structurelle et flot des données ;

⇒ 3. Codification et simulation d'une machine d'état ;

$\Rightarrow 4.$ Première partie du processeur par le langage *VHDL*

$\Rightarrow 5.$ Deuxième partie du processeur par le langage *VHDL*.

Remarques

$\Rightarrow 1.$ Le TP sera noté si le montage fonctionne.

$\Rightarrow 2.$ Du TP et du projet résultera une note négative si les étudiants ont copié et si les étudiants ont laissé copier ;

$\Rightarrow 3.$ Tricher à l'examen donne zéro comme note finale à la fin du semestre ;

$\Rightarrow 4.$ Les documentations sur le modèle du rapport, les information sur les composants seront disponibles à l'adresse suivante : $O : \backslash corriges \backslash jma \backslash lasysn$.

Partie II

Système Logique

Chapitre 1

Logique Combinatoire

1.1 *Introduction*

L'électronique logique ou électronique numérique ou encore système logique, comme l'illustre la Figure 1.1. Il mettent en jeu des signaux binaires (n'ayant que **2** états possibles) appelés bit (*binary digits*) et notés **0** (état logique bas) et **1** (état logique haut).

Figure 1.1: *Schéma du système logique.*

1.1.1 *Système Logique Combinatoire*

A l'instant discret $\mathbf{n}$, une sortie $\mathbf{S_j}$ notée $\mathbf{S_j^n}$, d'un système logique combinatoire, comme l'illustre la Figure 1.2, ne dépend que de ses entrées $\mathbf{e_1^n}, ..., \mathbf{e_p^n}$ au même instant.

$$S_j^n = f(e_1^n, ..., e_p^n) \text{ ou } 1 \leq j \leq m \tag{1.1.1}$$

Remarque 1.1.1 *Il faut observer que la seule connaissance des entrées suffit à déterminer les sorties.*

1.1.2 *Système Logique Séquentiel*

A l'instant discret $\mathbf{n}$, une sortie $\mathbf{S_j^n}$ d'un système logique séquentiel, comme l'illustre la Figure 1.3, dépend de ses entrées $\mathbf{e_1^n}, ..., \mathbf{e_p^n}$ mais aussi de l'état antérieur des sorties $\mathbf{S_1^{n-1}}, ..., \mathbf{S_m^{n-1}}$

Figure 1.2: *Schéma du système logique combinatoire.*

qui peuvent être considérées comme des entrées secondaires, alors que les entrées $\mathbf{e_1^n}, ..., \mathbf{e_p^n}$ sont appelées primaires.

$$S_1 j^n = f(e_1^n, ..., e_p^n, S_1^{n-1}, ..., S_m^{n-1}) \text{ ou } 1 \leq j \leq m \tag{1.1.2}$$

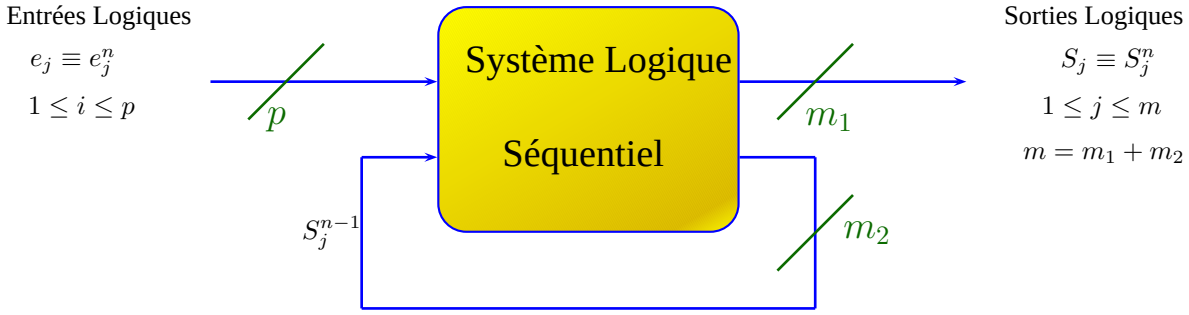

Figure 1.3: *Schéma du système logique séquentiel.*

Remarque 1.1.2 *Notion de mémoire, car les systèmes séquentiels sont bouclés, ou encore récursifs et la seule connaissance des entrées (primaires) ne suffit pas à déterminer l'état des sorties.*

1.2 *Algèbre de Boole*

1.2.1 *Opérateurs Fondamentaux*

En algèbre de *Boole*, une variable, ou une fonction, ne peut prendre que deux valeurs binaires que nous notons symboliquement **0** et **1**. A l'aide de variables binaires (donc à **2** états) nous pouvons néanmoins décrire des variables ayant un plus grand nombre d'états, en constituant ces dernières comme des mots binaires. Un mot binaire est une association de variables binaires. Ainsi un mot constitué de $\mathbf{m}$ bits ou variables binaires peut décrire 2^m combinaisons logiques.

Exemple 1.2.1 *Mot binaire M composé de m = 3 variables logiques $A_2, A_1, A_0 \rightarrow M = f(A_2, A_1, A_0)$ peut prendre $2^3 = 8$ valeurs.*

Le Complément

Le complément est égal à l'opérateur *NON* (*NOT*). Donc, le complément de $\mathbf{X}$ est $\overline{\mathbf{X}}$ ou $\overline{\mathbf{x}}$ qui vaut $\mathbf{1}$ si $\mathbf{X} = \mathbf{0}$ et qui vaut $\mathbf{0}$ si $\mathbf{X} = \mathbf{1}$. Un opérateur peut être associé à une représentation géométrique issue de la théorie des ensembles que nous appelons diagramme de *Venn*, comme l'illustre la Figure 1.4.

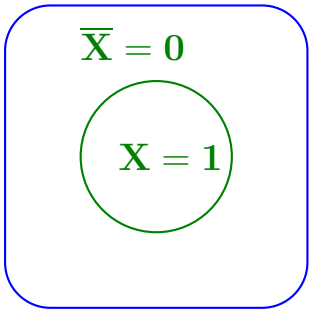

Figure 1.4: *Diagramme de Venn pour le complément.*

La Somme Logique

La somme logique est l'opérateur *OU* (*OR*) de deux variables *booléennes*. Cet opérateur est noté par le signe $+$, mais aussi par le signe $\bigvee$ ou encore $\bigcup$. Le résultat de cette opération logique de deux variables *booléennes* $A + B = 1$ si A ou B (ou les deux) vaut 1 et $A + B = 0$ sinon.

$$A + B \equiv A \vee B \equiv A \cup B \qquad (1.2.1)$$

Le signe $+$ n'a pas ici la signification habituelle, il est évident en effet qu'en algèbre de Boole, donc $1 + 1 = 1$. Si une ambiguïté peut exister, il vaut mieux alors utiliser la notation usuelle de la théorie des ensembles, comme l'équation $A \cup B$.

Nous considérons un plan dans lequel nous délimiterons une région où la variable A vaut 1 et une autre région dans laquelle B vaut 1. La somme logique a pour valeur 1 dans la surface formée par la réunion des deux régions précédentes, comme l'illustre le diagramme de *Venn* de la Figure 1.5.

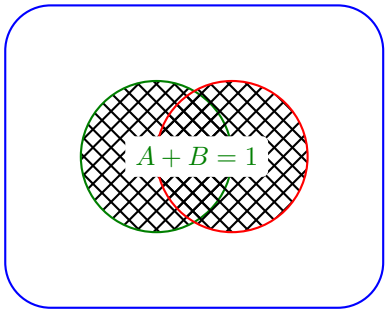

Figure 1.5: *Diagramme de Venn pour la somme logique.*

Le Produit Logique

Le produit logique est l'opérateur *ET* (*AND*) de deux variables *booléennes*. Cet opérateur est noté par le signe $\bullet$, mais aussi par le signe $\wedge$ ou encore $\cap$ ou même pas de signe, comme l'illustre l'équation 1.2.2. Le résultat de cette opération logique de deux variables *booléennes* $A.B = 1$ si A et B vaut 1 et $A + B = 0$ sinon.

$$A.B \equiv A \wedge B \equiv A \cap B \qquad (1.2.2)$$

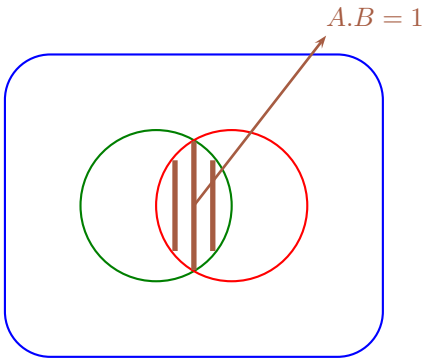

Figure 1.6: *Diagramme de Venn pour le produit logique.*

L'opération produit logique peut être représenté par le diagramme de *Venn* comme l'illustre la Figure 1.6.

1.3 *Propriétés Logiques*

1.3.1 *Élement Neutre*

$$\begin{array}{rcrcl} A & + & 0 & = & A \\ A & . & 1 & = & A \end{array} \tag{1.3.1}$$

1.3.2 *Élement Absorbant*

$$\begin{array}{rcrcl} A & + & 1 & = & 1 \\ A & . & 0 & = & 0 \end{array} \tag{1.3.2}$$

1.3.3 *Complément*

$$\begin{array}{rcrcl} A & + & \overline{A} & = & 1 \\ A & . & \overline{A} & = & 0 \\ & & \overline{\overline{A}} & = & A \end{array} \tag{1.3.3}$$

1.3.4 *Idempotence*

$$\begin{array}{rcrcl} A & + & A & = & A \\ A & . & A & = & A \end{array} \tag{1.3.4}$$

Les opérations d'addition et de multiplication logique ont les propriétés des opérations de même nom en arithmétique classique : commutativité, associativité, distributivité.

1.3.5 *Associativité*

$$\begin{array}{ccccccc} (A+B)+C & = & A+(B+C) & = & A+B+C \\ (A.B).C & = & A.(B.C) & = & A.B.C \end{array} \tag{1.3.5}$$

1.3.6 *Commutativité*

$$\begin{array}{ccc} A+B & = & B+A \\ A.B & = & B.A \end{array} \tag{1.3.6}$$

1.3.7 *Double Distributivité*

$$\begin{array}{lcll} A.(B+C) & = & A.B+A.C & Distributivité\ .\ par\ rapport\ \grave{A}+ \\ A+(B.C) & = & (A+B).(A+C) & Distributivité\ +\ par\ rapport\ à\ . \end{array} \tag{1.3.7}$$

Parmi les relations simples les plus utilisées nous citerons les relations d'absorption.

1.3.8 *Propriétés de l'Élément Absorption*

$$\begin{array}{ccc} A+A.B & = & A \\ A.(A+B) & = & A \\ A+\overline{A}.B & = & A+B \\ X.A+X.\overline{A} & = & X \end{array} \tag{1.3.8}$$

1.4 *Théorèmes De Morgan*

Ils définissent les relations entre l'opération complément et les deux autres opérations de base (OU et ET). Le complément d'une somme est égal au produit des compléments des termes, et le complément d'un produit est égal à la somme des compléments des termes.

Théorème 1.4.1 *(De Morgan)*

$$\begin{array}{ccc} \overline{A+B} & = & \overline{A}.\overline{B} \\ \overline{A.B} & = & \overline{A}+\overline{B} \end{array} \tag{1.4.1}$$

Ces théorèmes, comme l'illustre l'équation 1.4.1, peuvent être montrés à l'aide des diagrammes de *Venn* ou de table de vérité ou encore de façon algébrique.

1.5 *Principe de Dualité*

Une équation logique reste vraie si nous remplaçons $+$ (OU) par . (ET) et 0 *zéro* par 1 (un) et réciproquement.

Exemple 1.5.1

$$\begin{array}{ccc} A+B & = & B+A \\ \Downarrow & & \Downarrow \\ A.B & = & B.A \end{array} \tag{1.5.1}$$

$$\begin{array}{ccc} A+1 & = & 1 \\ \Downarrow & & \Downarrow \\ A.0 & = & 0 \end{array} \tag{1.5.2}$$

1.6 *Autres Fonctions Élémentaires de Deux Variables*

Les trois fonctions précédentes suffisent à elles seules à effectuer toutes les opérations. Nous définissons cependant quelques fonctions annexes.

1.6.1 *Le* OU Exclusif *(*XOR*)*

La fonction A *XOR* B est notée $A \oplus B$, cette fonction vaut 1 si A ou B vaut 1, mais pas les deux à la fois. Nous avons bien évidemment la relation $A \oplus B = \overline{A} \oplus \overline{B}$, et nous pouvons nous exprimer à partir des opérations élémentaires $A \oplus B = A.\overline{B} + \overline{A}.B$. Le *OU Exclusif* de deux variables A, B traduit l'inégalité de ces deux variables. La fonction inverse, traduisant l'égalité, est la coïncidence, notée $\odot$. Donc, la fonction $f(A,B) = A \odot B = \overline{A \oplus B}$ ou $f(A,B) = A \odot B = A.B + \overline{A}.\overline{B}$. Nous avons bien évidemment la relation, $A \odot B = \overline{A} \odot \overline{B}$. Le symbole utilisé pour le *OU Exclusif* est identique à celui désignant en arithmétique, i.e., une addition en modulo 2, il s'agit en effet de la même opération d'addition sans retenue.

1.6.2 *Les Fonctions* NOR *et* NAND

La fonction *NOR* est le complément de *OU* i.e., *NOR* est égal NON *OR*, et nous pouvons écrire $\overline{A+B}$, ou A *NOR* B ou nous trouvons parfois la notation suivante introduite par *PIERCE* : $A \downarrow B$.

La fonction *NAND* est le complément de *ET* i.e., *NAND* est égal *NON AND*, et nous pouvons écrire $\overline{A.B}$, ou A *NAND* B ou nous trouvons parfois la notation suivante introduite par *PIERCE* : $A \uparrow B$.

La combinaison de ces diverses opérations logiques entre plusieurs variables constitue ce que nous appelons les fonctions logiques, comme par exemple l'équation1.6.1.

$$f = f(A,B,C) = A + B.\overline{C} + A.\overline{B}.C \tag{1.6.1}$$

1.7 *Représentation des Fonctions*

1.7.1 *Diagramme de Venn*

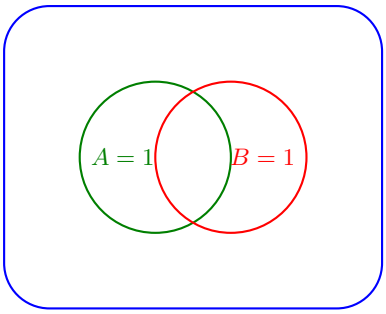

Figure 1.7: *Diagramme de Venn pour la théorie des ensembles.*

C'est la représentation, comme l'illustre la Figure 1.7, issue de la théorie des ensembles.

1.7.2 *Chronogramme*

C'est la représentation de la fonction logique en fonction du temps, comme l'illustre la Figure 1.8, pour diverses valeurs des variables d'entrées.

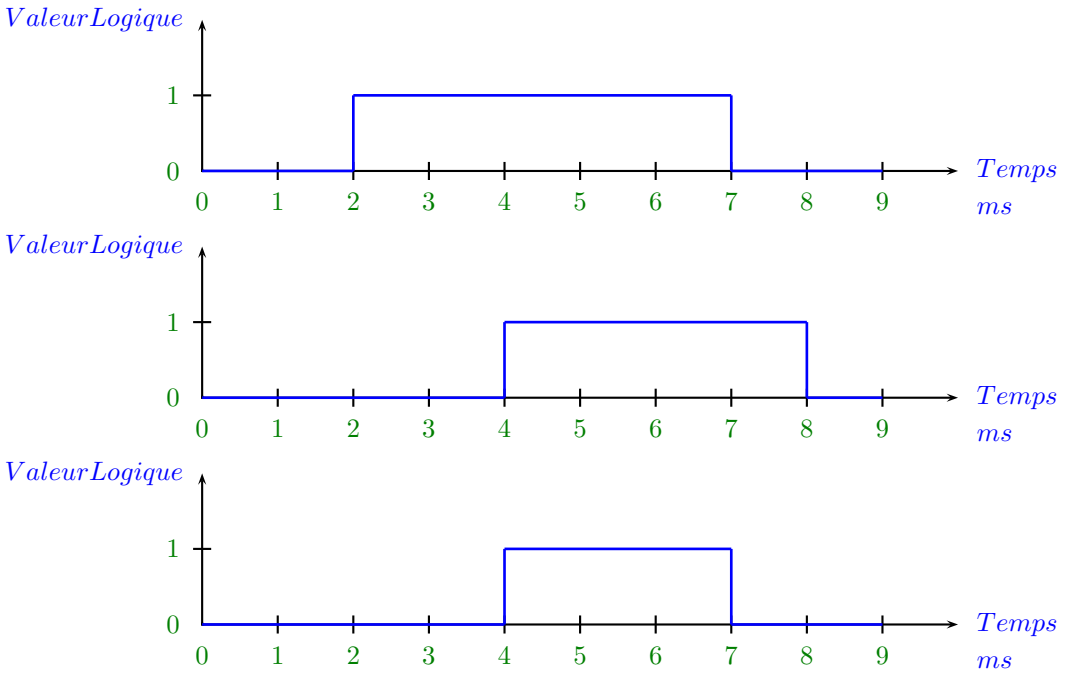

Figure 1.8: *Chronogramme de temps pour la fonction logique ET (AND).*

1.7.3 *Table de Vérité*

C'est le tableau des valeurs de la fonction pour toutes les valeurs possibles des variables d'entrées.

A	$\overline{A}$
0	1
1	0

Table 1.1: *Fonction logique NON (NOT).*

A	B	$A+B$
0	0	0
0	1	1
1	0	1
1	1	1

Table 1.2: *Fonction logique OU (OR).*

A	B	$A.B$
0	0	0
0	1	0
1	0	0
1	1	1

Table 1.3: *Fonction logique ET (AND).*

A	B	$\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

Table 1.4: *Fonction logique NON OU (*NOR*).*

A	B	$\overline{A.B}$
0	0	1
0	1	1
1	0	1
1	1	0

Table 1.5: *Fonction logique NON ET (*NAND*).*

A	**B**	**A $\oplus$ B**
0	**0**	**0**
0	**1**	**1**
1	**0**	**1**
1	**1**	**0**

Table 1.6: *Fonction logique* OU Exclusif *(*XOR*).*

A	**B**	**A $\odot$ B**
0	**0**	**1**
0	**1**	**0**
1	**0**	**0**
1	**1**	**1**

Table 1.7: *Fonction logique COÏNCIDENCE (N*XOR*).*

1.7.4 *Diagramme de Karnaugh*

C'est une forme particulière de la table de vérité. Le diagramme de *Karnaugh* se compose d'un rectangle divisé en 2^n cases, n étant le nombre de variables de la fonction considérée. Dans chacune de ces cases les variables ont une valeur déterminée et nous y plaçons un 0 ou un 1 suivant la valeur correspondante de la fonction. L'ordre des variables en abscisse et ordonnée est tel que lorsque nous le passons d'une case à la case adjacente une seule variable est modifiée. Deux cases sont adjacentes si elles sont voisines verticalement, horizontalement ou en coin, mais toujours de telle sorte qu'une seule variable d'entrée est modifiée lorsque nous le passons d'une case à une case adjacente.

Diagramme de Karnaugh à Deux Variables

Soient A et B les variables *booléennes,* donc nous aurons 2^2 cases, comme l'illustre la Figure 1.9.

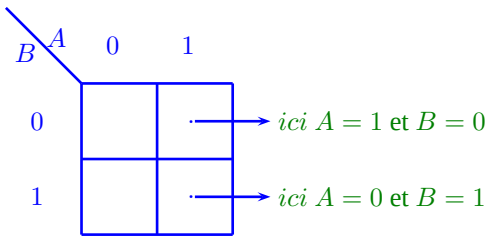

Figure 1.9: *Localisation des cases du diagramme de Venn.*

Exemple 1.7.1 *Représentons la fonction* OU Exclusif *: $S = A \oplus B$ sur le diagramme de* Karnaugh*, comme l'illustre la Figure 1.10, i.e., la sortie S est égale 1 quand $A \neq B$, sinon est 0. Nous observons qu'elles sont les 2 cases suivant la $2^{éme}$ diagonale.*

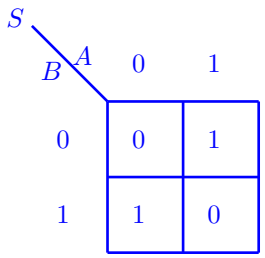

Figure 1.10: *Diagramme de Venn pour la fonction* OU Exclusif.

Diagramme de Karnaugh à Trois Variables

Nous aurons $2^3 = 8$ cases, donc il faut utiliser un rectangle ayant par exemple 4 lignes et 2 colonnes, mais nous pouvons prendre aussi 2 lignes et 4 colonnes. Nous remarquerons que de la deuxième à la troisième ligne nous passons de $(0\,1)$ à $(1\,1)$ et non de $(0\,1)$ à $(1\,0)$ de façon à ne modifier qu'une variable à la fois, comme le code *Gray*, comme l'illustre la Figure 1.11.

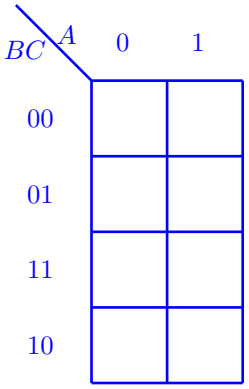

Figure 1.11: *Diagramme de Venn pour une fonction logique de trois variables.*

Exemple 1.7.2 *La fonction $S = A.\overline{B} + C.\overline{A}$ dont la table de vérité est illustrée par la Table 1.8 et le diagramme de* Karnaugh *est illustré par la Figure 1.12.*

A	**B**	**C**	**A.$\overline{\mathbf{B}}$**	**C.$\overline{\mathbf{A}}$**	**S**
0	**0**	**0**	**0**	**0**	**0**
0	**0**	**1**	**0**	**1**	**1**
0	**1**	**0**	**0**	**0**	**0**
0	**1**	**1**	**0**	**1**	**1**
1	**0**	**0**	**1**	**0**	**1**
1	**0**	**1**	**1**	**0**	**1**
1	**1**	**0**	**0**	**0**	**0**
1	**1**	**1**	**0**	**0**	**0**

Table 1.8: *Table de vérité pour la fonction logique $S = a \bullet b' + c \bullet a'$.*

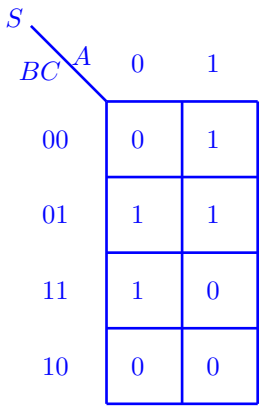

Figure 1.12: *Diagramme de Karnaugh pour une fonction logique* $S = A.\overline{B} + C\overline{A}$.

Diagramme de Karnaugh à Quatre Variables

Nous aurons $2^4 = 16$ cases, nous retombons sur un carré, comme l'illustre la Figure 1.13. L'ordre des variables est le même que le précédent. Donc, il faut utiliser le code *Gray* i.e., $00 - 01\ \ 11 - 10$.

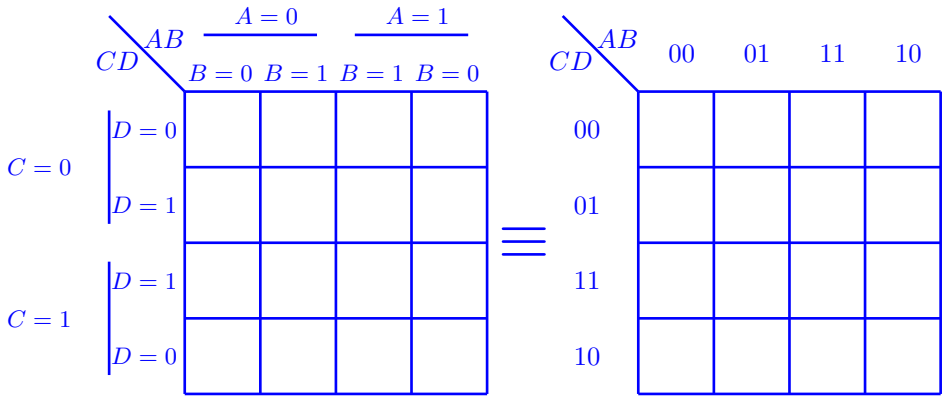

Figure 1.13: *Diagramme de Karnaugh pour la fonction logique de quatre variables.*

Exemple 1.7.3 *Nous pourrons, en établissant la table de vérité, montrer le diagramme de* Karnaugh *de la Figure 1.14.*

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

Figure 1.14: *Diagramme de Karnaugh pour la fonction logique* $S = A.\overline{B}.C.D + \overline{A}.B.C.$

Diagramme de Karnaugh à Cinq Variables

Lorsque nous le passons d'une case d'un diagramme de *Karnaugh* à la case adjacente, une seule variable est modifiée. Avec cinq variables, il faut que chaque case soit adjacente à cinq cases, ce qui n'est pas possible dans une représentation plane. Il faut faire appel à un volume à $2^5 = 32$ cases cubiques que nous pouvons remplacer par un double tableau carré, comme l'illustrent les Figures 1.15 et 1.16.

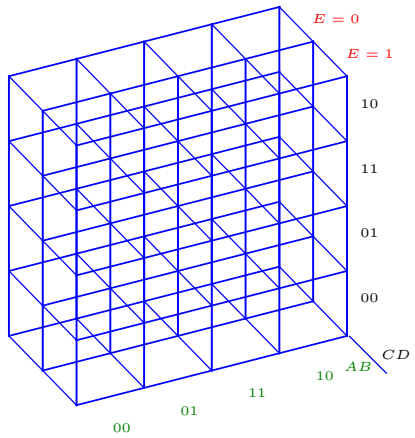

Figure 1.15: *Diagramme de Karnaugh pour la fonction logique de cinq variables superposeés.*

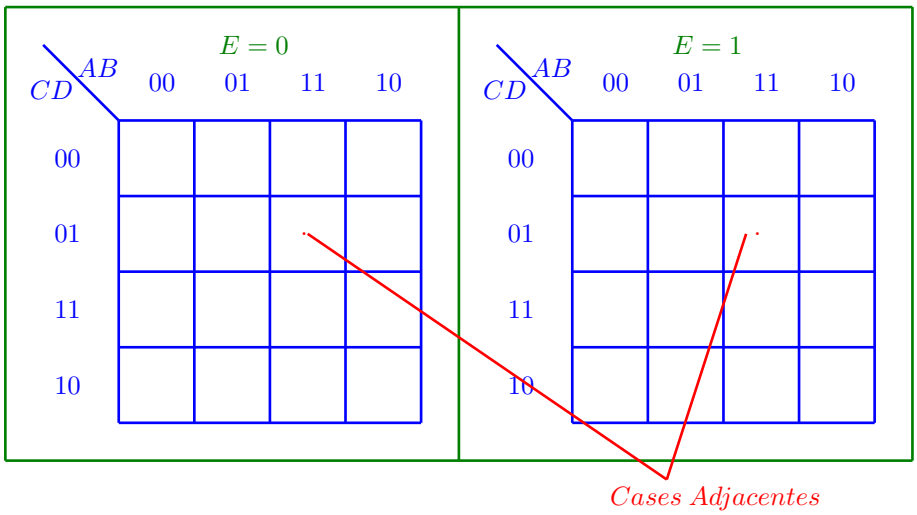

Figure 1.16: *Diagramme de Karnaugh pour la fonction logique de cinq variables.*

La commodité d'emploi est alors très réduite car il est plus délicat de repérer deux cases adjacentes. La situation est encore pire avec six variables où il faut travailler avec un cube ou quatre tableaux carrés. Au delà de six variables aucune représentation n'est possible et il faudra faire appel à d'autres procédés. Les diagrammes de *Karnaugh* permettent comme nous allons le voir de simplifier très facilement des fonctions *booléennes* complexes. Au delà de cinq variables des méthodes algébriques peuvent toujours les remplacer mais sont beaucoup moins souples et d'un intérêt contestable.

1.7.5 *Forme Canonique*

Toute fonction logique peut être mise sous forme canonique : - comme une somme de *mintermes* ou encore, comme un produit de *maxtermes*.

Nous appelons *minterme*, ou fonction unité de n variables, un produit de ces n variables ou de leur complément. Par exemple, $AB\overline{C}D$ est un *minterme* des quatre variables A, B, C, D mais $A\overline{C}D$ n'en est pas un, car il manque la variable B.

Une fonction *boolénne* de p variables est décrite comme une somme canonique si elle est mise sous la forme d'une somme de *mintermes* de ces p variables. Nous définissons également un produit canonique qui est le produit de sommes contenant chacune toutes les variables. Par exemple, $(A+\overline{B}+C+D).(\overline{A}+B+\overline{C}+D).(A+B+\overline{C}+D)$ est un produit canonique des quatre variables A,B, C, D.

Les diagrammes de *Karnaugh* permettent très facilement de mettre une fonction logique sous forme d'une somme canonique car chaque *minterme* correspond à un 1 dans une seule case du diagramme.

Remarque 1.7.1 *Chaque terme est un* maxterme *qui est le complément d'un* minterme.

1.8 *Minimisation*

Pourquoi simplifier une fonction logique ? Pour donner lieu à une réalisation matérielle la plus simple possible mettant en jeu un nombre minimal de circuits logiques.

1.8.1 *Méthode Algébrique*

La méthode utilise des propriétés des opérations logiques élémentaires et des théorèmes de *De Morgan*. La mise en équation d'un problème de logique peut conduire à une fonction booléenne assez complexe pouvant, par des opérations algébriques simples, se mettre sous une forme beaucoup plus condensée.

Exemple 1.8.1 *Soit la fonction logique de l'équation 1.8.1.*

$$S = AC + A\overline{B} + B + A\overline{D} + ABD + A\overline{C} + AB \tag{1.8.1}$$

A partir de la méthode algébrique, nous obtenons :

$$\begin{array}{ccccccccccccc} S & = & AC & + & A\overline{B} & + & B & + & A\overline{D} & + & ABD & + & A\overline{C} & + & AB \\ & & \Downarrow & & \Downarrow & & \Downarrow & & \Downarrow & & \Downarrow & & \Downarrow & & \Downarrow \\ & & 1 & & 2 & & 3 & & 4 & & 5 & & 6 & & 7 \end{array}$$

$$\begin{array}{lccccccccccc}
En\ groupant\ les\ termes\ 2\ et\ 4 \Rightarrow & A\overline{B} & + & A\overline{D} & = & A(\overline{B}+\overline{D}) & = & A\overline{BD} & & & \\
en\ ajoutant\ 5 \Rightarrow & A\overline{BD} & + & ABD & = & A(\overline{BD}+BD) & = & A & & & \\
 & & & & & & & \Downarrow & & & \\
 & & & & & & & 8 & & & \\
il\ reste \Rightarrow & S & = & AC & + & B & + & A\overline{C} & + & AB & + & A \\
 & & & \Downarrow & & \Downarrow & & \Downarrow & & \Downarrow & & \Downarrow \\
 & & & 1 & & 3 & & 6 & & 7 & & 8 \\
mais\ 1+6 \Rightarrow & AC & + & A\overline{C} & = & A(C+\overline{C}) & = & A & & & & \\
il\ reste \Rightarrow & S & = & A & + & B & + & AB & & & & \\
 & & & \Downarrow & & \Downarrow & & \Downarrow & & & & \\
 & & & 8 & & 3 & & 7 & & & & \\
mais\ encore\ 8+7 \Rightarrow & A & + & AB & = & A(1+B) & = & A & & & & \\
 & & & & & & & & & & & \\
donc\ finalement \Rightarrow & S & = & A & + & B & & & & & &
\end{array}$$

De façon générale de très nombreuses fonctions logiques sont susceptibles d'être simplifiées, mais la forme la plus compacte n'est pas toujours trouvée immédiatement car la voie de simplification algébrique la plus rapide n'est pas évidente. Dans le cas de quatre et cinq variables, les diagrammes de *Karnaugh* permettent d'effectuer cette simplification *automatiquement*.

1.8.2 *Méthode Graphique : Diagramme de Karnaugh*

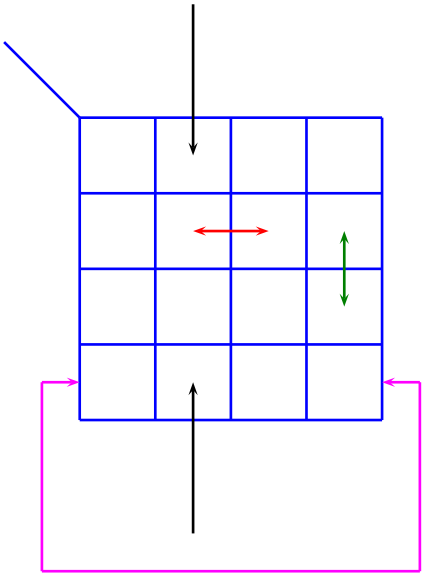

Figure 1.17: *Cases adjacentes.*

La méthode de simplification de *Karnaugh* consiste à mettre à profit la relation, $X.A+X.\overline{A}=X$ pour donner une expression simplifiée de la fonction à l'aide des opérateurs fondamentaux. Nous recherchons les termes ne différant que par un seul facteur qui apparaît complémentaire dans le premier et non complémentaire dans le second. Ces deux termes s'ils font partie d'une somme canonique correspondent dans le diagramme de *Karnaugh* à deux 1 placés dans des cases adjacentes. Nous formons alors ce que nous appelons une boucle d'ordre 2. Il faut observer que deux cases doivent être considérées comme adjacentes si nous le passons de l'une à l'autre en ne modifiant qu'une seule variable, ce qui est le cas de deux cases placées réellement côte à côte, mais aussi aux deux extrémités d'une ligne. La simplification se fera en regroupant les 1 en boucles d'ordre $2, 4, 8, ..., 2^n$, comme l'illustre la Figure 1.17.

Soit la fonction logique donnée par l'équation 1.8.2

$$\begin{array}{ccccccccccc} S & = & AB\overline{C}D & + & ABCD & + & \overline{A}.B.\overline{C}.\overline{D} & + & \overline{A}BC\overline{D} \\ Case & & \Downarrow & & \Downarrow & & \Downarrow & & \Downarrow \\ & & 13 & & 15 & & 4 & & 6 \\ S & = & ABD & + & \overline{A}B\overline{D} & & & & \end{array} \qquad (1.8.2)$$

Boucles d'Ordre Deux

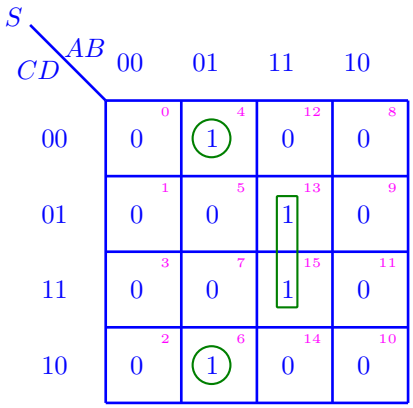

Figure 1.18: *Diagramme de Karnaugh pour la fonction logique de l'équation 1.8.3.*

Nous pouvons former deux boucles avec les cases adjacentes $13-15$ et $4-6$. La boucle $13-15$ donne un terme ABD. En effet, la variable c qui change en passant d'une case à l'autre s'élimine, comme l'illustre l'équation 1.8.3.

$$AB\overline{C}D \quad + \quad ABCD \quad = \quad ABD(C+\overline{C}) \quad = \quad ABD \tag{1.8.3}$$

La boucle $4-6$ donne $\overline{A}B\overline{D}$, donc :

$$S = \overline{A}B\overline{D} + ABD \text{ ou encore } S = B(\overline{A}.\overline{D} + AD) = B(\overline{A \oplus D})$$

Les boucles d'ordre deux font disparaître une variable dans les *mintermes* de la fonction. Cette variable est celle qui varie dans ces boucles, par conséquent les variables restantes sont celles qui sont constantes dans ces boucles. Si nous voulons écrire et simplifier S, tous les 1 du tableau doivent être groupés en boucles (avec des boucles comptant le plus grand nombre de termes possibles), les boucles pouvant éventuellement se recouper, du fait de la propriété *Idempotence* ou si ce n'est pas possible, compter individuellement.

Boucles Imbriquées

Soit la fonction logique de l'équation 1.8.4.

$$\begin{array}{cccccccc} S & = & AB\overline{C}D & + & ABCD & + & \overline{A}BCD & \\ & & \Downarrow & & \Downarrow & & \Downarrow & \\ Case & & 13 & & 15 & & 7 & \end{array} \tag{1.8.4}$$

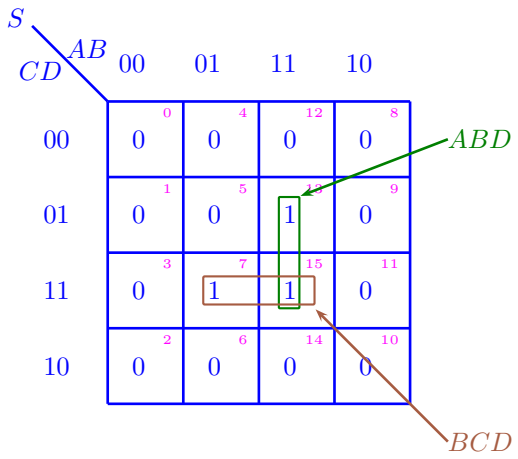

Figure 1.19: *Diagramme de Karnaugh pour la fonction logique de l'équation 1.8.4.*

Deux boucles sont possibles $13-15$ ou $7-15$, comme l'illustre la Figure 1.19, elles ont la case 15 en commun mais nous pouvons appliquer la règle précédente comme si ces boucles étaient disjointes. En effet S ne change pas si nous dédoublons un de ses termes.

Boucles d'Ordre Quatre

Supposons que deux boucles d'ordre 2 soient adjacentes, par exemple $5-7$ et $13-15$ de l'équation 1.8.5. Les deux résultats peuvent de nouveau se combiner et C disparaît. Donc, le résultat est donné par l'équation 1.8.6.

$$\begin{array}{ccccccccc} S & = & \overline{A}B\overline{C}D & + & AB\overline{C}D & + & \overline{A}BCD & + & ABCD \\ & & \Downarrow & & \Downarrow & & \Downarrow & & \Downarrow \\ Case & & 5 & & 13 & & 7 & & 15 \\ S & = & B\overline{C}D & + & BCD & & & & \end{array} \tag{1.8.5}$$

$$S = BD(C + \overline{C}) = BD \tag{1.8.6}$$

Les quatre cases ainsi groupées forment une boucle d'ordre quatre, comme l'illustre la Figure 1.20.

$$S = \overline{A}B\overline{C}D + AB\overline{C}D + \overline{A}BCD + ABCD \tag{1.8.7}$$

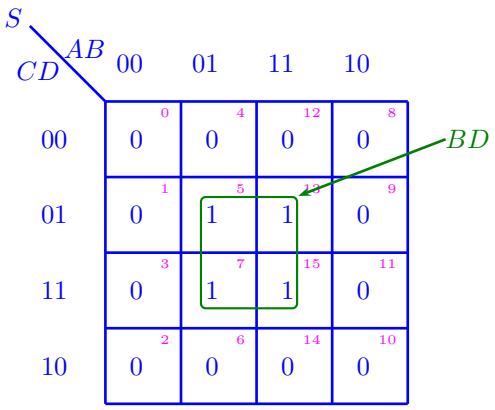

Figure 1.20: *Diagramme de Karnaugh de l'équation 1.8.7.*

Exemple 1.8.2 *Une boucle d'ordre quatre n'est pas forcément carrée, comme le montre la Figure 1.21.*

$$S = \overline{A}.B.\overline{C}.\overline{D} + \overline{A}.B.\overline{C}.D + \overline{A}.B.C.D + \overline{A}.B.C.\overline{D} \tag{1.8.8}$$

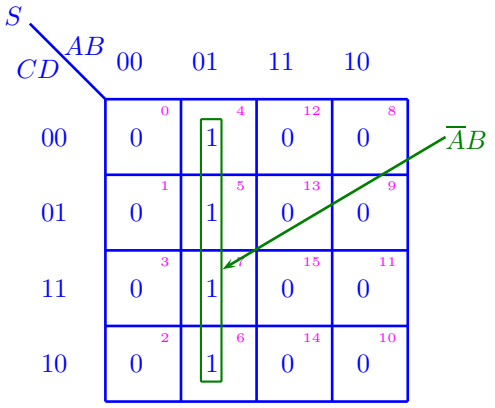

Figure 1.21: *Diagramme de Karnaugh de l'équation 1.8.8*

Exemple 1.8.3 *Elle peut être écartelée entre les deux bords ou même entre les quatre coins, comme le montre la Figure 1.22.*

$$S = \overline{A}.\overline{B}.\overline{C}.\overline{D} + A.\overline{B}.\overline{C}.\overline{D} + \overline{A}.\overline{B}.C.\overline{D} + A.\overline{B}.C.\overline{D} \tag{1.8.9}$$

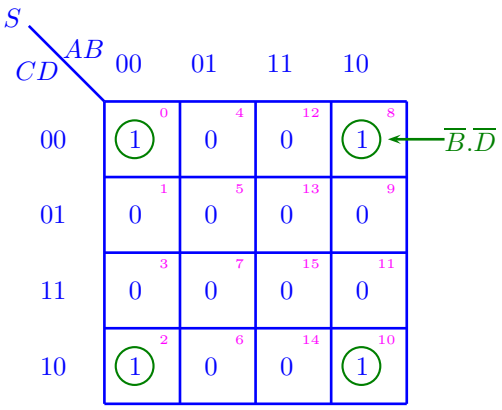

Figure 1.22: *Diagramme de Karnaugh de l'équation 1.8.9.*

Exemple 1.8.4 *Elle peut être en partie commune avec une boucle du deuxième ordre, comme le montre la Figure 1.23.*

$$S = \overline{A}.B.\overline{C}.\overline{D} + \overline{A}.B.\overline{C}.D + A.B.\overline{C}.\overline{D} + A.B.\overline{C}.D + A.\overline{B}.\overline{C}.D \tag{1.8.10}$$

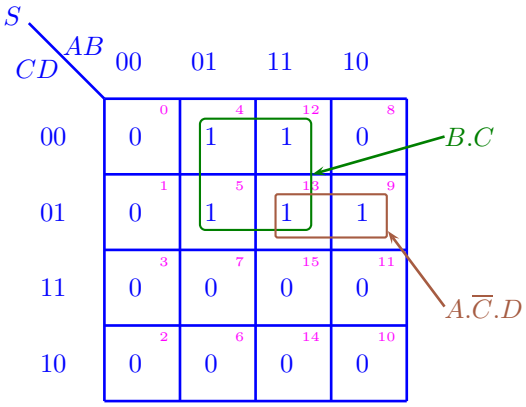

Figure 1.23: *Diagramme de Karnaugh de l'équation 1.8.10.*

Le résultat est donné par l'équation 1.8.11.

$$S = \overline{A}.\overline{B}.\overline{C}.D + A.B.\overline{C}.D + \overline{A}.B.\overline{C}.D + A.B.\overline{C}.D + A.\overline{B}.\overline{C}.D = A.\overline{C}.D + B.C \tag{1.8.11}$$

Nous pouvons observer que les boucles d'ordre quatre font disparaître deux variables dans les mintermes.

Boucles d'Ordre Huit

Si deux boucles d'ordre quatre sont adjacentes, nous pouvons former une boucle d'ordre huit pour laquelle trois variables disparaissent.

Exemple 1.8.5 *Les deux boucles d'ordre quatre :* $(0-1-2-3)$ *et* $(8-9-10-11)$ *donnent la relation suivante :* $\overline{A}\overline{B}+A\overline{B}=B$*, comme le montre la Figure 1.24.*

$$S=\overline{A}.\overline{B}.\overline{C}.\overline{D}+\overline{A}.\overline{B}.\overline{C}.D+\overline{A}.\overline{B}.C.\overline{D}+\overline{A}.\overline{B}.C.D+A.\overline{B}.\overline{C}.\overline{D}+A.\overline{B}.\overline{C}.D+A.\overline{B}.C.\overline{D}+A.\overline{B}.C.D \tag{1.8.12}$$

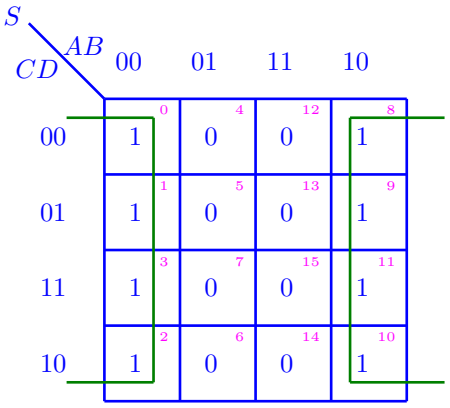

Figure 1.24: *Diagramme de Karnaugh de l'équation 1.8.12.*

Elles sont adjacentes et forment une boucle d'ordre huit où seule la variable B *est conservée. Les boucles d'ordre huit font disparaître trois variables dans les* mintermes.

Boucles d'Ordre 2^n

Les boucles d'ordre 2^n regroupent 2^n variables et font disparaître n variables dans les *mintermes* de la fonction. De façon générale, nous avons intérêt à effectuer les plus grands regroupements possibles i.e., boucles d'ordre le plus élevé, pour simplifier au maximum la fonction.

La Valeur Booléenne X ***ou*** $\emptyset$

Il existe des cas où toutes les combinaisons possibles des n variables ne sont pas utilisées, c'est le cas par exemple des quatre variables constituant une tétraèdres en code *DCB* (*Décimal Codé Binaire ou encore Binaire pur*), les six pseudo tétraèdres $(10$ à $15)$ sont exclus, comme l'illustre le Table 1.9.

Chiffre de 0 à 9	**Code DCB ⇒ ABCD**
0	**0000**
1	**0001**
2	**0010**
3	**0011**
4	**0100**
5	**0101**
6	**0110**
7	**0111**
8	**1000**
9	**1001**

Table 1.9: *Décimal Code Binaire (BCD).*

Dans ces conditions pour simplifier une fonction booléenne de quatre variables dont le champ de variation est limité, une valeur quelconque peut être donnée à la fonction dans les cases interdites de façon à constituer des boucles d'ordre le plus élevé possible sur le diagramme de *Karnaugh*. Le signe X (ou $\emptyset$) étant utilisé pour indiquer qu'un 1 aussi bien qu'un 0 convient. Nous parlons souvent de fonctions $\emptyset$ *booléennes*.

Soit par exemple à commander un voyant qui doit être allumé lorsque les chiffres 4 ou 5 apparaissent et seulement dans ces cas. Les chiffres sont codés en *DCB* sur quatre fils A, B, C et D. La fonction booléenne S à créer ne doit valoir 1 que lorsque les configurations 4 (0100) ou 5 (0101) apparaissent. En toute rigueur, $S = \overline{A}.B.\overline{C}.\overline{D} + \overline{A}.B.\overline{C}.D$ correspondant au diagramme de *Karnaugh* de la Figure 1.25 sur lequel apparaît une boucle d'ordre 2 amenant ainsi la simplification $S = \overline{A}B\overline{C}$.

Mais les cases marquées d'une croix (X ou $\emptyset$) correspondent à des situations interdites qui ne se présenteront jamais sur les quatre fils (pseudo tétraèdres) et peuvent être choisies comme nous voulons, par exemple X est équivalent à sans effet (*Don't Care*). Les X utilisés dans les regroupements sont mis à 1, les X non utilisés sont mis à 0.

Exemple 1.8.6 *Nous pouvons placer des* 1 *dans les deux cases correspondant aux états* 0001 *valeur* 1 *et* 0101 *valeur* 5 *de façon à former une boucle d'ordre quatre, comme le montre la Figure 1.25.*

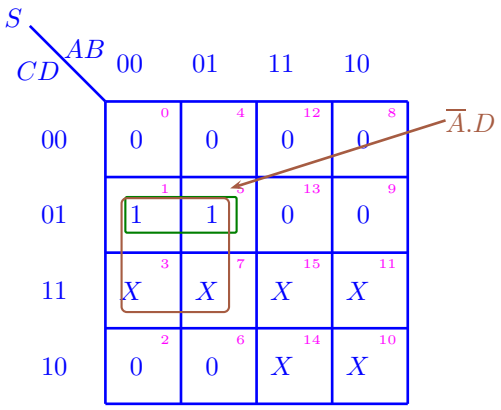

Figure 1.25: *Diagramme de Karnaugh de l'équation 1.8.13.*

$$S = \overline{A}.\overline{B}.\overline{C}.D + \overline{A}.B.\overline{C}.D \tag{1.8.13}$$

Amenant ainsi une simplification optimale de S. Donc, $S = \overline{A}D$. La lampe S s'allume dans les états 1 et 5 demandés et aussi pour 3 et 7 ce qui n'a pas d'importance puisque ces deux derniers états ne se présenteront jamais.

Remarque 1.8.1

$\Rightarrow 1$. *Ne jamais regrouper uniquement des X ensemble. Ils ne simplifient pas la fonction mais la compliquent en ajoutant un tenue inutile à la fonction.*

$\Rightarrow 2$. *Chaque X a une valeur indépendamment des autres X.*

$\Rightarrow 3$. *Ne jamais affecter des valeurs différentes à un même X , lors de différents regroupements le faisant intervenir.*

1.8.3 *Complémentaire sur la Simplification*

Si le tableau de *Karnaugh* comporte plus de 0 que de 1, nous avons alors intérêt à regrouper les 0 et non les 1 pour exprimer une fonction *booléenn*e en complément plutôt que la propre fonction *booléenne*.

De même pour une simplification algébrique, la simplification peut se faire plus simplement sur une fonction *booléenne* en complément plutôt que la propre fonction *booléenne*. Lorsque S est simplifiée au maximum, nous obtiendrons alors simplement une fonction *booléenne* par l'inversion de la fonction *booléenne* en complément.

1.9 *Conclusion*

Nous pouvons conclure que le système logique peut être un système combinatoire i.e., il n'a pas de boucle et que le système séquentiel est bouclé. Une description formelle à partir de l'algèbre

booléenne a été démontrée. Les axiomes, théorèmes et principe de dualité permettent de faire la minimisation de la fonction logique. La méthode algébrique est plus complexe en fonction de la quantité de variables de la fonction. Nous avons également constaté que le diagramme de *Karnaugh* devient de manipulation difficile à partir de six variables. Donc, la question qui reste encore ouverte est la suivante : - existe-t-il des méthodes plus simples qui permettent de faire la minimisation des fonctions *booléennes* avec plus de six variables ?

Chapitre 2

Circuits Logiques

2.1 *Introduction*

Nous étudierons les différents aspects des circuits logiques i.e., la logique positive et la négative, les caractéristiques fondamentales : niveaux logiques, temps de propagation, tension d'alimentation, entrance, sortance, et bien évidemment les différentes familles logiques.

2.1.1 *Logique Positive et Négative*

A toute grandeur physique ayant seulement deux valeurs possibles nous pouvons associer une variable *booléenne*. En électronique, les grandeurs considérées sont essentiellement le courant et la tension. Par exemple, la tension collecteur d'un transistor *NPN* alimenté sous 5 Volt (cas de la famille logique Transistor Transistor - *TTL*) et fonctionnant en régime de commutation (*régime de fonctionnement en logique*) peut valoir les valeurs d'opérations données par l'équation 2.1.1.

$$\begin{cases} V_{CC} = 0\,Volt & si \quad T\ est\ saturé \\ V_{CC} = +5\,Volt & si \quad T\ est\ bloqué \end{cases} \tag{2.1.1}$$

Nous pouvons par convention admettre que la variable booléenne C associée à la tension V_{CC}, vaut 1 si $V_{CC} = +5$ Volt, 0 si $V_{CC} = 0$ Volt. La valeur 1 est associée à la valeur la plus élevée de V_{CC} ; nous disons alors que nous avons défini une logique positive (équation 2.1.2).

$$\begin{cases} C = 0 & pour \quad V_{CC} = 0\,Volt \\ C = 1 & pour \quad V_{CC} = +5\,Volt \end{cases} \tag{2.1.2}$$

Le contraire, bien que moins courant, est également possible. la logique est qualifiée alors de négative (équation 2.1.3).

$$\begin{cases} C = 1 & pour \quad V_{CC} = 0\,Volt \\ C = 0 & pour \quad V_{CC} = +5\,Volt \end{cases} \tag{2.1.3}$$

Remarque 2.1.1

Dans ce qui suit, nous travaillerons toujours en logique positive.

2.1.2 *Symboles Logiques*

Une fonction (ou opérateur) logique élémentaire ou fondamental est matérialisée par un circuit logique appelé porte logique.

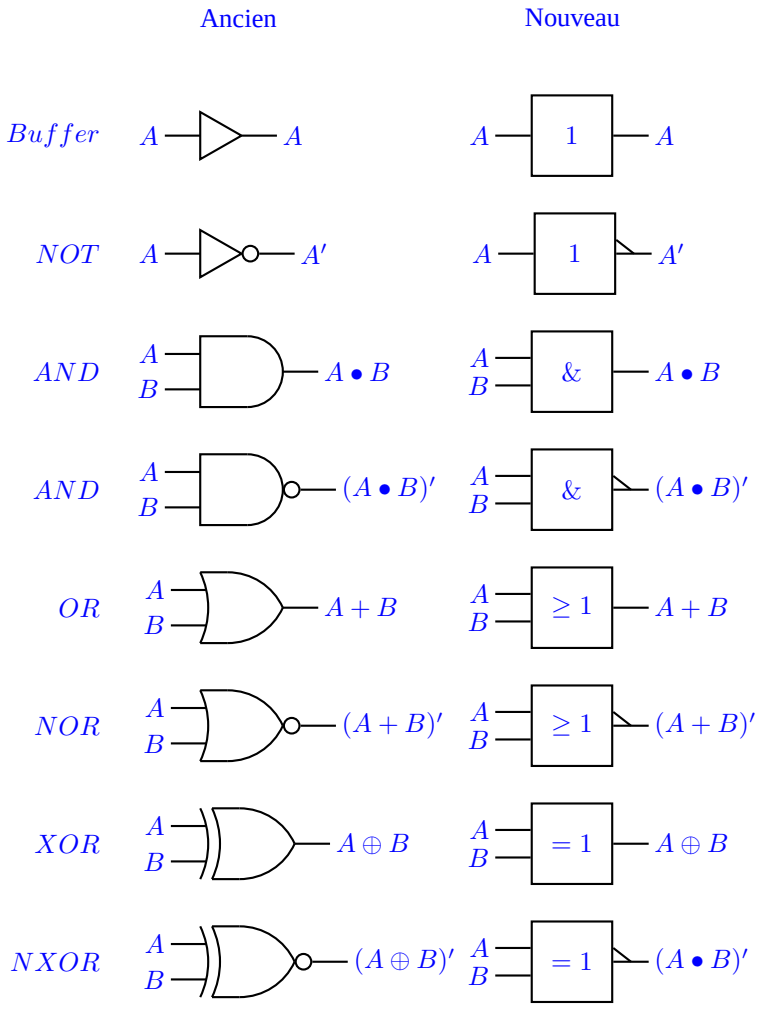

Figure 2.1: *Symboles logiques fondamentaux.*

Remarque 2.1.2

⇒ 1. *Un* buffer *ou* driver *rehausse au niveau haut une tension de niveau haut diminuée.*

⇒ 2. *Le petit cercle représente le complémentaire.*

⇒ 3. *La flèche représente le complémentaire en plus du sens de l'information.*

⇒ 4. *En l'absence de flèche, le signal circule implicitement de la gauche vers la droite.*

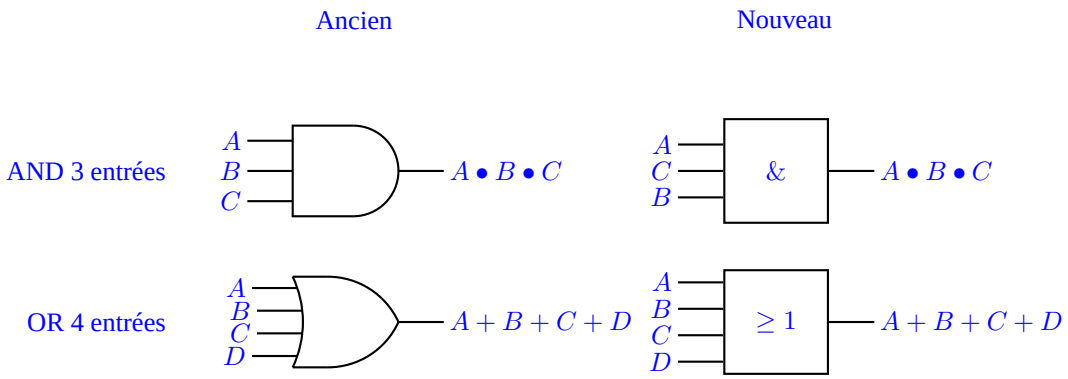

Figure 2.2: *Symboles logiques de portes élémentaires à 3 et 4 entrées.*

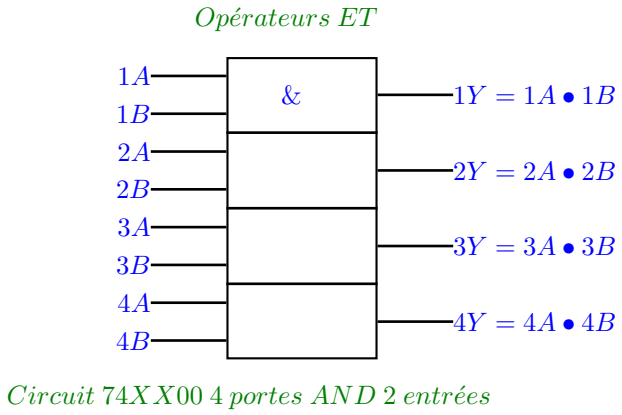

Figure 2.3: *Symbole logique de l'opérateur ET de multiples d'une porte élémentaire à 2 entrées.*

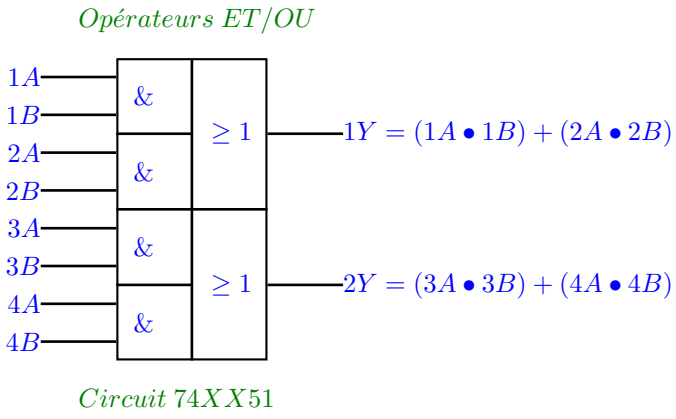

Figure 2.4: *Symbole logique de l'opérateur ET/OU de multiples d'une porte élémentaire à 2 entrées.*

Remarque 2.1.3

$\Rightarrow$ 1. *Le petit cercle représente le complémentaire.*

$\Rightarrow$ 2. *La flèche représente le complémentaire en plus du sens de l'information.*

$\Rightarrow$ 3. *En l'absence de flèche, le signal circule implicitement de la gauche vers la droite.*

Exemple 2.1.1 *La représentation de la fonction logique* $f(A,B) = \overline{\overline{A}B}$ *est donnée par la Figure 2.5.*

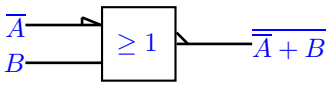

Figure 2.5: *Représentation de la fonction logique* $f(A,B) = \overline{\overline{A}+B}$.

2.2 *Caractéristiques Fondamentales*

2.2.1 *Définition des Niveaux Logiques*

Considérons le cas simple d'un inverseur en logique 5 Volt. C'est un circuit dont la sortie est à 5 Volt si l'entrée est à zéro et réciproquement, ce qui ne définit sur la caractéristique de transfert que les deux points A et B de la Figure 2.6. En réalité par suite de l'influence des autres circuits qui lui sont connectés, les niveaux d'entrée et de sortie d'un tel inverseur n'ont jamais ces valeurs idéales et il y a lieu de considérer la courbe de transfert complète de la Figure 2.6.

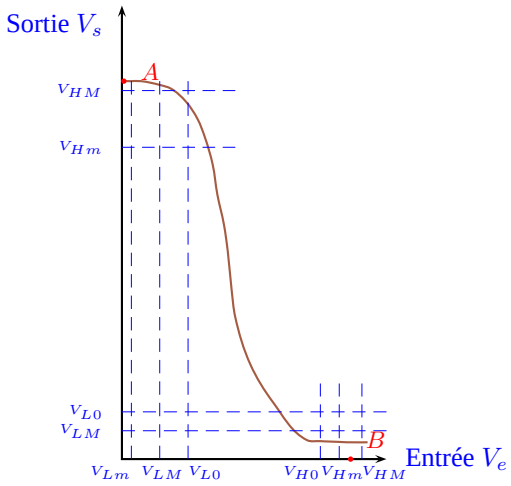

Figure 2.6: *La coubre de transfert d'immunité au bruit.*

En régime normal, la tension d'entrée au niveau zéro se trouve entre V_{Lm} et V_{LM}, la tension de sortie étant alors au niveau haut (1) entre V_{Hm} et V_{HM}. Si une impulsion parasite vient se superposer à la tension d'entrée, la tension de sortie restera compatible avec le niveau 1 si le niveau V_{L0} n'est pas dépassé. Nous définissons $M_1 = V_{L0} - V_{LM}$ comme la marge de bruit admissible à l'entrée au niveau bas. De même, si l'entrée est au niveau 1, une impulsion parasite ne doit pas faire tomber V_e en dessous de V_{H0}, et $M_0 = V_{Hm} - V_{H0}$ comme la marge de bruit admissible au niveau haut. Ces deux marges définissent ce que nous appellerons *immunité au bruit du circuit*.

2.2.2 *Temps de Propagation*

Si le niveau d'entrée d'un circuit change brutalement, son niveau de sortie ne varie qu'avec un certain retard appelé *temps de propagation* t_p.

La Figure 2.7 illustre le cas d'un inverseur. Les temps de propagation sont couramment de 20 ns et peuvent, pour les circuits les plus rapides, être inférieurs à 1 ns. La fréquence maximale d'utilisation au-delà de laquelle les signaux ne sont plus restitués par les circuits (limite haute de la *Bande Passante*) est liée au temps de propagation.

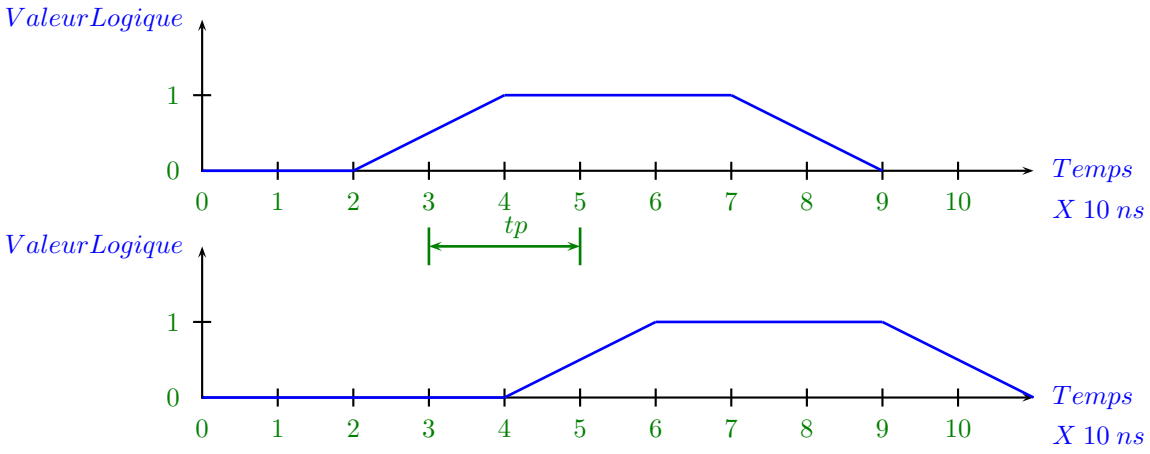

Figure 2.7: *Le temps de propagation de l'information.*

2.2.3 *Tension d'Alimentation*

Pour la famille logique transistor transistor (*TTL*), la tolérance sur les niveaux de tension est donnée par la Table 2.1. La puissance moyenne absorbée par chaque porte est de l'ordre de 10 mW et le courant moyen par porte est de l'ordre de milliampère. Dans la littérature technique, L exprime la valeur bas (Low), H la valeur haut ($High$), I est l'entrée ($Input$) et O la sortie ($Output$).

Tension d'alimentation	*Sigle*	$V_{cc} \pm 0.5$ *Volts*
Tension maximum d'entrée pour un niveau bas	V_{IL}	0.8 *Volt*
Tension minimum d'entrée pour un niveau haut	V_{IH}	2 *Volt*
Tension maximum de sortie pour un niveau bas	V_{OL}	0.4 *Volt*
Tension minimum de sortie pour un niveau haut	V_{OH}	2.4 *Volt*

Table 2.1: *La tolérance sur les niveaux de tension TTL.*

Pour la famille *CMOS* la tension d'alimentation est différente de la famille TTL. Donc, elle peut être de 3 à 18 Volt (*série MC14XXX*). Par contre, les nouvelles générations plus performantes n'autorisent que 2 à 6 Volts. La puissance consommée de la famille *CMOS* est inférieure à la famille *TTL* et elle est de l'ordre de 0.1 mW, puis le courant très faible est d'ordre inférieur à 1 mA. La tolérance sur les niveaux logiques sont du même ordre qu'en *TTL*.

2.2.4 *Entrance (Fan In) et Sortance (Fan Out)*

La source qui impose à l'entrée d'un circuit logique un niveau, 0 ou 1, doit fournir un certain courant. Ce courant est différent suivant l'état. Il peut être selon le cas, maximal pour l'état 1 ou l'état 0. Dans une même famille de circuits, ces valeurs sont des constantes, sauf pour certains circuits particuliers dont les exigences peuvent être plus importantes.

Entrance

Nous appelons entrance (*fan in*) d'un circuit la valeur du courant de commande d'une entrée de ce circuit exprimée en une unité qui est le courant de commande typique de la famille (appelé *charge*).

Exemple 2.2.1 *Un circuit ayant une entrance de* 2 *consomme (ou fournit) un courant d'entrée double de celui d'un circuit ordinaire de la même famille. Le courant unité correspond à ce que nous appelons une* charge.

Sortance

Il est clair qu'un circuit logique ne peut garantir sa tension de sortie que si le nombre de charges qui lui sont connectées est limité i.e., un niveau logique 1 de sortie chute à 0 si le nombre de charges est trop élevé, comme l'illustre la Figure 2.8.

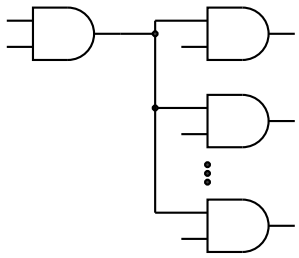

Figure 2.8: *La sortie limite les entrées logiques.*

Un circuit logique peut d'autre part, sans que le niveau logique dépasse les limites permises (fournir un courant maximal $I_{S_{max}}$). Le rapport entre ce courant maximal et celui correspondant à une charge est appelé *sortance* du circuit (ou *fan out*). Le facteur pyramidal de sortie i.e., c'est le nombre maximal de charges que peut commander une sortie (à entrée unitaire) en garantissant les niveaux logiques.

Exemple 2.2.2 *Soit un circuit ayant une* sortance *de* 10*. Nous pouvons connecter* 10 *charges tout en garantissant les niveaux de sortie de cette porte.*

Le sens des courants est également très important i.e., une famille logique dont les circuits doivent être pilotés par un courant entrant est dite à injection de courant. Dans le cas contraire, nous parlons de logique à extraction de courant, comme l'illustre la Figure 2.9.

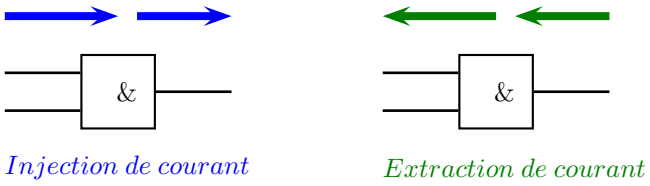

Figure 2.9: *Injection et extraction du courant.*

De façon à éviter l'action des signaux parasites, comme le fil (ou antenne) les entrées non utilisées d'un circuit à entrées multiples doivent être polarisées, soit en les reliant aux autres, soit en les connectant à la source d'alimentation ou à la masse suivant le cas. Ceci est particulièrement important dans le cas des circuits ayant de courant d'entrée très faible comme les circuits *CMOS*. Donc, une entrée ouverte a un état indéterminé qui prend en général la valeur 1 par effet d'antenne.

Circuits Câblés

Soit un circuit *ET* à 2 entrées effectuant l'opération, laquelle a comme fonction *booléenne* $f(A, B) = A.B$, comme l'illustre la Figure 2.10.

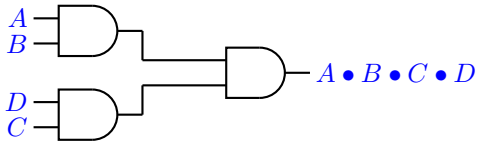

Figure 2.10: *Connexion de portes logiques pour obtenir un porte logique de quatre entrées.*

Mais, pour réaliser un circuit *ET* à 4 entrées qui réaliserait la fonction *booléenne* $f(A, B, C, D) = A.B.C.D$, nous pouvons songer à utiliser 3 circuits *ET* à 2 entrées en faisant le produit des 2 produits partiels $f(A, B) = A.B$ et $f(C, D) = C.D$, comme l'illustre la Figure 2.11.

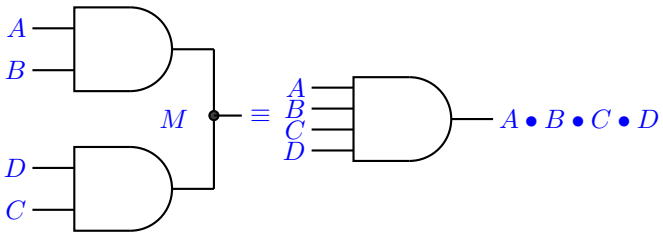

Figure 2.11: *L'ET câble.*

Dans certains cas, nous pouvons associer plus directement les sorties des 2 circuits *ET* sans dommage pour les circuits. Si ceci est possible, les circuits sont qualifiés d'expansif, comme l'illustre la Figure 2.12.

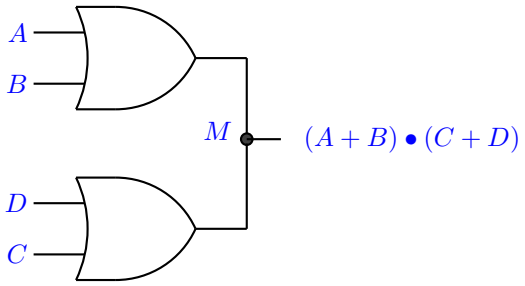

Figure 2.12: *L'OU câble.*

La jonction au point M est appelée *circuit ET câblé* et la jonction pour l'*circuit OU câblé* peuvent être utilisées dans d'autre cas.

2.3 *Les Familles Logiques*

Toutes les fonctions *booléennes* peuvent être construites à l'aide des trois opérateurs fondamentaux *ET*, *OU* et *complément*. Ce groupe de trois opérateurs forme ce que nous appelons un système logique complet. La matérialisation des fonctions logiques nécessite donc de pouvoir réaliser des systèmes physiques remplissant ces trois fonctions. Un système logique complet, permettant la construction de toute fonction, peut cependant être réalisé en utilisant un nombre plus faible de structures de base. Par exemple, le groupe des deux fonctions *ET* et complément constitue un système logique complet. En effet, la fonction *OU* peut être reconstituée à partir de ces deux fonctions seulement comme le montrent les théorèmes de *De Morgan*, i.e., $A + B = \overline{\overline{A + B}} = \overline{A}.\overline{B}$, comme l'illustre la Figure 2.13.

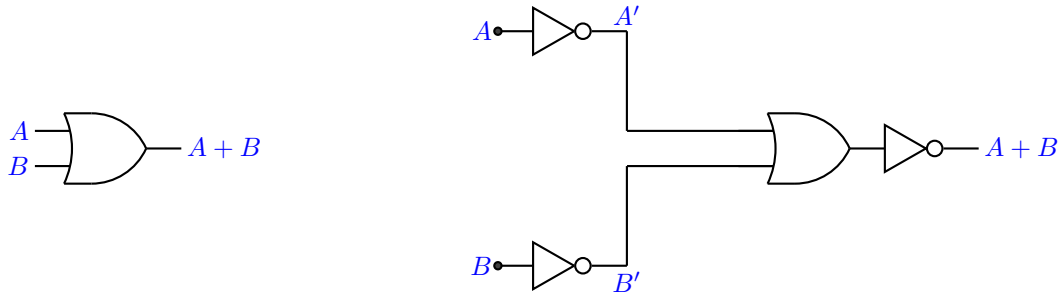

Figure 2.13: *Réalisation à partir du théorème De Morgan.*

2.3.1 *Circuits Logiques à Diodes*

Soient, deux diodes et une résistance connectées comme le montre la Figure 2.14. Admettons d'abord que les diodes sont parfaites, de résistance nulle dans le sens passant i.e., $R_d = 0$ pour $V > 0$. Si l'une ou l'autre des entrées A ou B est reliée à la masse ($V = 0$), la sortie $V_S = 0$. La sortie V_S n'est au potentiel haut i.e., $f(A, B) = 1$ en *logique positive* que si V_A et V_B sont au potentiel haut. Nous avons réalisé le produit logique $f(A, B) = A.B$, comme l'illustre la Table 2.2.

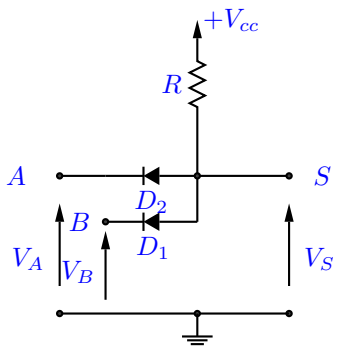

Figure 2.14: *Le circuit de la porte logique ET à diodes.*

$\mathbf{V_A}$	$\mathbf{V_B}$	$\mathbf{V_S}$
0	$+\mathbf{V_{cc}}$	*D_1 passante, D_2 bloquée et V_S à la masse.*
$+\mathbf{V_{cc}}$	**0**	*D_1 bloquée, D_2 passante et V_S à la masse.*
0	**0**	*D_1 passante, D_2 passante et V_S à la masse.*
$+\mathbf{V_{cc}}$	$+\mathbf{V_{cc}}$	*D_1 bloquée, D_2 bloquée et V_S à $+V_{cc}$.*

Table 2.2: *Conditions d'opérations de la porte logique ET à diodes.*

Le circuit de la Figure 2.15, la sortie ne vaut $+V_S$ que si A ou B valent 1, i.e., la somme logique $f(A, B) = A + B$.

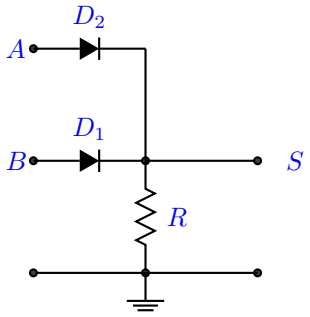

Figure 2.15: *Le circuit de la porte logique OU à diodes.*

Exemple 2.3.1 *Les portes à diodes sont aussi expansives, comme l'illustre la Figure 2.16.*

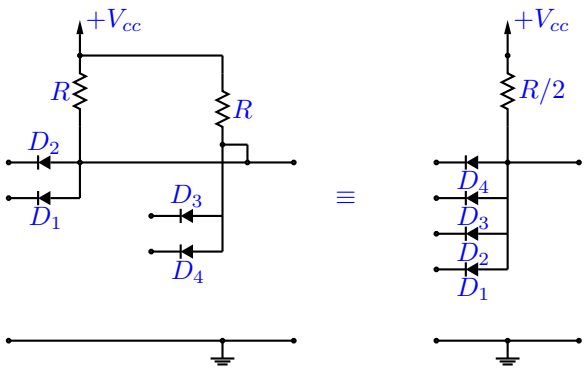

Figure 2.16: *Le circuit de la porte logique expansive à diodes* .

Compatibilité des Portes ET et OU à Diodes

La mise en série de portes de type différent pose un certain nombre de difficultés liées au fait que les portes *ET* sont à extraction de courant alors que les portes *OU* sont à injection de courant. Il faut leur adjoindre un élément actif du type transistor qui peut par contre à lui seul constituer un système complet comme dans la famille *RTL*.

2.3.2 *La Famille Résistance Transistor Logique*

Le transistor de la famille *Résistance Transistor Logique - RTL* permet très simplement d'obtenir le complément d'une variable logique, comme l'illustre la Figure 2.17.

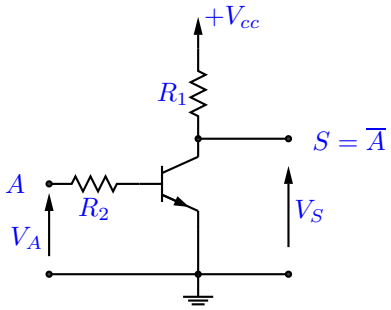

Figure 2.17: *Le principe de fonctionnement de porte logique complémentaire RTL.*

Si $V_A = 0$ i.e., $A = 0$ le transistor est bloqué et $V_S = +V_{CC}$ ou $S = 1$.

Si $V_A = V_{CC}$, sous réserve que la condition de saturation i.e., $R_2 > \frac{R_1}{\beta}$ soit satisfaite, le transistor est saturé i.e., $V_S = 0$ ou $S = 0$.

Nous avons donc réalisé le complément $S = \overline{A}$. Un transistor associé à des résistances (d'où le nom de ce type de circuits) permet de réaliser des opérations *ET* et *OU* ou plus exactement des *ET* et *OU* complémentaires soit des *NOR* et *NAND*, comme l'illustre la Figure 2.18.

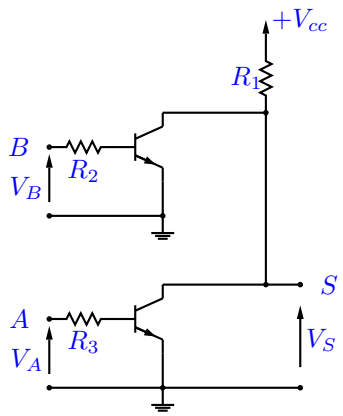

Figure 2.18: *Le principe de fonctionnement de la porte logique NOR RTL.*

Si $V_A = V_B = 0$ les deux transistors sont bloqués. Si l'une des tensions d'entrée vaut $+V_{CC}$, le transistor correspondant se sature et $V_S = 0$. D'où la table de vérité, comme l'illustre la Table 2.3, correspondant à la fonction *NOR* i.e., $S = A + B$.

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

Table 2.3: *Fonction logique NON OU (NOR).*

Pour réaliser la porte *ET*, nous pouvons utiliser deux *inverseurs* et un *NOR* conformément à l'expression $A \bullet B = \overline{\overline{A} + \overline{B}}$. Pour la porte *NAND*, on peut faire appel au montage direct de la, Figure 2.19, mais qui est peu utilisé car les entrées sont mal découplées entre elles.

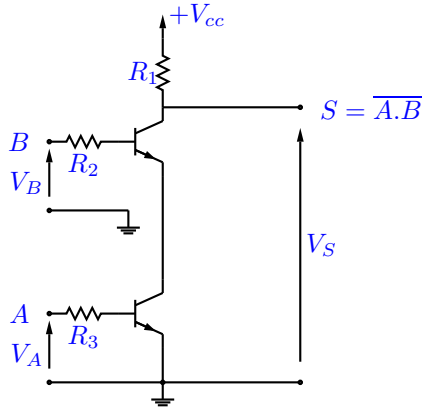

Figure 2.19: *Le principe de fonctionnement de la porte logique NAND RTL.*

La structure de base en *RTL* est la porte *NOR* qui constitue à elle seule, un système logique complet. Nous remarquons enfin que la *RTL* est une logique à injection du courant.

2.3.3 *La Famille Diode Transistor Logic*

La famille *Diode Transistor Logic - DTL* peut être considérée comme l'association d'une logique à diodes et d'un transistor *inverseur*. L'ensemble constitue alors un circuit *NAND* qui à lui seul forme un système logique complet. Le schéma de principe est présenté sur la Figure 2.20.

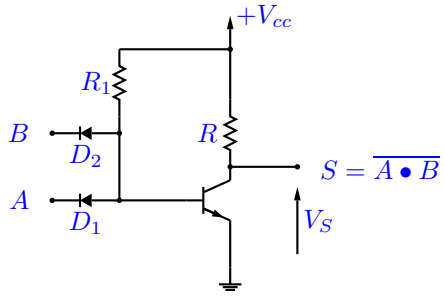

Figure 2.20: *Le principe de fonctionnement de la porte logique NAND DTL.*

Si $V_A = V_B = +V_{CC}$, les deux diodes sont bloquées et le transistor saturé par le courant de base traversant R_1 et $V_S = 0$. Si $V_A = 0$ la diode d'entrée parfaite bloque le transistor i.e., $V_S = V_{CC}$. En réalité si le point A est à la masse, l'anode de la diode correspondante est un potentiel voisin de 0.6 Volt qui est aussi le seuil de conduction du transistor. Le montage ne peut fonctionner que si le V_{BE} limite de conduction du transistor est plus élevée que la tension de conduction de la diode. Cela pourrait se faire avec des diodes au germanium associées à un transistor silicium (*solution incompatible avec l'intégration du circuit*). Une solution plus efficace consiste à utiliser des diodes remontant le seuil de conduction du transistor. Le circuit réel est représenté sur la Figure 2.21. Les deux diodes D_4 et D_5 remontent au voisinage de 1.8 Volt la tension en P nécessaire à la conduction du transistor. Alors, si $V_A = 0$ et $V_P = 0.6$ Volt le transistor est bloqué et $V_S = +V_{CC}$.

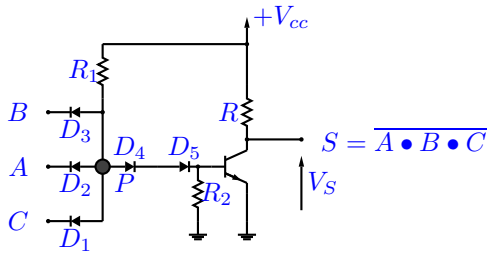

Figure 2.21: *Le principe de fonctionnement de la porte logique NAND 3 entrées DTL.*

Il n'existe pas de circuit spécifiquement *NOR* en *DTL*. La logique *DTL* est une logique à extraction de courant qui n'est donc pas compatible avec la *RTL*. Comme pour la logique à diodes les portes sont expansives, nous diminuons la résistance de charge du transistor.

2.3.4 *Transistor Transistor Logique - TTL*

Elle ne diffère de la DTL que par le remplacement du réseau de diodes d'entrée par un transistor spécial multi-émetteurs. C'est une logique qui ne se conçoit qu'en circuits intégrés, elle est de loin la plus courante actuellement sont dont la série 54/74. Comme en DTL, le circuit de base est une porte $NAND$. La Figure 2.22 représente le montage de base du circuit d'entrée, les diodes sont

remplacées par les jonctions EB des transistors. Les deux transistors d'entrée ont leurs bases et collecteurs reliés.

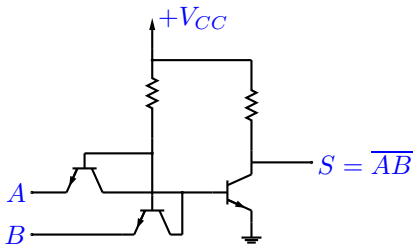

Figure 2.22: *Le schéma de base de la porte logique NAND (TTL).*

Lors de leur fabrication cette liaison peut aller jusqu'à la fusion totale conduisant à un transistor multi-émetteur qui réalise la fonction ET. D'où le circuit d'entrée de la Figure 2.23.

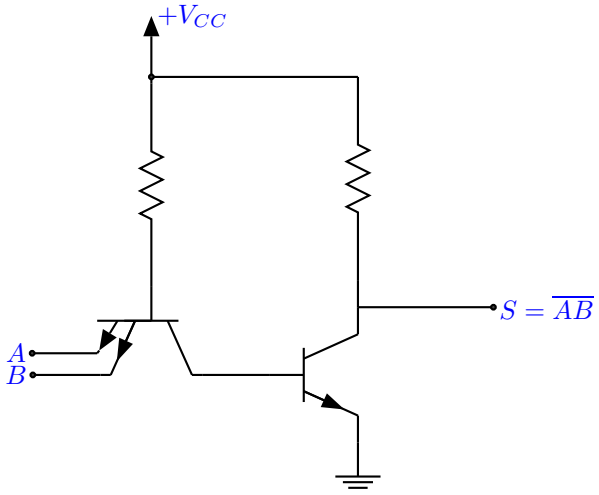

Figure 2.23: *Le principe de la porte logique NAND (TTL).*

Pour augmenter les performances, le circuit de sortie n'est pas un simple transistor. En effet un tel montage est très mal adapté pour attaquer les charges du condensateur. Considérons en effet les Figures 2.24 et 2.25.

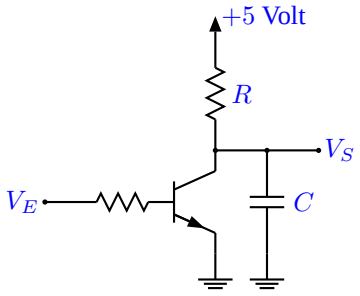

Figure 2.24: *L'effet du condensateur sur la sortie de la porte logique.*

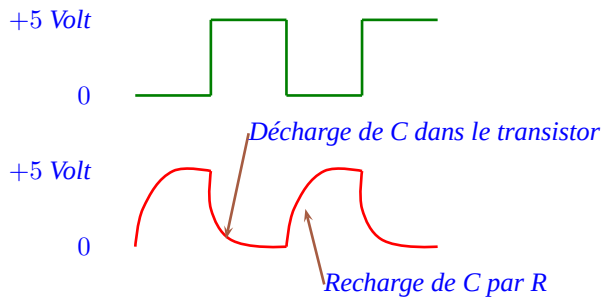

Figure 2.25: *La décharge et recharge du condensateur.*

Lorsque le niveau de sortie passe de $+V_{cc}$ à 0, le courant de décharge de C, traverse le transistor qui en régime de saturation se comporte presque comme un commutateur parfait. Lorsque le niveau de sortie passe de 0 à $+V_{cc}$ ce qui correspond au bloquage du transistor, la charge de C doit se faire grâce à un courant traversant R, donc avec une constante de temps RC appréciable. Pour diminuer le temps de montée, il faut diminuer R ce qui augmente proportionnellement la consommation du circuit. Pour augmenter la vitesse nous faisons appel à un deuxième transistor monté à la place de R qui en se saturant branche directement C à $+V_{CC}$. Le montage présente quelques analogies avec le *push-pull*, comme l'illustre les Figures 2.26 et 2.27.

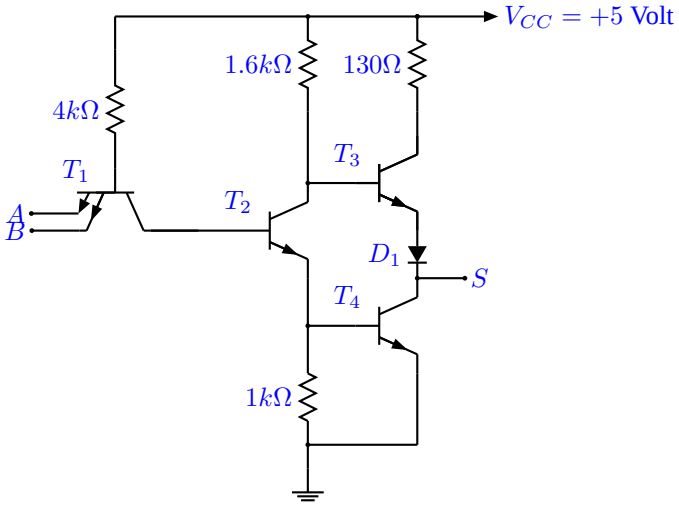

Figure 2.26: *Porte logique NAND SN7400.*

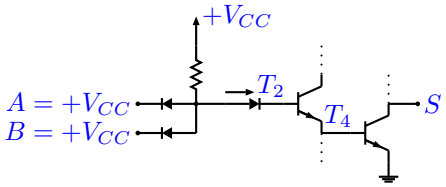

Figure 2.27: *Analyse pour les entrées $A = B = 1$.*

Si $V_A = V_B = +5$ Volt, les diodes émetteur-base de T_1 sont bloquées, par contre la diode

base-collecteur est conductrice. Un courant circule, donc T_2 et T_4 sont saturés. T_2 étant saturé, sa tension collecteur est égale à sa tension émetteur soit environ 0.6 Volt. L'autre possibilité est quand T_3 ne peut conduire à cause de D_1. Si la base est portée à environ 1.2 Volt ; il est donc bloqué. Alors $V_s = 0$ i.e., $S = 0$. Annulons l'une des tensions A ou B ou les deux. La résistance de 4 kΩ assure la saturation de T_1 ce qui amène à zéro le potentiel base de T_2 donc bloque T_2 et aussi T_4. Le transistor T_3 se trouve alors saturé grâce à la résistance de 1.6 kΩ reliant sa base à $+V_{cc}$, la sortie est alors au niveau haut. Ce circuit réalise donc bien la fonction $NAND$ i.e., $S = (AB)'$. Le montage constitué par les deux transistors de sortie T_3, T_4 est appelé *totem pole* i.e., pour chacun des 2 états logiques nous avons donc T_3 bloqué et T_4 saturé, ou l'inverse. Il permet des transitions rapides du niveau de sortie même sur charge du condensateur. Le temps de transit est couramment de 10 ns. Donc, comme la logique DTL, la logique TTL est une logique à extraction de courant.

Tolérance sur les Niveaux TTL

La Table 2.4 illustre les niveaux qui peut supporter les composants de la famille logique TTL.

Tension d'alimentation	V_{CC}	*5 Volts $\pm$0.5 Volts*
Tension maximum d'entrée pour un niveau bas	V_{IL}	*0.8 Volts*
Tension minimum d'entrée pour un niveau haut	V_{IH}	*2 Volts*
Tension maximum de sortie pour un niveau bas	V_{OL}	*0.4 Volts*
Tension minimum de sortie pour un niveau haut	V_{OH}	*2.4 Volts*

Table 2.4: *Table de tolérance sur les niveaux TTL.*

La puissance moyenne absorbée par porte est approximativement $10\ mW$ et le courant moyen par porte est de l'ordre de mA.

TTL Schottky

Dans la famille précédente les transistors travaillent en commutation i.e., qu'ils sont parfois saturés, où un transistor saturé stocke des charges dans sa base qui doivent ensuite être évacuées. Ceci limite fortement la vitesse de commutation. Pour augmenter la vitesse, il faut éviter la saturation, ceci peut se faire en plaçant une diode en parallèle sur l'espace base-collecteur, de façon à maintenir le collecteur à un potentiel très légèrement inférieur à celui de la base i.e., T ne peut pas se saturer car D conduirait, ce qui amènerait la base à $+0.15$ Volt bloquant le transistor, comme l'illustre la Figure 2.28

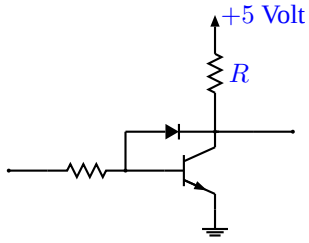

Figure 2.28: *Le principe de la famille TTL Schottky.*

En réalité le gain de vitesse n'est pas grand car la diode elle même stocke des charges. La solution est trouvée en remplaçant D par une diode *Schottky*. Une telle diode est constituée par un contact

métal semi-conducteur. Sa tension de conduction est de l'ordre de 0.4 Volt et elle est très rapide car le phénomène de stockage est très réduit. Dans un circuit TTL le transistor multi-émetteur d'entrée peut être également de type diode *Schottky* i.e., une diode *Schottky* où nous pouvons remplacer par diodes *Schottky* comme en DTL, comme l'illustre les Figures 2.29 et 2.30.

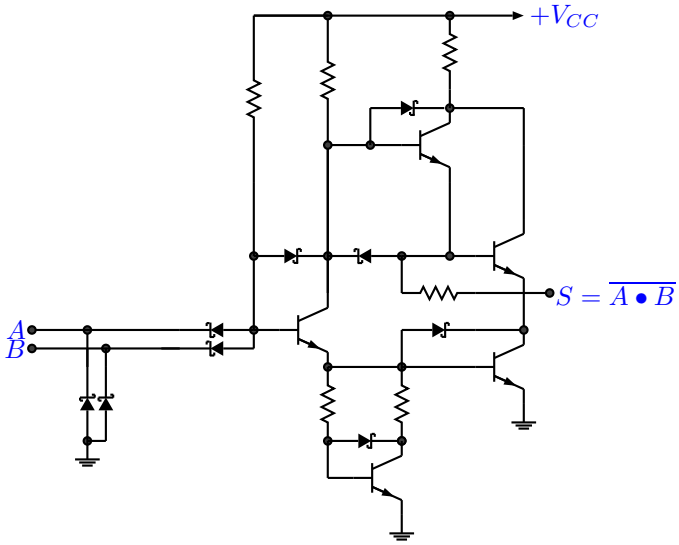

Figure 2.29: *Porte logique NAND TTL S.*

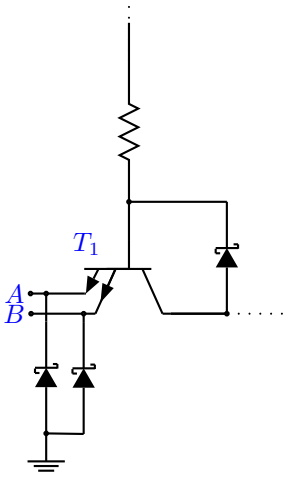

Figure 2.30: *Porte logique avec entrée multi-émetteur.*

Le gain en vitesse est important, les temps de transit étant de quelques nanosecondes seulement.

2.3.5 *Variantes du Circuit de Sortie*

La Sortie Totem-pole

La sortie totem-pole permet d'intéressantes performances en vitesse mais interdit le ET et le OU câblés, l'interconnexion directe des sorties peut en effet conduire à la destruction des circuits. Pour remédier à cet inconvénient, deux solutions ont été retenues i.e., les sorties *open collector* et *tri-state*. En sortie totem-pole, la charge est fixée par construction. Les deux transistors en alternant la commutation i.e., un est bloqué quand l'autre est saturé, donc la sortie augment la rapidité du circuit.

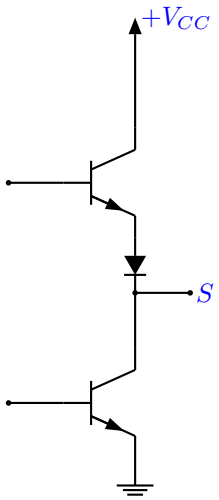

Figure 2.31: *La sortie totem-pole.*

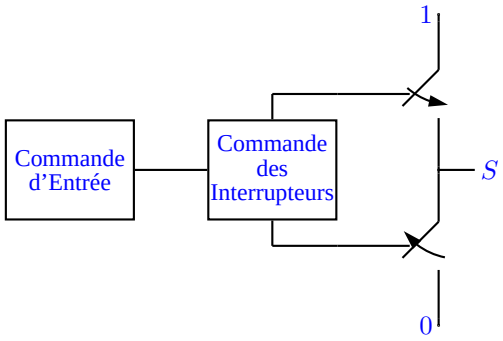

Figure 2.32: *Le schéma synoptique d'une porte logique à sortie totem-pole.*

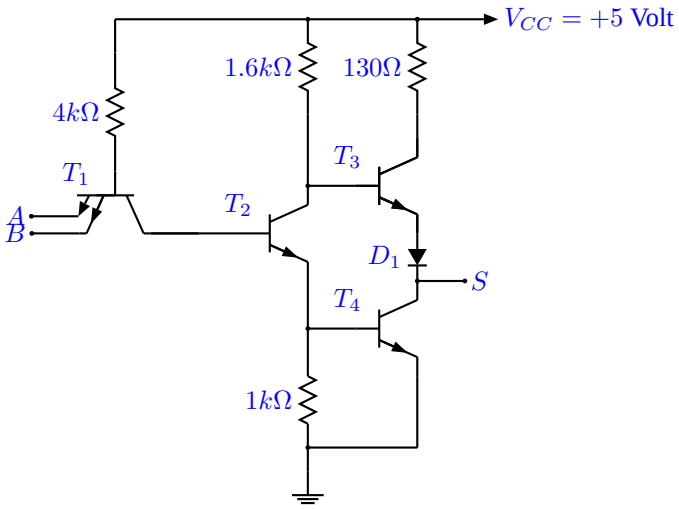

Figure 2.33: *La porte logique TTL SN7400.*

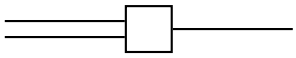

Figure 2.34: *Le symbole d'une porte logique totem-pole par défaut.*

La Sortie Open Collector

Le totem-pole est supprimé et remplacé par un seul transistor dont la résistance de collecteur n'est pas intégrée. Elle doit être mise en place par l'utilisateur i.e., en fonction de son problème. Le collecteur du transistor de sortie du circuit logique n'est pas connecté à une alimentation dans le circuit. C'est à l'utilisateur de placer la charge la mieux adaptée selon la sortance désirée. Le OU et le ET câblés deviennent ainsi possibles. De plus certains circuits sont prévus avec un transistor de sortie pouvant supporter une tension de plusieurs dizaines de volts et sont précieux comme générateurs d'impulsions de grande amplitude.

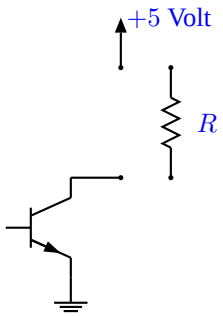

Figure 2.35: *Le schéma d'une sortie de la porte logique open collector.*

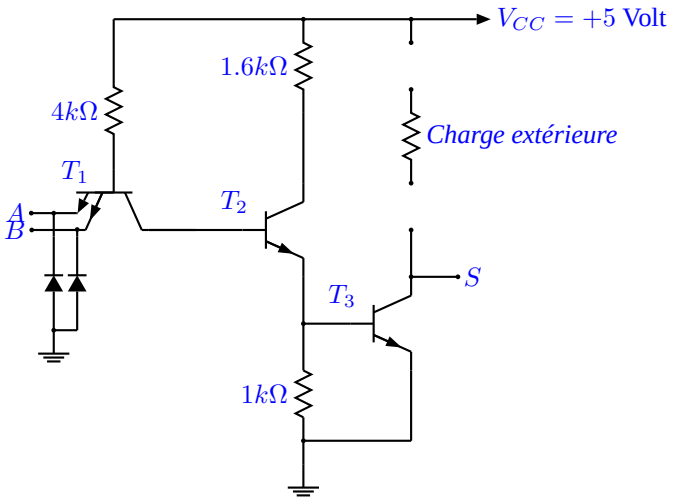

Figure 2.36: *Le schéma d'une porte logique NAND avec la sortie open collector.*

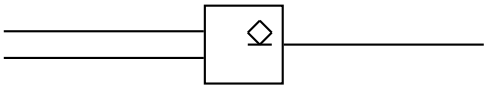

Figure 2.37: *Le symbole d'une porte logique collecteur ouvert.*

Sortie Tri-state

La charge est fixée par construction. Les deux transistors en alternant de commutation du $totem-pole$ sont désolidarisés pour donner en sortie trois états possibles i.e., les deux états logiques 0 et 1, et le troisième état nommé de haute impédance qui est obtenu lorsque les deux transistors de sortie sont bloqués. La sortie peut donc se présenter sous les trois états 0, 1 et l'état haute impédance ou circuit déconnecté.

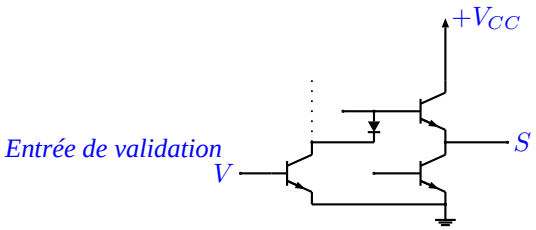

Figure 2.38: *Le sortie du circuit tri-state.*

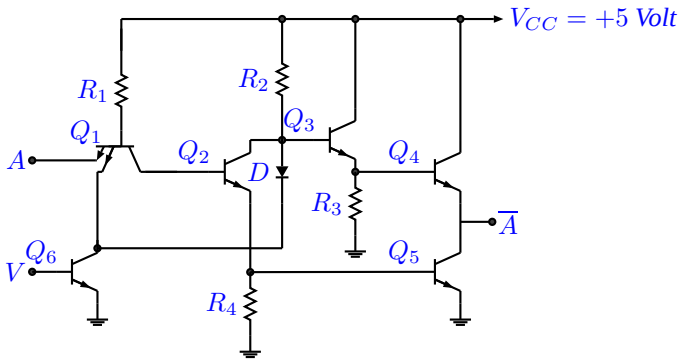

Figure 2.39: *La porte logique inverseur et la sortie tri-state.*

Si $V = 0$, Q_6 est bloqué,comme l'illustre la Figure 2.39, le système fonctionne comme un circuit TTL classique. Si $A = 0$, Q_1 conduit, la base de Q_2 est au zéro donc Q_2 est bloqué ainsi que Q_5, pendant que Q_3 est conducteur ainsi que Q_1, donc $S = A' = 1$.

Si $V = 1$, Q_6 est saturé donc Q_1 également et comme plus haut Q_2 et Q_6 sont bloqués, mais par la diode D reliée à la masse Q_3 est maintenue bloquée malgré le courant dans R_2. Le transistor Q_4 se trouve donc également bloqué. N'importe quel potentiel peut être imposé en S par un circuit extérieur sans détériorer le circuit. Il est ainsi possible de relier plusieurs sorties à condition qu'un seul circuit soit validé à la fois. Les portes logiques OU et ET câblés sont possibles à condition qu'un seul circuit soit validé à la fois. Dans la famille TTL, une sortie passe dans l'état haute impédance en désolidarisant les 2 interrupteurs du $totem - pole$, comme l'illustre la Figure 2.40.

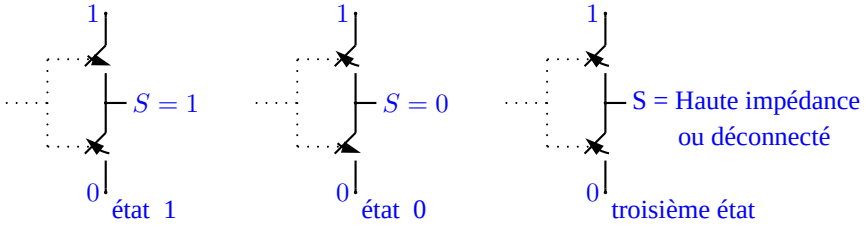

Figure 2.40: *Le principe de fonctionnement du circuit tri-state.*

Ce type de circuit est très utilisé dans les systèmes logiques complexes dans lesquels les informations circulent sur des lignes communes auxquelles sont reliées de nombreux circuits. Ce sont des BUS dans les ordinateurs. Nous pouvons observer sur l'exemple de la Figure 2.41 que pour faire circuler l'information de A à B il suffit de valider seulement les portes 1 et 6, toutes les autres portes devant être *inhibées* i.e., déconnectées par état haute impédance, sous peine de court-circuit destructeur. Les portes 1 à 6 pouvant être des circuits drivers [1] d'entités informatiques par exemple, mémoire, périphérique d'ordinateur.

[1] Driver est équivalent à pilote ou circuit de commande ou de contrôle ou d'interface.

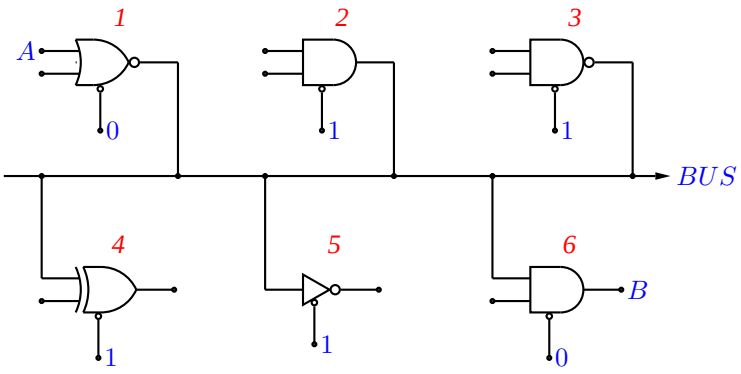

Figure 2.41: *Système de bus.*

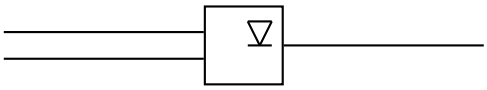

Figure 2.42: *Le symbole de la porte logique et la sortie tri-state.*

2.3.6 *La Logique* I^2L

L'I^2L est une technologie bipolaire rapide utilisée exclusivement dans les circuit intégrés très complexes du type microprocesseurs. La structure fondamentale est représentée sur la Figure 2.43 avec un transistor multi-collecteur.

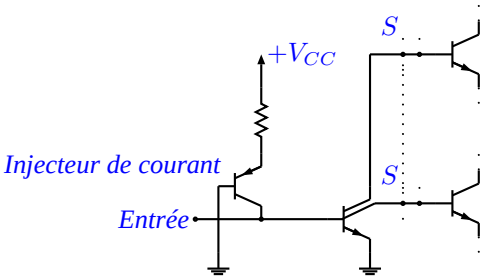

Figure 2.43: *La porte logique* I^2L.

2.3.7 *La Famille ECL*

Dans les montages précédents les transistors fonctionnent au blocage et à la saturation, or nous savons qu'un transistor saturé accumule dans sa base une charge qui doit être éliminée pour obtenir le blocage, ce qui prend un certain temps. Pour augmenter la vitesse de fonctionnement, des familles logiques où les transistors ne sont jamais saturés ont été développées, c'est la cas de la *Logique à Émetteurs Couplés - ECL* de la société *Motorola*. Pour permettre la mise à la masse des collecteurs de l'étage de sortie, l'alimentation est négative (-5 Volt) mais pour faciliter la liaison avec d'autres logiques une alimentation positive est possible. Le niveau logique niveau haut (1) est codé par -0.8 Volt et niveau bas (0) est codé par -1.8 Volt.

2.3.8 *La Famille MOS*

Le *Transistor à Effet de Champ à jonction - FET* n'est pas utilisé pour construire des circuits logiques, il n'en est pas de même du transistor MOS, ou transistor à effet de champ à grille isolée. Pour les transistors MOS construits actuellement, la tension de seuil peut être inférieure à 2 Volt, ce qui permet d'utiliser ces composants avec des tensions d'alimentation de 3 Volt seulement.

La Figure 2.44 représente un inverseur MOS. Sa structure est analogue à celle de l'inverseur à transistor bipolaire, sa caractéristique de transfert peut être tracée à partir du réseau de caractéristiques du MOS pour une valeur donnée de la résistance de charge. La transmission est d'autant plus brutale que la résistance est élevée. Nous observons que le système est compatible avec un niveau logique bas inférieur à 3 Volt, et haut supérieur à 7 Volt pour 10 Volt d'alimentation, comme l'illustre la Figure 2.45.

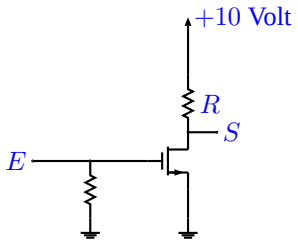

Figure 2.44: *La circuit MOS.*

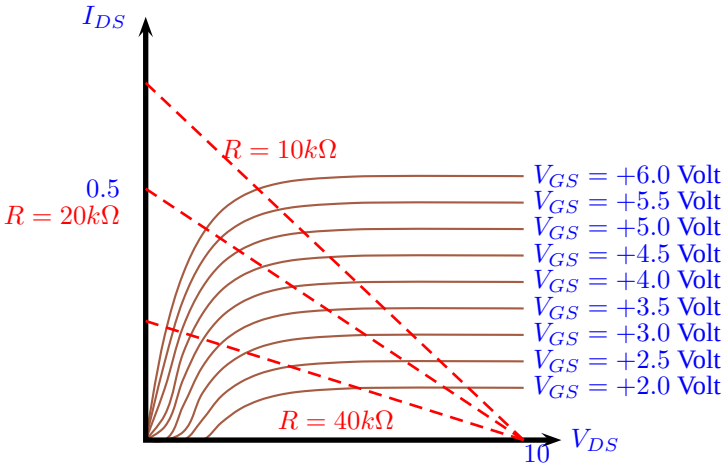

Figure 2.45: *La courant I_{DS} en fonction de IV_{DS} .*

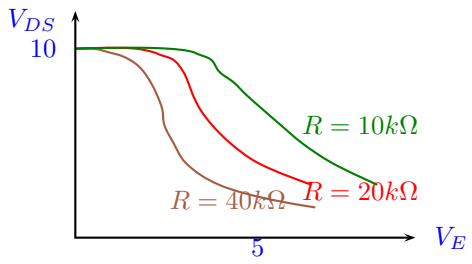

Figure 2.46: *La caractéristique de transfert d'un circuit FET.*

En circuit intégré, une résistance occupe d'autant plus de surface sur la *puce* que sa valeur est importante. Nous avons donc cherché à remplacer la résistance de charge par un second transistor MOS. Considérons un MOS canal N dont la grille est reliée au drain, il constitue un dipôle dont la caractéristique est tracée sur la Figure 2.47. C'est à un décalage de tension près, celle d'une résistance qui peut être utilisée comme charge dans le montage inverseur.

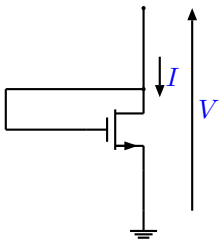

Figure 2.47: *Le MOS canal N.*

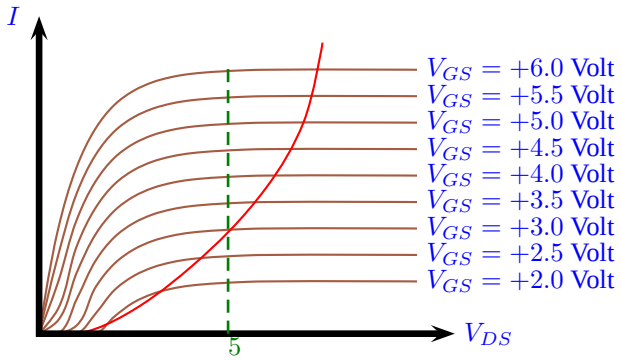

Figure 2.48: *La caractéristique de transfert d'un MOS canal N.*

La Figure 2.49 représente le montage fondamental de l'inverseur MOS dans lequel toute résistance a été bannie. Nous ne détaillons pas ici les nombreuses variantes technologiques mises au point depuis quelques années et qui sont en constante évolution i.e., grille en aluminium et isolement par de la silice, grille en silicium poly ou monocristalline, isolement par du nitrute de silicium ayant une constante diélectrique élevée etc. Les montages $NOR\ MOS$ sont donnés par la Figure 2.50 et la Figure 2.51 illustre la porte logique $NAND\ MOS$.

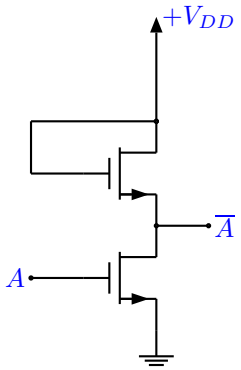

Figure 2.49: *Le montage d'inverseur MOS.*

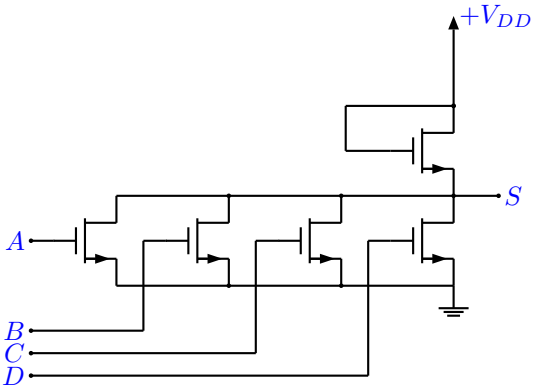

Figure 2.50: *Le montage NOR MOS de quatre entrées.*

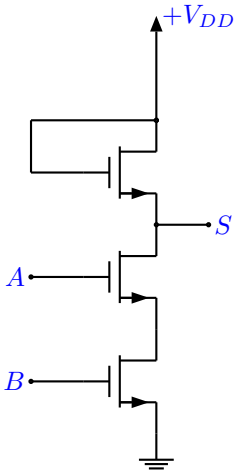

Figure 2.51: *Le montage NAND MOS.*

Une particularité des MOS liée à leur impédance d'entrée très élevée est la présence d'une capacité grille-substrat qui peut être utilisée comme élément de mémoire et qui limite les performances

en vitesse.

2.3.9 *La Famille CMOS*

L'emploi simultané de MOS complémentaire i.e., utiliser par paire deux circuits MOS, un canal N et l'autre canal P, permet de réaliser des circuits dont la consommation au repos est particulièrement basse. La firme américaine $Motorola$ s'est spécialisée dans cette technique et commercialise ces circuits logiques série MC14XXX. Le circuit fondamental est l'inverseur qui est illustré par la Figure 2.52.

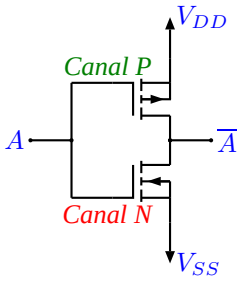

Figure 2.52: *Le circuit fondamental CMOS.*

Lorsque $V_E = V_{DD}$, niveau haut, le $MOS - N$ ayant sa grille positive est conducteur. Par contre le $MOS - P$ est bloqué. Donc V_S est petit i.e., proche de zéro mais le courant consommé est nul, $MOS - P$ étant bloqué. Lorsque $V_E = 0$ niveau bas, le $MOS - N$ est bloqué. Il ne faut pas oublier qu'il s'agit toujours de MOS à enrichissement ayant un ID_{SS} nul. Par contre, $MOS-P$ est conducteur et $V_S = V_{DD}$. Il faut observer que encore $MOS-N$ étant bloqué, le courant consommé par la cellule est nul. Les deux transistors ne sont pas simultanément conducteurs, le circuit ne consomme donc rien à l'état stable. Une consommation apparaît seulement en régime transitoire car il faut charger et décharger les capacités de structure. La consommation typique à vitesse moyenne peut être cent fois inférieure à celle de la cellule identique à transistors à jonctions mais la vitesse limite est actuellement plus faible, typiquement 10 MHz contre plus de 500 MHz pour des ECL.

La tension d'alimentation est différente de la famille TTL. Elle peut donc être de 3 à 18 Volt. Les nouvelles générations plus performantes n'autorisent que 2 à 6 Volt. La puissance consommée est inférieure à TTL et de l'ordre de 0.1 mW i.e., le courant est très faible et inférieur à 1 mA. La tolérance sur les niveaux logiques sont du même ordre qu'en TTL.

Un des problèmes à maîtriser a été la protection des entrées contre les surtensions d'origine statique, la couche d'oxyde des grilles est en effet très fragile. La solution a été trouvée en intégrant des diodes au niveau des entrées. Actuellement ce système fonctionne bien et enlève tout souci à l'utilisateur concernant des manipulations destructrices, comme l'illustre la Figure 2.53. Le circuit de protection constitué d'une série de résistance de protection R_S (1.5 $k\Omega$), de diodes D_3 et D_4, lequel *clam* la tension d'entrée entre V_{DD} et V_{SS}. Les diodes D_1 et D_2 sont de structure distribuée qui est un résultat d'un procèdé de fabrication.

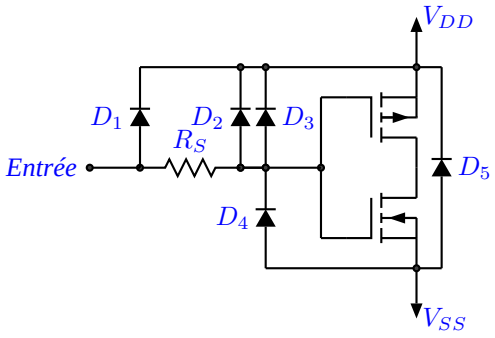

Figure 2.53: *Le circuit de protection d'entrée.*

La Figure 2.54 représente un $NAND$ à deux entrées. Si l'une des entrées est à zéro le MOS correspondant M_1 ou M_2 de type P est conducteur amenant S au $+V_{DD}$. Si au contraire A et B sont à V_{DD}, M_1 et M_2 sont bloqués mais M_3 et M_4 conducteurs, fixent S au zéro.

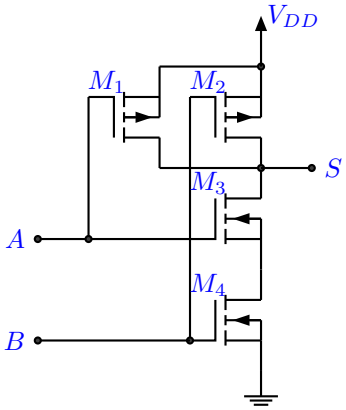

Figure 2.54: *La porte logique NAND à CMOS.*

Un MOS n'ayant pas de tension d'offset les niveaux de sortie (sans charge) sont rigoureusement $+V_{DD}$) et zéro, les impédances de sortie étant les résistances des canaux, ces résistances sont de l'ordre du kΩ. Le courant d'entrée est toujours très faible, typiquement 10 pA, le courant susceptible d'être délivré en sortie est au maximum de l'ordre du milliampère. Au moins en fonctionnement lent, la sortance est donc très grande. Les constructeurs l'annoncent supérieure à 50. Comme pour la famille TTL, il y a 3 variantes pour le circuit de sortie : - *totem-pole, open drain et tri-state*.

Les circuits $CMOS$ sont de plus en plus utilisés grâce à leur souplesse d'emploi i.e., les niveaux de sortie très bien définis et de grande immunité au bruit. Par contre les impédances d'entrée favorisent la réception de signaux parasites rayonnés. Cette possibilité fonctionne dans une large plage de tensions d'alimentation avec une très faible consommation. La dernière propriété des $CMOS$, est liée à leur impédance d'entrée et comportement en sortie et la possibilité de les utiliser dans les configurations où ils fonctionnent de façon pseudo-linéaire i.e., comme un amplificateur, un oscillateur, etc. Nous trouvons aussi la série $74C$ dont les circuits sont compatibles broche à broche avec ceux portant le numéro correspondant en logique TTL.

2.3.10 *Les Interfaces entre Familles*

Pour des raisons d'incompatibilité entre les familles logiques, tous les circuits logiques connectés d'un montage doivent être de la même famille; dans le cas contraire, il faut en outre prévoir des circuits d'interfaçage. Ce sont des circuits permettant l'association de circuits logiques appartenant à des familles logiques différentes. Les cas les plus souvent rencontrés sont les suivants :

$\Rightarrow$ 1. L'attaque d'une logique lente, le plus souvent TTL, par une logique ultra rapide la famille ECL. L'inverse étant sans intérêt.

$\Rightarrow$ 2. Une association de circuits TTL et $CMOS$.

Attaque d'un Circuit CMOS par un Circuit TTL

Une porte logique TTL totem-pole fournit en sortie au plus 0.4 Volt au niveau 0 et au moins 3.6 Volt au niveau 1, comme l'illustre la Figure 2.55.

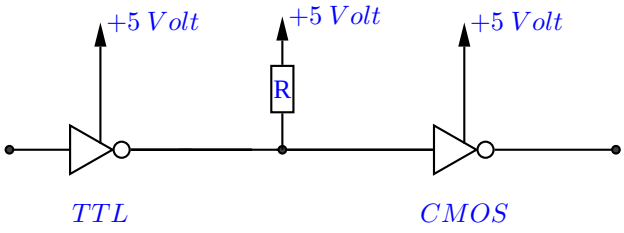

Figure 2.55: *L'interface CMOS-TTL.*

Sous $V_{DD} = +5$ Volt, il faut pour le $CMOS$ au plus 1.5 Volt au niveau 0 et au moins 3.5 Volt au niveau 1. En conséquence, la TTL pilote sans problème le $CMOS$ au niveau 0. C'est par contre un peu juste au niveau 1. Nous utilisons alors une résistance R dite de *pull-up* ou R de l'ordre de 10 kΩ qui remonte le niveau haut de la TTL.

Attaque d'un Circuit TTL par un Circuit CMOS

Au niveau 1 il n'y a pas de problème car l'entrée TTL se contente d'un courant faible. Il n'en est pas de même au niveau 0 i.e., pour une tension de 0.8 Volt maximum il faut extraire d'une entrée TTL un courant de 1.6 mA. Une porte $CMOS$ peut tout juste accepter 0.8 mA pour cette valeur de tension. La liaison directe est donc impossible, comme l'illustre la Figure 2.56.

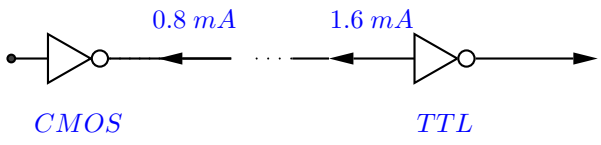

Figure 2.56: *Le problème interface TTL-CMOS.*

Nous pouvons alors utiliser un circuit d'interface spécialisé ou même un transistor intermédiaire qui impliquera alors une inversion, comme l'illustre la Figure 2.57.

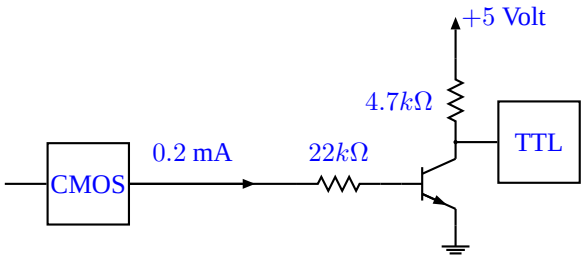

Figure 2.57: *Le circuit interface TTL-CMOS.*

Interface ECL-TTL

La difficulté est différente suivant que l'ECL est alimenté entre 0 et $+5$ Volt ou entre -5 Volt et 0. Dans le cas où une association avec des circuits TTL est prévue, la première solution est généralement retenue. Des circuits spécialisés ou des montages adaptateurs sont proposés par le constructeur.

2.4 *Conclusion*

Nous venons d'étudier les différentes familles logiques et nous pouvons conclure que chaque famille a des caractéristiques technologiques différents et qui nous obligent de choisir la technologie la plus adaptée pour réaliser un projet. Dans le prochain chapitre, nous étudierons des différentes applications de la logique combinatoire sans prendre en compte les aspects de la technologie.

Chapitre 3

Réalisation Spéciale

3.1 *Introduction*

A partir de la logique combinatoire, nous pouvons réaliser un ensemble de différentes applications. Donc, nous étudierons le codeur, le décodeur, le multiplexeur, le transcodeur BCD/7 segments, la conversion parallèle-série (registre à décalage), la conversion série-parallèle, l'affichage multiplexé, les circuits arithmétiques (additionneur et soustracteur), le comparateur, le générateur de parité et pour finir l'unité logique et arithmétique.

3.2 *Codeur*

Définition 3.2.1 *Un codeur est un dispositif qui traduit les valeurs d'une entrée dans un code choisi. Par exemple, un clavier de console ou de machine à écrire comporte n touches. Chaque touche, représentative d'un caractère, est affectée d'un numéro. Donc à chaque caractère, un équivalent binaire, i.e., un mot composé d'éléments binaires.*

Dans la symbolique du schéma de la Figure 3.1 et contrairement à la majorité des technologies, une entrée en l'air *est au niveau logique* 0.

Si $i \Rightarrow 4$ et soit $N = S_3S_2S_1S_0$, donc nous avons $S_3 = 0$, $S_2 = 1$, $S_1 = 0$, $S_0 = 0$ pour un codeur binaire. Si seul le bouton numéro i est actionné, le nombre binaire à 4 éléments $N = S_3S_2S_1S_0$ est égal à i, dans le code choisi. Les diagrammes de Karnaugh, *une table pour chaque sortie, ne sont pas utilisés car le nombre d'éventualités est réduit à 1 seule entrée activée.*

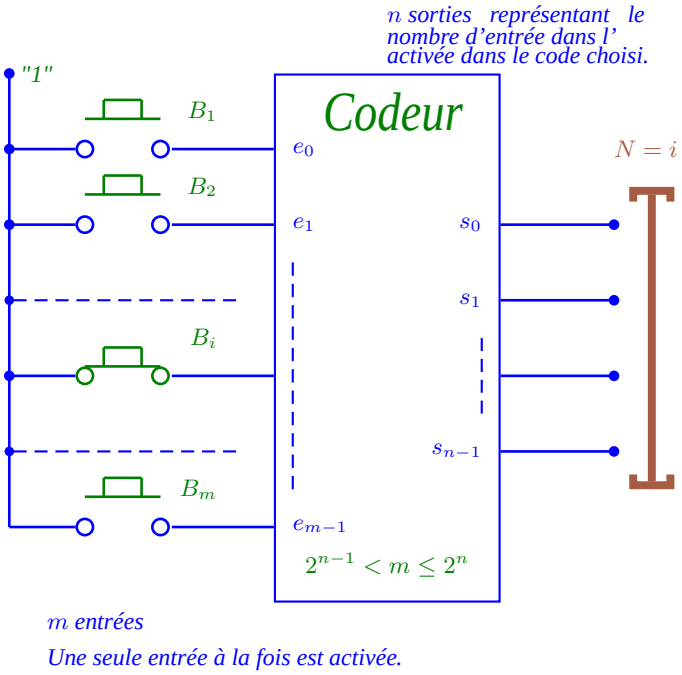

Figure 3.1: *Schéma symbolique d'un codeur.*

3.2.1 *Intérêt du Codage*

Si le nombre de boutons est de 10, codé en binaire pur, 4 variables suffisent. Pour un clavier classique, la quarantaine de touches se code facilement avec 6 variables binaires. Le codage des informations apporte une réduction non négligeable du nombre de variables à traiter.

3.2.2 *Codeur Prioritaire*

Si maladroitement plusieurs boutons sont enfoncés simultanément, le codeur classique donne un résultat erroné car il ne sait plus quel numéro doit être codé. Un codeur prioritaire est un dispositif réalisant le codage du numéro le plus élevé dans le cas où plusieurs boutons seraient actionnés simultanément. Si une seule commande est envoyée sur le codeur prioritaire, celui-ci fonctionne comme un codeur classique. Si 2 entrées ou plus sont activées simultanément, comme l'illustre la Figure 3.2, l'entrée sélectionnée pour le codage est celle avant le numéro d'entrée le plus élevé. Sinon le codeur prioritaire se comporte comme un codeur classique.

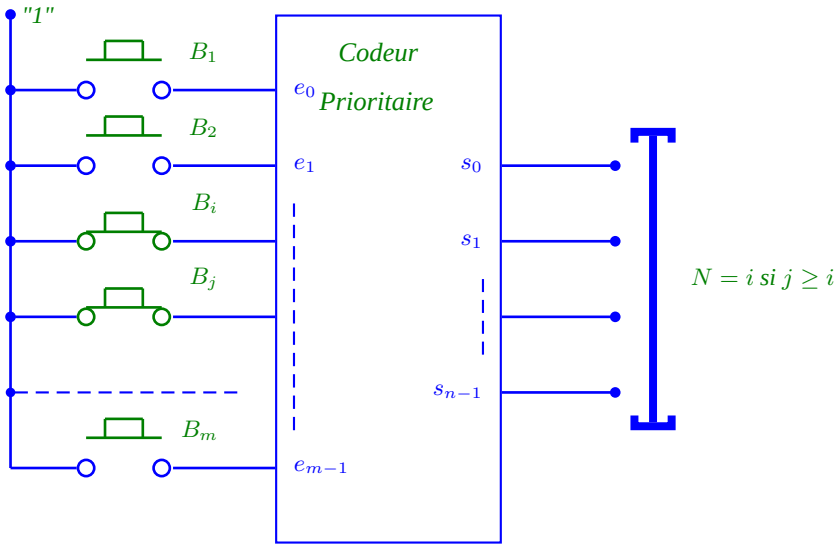

Figure 3.2: *Schéma symbolique d'un codeur prioritaire.*

3.2.3 *Réalisation Pratique de Codeur*

Dans sa version la plus générale, un codeur est un ensemble de circuits *OU*.

Exemple 3.2.1 *Soit la table de codage suivante pour des entrées de e_o à e_g en code* DCB *qui est traduite par $ABCD$, comme l'illustre la Table 3.1. Le codage des sorties est donné par l'équation 3.2.1. L'implémentation du circuit est donné par la Figure 3.3 et la représentation symbolique est illustrée par la Figure 3.4.*

Entrée	Sorties
$\mathbf{E_0}$	**0000**
$\mathbf{E_1}$	**0001**
$\mathbf{E_2}$	**0010**
$\mathbf{E_3}$	**0011**
$\mathbf{E_4}$	**0100**
$\mathbf{E_5}$	**0101**
$\mathbf{E_6}$	**0110**
$\mathbf{E_7}$	**0111**
$\mathbf{E_8}$	**1000**
$\mathbf{E_9}$	**1001**

Table 3.1: *Les codages de sorties.*

$$\begin{array}{rcl} S_0 & = & E_7 + E_8 \\ S_1 & = & E_3 + E_4 + E_5 + E_6 \\ S_2 & = & E_1 + E_2 + E_5 + E_6 \\ S_3 & = & E_0 + E_2 + E_4 + E_6 + E_8 \end{array} \tag{3.2.1}$$

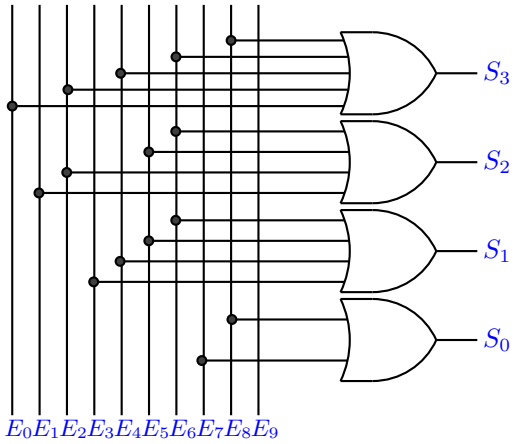

Figure 3.3: *Le schéma du codeur BCD.*

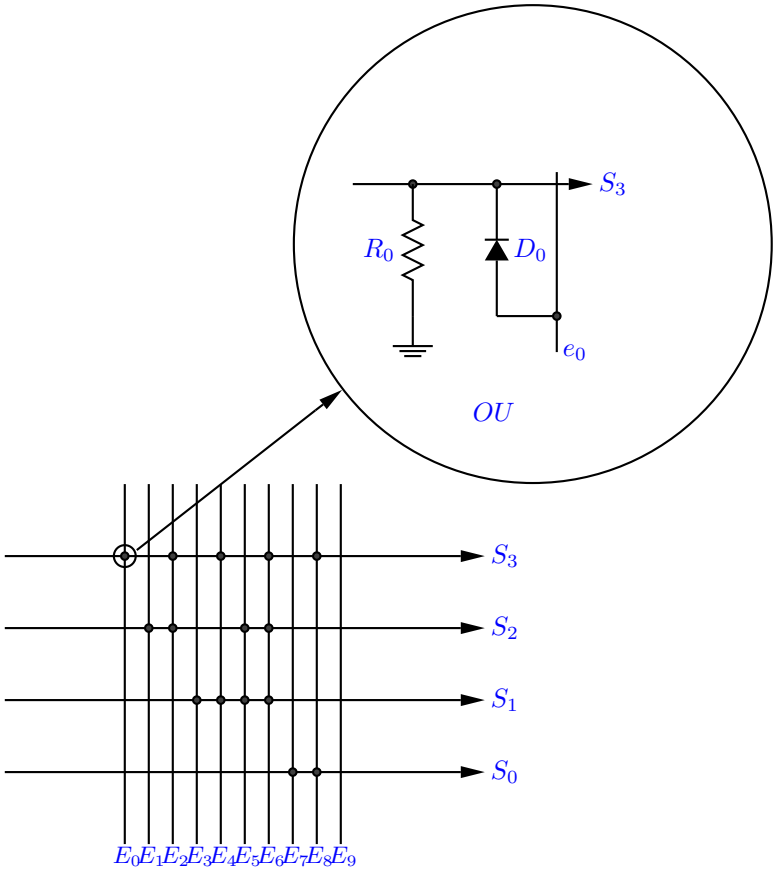

Figure 3.4: *La représentation symbolique du codeur BCD.*

3.3 *Décodeur*

Définition 3.3.1 *Un décodeur est un circuit qui délivre une (ou des) information(s) lorsque la combinaison des variables binaires d'entrée est représentative du (ou des) mot(s)-code choisi(s). Un décodeur réalise la fonction inverse ou duale du codeur, comme l'illustre la Figure 3.5.*

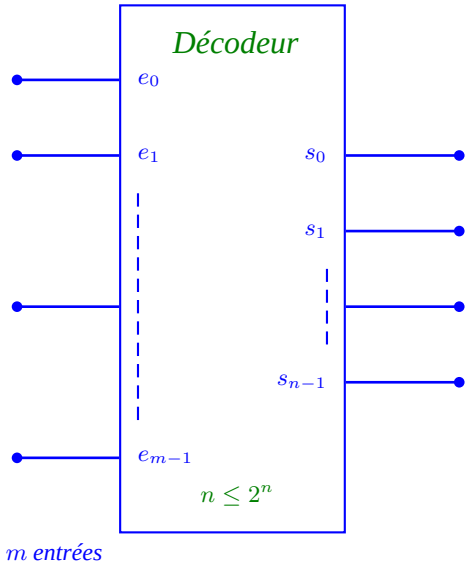

Figure 3.5: *Le schéma d'un décodeur.*

Exemple 3.3.1 *Un décodeur typique est illustré par la Figure 3.6.*

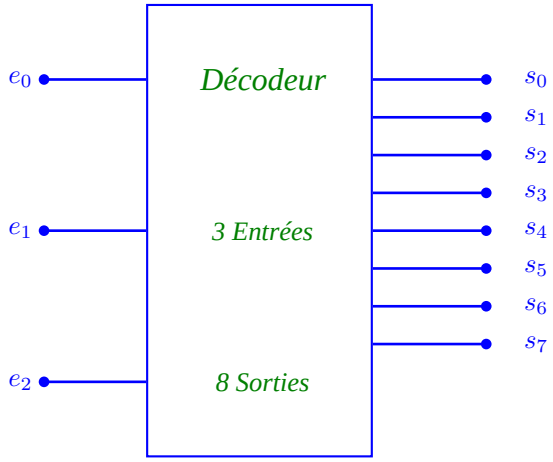

Figure 3.6: *Le décodeur trois entrées et huit sorties.*

3.3.1 *Réalisation Pratique de Décodeur*

La réalisation des décodeurs se fait à partir d'une matrice *ET*. L'expression d'une sortie S_i d'un décodeur est un $minterme$ sur les entrées e_i ou $S_i = m_i$.

Exemple 3.3.2 *Décodeur 3 entrées 8 sorties défini par l'équation 3.3.1. La réalisation est illustrée par la Figure 3.7 et la représentation symbolique par la Figure 3.8.*

$$\begin{array}{rcl} S_0 & = & \overline{E_2}.\overline{E_1}.\overline{E_0} \\ S_1 & = & \overline{E_2}.\overline{E_1}.E_0 \\ S_2 & = & \overline{E_2}.E_1.\overline{E_0} \\ S_3 & = & \overline{E_2}.E_1.E_0 \\ S_4 & = & E_2.\overline{E_1}.\overline{E_0} \\ S_5 & = & E_2.\overline{E_1}.E_0 \\ S_6 & = & E_2.E_1.\overline{E_0} \\ S_7 & = & E_2.E_1.E_0 \end{array} \qquad (3.3.1)$$

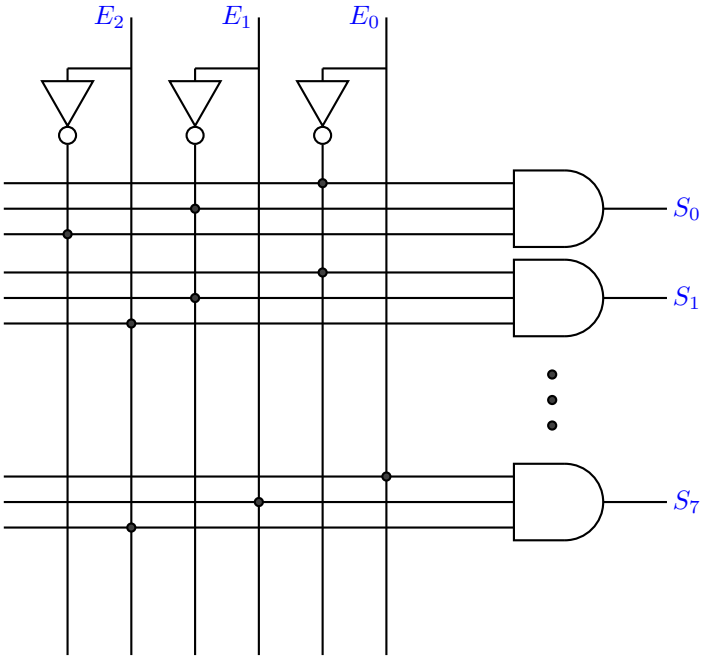

Figure 3.7: *L'implémentation d'un décodeur.*

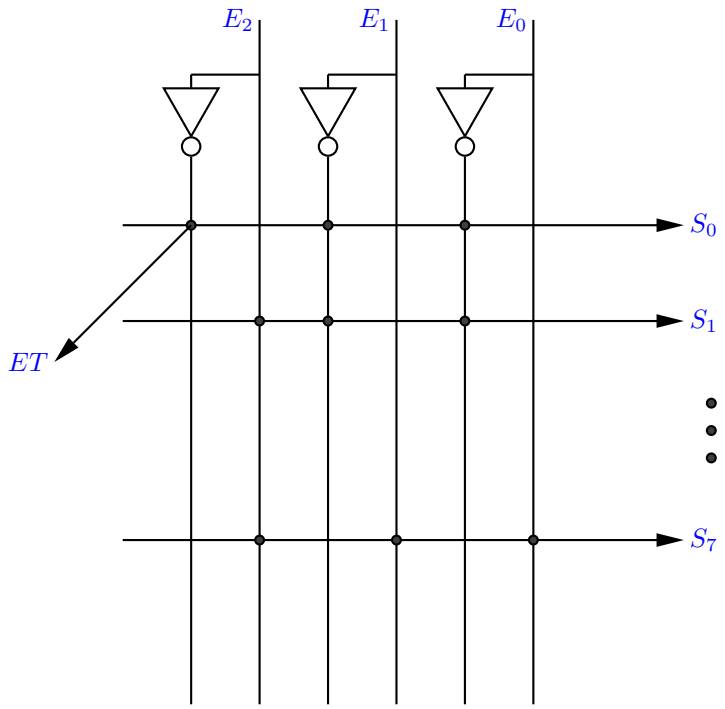

Figure 3.8: *La représentation symbolique du décodeur trois entrées et huit sorties.*

3.3.2 *Application de Décodeur*

Adressage d'une Mémoire

Représentons une mémoire comme un tableau d'éléments binaires. Ce tableau est divisé en lignes et en colonnes. Pour lire un mot mémoire, il faut lui envoyer le numéro de la ligne souhaitée i.e., c'est son adresse. Une mémoire ayant 1024 lignes, par exemple, nécessite 10 bits d'adresse. Un décodeur interne à la mémoire permet la sélection d'une ligne et d'une seule à un instant donné, comme l'illustre la Figure 3.9.

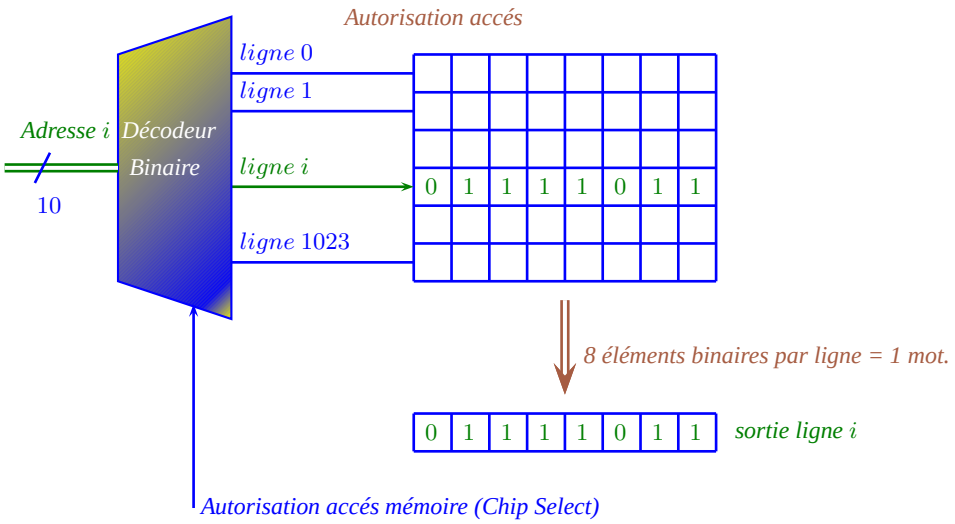

Figure 3.9: *Le schéma d'adressage d'une mémoire.*

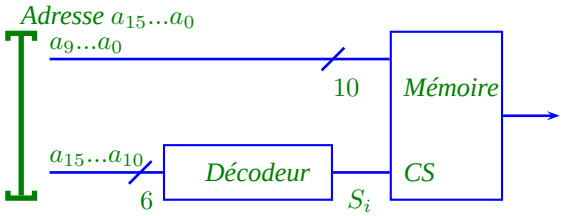

Figure 3.10: *Application d'un décodeur pour l'adressage de mémoire.*

Si un microprocesseur délivre une adresse sur 16 fils, il possède une capacité d'adressage (*espace d'adresse*) de 2^{16} soit 65.536 mots. Les 1024 mots de la mémoire précédente occupent donc une faible partie de cet espace. Il est alors commode de partager celui-ci en 64 pages de 1024 mots, chaque page pouvant correspondre à un boîtier mémoire. La sélection du numéro de page, donc du boîtier correspondant (*Chip Select*), est effectué par le décodage de 6 bits parmi les 16 (en général les poids forts). Les bits restants permettent la sélection interne d'un mot mémoire, comme l'illustre la Figure 3.10. La sortie S_i du décodeur est connectée à l'entrée *Chip Select* (*CS*). Si le nombre décimal équivalent à $(a_{15}, ..., a_{10})_2$ est différent de i, la mémoire ne délivre aucune information en sortie. Dans le cas contraire, la sortie de la mémoire est le contenu de la ligne dont le numéro est fixé par les adresses $(a_9, ..., a_0)$ avec $0 \leq i \leq 63$. Avec $i = 3$, pour accéder à la mémoire, le microprocesseur doit envoyer une adresse $(a_{15}, ..., a_{10})$ telle que $(a_{15}, ..., a_{10})_2 = 3_{10}$ ce qui correspond à des adresses donnée par l'équation 3.3.2.

$$(0C00)_{16} \leq (000011A_9...A_0)_2 \leq (0FFF)_{16}$$

$$1\ caractére\ hédécimal\ est\ égal\ 4\ bits\ car\ 2^4 = 16 \tag{3.3.2}$$

$$ou\ bien\ \Rightarrow (3072)_{10} \leq Adresse \leq (4095)_{10}$$

Génération de Fonction

Comme toute fonction logique S peut s'exprimer comme une somme de mintermes i.e., $S = \sum_i m_i$, il suffit, pour engendrer S, de faire un *OU* avec les sorties $S_i = m_i$ d'un décodeur $S = \sum_i S_i$ et nous pouvons conclure qu'une sortie de décodeur est un *minterme*.

Exemple 3.3.3 *La Table 3.2 et figure 3.11*

N	E_2	E_1	E_0	$f(E_2, E_1, E_0)$
0	**0**	**0**	**0**	**0**
1	**0**	**0**	**1**	**0**
2	**0**	**1**	**0**	**0**
3	**0**	**1**	**1**	**1**
4	**1**	**0**	**0**	**0**
5	**1**	**0**	**1**	**1**
6	**1**	**1**	**0**	**1**
7	**1**	**1**	**1**	**1**

Table 3.2: *Table pour la synthèse d'une fonction logique.*

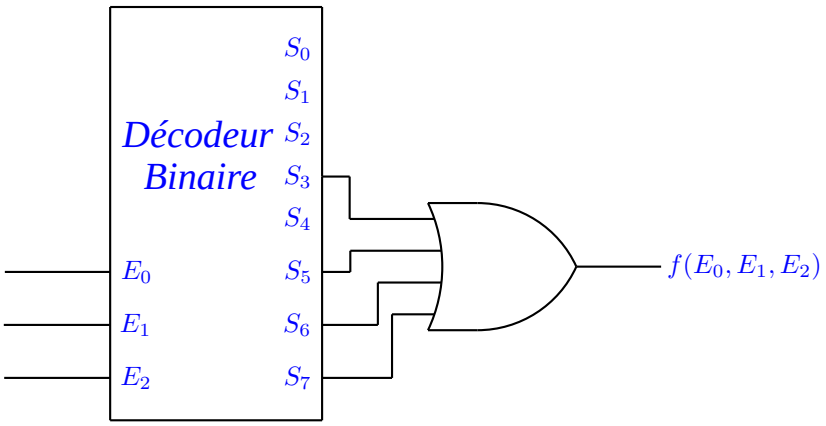

Figure 3.11: *Implémentation de la fonction logique de la Table 3.2.*

3.3.3 *Transcodeur*

Un transcodeur est un dispositif permettant de passer du nombre N écrit dans un code C_i au même nombre N écrit dans le code C_2.

Synthèse d'un Transcodeur

Le nombre N dans le code C_i s'exprime à l'aide des variables A,B,C,D par exemple, et dans le code C_2 avec les variables X, Y, Z et il faut observer que le nombre des variables dans chaque code n'est pas forcément identique. Le problème de la synthèse d'un transcodeur revient à calculer chacune des sorties, i.e., les variables de C_2, dans le cas X, Y, Z en fonction des entrées ou variables du code C_1 dans le cas A, B, C, D. Nous avons besoin d'exprimer les trois variables de C_2 en fonction des variables A,B,C,D, comme le montre l'équation 3.3.3.

$$\begin{array}{rcl} X & = & f_1(A,B,C,D) \\ Y & = & f_2(A,B,C,D) \\ Z & = & f_3(A,B,C,D) \end{array} \tag{3.3.3}$$

Exemple 3.3.4 *Transcodage d'un nombre N en code* Gray *vers un code binaire pur, est égal à la moitié du nombre N arrondie à la partie entière inférieure. Par définition, le code de* Gray *est défini tel qu'l seul bit change au passage d'un mot au suivant, comme l'illustre la Table 3.3. La variable X est égale à 1 si les variables $ABCD$ prennent les valeurs* 1100, 1101, 1111, 1110, 101, 1011, 1001 *ou* 1000, *ce que l'on peut reporter dans un diagramme de* Karnaugh *pour obtenir l'expression la plus simple de la fonction. Nous procédons de la même façon pour les variables Y et Z ce qui donne les Figures 3.12,3.13 et 3.14. Le projet final est donné par la Figure 3.15.*

N	$ABCD \Rightarrow$ *Code Gray*	$XYZ \Rightarrow$ *Moitié en Binaire*
0	**0000**	**000**
1	**0001**	**000**
2	**0011**	**001**
3	**0010**	**001**
4	**0110**	**010**
5	**0111**	**010**
6	**0101**	**011**
7	**0100**	**011**
8	**1100**	**100**
9	**1101**	**100**
10	**1111**	**101**
11	**1110**	**101**
12	**1010**	**110**
13	**1011**	**110**
14	**1001**	**111**
15	**1000**	**111**

Table 3.3: *Table de conversion de code Gray en binaire.*

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

$X = A$

Figure 3.12: *Diagramme de Karnaugh pour la sortie X.*

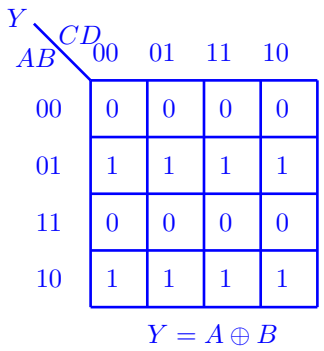

Figure 3.13: *Diagramme de Karnaugh pour la sortie Y.*

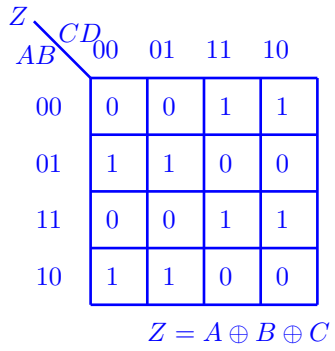

Figure 3.14: *Diagramme de Karnaugh pour la sortie Z.*

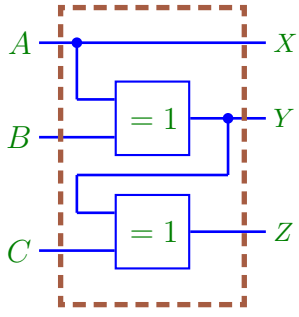

Figure 3.15: *Le schéma pour les fonctions logiques X, Y et Z.*

Synthèse d'un Transcodeur BCD à 7 Segments

Nous appelons transcodeur *BCD* à 7 segments le dispositif de transcodage permettant de passer du *Décimal Codé Binaire* encore appelé binaire pur ou du code Hexadécimal Codé Binaire au code d'affichage du chiffre sur un afficheur 7 segments. L'opération de décodage du chiffre est réalisé visuellement i.e.,interprétation visuelle de la forme du chiffre formé par l'allumage des segments. Soient a, b, c, d, e, f et g les variables correspondant aux 7 segments. Si une variable est au niveau actif, par exemple 1, le segment correspondant est allumé. Les segments sont répartis comme l'indique la Figure 3.16.

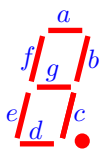

Figure 3.16: *Afficheur sept segments.*

Les chiffres étant formés sont montrés par la Figure 3.17.

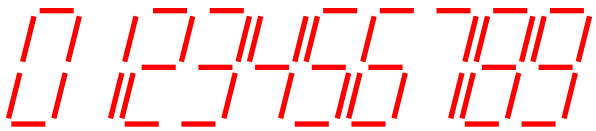

Figure 3.17: *Les chiffres de zéro à neuf.*

Le code à 7 segments correspondant est donné par la Table 3.4.

N	$abcdefg$
0	**1111110**
1	**0110000**
2	**1101101**
3	**1111001**
4	**0110011**
5	**1011011**
6	**1011111**
7	**1110000**
8	**1111111**
9	**1111011**

Table 3.4: *Valeurs logiques des segments de l'afficheur.*

La synthèse d'un décodeur 7 segments s'effectue comme pour un transcodeur classique. Le code C_2 de représentation du nombre N ayant 7 variables, il y a 7 fonctions à calculer (a, b, c, d e f et g) en fonction des variables du code C_1.

3.3.4 *Multiplexeur*

Définition 3.3.2 *Un multiplexeur est un circuit réalisant un aiguillage de l'une des entrées de données, par la commande des entrées d'adresse, vers une sortie unique. Il y a sélection d'une donnée parmi 2^n où n correspond aux entrées d'adresses, comme l'illustre la Figure 3.18.*

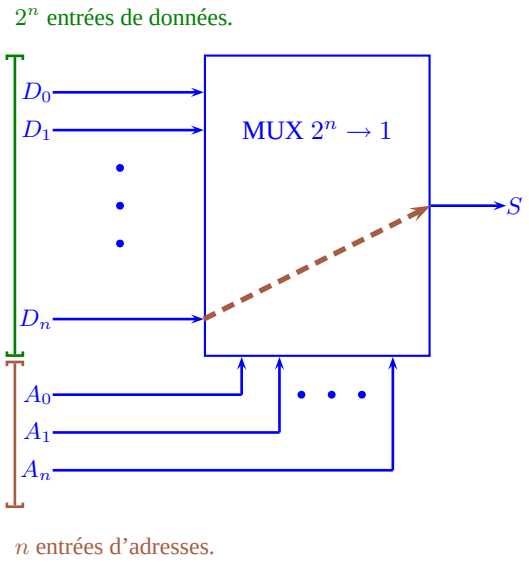

Figure 3.18: *Le schéma d'une MUX 2^n entrées et une sortie.*

Exemple 3.3.5 *Multiplexeur $2 \rightarrow 1$, comme l'illustre la Figure 3.19, la Table de vérité 3.5 et l'équation 3.3.4.*

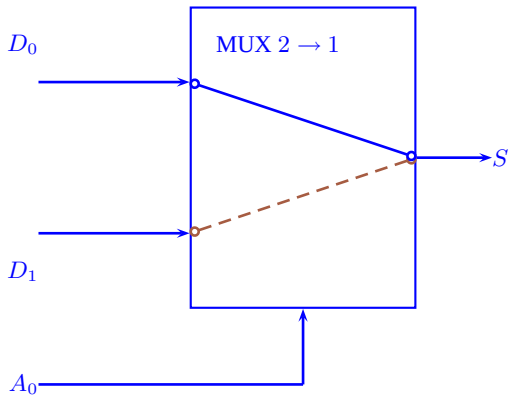

Figure 3.19: *Le schéma d'une MUX deux entrées et une sortie.*

$\mathbf{A_0}$	$\mathbf{S}$
0	$\mathbf{D_0}$
1	$\mathbf{D_1}$

Table 3.5: *Table de vérité pour la MUX* $2 \rightarrow 1$.

$$\begin{cases} S = D_0 & si\ A_0 = 0 \\ S = D_1 & si\ A_0 = 1 \end{cases} \qquad (3.3.4)$$

$$soit \qquad S = D_0\overline{A_0} + D_1 A_0$$

Exemple 3.3.6 *Multiplexeur* $4 \rightarrow 1$*, comme l'illustre la Figure 3.20, la Table de vérité 3.6 et l'équation 3.3.5. La réalisation est illustrée par la Figure 3.21 et la représentation symbolique par la Figure 3.22.*

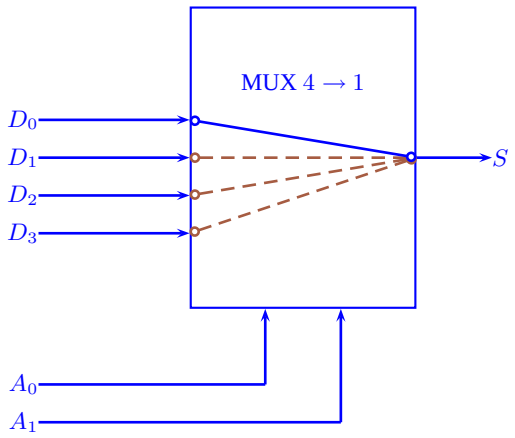

Figure 3.20: *Le schéma d'une MUX quatre entrées et une sortie.*

$\mathbf{A_1}$	$\mathbf{A_0}$	$\mathbf{S}$
0	**0**	$\mathbf{D_0}$
0	**1**	$\mathbf{D_1}$
1	**0**	$\mathbf{D_2}$
1	**1**	$\mathbf{D_3}$

Table 3.6: *Table de vérité pour la MUX* $4 \rightarrow 1$.

$$\begin{cases} S = D_0 & si \quad A_1A_0 = 00 \\ S = D_1 & si \quad A_1A_0 = 01 \\ S = D_2 & si \quad A_1A_0 = 10 \\ S = D_3 & si \quad A_1A_0 = 11 \end{cases} \tag{3.3.5}$$

$$\Downarrow$$

$$S = D_0.\overline{A_1}.\overline{A_0} + D_1.\overline{A_1}.A_0 + D_2.A_1.\overline{A_0} + D_3.A_1.A_0$$

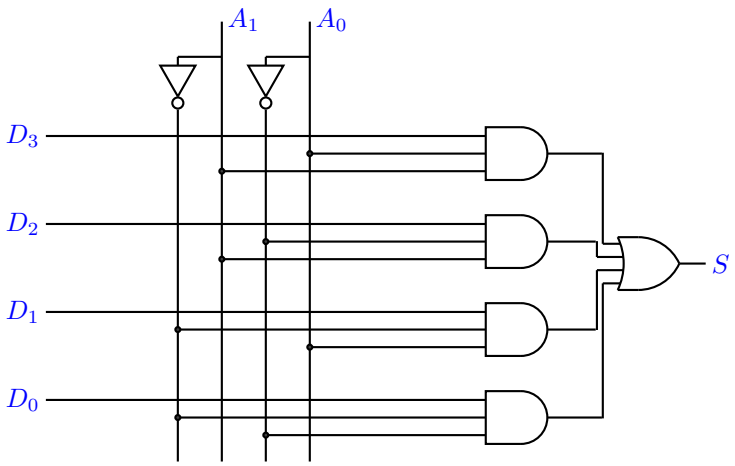

Figure 3.21: *Implémentation d'un multiplexeur* $4 \rightarrow 1$.

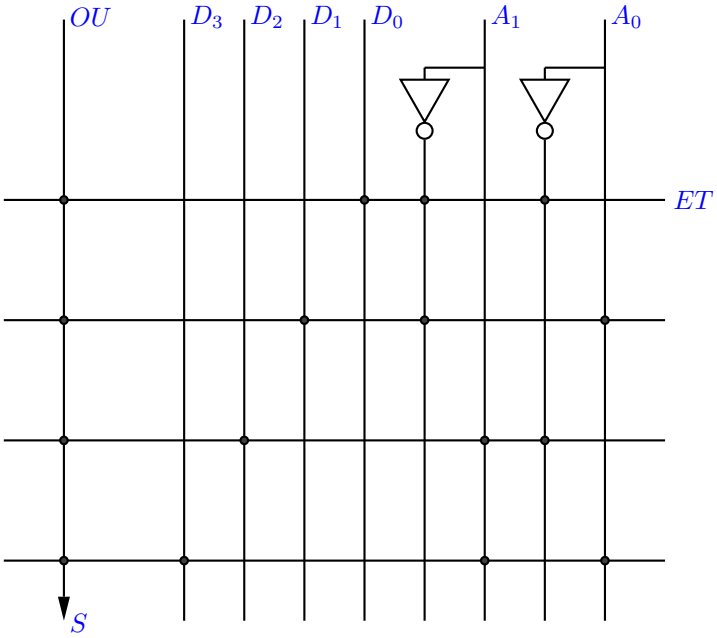

Figure 3.22: *La représentation symbolique d'un multiplexeur* $4 \rightarrow 1$.

De façon générale, la sortie d'un multiplexeur à n entrées d'adresses s'exprime en fonction des entrées de données D_i, et des mintermes m_i sur les entrées d'adresses exprimée par l'équation 3.3.6.

$$S = \sum_{i=1}^{2^n} D_i m_i \tag{3.3.6}$$

Application de Multiplexeur

Exemple 3.3.7 *Sélection d'un mot de* 3 *bits parmi les* 4 *mots de* 3 *bits. Il faut autant de multiplexeurs qu'il y a de bits dans le mot, dans le cas* 3 *multiplexeurs, comme l'illustre la Figure 3.23. La réalisation ou implémentation est illustré par la Figure 3.24.*

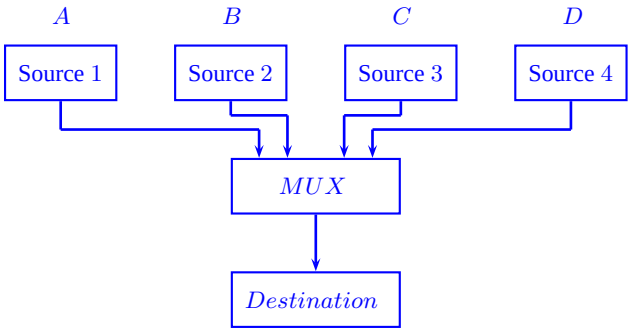

Figure 3.23: *Quatre mots de données* $ABCD$ *issus de quatre lecteurs de bande.*

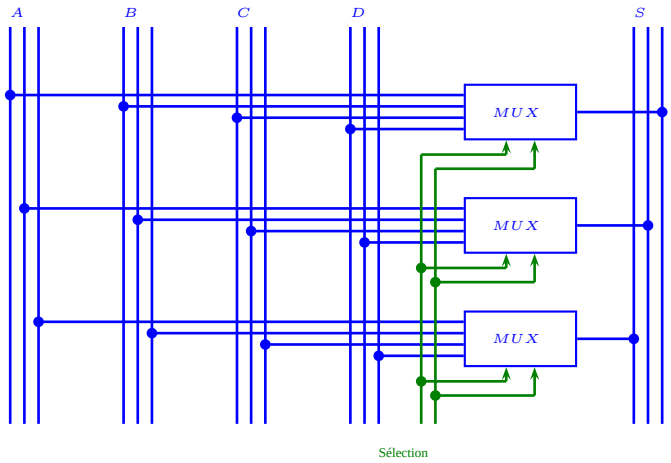

Figure 3.24: *Réalisation du circuit pour quatre mots de données.*

Exemple 3.3.8 *Transmission de plusieurs conversations sur une seule ligne téléphonique numérique est illustrée par la Figure 3.25. Les signaux d'entrée* $s_{m_1}(t)$ *et* $s_{m_2}(t)$ *sont illustrés par les Figures 3.26 et 3.27 respectivement.*[1].

[1] *LAPI- Laboratoire en Processus Intelligents*

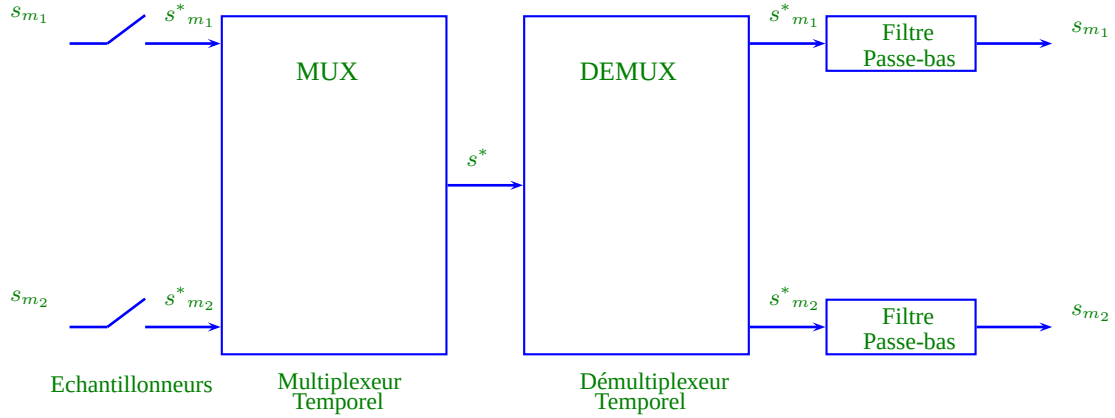

Figure 3.25: *Multiplexeur sur une ligne téléphonique numérique.*

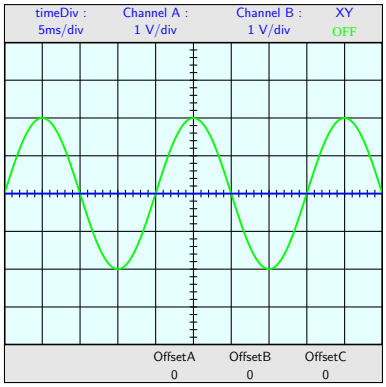

Figure 3.26: *Le signal d'entrée* $s_{m_1}(t)$.

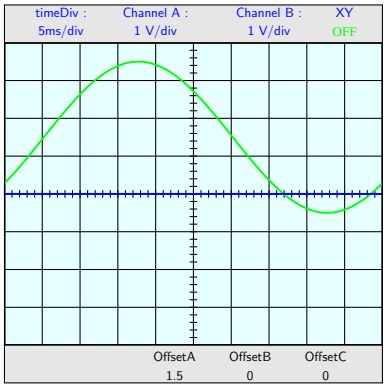

Figure 3.27: *Le signal d'entrée* $s_{m_2}(t)$.

Exemple 3.3.9 *Matérialisation d'une fonction logique i.e., la sortie d'un multiplexeur s'exprimant comme une somme de mintermes (*forme canonique*), et comme toute fonction logique peut se mettre sous forme canonique, elle peut donc s'exprimer comme la sortie d'un multiplexeur. La fonction* f *de* 2 *variables* A *et* B *exprimée sous forme canonique (*somme de mintermes*) i.e.,* $f(A,B) =$

$\overline{A}.\overline{B}.f(0,0) + \overline{A}.B.f(0,1) + A.\overline{B}.f(1,0) + A.B.f(1,1)$ *avec* $f(i,j)$ *égal la valeur particulière fonction logique* f *lorsque* $A = i$ *et* $B = j$*. Donc, nous pouvons utiliser un multiplexeur à* 4 *entrées de données donc* 2 *entrées d'adresses, comme l'illustre la Figure 3.28. La Table de vérité (3.7) illustre les valeurs de la fonction logique et la Figure 3.29 est la réalisation de la fonction logique* ET *à partir du multiplexeur.*

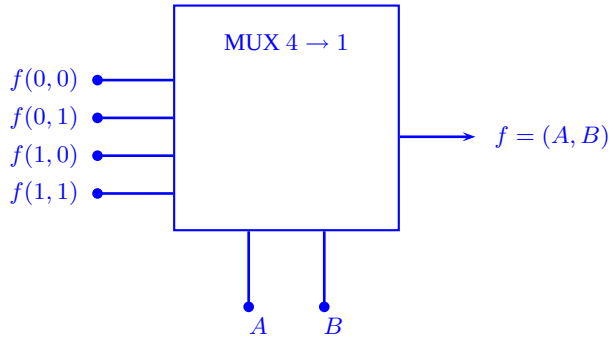

Figure 3.28: *Le diagramme synoptique du multiplexeur.*

A	**B**	**S = A.B**
0	**0**	$\mathbf{D_0 = 0}$
0	**1**	$\mathbf{D_1 = 0}$
1	**0**	$\mathbf{D_2 = 0}$
1	**1**	$\mathbf{D_3 = 1}$

Table 3.7: *Le multiplexeur pour implémenter la fonction logique ET.*

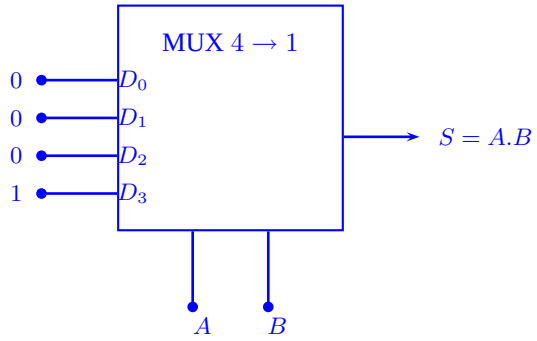

Figure 3.29: *L'implémentation de la fonction logique ET à partir d'un multiplexeur.*

Le multiplexeur est un opérateur programmable. Pour une réalisation par un multiplexeur d'une fonction de n variables, il faut un multiplexeur à 2^n entrées de données ou n représente les entrées d'adresses de multiplexeur $2^n \rightarrow 1$.

Conversion Parallèle-série (Registre à Décalage)

Soit un mot binaire $D = d_3d_2d_1d_0$ disponible en mode parallèle, c'est à dire sur quatre fils, chaque fil étant affecté à un élément binaire (*bit*) du mot. Pour transmettre les éléments binaires en série, i.e., les uns à la suite des autres sur un seul fil, il faut d'abord commencer par le *LSB*

(*Least Significant Bit*), bit de plus faible poids transmettre d_0, puis d_1, ensuite d_2 et enfin d_3. Ceci revient à sélectionner ou aiguiller l'un des éléments binaires de D sur le fil unique de sortie série. Le multiplexeur est capable d'effectuer cette tâche si les combinaisons correspondantes sont placées successivement sur les commandes de sélection. Comme le montre le chronogramme de la Figure 3.30. Dans le premier temps il faut que $A_1 = A_0 = 0$ pour que $S = D_0 = d_0$. Ensuite A_0 passe à 1 ce qui impose $S = D_1 = d_1$. Puis $A_1 = 1$ et $A_0 = 0$ d'où $S = D_2 = d_2$. Et enfin $A_1 = A_0 = 1$ alors $S = D_3 = d_3$, comme l'illustre la Figure 3.31.

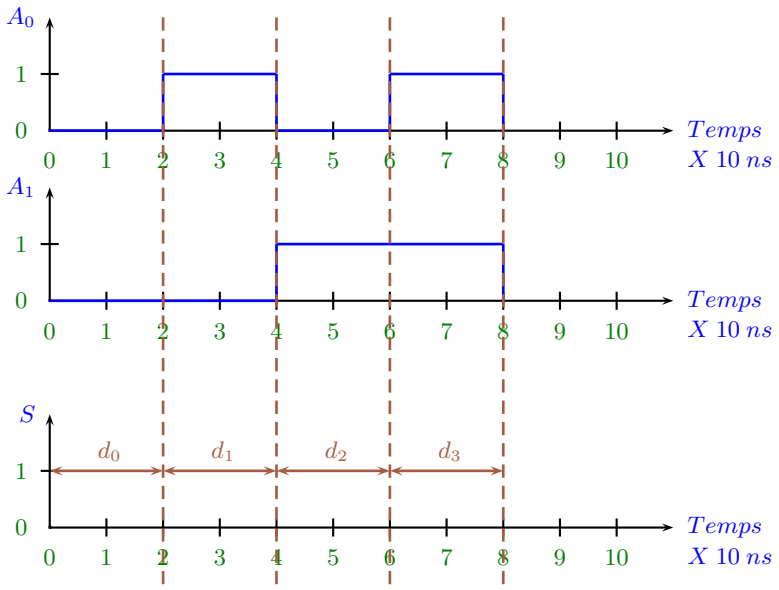

Figure 3.30: *Le diagramme de temps pour la conversion parallèle-série.*

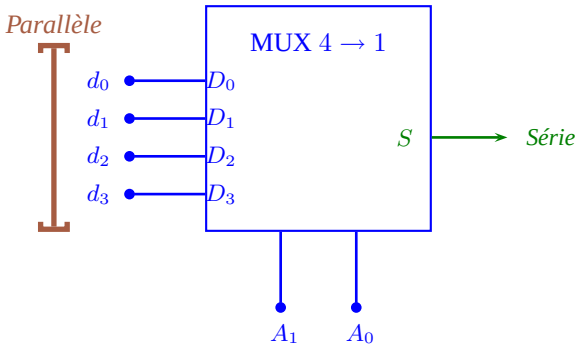

Figure 3.31: *L'implémentation pour la conversion parallèle-série.*

3.4 *Démultiplexeur*

Définition 3.4.1 *Un démultiplexeur réalise l'opération duale du multiplexeur. Il aiguille l'entrée donnée sur* 1 *parmi* 2^n *sorties où* n *entrées d'adresses.*

Les sorties non aiguillées sont non activées, comme l'illustre la Figure 3.32 . Par convention nous les supposons au niveau 0. En fait cela peut être 0 ou 1 suivant la technologie utilisée par le constructeur. De façon générale, la sortie S, d'un démultiplexeur à n entrées d'adresses s'exprime en fonction de l'entrée de donnée d et d'un $minterme\ m_i$ sur les entrées d'adresses (équation 3.4.1).

$$S = m_i d \tag{3.4.1}$$

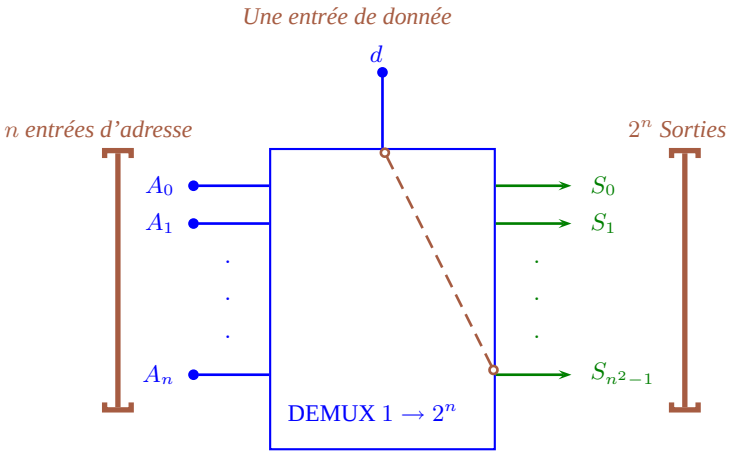

Figure 3.32: *Le diagramme d'un démultiplexeur.*

Exemple 3.4.1 *Soit la fonction logique donnée par la Table 3.8, le diagramme synoptique par la Figure 3.33. L'implémentation est donnée par la Figure 3.34 et la représentation symbolique est donnée par la Figure 3.35.*

$A_2A_1A_0$	S_i
000	$\mathbf{S_0 = d}$
001	$\mathbf{S_1 = d}$
010	$\mathbf{S_2 = d}$
011	$\mathbf{S_3 = d}$
100	$\mathbf{S_4 = d}$
101	$\mathbf{S_5 = d}$
110	$\mathbf{S_6 = d}$
111	$\mathbf{S_7 = d}$

Table 3.8: *La table de vérité d'un démultiplexeur.*

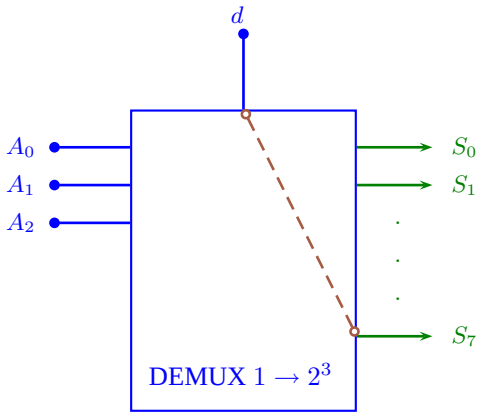

Figure 3.33: *Le diagramme synoptique d'un DEMUX* $1 \rightarrow 2^3$.

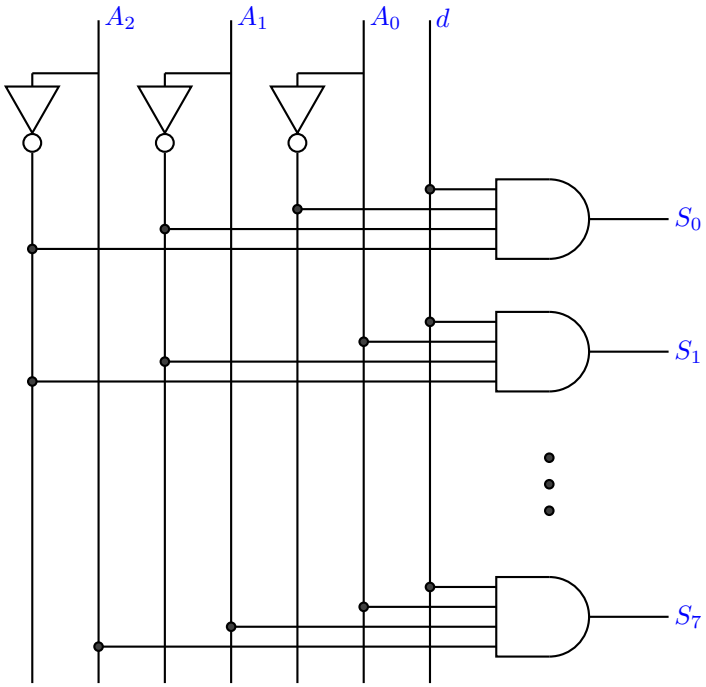

Figure 3.34: *Implémentation d'un démultiplexeur.*

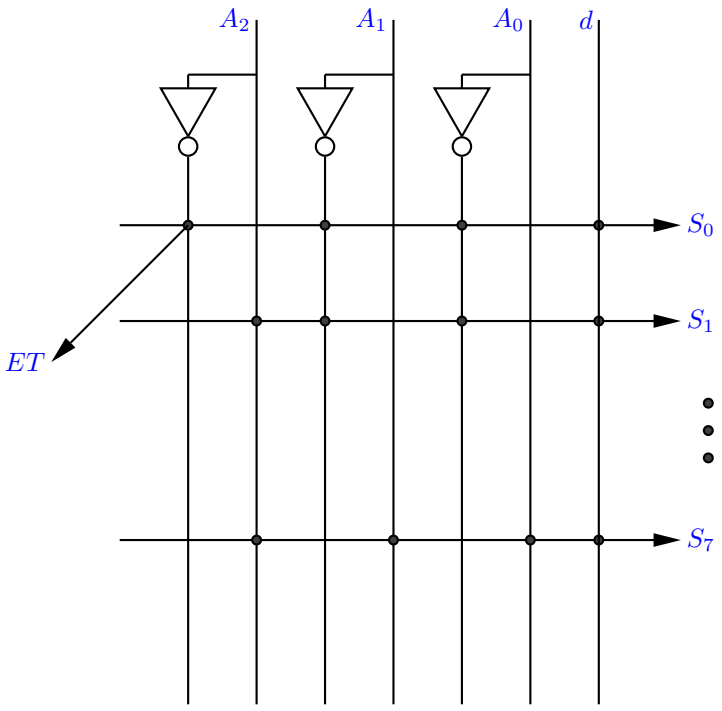

Figure 3.35: *Représentation d'un démultiplexeur.*

3.4.1 *Applications du Démultiplexeur*

Conversion Série-parallèle (Registre à Décalage)

Soit 1 capteur type *ON/OFF* de quatre entrées multiplexées et les informations doivent être stockées dans une mémoire, comme l'illustre la Figure 3.36.

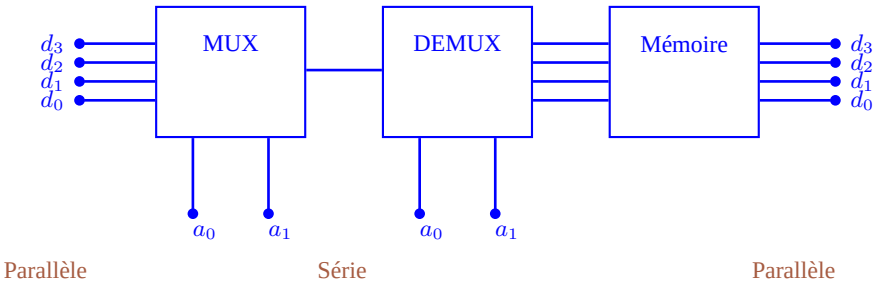

Figure 3.36: *Le schéma d'une application du démultiplexeur.*

Affichage Multiplexé

Soit 4 chiffres à afficher, nous pouvons afficher les chiffres l'un après l'autre très vite pour donner l'impression de simultanéité à l'oeil, comme l'illustre la Figure 3.37.

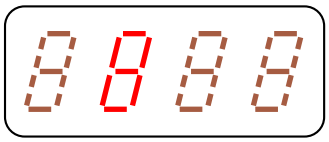

Figure 3.37: *L'affichage multiplxé.*

3.5 *Circuits Intégrés Arithmétiques*

3.5.1 *Additionneur*

C'est un circuit réalisant l'addition de deux nombres binaires. La table d'addition de deux nombres à un élément binaire est donnée par la Table 3.9. Le résultat de l'opération comporte deux parties une est la somme ($\sum$) (Table 3.10) et l'autre est la retenue (r) (Table 3.11) générée.

a	**b**	$\mathbf{S = a \oplus b}$
0	**0**	**0**
0	**1**	**1**
1	**0**	**1**
1	**1**	**10**

Table 3.9: *L'addition de* $a \oplus b$.

a	**b**	$\mathbf{S = a \oplus b}$
0	**0**	**0**
0	**1**	**1**
1	**0**	**1**
1	**1**	**0**

Table 3.10: *Le résultat de l'addition* $\sum = a \oplus b$.

a	**b**	$\mathbf{S = a \oplus b}$
0	**0**	**0**
0	**1**	**1**
1	**0**	**1**
1	**1**	**1**

Table 3.11: *La rétenue* $r = a \bullet b$.

Le circuit élémentaire réalisant cette opération est le *demi-additionneur* et il est illustré par la Figure 3.38.

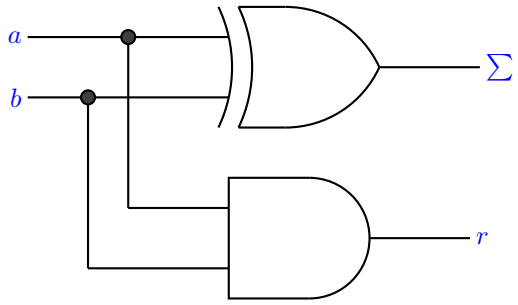

Figure 3.38: *Le circuit demi-additionneur.*

La structure de l'additionneur de deux mots est alors répétitive. Une cellule élémentaire peut donc être utilisée pour chaque poids. Elle est appelée *additionneur complet*. L'addition globale est réalisée par la mise en cascade des cellules au sens des retenues. L'additionneur complet est défini par la Table de vérité 3.12 où r est la retenue propagée de l'étage précédent du mot et r_{i+1} est la retenue générée.

a	b	r_i	r_{i+1}	$\sum_i$
0	**0**	**0**	**0**	**0**
0	**0**	**1**	**0**	**1**
0	**1**	**0**	**0**	**1**
0	**1**	**1**	**1**	**0**
1	**0**	**0**	**0**	**1**
1	**0**	**1**	**1**	**0**
1	**1**	**0**	**1**	**0**
1	**1**	**1**	**1**	**1**

Table 3.12: *La table de vérité de l'additionneur complet.*

La représentation synoptique de l'additionneur complet est illustrée par la Figure 3.39, le diagramme de *Karnaugh* de l'addition est donné par la Figure 3.40 et le diagramme de *Karnaugh* de la retenue est donné par la Figure 3.41.

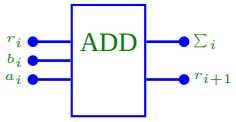

Figure 3.39: *Le synoptique de l'additionneur complet.*

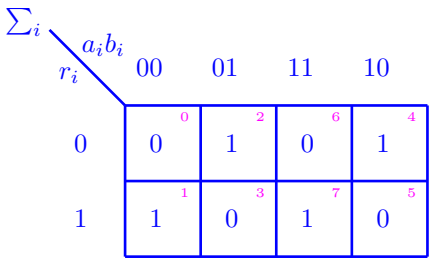

Figure 3.40: *Diagramme de Karnaugh pour l'opération logique* $\sum_i = a_i \oplus b_i \oplus r_i$.

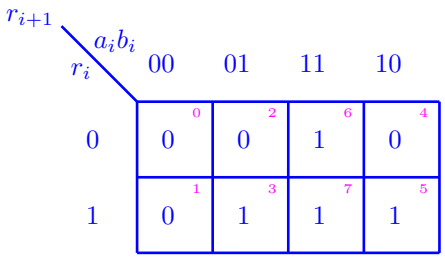

Figure 3.41: *Diagramme de Karnaugh pour l'opération logique* $r_{i+1} = a_i \bullet b_i + r_i \bullet (a_i \oplus b_i)$.

Ce qui donne le schéma de réalisation de la Figure 3.42.

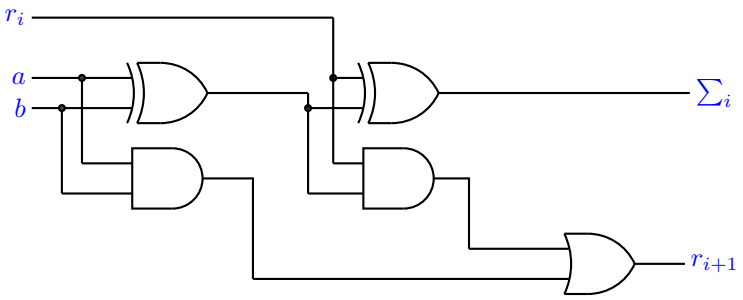

Figure 3.42: *Le circuit de l'additionneur complet.*

L'addition de deux mots de n bits nécessite n additionneurs complets. La retenue appliquée sur les plus faibles poids est nulle et chaque retenue calculée est appliquée au chiffre de poids immédiatement supérieur, comme l'illustre l'équation 3.5.1 et le circuit d'addition de n bits est donné par la Figure 3.43.

$$\begin{array}{cccccc} & & & & r_0 = 0 & \\ & & a_{n-1} & \dots & a_0 & \\ \oplus & & b_{n-1} & \dots & b_0 & \\ & & \Downarrow & & \Downarrow & \\ \sum_n & \sum_{n-1} & & & \sum_0 & \end{array} \tag{3.5.1}$$

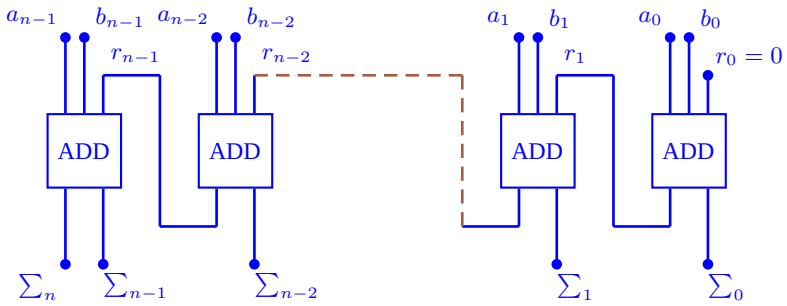

Figure 3.43: *Le circuit d'addition de n bits.*

Cette solution est intéressante d'un point de vue du matériel parce qu'elle est répétitive. Par contre, comme le résultat d'une addition ne peut pas être obtenu instantanément, le temps maximum mis pour obtenir le résultat est directement proportionnel au nombre d'additionneurs. En effet, après le premier temps de calcul la retenue r_1, est appliquée au second additionneur. Ce n'est qu'après le second temps de calcul que la retenue r_2, est délivrée et ainsi de suite, jusqu'au dernier additionneur. Pour cette raison, l'additionneur ainsi réalisée porte le nom d'*additionneur à propagation de la retenue* ou *additionneur à retenue série*. Pour éliminer cet inconvénient, la seconde technique consiste à calculer toutes les retenues en parallèle, directement à partir des données sans même calculer les sommes partielles. Le circuit ainsi réalisé est alors appelé *additionneur à retenue anticipée*. En reprenant le tableau de Karnaugh relatif au calcul de la retenue, le résultat de la minimisation est l'équation 3.5.2.

$$r_{i+1} = a_i \bullet b_i + r_i \bullet (a_i \oplus b_i) \tag{3.5.2}$$

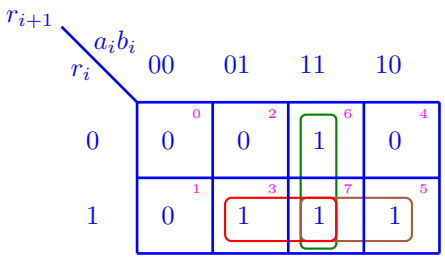

Figure 3.44: *Le diagramme de Karnaugh pour l'additionneur à retenue anticipée.*

Afin d'éviter des temps de calcul cumulables, il ne faut pas utiliser la relation en tant que relation de récurrence, c'est-à-dire qu'il ne faut pas utiliser un résultat de calcul pour le calcul suivant. Il faut systématiquement recalculer chaque terme, ce qui donne, en posant comme résultat les équations 3.5.3 et 3.5.4 et le circuit de la Figure 3.45.

$$\begin{aligned} S_i &= a_i + b_i \\ P_i &= a_{ib_i} \end{aligned} \tag{3.5.3}$$

$$r_i = P_0 + r_0 S_0 \tag{3.5.4}$$

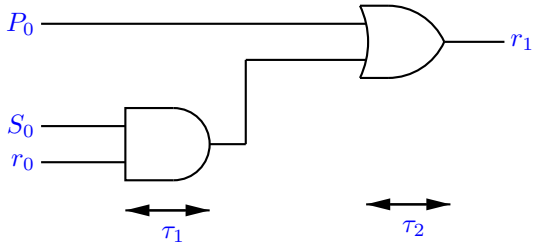

Figure 3.45: *Le circuit de la retenue anticipée* r_1.

De même pour la retenue r_1, nous avons l' équation 3.5.5 et le circuit de la Figure 3.46.

$$r_2 = P_1 + r_1 S_1 = P_1 + S_1(P_0 + r_0 S_0) = P_1 + P_0 S_1 + r_0 S_0 S_1 \tag{3.5.5}$$

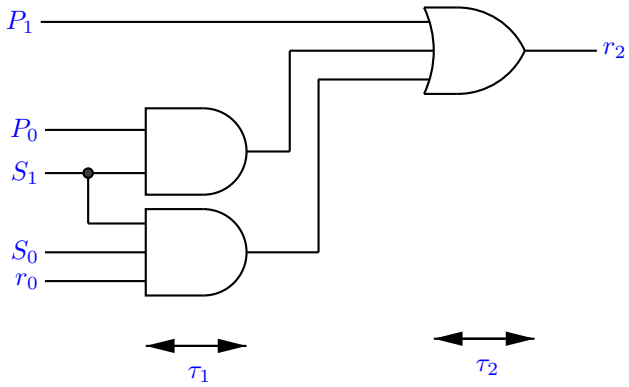

Figure 3.46: *Le circuit de la retenue anticipée* r_2.

Et ainsi de suite, pour la retenue r_3 (équation 3.5.7) et r_4 (équation 3.5.8).

$$r_3 = P_2 + r_2 S_2 = P_2 + S_2(P_1 + P_0 S_1 + r_0 S_0 S_1) \tag{3.5.6}$$

$$r_3 = P_2 + P_1 S_2 + r_0 S_0 S_1 S_2 \tag{3.5.7}$$

$$r_4 = P_3 + r_3 S_3 = P_3 + P_2 S_3 + P_1 S_2 S_3 + P_0 S_1 S_2 S_3 + r_0 S_0 S_1 S_2 S_3 \tag{3.5.8}$$

Nous constatons que les temps de calcul des retenues sont tous égaux. Ils correspondent au temps de transit de l'information dans une porte *ET* (τ_1) et une porte *OU* (τ_2) en cascade, i.e., le nombre d'entrées d'une porte n'affectant pas son temps de transit. La structure d'un additionneur de 4 bits, utilisant la technique de calcul anticipé des retenues, est illustrée par la Figure 3.47. La Table 3.13 montre la comparaison des retenues propagée et anticipée.

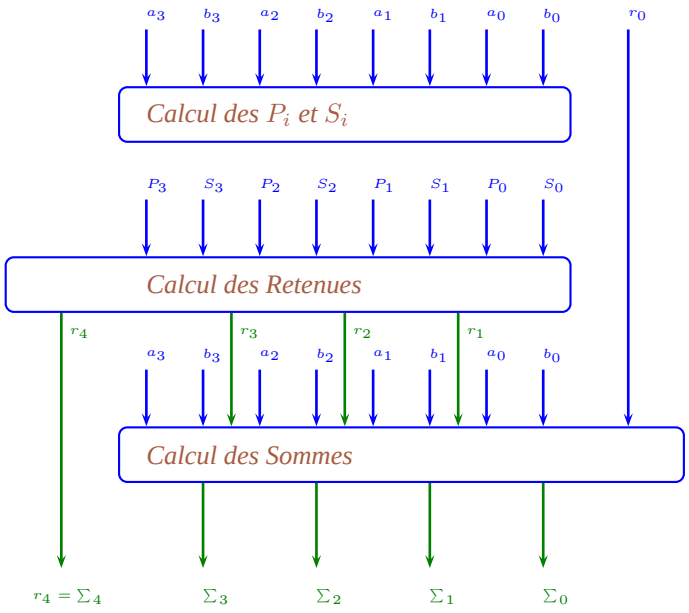

Figure 3.47: *Le circuit d'un additionneur complet à retenue anticipée de* 4 *bits.*

Format des mots	*Temps de calculs en ns (Logique TTL série N)*		
	Propagation de la retenue	*Retenue anticipée*	
4 bits	24	24	*avec additionneur 4 bits intégré*
8 bits	36	36	*avec utilisation d'un générateur de retenue*
12 bits	48	36	*avec utilisation d'un générateur de retenue*
16 bits	60	36	*avec utilisation d'un générateur de retenue*
64 bits	192	60	*avec deux générateurs de retenue en cascade*

Table 3.13: *Comparaison des retenues propagées et anticipées.*

3.5.2 *Soustraction*

Pour la soustraction, nous nous ramenons à une addition. Le nombre négatif est codé en code complément à 2. D'abord, nous faisons le complément de valeur et ensuite l'addition de la valeur 1 à la valeur complémentaire. Il faut observer que la soustraction est une addition numérique et non logique.

Exemple 3.5.1 *Soit la valeur logique* $B = 1001$. *Le complémentaire est* $B' = 0110$, *ensuite, nous faisons l'addition de la valeur unitaire. Donc, le résultat de* B *en complément à deux sera* $B = 0111$.

3.5.3 *Comparateur*

Un comparateur est un dispositif capable de détecter l'égalité de deux nombres et éventuellement d'indiquer le nombre le plus grand ou le plus petit. Pour effectuer la comparaison de deux nombres

A et B, deux techniques sont couramment utilisées. La première, la soustraction de deux nombres. Si le résultat de l'opération $A - B$ est positif, cela signifie que A est supérieur à B. Si le résultat est nul, les deux nombres sont égaux. La seconde, une comparaison *bit* à *bit*. C'est cette méthode qui est utilisée dans la plupart des circuits intégrés commercialisés. La comparaison s'effectue poids à poids en commençant par le chiffre le plus significatif.

Les nombres A et B ayant le même format, le nombre A est forcément supérieur à B si son élément binaire le plus significatif (*Most Significatif Bit - MSB*) est supérieur au *MSB* de B. Si ces deux bits sont égaux, la supériorité (ou l'infériorité) ne peut être déterminée que par l'examen des bits de poids immédiatement inférieur et ainsi de suite. L'examen des poids successifs s'arrête dès que l'un des éléments binaires est supérieur ou inférieur à l'autre. Les deux nombres A et B sont égaux si, après avoir examiné tous les éléments binaires, il n'a pas été détecté de supériorité ou d'infériorité.

Comparateur Donnant l'Égalité de Deux Nombres

C'est le comparateur le plus simple. Deux nombres sont égaux si tous les chiffres sont égaux deux à deux. Pour détecter l'égalité de deux éléments binaires, un opérateur *OU Exclusif* complémentaire est indispensable. Un opérateur *ET* indique la simultanéité de toutes les inégalités partielles.

Soient deux nombres A et B de quatre éléments binaires chacun, $A = a_3a_2a_1a_0$ et $B = b_3b_2b_1b_0$. Le $A = B$ si $(a_3 = b_3)$*ET* $(a_2 = b_2)$ *ET*$(a_1 = b_1)$ *ET*$(a_0 = b_0)$. Ce qui donne le schéma de la Figure 3.48.

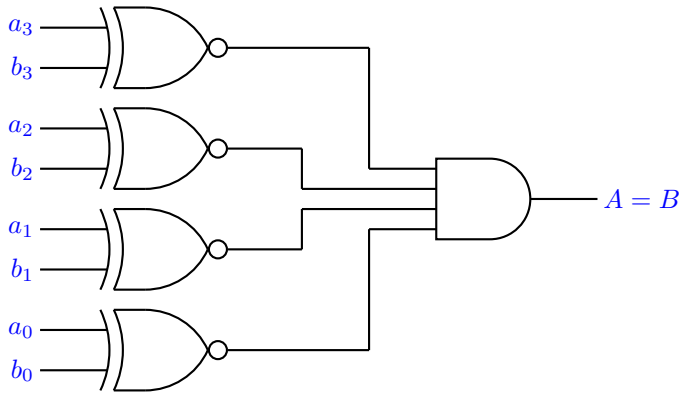

Figure 3.48: *Le comparateur bit à bit* .

Comparateur Complet

Par analogie avec l'additionneur, la conception d'un comparateur complet pour des nombres de quatre éléments binaires peut se faire de deux façons différentes.

⇒ Première Solution

En cascade, c'est-à-dire avec propagation des égalités partielles. Les poids de A et de B sont comparés en commençant par le plus élevé. La comparaison sur les poids faibles ne peut être faite que si tous les bits de poids plus élevés sont égaux deux à deux. La cellule élémentaire de comparaison comporte trois entrées, les éléments binaires a et b de même poids de chaque nombre et une entrée E pour autoriser la comparaison, ce qui dorme la Table 3.14 de vérité, et l'équation 3.5.9.

E	a_i	b_i	$E_i\ (a_i = b_i)$	$S_i\ (a_i > b_i)$	$I_i\ (a_i < b_i)$	*Commentaire*
0	0	0	0	0	0	*Pas de comparaison*
0	0	1	0	0	0	*Pas de comparaison*
0	1	0	0	0	0	*Pas de comparaison*
0	1	1	0	0	0	*Pas de comparaison*
1	0	0	1	0	0	$E_i = 1$ *si* $a_i = b_i$
1	0	1	0	0	1	$I_i = 1$ *si* $a_i < b_i$
1	1	0	0	1	0	$S_i = 1$ *si* $a_i > b_i$
1	1	1	1	0	0	$E_i = 1$ *si* $a_i = b_i$

Table 3.14: *Comparaison en cascade.*

$$\begin{array}{rcl} S_i & = & E(a_i\overline{b_i}) \\ I_i & = & E(\overline{a_i}b_i) \\ E_i & = & E\overline{(S_i + I_i)} \end{array} \qquad (3.5.9)$$

D'où le schéma d'une cellule de comparaison, notée C_i est illustrée par la Figure 3.49.

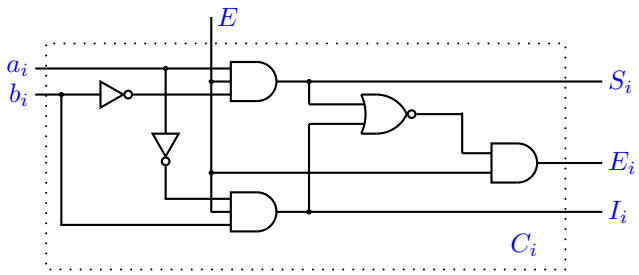

Figure 3.49: *La cellule* C_i.

L'entrée d'autorisation E est en fait la détection d'égalité des éléments binaires de poids supérieurs, le schéma de l'ensemble est alors donné par la Figure 3.50.

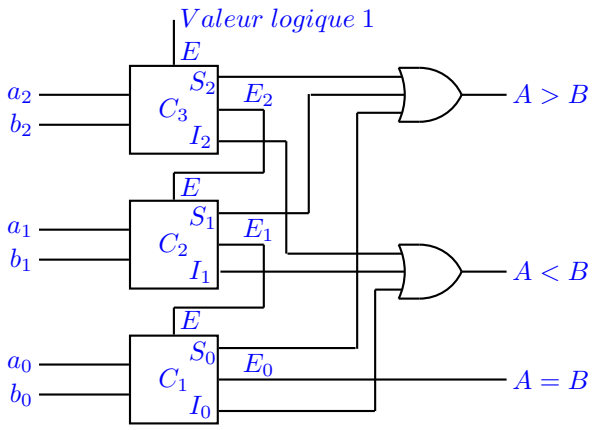

Figure 3.50: *Le circuit comparateur de trois bits.*

Remarque 3.5.1

Comme pour l'additionneur à propagation de la retenue, le résultat de la comparaison apparaît après un temps directement lié aux nombres de cellules à traverser à cause de la mise en cascade i.e., calcul série. Pour palier cet inconvénient, c'est une structure parallèle qu'il faut adopter.

⇒ Deuxième Solution

Comparaison parallèle. Tous les éléments binaires de même poids sont systématiquement et simultanément comparés. Le blocage s'effectue alors sur les résultats de chaque comparaison. La cellule élémentaire de C_i est illustrée par la Figure 3.51.

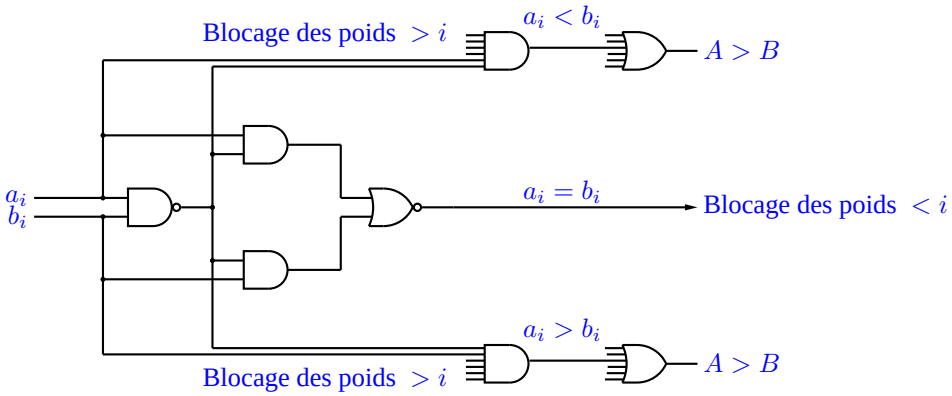

Figure 3.51: *La cellule C_i d'un comparateur parallèle.*

Le blocage des sorties $b_i > a_i$ ou $b_i < a_i$ se fait par une porte *ET* recevant toutes les sorties détectant les égalités $b_j = a_j$ des poids supérieurs au rang i ($\forall j > i$). Le nombre d'entrées de cette porte augmente donc au fur et à mesure que l'on s'éloigne du *MSB*. L'information $A = B$ est fournie par une porte *ET* vérifiant la simultanéité des égalités par partielles. Le schéma de la Figure 3.52 et le symbole illustré par la Figure 3.53 du circuit $SN7485$ de la *Texas Instruments*, montrent l'ensemble d'un comparateur de 4 bits en cascade. Les entrées $a < b$, $a = b$ et $a > b$, dites entrées

de mise en cascade, sont représentatives des résultats des comparaisons sur les éléments binaires d'indice inférieur. Ainsi, pour effectuer la comparaison de deux nombres de huit éléments binaires, nous adoptons le montage de la Figure 3.54.

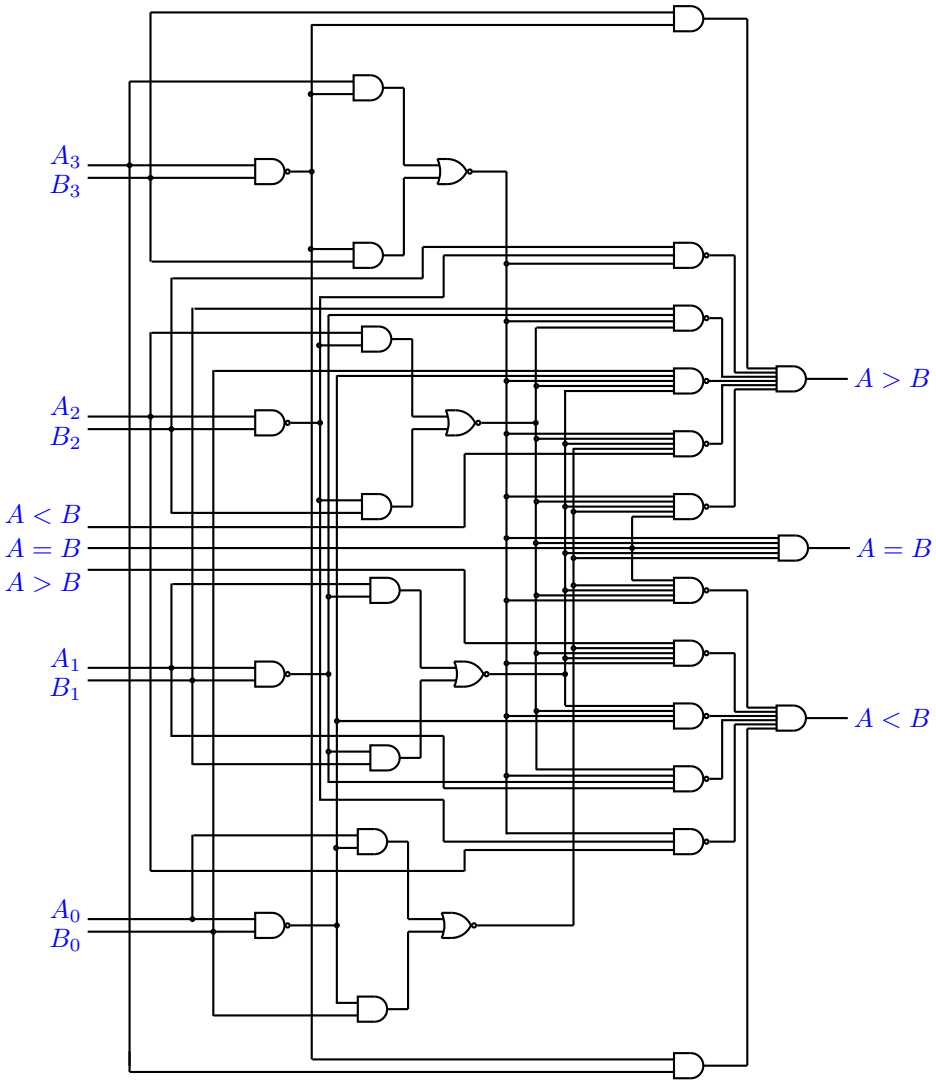

Figure 3.52: *Le comparateur* $SN7485$.

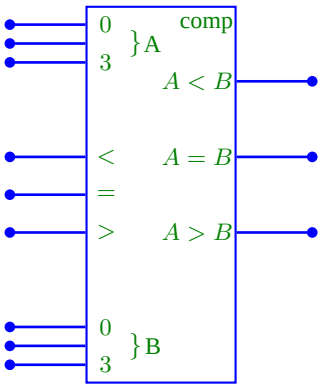

Figure 3.53: *Le symbole du comparateur* $SN7485$.

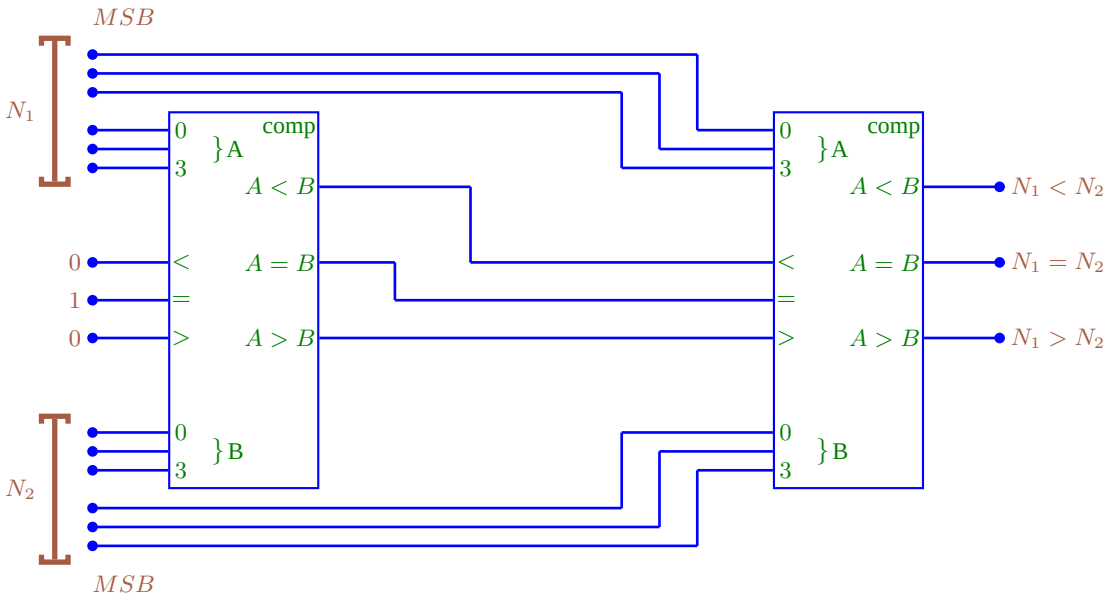

Figure 3.54: *Le comparateur parallèle en cascade.*

3.5.4 *Générateur de Parité*

Nous appelons parité d'un mot binaire N, le nombre de 1 contenu dans ce mot. Le mot a une parité *paire* si ce nombre de 1 est pair. Afin de rendre les transmissions numériques plus robustes au bruit, nous adjoindrons un bit à tous les mots transmis. Ce bit dit de parité, est choisi de façon à ce que le mot complet formé du mot et du bit de parité ait une parité *paire* (dans le cas de la parité paire). Le principe utilisé pour engendrer ce bit de parité repose sur la propriété du *OU Exclusif* i.e., $a \oplus b \oplus c \oplus ... \oplus m$ vaut 1 si un nombre impair de variables est au niveau 1, comme l'illustre la Figure 3.55.

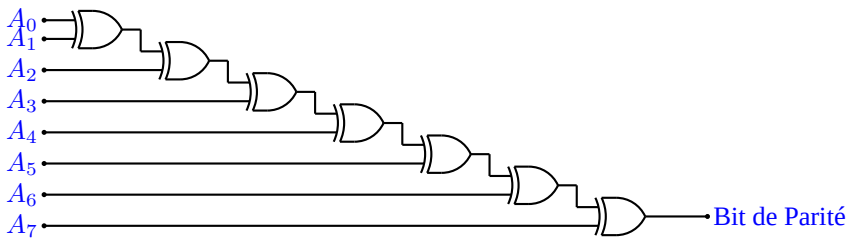

Figure 3.55: *Le générateur de parité.*

3.5.5 *Unité Arithmétique et Logique (UAL ou ALU)*

Ce circuit est utilisé dans quasiment tous les processeurs de calcul. C'est un opérateur capable d'effectuer, comme son nom l'indique, un ensemble de traitements arithmétiques (addition, soustraction, multiplication par 2 avec décalage d'un cran vers la gauche des bits du mot, division par 2 avec décalage d'un cran vers la droite des bits du mot ou opérations logiques tels comme : *ET*, *OU* etc sur des mots binaires de longueur donnée. Le choix de l'opération est déterminé par des bits de commande. C'est donc un opérateur programmable. L'*ALU* résulte d'une grande partie des circuits déjà présentés. Par contre, il est important de comprendre l'action de l'unité arithmétique et logique sur les mots binaires. La Figure 3.56 illustre la représentation fonctionnelle du circuit $SN74181$ et la Figure 3.58 illustre la représentation symbolique logique du composant.

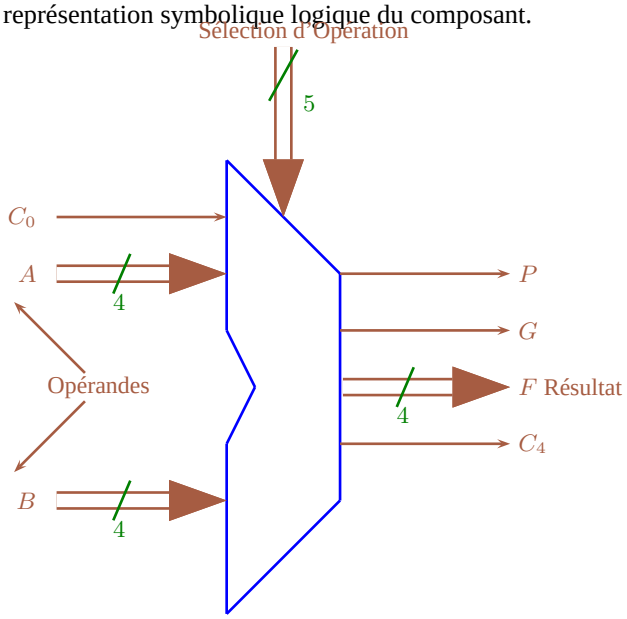

Figure 3.56: *La représentation fonctionnelle du composant* $SN74181$.

Ce circuit intégré utilise des mots de quatre éléments binaires. Les cinq signaux d'entrée de sélection permettent un choix parmi 32 fonctions groupées en 16 opérations arithmétiques et 16 opérations logiques. Indépendamment de la fonction réalisée, ce circuit dispose d'une sortie détectant l'égalité des données en entrée. Lors des opérations arithmétiques sur des nombres de plus de quatre bits, il existe la possibilité de mise en cascade des boîtiers avec la technique de la propagation

de la retenue i.e., C_0 retenue entrante et C_4 retenue sortante, comme l'illustre la Figure 3.57.

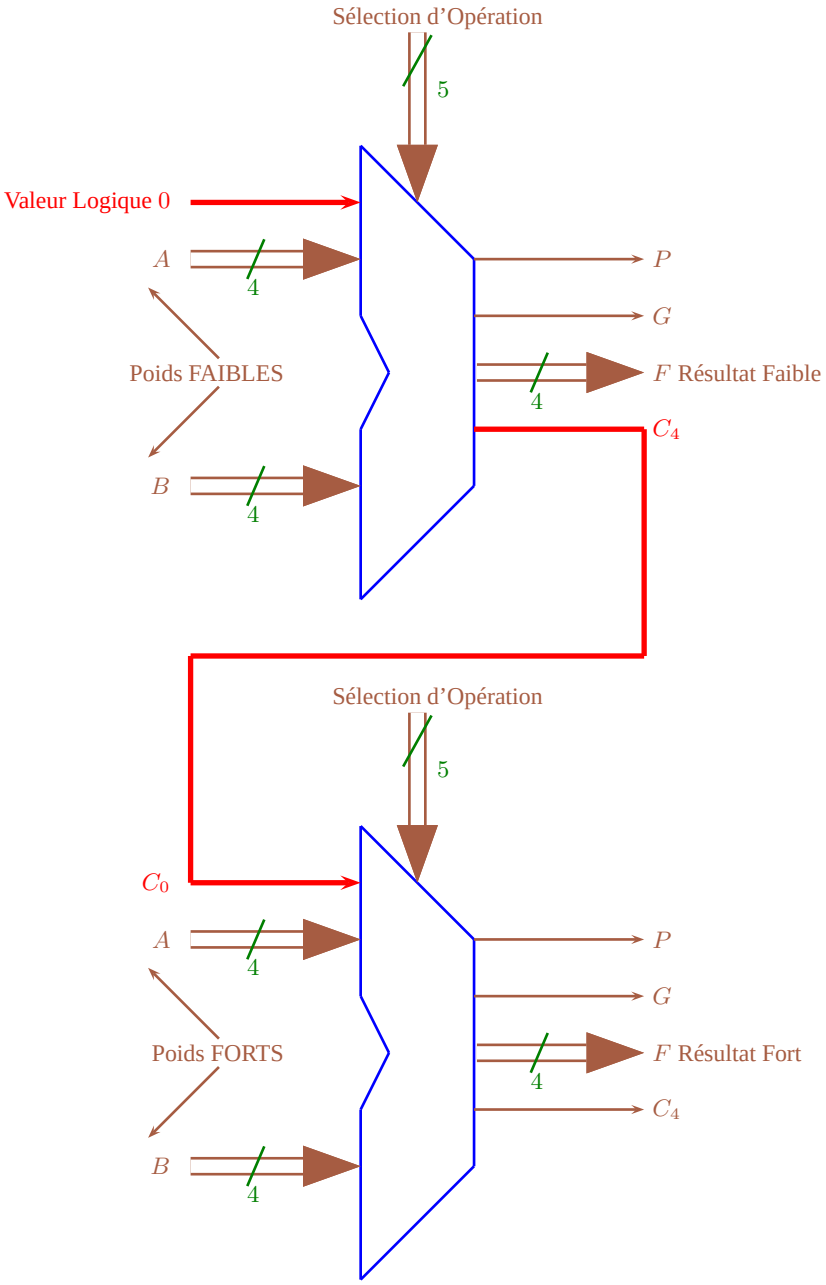

Figure 3.57: *La connexion en cascade du composant* $SN74181$.

Nous pouvons également utiliser la technique de la retenue anticipée en utilisant un circuit supplémentaire spécialisé dans le calcul des retenues i.e., nous utilisons les entrées P et G, comme l'illustre la Figure 3.59.

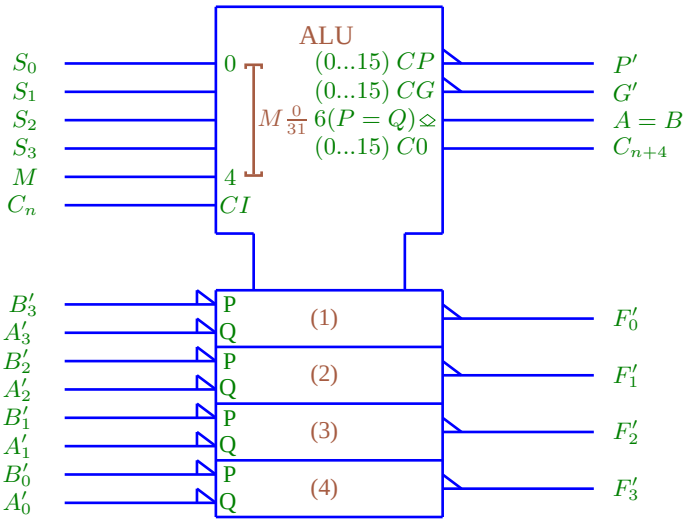

Figure 3.58: *Le symbole logique du composant* $SN74181$.

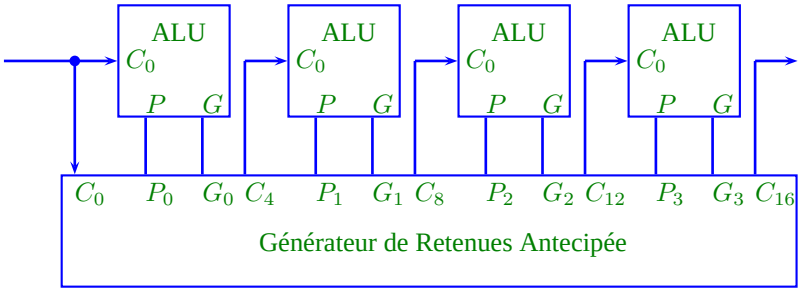

Figure 3.59: *La connexion en cascade du composant* $SN74181$ *avec le calcul des retenues anticipées.*

3.6 *Conclusion*

Nous venons d'étudier différentes applications de la logique combinatoire. Nous avons observer que la sortie est fonction seulement des entrées des circuits i.e., les circuits n'ont pas de mémoire. Nous verrons dans le prochain chapitre la logique séquentielle qui permet de stocker les informations et générer tous les contrôles pour les circuits combinatoires.

Chapitre 4

Logique Séquentielle

4.1 *Introduction*

Les circuits pratiques incluent quelques éléments de mémoire, lesquels sont décrits par la logique séquentielle i.e., la sortie est fonction des entrées et de l'état antérieur de la sortie. Nous étudierons la connexion des portes logiques combinatoires qui permettent de construire les circuits séquentiels i.e., les circuits logiques avec mémoire.

4.1.1 *Système Logique Séquentiel*

A l'instant discret $\mathbf{n}$, une sortie $\mathbf{S_j^n}$ d'un système logique séquentiel, comme l'illustre la Figure 4.1, dépend de ses entrées $\mathbf{e_1^n}, ..., \mathbf{e_p^n}$ mais aussi de l'état antérieur des sorties $\mathbf{S_1^{n-1}}, ..., \mathbf{S_m^{n-1}}$ qui peuvent être considérées comme des entrées secondaires, alors que les entrées $\mathbf{e_1^n}, ..., \mathbf{e_p^n}$ sont appelées primaires.

$$S_1 j^n = f(e_1^n, ..., e_p^n, S_1^{n-1}, ..., S_m^{n-1}) \text{ ou } 1 \leq j \leq m \tag{4.1.1}$$

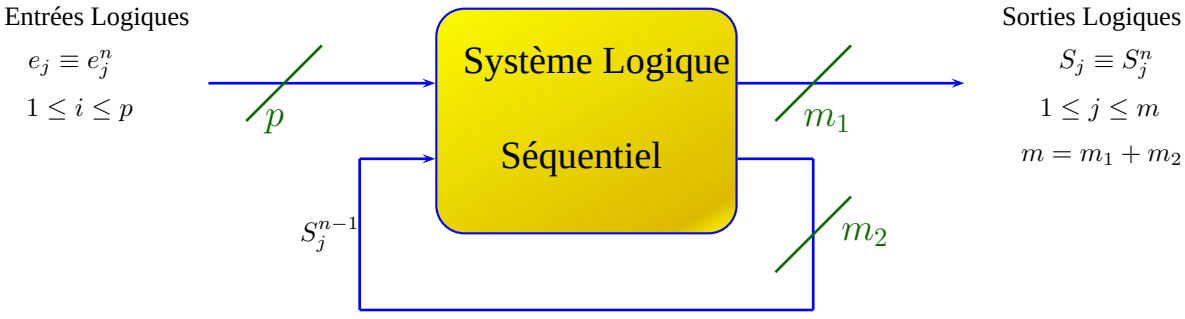

Figure 4.1: *Schéma du système logique séquentiel.*

Remarque 4.1.1 *Notion de mémoire, car les systèmes séquentiels sont bouclés, ou encore récursifs et la seule connaissance des entrées (primaires) ne suffit pas à déterminer l'état des sorties.*

Exemple 4.1.1

*Supposons $s_i = 0$ initialement (*état initial lié à la technologie employée*). Donc, si le contenu est la mémorisation de e_1 i.e., dès que e_1 passe à 1, s_1 passe à 1 et y reste ensuite pour tous les e_1, comme l'illustre la Figure 4.2.*

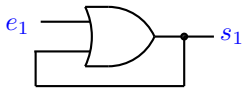

Figure 4.2: *Le principe de la mémoire.*

Définition 4.1.1 *La bascule est un circuit séquentiel dont la sortie possède 2 états stables. Le circuit est nommé aussi de* Bistable *ou* Flip-Flop.

4.2 *Les Bascules*

4.2.1 *La Bascule RS*

Soit un système séquentiel type bascule RS , où l'entrée R qui est nommée *Reset* pour mettre la sortie au valeur logique 0 et l'entrée S qui est nommée *Set* pour mettre la sortie au valeur logique 1, comme l'illustre la Figure 4.3.

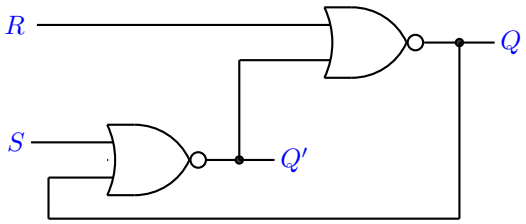

Figure 4.3: *Le circuit de la bascule RS.*

Le circuit peut être autrement représenté, comme l'illustre la Figure 4.4, et par exemple, par la séquence d'entrée comme le montre la Figure 4.5.

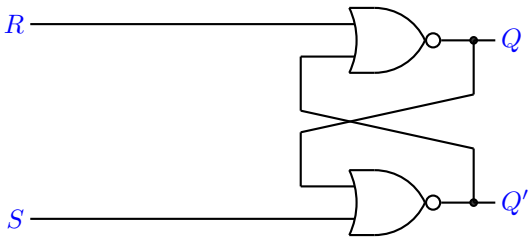

Figure 4.4: *L'implémentation de la bascule RS à partir de portes logiques NOR.*

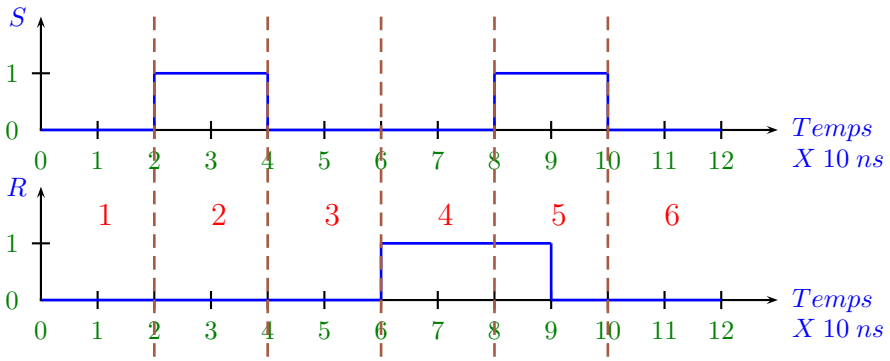

Figure 4.5: *La séquence des signaux d'entrée d'une bascule type RS.*

A partir de la séquence du signal entrée, nous devons faire l'analyse du comportement de la bascule, qui est illustrée par la Table 4.1. Nous supposons que l'état initial de la sortie Q est au repos i.e., $Q = 0$ et la sortie $\overline{Q} = 1$.

Phase	S	R	Q	Observation	Conclusion	États
1	0	0	0	État initial : $Q \equiv 0$ et $Q' \equiv 1$	Mémorisation de $Q_n = Q_{n-1}$	Stables
2	1	0	1	$S = 1$ donc $Q' = 0$ et $Q = 1$	Set de Q	Stables
3	0	0	1	$S = 0$ donc $Q' \equiv 0$ et $Q \equiv 1$	Mémorisation de $Q_n = Q_{n-1}$	Stables
4	0	1	0	$R = 1$ donc $Q = 0$ et $Q' = 1$	Reset de $Q' = \overline{Q}$	Stables
5	1	1	0	$S = 1$ donc $Q' = 0$ et $Q \equiv 0$	Combinaison interdite, car $Q' \neq \overline{Q}$	Instables

Table 4.1: *Les opérations de la bascule RS.*

Pour la phase 1, nous avons $S = 0$ et $R = 0$, donc pour l'aléa de fonctionnement, l'état Q dépend de la rapidité relative entre les deux portes logiques car les deux entrées S et R changent d'état simultanément. Si la porte *NOR* d'entrée R est plus rapide que celle d'entrée S, nous avons $Q = 1$ et $\overline{Q} = 0$. Sinon, nous avons $Q = 0$ et $\overline{Q} = 1$, dans les deux cas $Q' = \overline{Q}$ et les états sont stables.

Remarque 4.2.1

$\Rightarrow$ 1. *La combinaison d'entrées ($S = 1$ et $R = 1$) de la phase 5 est à proscrire car elle ne conduit pas à $\overline{Q} = Q$ i.e., les bascules ont leurs sorties complémentaires Q et $\overline{Q}$.*

$\Rightarrow$ 2. *Les configurations pour lesquelles les 2 entrées changent d'état simultanément (*comme à la phase 6*) sont à proscrire car elles conduisent à un aléa de fonctionnement.*

4.2.2 *Table de Vérité de la Bascule RS*

Q_n représente l'état stable de Q à l'instant discret n et Q_{n-1} représente l'état stable de Q à l'instant précédant la configuration d'entrée courante i.e., l'état précédent de la sortie Q avant

changement des entrées aux nouvelles valeurs que sont les valeurs courantes spécifiées. Ce changement place Q à l'état Q_n.

La restriction de fonctionnement de la bascule *RS* est la suivante : 1 seule des 2 (deux) entrées doit changer d'état à la fois. Si les 2 entrées changent d'état en même temps (*impossible cependant en asynchrone*), donc l'aléa pour Q, comme l'illustre la Table 4.2.

S	R	Q_n	*Fonction*
0	0	Q_{n-1}	*Mémorisation*
0	1	0	*Reset (Mise à 0 de Q)*
1	0	Q_{n-1}	*Set (Mise à 1 de Q)*
1	1		*Combinaison interdite*

Table 4.2: *Le fonctionnement de la bascule RS.*

Nous pouvons aussi faire l'implémentation de la bascule *RS* à partir des portes logiques *NAND*, comme l'illustre la Figure 4.6.

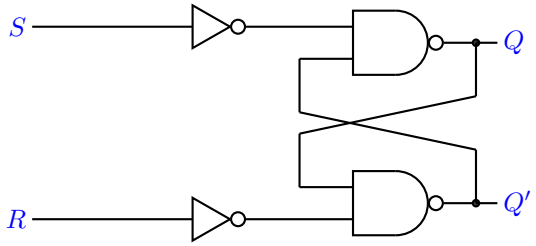

Figure 4.6: *L'implémentation de la bascule RS à partir de portes logiques NAND.*

La combinaison interdite engendre $Q = \overline{Q} = 1$, contrairement à la réalisation à portes *NOR* pour laquelle elle engendre $Q = \overline{Q} = 0$. Le symbole de la bascule *RS* est donné par la Figure 4.7. La bascule *RS* est l'élément de base de la logique séquentielle. C'est la seule bascule asynchrone.

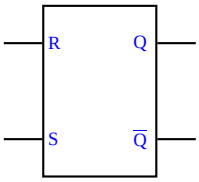

Figure 4.7: *Le symbole de la bascule RS.*

Définition 4.2.1

*En asynchrone, la sortie de la bascule change d'état uniquement en fonction des grandeurs d'entrée. Le système livré à lui-même, est ainsi plus rapide que les systèmes synchrones, mais il présente des temps de propagation (*délais*) difficiles à maîtriser. Donc, nous préférons l'utilisation de systèmes synchrones.*

Définition 4.2.2

La prise en compte des entrées est conditionnée par une autorisation donnée par un signal d'horloge. Ainsi, les entrées du système sont prises en compte (provoquant alors l'état de sortie correspondant) uniquement s'il y a autorisation par l'horloge i.e., l'horloge est alors dite active. Sinon, pas d'autorisation de la part de l'horloge, les entrées sont ignorées et leur changement d'état ne peut entraîner le basculement de la sortie : celle-ci demeure à son état antérieur.

4.2.3 *Synchronisation*

L'autorisation ou la synchronisation de l'horloge peuvent se faire de façons différentes : - synchronisation sur niveau (*latch*), synchronisation sur front ou flanc ou transition (*edge triggered*) et synchronisation par impulsion (*pulse triggered*).

Synchronisation sur Niveau

Il suffit d'appliquer le niveau logique convenable, dit niveau actif, sur l'entrée d'horloge, pour que la sortie de la bascule puisse réagir aux entrées de données ou H est le signal d'horloge (*noté aussi CLK*) et D est l'entrée de donnée, comme illustrent les Figures 4.8 et 4.9.

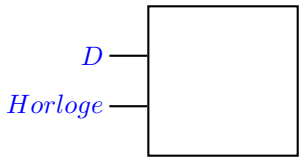

Figure 4.8: *Synchronisation sur niveau haut ou positive latch.*

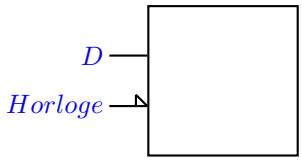

Figure 4.9: *Synchronisation sur niveau bas ou négative latch.*

Synchronisation sur Front

La sortie de la bascule réagira aux entrées de données à l'instant où se produit un front i.e., une transition d'état de l'horloge. Ce front actif peut être montant (*positif*) ou descendant (*négatif*), comme les illustrent les Figures 4.10 et 4.11.

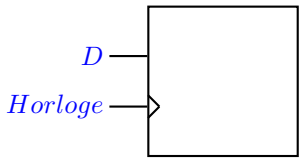

Figure 4.10: *Synchronisation sur front montant positif* positive edge triggered.

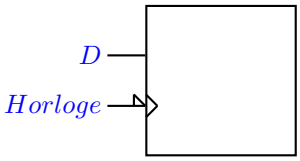

Figure 4.11: *Synchronisation sur front descendant négatif* negative edge triggered.

Synchronisation par Impulsion

Une impulsion de synchronisation de l'horloge, comme l'illustre la Figure 4.12, est composée de 2 fronts. L'un positif et l'autre négatif. Le premier front sert à la synchronisation des entrées, le second front sert à la synchronisation des sorties. Ce type de synchronisation est utilisé pour les systèmes maître-esclave.

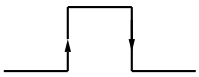

Figure 4.12: *L'impulsion montant et descendant.*

4.3 *Les Bascules Synchrones*

La bascule *RS* synchronisée (*RST*) constitue une amélioration de la bascule *RS* asynchrone. Bien que plus lents que les systèmes asynchrones i.e., il faut attendre la validation d'horloge pour prendre en compte les données. Les systèmes synchrones ont l'avantage d'introduire un certain déterminisme i.e., prévision, régularité, maîtrise des séquences dans les traitements et sont ainsi beaucoup plus utilisés que les systèmes asynchrones.

4.3.1 *La Bascule RST*

Bascule *RST* synchronisée sur niveau haut de l'horloge T, comme l'illustre la Figure 4.13 et Table de fonctionnement 4.3.

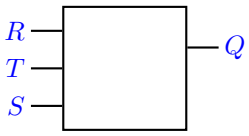

Figure 4.13: *Le symbole de la bascule RST.*

$Horloge\ T$	T	S	R	Q_n	*Fonction*
$Inactive$	0	X	X	Q_{n-1}	*Mémorisation*
$Active$	1	0	0	Q_{n-1}	*Mémorisation*
$Active$	1	0	1	1	*Set (Remise à 1 de Q)*
$Active$	1	1	0	0	*Reset (Remise à 0 de Q)*
$Active$	1	1	1	⊠	*Interdit*

Table 4.3: *Le fonctionnement de la bascule RST.*

Remarque 4.3.1

$\Rightarrow$ 1. *Si* $T = 0$ *la sortie ne change pas quelles que soient les entrées* R *et* S*. C'est le fonctionnement en mémoire. La bascule n'est pas synchronisée.*

$\Rightarrow$ 2. *Si* $T = 1$ *la bascule est alors synchronisée. Sa sortie respecte la table de fonctionnement de la bascule* RST *avec les mêmes restrictions.*

Exemple 4.3.1 *Soit la séquence d'entrée et sortie de la Figure 4.14, pour une bascule* RST*, synchronisée sur niveau haut de l'horloge* T*.*

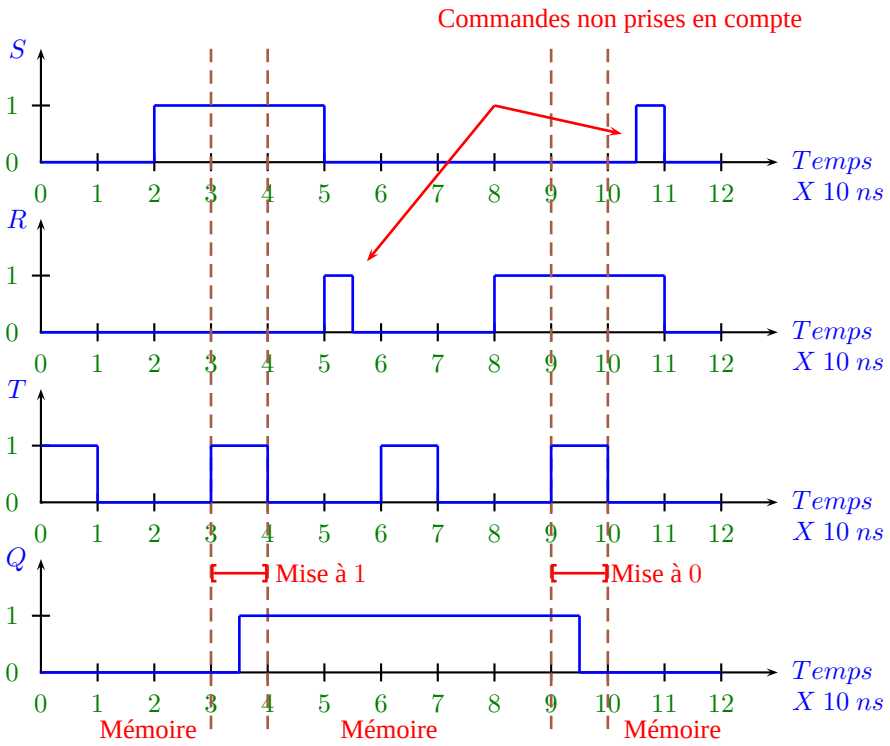

Figure 4.14: *Le diagramme des entrées et de la sortie pour la bascule RST latch.*

Le léger retard de Q *par rapport à* S *et* R *témoigne du temps de réponse du circuit. Étant donné que pour la combinaison* $R = S = 0$ *avec* $T = 1$*, nous avons un fonctionnement identique à la combinaison* $T = 0$ *quels que soient* R *et* S*, il faut fabriquer deux variables* r *et* s *entrées d'une bascule* RS *asynchrone telles que les Tables de vérité 4.4 et 4.5 soient vérifiées.*

T	R	r	Fonction
0	0	0	Mémorisation
0	1	0	Mémorisation
1	0	0	Mémorisation
1	1	1	Mise à 0

Table 4.4: *La table de vérité pour les signaux* T*,* S *et* s*.*

T	S	s	Fonction
0	0	0	Mémorisation
0	1	0	Mémorisation
1	0	0	Mémorisation
1	1	1	Mise à 1

Table 4.5: *La table de vérité pour les signaux T, R et r.*

Ceci est facilement réalisé à l'aide d'une porte ET *qui permet de bloquer les commandes R et S tant que $T = 0$. Le schéma de la bascule* RST *synchronisée sur niveau haut de T peut donc être illustrée par la Figure 4.15 .*

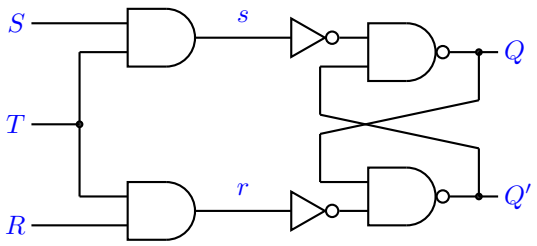

Figure 4.15: *L'implémentation du circuit pour l'exemple.*

Le circuit ET *suivi de l'inverseur peut avantageusement être remplacé par un opérateur* NAND.

4.3.2 *La Bascule D ou Gated D-Latch*

Elle constitue un élément mémoire. Elle supprime la configuration interdite de la bascule *RST*. Elle possède 1 entrée de donnée D (*Data*) et est réalisée à partir d'une bascule *RST* avec les entrées R et S liées par la relation $D = S = \overline{R}$.

Fonctionnement

Cette bascule dispose d'une seule entrée appelée D. Le signal de synchronisation peut être actif soit sur un niveau i.e., la bascule est alors appelée *D latch* , soit sur un front et alors nommée bascule *edge triggered*. C est l'horloge de synchronisation, comme l'illustre la Figure 4.16 et 4.17.

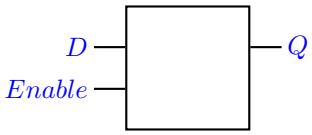

Figure 4.16: *La bascule D synchronisée sur niveau haut.*

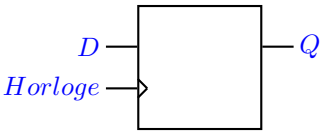

Figure 4.17: *La bascule D synchronisée sur front montant (edge triggered).*

Avec une seule entrée on ne peut trouver que deux modes de fonctionnement suivants :

$\Rightarrow$ 1. Le signal de synchronisation est actif, la sortie Q recopie l'entrée D.

$\Rightarrow$ 2. Le signal de synchronisation n'est pas actif la sortie Q ne change pas.

C'est le fonctionnement en mémoire. Lors du passage en position mémoire, la dernière valeur recopiée est mémorisée.

Exemple 4.3.2 *Soient les séquences d'entrée et sortie de la Figure 4.18 pour une bascule* D *synchronisée sur niveau et pour une bascule* D *synchronisée sur front positif la Figure 4.19.*

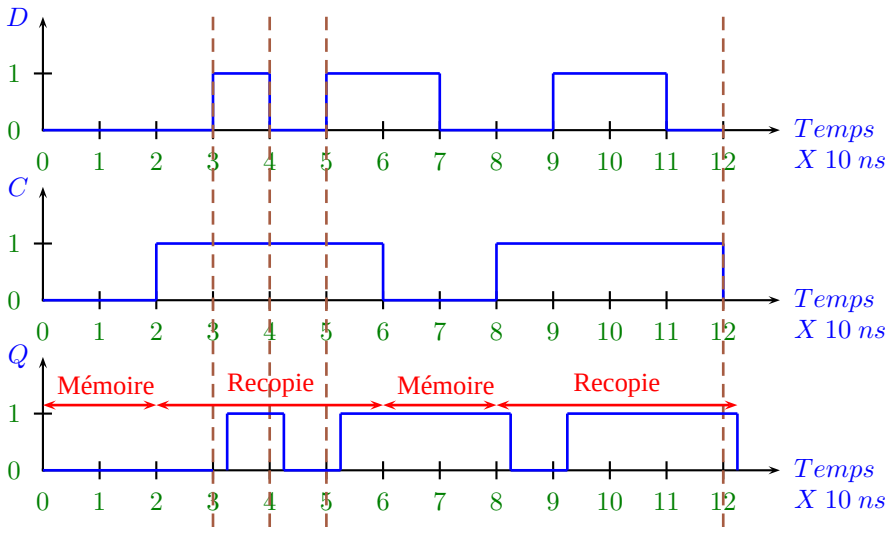

Figure 4.18: *Le temps de propagation au travers de la bascule.*

Le léger retard de Q par rapport à D témoigne du temps de propagation à travers la bascule.

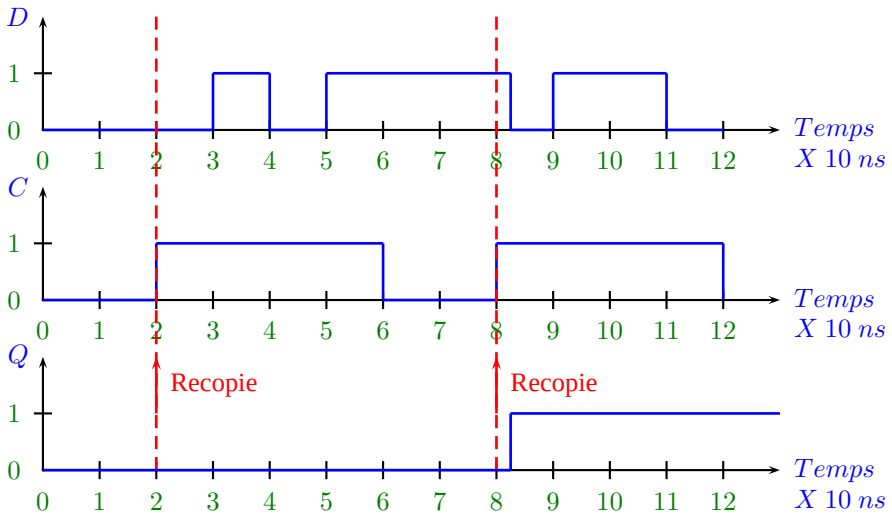

Figure 4.19: *La bascule D (edge triggered) synchronisée sur front positif.*

Le léger retard de Q par rapport à la prise en compte de D témoigne du temps de propagation à travers la bascule. Une bascule D est issue d'une bascule RST *avec les entrées R et S liées par la relation $D = S = \overline{R}$. Pendant la phase où l'horloge de synchronisation est inactive, nous avons $Q_n = Q_{n-1}$ i.e., fonction mémoire. L'équation de fonctionnement dans la phase d'activité de l'horloge est donnée par $Q_n = S + \overline{R}.Q = D_{n-1}$ i.e., fonction recopiée. Le schéma de réalisation est donné par la Figure 4.20 et par les Tables de fonctionnement 4.6 et 4.7.*

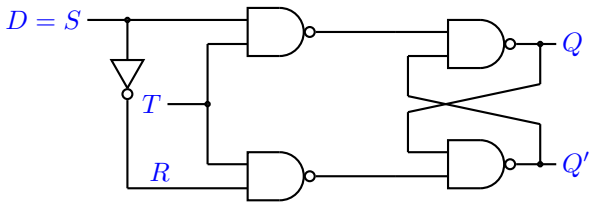

Figure 4.20: *La réalisation de la bascule type D avec synchronisation sur niveau.*

Horloge T	T	D	Q_n	*Fonction*
Inactive	0	X	Q_{n-1}	*Mémorisation*
Active	1	0	0	*Recopie*
Active	1	1	1	*Recopie*

Table 4.6: *Le fonctionnement de la bascule RS.*

Horloge T	T	D	Q_n	*Fonction*
Inactive	0	X	Q_{n-1}	*Mémorisation*
Active	1	$\boxtimes$	D	*Recopie*

Table 4.7: *Le fonctionnement de la bascule RS.*

La discrimination du front, c'est-à-dire du changement de niveau, ne s'effectue pas avec un circuit dérivateur mais par le jeu de trois mémoires internes à la bascule. La réalisation simplifiée d'une bascule D edge triggered *est donnée par la Figure 4.21 et par les Tables de fonctionnement 4.8 et 4.9*

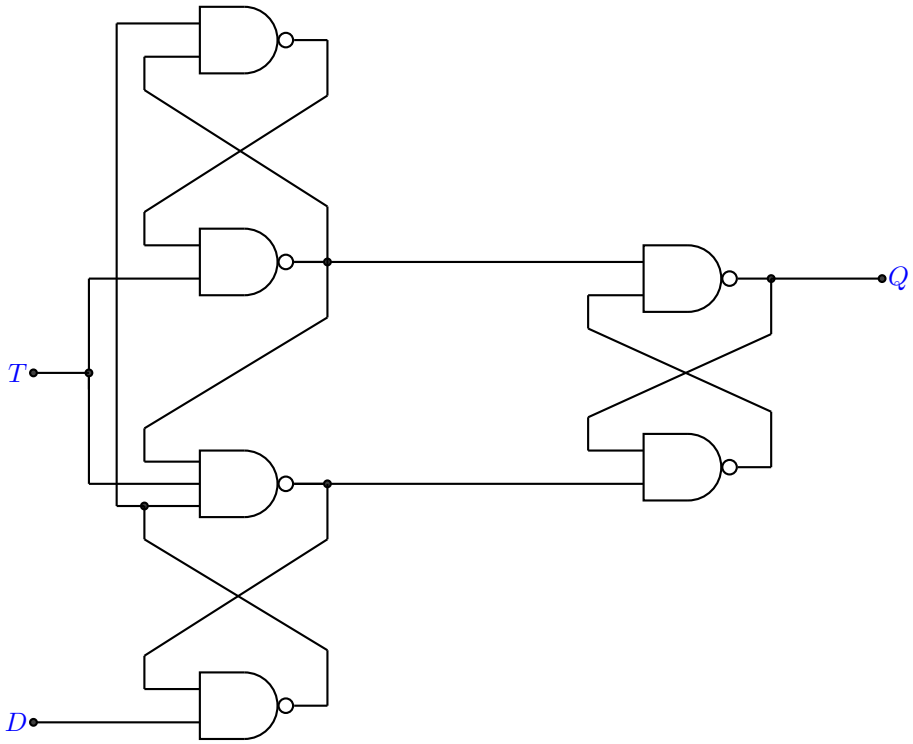

Figure 4.21: *La bascule D edge triggered.*

Horloge T	T	D	Q_n	*Fonction*
Inactive	X (0 ou 1 ou $\downarrow$)	X	Q_{n-1}	*Mémorisation*
Active	$\uparrow$	0	0	*Recopie*
Active	$\uparrow$	1	1	*Recopie*

Table 4.8: *Le fonctionnement de la bascule RS.*

Horloge T	T	D	Q_n	*Fonction*
Inactive	X (0 ou 1 ou $\downarrow$)	X	Q_{n-1}	*Mémorisation*
Active	$\uparrow$	$\boxtimes$	D	*Recopie*

Table 4.9: *Le fonctionnement de la bascule RS.*

Le circuit ET *suivi de l'inverseur peut avantageusement être remplacé par un opérateur* NAND.

L'analyse du fonctionnement de cette bascule peut être faite comme s'il s'agissait d'un système séquentiel asynchrone ayant deux variables d'entrées D et C.

Remarque 4.3.2

La bascule D impose une restriction pour le bon fonctionnement i.e., pour une bascule D latch, *l'entrée D ne doit pas changer d'état pendant que $C = 1$, sinon la bascule aura le problème asynchrone* RS *qui réapparaît. La bascule* JK *va apporter une amélioration pour éviter le problème.*

4.3.3 *Bascule JK*

C'est une bascule disposant de deux entrées, respectivement appelées J et K. Comme pour la bascule RS, l'entrée J sert à la mise à 1 et l'entrée K à la remise à 0. La différence entre la bascule JK et la bascule RS réside dans le fait qu'il n'y a plus d'état interdit pour les entrées, au profit de la combinaison $J = K = 1$ utilisée pour obtenir un fonctionnement type T. La bascule JK est égale à la bascule RS avec $J = S$, $K = R$ et la combinaison $J = K = 1$ est non interdite et de type bascule T. Le symbole de la bascule JK est donnée par la Figure 4.22, le circuit par la Figure 4.23 et par les Tables de vérité 4.10 et 4.11.

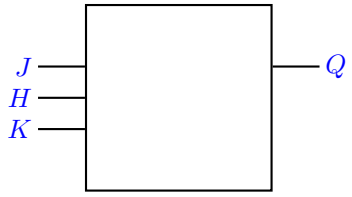

Figure 4.22: *Le symbole de la bascule JK.*

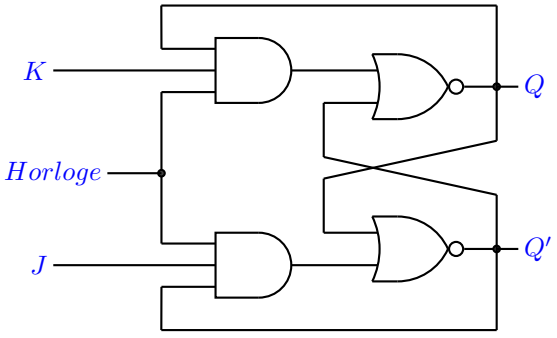

Figure 4.23: *Le circuit de la bascule JK.*

Horloge H	H	J	K	Q_n	*Fonction*
Inactive	X(0 ou 1 ou $\downarrow$)	X	X	Q_{n-1}	*Mémorisation*
Active	$\uparrow$	0	0	Q_{n-1}	*Mémorisation*
Active	$\uparrow$	0	1	0	*Reset (Remise Q à 0)*
Active	$\uparrow$	1	0	1	*Set (Remise Q à 1)*
Active	$\uparrow$	1	1	Q'_{n-1}	*Complémentaire*

Table 4.10: *Le fonctionnement de la bascule JK edge triggered.*

Transition $Q_{n-1} \rightarrow Q_n$	J	K
$0 \rightarrow 0$	0	X
$0 \rightarrow 1$	1	X
$1 \rightarrow 0$	X	1
$1 \rightarrow 1$	X	0

Table 4.11: *La table transitions de la bascule JK edge triggered.*

Remarque 4.3.3 *La table de vérité de la bascule JK montre que nous pouvons la substituer à n'importe quelle autre bascule.*

Conversion de la Bascule JK en RST

Il est facile de fabriquer une bascule *RST* en faisant la correspondance $J = S$, $K = R$ et en s'interdisant $J = K = 1$, comme l'illustre la Figure 4.24.

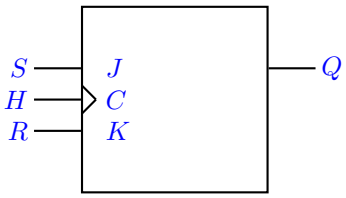

Figure 4.24: *Bascule JK comme bascule RST.*

Conversion de la Bascule JK edge triggered en D edge triggered.

Dans le cas où $J = \overline{K} = D$, nous obtenons une bascule D, comme l'illustre la Figure 4.25.

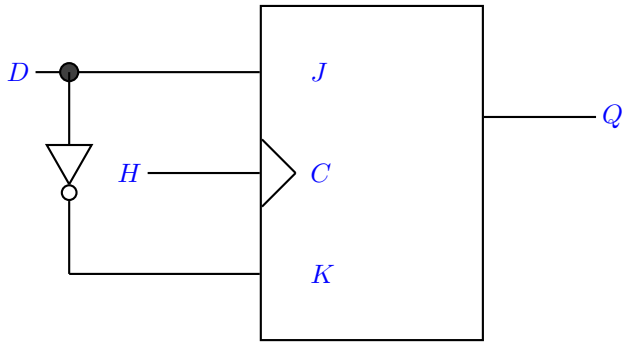

Figure 4.25: *Transformation d'une bascule JK edge triggered en bascule D edge triggered.*

Conversion de la Bascule JK en T

Si $J = K = 1$ ou si $D = \overline{Q}$ i.e., $J = \overline{Q}$ et $K = Q$, la bascule JK fonctionne comme la bascule T (Figure 4.26).

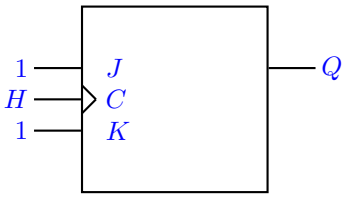

Figure 4.26: *Transformation de la bascule JK en bascule T.*

4.3.4 *Implémentation de la Bascule JK*

A priori une bascule JK est fabriquée à partir d'une bascule RS synchrone où il est fait un rebouclage tel que $S = J.\overline{Q}$ et $R = K.Q$, comme l'illustre la Figure 4.27.

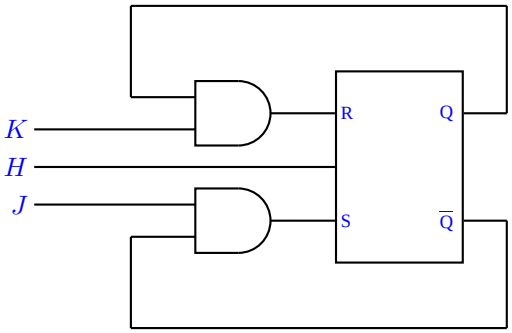

Figure 4.27: *Implémentation de la bascule JK.*

L'équation de fonctionnement de la bascule RS devient $Q_n = S + \overline{R}Q_{n-1} = J\overline{Q_{n-1}} + \overline{K}Q_{n-1}$. A cause du fonctionnement en type T, il y a le risque d'instabilité décrit lors de l'étude de cette bascule. Afin d'éliminer toute instabilité il existe les solutions suivantes :

$\Rightarrow$ 1. Limiter la durée du fonctionnement autonome du système en réduisant la période de sensibilité. Ceci conduit à rendre minimum la largeur de l'impulsion d'horloge. A la limite, cette solution consiste à synchroniser la bascule sur un front (solution utilisable également pour les bascules de type RS ou D).

$\Rightarrow$ 2. Ouvrir la boucle du système. Cette solution impose l'utilisation d'une mémoire intermédiaire pour conserver le résultat précédent alors que la boucle est ouverte. C'est la structure maître-esclave, i.e., une bascule maître du type RS et autre esclave du type bascule D. Cette structure peut aussi être utilisée pour les bascules de type RS ou D.

4.3.5 *Bascule T*

Une bascule type T est une version de la bascule JK qui dispose d'une seule entrée, comme l'illustre la Figure 4.28.

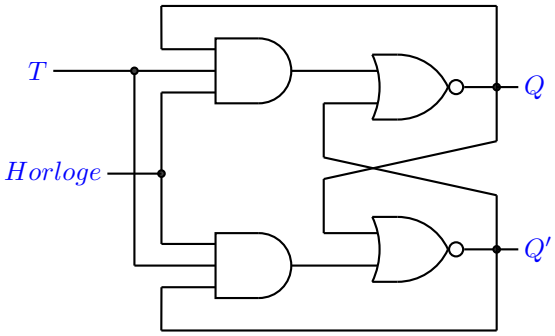

Figure 4.28: *La bascule T edge triggered.*

Exemple 4.3.3 *Transformation d'une bascule* D edge triggered *en bascule* T edge triggered*. L'équation de la bascule* D *étant* $Q_n = D$*, il suffit de relier l'entrée* D *une porte logique XOR, dont une entrée est l'excitation et l'autre est relié à la sortie* $\overline{Q}$ *pour obtenir* $D = T \oplus Q_n$*, comme l'illustre la Figure 4.29 et la Table de vérité 4.12.*

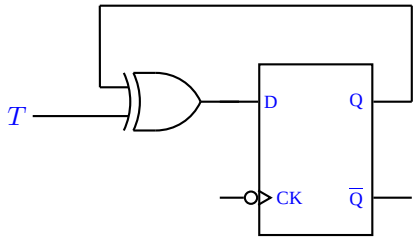

Figure 4.29: *Transformation d'une bascule* D *edge triggered en bascule* T *edge triggered.*

Q_n	T	Q_{n+1}
0	0	0
0	1	1
1	0	1
1	1	0

Table 4.12: *Le fonctionnement de la bascule T.*

4.3.6 *Initialisation d'une Bascule*

En plus des entrées de données, les bascules possèdent des entrées dites asynchrones permettant d'initialiser les sorties, ou même de fixer celles-ci à un état constant quelque soient les entrées de données ou d'horloge.

Les entrées asynchrones A mises à 0 ou mises à 1, souvent actives à l'état bas (donc notées A') conduira la sortie à la valeur logique 0. Par exemple, pour la bascule JK, l'entrée asynchrones mise à 0 conduira la sortie Q à 0, quelles que soient les entrées d'horloge et de données J et K. Ce sont des commandes d'effacement et d'initialisation et qui sont appelées aussi *Clear* et *Preset* ou encore *Reset* et *Set*. Ces entrées peuvent être activées pour fixer l'état initial de la sortie et qui doivent ensuite être inactivées pour permettre le fonctionnement normal de la bascule. Un simple circuit RC connecté à l'entrée *Preset* pour fixer l'état initial Q peut être utilisé.

Les Entrées Asynchrones Actives à l'État Haut

A l'aide d'un circuit RC, nous activons l'entrée asynchrone un court instant au démarrage par un niveau haut, comme l'illustre la Figure 4.30, puis nous ramenons le signal asynchrone d'initialisation à 0. Appliqué à l'entrée *Clear*, ceci initialisera Q à 0, mais appliqué à l'entrée *Preset*, ceci initialisera Q à 1. A chaque fois l'entrée asynchrone non utilisée est placée à l'état inactif soit l'état 0. Le maintien prolongé d'une entrée asynchrone à son niveau actif fixe Q constant i.e., la sortie est mise à 0 si *Clear* est actif, comme l'illustre la Figure 4.31. La sortie est mise à 1 quand le *Preset* est actif, comme l'illustre la Figure 4.32.

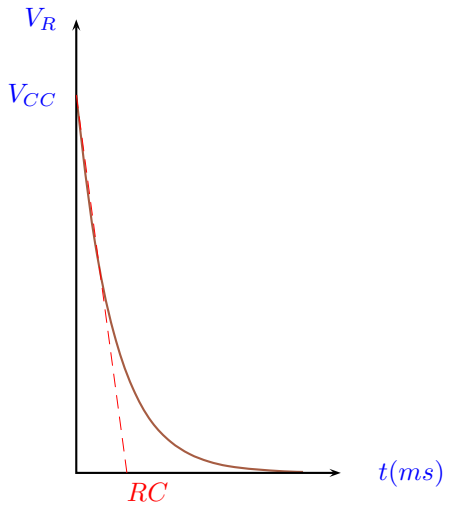

Figure 4.30: *La constante de temps pour le circuit RC.*

La tension sur le condensateur est donnée par l'équation 4.3.1.

$$V_C = V_{CC} e^{-\frac{t}{RC}} \tag{4.3.1}$$

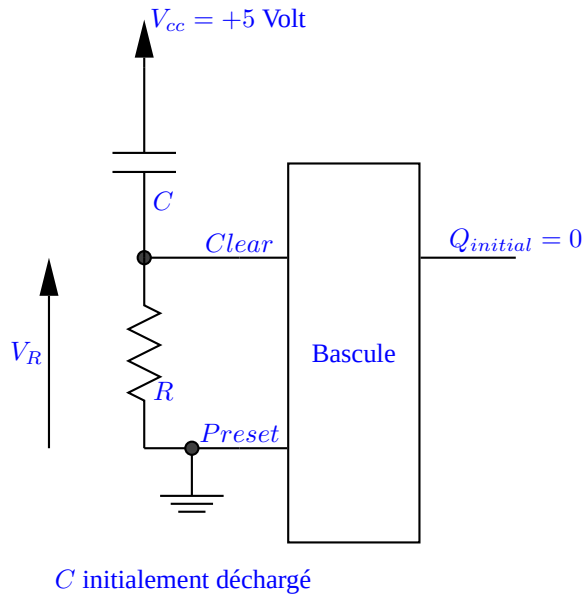

Figure 4.31: *La sortie mise à* 0.

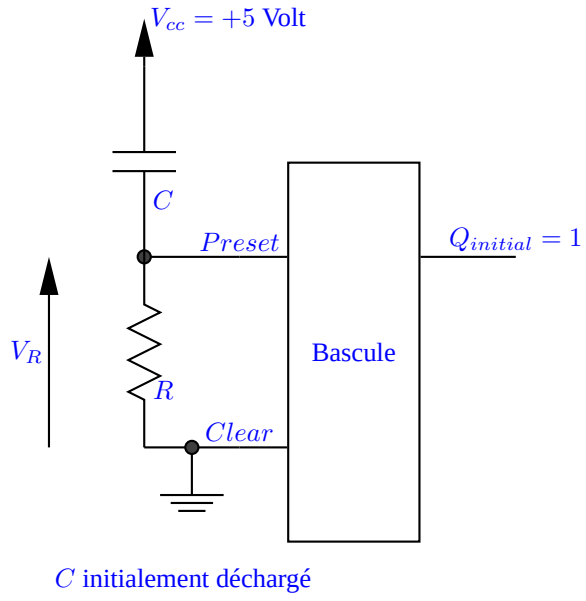

Figure 4.32: *La sortie mise à* 1.

Les Entrées Asynchrones Actives à l'État Bas

A l'aide d'un circuit *RC*, nous activons l'entrée asynchrone un court instant au démarrage par un niveau bas, comme l'illustre la Figure 4.33, puis nous ramenons le signal asynchrone d'initialisation

à 1. Appliqué à l'entrée $\overline{Clear}$, ceci initialise Q à 0, mais appliqué à l'entrée $\overline{Preset}$, ceci initialise Q à 1. A chaque fois l'entrée asynchrone non utilisée est placée à l'état inactif, soit l'état 0. Le maintien prolongé d'une entrée asynchrone à son niveau actif fixe Q constant i.e., la sortie est mise à 0 si $\overline{Clear}$ est actif comme l'illustre la Figure 4.34. La sortie est mise à 1 si $\overline{Preset}$ est actif, comme l'illustre la Figure 4.35.

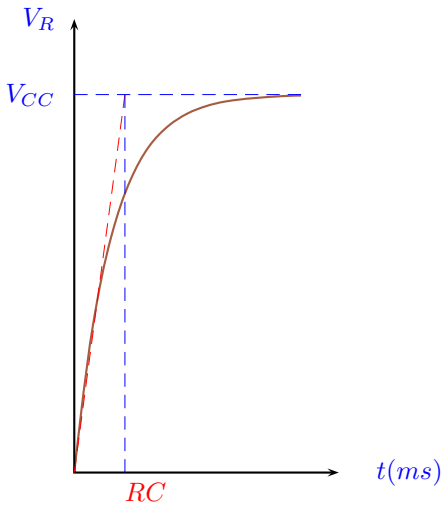

Figure 4.33: *La constante de temps pour le circuit RC.*

La tension sur le condensateur est donnée par l'équation 4.3.2.

$$V_C = V_{CC}(1 - e^{-\frac{t}{RC}}) \tag{4.3.2}$$

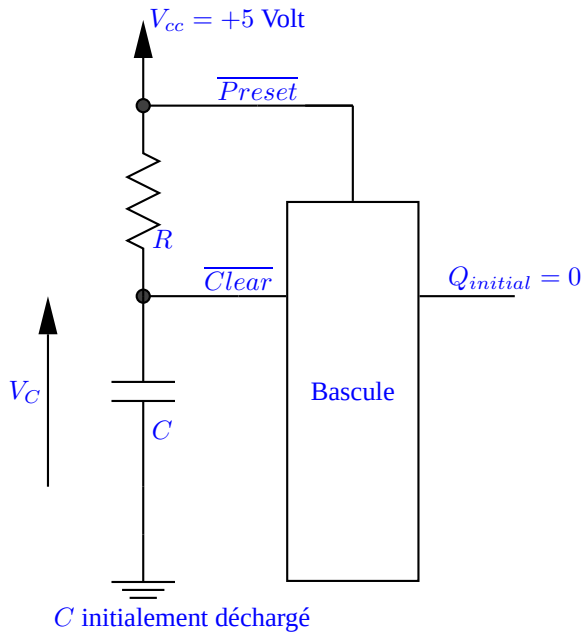

Figure 4.34: *La sortie mise à* 0.

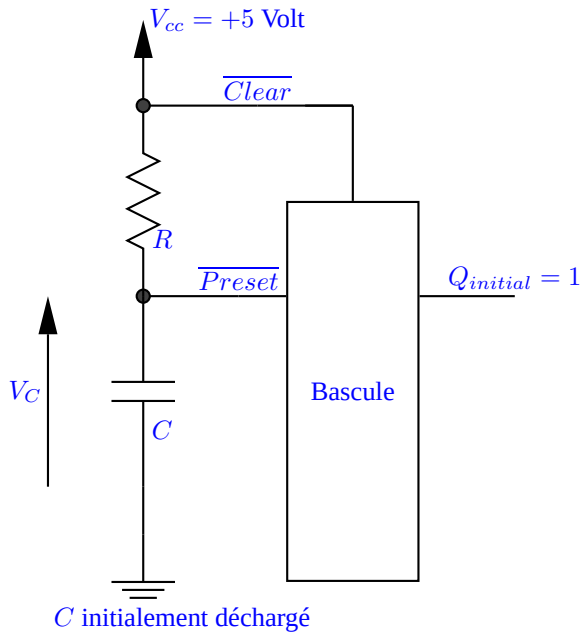

Figure 4.35: *La sortie mise à* 1.

La Bascule JK à Déclenchement sur Front

Il existe dans les circuits actuellement commercialisés deux façons de sélectionner un front qui sont les suivantes :

$\Rightarrow$ 1. Le signal d'horloge est dérivé, par exemple le composant $SN5470$.

$\Rightarrow$ 1. Le signal d'horloge n'est pas dérivé. La discrimination du front s'effectue comme pour la bascule D *edge triggered* à l'aide de mémoires internes.

Le circuit type $SN54109N$ ou $SN74109$, comme l'illustre la Figure 4.36, de la *Texas Instruments* qui est décrit à l'aide du schéma suivant constitue une bascule JK qui est synchronisée sur les fronts positifs. Nous pouvons étudier son fonctionnement en considérant cette bascule comme un système séquentiel asynchrone à trois variables d'entrée J, K et *Horloge*.

Les entrées asynchrones A mise à 0 et mise à 1 généralement actives à l'état bas (donc notées $\overline{A}$) sont telles que lorsque mise à 0 est activée, Q est placé à l'état 0 quelles que soient les entrées d'horloge et de données J et K. Ce sont des commandes d'effacement et d'initialisation (appelées aussi *Clear* et *Preset*) qui peuvent être activées pour fixer l'état initial de la sortie et qui doivent ensuite être inactives pour permettre le fonctionnement normal de la bascule. Un simple circuit RC connecté à l'entrée *Preset* pour fixer l'état initial $Q = 1$ peut être utilisé, comme montrent les Figures 4.37 et 4.38 pour le $\overline{Preset}$ et les Figures 4.39 et 4.40 pour le *Preset*.

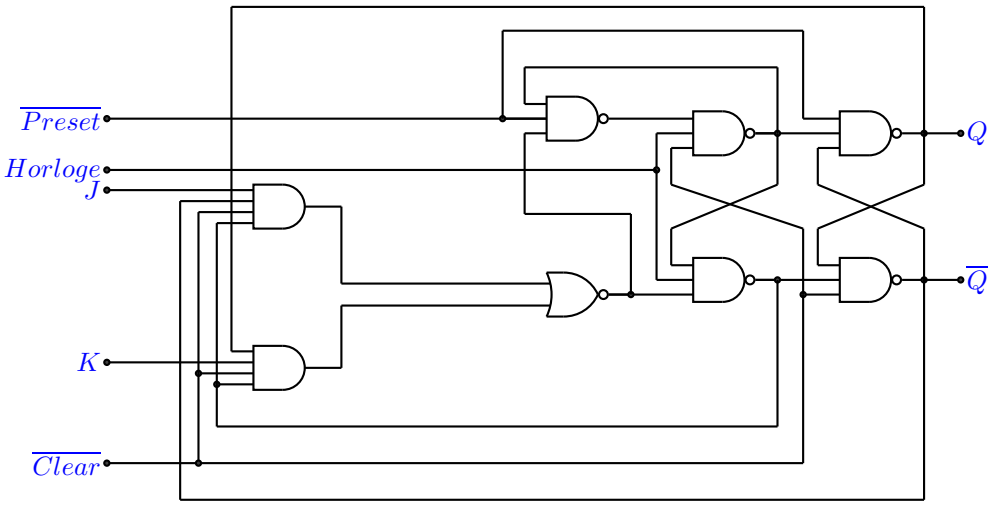

Figure 4.36: *Le circuit de la bascule JK.*

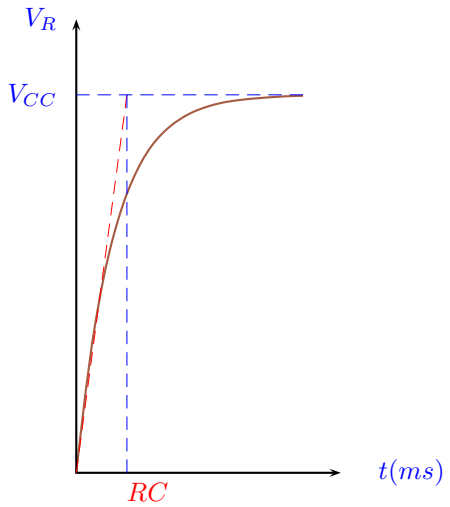

Figure 4.37: *La constante de temps pour le signal* $\overline{Preset}$.

La tension sur le condensateur est donnée par l'équation 4.3.3.

$$V_C = V_{CC}(1 - e^{-\frac{t}{RC}}) \tag{4.3.3}$$

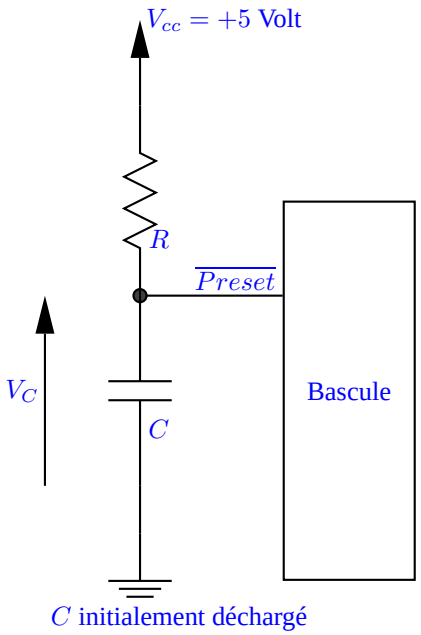

Figure 4.38: *Le circuit pour le signal* $\overline{Preset}$.

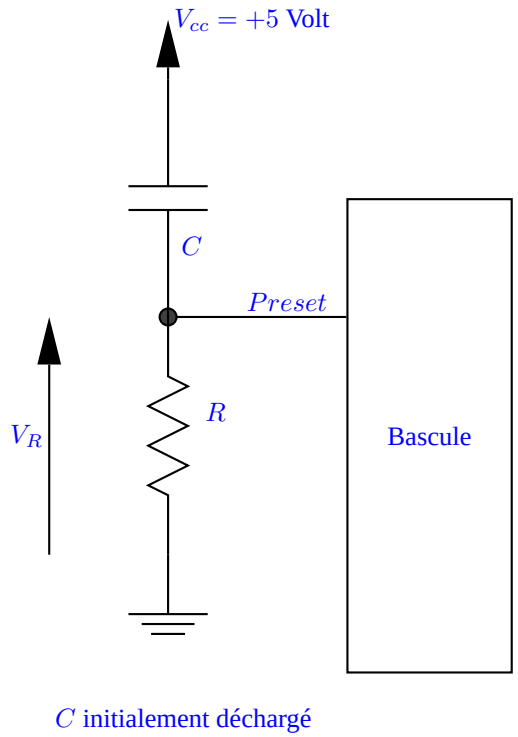

Figure 4.39: *Le circuit pour le signal* Preset.

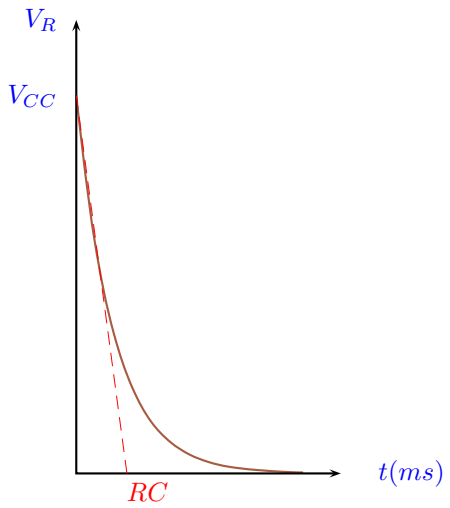

Figure 4.40: *La constante de temps pour le signal* Preset.

La tension sur le condensateur est donnée par l'équation 4.3.4.

$$V_C = V_{CC} e^{-\frac{t}{RC}} \tag{4.3.4}$$

Structure Maître-esclave

La structure maître-esclave est une structure à deux bascules synchrones. L'une, appelée le maître, est placée à l'entrée, l'autre, l'esclave, de type D est placée en sortie. Pour simplifier l'étude du fonctionnement de cet ensemble, la synchronisation de chaque bascule est symbolisée par un interrupteur qui se ferme pendant la phase active de l'horloge, comme l'illustre la Figure 4.41. Les interrupteurs I_M et I_E sont commandés par le niveau du signal d'horloge par rapport à un seuil déterminé.

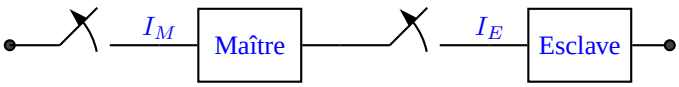

Figure 4.41: *Le schéma maître-esclave.*

Le signal d'horloge étant une impulsion, les niveaux correspondant aux seuils définissent 4 points ou 5 zones distinctes, comme l'illustre la Figure 4.42.

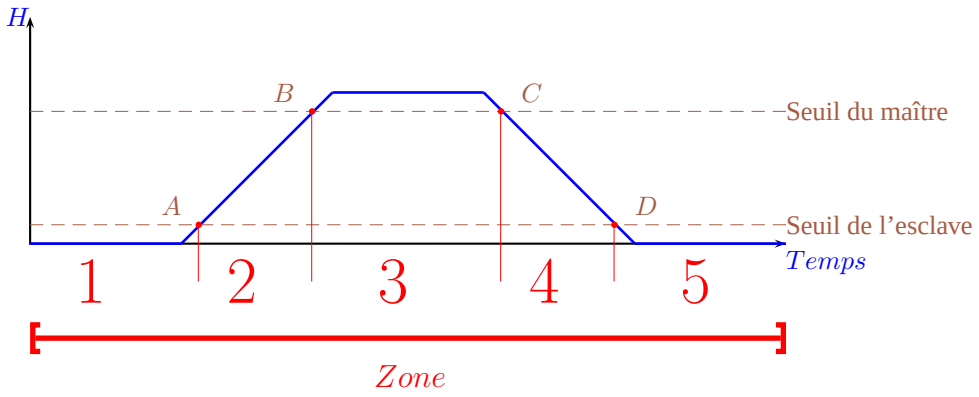

Figure 4.42: *Le signal d'horloge.*

Dans la zone temporelle 1 les interrupteurs du maître et de l'esclave sont respectivement ouverts et fermés. Le maître fonctionne alors en mémoire, l'esclave recopie la sortie du maître (Figure 4.43).

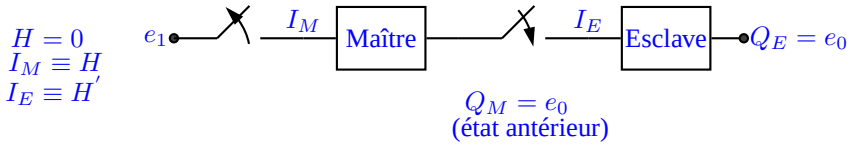

Figure 4.43: *L'analyse de la zone* 1.

Pour accéder dans la zone 2 le seuil de l'esclave est franchi et l'interrupteur correspondant change d'état. L'esclave est alors séparé du maître et fonctionne en mémoire (Figure 4.44).

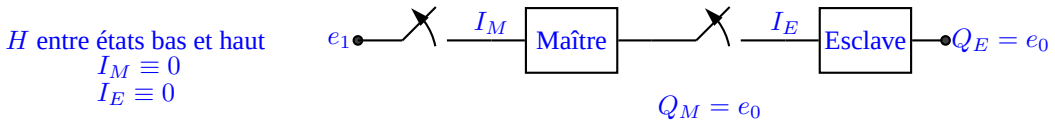

Figure 4.44: *L'analyse de la zone* 2.

Dans la zone 3 l'esclave mémorise toujours la même information et le signal de sortie n'a pas encore changé. Par contre, le maître prend en compte l'information d'entrée (Figure 4.45).

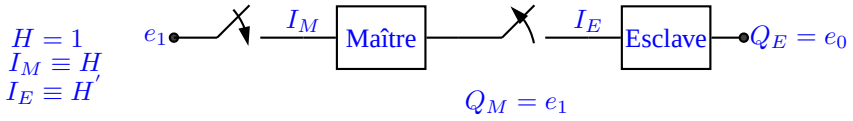

Figure 4.45: *L'analyse de la zone* 3.

Dans la zone 4 le maître est de nouveau isolé de l'entrée. Il a mémorisé la nouvelle valeur de la sortie (e_1). L'esclave reste encore isolé du maître par conséquent la valeur de la première sortie n'est pas encore modifiée (Figure 4.46).

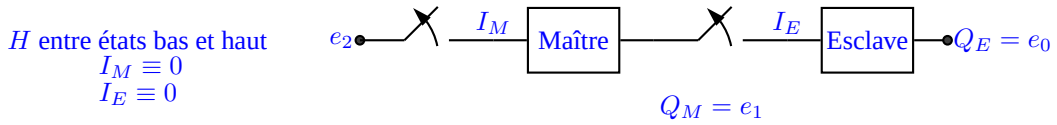

Figure 4.46: *L'analyse de la zone* 4.

C'est seulement dans la zone 5 que le maître communique la nouvelle valeur à l'esclave qui la transmet en sortie (Figure 4.47).

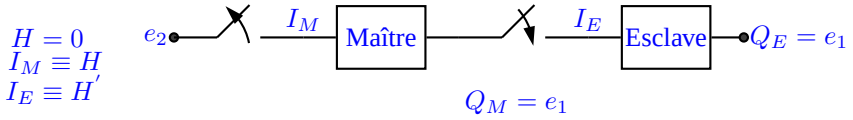

Figure 4.47: *L'analyse de la zone* 5.

Nous pouvons remarquer qu'il n'existe pas de configuration où les deux interrupteurs sont simultanément fermés, ce qui permet d'effectuer un rebouclage sortie vers l'entrée sans craindre une instabilité.

D'autre part le décalage temporel entre les commandes des interrupteurs I_M et I_E est inévitable puisqu'il est impossible d'obtenir le signal de l'horloge capable de passer instantanément du niveau 0 au niveau 1 (ou inversement). En l'autres termes, nous pouvons dire que la qualité des fronts $\frac{dv}{dy}$ du signal l'horloge $v(t)$ n'influence pas le principe décrit. Néanmoins, il faut que l'impulsion de l'horloge ait une durée suffisante compte tenu de la vitesse d'évolution de la bascule.

Enfin, on remarque que si le signal d'horloge est supérieur au seuil de l'esclave, alors l'interrupteur I_E est ouvert. Et si le signal d'horloge est supérieur au seuil du maître, I_M est fermé. Les deux interrupteurs peuvent donc être commandés par des signaux complémentaires. Les phases 2 et 4 où les interrupteurs sont ouverts simultanément sont dûes à la transition H de 1 vers 0 et $\overline{H}$ de 0 vers 1. Nous pouvons faire une analogie entre le mécanisme qui nous avons utilisé et les circuits, comme l'illustre la Figure 4.48.

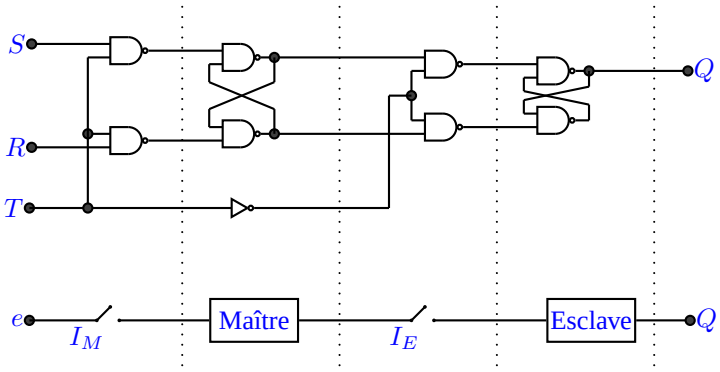

Figure 4.48: *Le schéma d'une bascule RST maître-esclave.*

La bascule JK maître-esclave, comme l'illustre la Figure 4.49, se déduit de la bascule précédente en réalisant le rebouclage $S = J\overline{Q}$ et $R = KQ$.

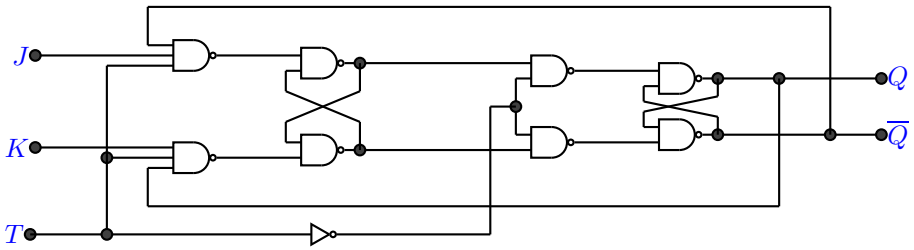

Figure 4.49: *Le schéma d'une bascule JK maître-esclave.*

Suivant le mode de synchronisation du maître lors de la phase 3, il faut observer qu'il existe deux types de bascules maître. En effet, l'acquisition de la nouvelle valeur peut être faite sur le front montant du signal de synchronisation ou sur son niveau. Pour distinguer ces deux types de bascules, on appelle bascule maître-esclave à verrouillage la structure dont le maître est synchronisé sur le front montant de l'horloge. Enfin, il faut remarquer que la stabilité apportée à une bascule par une structure maître-esclave se fait au prix d'un coût en temps de propagation de la bascule.

4.4 *Tableau Général*

Chaque type de bascule est donnée avec table de fonctionnement et parfois un exemple de circuit intégré - *CI* de la société *Texas Instruments*.

4.4.1 *Bascule RS*

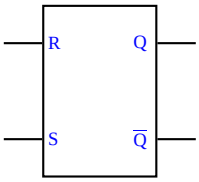

Figure 4.50: *Le symbole de la bascule RS.*

S	R	Q_n	*Fonction*
0	0	Q_{n-1}	*Mémorisation*
0	1	0	*Recopie de* S
1	0	1	*Recopie de* S
1	1	$\boxtimes$	*Interdit car* $Q_n = Q_n^{'}$

Table 4.13: *Table fonctionnement de la bascule RS.*

La bascule RS est sujette à des aléas de fonctionnement (sorties imprévisibles) lorsque les 2 entrées S et R changent d'état simultanément.

4.4.2 *Bascule D ou Gated D-Latch*

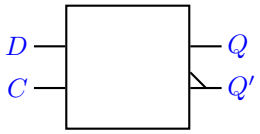

Figure 4.51: *Le symbole de la bascule D ou Gated D-Latch.*

C	D	Q_n	*Fonction*
0	X	Q_{n-1}	*Mémorisation*
1	0	0	*Recopie*
1	1	1	*Recopie*

Table 4.14: *Table fonctionnement de la bascule D ou Gated D-Latch.*

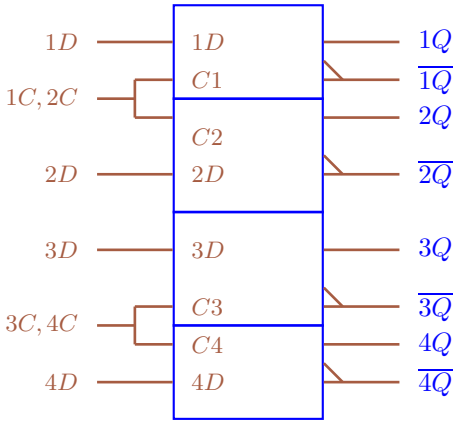

Figure 4.52: *Le composant SN74XX75.*

4.4.3 *Bascule D Edge Triggred*

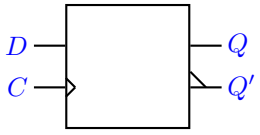

Figure 4.53: *Le symbole de la bascule D edge triggered.*

C	D	Q_n	*Fonction*
X	X	Q_{n-1}	*Mémorisation*
$\uparrow$	0	0	*Recopie*
$\uparrow$	1	1	*Recopie*

Table 4.15: *Table de fonctionnement de la bascule D edge triggered.*

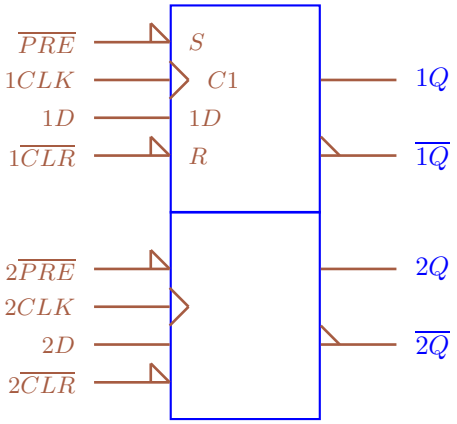

Figure 4.54: *Le composant SN74XX74.*

Les entrées asynchrones *Preset* et *Clear*, actives à l'état bas, mise à 1 et à 0 servent à l'initialisation ou l'effacement.

4.4.4 *Bascule JK Edge Triggred*

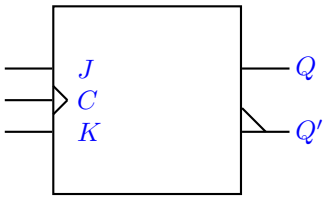

Figure 4.55: *Le symbole de la bascule JK edge triggered.*

C	J	K	Q_n	*Fonction*
X	X	X	Q_{n-1}	*Mémorisation*
$\uparrow$	0	0	Q_{n-1}	*Mémorisation*
$\uparrow$	0	1	0	*Recopie de J*
$\uparrow$	1	0	1	*Recopie de J*
$\uparrow$	1	1	Q'_{n-1}	*Complémentaire*

Table 4.16: *Table de fonctionnement de la bascule JK edge triggered.*

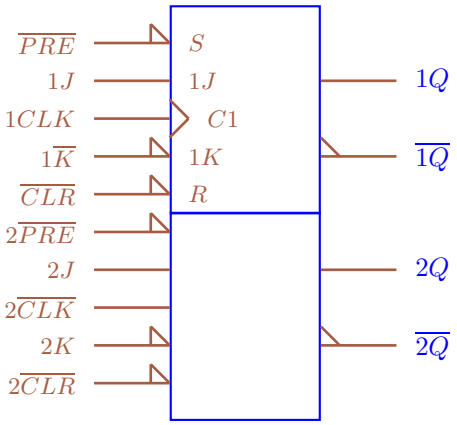

Figure 4.56: *Le composant SN74XX109.*

Les entrées asynchrones *Preset* et *Clear*, actives à l'état bas, de mise à 1 et à 0 servent à l'initialisation ou l'effacement.

4.4.5 *Bascule JK Maître-esclave*

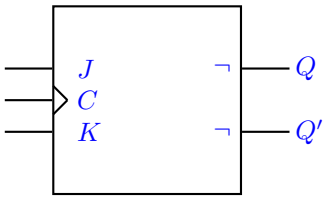

Figure 4.57: *Le symbole de la bascule JK maître-esclave.*

C	J	K	Q_n	*Fonction*
X	X	X	Q_{n-1}	*Mémorisation*
$\downarrow$	0	0	Q_{n-1}	*Mémorisation*
$\downarrow$	0	1	0	*Recopie de J*
$\downarrow$	1	0	1	*Recopie de J*
$\downarrow$	1	1	Q'_{n-1}	*Complémentaire*

Table 4.17: *Table de fonctionnement de la bascule JK maître-esclave.*

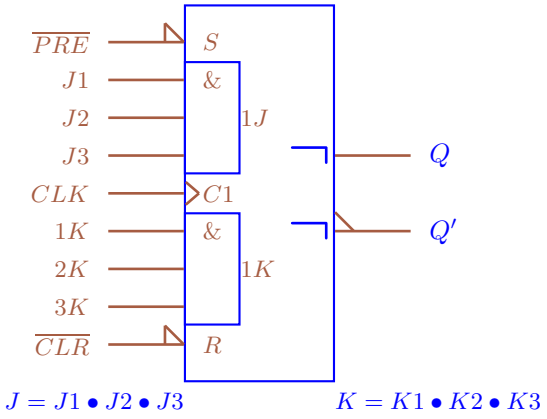

Figure 4.58: *Le composant SN74XX72.*

Le entrées J et K sont calculés à l'aide d'une porte *ET* à 3 entrées. Le symbole $\neg$ indique que la sortie n'évolue qu'après le retour à l'état initial de l'horloge C.

4.4.6 *Bascule JK avec Verrouillage de la Donnée*

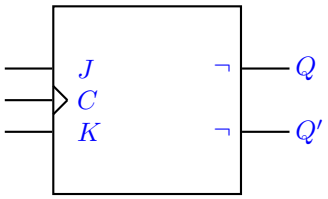

Figure 4.59: *Le symbole de la bascule JK avec verrouillage de la donnée.*

C	J	K	Q_n	*Fonction*
X	X	X	Q_{n-1}	*Mémorisation*
$\uparrow$ _ $\sqcap$ _ $\downarrow$	0	0	Q_{n-1}	*Mémorisation*
$\uparrow$ _ $\sqcap$ _ $\downarrow$	0	1	0	*Recopie de J*
$\uparrow$ _ $\sqcap$ _ $\downarrow$	1	0	1	*Recopie de J*
$\uparrow$ _ $\sqcap$ _ $\downarrow$	1	1	Q'_{n-1}	*Complémentaire*

Table 4.18: *Table de fonctionnement de la bascule JK maître-esclave* $SN74X110$.

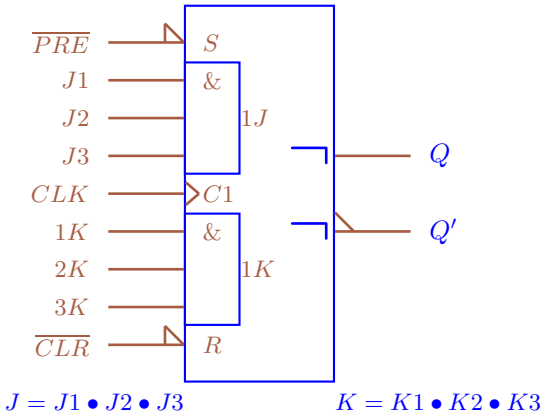

Figure 4.60: *Le composant SN74X110.*

Par rapport à la bascule JK maître-esclave, l'entrée de contrôle C est munie du triangle, symbole d'une activité sur un front positif, ce qui signifie que les entrées J et K sont échantillonnés sur le front montant de C. Le résultat n'est transmis en sortie qu'après le retour à l'état initial de l'information d'horloge C.

4.4.7 *Bascule RS Maître-esclave*

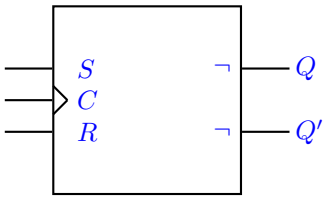

Figure 4.61: *Le symbole de la bascule RS maître-esclave.*

C	S	R	Q_n	*Fonction*
X	X	X	Q_{n-1}	*Mémorisation*
$\downarrow$	0	0	Q_{n-1}	*Mémorisation*
$\downarrow$	0	1	0	*Recopie de S*
$\downarrow$	1	0	1	*Recopie de S*
$\downarrow$	1	1	$\boxtimes$	*Interdit*

Table 4.19: *Le fonctionnement de la bascule RS maître-esclave.*

4.5 *Conclusion*

Nous avons observé qu'il y a différents types de bascules qui pourrons être utilisé en différents types d'applications. Nous devons bien connaître les tables de vérités de toutes les bascules. Les autres caractéristiques (synchrone, asynchrones, etc) sont fonction de l'application. Dans les prochains chapitres, nous étudierons les applications des circuits séquentiels pour faire les réalisations des registres, des compteurs et des séquenceurs.

Chapitre 5

Les Registres

5.1 *Introduction*

Un registre est d'abord un ensemble de cases ou cellules mémoire capables de stocker une information égale à un mot. La position des cases mémoire entre elles est responsable de l'ordre des chiffres, c'est à dire de la structure de l'information. Dans le système binaire, une case mémoire est définie à l'aide d'une bascule. Un registre est donc un ensemble ordonné de bascules. De plus, l'interconnexion entre les bascules permet certaines manipulations de l'information stockée.

5.2 *Fonctionnement*

Un registre sert à mémoriser un mot ou un nombre binaire. Le schéma d'un tel système comporte autant de bascule type D que d'éléments binaires à mémoriser. Toutes les bascules sont commandées par le même signal d'horloge, comme l'illustre la Figure 5.1.

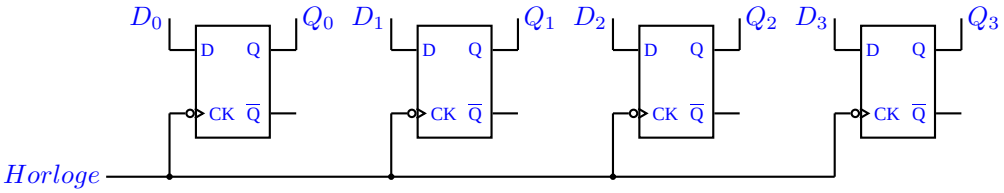

Figure 5.1: *Le registre de quatre bits.*

Moyennant une interconnexion entre les cellules, le registre précédent devient capable d'opérer une translation des chiffres ou bits du nombre ou mot initialement stocké. Le déplacement s'effectue soit vers la droite soit vers la gauche. Le registre est alors appelé registre à décalage. Les nombreuses applications résultent de cette possibilité de décalage comme :

$\Rightarrow$ 1. La conversion série-parallèle d'une information numérique ;

$\Rightarrow$ 2. Les opérations de multiplication et division par 2 ;

$\Rightarrow$ 3. La ligne à retard numérique.

Plus généralement un registre peut se représenter par le schéma de la Figure 5.2.

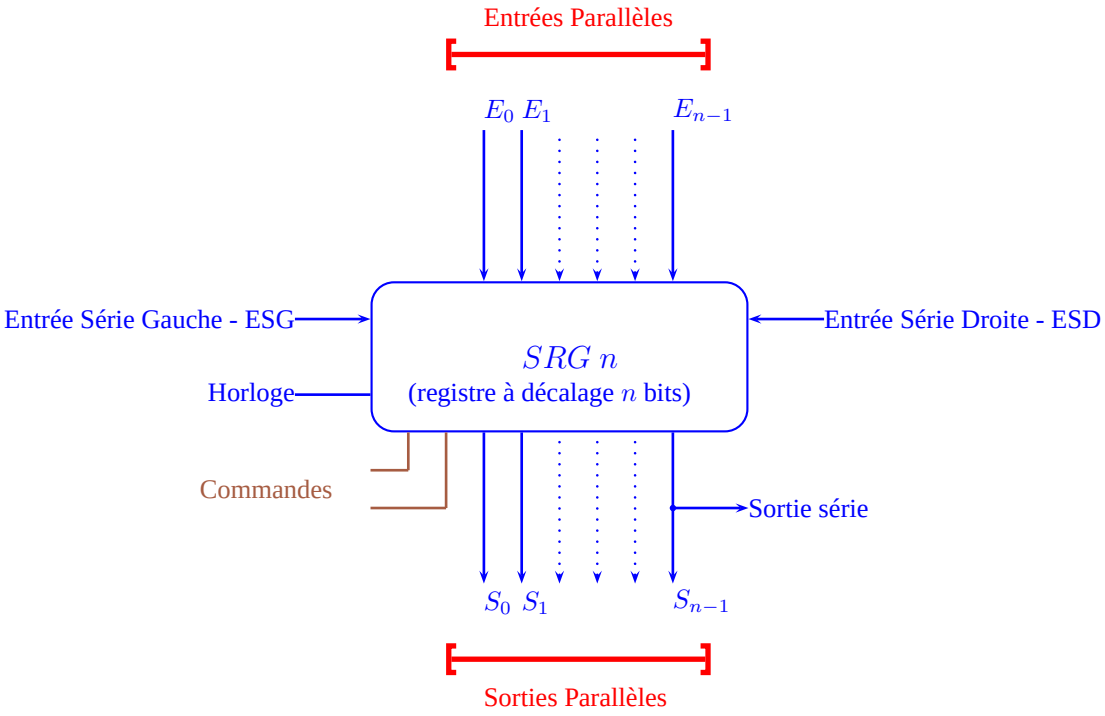

Figure 5.2: *La représentation générale d'un registre.*

Les registres disposant de toutes ces entrées, sorties et commandes sont appelés registres universels à n cellules. Tous les registres actuellement commercialisés n'ont pas toutes les possibilités de commande du registre universel essentiellement à cause de la limitation du nombre de broches disponibles par boîtier. Les sorties $S_0 ... S_{n-1}$ sont les sorties des bascules constituant les cellules du registre.

Les signaux de commande du registre permettent de garder une information en mémoire. Chaque bascule conserve sa valeur malgré les impulsions d'horloge, comme l'illustre la Figure 5.3.

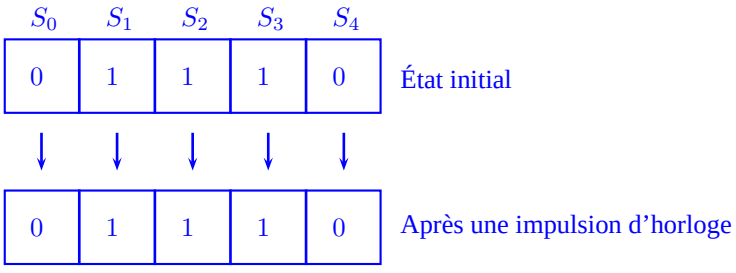

Figure 5.3: *La conservation de l'information.*

Les signaux de commande du registre permettent de décaler une information de la gauche vers

la droite. Le contenu de la bascule de rang i est transmis à celle de rang $i+1$ à chaque impulsion d'horloge, comme l'illustre la Figure 5.4.

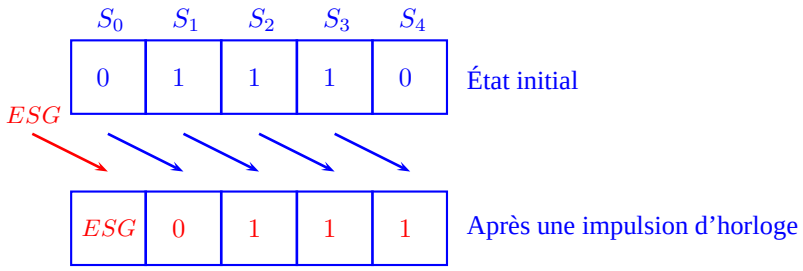

Figure 5.4: *Le décalage à droite.*

La sortie S_0 de la première bascule prend alors la valeur de l'entrée du registre appelé *Entrée Série Gauche - ESG*. Les signaux de commande du registre permettent de décaler une information de la droite vers la gauche. Le fonctionnement est semblable à celui décrit ci-dessus en inversant le sens d'évolution des éléments binaires de chaque sortie. La bascule de rang i prend la valeur de la sortie $j+1$ à chaque impulsion d'horloge, comme l'illustre la Figure 5.5.

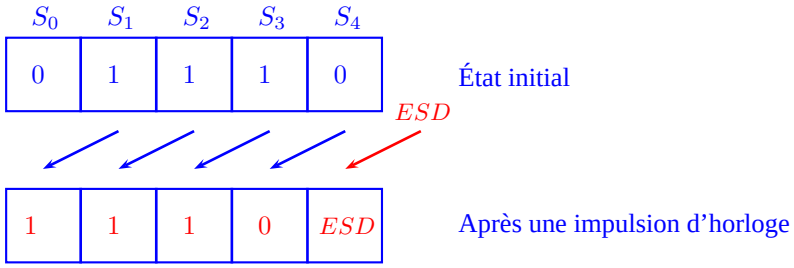

Figure 5.5: *Le décalage à gauche.*

Cette fois c'est la valeur de l'*Entrée Série Droite - ESD* qui est inscrite dans la dernière bascule S_4.

5.3 *Constitution d'un Registre*

Pour assurer correctement la fonction de décalage, un registre doit comporter des cellules constituées de bascules de type maître-esclave ou à déclenchement par front i.e., sans quoi le décalage n'est pas contrôlé. Le décalage n'est pas la seule fonction que doit pouvoir accomplir un registre.

5.3.1 *La Fonction Décalage à Droite*

La bascule D de rang i doit recopier la sortie de la bascule de rang $i-1$. Son entrée D doit donc être connectée à la sortie $i-1$. Le schéma de l'interconnexion entre les bascules est donné par la Figure 5.6.

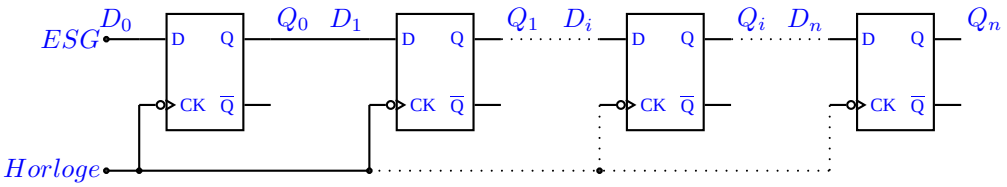

Figure 5.6: *La fonction décalage à droite.*

5.3.2 *La Fonction Décalage à Gauche*

Si nous supposons que la position de chaque bascule est fixe, c'est le câblage qui doit réaliser le décalage vers la gauche. Par conséquent, l'entrée D de la bascule de rang i est reliée à la sortie de rang $i+1$, comme l'illustre la Figure 5.7.

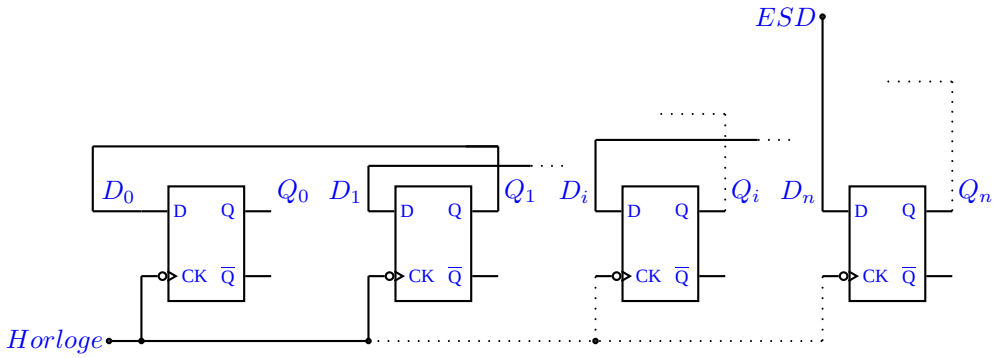

Figure 5.7: *La fonction décalage à gauche.*

5.3.3 *La Fonction Mémoire*

Suivant la nature de la bascule utilisée dans chaque cellule, la fonction mémoire peut se réaliser de différentes façons. La Table 5.1 donne la combinaison des entrées des bascules qui réalise la fonction mémoire.

Type	*Entrée*
RS	$R = S = 0$
JK	$J = K = 0$ *ou* $J = \overline{K} = Q$
D	$D = Q$

Table 5.1: *Fonction mémoire des bascules.*

Une autre solution consiste à interdire l'action de l'horloge en intercalant une porte *ET* en série , comme l'illustre la Figure 5.8. Cette dernière solution est à proscrire car elle a créé un décalage entre les différents signaux d'horloge d'un même système à cause du temps de propagation à travers la porte *ET* (*phénomène de Skew*). Le fonctionnement du registre n'est alors plus synchrone avec les autres circuits du système.

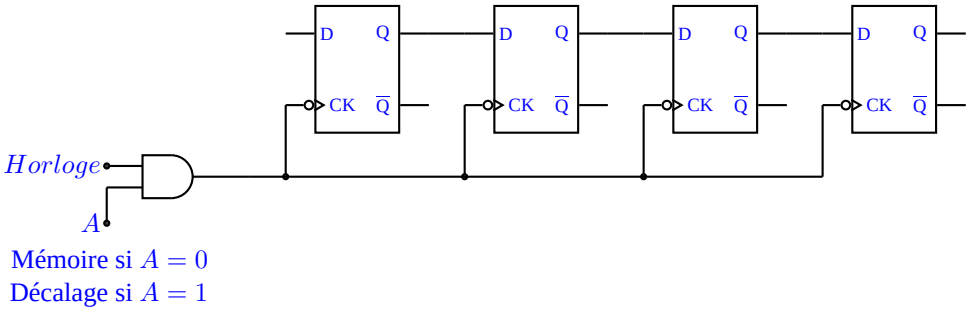

Figure 5.8: *Le décalage à partir de différents signaux d'horloge.*

5.3.4 *Écriture Asynchrone*

Il faut utiliser les entrées asynchrones i.e., les entrées prioritaires *Preset* - *PRE* de mise à 1 et *Clear* - *CLR* de mise à 0 de chaque bascule pour forcer l'information qui doit être écrite. Le forçage ne doit se faire qu'au moment de l'écriture, ce qui signifie qu'il est nécessaire d'ajouter des circuits à chaque bascule de façon à synchroniser les entrées de forçage par un ordre particulier généralement appelé ordre d'écriture ou de chargement (*LOAD*). La commande d'écriture est dans ce cas généralement asynchrone. Le principe de la synchronisation consiste à transformer une bascule asynchrone en une bascule *D latch*. Le schéma correspondant est donné par les Figures 5.9 et 5.10.

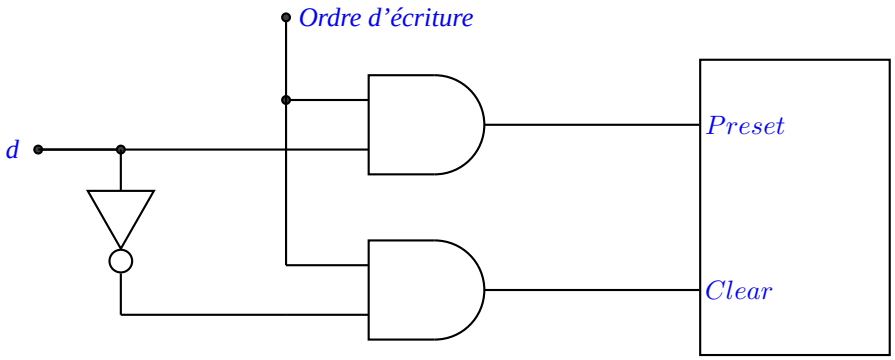

Figure 5.9: *La bascule sensible sur les niveaux* 1 *des entrées asynchrones.*

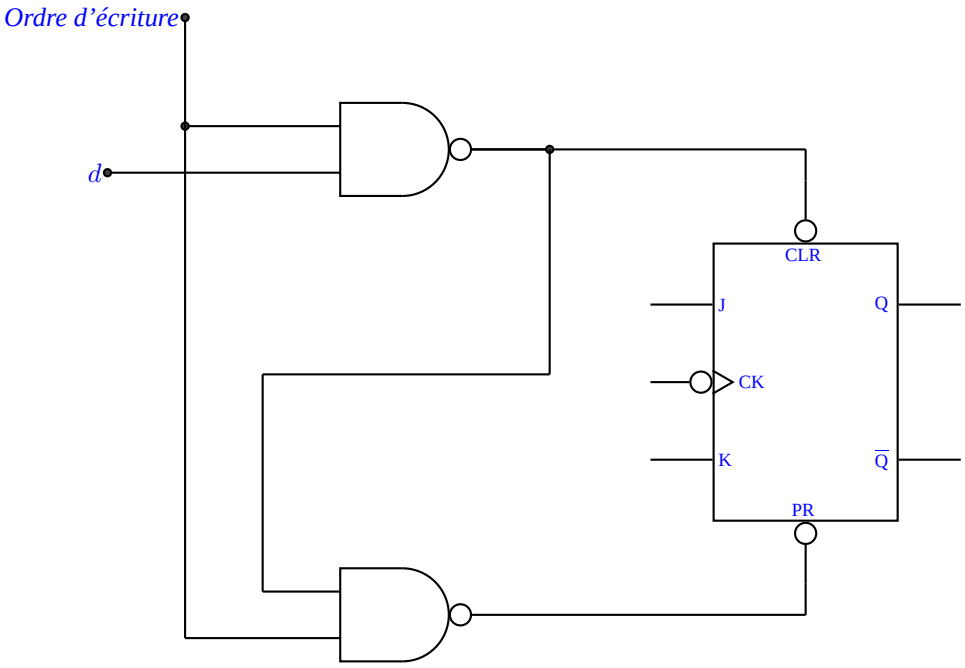

Figure 5.10: *La bascule sensible sur les niveaux* 0 *des entrées asynchrones.*

Nous remarquons que l'utilisation d'un opérateur *NAND* permet à la fois la synchronisation avec l'ordre d'écriture et obtenir le complémentaire de l'information d'entrée d.

5.3.5 *Écriture Synchrone*

La sortie d'une bascule D recopie son entrée au moment de la phase active de l'horloge. Pour provoquer l'écriture d'une donnée en synchronisme avec l'horloge il faut placer cette donnée sur l'entrée D de la bascule comme l'illustre la Figure 5.11.

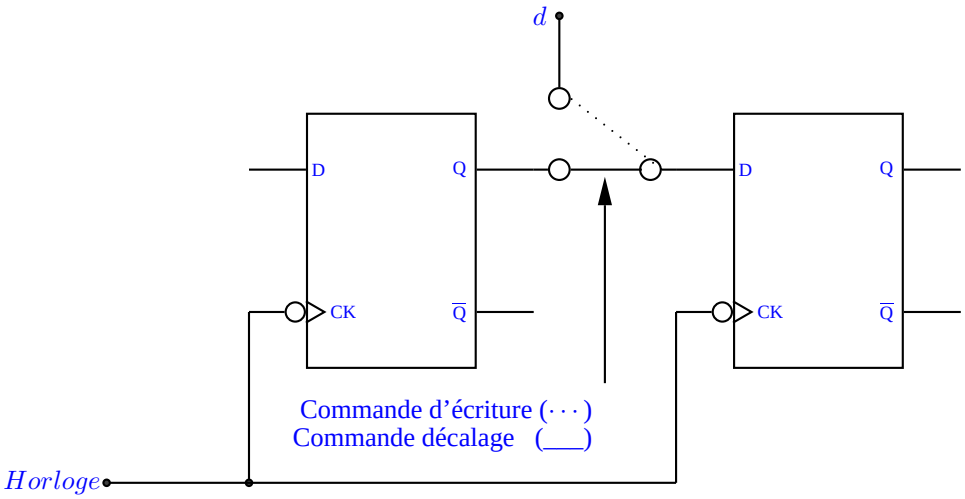

Figure 5.11: *L'écriture synchrone.*

5.3.6 *Initialisation de Registres*

Cette initialisation consiste à imposer pour toutes les bascules du registre la même valeur, en général 0, à l'aide d'une commande asynchrone. Le schéma d'un registre disposant d'une remise à zéro *RAZ* générale est donné par la Figure 5.12.

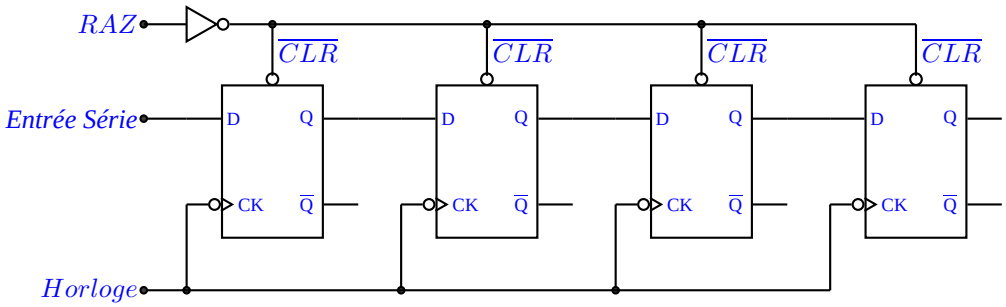

Figure 5.12: *L'initialisation des registres.*

5.3.7 *Schéma d'un Registre Universel Bidirectionnel*

Pour permettre la réalisation de l'une des fonctions possibles du registre, l'interconnexion entre les bascules doit être modifiée. Le choix de cette interconnexion est facilement réalisé en utilisant un circuit multiplexeur devant chaque entrée D des bascules. Le schéma du circuit intégré $SN74AS195$ est donné par la Figure 5.13. Nous distinguons facilement les multiplexeurs réalisant la sélection des interconnexions et le blocage de l'horloge pour réaliser la fonction mémoire. Ce circuit dispose également d'une remise à zéro ($\overline{Clear}$) globale pour une éventuelle initialisation.

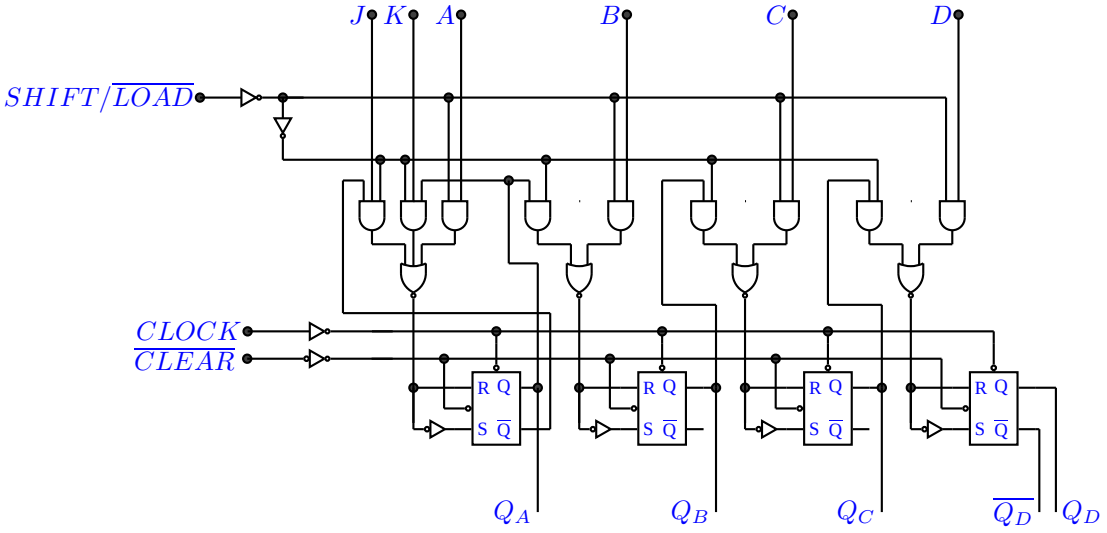

Figure 5.13: *Le schéma d'un registre universel bidirectionnel ($SN74AS195$).*

5.4 *Conclusion*

Nous venons d'étudier l'application de circuit séquentiels en registres. Nous pouvons observer que tous les systèmes numériques utilisent des mémoires pour stocker les informations de façon temporaire à partir de registres. Une autre type d'application, les compteurs seront objet d'études du prochain chapitre.

Chapitre 6

Les Compteurs

Un compteur est un ensemble de bascules dont les sorties forment un mot et qui compte dans une séquence donnée à chaque coup d'horloge. Il est représenté par un automate d'états fini encore appelé machines d'états: - c'est l'horloge qui fait passer d'un état au suivant, contrairement au *séquenceur* dont les changements d'état sont soumis aussi à des actions et des transitions.

6.1 *Introduction*

Les compteurs sont des éléments essentiels de logique séquentielle ; ils permettent en effet d'établir une relation d'ordre de succession d'événements. Leur emploi ne se limite pas, loin de là, aux systèmes arithmétiques. Ils sont utiles partout où il est souhaitable de définir facilement une suite d'états. L'élément de base des compteurs est, comme pour les registres, une bascule. L'état du compteur est défini par le nombre binaire formé avec l'ensemble des sorties des bascules. La synthèse d'un compteur consiste à définir les niveaux logiques des commandes des bascules pour assurer le passage successif d'un état à l'autre suivant l'ordre du cycle prévu ou échelle de comptage.

Ils sont classés en deux catégories suivant leur mode de fonctionnement : - les compteurs asynchrones (*compteurs série*) et les compteurs synchrones (*compteurs parallèles*). La caractéristique principale des compteurs asynchrones est la propagation en cascade de l'ordre de changement d'état des bascules. L'horloge synchronise la première bascule, dont la sortie va synchroniser la bascule suivante. L'autre possibilité sont les compteurs synchrones ou compteurs parallèles. Le signal d'horloge synchronise toutes les bascules simultanément.

6.2 *Les Compteurs Asynchrones*

Dans un compteur asynchrone, l'horloge déclenche la bascule B_1, dont la sortie sert de signal d'horloge à la bascule B_2. Plus généralement, le signal d'horloge d'une bascule B_i est issu d'une combinaison logique des sorties des bascules B_j avec j inférieur à i. Comme les sorties changent en cascade après le signal d'horloge, il ne va pas exister un synchronisme dans l'évolution des sorties des bascules. Ceci justifie le nom de ces compteurs.

6.2.1 *Les Compteurs Binaires*

C'est le plus simple des compteurs asynchrones i.e., les sorties des bascules qui le composent évoluent, au rythme de l'horloge, de manière à représenter la succession croissante des nombres

exprimés en base 2, comme l'illustre la Figure 6.1.

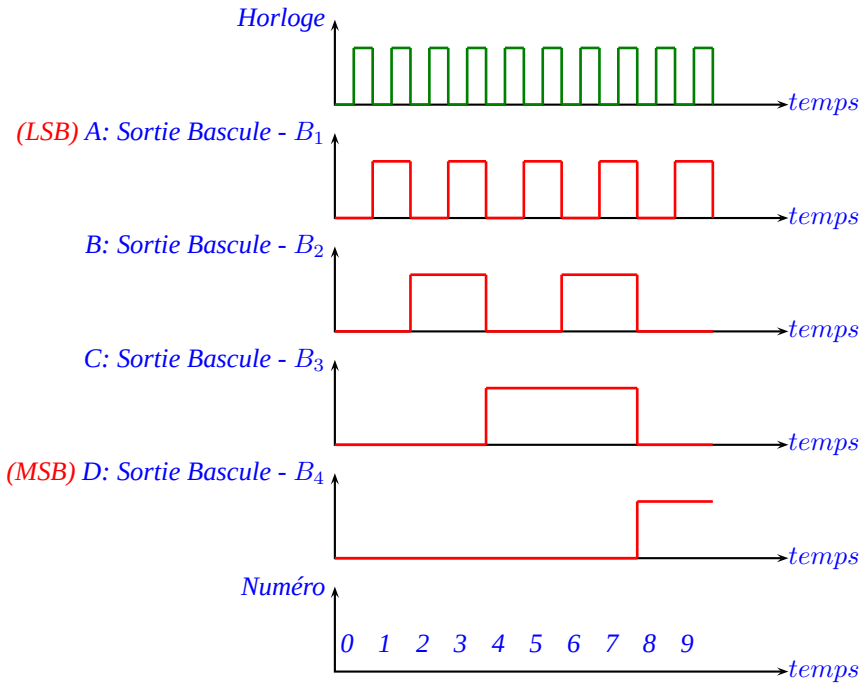

Figure 6.1: *Le registre de quatre bits.*

Si nous les affectons des poids $1, 2, 4, 8$ respectivement aux sorties A, B, C, D des bascules B_l, B_2, B_3, B_4, le nombre binaire ainsi représenté suit l'ordre croissant de 0 à 15. Un compteur binaire m bits compte donc de 0 à 2^{m-1}.

Nous remarquons facilement que la première bascule $B_i(A)$ change d'état à chaque impulsion d'horloge, que la sortie B change d'état à chaque fois que la sortie A présente une transition descendante ($1 \rightarrow 0$) et ainsi de suite. En d'autres termes, la période de la sortie A est égale au double de la période d'horloge, la période de B est égale à 2 fois celle de la sortie A. Ceci est facilement réalisé en utilisant des bascules câblées en type T, par des bascules JK avec $J = K = 1$. Le schéma d'un compteur binaire est illustré par la Figure 6.2.

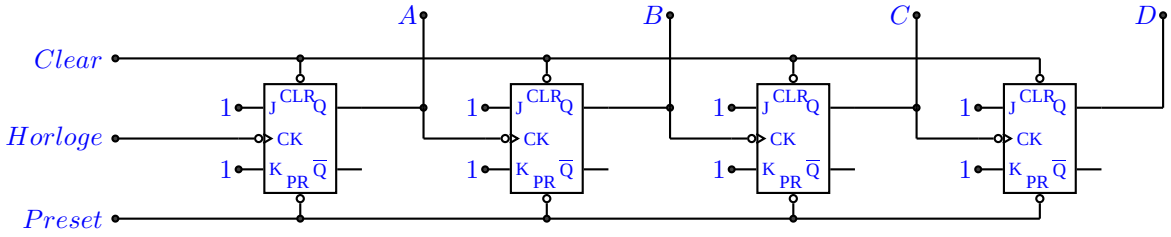

Figure 6.2: *Le compteur binaire.*

Afin d'illustrer le désynchronisme qui apparaît entre les sorties, examinons la transition entre 7

et 8, comme l'illustre la Figure 6.3. Ce cas est particulièrement perturbé puisque toutes les sorties changent.

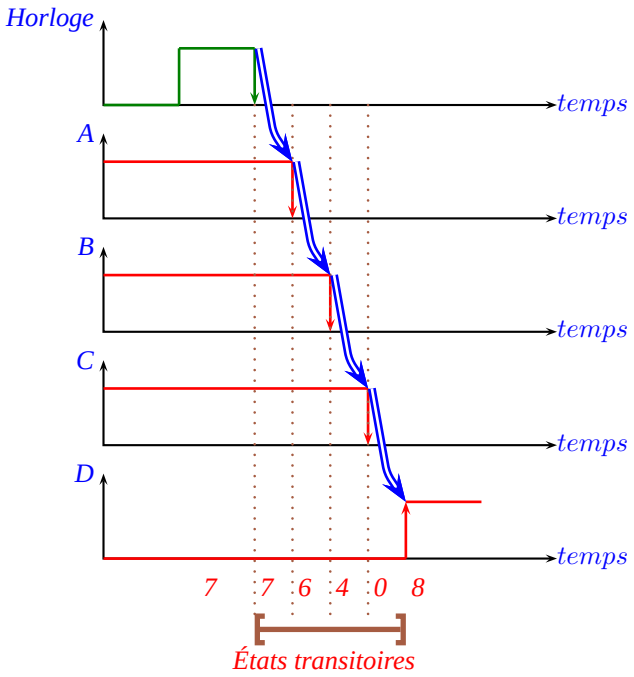

Figure 6.3: *L'analyse de la transition 7 et 8.*

La structure en cascade implique l'accumulation des retards entre la transition descendante de l'horloge et la stabilisation des sorties des bascules. Comme les sorties changent les unes après les autres, il apparaît des états transitoires indésirables. Sur le chronogramme de la Figure 6.3, ces différents états sont repérés par leur équivalent décimal. Pour un compteur asynchrone, si n bascules changent d'état après une impulsion d'horloge, il existe $(n-1)$ états transitoires.

6.2.2 *Les Compteurs Modulo N*

Nous appelons compteur modulo N, un compteur décrivant la succession des nombres binaires compris entre 0 et $N-1$, c'est à dire la suite des chiffres d'une base N traduite en binaire. Par exemple, pour $N = 10$, la succession des états du compteur est donnée par le cycle de la Figure 6.4.

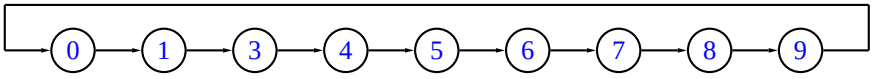

Figure 6.4: *Le cycle d'un compteur de modulo* 10.

Pour réaliser un tel compteur il existe deux types de solutions : - un rebouclage asynchrone, ou un conditionnement des entrées des bascules.

Rebouclage Asynchrone

Le compteur modulo N est dans ce cas considéré comme un compteur binaire m bits comptant de 0 à 2^{m-1} dont le cycle est interrompu à $N = 2^{m-1}$. En effet entre 0 et 9 par exemple, la succession des états est identique pour les deux types de compteurs: - compteur binaire 4 bits et compteur modulo 10. Si, dans l'état 9, une impulsion d'horloge est appliquée au compteur. celui-ci inscrira la combinaison 10 en binaire. Cet état est indésirable. Par conséquent, dès qu'il est détecté par décodage, une remise à zéro (*Clear - CLR*) est automatiquement imposée au compteur pour satisfaire le cycle voulu. Ce qui donne le schéma de la Figure 6.5.

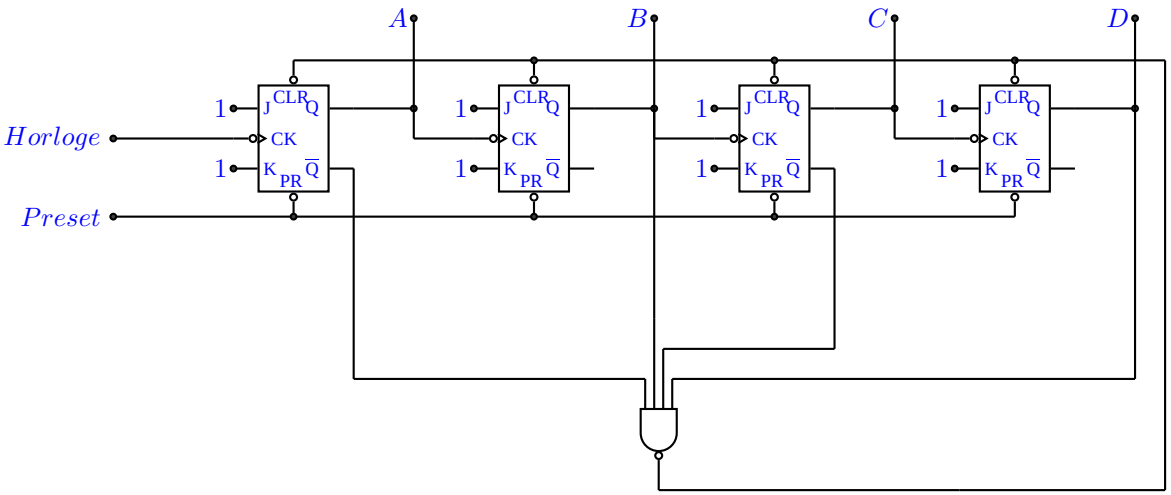

Figure 6.5: *Le compteur de la remise à zéro.*

Remarque 6.2.1

$\Rightarrow 1$. *Le décodage de 10 se limite à vérifier que B et D sont à 1, les combinaisons 11 à 15 ne devant pas apparaître en fonctionnement normal.*

$\Rightarrow 2$. *Il ne faut pas limiter la remise à zéro aux seules bascules qui sont à 1. En effet si seules les bascules B et D sont remises à 0, le changement d'état de B provoque le changement d'état de C.*

$\Rightarrow 3$. *Enfin cette solution ne donne satisfaction que si toutes les bascules ont des temps de réaction semblables.*

Conditionnement des Entrées

Dans un compteur asynchrone il existe deux possibilités de commander une bascule : - par action sur l'horloge ou par action sur les entrées des bascules. La succession des états dans l'exemple précédent ($N = 10$) est donné par la Table 6.1 dans laquelle le passage d'une ligne à la suivante est conditionné par une impulsion d'horloge.

Numéro	*D*	*C*	*B*	*A*
0	*0*	*0*	*0*	*0*
1	*0*	*0*	*0*	*1*
2	*0*	*0*	*1*	*0*
3	*0*	*0*	*1*	*1*
4	*0*	*1*	*0*	*0*
5	*0*	*1*	*0*	*1*
6	*0*	*1*	*1*	*0*
7	*0*	*1*	*1*	*1*
8	*1*	*0*	*0*	*0*
9	*1*	*0*	*0*	*1*
0	*0*	*0*	*0*	*0*

Table 6.1: *Le compteur de dix.*

La bascule A change d'état sur chaque transition 1 vers 0 de la sortie A sauf pour la transition 9 vers 0. Dans ce cas particulier il faut agir sur les entrées de la bascule pour qu'elle ne change pas d'état malgré la transition active sur son horloge. Pour tous les états de 0 à 7 inclus il faut impérativement que les entrées soient telles que le fonctionnement correspondant soit du type T. Pour la combinaison 9 il faut conditionner la bascule pour qu'elle reste en mémoire. Pour la combinaison 8 il existe un degré de liberté. Pour une bascule JK par exemple, la solution consiste à imposer les entrées $J = K = 1$ pour les états 0 à 7, et $J = K = 0$ pour l'état 9 et éventuellement 8. La bascule C change d'état à chaque transition négative de la sortie B. Enfin la bascule D ne peut pas être commandée uniquement par la sortie C. En effet, cette sortie présente une transition négative qui va provoquer le passage de 7 à 8 mais il n'existe pas d'autre transition négative pour le changement de 9 à 0. Il faut donc trouver un autre signal qui présente une transition négative après l'état 9 et après l'état 7, pour commander la bascule, le signal fabriqué avec l'équation logique $C + DA$ satisfait cette condition, comme l'illustre la Figure 6.6.

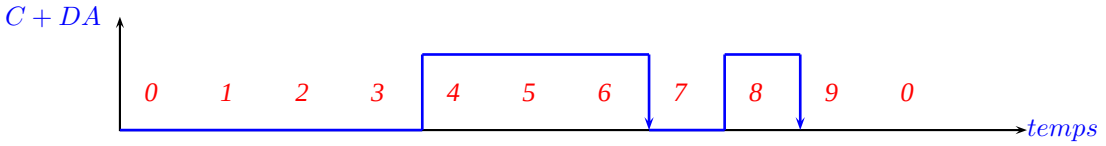

Figure 6.6: *Le signal logique de la fonction $C + DA$.*

Le schéma du compteur modulo 10 réalisé par la méthode du conditionnement des entrées est illustré par la Figure 6.7.

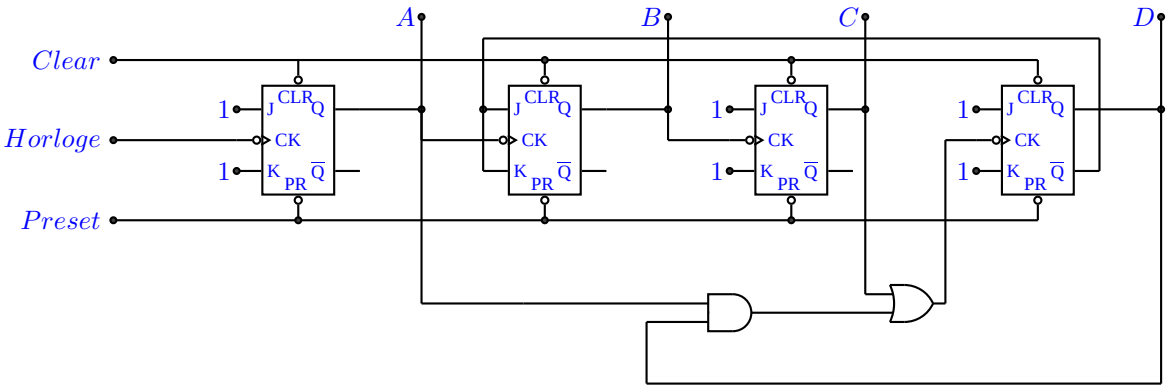

Figure 6.7: *Le compteur modulo* $N = 10$.

6.3 *Les Compteurs Synchrones*

Les compteurs synchrones permettent d'une part d'éliminer les états transitoires des sorties et d'autre part de rendre possible l'exécution d'un cycle quelconque. Pour satisfaire ces exigences et en particulier la première, il est indispensable que toutes les bascules soient synchronisées par le même signal i.e., le signal d'horloge. En conséquence, il ne reste plus qu'un seul degré de liberté pour conditionner l'évolution de chaque bascule. Les entrées synchrones de chaque bascule doivent être calculées pour que le compteur suive la succession d'état prévue. La synthèse d'un compteur synchrone, qui consiste à concevoir un système séquentiel à partir du cycle de fonctionnement souhaité, peut être effectuée par exemple par la méthode de *Marcus*.

Exemple 6.3.1 *Réaliser la séquence suivante avec des bascules JK : 0, 8, 12, 14, 7, 11, 13, 6, 3, 9, 4, 10, 5, 2- 1, 0, ...*

Ces nombres décimaux s'écrivent en binaire avec 4 *bits. En conséquence, cette séquence impose l'utilisation de quatre bascules, d'où la Table 6.2. Nous remarquons que* $\emptyset$ *est l'état indifférent i.e., il peut prendre la valeur* 0 *ou* 1.

Numéro	*Sorties*				*Entrées*							
	A	B	C	D	J_A	K_A	J_B	K_B	J_C	K_C	J_D	K_D
0	0	0	0	0	1	$\emptyset$	0	$\emptyset$	0	$\emptyset$	0	$\emptyset$
8	1	0	0	0	$\emptyset$	0	1	$\emptyset$	0	$\emptyset$	0	$\emptyset$
12	1	1	0	0	$\emptyset$	0	$\emptyset$	0	1	$\emptyset$	0	$\emptyset$
14	1	1	1	0	$\emptyset$	1	$\emptyset$	0	$\emptyset$	0	1	$\emptyset$
7	0	1	1	1	1	$\emptyset$	$\emptyset$	1	$\emptyset$	0	$\emptyset$	0
11	1	0	1	1	$\emptyset$	0	1	$\emptyset$	$\emptyset$	1	$\emptyset$	0
13	1	1	0	1	$\emptyset$	1	$\emptyset$	0	1	$\emptyset$	$\emptyset$	1
6	0	1	1	0	0	$\emptyset$	$\emptyset$	1	$\emptyset$	0	1	$\emptyset$
3	0	0	1	1	1	$\emptyset$	0	$\emptyset$	$\emptyset$	1	$\emptyset$	0
9	1	0	0	1	$\emptyset$	1	1	$\emptyset$	0	$\emptyset$	$\emptyset$	1
4	0	1	0	0	1	$\emptyset$	$\emptyset$	1	1	$\emptyset$	0	$\emptyset$
10	1	0	1	0	$\emptyset$	1	1	$\emptyset$	$\emptyset$	1	1	$\emptyset$
5	0	1	0	1	0	$\emptyset$	$\emptyset$	1	1	$\emptyset$	$\emptyset$	1
2	0	0	1	0	0	$\emptyset$	0	$\emptyset$	$\emptyset$	1	1	$\emptyset$
1	0	0	0	1	0	$\emptyset$	0	$\emptyset$	0	$\emptyset$	$\emptyset$	1

Table 6.2: *Table pour la séquence avec les bascules* JK.

La combinaison 1111 *ou* 15 *en décimal n'apparaît jamais dans la séquence désirée. C'est donc une combinaison disponible pour la simplification et dans les diagrammes de* Karnaugh *cette case sera remplie avec un* 0*. Après simplification de* 8 *diagrammes de* Karnaugh *d'entrées* A, B, C, D *et de sorties* $J_A, K_A, J_B, K_B, J_C, K_C, J_D$ *et* K_D*, des états équivalents et des fonctions logiques, les entrées* J *et* K *des bascules ont pour équation 6.3.1.*

$$\begin{array}{lcclcc} J_A & = & \overline{C \oplus D} & K_A & = & C \oplus D \\ J_B & = & A & K_B & = & \overline{A} \\ J_C & = & B & K_C & = & \overline{B} \\ J_D & = & C & K_D & = & \overline{C} \end{array} \tag{6.3.1}$$

Si, à la mise sous tension, le compteur se positionne dans la combinaison interdite compte tenu du schéma, il y restera malgré les impulsions d'horloge. Il faut donc prévoir une action manuelle ou automatique pour retourner dans le cycle. Un circuit de décodage de la combinaison interdite permet de remettre le compteur dans son cycle si par malheur cette combinaison apparaissait. Dans le schéma de la Figure 6.8 la remise en cycle est faite par une remise à zéro de toutes les bascules puisque la combinaison 0_{10} *appartient au cycle.*

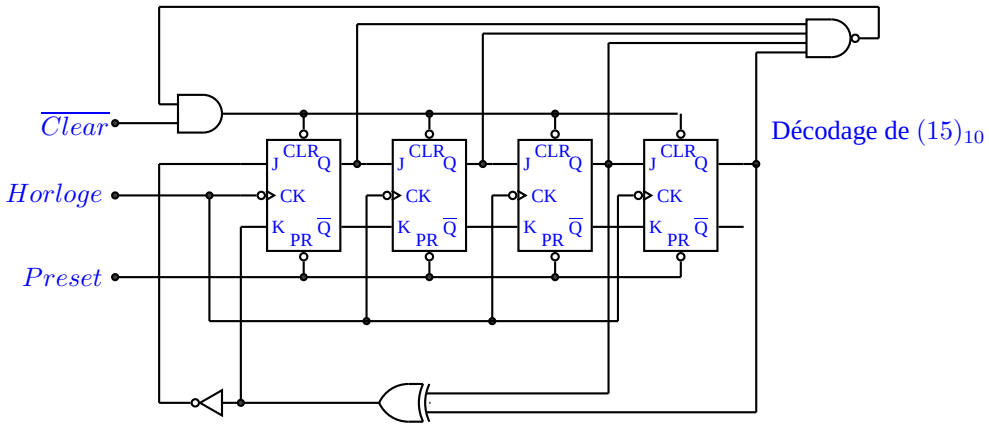

Figure 6.8: *Le compteur de l'exemple.*

6.3.1 *Initialisation d'un Compteur*

L'état initial d'un compteur est défini à l'aide des entrées asynchrones d'initialisation (*Preset et Clear*).

6.4 *Mise en Cascade de Compteur*

Les compteurs commercialisés délivrent bien évidemment la valeur contenue dans le compteur, et éventuellement une information supplémentaire permettant la mise en cascade de plusieurs boîtiers, soit *Ripple Count Enable - RCE* cette sortie est également appelée *Terminal Count - TC*. La commande des différents boîtiers de comptage peut se faire suivant deux grands principes : - la mise en cascade asynchrone i.e., la sortie RCE d'un boîtier sert d'horloge du boîtier suivant, comme l'illustre la Figure 6.9. La conséquence de la mise en cascade asynchrone est que les sorties du second boîtier sont décalées dans le temps par rapport aux sorties du premier.

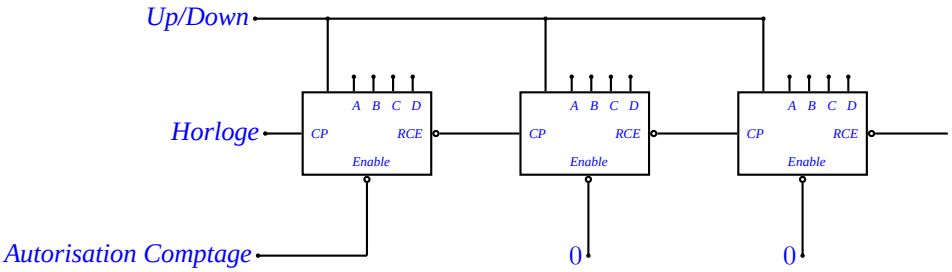

Figure 6.9: *La connexion de ripple count enable.*

La deuxième méthode, la mise en cascade synchrone. Comme l'horloge est commune à tous les boîtiers, il faut conditionner l'évolution du second boîtier par la commande (*Enable* ou *Chip Enable*) d'autorisation de comptage, comme l'illustre la Figure 6.10.

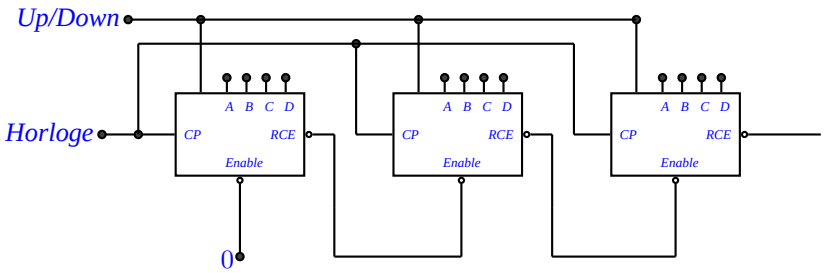

Figure 6.10: *Le compteur en cascade synchrone.*

6.5 *Conclusion*

Un autre type d'application de circuit séquentiel est celui que nous venons d'étudier. Il ne faut pas oublier que les compteurs sont systèmes dynamiques et donc ils ont une limitation de fréquence. Pour réaliser un système complet, nous avons également besoin d'un système de contrôle qui fera objet de l'étude du prochain chapitre.

Chapitre 7

Les Séquenceurs

7.1 *Introduction*

Les séquenceurs sont des systèmes séquentiels dont l'état évolue en fonction d'événements i.e., ils peuvent être différents types horloge, contact, interruption, etc et que nous appelons actions. La séquence des états et des transitions marquant les changements d'états (les actions) s'appelle un automate ou séquenceur ou encore machine d'états. (Par exemple, le distributeur de café dans le couloir de l'école. Nous trouvons principalement des séquenceurs dans les systèmes suivants :

$\Rightarrow$ 1. Dans les automatismes ou processus automatiques industriels où ils jouent le rôle d'un automate capable de diriger les opérations qui doivent se dérouler dans un ordre prévu à l'avance ;

$\Rightarrow$ 2. Dans les calculateurs où ils sont chargés d'un rôle d'organisateur de la succession des opérations à réaliser selon un programme préétabli.

Un séquenceur n'est qu'exceptionnellement seul. En général il commande un système. Dans la plupart des cas, ce dernier l'informe de son état, par exemple la fin d'exécution d'une tâche, la présence d'un débordement ou d'un résultat nul, etc. Le séquenceur doit pouvoir, si nécessaire, prendre des décisions suivant l'état du système, comme l'illustre la Figure 7.1.

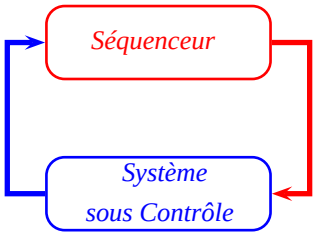

Figure 7.1: *Liaison entre le séquenceur et le système sous contrôle.*

7.2 *Synthèse d'un Séquenceur*

Soit un système de production qui doit faire l'insertion de deux types de pièces différents A et B et qui sont sélectionnés par les capteurs, après que la température de la machine arrive à 65° Celsius. Le capteur C_A pour l'insertion de la pièce A et le capteur C_B pour l'insertion de la pièce B. Á la fin de l'insertion, le système recevra l'information de fin d'insertion de la pièce ($Fin\_Insertion$). Le système est initialisé à l'état nommée $Initial$ et il doit retourner à cet état après avoir reçu l'information de fin d'insertion.

Nous supposons que les capteurs des pièces pourront tomber en panne, la valeur logique quand ils sont en panne sera 1, et que les pièces A et B ne manquerons pas. Puis, nous définissions les conditions suivantes pour le système :

$\Rightarrow$ 1. La valeur logique 1 pour le capteur (pièces et température) actif et la valeur 0 pour le capteur au repos ;

$\Rightarrow$ 2. Le signal de fin d'insertion aura la valeur logique 1 quand le système a fini de réaliser l'insertion ;

$\Rightarrow$ 3. L'action de commande pour l'insertion de la pièce A ou B aura la valeur logique 1 quand elle est active et égale 0 i.e., au repos ;

$\Rightarrow$ 4. Les valeurs initiales pour les entrées et sorties sont à la valeur logique 0 ;

$\Rightarrow$ 5. Il faut utiliser des multiplexeurs pour sélectionner les informations qui arriveront aux bascules ;

$\Rightarrow$ 6. Les bascules seront initialisées pour le signal *Power Up Reset* qui aura la valeur logique 0 quand il sera activé.

7.2.1 *Automate ou Graphe d'Influence*

Le système est composé par quatre entrées C_A, C_B, $Fin\_Insertion$ et *Power Up Reset*. Les deux commandes de sortie sont nommées C_{MA} et C_{MB}. A partir des ces informations, nous auront développé l'automate comme l'illustre la Figure 7.2.

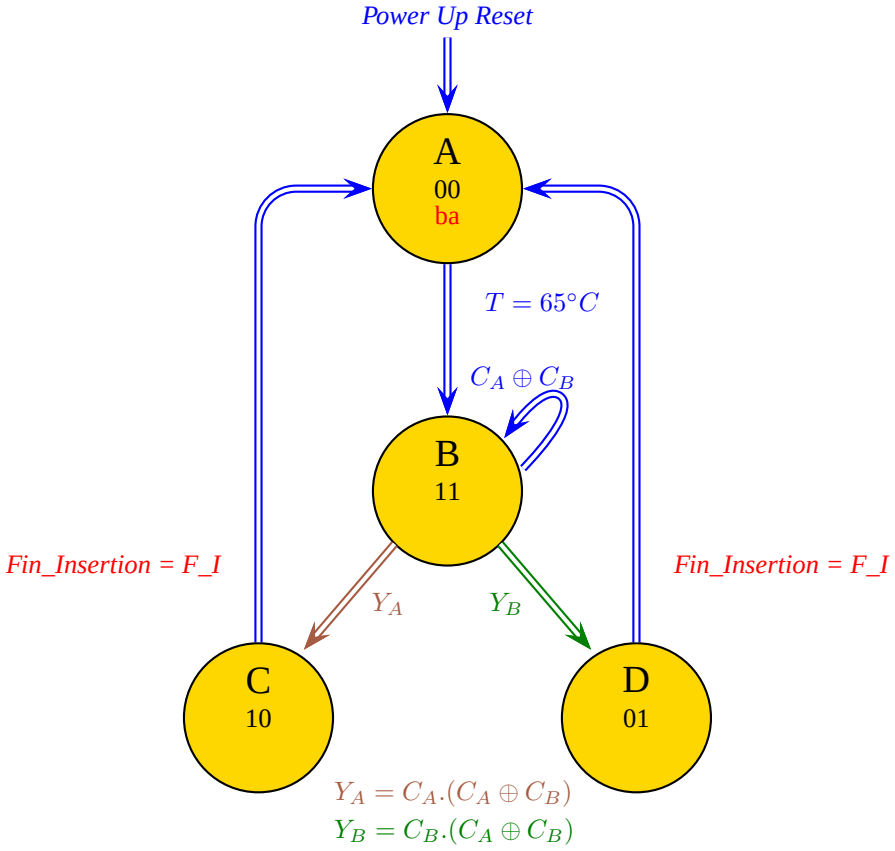

Figure 7.2: *L'automate ou graphe d'influence.*

7.2.2 *Diagramme d'États*

Le système peut être résumé par la Table d'état 7.3.

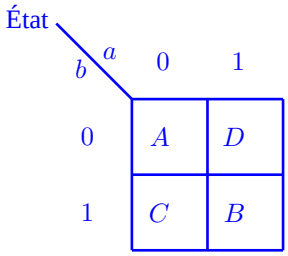

Figure 7.3: *Diagramme des états.*

7.2.3 *Les Équations du Système*

Nous utiliserons les bascules type $D\ edge\ triggered$ pour développer le système et obtenir les équations, comme les illustrent les diagrammes de $Karnaugh$ des Figures 7.4 et 7.5 et leurs équations respectives 7.2.1 et 7.2.2.

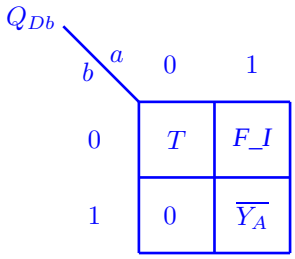

Figure 7.4: *Diagramme pour la bascule Q_{Db}.*

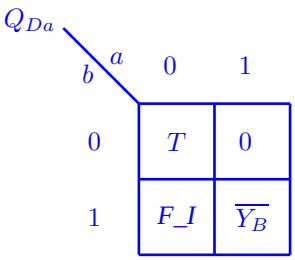

Figure 7.5: *Diagramme pour la bascule Q_{Da}.*

$$Q_{Db} = \overline{a}.\overline{b}.T + a.b.\overline{Y_B} + \overline{a}.b + a.\overline{b}.F\_I \tag{7.2.1}$$

$$Q_{Da} = \overline{a}.\overline{b}.T + a.b.\overline{Y_A} + a.\overline{b} + +\overline{a}.b.F\_I \tag{7.2.2}$$

7.2.4 *Réalisation du Séquenceur*

A partir des équations 7.2.1 et 7.2.2, nous faisons la réalisation comme illustre la Figure 7.6.

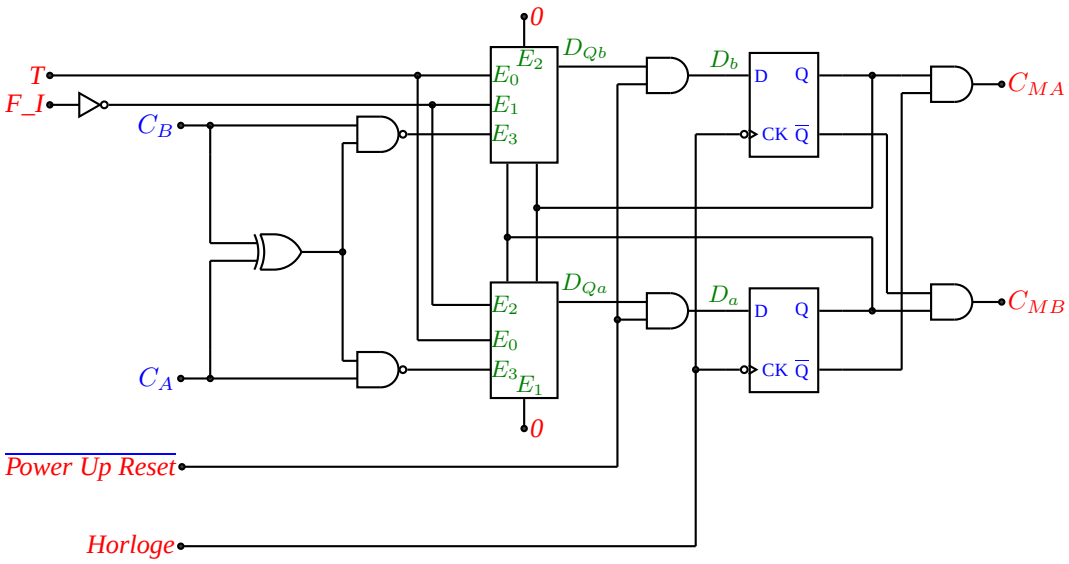

Figure 7.6: *Le synoptique du projet.*

7.3 *Conclusion*

A partir de la connaissance des systèmes logiques combinatoire et séquentiel, nous devons être capable de faire la réalisation de tous les types d'applications. Mais, nous venons d'étudier la base des systèmes logiques. Pour savoir toutes les contraintes et les autres techniques nécessaires pour faire vraiment la totalité de projets, il faut faire une étude plus complete et plus approfondie.

Partie III

Travail

Chapitre 8

Travail Dirigé

Étude des circuits logiques. Les concepts de la logique combinatoire et séquentielle permettont de comprendre les différents éléments de base utilisé par les ordinateurs avec un ou plusieurs processeurs. Nous utiliserons deux classes d'exercices : - Théorique et Pratique. Par conséquence, cette étude est la base de connaissance pour le cours d'Architecture de Systèmes Informatiques.

8.1 *TD* $N°1$

8.1.1 *Travail Résolu*

Travail Théorique

1. Démontrez : Pour chaque élément A dans l'ensemble B, $A + A = A$ et $A.A = A$.

2. Démontrez : Pour chaque élément A dans l'ensemble B, $A + 1 = 1$ et $A.0 = 0$.

3. Démontrez : Pour la paire d'éléments A et B dans B, $A + AB = A$ et $A(A + B) = A$.

8.1.2 *Travail Pratique*

Faire la minimisation par le diagramme de *Karnaugh*.

1. Forme canonique somme de produits.
   1.1. $f(A, B, C) = \sum_{4} \mathrm{m}(0, 1, 5, 7)$

2. Forme canonique produit de sommes.
   2.1. $f(A, B, C, D) = \prod_{4} \mathrm{M}(2, 3, 5, 7)$

8.1.3 *Entraînez-vous !*

8.1.4 *Entraînez-vous - Théorique*

1. Démontrez le théorème *De Morgan*.

8.1.5 *Entraînez-vous - Pratique*

1. Faire la minimisation par la Table de *Karnaugh*.

   Forme canonique somme de produit.

   1.1. $f = \sum_{8} \mathrm{m}(0, 1, 3, 5, 8, 9, 10, 12)$

   1.2. $f = \sum_{9} \mathrm{m}(1, 2, 7, 11, 12, 15, 20, 29, 30)$

   1.3. $f = \sum_{10} \mathrm{m}(1, 2, 7, 11, 12, 15, 19, 25, 26, 27)$

   Forme canonique produit de somme.

   2.1. $f = \prod_{8} \mathrm{M}(0, 1, 3, 5, 8, 9, 10, 12)$

   2.2. $f = \prod_{9} \mathrm{M}(1, 2, 7, 11, 12, 15, 20, 29, 30)$

   2.3. $f = \prod_{10} \mathrm{M}(1, 2, 7, 11, 12, 15, 19, 25, 26, 27)$

8.2 *TD* $N°2$

8.2.1 *Travail Résolu*

Travail Théorique

1. Soit un nombre N qui est noté en base s. Démontrez la conversion en base r.

Travail Pratique

1. Faire la conversion $553_{(10)}$ en base 2 et en base 5.

2. Faire la conversion $1306_{(10)}$ en base 12 et en base 14.

3. Faire la conversion $23,6751_{(10)}$ en base 2.

4. Réaliser les opérations en base 2 suivantes :

   a. $32_{(10)} + 13_{(10)}$.

   b. $32_{(10)} + 21,5_{(10)}$.

   c. $32_{(10)} - 15_{(10)}$.

   d. $32_{(10)}$. $19_{(10)}$.

8.2.2 *Entraînez-vous !*

8.2.3 *Entraînez-vous - Théorique*

1. Démontrez la conversion des nombres fractionnaires d'une base S à une autre base r.

8.2.4 *Entraînez-vous - Pratique*

1.

   1.1. Faire la conversion $134_{(10)}$ en base 2 et en base 5.

   1.2. Faire la conversion $941_{(10)}$ en base 12 et en base 14.

   1.3. Faire la conversion $0,00625_{(10)}$ en base 2.

1.4. Réaliser les opérations en base 5 suivantes :

a. $243_{(10)} + 117_{(10)}$.

b. $274_{(10)} - 315_{(10)}$.

c. $234_{(10)}/121_{(10)}$.

d. $294_{(10)}$.$11_{(10)}$.

1.5. Faire le complément à deux, du nombre $0,01110111_{(2)}$.

2.

2.1. Soient 2 (deux) bits A et B. Donner le circuit élémentaire réalisant la somme arithmétique entre A et B, nommé demi-additionneur.

2.2. Soit en plus de A_i et B_i, le bit R_i figurant la retenue de la somme élémentaire précédente entre A_{i-1} et B_{i-1} lorsque nous le désirons faire la somme de 2 (deux) mots $A = A_{n-1}A_{n-1}...A_0$ et $B = B_{n-1}B_{n-1}...B_0$.

2.2.1. Donner le circuit additionneur complet entre A_i, B_i et R_i.

2.2.2. Donner l'additionneur à propagation de retenue entre A et B.

2.2.3. Donner l'additionneur anticipée (plus rapide) entre les mots A et B.

8.3 *TD* $N°3$

8.3.1 *Travail Résolu*

Travail Pratique

1. Projeter le compteur code Gray de 2 (deux) bits en utilisant les bascules type D.
2. Utiliser les bascules type T pour projeter le compteur code Gray de 2 (deux) bits.
3. Projeter un compteur qui exécute la séquence suivante : $1-3-5-7-2-4-6-0-1$.
    3.1. Utiliser les bascules type D;
    3.2. Utiliser les bascules type JK.

8.3.2 *Entraînez-vous !*

Entraînez-vous - Pratique

1. Utiliser les bascules type JK pour projeter le compteur code Gray de 2 (deux) bits.
2. Projeter un générateur de parité pour un message de 3 (trois) bits.
    2.1. Utiliser les bascules type D;
    2.2. Utiliser les bascules type JK.
3. Projeter un compteur qui exécute la séquence suivante : $0-1-3-5-15-13-11-9-7-0$.
    3.1. Utiliser les bascules type D;
    3.2. Utiliser les bascules type JK.

8.4 *TD* $N°4$

8.4.1 *Travail Résolu*

Travail Pratique

Projeter le circuit de contrôle conforme présenté sur la Figure 8.1, en utilisant les bascules type D.

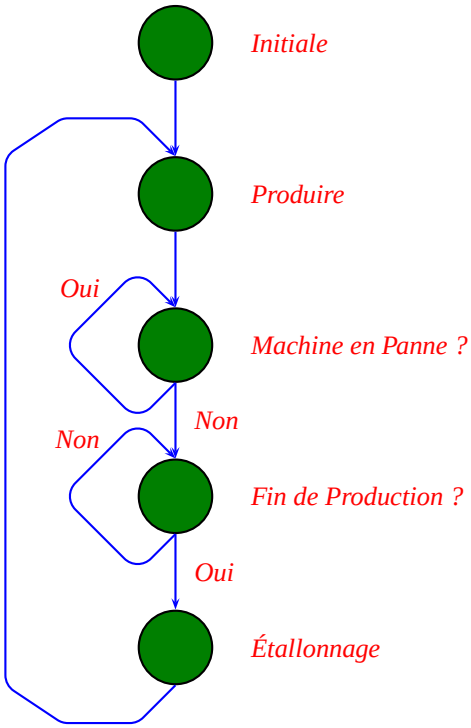

Figure 8.1: *Unité de production.*

8.4.2 *Entraînez-vous !*

Projeter le circuit de contrôle conforme présenté sur la Figure 8.2, en utilisant les bascules type D et JK.

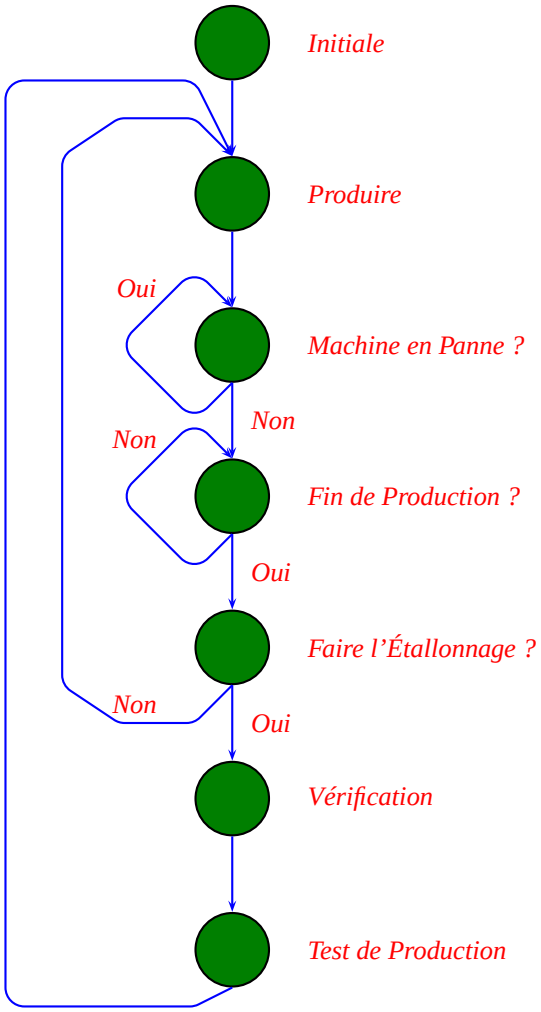

Figure 8.2: *Unité de production II.*

8.5 TD $N°5$

8.5.1 *Travail Résolu*

Travail Pratique

Projeter le circuit universel bidirectionnel, en utilisant les bascules type SR.

8.5.2 Entraînez-vous !

Projeter le circuit universel bidirectionnel, en utilisant les bascules type JK.

8.6 *TD* $N°6$ *et TD* $N°7$

8.6.1 *Projet*

Projeter un microprocesseur, comme l'illustre la Figure 8.3. Le Tableau 8.1 décrit les caractéristiques des registres et le Tableau 8.3 définit les instructions à réaliser.

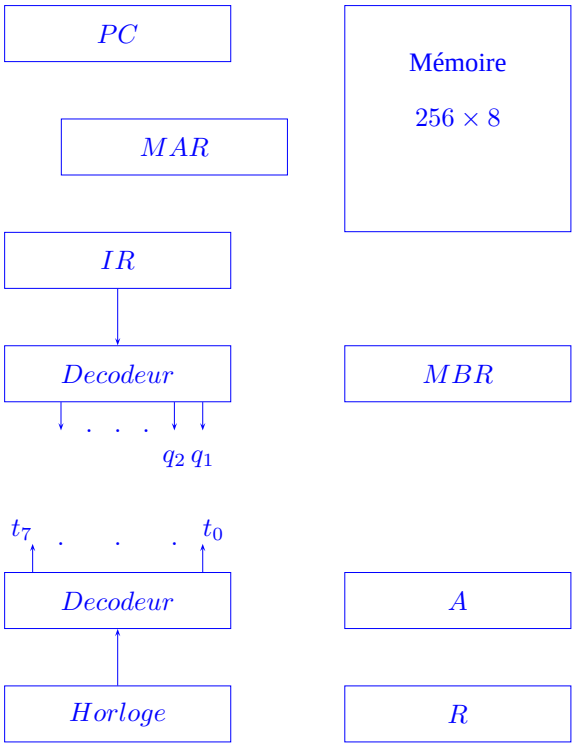

Figure 8.3: *Diagramme du microprocesseur.*

Symbole	Nombre de bits	Nom	Fonction
MAR	8	Registre d'Adresse de Mémoire	Maintenir l'Adressent de la Mémoire
MBR	8	Registre Tampom de Mémoire	Maintenir le Contenu de la Mémoire
A	8	Registre A	Registre du Processeur
R	8	Registre R	Registre du Processeur
PC	8	Compteur de Programme	Maintenir l'Adrresse de l'Instruction
IR	8	Registre Instruction	Maintenir le Code d' l'Opération
T	8	Compteur de Temps	Générateur de la Séquence

Table 8.1: *Les registres.*

Code Opération	Mnémonique	Description	Fonction
00000001	MOV R	Déplacez R vers A	$A \leftarrow R$
00000010	LDI OPRD	Chargez OPRD dans A	$A \leftarrow OPRD$
00000011	LDA ADRS	Chargez l'Opérande Spécifiée par ADRS dans A	$A \leftarrow M[ADRS]$

Table 8.2: *Les instructions.*

Chapitre 9

Travail Pratique

Étude des circuits logiques. Le concept de la logique combinatoire et séquentielle permet de comprendre les différents éléments de base utilisés par les ordinateurs avec un ou plusieurs processeurs. Nous utiliserons le montage et la simulation pour comprendre les principales caractéristiques des circuits communs de la logique combinatoire. Par conséquenct, cette étude est la base de connaissance pour l'Architecture de Systèmes Informatiques.

9.1 *Lay-out Plaquette*

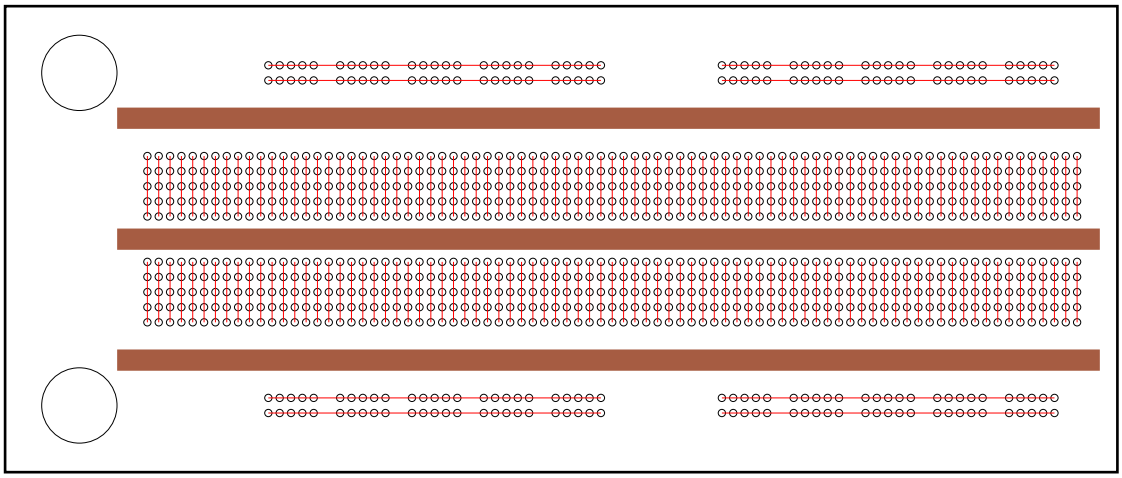

Figure 9.1: *Schéma de la plaquette.*

9.2 *TP* $N°1$

9.2.1 *Étude Théorique*

Les Additionneurs

$\Rightarrow$ 1. Demi-additionneurs

Soient A_n et B_n 2 (deux) bits à additionner. L'expression binaire de $(A_n + B_n)$ s'écrit (R_n, S_n) où R_n représente la retenue (*carry*) générée de l'opération et S_n le résultat de l'addition.

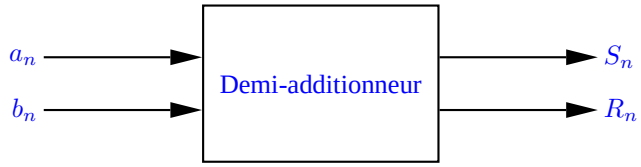

Figure 9.2: *Le demi-additionneur.*

$\Rightarrow$ 2. Développement Additionneur

a. Donner la table de vérité du demi-additionneur, i.e., pas de prise en compte de l'éventuelle retenue issue d'un étage additionneur précédent dans le cadre d'une addition de 2 (deux) mots binaires ;

b. Tracer le diagramme de *Karnaugh* des deux fonctions R_n et S_n ;

c. Donner les fonctions logiques correspondantes R_n et S_n en utilisant des portes logiques *AND*, *OR* et *NOT* ;

d. Donner le schéma symbolique de la réalisation.

$\Rightarrow$ 3. Additionneur Complet

Soient A_n et B_n bits additionneur en tenant compte ici de la retenue précédente R_{n-1} issue d'un étage additionneur précédent (dans le cadre d'une addition de 2 (deux) mots binaires).

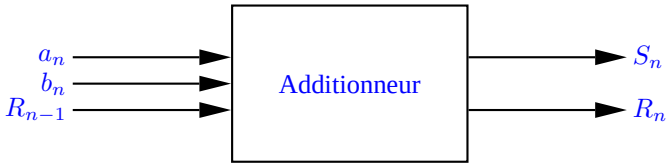

Figure 9.3: *L'additionneur complet.*

$\Rightarrow$ 3. Développement Additionneur Complet

a. Donner la table de vérité de l'additionneur complet, i.e., prise en compte de l'éventuelle retenue issue d'un étage additionneur précédent dans le cadre d'une addition de 2 (deux) mots binaires ;

b. Tracer le diagramme de *Karnaugh* des deux fonctions R_n et S_n ;

c. Donner les fonctions logiques correspondantes R_n et S_n en utilisant des portes logiques *AND*, *OR* et *NOT* ;

d. Donner le schéma symbolique de la réalisation.

$\Rightarrow$ 4. Additionneur de 2 Mots de 2 Bits

Soient 2 (deux) mots de 2 (deux) bits $A = A_1A_0$ et $B = B_1B_0$ à additionner $(A_1B_1 : MSB)$. En utilisant les résultats précédents, donner les expressions des sorties et de la retenue générée : $S_1, S0$ et R_1 de l'additionneur de 2 (deux) mots de 2 (deux) bits où S_1 est *MSB*, après avoir établi sa table de vérité. [1]

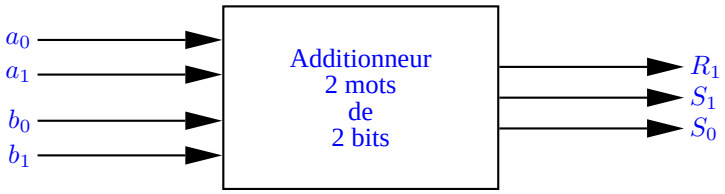

Figure 9.4: *L'additionneur 2 (deux) mots de 2 (deux) bits.*

a. Donner le schéma de la réalisation avec les composants TTL (*Logique Transistor Transistor*).

Les Codeurs

1. Un code est la représentation d'un nombre tel qu'à chaque nombre correspond une configuration et une seule, et qu'à chaque configuration ne correspond qu'un seul nombre.

2. En général, le codage d'un nombre décimal s'effectue en codant chaque chiffre qui le compose. Le nombre codé est obtenu en juxtaposant leurs représentations.

3. Pour coder en binaire les chiffres de 0 à 9, il faut au moins 4 (quatre) bits. Il existe de nombreuses possibilités pour coder ces chiffres. Cependant, seuls quelques unes revêtent une grande importance.

4. Le transcodage est une opération qui consiste à passer d'un code à un autre code.

[1] Mano, M. M., "Digital Logic an Computer Design," Prentice-Hall, Inc., Englewood Cliffs, N.Y, 1972.

Transcodeur BCD - 7 Segments

Un afficheur 7 (sept) segments est constitué de 7 (sept) *LEDs* ayant tous un point commun : l'anode *(A)* ou la cathode *(C)*. Si l'afficheur est à anodes communes, il est nécessaire d'inverser *(complémenter)* le signal à afficher pour qu'un *LED* allumé corresponde à une logique.

Figure 9.5: *Décodeur de 7 segments.*

Développement d'un Codeur

a. Donner la table de vérité d'un transcodeur permettant de passer du code *BCD (Décimal Codé Binaire)* au code permettant l'affichage sur 7 (sept) segments *a* à *g* d'un mot *ABCD* de 4 bits et soit *A le MSB* ;

b. Tracer les tableaux de *Karnaugh* de chaque segment en simplifiant les fonctions au maximum, et en faisant apparaître si possible les fonctions logiques *OU exclusives* ;

c. En déduire les fonctions correspondantes ;

d. Donner le schéma symbolique de réalisation du transcodeur.

Les Comparateurs

Nous désirons réaliser une étude permettant de comparer 2 (deux) mots binaires de 2 (deux) bits ($A = A_1A_0$ et $B = B_1B_0$ où A_1, B_1 sont *MSB*). Cette étude doit permettre d'identifier les 3 (trois) cas suivants : $A > B, A = B$ et $A < B$.

$\Rightarrow$ 1. Développement Comparateur

a. Donner la table de vérité ;

b. Tracer les diagrammes de *Karnaugh* des 3 (trois) fonctions de comparaison en les simplifiant au maximum ;

c. Donner les fonctions correspondantes ;

d. Donner le schéma symbolique de réalisation du comparateur.

Étude Expérimentale

Le schéma synoptique de la Figure 9.6 utilise un ensemble de composants de la famille *TTL*. Réaliser le projet du circuit, implémenter la montage et faire la simulation (*CircuitMark*).

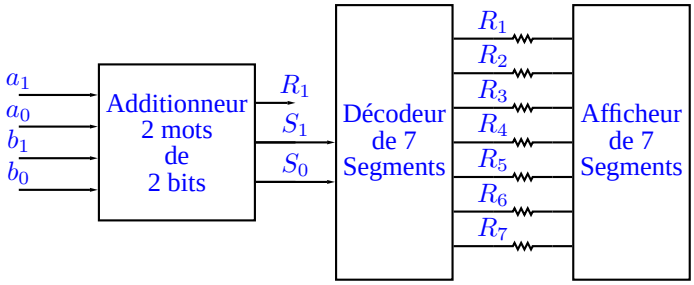

Figure 9.6: *Le circuit du montage.*

$\Rightarrow$ 1. Liste de Matériel

Pour développer le montage, nous avons besoin du matériel suivant :

- Oscilloscope;
- Générateur de signaux basses fréquences (*GBF*);
- Alimentation stabilisée ;
- Multimètre;
- Moniteurs *MS05* (plaquette de câblage);
- Câbles :
    - 2 (deux) sondes de l'oscilloscope ;
    - 6 (six) fils banane;
    - 1 (un) BNC- banane;
- Composants :
    - 1 (un) SN74ALS86N -> 4 (quatre) *XOR* à 2 entrées;
    - 1 (un) SN74ALS32N -> 4 (quatre) *OR* à 2 entrées;
    - 1 (un) SN74ALS08N -> 4 (quatre) *AND* à 2 entrées;
    - 1 (un) SN74LS47N -> Decodeur 7 (sept) segments;
    - 1 (un) -> Plaquette d'afficheur 7 (sept) segments;
    - 4 (quatre) résistances de *4K7*Ω et *1/4 Watt*.

9.3 *TP* $N^{\circ}2$

9.3.1 *Étude Théorique*

Les Comparateurs

Soient deux nombres binaires A et B, le comparateur numérique permet de savoir si A est supérieur, égal ou inférieur à B. La synthèse du comparateur peut être effectuée de la manière suivante:

Considérons tout d'abord deux nombres de 1 (un) digit, comme l'illustre la Figure 9.7 ; soient A_0 et B_0, lorsque:

Si $A_0 = B_0$,
alors $\overline{A_0}\,\overline{B_0} + A_0B_0 = 1 = E_0$

Si $A_0 > B_0$,
alors $A_0 = 1$, $B_0 = 0$ soit
$C_0 = A_0\overline{B_0} = 1$

A partir de cette étude préliminaire, nous proposons d'examiner la comparaison de deux nombres A et B de quatre digits : $A = A_3A_2A_1A_0$ et $B = B_3B_2B_1B_0$; A_3 et B_3 étant respectivement les MSB de A et de B. Lorsque $A > B$, nous pourrons avoir les possibilités suivantes:

1. Si $A_3 > B_3$ implique $C_3 = 1$;

2. Si $A_3 = B_3$ et $A_2 > B_2$ ce qui implique $E_3 = 1$ et $C_2 = 1$;

3. Si $A_3 = B_3$, $A_2 = B_2$ et $A_1 > B_1$ ce qui implique $E_3 = E_2 = 1$ et $C_1 = 1$;

4. Si $A_3 = B_3$, $A_2 = B_2$, $A_1 = B_1$ et $A_0 > B_0$ ce qui implique $E_3 = E_2 = E_2 = 1$ et $C_0 = 1$.

Soit f la sortie du comparateur. La fonction logique exprimant $f = 1$ lorsque $A > B$ (équation 9.3.1).

$$f = C_3 + E_3.C_2 + E_3.E_2.C_1 + E_3.E_2.E_1.C_0 \tag{9.3.1}$$

Cette expression est facilement généralisée pour le comparateur de N digits. Ce comparateur est réalisable à partir du comparateur élémentaire de deux nombres de 1 (un) digit.

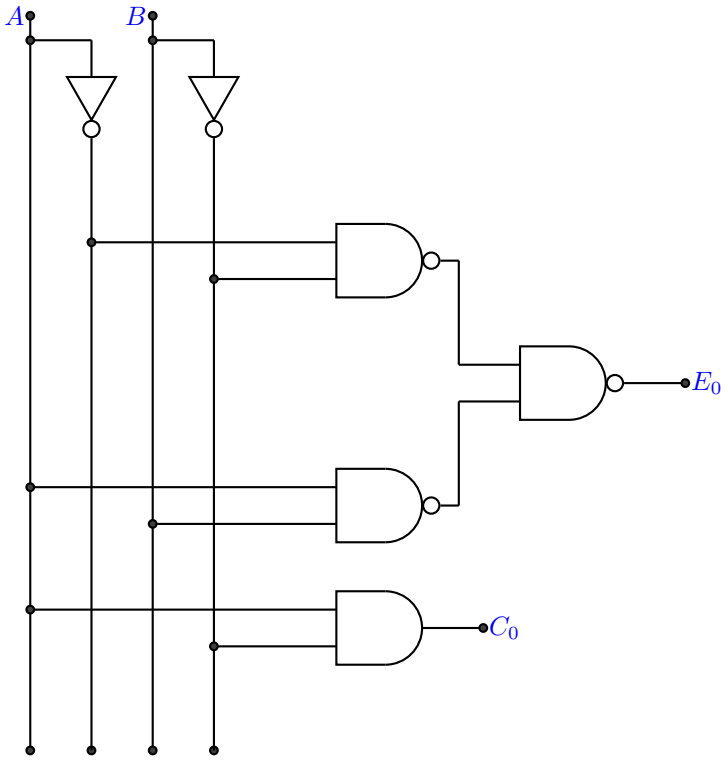

Figure 9.7: *Le comparateur de 2 (deux) nombres d'un bit.*

Le Multiplexeur

Ce sont des circuits logiques combinatoires qui possèdent N entrées (en général de 2 à 16), n entrées de commande (en général de 1 à 4) et une sortie. Ils transmettent à leur sortie une de ces entrées sélectionnées par une adresse codée appliquée aux n entrées de commande (appelées entrées de sélection ou plus simplement adresses, comme l'illustre la Figure 9.8).

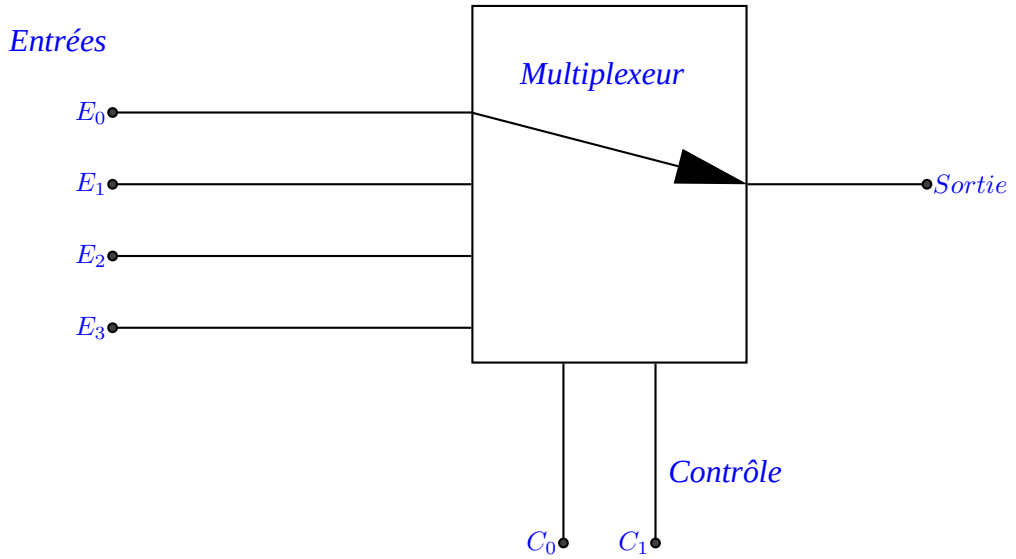

Figure 9.8: *Le multiplexeur à quatre entrées.*

Troisième État

En plus des deux états traditionnels qui sont 1 et 0, nous ajoutons un état à haute impédance. Cette propriété permet de relier plusieurs sorties à condition qu'un seul circuit soit *validé* à la fois. Cela laisse entrevoir des applications très intéressantes lorsque les informations sont transmises par des lignes communes appelées *bus*, comme l'illustre la Figure 9.9 auxquelles sont reliées de nombreux circuits. Certains *buffers* sont inhibés par un niveau haut, d'autres sont inhibés par un niveau bas. Les *bus* jouent un rôle très important dans les systèmes programmables.

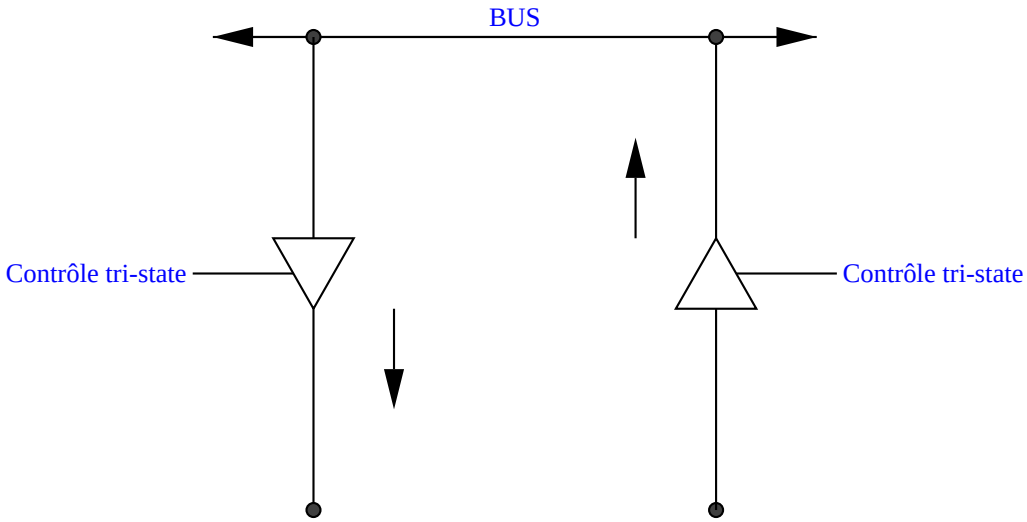

Figure 9.9: *Buffer à troisième état.*

9.3.2 *Étude Expérimentale*

Les schémas synoptiques de la Figure 9.10 et de la Figure 9.11 utilisent un ensemble de composants de la famille *TTL*. Réaliser le projet des circuits, implementer la montage et faire la simulation (*CircuitMark*).

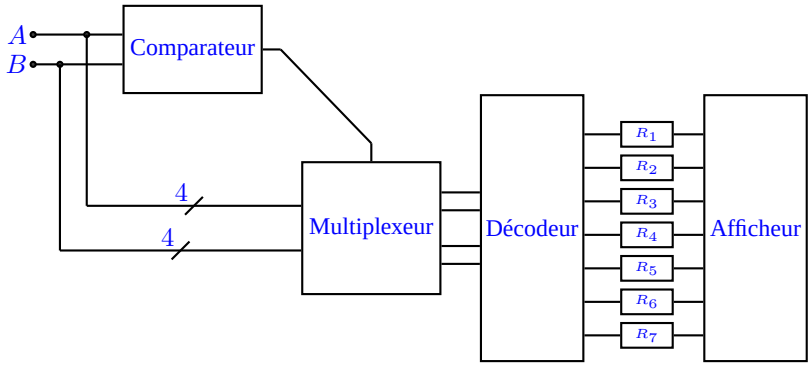

Figure 9.10: *Le circuit du montage avec le composant multiplexeur.*

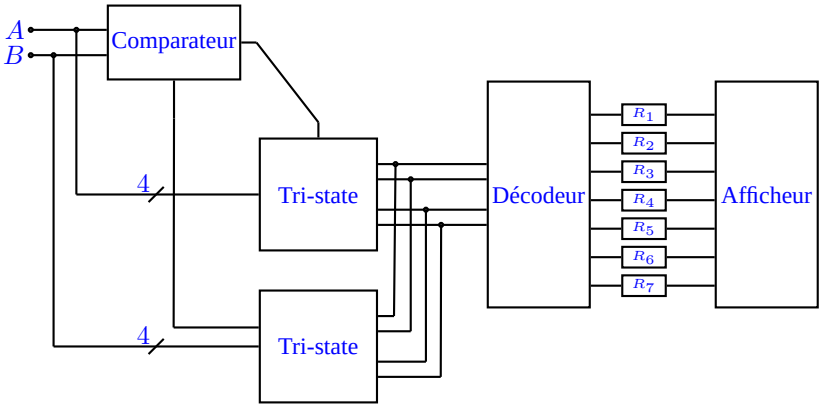

Figure 9.11: *Le circuit du montage avec le composant troisième état.*

Description du Problème

Projeter deux circuits qui affichent la valeur la plus grande entre les mots A et B de 4 (quatre) bits. Pour les projets, utiliser le bloc fonctionnel *SSI* multiplexeur et en suite le bloc fonctionnel *SSI* troisième état. Le circuit aussi permettra de visualiser le contenu par un afficheur de sept segments.

Liste de Matériel

Pour développer le montage, nous avons besoin du matériel suivant :

- Oscilloscope;

- Alimentation stabilisée ;

- Multimètre;
- Moniteurs *MS05* (plaquette de câblage);
- Câbles :
    - 2 (deux) sondes fr l'oscilloscope;
    - 6 (six) fils banane;
    - 1 (un) BNC-banane;
- Composants :
    - 1 (un) SN74ALS85N -> comparateur de 4 (quatre) bits;
    - 1 (un) SN74ALS157N -> 4 (quatre) *MUX*;
    - 1 (un) SN74ALS04N -> 6 (six) *Inverseur*;
    - 1 (un) SN74ALS244N -> 8 (huit) *Buffer Troisième État*;
    - 1 (un) SN74LS47N -> Décodeur 7 (sept) segments;
    - 1 (un) -> Plaquette d'afficheur 7 (sept) segments;
    - 8 (huit) résistances de *4K7*Ω et *1/4 Watt*.

9.4 *TP* $N°3$

9.4.1 *Les Compteurs*

Les compteurs sont les applications directes des bascules étudiées dans le chapitre précédent. Il ne faut pas croire que les compteurs sont destinés uniquement au comptage, ils jouent en électronique numérique un rôle fondamental car ils gèrent les séquences des opérations, ils divisent les fréquences, ils participent à de nombreuses manipulations mathématiques, à des conversions de code, à des conversions analogiques numériques et numériques analogiques, etc ...

Un compteur binaire est dit modulo N lorsqu'il peut compter jusqu'à $N-1$; la énième impulsion remet le compteur à zéro. En général, $N = 2^n$, où n représente le nombre d'étages constituant le compteur qui possède 2^n états possibles. Nous distinguons deux catégories de compteurs : les compteurs synchrones et les compteurs asynchrones. Nous proposons d'étudier les principaux circuits de compteurs et leurs caractéristiques, tandis que les registres à décalage qui permettent aussi la conception de certains compteurs particuliers seront exposés à la session suivante.

Compteur Diviseur par 2

Pour mieux comprendre le rôle des bascules dans le comptage, reprenons une bascule T ; lorsque $T = 0$ une impulsion appliquée à l'entrée H (*Horloge*) n'a aucun effet sur la sortie de la bascule, par contre lorsque $T = 1$, la sortie change d'état à chaque impulsion. Supposons que le signal appliqué à H est un signal rectangulaire de fréquence f. Nous aurons à la sortie de la bascule un signal rectangulaire de fréquence $f/2$. Ce compteur effectue donc une division de la fréquence par 2. Nous avons pris l'exemple d'une bascule T, cet exemple n'est pas limitatif, en plus nous savons que la bascule T est dans la pratique réalisée à partir d'autres bascules. La Figure 9.12 rappelle deux montages usuels de bascule T réalisés à l'aide d'une bascule JK ou D.

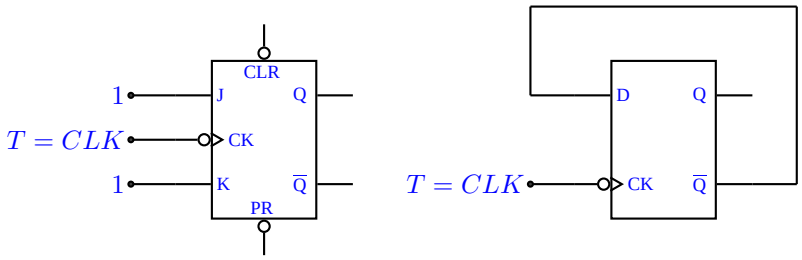

Figure 9.12: *Montages usuels des bascules* T.

Compteurs Asynchrones

Ces compteurs sont très simples. Ils sont très nombreux sous forme de circuits intégrés. Le terme asynchrone est justifié par le fait que les bascules du compteur ne sont pas commandées par les impulsions d'une seule horloge. La simplicité du compteur est contrebalancée par la limitation de sa vitesse de fonctionnement.

Un compteur asynchrone est constitué de plusieurs bascules montées en cascade (Figure 9.13). Chaque bascule constitue un étage du compteur. Le premier étage reçoit le signal de comptage H, le deuxième étage reçoit le signal de sortie du précédent. D'une manière générale, le signal de comptage d'une bascule de rang n n'est autre que le signal de sortie de la bascule de rang $n-1$. La sortie

de chaque étage représente un bit du compteur.

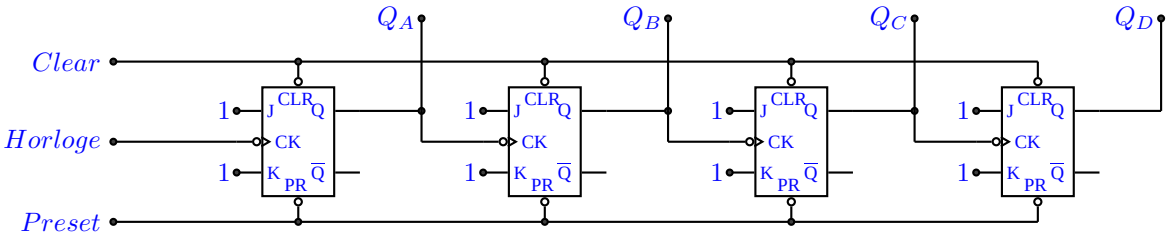

Figure 9.13: *Compteur binaire asynchrone de 4 (quatre) bits.*

La Figure 9.14 donne un exemple d'un compteur binaire asynchrone à 4 (quatre) bits. Initialement les quatre bascules sont remises à zéro $Q_A = Q_B = Q_C = Q_D = 0$. La première impulsion de comptage appliquée à l'entrée H de la bascule A fait basculer Q_A de 0 à 1. Nous supposons que les bascules sont commandées par front descendant. La bascule B change d'état lorsque Q_A repasse à 0, c'est-à-dire au deuxième front descendant du signal de comptage et ainsi de suite. Le diagramme temporel (Figure 9.13) montre que, avant l'arrivée de la 16^{eme} impulsion, toutes les bascules se trouvent à 1. C'est cette impulsion qui doit les remettre à 0. Le compteur divise la fréquence du signal d'entrée par 16, c'est un compteur diviseur par 16. Ce résultat peut être généralisé de la manière suivante: lorsque nous avons n bascules T en cascade, le compteur est diviseur par N où N est défini par $N = 2^n$.

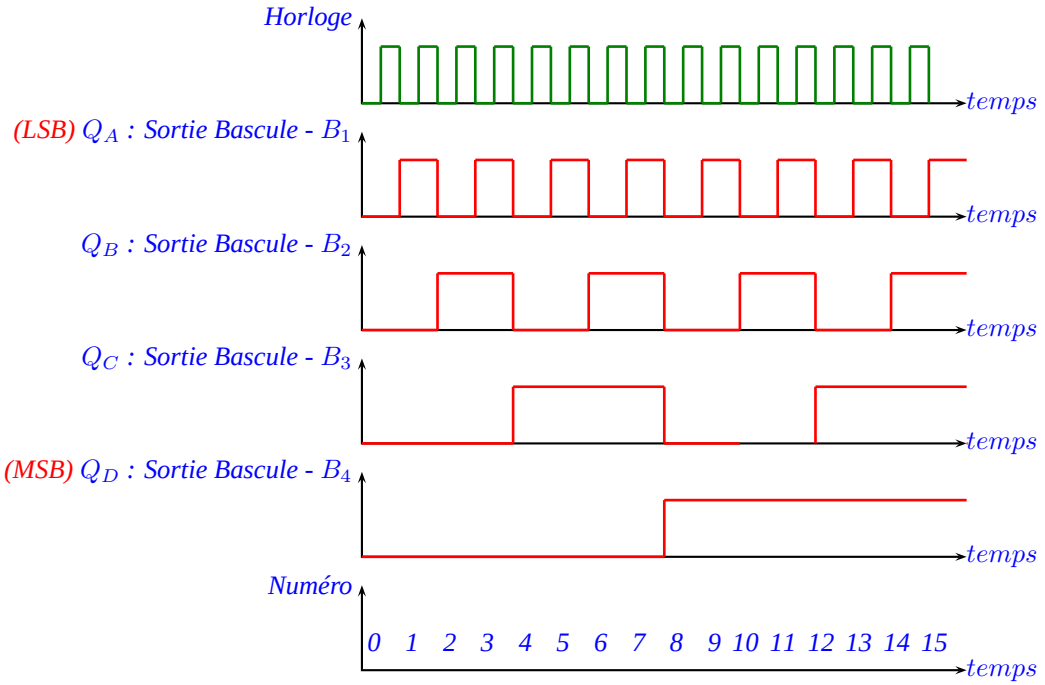

Figure 9.14: *Diagramme de temps du compteur binaire asynchrone de 4 (quatre) bits.*

La table de vérité (Figure9.15) indique l'état des quatre bits correspondant à l'impulsion d'entrée.

État	Q_D	Q_C	Q_B	Q_A
00	0	0	0	0
01	0	0	0	1
02	0	0	1	0
03	0	0	1	1
04	0	1	0	0
05	0	1	0	1
06	0	1	1	0
07	0	1	1	1
08	1	0	0	0
09	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

Figure 9.15: *Table de vérité du compteur binaire asynchrone de 4 (quatre) bits.*

$\Rightarrow$ 1. Compteur Asynchrone Diviseur par un Nombre Entier N

Le cas difficile à réaliser est évidemment celui où : $2^{n-1} \leq N \leq 2^n$. C'est aussi le cas qui nous intéresse ici. Nous pouvons alors suivre la méthode simple suivante :

1. Détermination du nombre de bascules: lorsque N n'est pas une puissance de 2, nous devons utiliser la puissance de 2 immédiatement supérieure.

2. Nous effectuons les liaisons classiques du compteur asynchrone, comme nous l'avons fait pour le compteur Figure 9.13, c'est-à-dire que nous mettons les bascules en cascade.

3. Détermination du nombre binaire de $N - 1$.

4. Les sorties des bascules correspondant aux bits 1 du nombre binaire $N - 1$ sont envoyées à l'entrée d'une porte *NAND* qui reçoit également le signal de comptage.

5. La sortie de la porte *NAND* est ensuite appliquée à l'entrée directe *PR (PRESET)* des bascules dont les sorties correspondent aux bits 0 du nombre binaire de $N - 1$. La remise à zéro du compteur est effectuée en deux étapes: - au front montant de la énième impulsion les bascules sont remises à 1, ensuite au front descendant de cette même impulsion le compteur revient à zéro, et le nouveau cycle de comptage peut recommencer.

$\Rightarrow$ 1. Comptage des Signaux Non Périodiques

Le compteur asynchrone se prête bien au comptage des signaux non périodiques qui sont constitués d'impulsions apparaissant à des intervalles de temps irréguliers. Ces impulsions sont généralement mises en forme par une bascule de *Schmitt* avant d'être appliquées au compteur ; c'est par exemple, le cas des impulsions issues d'un détecteur nucléaire; pour connaître le nombre d'impulsions pendant un temps déterminé, nous remettons toutes les bascules à zéro par l'entrée directe *Clear*, puis nous appliquons la sortie du détecteur à l'entrée du compteur pendant le temps désiré. A la fin de cet intervalle de temps l'état de sortie des bascules affiche le nombre d'impulsions comptées.

$\Rightarrow$ 3. Décodage

Dans de nombreuses applications pratiques, nous devons *lire* le contenu du compteur durant son cycle de comptage pour générer un signal de commande lorsque le comptage atteint un nombre fixé d'avance. On doit utiliser ici un décodeur.

$\Rightarrow$ 4. Caractéristiques Dynamiques des Compteurs Asynchrones

Les caractéristiques dynamiques des compteurs sont étroitement liées à celles de leurs bascules constitutives qui sont f_{max}, f_{min}, t_{PLH} (ou t_{PHL}), t_{setup} et t_{hold}.

$\Rightarrow$ 4.1. Vitesse Maximale de Comptage

Dans un compteur, si les bascules utilisées sont d'un même type et si la lecture du contenu s'effectue à la fin de chaque comptage, la vitesse de fonctionnement du comptage dépend surtout de la première bascule (qui est la bascule donnant le bit LSB). Il est donc nécessaire de vérifier la compatibilité des caractéristiques dynamiques f_max, et les divers temps de la bascule avec la forme et la fréquence du signal de comptage. Aussi faut-il noter que la condition *fréquence du signal* $\leq f_{max}$ est nécessaire mais souvent insuffisante. La forme du signal (symétrique ou asymétrique) peut jouer un rôle important.

Le problème peut devenir délicat lorsque nous devons travailler à des fréquences limites avec un compteur constitué de bascules de types différents. Dans ce cas, il est souvent utile de déterminer le diagramme temporel à l'aide des paramètres de chaque bascule et vérifier que les contraintes temporelles sont satisfaisantes.

Par exemple, supposons que la bascule LSB d'un compteur ait les caractéristiques suivantes : f_{max} = 50 MHz, $t_{PLH=15}$ ns. C'est une bascule D à commande par front montant et il ne faut pas tenir compte de t_{min}, t_{PLH} (ou t_{PHL}), t_{setup} et t_{hold}.

Sur le diagramme temporel nous constatons que le front montant de Q apparaît 15 ns après celui du signal d'entrée. Nous aurions pu croire que c'est le front descendant du signal d'entrée qui est actif. Le fonctionnement de la bascule est correct. Cependant, lorsque nous devons travailler à la limite des possibilités de la bascule (c'est le cas du problème), il est toujours utile de vérifier que les autres conditions temporelles sont satisfaites.

Nous devons vérifier que :

1. La largeur minimum du signal H : $t_{Low-min} \leq 20/2$ ns ;

2. Le temps d'établissement : $t_{setup} \leq 5$ ns.

Nous remarquons que la condition sur le temps de maintien t_{hold} est ici facilement satisfaite car le changement de l'entrée de commande D a lieu 15 ns après le front actif du signal de comptage.

$\Rightarrow 4.2.$ États Parasites (Glitches)

A la limitation de vitesse de fonctionnement exposée ci-dessus, s'ajoute celle des états parasites qui existent dans les compteurs asynchrones. Toutes les bascules ne changeant pas d'état simultanément, au moment du changement le compteur présente inévitablement des états parasites. Cet inconvénient est surtout important lorsque la lecture doit s'effectuer au cours du comptage et que le contenu du compteur lu est ensuite décodé : - c'est au niveau de l'entrée du décodeur que nous voyons apparaître des impulsions parasites *(glitches)* gênantes.

Le remède utilisé couramment pour corriger les erreurs dues à ces impulsions consiste à renvoyer les sorties décodées à l'entrée par l'intermédiaire d'une nouvelle porte ET.

$\Rightarrow 5.$ Compteurs Synchrones

Toutes les bascules reçoivent en même temps le même signal d'horloge H. L'inconvénient des impulsions parasites des compteurs asynchrones est supprimé car les états des bascules changent simultanément. Certes les compteurs synchrones sont plus complexes que les compteurs asynchrones, cependant leurs performances et leur souplesse d'emploi les rendent plus populaires que les autres ; en plus leur complexité deviennent de moins en moins significative avec l'évolution de la technologie intégrée.

9.4.2 *Registres à Décalage*

Un registre à décalage est constitué de bascules interconnectées de façon à ce que l'état de la bascule j soit transmis à la suivante à chaque coup d'horloge appliqué simultanément à toutes les bascules. L'information peut être présentée à l'entrée de la bascule pour être véhiculée à travers le registre. Nous disons que nous avons une entrée en série. Cette information ressort par la dernière bascule, c'est donc une sortie en série. Les sorties parallèle Q_A, Q_B, Q_C sont possibles, cependant dans la pratique nous ne pouvons pas avoir accès à ces sorties à cause de la limitation du nombre d'électrodes du circuit intégré.

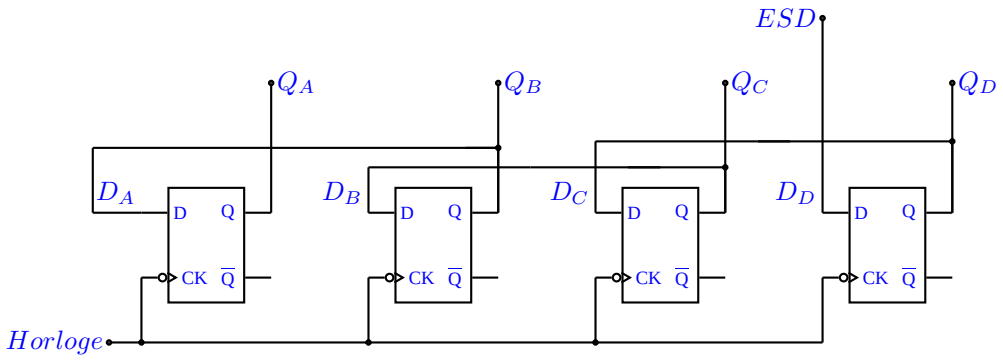

Figure 9.16: *Décalage de droite à gauche.*

Le décalage peut s'effectuer dans le sens inverse c'est à dire de droite à gauche Figure 9.16. Il peut être bidirectionnel, dans ce cas la direction sera commandée par un signal d'entrée. L'information peut être introduite d'un seul coup dans le registre (Figure 9.17) par le biais des entrées des bascules. Généralement un signal de validation permet le transfert ou l'isolement entre le registre et l'extérieur. Nous disons que nous avons une entrée parallèle. Le décalage s'effectue toujours en synchronisation avec l'horloge.

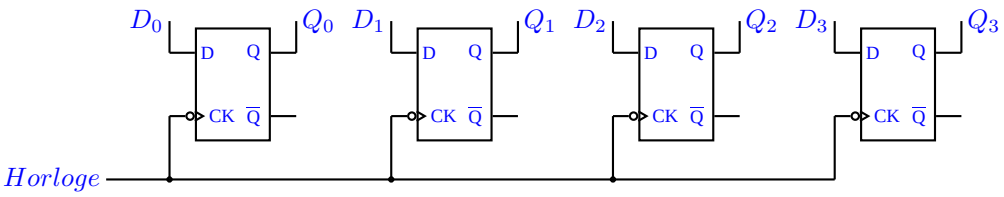

Figure 9.17: *Registre à entrées parallèles.*

En définitive un registre de décalage universel peut être schématisé par la Figure 9.18 où nous distinguons toutes les différentes entrées et sorties fournissant tous les modes de fonctionnement.

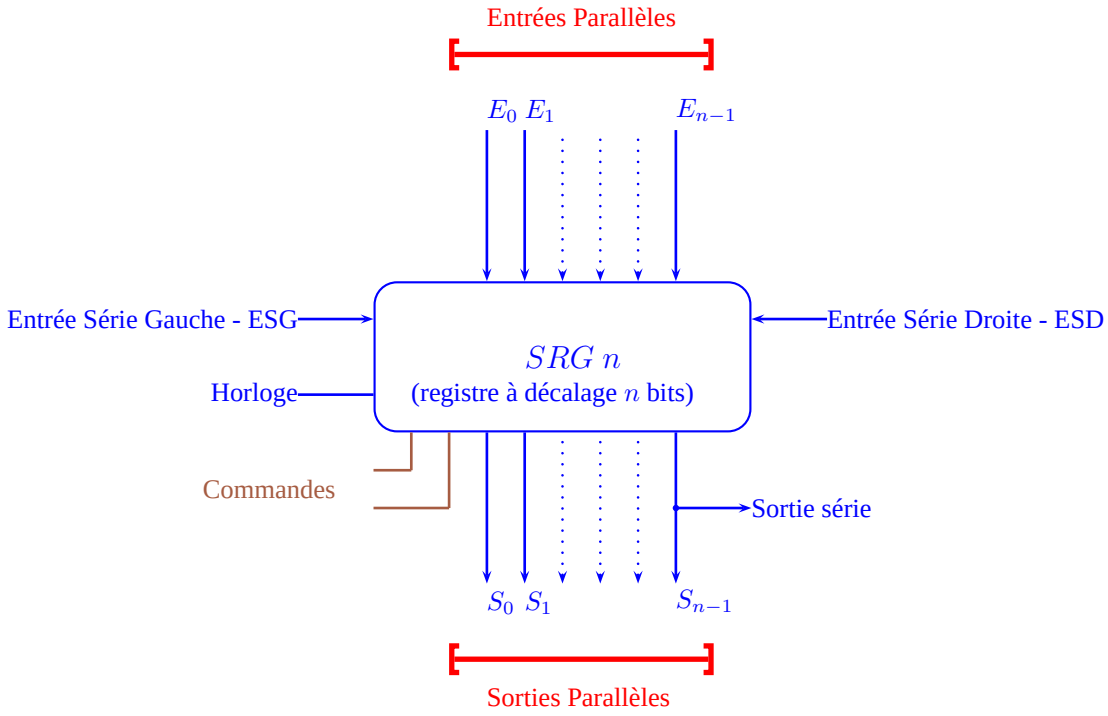

Figure 9.18: *Registre à décalage universel.*

9.4.3 *Étude Expérimentale*

Les schémas synoptiques de la Figure 9.19 utilisent un ensemble de composants de la famille *TTL*. Réaliser le projet du circuit, implémenter la montage et faire la simulation (*CircuitMark*).

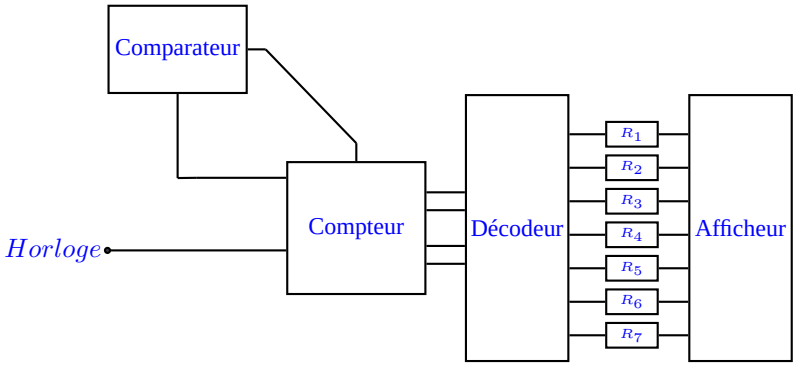

Figure 9.19: *Le circuit du montage compteur maximum égal à 17 en base décimale.*

Description du Problème

Projeter un compteur synchrone qui réalise le comptage d'un numéro maximum d'impulsion égal à 17 (dix sept). Pour le projet, utiliser les blocs fonctionnels SSI avec les entrées en parallèles. De plus, le circuit permettra de visualiser le contenu par un afficheur de sept segments.

Liste de Matériel

Pour développer le montage, nous avons besoin du matériel suivant :

- Oscilloscope;
- Alimentation stabilisée ;
- Multimètre;
- Générateur BF;
- Moniteurs *MS05* (plaquette de câblage);
- Câbles :
    - 2 (deux) sondes de l'oscilloscope ;
    - 6 (six) fils banane;
    - 1 (un) BNC-banane;
- Composants :
    - 2 (deux) SN74ALS161N -> 4 (un) *Compteur* à 2 entrées;
    - 1 (un) SN74ALS30N -> 1 (un) *NANDR* à 8 entrées;
    - 1 (un) SN74ALS04N -> 6 (six) *Inverseur*;
    - 2 (deux) SN74LS47N -> Decodeur 7 (sept) segments;
    - 2 (un) -> Plaquette d'afficheur 7 (sept) segments;
    - 8 (huit) résistances de *4K7*Ω et *1/4 Watt*.

9.5 *TP* $N°4$

9.5.1 *Projet*

Projeter un microprocesseur, comme l'illustre la Figure 9.20. Le Tableau 9.1 décrit les caractéristiques des registres et le Tableau 9.20 définit les instructions à réaliser.

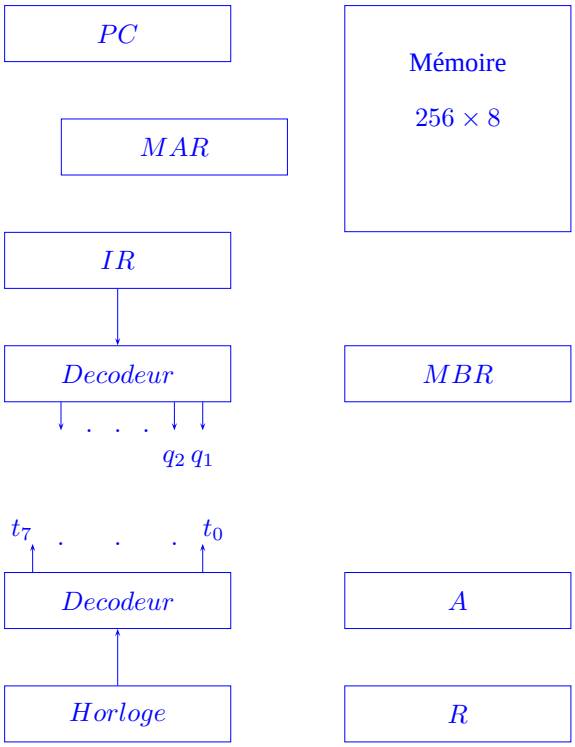

Figure 9.20: *Diagramme du microprocesseur.*

Symbole	Nombre de bits	Nom	Fonction
MAR	8	Registre d'Adresse de Mémoire	Maintenir l'Adressent de la Mémoire
MBR	8	Registre Tampom de Mémoire	Maintenir le Contenu de la Mémoire
A	8	Registre A	Registre du Processeur
R	8	Registre R	Registre du Processeur
PC	8	Compteur de Programme	Maintenir l'Adrresse de l'Instruction
IR	8	Registre Instruction	Maintenir le Code d' l'Opération
T	8	Compteur de Temps	Générateur de la Séquence

Table 9.1: *Les registres.*

Code Opération	Mnémonique	Description	Fonction
00000001	MOV R	Déplacez R vers A	$A \leftarrow R$
00000010	LDI OPRD	Chargez OPRD dans A	$A \leftarrow OPRD$
00000011	LDA ADRS	Chargez l'Opérande Spécifiée par ADRS dans A	$A \leftarrow M[ADRS]$

Table 9.2: *Les instructions.*

9.5.2 *Compte-rendu*

Pour faire le projet, utiliser les composants de la famille *TTL*. L'adresse électronique pour les composants est la suivante : http://www.ti.com.

Le mémoire, (en *Latex*), du projet devra contenir les éléments suivants :

1. La description de la phase du projet ;
2. L'automate et la description du fonctionnement ;
3. Les diagrammes de Karnaugh de minimisation ;
4. Le projet au niveau des composants ;
5. La description complète du fonctionnement du projet ;
6. La date pour rendre le projet sera le jour du dernier TP.

Partie IV

I*nstrumentation*

Laboratoire

Chapitre 10

Appareil du Laboratoire

10.1 *Digital Multimeter*

10.1.1 *DC Voltage Measurement*

1. Connect red test lead to $V - \Omega$ input connector and black test lead to COM input connector.

2. Set Function/Range switch to desired DCV position. If magnitude of voltage is not known, set switch to the highest range and reduce until a satisfactory reading is obtained.

3. Turn off power to the device or circuit being tested and discharge all capacitors.

4. Connect test leads to the device or circuit being measured.

5. Turn on power to the device or circuit being measured. Voltage value will appear on the digital display along with the voltage polarity.

6. Turn off power to the device or circuit being tested and discharge all capacitors prior to disconnecting test leads.

10.1.2 *AC Voltage Measurement*

1. Connect red test lead to $V - \Omega$ input connector and black test lead to COM input connector.

2. Set Function/Range Switch to desired ACV position. If magnitude of voltage is not known, set switch to highest range and reduce until a satisfactory reading is obtained.

3. Turn off power to the device or circuit being tested and discharge all capacitors.

4. Connect the test leads to device or circuit being measured.

5. Turn on power to the device or circuit being measured. Voltage value will appear on digital display.

6. Turn off power to the device or circuit being tested and discharge all capacitors prior to disconnecting test leads.

10.1.3 *DC Current Measurement*

1. Connect red test lead to the mA input connector for current measurements up to 200 milliamperes. Connect black lead to the COM input connector.

2. Set Function/Range Switch to desired DCA position. If magnitude of current is not known, set switch to highest range and reduce until satisfactory reading is obtained.

3. Turn off power to the device or circuit being tested and discharge all capacitors.

4. Open the circuit in which current is to be measured. Now securely connect test leads in series with the load in which current is to be measured.

5. Turn on power to the circuit being tested.

6. Read current value on digital display.

7. Turn off all power to the circuit being tested and discharge all capacitors.

8. Disconnect test leads from circuit and reconnect circuit that was being tested.

10.1.4 *Resistance Measurement*

All resistance ranges on the multimeter are low-power ohms except for the 200-ohm range. The low power ohm allows accurate measurements of in-circuit resistance, as test voltage is below that necessary to turn on a diode junction Note: In the 200MΩ range, the continuity beeper function is activated (Model 375B only).

1. Connect red test lead to the $V-\Omega$ input connector and black test lead to the COM input connector.

2. Set Function/Range Switch to desired Ω position. If magnitude of resistance is not known, set switch to highest range and reduce until satisfactory reading is obtained.

3. If the resistance being measured is connected to a circuit, turn off power to the circuit being tested and discharge all capacitors.

4. Connect test leads to the circuit being measured. When measuring high resistance, be sure not to contact adjacent points even if insulated, because some insulators have a relatively low insulation resistance causing the measured resistance to be lower than the actual resistance.

5. Read resistance value on digital display. If a high resistance value is shunted by a large value of capacitance. allow digital to stabilize.

10.1.5 *Diode and Transistor Tester Measurements*

The special *Diode Test Function* allows relative measurements of forward voltage drops across diodes and transistor junctions. This function also permits measurement of in-circuit semiconductor junctions.

Diode Tests

1. Connect red test lead to the $V-\Omega$ input connector and black test lead to the COM input connector.

2. Set Function/Range Switch to the diode test position.

3. If the semiconductor junction being measured is connected to a circuit, turn off power to circuit being tested and discharge all capacitors.

4. Connect test leads to the device.

5. Read forward value on digital display.

6. If the digital display reads overrange (1), reverse the lead connections. The placement of the test leads when the forward reading is displayed indicates the orientation of the diode. The reed lead is positive and the black lead is negative. If overrange (1) is displayed with both lead connections, the junction is open. If a low-reading (less than 1,000) is obtained with both lead connections, the junction is shorted internally or (if junction is measured in a circuit) the junction is shunted by a resistance less than $1K\pi$. In the latter case the junction must be disconnected from the circuit in order to verify its operation.

Transistor Junction Tests

1. Bipolar transistors can be tested in the same manner as diode junctions formed between the base and emitter and the base and collector of the transistor. Measurement between the collector and emitter also should be made to determine if a short is present.

BEEPER NOTE: In the Continuity Positions the meter will beep when the resistance is below approximately 80 ohms.

10.2 *Alimentation Maître-esclave*

La tension négative est asservie à la tension Positive, même en cas de variation rapide, transitoire ou bruit de fond, et également en cas de variation lente due à la dérives thermique. En position $Tracking$ les deux sources sont asservies. En position $Independent$ elles sont totalement indépendantes et isolées galvaniquement.

Chaque alimentation est régulée en courant et en tension, le passage d'un mode à l'autre se faisant automatiquement en cas de surcharge. Pour chaque alimentation il est prévu un réglage par potentiomètre de tension et de courant limite.

10.2.1 *Position Tracking*

Cette position asservir l'alimentation *esclave* (à gauche) à l'alimentation *maitre* (à droite). Cet asservissement peut être effectué selon deux modes :

$\Rightarrow$ 1. Mode Série

Les deux alimentations sont connectées en série, le + de l'esclave étant raccordé intérieurement au - du maître. L'ensemble se comporte comme une alimentation symétrique par rapport à ce point commun. Ce dernier peut être laissé flottant ou peut être raccordé à la masse du chassis, à l'aide du cavalier. Dans ce cas serrer le cavalier entre le commun et une borne verte de masse.

$\Rightarrow$ 1. Mode Parallèle

Les deux alimentations sont connectées en parallèle c'est à dire que les deux $+$ (respect les deux $-$) sont reliés ensemble. Nous disposons alors d'une source 30 Volt dont le débit est égal à la somme des deux débits fixés par les potentiomètres de limitation en courant.

Figure 10.1: *L'alimentation maître-esclave.*

10.3 *Générateur de Signaux*

Figure 10.2: *Le générateur de signaux avec balayage des fréquences (sweep) et fréquencemètre.*

Figure 10.3: *La face avant.*

Figure 10.4: *La face arrière.*

10.3.1 *Modes de Fonctionnement*

Repères 30 et 31 Mode $\longleftrightarrow$

Ces deux touches fugitives (action par une simple pression) permettent de changer de mode de fonctionnement (mode générateur avec ou sans balayage de fréquence, ou mode fréquencemètre LF ou HF).

10.3.2 *Générateur de Fonction*

Repère 1 Frequency

Ensemble de sept touches poussoirs qui permettent 10 choix de la gamme de fréquence du signal généré (gammes se, recouvrant à 20%). L'enfoncement d'une touche (qui devient active) relâche la touche précédemment enfoncée. La valeur indiquée au-dessus des touches correspond à la valeur du milieu de gamme de la fréquence.

Repère 2 INV

Touche poussoir active si enfoncée; elle inverse le signal généré, donc le sens des impulsions définies par la commande rotative $symmetry$ (rapport cyclique, repère 9). Cette touche active n'est pas Indiquée sur le LCD.

Repère 3 Sinusoïdal, Rectangulaire, Carré et DC

Forme du signal généré sur output (repère 5). La touche active est enfoncée et la forme choisie est indiquée sur le LCD. Le signal rectangulaire, signal triangulaire, signal sinusoïdal et DC quand les trois touches sont relâché. une tension continue de décalage est générée. Cette tension est réglable par la commande rotative $DC\ OFFSET$ (repère 8).

Repère 4 20dB/ATT

La touche poussoir qui met en service (touche enfoncée) l'atténuateur fixe de $-20dB$ à la sortie du générateur (à utiliser pour des signaux d'amplitude $> 500\ mV$ valeur continu). Cette touche active est indiquée sur le LCD ($-20dB$).

Repère 5 Output

La sortie par prise BNC du signal généré et défini par les commandes précédentes (repère 1 à 4) et les commandes suivantes (repère 7 à 10). Quand nous passons en mode fréquencemètre, la signal généré antérieurement est maintenu sur la sortie (sans le balayage des fréquences, s'il était sélectionné).

Repère 6 Output TTL

La sortie par prise BNC d'impulsion de niveau logique TTL (pour synchronisation par exemple). La période de ces impulsions est celle du signal généré sur la sortie out (repère 5), en phase, et de rapport cyclique défini par la commande rotative $symmetry$ (repére 9).

Repère 7 Level

Le commande rotative qui permet de régler de façon continue, l'amplitude du signal de sortie généré, de 0 à 20 Volt crête à crête en circuit ouvert (0 à 10 Volts valeur continu sur 50 ohm).

Repère 8 DC Offset

Le commande rotative et poussée-tirée de décalage. En position poussée, cette commande est inopérante, le signal de sortit généré est centré sur 0 V. En position tirée (indication $OFFSET$ sur le LCD), elle permet de superposer une tension continue ajustable (fonction rotative) au signal de sortie généré. La plage de décalage maximale est de ± 10 Volt en circuit ouvert (± 5 Volt sur 50 ohm). La somme tension de décalage plus tension créte du signal généré sans décalage) ne doit jamais dépasser ± 20 Volt; le cas échéant, réajuster le niveau du décalage ou l'amplitude du signal ($Level$, repère 7).

Repère 9 Symmetry (Duty)

Le commande rotative et poussée-tirée de rapport cyclique (symétrie dans le temps). En position poussée, cette commande est inopérante, le rapport cyclique du signal de sortie généré est de 50%. En position tirée, la commande est active. La fréquence du signal de sortie généré et F_{MAX} en balayage, sont divisées par 10. L'indication $Duty$ s'affiche sur le LCD et le type d'affichage est celui de la gamme immédiatement inférieure. Cette commande permet d'ajuster (fonction rotative) le rapport cyclique du signal généré sur la sortie output (repère 5) entre 20% et 80%. Il est possible, ainsi, d'obtenir des signaux triangulaires dissymétriques et des sinusoïdes avec distorsion.

Repère 10 Réglage de la Fréquence

Le commande rotative qui permet de régler la fréquence du signal de sortie généré dans un rapport de $0,2$ à 2 (recouvrement des gammes: 20% environ). L'indication du cadran est le facteur multiplicateur de la gamme de fréquence précédemment choisie (Frequency, repère 1). Pour la gamme 1 k et l'index sur la graduation $1,2$ donne une fréquence de signal de sortie de 1,2 kHz.

Repère 20 Écran LCD

Pour affichage des symboles et de la mesure.

10.3.3 *Générateur de Balayage*

Repère 40 Sweep

Le commande par touche fugitive (action par une simple pression) qui met en service ($sweep$ affiché sur la LCD) ou hors service le mode balayage de fréquence. La fréquence du signal généré en sortie, est modulée selon une loi linéaire (LIN) ou logarithmique (LOG), le temps d'une dent de scie en interne (INT), selon la commande de fréquence par tension (entrée VCF, sur face arrière, repère 102) en externe (EXT). Pour l'utilisation de cette touche, voir repère 42.

La plage de fréquence balayée est : - pour le début du balayage (départ de la campe) la fréquence du signal réglée en mode générateur de fonctions (gamme avec repère 1 et valeur avec repère 10), et pour la fin du balayage (fin de rampe) la fréquence réglée par la commande F_{MAX} (repère 47) dans la gamme considérée.

La valeur de F_{MAX} doit toujours être inférieure à la fréquence maximale de la plage balayée (selon la gamme). Si la commande $symmetry/duty$ est active (repère 9 en position tirée, $duty$ affiché sur le LCD), la fréquence balayée est divisé par 10. La rampe ou dent de scie linéaire servant au balayage de fréquence, est disponible sur la face arrière, sur la borne BNC $sweep$ repère 103.

Repère 41 LIN/LOG

Le commande par touche fugitive (action par une pression selon l'état antérieur) qui permet de choisir la loi de balayage de fréquences (en interne INT uniquement) : - loi linéaire (LIN affiché sur le LCD) et la logarithmique (LOG affiché sur le LCD).

Repère 42 INT/EXT

Le commande par touche fugitive (action par une pression selon l'état antérieur) qui permet de choisir le balayage de fréquence par commande interne INT) ou par commande externe (EXT). En commande interne INT affiché sur le LCD), le balayage de fréquence peut être linéaire (LIN) ou logarithmique (LOG). En commande externe (EXT affiché sur le LCD), la tension variable appliquée à l'entrée VCF (borne BNC située sur la face arrière (repère 102) fixe la fréquence balayée dans la gamme choisie à l'aide d'une des touches poussoirs (repère 1). Voir le tableau des plages de fréquence balayée repère 40, $sweep$. La plage de tension appliquée sur l'entrée VCF est de : 0 Volt/$+10$ Volt pour le rapport de fréquence $100/1$ (avec réglage de la fréquence au maximum; repère 10) et 0 Volt/-10 Volt pour le rapport de fréquence $1/100$ (avec réglage de la fréquence au minimum, repère 10).

Repère 43 $STOP/START$

Le commande par touche fugitive (action par simple pression) qui permet d'agir sur l'évolution de la rampe de balayage de fréquence pour la lancer, l'arrêter ou la relancer. Si l'action de la touche arrête la rampe, $STOP$ est affiché sur la LCD.

Repère 44 $SGL/MONO$

Le commande par touche fugitive (action par simple pression) qui permet d'obtenir un seul balayage de fréquence ($single$ affiché sur le LCD quand la touche est active), au lieu d'un balayage de fréquence répété.

Repère 45 F_{MAX}

Le commande par touche fugitive (action par une simple pression) qui agit sur l'évolution de la rampe du balayage (Max affiché sur le LCD et permet l'arrêt de la rampe à F_{MAX} avec l'affichage de sa valeur.

Repère 46 SW FREQ

Le commande rotative qui permet de régler la durée du balayage de la fréquence (durée de la rampe ou'dent de scie) entre $10\ ms$ et 10 s.

Repère 47 F_{MAX}

Le commande rotative qui permet de régler la fréquence maximale balayée (valeur pour le sommet de la rampe ou dent de scie du balayage) dans la gamme choisie ($frequency$, repère 1). Si la commande $symmetry$ est active (position tirée, repère 9), F_{MAX} est divisée par 10. La valeur de F_{MAX} doit toujours étre inférieur ou égale à la fréquence maximale de la plage balayée.

Repère 102 VCF Input (Face Arrière)

L'entrée externe de commande de balayage de fréquences.

Repère 103 Sweep Out (Face Arrière)

La sortie de la dent de scie linéaire qui commande le balayage de fréquences.

Repère 40 Filter

La commande par touche fugitive (action par une simple pression). Une pression sur cette touche met en service un filtre passe-bas. Ce filtre peut être utilisé dans le cas d'une mesure d'un signal bruité. Quand le filtre est actif le symbole LF de l'afficheur LCD clignoté. La fréquence de coupure de ce filtre passe-bas, est de 1 kHz. Une autre pression sur la touche met hors service le filtre; le clignotement du symbole LF s'arrête.

Repère 50 FREQ Input

L'entrée BNC pour mesurer la fréquence d'un signal extérieur (5 chiffres sur le LCD), selon deux modes (choix par les touches MODE $\leftarrow\rightarrow$ repère 30 et 31).

L'$External\ Counter\ LF$ (affiché sur le LCD) de 5 Hz à 5 MHz, l'affichage de la mesure est du même type qu'on mode générateur de fonctions. Les changements de gammes sont automatiques :

Gamme de Mesure	*Affichage milieu de gamme*	*Unité*
$<$ 5 Hz	*0.000 au affichage erroné*	*Hz*
5 Hz - 25 Hz	*10.000*	*Hz*
16 Hz - 250 Hz	*100.00*	*Hz*
0.16 kHz - 2.5 kHz	*1.0000*	*kHz*
1.6 kHz - 25 kHz	*10.000*	*kHz*
16 kHz - 250 kHz	*100.00*	*kHz*
0.16 MHz - 5 MHz	*1.0000*	*MHz*
$>$ 5 MHz (*)	*C_HF ou affichage erroné*	*MHz*

Table 10.1: *Les gammes d'opérations du conteur LF.*

(*) Passer en $Extemal\ Counter\ HF$ avec les touches $MODE$ (repères 30 et 31). En mode LF, si la fréquence à mesurer est supérieur à 5 MHz, une mesure erronée peut s'afficher à la place de C_{HF}. Une mesure en mode HF est alors conseillés.

L' $External\ Counter\ HF$ (affiché sur le LCD) de 4 MHz à 120 MHz; l'affichage de la mesure est du même type qu'on mode générateur de fonctions. Les changements de gammes sont automatiques :

Gamme de Mesure	*Affichage milieu de gamme*	*Unité*
$<$ 1 MHz (*)	*C_HF*	*MHz*
1 MHz - 4 MHz (*)	*2.500*	*MHz*
4 MHz - 25 MHz	*10.000*	*MHz*
16 MHz - 120 MHz	*100.00*	*MHz*
$>$ 120 MHz (**)	*O.F. ou affichage erroné*	*MHz*

Table 10.2: *Les gammes d'opérations du conteur HF.*

(*) Passer en $External\ Counter\ LF$ avec les touches $MODE$ repère 30 et 31).

(**) Si la fréquence à mesurer est supérieur à 120 MHz, une mesure erronée peut s'afficher à la place de $O.F.$, afin d'éviter cette erreur, l'utilisation d'un pré-diviseur est conseillée.

10.4 *Oscilloscope*

10.4.1 *Principales Caractéristiques Techniques PM3335/PM3337*

Figure 10.5: *L'Oscilloscope.*

⇒ Oscilloscope analogique 60 MHz ;
⇒ Acquisition numérique coup par coup de 20 Méchantillon/s;
⇒ Mémoire d'acquisition de grande capacité (8 K) ;
⇒ Mode réglages automatiques pour observation instantanée d'un signal sur les deux modes, analogique et numérique. de l'oscilloscope ;
⇒ Curseurs permettant la mesure des amplitudes et des temps en mode oscilloscope numérique;
⇒ Mémoires de référence et stockage non volatile de toutes les traces;
⇒ Interfaces en option pour télécommande et sortie copie permanente des traces.

10.4.2 *Instructions d'Utilisation*

Mise en Service

⇒ Brancher l'oscilloscope à la tension secteur.

⇒ Mettre l'oscilloscope sous tension au moyen de l'interrupteur général à touches *POWER LINE ON*.

⇒ Avec un tournevis régler la rotation de la trace (*TRACE ROTATION*) de manière à ce qu'elle soit parallèle à une ligne horizontale de graticule.

⇒ Brancher une sonde sur l'entrée A de l'oscilloscope.

⇒ Brancher l'autre extrémité de la sonde sur la sortie CAL de l'oscilloscope.

⇒ Appuyer sur la touche $AUTO\ SET$.

La fonction *AUTO SET* (réglages automatiques) place automatiquement tous les paramètres de l'oscilloscope correspondant (réglages d'atténuation, de base de temps et de source de déclenchement) de manière à pouvoir observer la meilleure trace.

Si la sonde utilisée est une sonde au dixième ou au centième, il peut être obligatoire d'ajuster sa compensation, suivant les instructions de la sonde.

Connecteurs

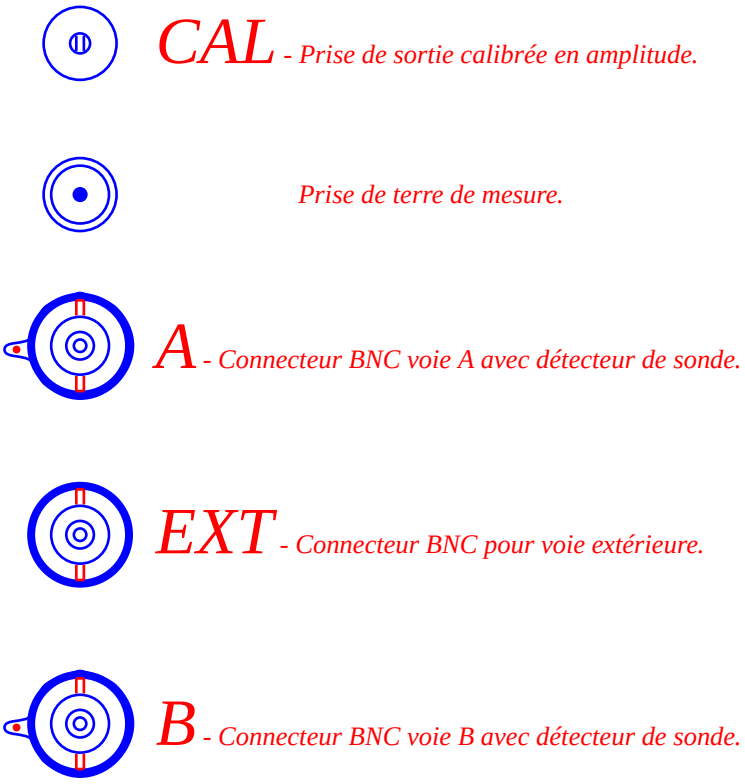

Figure 10.6: *Les connecteurs.*

10.4.3 *Disposition des Commandes et Manipulations*

Disposition des Commandes

La face avant a été conçue pour une configuration ergonomique et logique optimale des commandes. De gauche à droite et de haut en bas comme pour lire un livre. L'accès aux commandes de l'oscilloscope est facilité par une division en sept zones principales.

$\Rightarrow$ Les commandes écran.

$\Rightarrow$ La zone d'affichage (écran cathodique plus affichage à cristaux liquides).

$\Rightarrow$ Les touches haut-bas ($up - down$).

$\Rightarrow$ Les touches de fonction.

$\Rightarrow$ Les boutons rotatifs.

$\Rightarrow$ Les touches programmables.

$\Rightarrow$ Les entrées et les sorties.

Disposition Écran

Après avoir mis l'appareil sous tension avec la touche $POWER\ LINE\ ON$, ajuster les Commandes de l'écran pour avoir une luminosité, une trace et une netteté optimales.

En tournant le bouton $INTENS$ nous ajustons l'intensité de la trace et du texte, en tournant le bouton $FOCUS$ nous obtenons une image nette.

La commande $TRACE\ ROTATION$ est utilisée avec un tournevis pour régler une ligne du zéro parallèle à une ligne horizontale du graticule.

Le graticule de l'écran peut être allumé par le bouton $ILLUMINATION$, pour un usage en milieu sombre ou pour réaliser des photos.

Zone d'Affichage

La zone d'affichage est composée de l'écran du tube cathodique (CRT) et de l'affichage à cristaux liquides (LCD). L'écran du tube cathodique est divisé en 10 x 8 carrés. Chaque carré contient 5 sub-divisions. Sur la gauche de l'écran cathodique 4 lignes sont inscrites avec les valeurs 100%, 90%, 10% et 0%. Ces lignes peuvent être utilisées pour des mesures de pourcentage ou de temps de montée.

L'affichage à cristaux liquides indique les fonctions qui ont été sélectionnées et activées, les réglages et les paramètres. Sa disposition est divisée en zones logiques selon les touches et les boutons.

Un segment qui clignote indique :

$\Rightarrow$ Un mauvais choix de bouton, ou de touche.

$\Rightarrow$ Un bouton rotatif VAR en position non calibrée.

$\Rightarrow$ L'extrémité d'une gamme commandée par $HAUT - BAS$.

$\Rightarrow$ Une touche sollicitée au cours d'une action tracé de courbe ($PLOT$).

$\Rightarrow$ La touche $DISPL\ PART\ HAUT - BAS$ a été sollicitée dans le mode $TB\ MAGN\ *1$.

Touches UP-DOWN

Les touches $UP - DOWN$ permettent de sélectionner les réglages désirés dans toute la gamme des réglages possibles de cette fonction. Elles sont utilisées pour les réglages d'atténuation, de base de temps, de retard au déclenchement et de réglage d'affichage.

Gammes des Touches UP-DOWN

$\Rightarrow$ Atténuation Verticale , Voie A

Permet de sélectionner l'atténuation verticale sur la A.

En analogique : 2 mV/div à 10 V/div en séquence 1-2-5.

En numérique : 2 mV/div à 10 V/div en séquence 1-2-5.

$\Rightarrow$ Atténuation Verticale , Voie B

Permet de sélectionner l'atténuation verticale sur la B.

En analogique : 2 mV/div à 10 V/div en séquence 1-2-5.

En numérique : 2 mV/div à 10 V/div en séquence 1-2-5.

$\Rightarrow$ Base de Temps Horizontale

Permet de sélectionner la base de temps (vitesse de balayage).

En analogique : 0.5 s/div à 50 ns/div en séquence 1-2-5.

En numérique : 50 s/div à 50 μs/div en séquence 1-2-5.

$\Rightarrow$ Retard au Déclenchement

Permet de sélectionner le retard du déclenchement.

En analogique : non prévu.

En numérique : -20 div à 0 div par fraction de 1 division.

⇒ Partie Affichage Cathodique

Permet de sélectionner la partie d'une trace agrandie à observer.

En analogique: non prévu.

En numérique: par fraction de $1/16$ à $16/16$ en fonction du rapport de réglage de $TB\ MAGNIFY$.

10.4.4 *Utilisation en Oscilloscope Analogique*

Si nous le désirons utiliser l'appareil en oscilloscope analogique, il est nécessaire de définir trois fonctions principales:

⇒ Mode vertical.

⇒ Mode horizontal.

⇒ Mode de déclenchement.

S'assurer d'abord que l'oscilloscope est positionné pour le mode analogique (l'indication mémoire numérique de l'affichage à cristaux liquides est éteinte), sinon appuyer sur la touche $DIGITAL\ MEMORY$. En appuyant sur la touche $AUTO\ SET$, l'oscilloscope se configure automatiquement avec les meilleurs réglages des fonctions principales. Il est possible ensuite de procéder à d'autres réglages et sélections au moyen des différentes touches et boutons.

Mode Vertical

En mode vertical les réglages suivants sont possibles:

⇒ Sélection du mode d'affichage vertical par action sur la touche A/B.

⇒ Sélectionner la voie choisie A, B ou A et B simultanément. Si la fonction ADD est en service, il est possible de couper les voies A et B.

⇒ Réglage de la ligne de zéro absolu. Au moyen de la touche GND et du bouton $Y\ POS$ il est possible d'ajuster la ligne de base n'importe où sur l'écran.

En appuyant sur la touche GND la signal d'entrée est coupé et le circuit d'entrée du signal est mi à la masse. Le bouton $Y\ POS$ permet de placer la ligne de base à n'importe quelle hauteur sur l'écran. En appuyant sur la touche GND une seconde fois le signal d'entrée réapparaît sur l'écran. L'ajustement de la ligne de base est nécessaire pour toutes les mesures en courant continu.

Mode de Couplage des Voies par la Touche AC/DC

Dans la position AC de la touche, la composante continue du signal est éliminée. En position DC, le signal est disponible dans toute sa bande de fréquences. Le couplage en mode alternatif (AC) est utilisé lorsque nous désirons mesurer un petit signal alternatif lui-même superposé à une tension continue importante. Donc, sélection du mode d'affichage $ALTERN$ ou $HACH$ des voles à l'aide de la touche $ALT/CHOP$. Nous utilison cette sélection lorsque deux signaux sont présents à l'écran.

En mode alterné, le faisceau d'électrons du tube cathodique trace alternativement un signal pendant un balayage, puis l'autre signal pour l'autre balayage. A des vitesses de balayage faibles, ce phénomène peut s'observer. aussi utilisons nous principalement le mode ALT pour les bases de temps rapides (0, 1 ms et moins). En mode haché, le faisceau d'électrons du tube explore un signal puis l'autre à la fréquence de commutation rapide et convient par conséquent pour les signaux à faible vitesse de balayage.

Sélection du mode d'affichage somme (ADD), normal ou inverseur qui est exécuté par la touche $ADD/INVERT$. La fonction d'inversion concerne la voie B. En mode ADD, la voie B est ajoutée à la voie A ($A + B$). En mode $ADD\ INVERT$, la voie B est soustraite de la voie A ($A - B$). Le mode soustractif ($A - B$) est utile pour éliminer les signaux en mode commun. En mesurant suivant le mode soustractif le signal mode commun annule le signal mode commun de l'autre voie et apparaît sur l'écran le signal réel.

Mode Horizontal

Pour le mode horizontal. les sélections suivantes sont réalisables :

Sélection de l'échelle de temps horizontale parla touche $TB\ s...\mu s\ UP - DOWN$ en tournant le bouton VAR ou par la touche $X\ MAGN$. La touche $TB\ s...\mu s\ UP - DOWN$ permet de sélectionner la gamme de base de temps selon la séquence 1-2-5. En tournant le bouton VAR nous pouvons procéder au réglage fin entre les gammes 1-2-5. La touche $X\ MAGN$ permet d'amplifier la base de temps 10 fois. En tournant le bouton $X\ POS$ nous pouvons décaler la trace observée à l'écran dans la direction horizontale.

Sélection de la déviation sur l'axe des X ou sur la base de temps la touche $X\ DEFL$. En mode déviation X l'oscilloscope affiche un signal en fonction d'un autre signal (graphique $X - Y$). Le signal sur l'axe des X requis est sélectionné par la touche $TRIG\ OR\ X\ SOURCE$ (voie A, voie B, $Extrieure\ CA$ ou $Extrieure\ CC$). Le singal sur l'axe des Y est la voie A. Le mode $X - Y$ permet une grande variété d'applications, exemple :

$\Rightarrow$ Amplitude des circuits et des filtres par rapport à la fréquence;

$\Rightarrow$ Courant de sortie des semi-conducteurs par rapport à la tension d'entrée;

$\Rightarrow$ Comparaison de décalage de fréquences ou de déphasage par courbe de Lissajous.

En mode base de temps horizontal, la trace représente le signal mesuré par rapport au temps (graphe Y-t). L'unité de temps peut être sélectionnée par la touche TB s...,us UP-DOWN.

La fonction $LEVEL\ VIEW$ n'apparaît que si le couplage de déclenchement CC (DC) est actif.

Mode de Déclenchement de la Base de Temps

Le mode de déclenchement permet de sélectionner les fonctions du mode de déclenchement de la base de temps par la touche $TB\ TRIG\ MODE$. La base de temps fonctionne alors selon trois modes :

$\Rightarrow AUTO$

Le balayage horizontal est lancé normalement à l'apparition d'un déclenchement. Si aucun signal de déclenchement n'est détecté dans les 100 ms suivant le dernier balayage, un nouveau balayage est automatiquement lancé. Ceci produit une trace continuellement visible sur l'écran.

$\Rightarrow TRIG$

Le balayage horizontal ne se produit que sur une impulsion provenant de la source de déclenchement choisie. Le bouton rotatif $TRIG\ LEVEL$ doit être ajusté sur le niveau de déclenchement désiré.

$\Rightarrow SINGLE$

Le balayage horizontal ne s'effectue qu'une fois après avoir été armé par la touche $RESET$ et sur réception d'une impulsion de déclenchement provenant de la source de déclenchement sélectionnée. Ce mode est très utile saisir des événements simples.

Si le mode d'affichage est en position ALT (alternatif), une sélection spéciale de la version coup par coup ($SINGLE$) est possible par application du menu de pré-sélection ($APPL$).

$\Rightarrow$ 1. Appeler le menu $APPL$ en appuyant sur les touches $RESET$ et $AUTO\ SET$ simultanément.

$\Rightarrow$ 2. Sélectionner la touche de fonction $APPL$ du CT.

$\Rightarrow$ 3. Sélectionner la touche de fonction $FIRST$ ou $MULTI$ du CT.

$\Rightarrow FIRST$

Le balayage $MONO$ coup est toujours il, sur la voie A (pourvu que cette voie soit en service).

$\Rightarrow MULTI$

Balayage $MONO$ coup alternativement entre les voies A, B et ADD (pourvu que ces fonctions soient en service).

Quitter le menu de présélection d'application en enfonçant la touche $AUTO\ SET$.

Sélection d'une source de déclenchement par la touche $TRIG\ OR\ X\ SOURCE$. Les sources de déclenchement possibles sont à sélectionner :

$\Rightarrow A$

La voie A ;

$\Rightarrow A\ B$

La source composite voie A et voie B ;

$\Rightarrow B$

La voie B ;

$\Rightarrow EXT\ AC$

L'entrée vole extérieure, couplé en alternatif ;

$\Rightarrow EXT\ AC$

L'entrée voie extérieure, couplé en continu ;

$\Rightarrow LINE$

La source provenant de la tension secteur sinusoïdale à 50 Hz ou 60 Hz.

Dans le mode de déclenchement composite $A\ B$, la source de déclenchement alterne entre la voie A et la voie B, par conséquent ce mode n'est possible qu'en mode d'affichage alterné (ALT). Remarquer que dans ce mode les différentes traces produites à l'écran n'ont pas de rapport dans le temps, car les impulsions de déclenchement surviennent à des instants différents.

Le seuil de déclenchement procéder au réglage en tournant le bouton $TRIG\ LEVEL$. Pour les signaux répétitifs, l'observation stable sans sautillements de la trace est obtenue pour un déclenchement de la base de temps se produisant systématiquement au même point précis du signal. Le niveau de déclenchement se règle en tournant le bouton $TRIG\ LEVEL$.

Le couplage de déclenchement par la touche $TRIG\ COUPL$. Six modes sont possibles pour sélectionner le mode couplage de déclenchement.

$\Rightarrow P-P$ (cc)

Le déclenchement crête crête produit une calibration automatique du bouton $LEVEL$, la calibration étant fixée par les valeurs crête à crête du signal. Les composantes continues sont éliminées.

$\Rightarrow DC$ (continu)

Déclenchement continu. toute la bande de fréquence présente sur la voie déclenchement est disponible, la gamme de seuils possible par le bouton de réglage dépasse ± 8 divisions. Dans ce mode le seuil de déclenchement peut également être indiqué par la touche $LEVEL\ VIEW$. En appuyant sur la touche $LEVEL\ VIEW$ le seuil de déclenchement est considéré comme une tension

continue rattachée à la ligne médiane de l'écran. Après avoir réglé le bouton $LEVEL$ appuyer sur le bouton $LEVEL\ VIEW$ à nouveau pour retrouver l'écran d'origine.

$\Rightarrow TVF$

La trame TV permet aussi de déclencher suivant les impulsions de synchronisation lente trame TV, le bouton $LEVEL$ est inopérant.

$\Rightarrow TVL$

Le déclenchement sur ligne télévision permet le déclenchement sur les impulsions rapides de synchronisation ligne TV, le bouton $LEVEL$ est inopérant.

$\Rightarrow LF$

Dans ce mode, le Signal Utilisé Pour 10 déclenchement est d'abord passé au travers d'un filtre passe-bas avec une fréquence de coupure de 50 kHz. Tous les composants à fréquences élevées sont rejetés.

$\Rightarrow HF$

Dans ce, mode, le signal utilisé pour le déclenchement est d'abord passé au travers d'un filtre passe-haut avec une fréquence de coupure de 50 kHz. Tous les composants à fréquences basses sont rejetés.

Sélection de la pente de déclenchement par la touche front montant et front descendant. Nous obtenons le déclenchement sur front montant ou déclenchement sur front descendant. Si le mode de couplage de déclenchement TVF ou TFL a été sélectionné, l'écran à cristaux liquides indique un $+$ ou un $-$ à la place des signes de pente montant et descendant. Le signal $+$ est égal à vidéo positive, et le signal $-$ vidéo négative.

Si le mode horizontal $X\ DEFL$ a été sélectionné, l'écran à cristaux liquides indique $+$ et $-$ à la place des signes de pente du front montant et front descendant. Le signal $+$ est égal à signal non inversé et le signal $-$ est inversé.

Le temps de rétention ($hold-off$) sélectionné en tournant le bouton $HOLD\ OFF$. Le bouton $HOLD\ OFF$ est utilisé pour éviter les faux déclenchements et les double surimpressions lorsque nous examinons des signaux à impulsions multiples. Lorsqu'un balayage horizontal a été exécuté, le déclenchement est inhibé pendant un certain laps de temps par le bouton $HOLD\ OFF$. Le réglage à l'aide du bouton $HOLD\ OFF$ permet de synchroniser le déclenchement sur la même impulsion et ainsi de produire un affichage stable. Par exemple, considérons un signal d'entrée composé d'une double impulsion répétitive, le signal de déclenchement sélectionné détecte sur la seconde impulsion la même condition que sur la première, ainsi la base de temps démarre trop tôt. L'effet de sur-impression est observé comme une prolongation de la ligne de base par rapport à la trace supérieure. En ajustant le temps de rétention, nous inhibons le déclenchement jusqu'à ce que le circuit de déclenchement détecte la première impulsion à nouveau.

10.5 *Code de Couleurs*

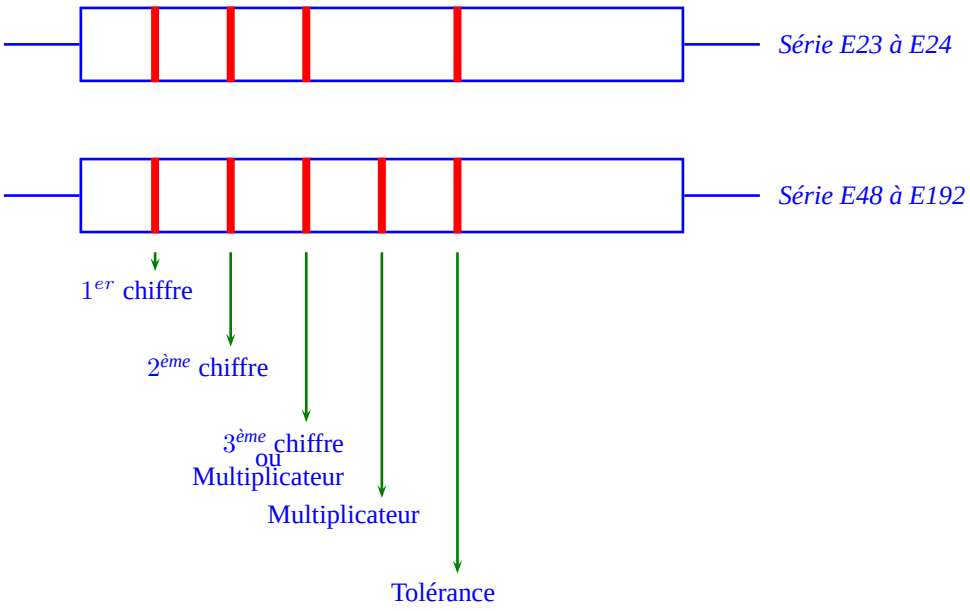

Figure 10.7: *Le code de couleurs.*

Couleur	1er chiffre	2ème chiffre	3ème chiffre	Multiplicateur	Tolérance
Argent				*x 0.01 Ω*	±10%
Or				*x 0.1 Ω*	±5%
Noir		*0*	*0*	*x 1 Ω*	±20%
Marron	*1*	*1*	*1*	*x 10 Ω*	±1%
Rouge	*2*	*2*	*2*	*x 100 Ω*	±2%
Orange	*3*	*3*	*3*	*x 1 kΩ*	
Jaune	*4*	*4*	*4*	*x 10 kΩ*	
Vert	*5*	*5*	*5*	*x 100 kΩ*	
Bleu	*6*	*6*	*6*	*x 1 MΩ*	
Violet	*7*	*7*	*7*		
Gris	*8*	*8*	*8*		
Blanc	*9*	*9*	*9*		

Table 10.3: *Marquage des résistances.*

Bibliographie

[1] Venn, J.,"On the Diagrammatic and Mechanical Representation of Propositions and Reasonings, The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science, Vol. 9, pp.1-18, 1880.

[2] Grünbaum, B.,"Venn Diagrams and Independent Families of Sets," Mathematics Magazine, January-February, pp. 13-23, 1975.

[3] Boole, G.,"An Investigation of the Laws of Thought," Dover Publications, New York, 1854.

[4] Huntingyon, E. V.,"Sets of Independent Postulates for the Algebra of Logic," Trans. American Math. Soc., **5**, pp. 288-309, 1904.

[5] Hill, F. J. & Peterson, G. R.,"Introduction to Switching Theory and Logical Design," John Wiley & Sons, New York, 1981.

[6] Kohavi, Z.,"Switching and Finite Automata Theory," Tata McGraw Hill Publishing Co. LTD., 2^{nd} Édition, New Delhi, 1978.

[7] Zuffo, J. A.,"Subsistemas Digitais e Circuitos de Pulso, " Livraria Editora Técnica Ltda, Vol. 1, 2^{a} Édition, São Paulo, 1974.

Partie V

Data Sheet