Safwan El Assad, Philippe Guyonnet

Électronique numérique

Fondements et applications

Incluant 57 exercices avec solutions détaillées,
5 travaux pratiques et un mini-projet en VHDL
avec éléments concis de solutions



ellipses

Électronique numérique

Références sciences

Électronique numérique

Fondements et applications

Safwan El Assad, Philippe Guyonnet

ellipses

Collection Références sciences

dirigée par Paul de Laboulaye
paul.delaboulaye@editions-ellipses.fr

Retrouvez tous les livres de la collection et des extraits sur www.editions-ellipses.fr

ISBN 9782340-106000
Dépôt légal : octobre 2025
©Ellipses Édition Marketing S.A.
8/10 rue la Quintinie 75015 Paris

DANGER LE PHOTOCOPILLAGE TUE LE LIVRE

Le Code de la propriété intellectuelle et artistique n'autorisant, aux termes des alinéas 2 et 3 de l'article L. 122-5, d'une part, que les « copies ou reproductions strictement réservées à l'usage privé du copiste et non destinées à une utilisation collective » et, d'autre part, que les analyses et les courtes citations dans un but d'exemple et d'illustration, « toute représentation ou reproduction intégrale, ou partielle, faite sans le consentement de l'auteur ou de ses ayants droit ou ayants cause, est illicite » (alinéa 1er de l'article L. 122-4).
Cette représentation ou reproduction, par quelque procédé que ce soit, constituerait donc une contrefaçon sanctionnée par les articles L. 335-2 et suivants du Code de la propriété intellectuelle.

www.editions-ellipses.fr

Remerciements

À nos familles au sens large.

À tous les collègues qui ont participé, d'une manière ou d'une autre, au contenu de ce livre. Nous citons par ordre chronologique : Jean-Paul Calvez, Sylvie Le Landais, Éric Friot, Olivier Pasquier, Patrick Le Callet, Pascal Chargé, Patrice Graziotin, Olivier Deforges, Fethi Dridi et Olivier Retif pour les photos.

Avant-propos

Nous avons rédigé cet ouvrage de formation sur l'électronique numérique dans l'esprit de présenter sous une forme intégrée les connaissances de base sur lesquelles s'appuient les systèmes numériques et la manière dont elles sont mises en œuvre techniquement, tant dans leur principe que dans leur réalisation. Cet ouvrage est le fruit d'une longue expérience de formation dans ce domaine en école d'ingénieurs (Polytech Nantes).

Il s'agit d'une formation complète : cours, travaux dirigés présentant de nombreux exercices types ciblés avec solutions détaillées, et travaux pratiques illustrant concrètement divers aspects des techniques de mise en œuvre.

Comme nous l'avons mentionné, bien que notre expérience soit d'abord celle de former des ingénieurs, nous avons, par des adaptations du contenu, souhaité nous adresser à des publics plus larges : d'abord en formation initiale, aux ingénieurs (en troisième année), aux Bachelors (deuxième et troisième année) spécialisés en génie électrique et informatique industrielle, en génie électrique et électronique, en informatique et électronique, aux BTS, aux licences spécialisées en EEA, ou d'autres spécialités connexes. Mais aussi aux formateurs en leur apportant, *via* les travaux dirigés et pratiques (TDP), des contenus qui peuvent leur être fort utiles dans la construction de la formation qu'ils dispensent. En formation continue, cet ouvrage s'adresse aussi aux techniciens en électronique.

Cet ouvrage, qui est composé de deux parties associées, est présenté dans son premier aspect de cours sous la forme d'une synthèse très concise et complète sur les fondements et applications en électronique numérique (Partie 1 du livre). Elle se compose de quatre chapitres. Le premier chapitre, concerne la représentation des nombres et conversion, l'arithmétique binaire, l'algèbre de Boole, et les fonctions logiques combinatoires (représentation et simplification). Il décrit aussi quelques opérateurs combinatoires standards (multiplexeurs / démultiplexeurs, codeurs / décodeurs, et transcodeurs), et traite enfin la question de la conception d'une fonction logique combinatoire quelconque. Le deuxième chapitre considère succinctement les aspects électriques et temporels des technologies TTL et CMOS. Le troisième chapitre introduit les éléments de base (les bascules) et les modèles (Moore et Mealy) pour la synthèse et l'analyse des fonctions séquentielles synchrones. Ensuite, les spécifications par chronogramme, automate à états finis, algorithme et symbole logique sont abordées. Enfin, la démarche de conception des fonctions séquentielles synchrones et diverses applications

sont données. Le quatrième chapitre, décrit les fonctionnalités essentielles en VHDL, et le flux de conception et outils (Vivado) pour la conception et l'implantation sur carte FPGA d'une description VHDL donnée. Puis à travers des exemples concrets, les notions de structure d'une description VHDL (« *librairies, entity, architecture* »), et opérateurs et éléments de syntaxe sont introduites. Ensuite, quelques applications concrètes (multiplexeur, compteur/décompteur, automate à états finis) sont implantées et évaluées (performances matérielles). Enfin, les étapes de la simulation (*Testbench*, utilisant le simulateur VHDL) sont réalisées via un exemple compteur/décompteur.

Un deuxième aspect fort important dans l'apprentissage des connaissances et du savoir-faire d'un apprenant compose cet ouvrage (Partie 2A du livre). Il concerne les travaux dirigés (TD) d'une formation. Il s'agit ici d'un ensemble ordonné de 57 exercices types avec solutions détaillées couvrant les différents chapitres du cours. L'ensemble doit permettre à un apprenant de comprendre progressivement et en profondeur l'essentiel de ce domaine et d'acquérir les compétences nécessaires pour les pratiquer dans le monde industriel.

Enfin, le dernier aspect concerne la pratique au sens propre du terme, complément indispensable d'une formation allant jusqu'au savoir-faire (Partie 2B du livre). Nous proposons ici d'une part, un ensemble de 5 travaux pratiques (3 H chacun) dont 4 travaux portent sur la conception et réalisation des fonctions logiques combinatoire et séquentielle et le cinquième porte sur la description VHDL et simulation (*Testbench*), et d'autre part, un mini-projet en VHDL (18 H). L'intérêt de ceux-ci est qu'ils vont des conceptions et réalisations des fonctions combinatoires et séquentielles jusqu'à la description en VHDL, implantation sur carte FPGA et simulation des applications concrètes.

Remarque : Les auteurs mettent à la disposition des lecteurs le TP et le mini-projet en VHDL. Ils aident également au montage des quatre TP matériels. Pour accéder à ces informations, vous pouvez contacter les auteurs directement à ces adresses : safwan.elassad@univ-nantes.fr et safwan.elassad@gmail.com

Table des matières

PARTIE 1 : COURS 1

CHAPITRE 1. LOGIQUE COMBINATOIRE 3

1.1. Représentation des nombres et conversion 3
1.1.1. Numération binaire 3
1.1.2. Changement de base de numération 3
1.1.2.1. De la base ***b*** vers la base 10 3
1.1.2.2. De la base 10 vers la base ***b*** 3
1.1.2.2.1. Détermination de la partie entière par la méthode divisions 4
1.1.2.2.2. Détermination de la partie entière par la méthode soustractions successives de la plus grande puissance entière de la base 5
1.1.2.2.3. Détermination de la partie fractionnaire : 5
1.1.2.3. Passage de la base 2^n vers la base 2 6
1.1.2.4. Passage de la base 2 vers la base 2^n 7
1.1.2.5. Passage d'une base ***i*** à une base ***j*** 7
1.1.3. Norme IEEE 754 pour représenter des nombres à virgule flottante, cas format simple précision sur 32 bits 7

1.2. Arithmétique binaire 9

1.3. Fonctions logiques binaires de base 10

1.4. Représentation des nombres binaires signés (entiers relatifs) 13
1.4.1. Représentation sous forme module plus signe 13
1.4.2. Représentation sous forme de complément à deux 14

1.5. Codage de l'information 16
1.5.1. Codes couramment utilisés en électronique numérique 17
1.5.1.1. Codes décimaux utilisés pour la représentation des chiffres du système décimal 17
1.5.1.2. Code de Gray (code à distance unité) ou code binaire réfléchi 17
1.5.1.3. Code ASCII : *American Standard Code for Information Interchange* 19

1.6. Fonctions logiques – Algèbre de Boole (1850) 20
1.6.1. Variable logique ou booléenne 20
1.6.2. Normes CEI (internationale) et MIL (américaine) pour la représentation des portes logiques de deux variables 21
1.6.3. Algèbre de Boole 22
1.6.3.1. Lois fondamentales 22
1.6.3.2. Relations de base 22
1.6.3.3. Théorèmes de De Morgan et de Shannon 23

1.7. Représentation et simplification des fonctions logiques combinatoires..... 23
1.7.1. Représentation d'une fonction logique..... 23
1.7.1.1. Écriture algébrique..... 23
1.7.1.2. Table de vérité..... 24
1.7.1.3. Table de Karnaugh..... 24
1.7.1.4. Formes canoniques..... 25
1.7.1.5. Logigramme..... 26
1.7.2. Simplification (ou réduction) des équations logiques..... 26
1.7.2.1. Simplification par la méthode algébrique..... 26
1.7.2.2. Simplification par la table de Karnaugh..... 26

1.8. Opérateurs combinatoires standards..... 28
1.8.1. Multiplexeurs / Démultiplexeurs..... 28
1.8.1.1. Multiplexeur $2^n \rightarrow \mathbf{1}$..... 28
1.8.1.2. Démultiplexeur $\mathbf{1} \rightarrow \mathbf{2}^n$..... 30
1.8.1.3. Applications des multiplexeurs / démultiplexeurs..... 30
1.8.1.3.1. Générateur de fonctions..... 31
1.8.1.3.2. Conversion parallèle-série..... 33
1.8.2. Codeurs ou encodeurs / décodeurs - transcodeurs..... 34
1.8.2.1. Encodeur binaire de priorité $\mathbf{2}^n \rightarrow \boldsymbol{n}$..... 34
1.8.2.2. Décodeur binaire $\boldsymbol{n} \rightarrow \mathbf{2}^n$..... 35
1.8.2.3. Transcodeurs..... 38
1.8.2.3.1. Transcodeur $\boldsymbol{Y} = \mathbf{log}_2 \boldsymbol{X}$..... 38
1.8.2.3.2. Transcodeur DCB – Affichage sur 7 segments..... 39

1.9. Conception d'une fonction combinatoire quelconque..... 42
1.9.1. Spécifications des fonctions combinatoires..... 42
1.9.2. Modèles de structure..... 43
1.9.3. Technologies de réalisation..... 45
1.9.4. Réalisation..... 45

CHAPITRE 2. CONSIDERATIONS TECHNOLOGIQUES..... 47

2.1. Aspects temporels et électriques..... 47

2.2. Variantes technologiques des familles logiques TTL et CMOS..... 49
2.2.1. Variantes technologiques de la famille TTL..... 50
2.2.2. Variantes technologiques de la famille CMOS..... 50

2.3. Réalisation des portes logiques TTL et CMOS..... 51
2.3.1. Porte NAND en TTL..... 51
2.3.2. Portes : inverseur, NAND et NOR en technologie CMOS..... 52
2.3.2.1. Porte inverseur CMOS..... 53
2.3.2.2. Porte NAND CMOS..... 53
2.3.2.3. Porte NOR CMOS..... 54

2.4. Interfaçage des circuits logiques..... 55
2.4.1. Interfaçage des circuits logiques CMOS/TTL et TTL/CMOS 55
2.4.2. Commande (polarisation) d'une entrée d'un circuit logique par un contact 56
2.4.2.1. Contact commandant un niveau bas, polarisation via une résistance de tirage (pull-up)..... 56
2.4.2.2. Contact commandant un niveau haut, polarisation via une résistance de rappel (pull-down)..... 57

2.5. Entrance (*FAN-IN*) – Sortance (*FAN-OUT*) 57

2.6. Sorties à trois états (Tri–state) : Exemple LVTTL 59

CHAPITRE 3. LOGIQUE SEQUENTIELLE 61

3.1. Introduction..... 61

3.2. Fonctions de base de la logique séquentielle : les bascules..... 62
3.2.1. Types de synchronisation d'une bascule 62
3.2.1.1. Synchronisation sur niveau haut ou bas de l'horloge (Bascules $\boldsymbol{RS}$ et $\boldsymbol{D}$ Latch) 62
3.2.1.2. Synchronisation sur front montant ou front descendant de l'horloge..... 62
3.2.1.3. Synchronisation maître-esclave..... 62
3.2.1.4. Synchronisation sur front maître-esclave..... 63
3.2.2. Conditions de bon fonctionnement dynamique à respecter..... 63
3.2.2.1. Conditions de bon fonctionnement dynamique de la bascule *JK* synchronisée sur front montant de l'horloge..... 63
3.2.2.2. Conditions de bon fonctionnement dynamique de la bascule $\boldsymbol{JK}$ synchronisée maître-esclave 64
3.2.2.3. Conditions de bon fonctionnement dynamique de la bascule *JK*, synchronisée sur front maître-esclave 64
3.2.3. Symbole logique, tables de vérité et équations des différentes bascules..... 65
3.2.3.1. Bascule $\boldsymbol{RS}$ asynchrone 65
3.2.3.2. Bascule $\boldsymbol{RS}$ synchrone 65
3.2.3.3. Bascule $\boldsymbol{JK}$ synchrone avec entrées $\boldsymbol{RS}$ asynchrones 66
3.2.3.4. Bascule $\boldsymbol{D}$ synchrone 66
3.2.3.5. Bascule $\boldsymbol{T}$ synchrone..... 67
3.2.4. Table des transitions et graphe des états des différentes bascules 68

3.3. Démarche de conception des fonctions séquentielles synchrones et réalisation..... 68

3.4. Structure de la machine de Moore 69

3.5. Structure de la machine de Mealy..... 70

3.6. Fréquence maximale de fonctionnement des Machines de Moore et de Mealy..... 71

3.7. Synthèse d'un compteur synchrone à deux cycles de fonctionnement 71
3.7.1. Comportement des états hors cycle (HC) : Analyse du compteur conçu 73
3.7.2. Graphe des états du compteur 74
3.7.3. Synthèse d'un compteur synchrone, suite 74

3.8. Spécifications 76
3.8.1. Spécifications par chronogramme 76
3.8.2. Spécifications par automate à états finis sur niveaux (Finite State Machine) 76
3.8.3. Spécifications par algorithme 78
3.8.4. Spécifications par symbole logique 78

3.9. Difficultés spécifiques liées aux systèmes séquentiels par rapport aux systèmes combinatoires 79
3.9.1. Fréquence maximale de fonctionnement 79
3.9.2. Initialisation à la mise sous tension 79
3.9.3. Problème lié à l'instant d'activation de l'entrée : conséquences du non-respect de la fenêtre temporelle garantissant un échantillonnage correct de l'entrée d'une bascule 79
3.9.4. Evolution temporelle des sorties : cas le plus général avec synchronisation des entrées 81

3.10. Applications spécifiques 81
3.10.1. Sous-cadencement synchrone de fonctions séquentielles 81
3.10.2. Mise en cascade de compteurs synchrones (décimaux) 83
3.10.3. Conception des générateurs de signaux 84
3.10.4. Conception des contrôleurs de signaux 84

CHAPITRE 4. LE VHDL : INITIATION – APPLICATIONS 87

4.1. Généralités et principes fondamentaux 87

4.2. Fonctionnalités essentielles en VHDL et en Verilog 87

4.3. Étapes d'un flux de conception VHDL avec l'outil Vivado 87

4.4. Architecture des cartes FPGA Xilinx séries-7 88

4.5. Structure d'une description de circuit en VHDL 91

4.6. Structure d'une description de circuit en VHDL, suite 95

4.7. Déclaration de l'entité (Entity declaration) : éléments de syntaxe 96

4.8. Autres types prédéfinis en VHDL 97

4.9. Conversion de type en VHDL (VHDL type conversion)..... 98

4.10. Définition de l'architecture et instanciation (*architecture definition, and instantiation*)..... 99

4.11. Opérateurs prédéfinis pour VHDL..... 100

4.12. Éléments de syntaxe 101
4.12.1. Les affectations du mode « concurrent ou parallèle » : utilisées hors processus..... 102
4.12.2. Les structures du mode « séquentiel » utilisées seulement à l'intérieur des processus..... 102

4.13. Exemples concrets en VHDL 104
4.13.1. Multiplexeur 4 to 1_8bits 104
4.13.2. Compteur/décompteur modulo 12, avec reset asynchrone et utilisation de variable locale/signal global 107
4.13.2.1. Description VHDL du compteur/décompteur modulo 12, avec reset asynchrone et utilisation de variable locale 107
4.13.2.2. Compteur/décompteur modulo 12, avec reset asynchrone et utilisation de signal global 108
4.13.3. Automate à états finis « FSM1 » : description selon la machine de Moore 111

4.14. Simulation (Testbench) du compteur/décompteur modulo 12, décrit dans le paragraphe 4.13.2.1..... 118

PARTIE 2A : TRAVAUX DIRIGES 121

CHAPITRE 1. LOGIQUE COMBINATOIRE : EXERCICES 1 A 21 123

1.1. Exercice 1 - Représentation des nombres et conversion 123

1.2. Exercice 2 - Décalage et codage ASCII 129

1.3. Exercice 3 - Représentation et addition des nombres binaires en complément à deux : $C2N$ 130

1.4. Exercice 4 - Analyse d'un schéma logique combinatoire 133

1.5. Exercice 5 - Représentation d'une fonction combinatoire logique sous diverses formes..... 134

1.6. Exercice 6 - Réalisation d'une fonction combinatoire à partir des circuits intégrés 136

1.7. Exercice 7 - Visualisation d'un état parmi un ensemble d'états possibles d'un dispositif 138

1.8. Exercice 8 - Montage avec multiplexeurs 140

1.9. Exercice 9 - Analyse d'un montage et réalisation par des circuits intégrés 143

1.10. Exercice 10 - Opérateur logique universel 145

1.11. Exercice 11 - Transcodage 148

1.12. Exercice 12 - Réalisation d'une fonction de transcodage 151

1.13. Exercice 13 - Réalisation d'une fonction de transcodage 152

1.14. Exercice 14 - Réalisation d'une fonction de transcodage 154

1.15. Exercice 15 - Réalisation d'une fonction combinatoire 156

1.16. Exercice 16 - Encodeur de priorité 158

1.17. Exercice 17 - Conversion d'un nombre signé sous forme module plus signe en un nombre signé sous forme complément à 2 159

1.18. Exercice 18 - Étude d'un soustracteur 3 bits en complément à 2 161

1.19. Exercice 19 - Association de plusieurs additionneurs 4 bits : calcul de temps de propagation 164

1.20. Exercice 20 - Unité arithmétique rudimentaire pour l'addition et la soustraction en complément à 2 166

1.21. Exercice 21 - Circuit de « $Reset$ » avec trigger de Schmitt 168

CHAPITRE 2. CONSIDERATIONS TECHNOLOGIQUES : EXERCICES 22 A 30 181

2.1. Exercice 22 - Structure interne d'une porte NAND en technologies TTL et CMOS .. 181

2.2. Exercice 23 - Connexion de portes à sorties trois-états (LVTTL) 185

2.3. Exercice 24 - Opérateur logique en technologie CMOS 188

2.4. Exercice 25 - Sortance en technologie CMOS 189

2.5. Exercice 26 - Interface entre un circuit TTL-LS et un circuit CMOS-B 194

2.6. Exercice 27 - Polarisation des entrées d'un opérateur logique TTL-LS 196

2.7. Exercice 28 - Commande d'une LED par un opérateur logique 197

2.8. Exercice 29 - Adaptation de niveaux logiques 198

2.9. Exercice 30 - Pull-up, mode haute impédance et mode totem pole 201

CHAPITRE 3. LOGIQUE SEQUENTIELLE : EXERCICES 31 A 49..... 209

3.1. Exercice 31 - Analyse d'un registre à décalage à 3 étages rebouclé sur lui-même.. 209

3.2. Exercice 32 - Analyse d'un circuit séquentiel synchrone à base des bascules D 211

3.3. Exercice 33 - Analyse d'un circuit séquentiel synchrone à base des bascules J K ... 212

3.4. Exercice 34 - Synthèse d'un compteur synchrone à base des bascules D 214

3.5. Exercice 35 - Synthèse d'un compteur synchrone à base des bascules J K et des bascules D 216

3.6. Exercice 36 - Synthèse d'un compteur synchrone à deux modes : modulo 6 ou modulo 4 220

3.7. Exercice 37 - Génération d'une forme d'onde 223

3.8. Exercice 38 - Générateur de signaux périodiques synchrones 225

3.9. Exercice 39 - Conception d'un registre universel spécifié par son symbole logique 229

3.10. Exercice 40 - Conception d'un compteur universel spécifié par son symbole logique 231

3.11. Exercice 41 - Analyse d'un système séquentiel : temps de propagation et fréquence maximale de l'horloge 236

3.12. Exercice 42 - Conception d'un contrôleur synchrone : gestion BP (auto-repeat plus anti-rebond) 240

3.13. Exercice 43 - Conception d'un système synchrone : cycle de fabrication mélange 242

3.14. Exercice 44 - Implantation d'un automate à états finis selon une machine de Moore..... 246

3.15. Exercice 45 - Conception d'un contrôleur synchrone : système d'éclairage à deux niveaux de puissance 250

3.16. Exercice 46 - Synthèse d'un contrôleur synchrone : fonction « arbitre » 253

3.17. Exercice 47 - Conception d'un contrôleur synchrone : commande de relais temporisée..... 256

3.18. Exercice 48 - Spécification d'un contrôleur synchrone : dispositif d'éclairage arrière de bicyclette..... 261

3.19. Exercice 49 - Conception d'un système séquentiel synchrone de commande des essuie-glaces 262

CHAPITRE 4. VHDL : EXERCICES 50 A 57..... 267

4.1. Exercice 50 - Écriture d'un modèle VHDL combinatoire simple 267

4.2. Exercice 51 - Écriture d'un modèle VHDL de l'opérateur logique universel de l'exercice 10..... 273

4.3. Exercice 52 - Écriture d'un modèle VHDL du soustracteur 3 bits en complément à 2 de l'exercice 18. 275

4.4. Exercice 53 - Ecriture d'un modèle VHDL pour la réalisation d'un compteur synchrone..... 284

4.5. Exercice 54 - Modèle VHDL d'une machine de Moore/Mealy 290

4.6. Exercice 55 - Analyse d'un modèle VHDL 296

4.7. Exercice 56 - Écriture d'un modèle VHDL : sous-cadencement d'opérateurs séquentiels synchrones..... 298

4.8. Exercice 57 - Écriture d'un modèle VHDL : Diviseur de fréquence 302

PARTIE 2B : TRAVAUX PRATIQUES 305

CHAPITRE 5. DECODAGE - AFFICHAGE - TRANSCODAGE 307

5.1. Étude d'un afficheur 7 segments et d'un décodeur DCB -7 segments..... 307

5.2. Utilisation du décodeur DCB - 7 segments « SN74LS47 » 309

5.3. Transcodage et affichage du résultat..... 309

Éléments de solution du chapitre 5 310

CHAPITRE 6. COMPARAISON LOGIQUE COMBINATOIRE..... 315

6.1. Synthèse d'un comparateur combinatoire 315

6.2. Analyse du comparateur intégré de deux nombres de quatre bits chacun, circuit « SN7485 » et extension de la comparaison aux mots à huit bits 316

Éléments de solution du chapitre 6 316

CHAPITRE 7. SYNTHESE D'UN COMPTEUR SYNCHRONE A L'AIDE DE BASCULES 319

7.1. Synthèse d'un compteur synchrone configurable 319

7.2. Application à la commande d'un chenillard..... 319

7.3. Analyse du fonctionnement d'un compteur intégré 320

7.4. Altération du cycle d'un compteur standard 320

Éléments de solution du chapitre 7 321

CHAPITRE 8. CONVERSION PARALLELE - SERIE ; SERIE - PARALLELE..... 323

8.1. Conversion Parallèle - Série..... 323

8.2. Conversion parallèle - série – parallèle..... 323

Éléments de solution du chapitre 8 324

CHAPITRE 9. DESCRIPTION VHDL D'UN COMPTEUR UNIVERSEL ET SIMULATION DE SON FONCTIONNEMENT COMPORTEMENTAL AVEC VIVADO 327

Éléments de solution du chapitre 9 328

CHAPITRE 10. MINI-PROJET : MODELISATION VHDL ET SYNTHESE LOGIQUE SUR CIBLE FPGA 329

10.1. Objectif 329

10.2. Cahier des charges (CDC) : fonctionnement souhaité 329

10.3. Interface homme machine (IHM) du digicode 330

10.4. Cadencement des fonctions séquentielles 330

10.5. Spécifications fonctionnelles du circuit 332
10.5.1. Comportement vis-à-vis des entrées boutons poussoirs « Bps » : fonction filtrage 332
10.5.2. Comportement vis-à-vis des entrées « Bp_Suiv », « Bp_Prec » : fonction choix chiffre 333
10.5.3. Comportement vis-à-vis des entrées « Bp_Inc » et « Bp_Dec » : fonction modification de valeur 334
10.5.4. Comportement vis-à-vis de l'entré « Bp_Valid » : fonction autorisation ouverture 335
10.5.5. Pilotage des afficheurs 336
10.5.6. Fonction clignotement 336
10.5.7. Générateur enable : **clk4_en** et **clk20_en** 337

10.6. Conception architecturale 337

10.7. Méthode de travail et règles à respecter impérativement 338

10.8. Réalisation : écriture du modèle VHDL du digicode 340
10.8.1. Affichage des 3 digits multiplexés manuellement 341
10.8.2. Affichage des 3 digits multiplexés automatiquement (balayage périodique) ... 342
10.8.3. Clignotement d'un « digit » à 1 Hz 343
10.8.4. Combinaison numérique modifiable de 000 à 999 343
10.8.4.1. Écriture d'un modèle VHDL de compteur/décompteur modulo 10 344
10.8.4.2. Instanciation des trois compteurs-décompteurs 344
10.8.5. Écriture d'un modèle VHDL de filtre anti-rebond et instanciation 345
10.8.6. Choix du chiffre à modifier 346
10.8.7. Modification de valeur du chiffre sélectionné (« Bp_Inc », « Bp_Dec » plus « auto-repeat ») 347
10.8.8. Fonction « autorisation ouverture » : comportement vis-à-vis de l'entrée « Bp_Valid » 347

Éléments de solution du chapitre 10 347

ANNEXES CIRCUITS LOGIQUES 373

Annexe sur le chapitre 1 : logique combinatoire 373

Annexe sur le chapitre 2 : considérations technologiques 378

Annexe sur le chapitre 3 : logique séquentielle 380

BIBLIOGRAPHIE..... 383

INDEX 385

Partie 1 : Cours

Chapitre 1. Logique combinatoire

1.1. Représentation des nombres et conversion

1.1.1. Numération binaire

Un nombre N exprimé en base b donnée, $(N)_b$ est écrit sous forme d'une juxtaposition de symboles (chiffres) appartenant à la base b en question.

$$(N)_b = (a_n\ a_{n-1} \cdots a_i\ \cdots a_1\ a_0\ , a_{-1}\ a_{-2}\ \cdots a_{-m})_b \quad (1.1)$$

La valeur numérique de ce nombre est obtenue en exprimant sous forme polynomiale le nombre $(N)_b$, soit :

$$\begin{aligned}(N)_b = a_n b^n + a_{n-1}b^{n-1} + \cdots + a_i b^i + \cdots + a_1 b^1 + a_0 b^0, + a_{-1}b^{-1} + a_{-2}b^{-2} + \cdots \\ + a_{-m}b^{-m} \\ = \sum_{k=-m}^{n} a_k b^k \end{aligned} \quad (1.2)$$

Où : a_k : symbole (chiffre) de rang k, appartenant à la base b

b^k : poids du symbole (chiffre) a_k

k : rang du symbole (chiffre) a_k Les bases utilisées en pratique sont :

base 2 (binaire), $b = 2$, $a_k \in [0, 1]$,

base 8 (octale), $b = 8$, $a_k \in [0, 1, \cdots, 6, 7]$,

base 10 (décimale), $b = 10$, $a_k \in [0, 1, \cdots, 8, 9]$,

base 16 (hexadécimale), $b = 16$, $a_k \in [0, 1, \cdots, 8, 9, A, B, C, D, E, F]$, avec $A = 10, \cdots, F = 15$.

1.1.2. Changement de base de numération

1.1.2.1. De la base b vers la base 10

Dans ce cas, il suffit d'exprimer le nombre sous sa forme polynomiale et de faire le calcul

Exemples :

$$\begin{aligned}(101,11)_2 &= 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0, 1 \times 2^{-1} + 1 \times 2^{-2} = 4 + 0 + 1, 0{,}5 + 0{,}25 \\ &= (5{,}75)_{10} \\ (257,24)_8 &= 2 \times 8^2 + 5 \times 8^1 + 7 \times 8^0, 2 \times 8^{-1} + 4 \times 8^{-2} \\ &= 128 + 40 + 7, 0{,}25 + 0{,}0625 = (175{,}3125)_{10} \\ (2A9F, A)_{16} &= 2 \times 16^3 + 10 \times 16^2 + 9 \times 16^1 + 15 \times 16^0, 10 \times 16^{-1} \\ &= 8192 + 2560 + 144 + 15, 0{,}625 = (10911{,}625)_{10}\end{aligned}$$

1.1.2.2. De la base 10 vers la base b

Pour convertir un nombre écrit en base 10, $(N)_{10}$ en un nombre écrit en base b donné, $(Nx)_b$, il faut trouver les valeurs des différents symboles a_k de la base b.

$$(Nx)_b = (a_n\ a_{n-1}\ \cdots\cdots a_0, a_{-1}\ a_{-2}\ \cdots\cdots a_{-m})_b$$
$$= (a_n\ a_{n-1}\ \cdots\cdots a_0) + (0, a_{-1}\ a_{-2}\ \cdots\cdots a_{-m}\ )$$
$$= PE(Nx)_b + PF(Nx)_b \qquad (1.3)$$

Où : $PE(Nx)_b$ est la partie entière du nombre et $PF(Nx)_b$ est la partie fractionnaire du nombre.

Pour la partie entière du nombre, deux méthodes peuvent être utilisées pour déterminer les valeurs des différents symboles a_k, qui sont :

- divisions successives par la base b ;
- soustractions successives de la plus grande puissance entière de la base b.

Pour la partie fractionnaire du nombre une seule méthode est disponible pour déterminer les valeurs des différents symboles a_k :

- multiplications successives par la base b.

1.1.2.2.1. Détermination de la partie entière par la méthode divisions successives par la base b : $PE(N)_{10} \Longrightarrow PE(Nx)_b$

Écrivons la partie entière $PE(N)_b$ sous forme polynomiale :

$PE(N)_b = a_n b^n + a_{n-1}b^{n-1} + \cdots\cdots + a_1 b^1 + a_0$

Faisons les divisions successives par la base b :

$PE(N)_b/b = \underbrace{[a_n b^{n-1} + a_{n-1}b^{n-2} + \cdots\cdots + a_1]}_{quotien=q_1} + \underbrace{a_0}_{reste}$: la première division par b détermine le symbole a_0

$q_1/b = \underbrace{[a_n b^{n-2} + a_{n-1}b^{n-3} + \cdots\cdots + a_2]}_{quotien=q2} + \underbrace{a_1}_{reste}$: la deuxième division par b détermine le symbole a_1

Et ainsi de suite.

Après $(n-1)$ divisions, on aboutit à :

$q_{(n-1)}/b = \underbrace{a_n}_{q_n} + \underbrace{a_{n-1}}_{reste}$

Exemple : $(75)_{10} = (?\cdots?)_2$

Par divisons successives on obtient :

75	2					
–74	37	2				
1 a_0	–36	18	2			
	1	–18	9	2		
		0	–8	4	2	
			1	–4	2	2
				0	–2	1 a_6
					0	

D'où :

$(75)_{10} = (1\ 0\ 0\ 1\ 0\ 1\ 1\ )_2$

a_6 a_0

1.1.2.2.2. Détermination de la partie entière par la méthode soustractions successives de la plus grande puissance entière de la base

$b : PE(N)_{10} \Longrightarrow PE(Nx)_b$

On cherche i tel que : $b^i \leq PE(N)_{10} < b^{i+1}$

Puis on soustrait b^i au nombre de départ : $PE_1(N)_{10} = PE(N)_{10} - b^i$

On répète ces deux opérations avec $PE_1(N)_{10}$, et ainsi de suite jusqu'à ce que $PE_k(N)_{10} < b$

Appliquons la méthode sur l'exemple précédent $(75)_{10} = (?\cdots?)_2$

i	6	5	4	3	2	1	0	
b^i	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
	64	32	16	8	4	2	1	
	1 a_6	0 a_5	0 a_4	1 a_3	0 a_2	1 a_1	1 a_0	
								75 –64 ← 1×2^6
								11 –8 ← 1×2^3
								3 –2 ← 1×2^1
								1 –1 ← 1×2^0
								0

On trouve forcément le même résultat obtenu par la méthode de divisions successives.

1.1.2.2.3. Détermination de la partie fractionnaire : $PF(N)_{10} \Longrightarrow PF(Nx)_b$

La multiplication successive par la base b permet d'avoir les différents symboles $a_{-i}\ i = 1, 2, \cdots$

Écrivons la partie fractionnaire $PF(N)_b$ sous forme polynomiale :

$$PF(N)_b = a_{-1}b^{-1} + a_{-2}b^{-2} + a_{-3}b^{-3} \cdots\cdots + a_{-m}b^{-m}$$

Faisons les multiplications successives par la base b .

$PF(N)_b \times b = \underbrace{a_{-1}}_{Partie\ entière} + \underbrace{a_{-2}b^{-1} + a_{-3}b^{-2} \cdots + a_{-m}b^{-(m-1)}}_{PF_1 : Nouvelle\ partie\ fractionnaire}$: première étape détermine le symbole a_{-1}

$PF_1 \times b = \underbrace{a_{-2}}_{Partie\ entière} + \underbrace{a_{-3}b^{-1} + a_{-4}b^{-2} \cdots + a_{-m}b^{-(m-2)}}_{PF_2 : Nouvelle\ partie\ fractionnaire}$: deuxième étape détermine le symbole a_{-2}

On répète ce procédé jusqu'à obtenir une « partie fractionnaire nulle » ou jusqu'au rang nécessaire pour satisfaire une précision donnée. Attention, une suite fractionnaire limitée dans une base n'est pas nécessairement limitée dans une autre.

Exemple : $(0{,}27)_{10} = (?\cdots?)_2$

$0{,}27 \times 2 = 0{,}54$ a_{-1}
$0{,}54 \times 2 = 1{,}08$
$0{,}08 \times 2 = 0{,}16$
$0{,}16 \times 2 = 0{,}32$
$0{,}32 \times 2 = 0{,}64$
$0{,}64 \times 2 = 1{,}28$
$0{,}28 \times 2 = 0{,}56$ a_{-7}
⋮

La suite fractionnaire n'est pas encore terminée. Elle peut être illimitée, périodique ou non.

Si on arrête comme ici le procédé à 7 chiffres après la virgule, on fait une erreur maximale de 2^{-7}, cas extrême d'une suite illimitée de 1 à partir du symbole a_{-8}

D'où :

$$(0{,}27)_{10} = \left(\begin{array}{c} 0,\ 0\ 1\ 0\ 0\ 0\ 1\ 0 \\ a_{-1} \qquad\qquad a_{-7} \end{array} \right)_2$$

Rappelons que les trois méthodes de conversion ci-dessus s'appliquent quelle que soit la base b utilisée. Les exemples donnés ci-dessus ont été appliqués sur la base binaire $b = 2$

1.1.2.3. Passage de la base 2^n vers la base 2

En électronique numérique, en particulier en micro-informatique, on utilise essentiellement les bases $2^3 = 8$ (octal) et 2^4 (hexadécimal).

Le passage d'une de ces bases à la base 2 consiste simplement à traduire les chiffres octaux (ou hexadécimaux) en leurs équivalents binaires.

Chaque chiffre octal (respectivement hexadécimal) s'écrit en binaire avec 3 (respectivement 4) bits.

Exemples : $(3A9)_{16} = (?\cdots?)_2$ et $(264)_8 = (?\cdots?)_2$

$$\begin{cases} 3 & A & 9 \\ 0011 & 1010 & 1001 \end{cases} \text{ et } \begin{cases} 2 & 6 & 4 \\ 010 & 110 & 100 \end{cases}$$

D'où : $(3A9)_{16} = (001110101001)_2$ et $(264)_8 = (010110100)_2$

1.1.2.4. Passage de la base 2 vers la base 2^n

Pour passer de la base 2 à la base 8 (respectivement 16), il suffit de grouper les chiffres binaires (bits) par 3 (respectivement 4) et convertir ces sous-ensembles en leur équivalent octal (respectivement hexadécimal).

Exemples :

$(1011101,011\,01)_2 = (?\cdots?)_8$ et $(010001101101001,0110001101)_2 = (?\cdots?)_{16}$

$\begin{cases} (001\ 011\ 101,011\ 010)_2 \\ (\ 1\quad 3\quad 5,3\quad 2)_8 \end{cases}$ et $\begin{cases} (0010\ 0011\ 0110\ 1001,0110\ 0011\ 0100)_2 \\ (\ 2\quad 3\quad 6\quad 9,6\quad 3\quad 4)_{16} \end{cases}$

D'où :

$(1011101,011\,01)_2 = (135,32)_8$

et $(010001101101001,0110001101)_2 = (2369,634)_{16}$

1.1.2.5. Passage d'une base i à une base j

Les cas où i et j valent 2 et 10 ont été traités dans les paragraphes précédents.

Dans le cas général, deux situations peuvent se présenter :

- i et j sont des puissances de 2 alors on passe par la base 2 :

$(N)_i = (N')_2 = (N'')_j$

- i et j ne sont pas des puissances de 2 alors on passe par la base 10 :

$(N)_i = (N')_{10} = (N'')_j$

1.1.3. Norme IEEE 754 pour représenter des nombres à virgule flottante, cas format simple précision sur 32 bits

Un nombre simple précision sur 32 bits (1 bit de signe, 8 bits d'exposant et 23 bits de mantisse) représenté en base binaire IEEE, s'écrit (les 3 autres formats de la norme IEEE 754 sont traités de la même manière que ci-dessous):

$$(N)_{2IEEE} = \underbrace{s}\ \underbrace{E\cdots\cdots\cdots E}\ \underbrace{m\cdots\cdots\cdots\cdots\cdots\cdots\cdots m} \qquad (1.4)$$

↑	↑	↑
1 bit de signe	Exposant E sur 8 bits	Mantisse m sur 23 bits

Si $s = 1$, le nombre est négatif, autrement $(s = 0)$ le nombre est positif.

La valeur numérique du nombre peut se mettre sous la forme suivante :

$$(Nx)_{10} = V = (-1)^s \times 2^{(E-127)} \times \underbrace{(1,m)_2}$$

Équivalent décimal

Où dans le terme $\underbrace{(1,m)_2}$ le 1 est implicite. Soit encore :

$$(Nx)_{10} = V = (-1)^s \times 2^e \times \underbrace{(1,m)_2}_{\text{Équivalent décimal}} \qquad (1.5)$$

avec : $e = E - 127$ qui est l'exposant après décalage de 127

Exemples : $(-101{,}5)_{10} = (\,?\,)_{16IEEE}$ et $(47800180)_{16IEEE} = (\,?\,)_{10}$

Pour $(-101{,}5)_{10} = (\,?\,)_{16IEEE}$, on convertit d'abord $(-101{,}5)_{10}$ en base binaire $(\,?\,)_2$, puis on met le résultat sous la forme donnée par la relation (1.5), qui permet d'identifier tous les paramètres de la forme $(\,?\,)_{2IEEE}$. On écrit le résultat sous la forme $(\,?\,)_{2IEEE}$, donnée par la relation (1.4), ensuite on passe à la forme demandée $(\,?\,)_{16IEEE}$.

$(-101{,}5)_{10} = (-1100101{,}1\,)_2 = (-1{,}1001011\,)_2 \times 2^6$

Écrivons le résultat obtenu sous la forme donnée par la relation (1.5) puis identifions s, e et m puis déduisons E.

$$(-1{,}1001011\,)_2 \times 2^6 = (-1)^1 \times 2^6 \times (1,m)_2 \Longrightarrow \begin{cases} s = 1 \\ e = 6 \\ m = 1001011\underbrace{0\cdots\cdots0}_{16\ z\acute{e}ros} \end{cases}$$

$E = e + 127 = 6 + 127 = 133 = (10000101)_2$

D'où, la forme $(\,?\,)_{2IEEE}$, puis $(\,?\,)_{16IEEE}$

$$(-101{,}5)_{10} = (N)_{2IEEE} = \Big(\underbrace{1}_{s}\ \underbrace{10000101}_{E}\ \underbrace{10010110000000000000000}_{m}\Big)_{2IEEE}$$

$$(-101{,}5)_{10} = (N)_{2IEEE} = \Big(\underline{1100}\ \underline{0010}\ \underline{1100}\ \underline{1011}\ \underline{0000}\ \underline{0000}\ \underline{0000}\ \underline{0000}\Big)_{2IEEE}$$
$$= (C2CB0000)_{16IEEE}$$

Pour $(47800180)_{16IEEE} = (\,?\,)_{10}$, on convertit d'abord $(47800180)_{16IEEE}$ en $(\,?\,)_2$, puis on met le résultat sous la forme $(\,?\,)_{2IEEE}$, qui permet d'identifier tous les paramètres de la forme donnée par la relation (1.5). On écrit le résultat sous la forme (1.5), ensuite on passe à la forme demandée $(\,?\,)_{10}$

$$(47800180)_{16IEEE} = (\,0100\ 0111\ 1000\ 0000\ 0000\ 0001\ 1000\ 0000)_2 =$$
$$\Big(\underbrace{0}_{s}\ \underbrace{100\ 01111}_{E}\ \underbrace{000\ 0000\ 0000\ 0001\ 1000\ 0000}_{m}\Big)_{2IEEE}$$

$$\Longrightarrow \begin{cases} s = 0 \\ E = (100\ 01111)_2 = (143)_{10} \end{cases} \Longrightarrow e = E - 127 = 16$$

D'où :

$$(47800180)_{16IEEE} = V = (-1)^0 \times 2^{16} \times \underbrace{\Big(1,\ \underline{000\ 0000\ 0000\ 0001\ 1000\ 0000}\Big)_2}_{\text{Équivalent décimal}}$$

$$V = (-1)^0 \times 2^{16}(1 + 1 \times 2^{-15} + 1 \times 2^{-16}) = 2^{16} + 2^1 + 2^0 = 65536 + 2 + 1 = 65539$$

Ou aussi par décalage de la virgule de 16 positions vers la droite, on obtient :

$$V = (1000\ 0000\ 0000\ 0001\ 1)_2 = (1 \times 2^{16} + 1 \times 2^1 + 1 \times 2^0)_{10} = 65539$$

1.2. Arithmétique binaire

Les 4 opérations sur les nombres binaires non signés sont :

Addition :

$0+0=0$	
$0+1=1$	
$1+0=1$	
$1+1=0$	et la retenue $=1$

Soustraction :

$0-0=0$	
$0-1=1$	et la retenue $=1$
$1-0=1$	
$1-1=0$	

Pour ces deux opérations « addition et soustraction », les retenues doivent être propagées, comme en décimal.

Multiplication :

$0\times0=0$
$0\times1=0$
$1\times0=0$
$1\times1=1$

Division :

$0\div0$	non défini
$0\div1=0$	
$1\div0$	non défini
$1\div1=1$	

Les opérations de multiplication et de division se font de la même façon qu'en décimal.

Exemple : $10101\times1001=?$

1 0 1 0 1 × 1 0 0 1	21 × 9
1 0 1 0 1 0 0 0 0 0 + 0 0 0 0 0 1 0 1 0 1	
1 0 1 1 1 1 0 1	189

Exemple : $10110100{,}1 \div 100110 = ?$

$1\,0\,1\,1\,0\,1\,0\,0{,}1\,0$ $\div$ $-\,1\,0\,0\,1\,1\,0$	$1\,0\,0\,1\,1\,0$	$180{,}5 \div 38$
$0\,0\,0\,1\,1\,1\,0\,0\,1$ $-\quad 1\,0\,0\,1\,1\,0$	$1\,0\,0{,}1\,1$	$= 4{,}75$
$0\,1\,0\,0\,1\,1\,0$ $-\quad 1\,0\,0\,1\,1\,0$		
$0\,0\,0\,0\,0\,0$		

1.3. Fonctions logiques binaires de base

Nous donnons ci-après la table de vérité, l'interprétation électrique et le masque binaire utilisé dans les microprocesseurs, de trois fonctions logiques de deux variables : ET (AND), OU (OR) et $OU_{ex}(XOR)$.
Ensuite, nous introduisons la fonction logique NON (NOT) et les opérations de décalage sur les mots binaires.

Fonction *ET* (AND)

La table de vérité de la fonction logique ET de deux variables est donnée par le tableau 1.1.

x_1	x_0		$z = x_1 \times x_0$
0	0		0
0	1		0
1	0		0
1	1		1

Tableau 1.1. Table de vérité de la fonction *ET*

L'interprétation électrique de cette fonction ET de deux variables est donnée par la figure 1.1.

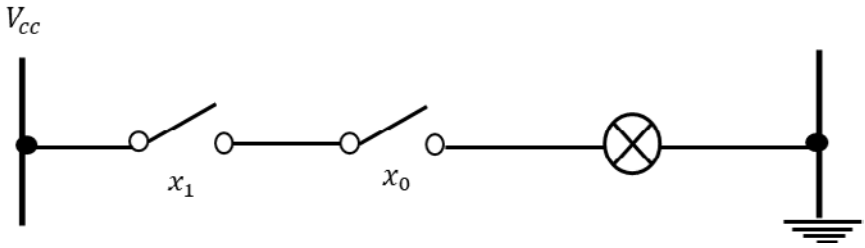


Figure 1.1. Interprétation électrique de la fonction *ET* de deux variables

Si l'un des interrupteurs x_1, ou x_0 est ouvert, la lampe est éteinte, elle s'allume lorsque les deux interrupteurs x_1 et x_0 sont fermés.

Pour forcer sélectivement des bits à zéro dans un mot binaire, il suffit d'utiliser un mot binaire masque ET, comme montré sur la figure 1.2, sur un mot de huit bits.

$x_7\ x_6\ x_5\ x_4\ x_3\ x_2\ x_1\ x_0$ $1\ \ 0\ \ 1\ \ 0\ \ 1\ \ 1\ \ 1\ \ 1$	← Mot binaire ← Mot binaire masque ET
$x_7\ 0\ x_5\ 0\ x_3\ x_2\ x_1\ x_0$	← Résultat

Figure 1.2. Mot binaire masque ET

Fonction OU (OR)

La table de vérité de la fonction logique OU de deux variables est donnée par le tableau 1.2.

x_1	x_0		$z = x_1 + x_0$
0	0		0
0	1		1
1	0		1
1	1		1

Tableau 1.2. Table de vérité de la fonction OU

L'interprétation électrique de la fonction OU de deux variables est donnée par la figure 1.3.

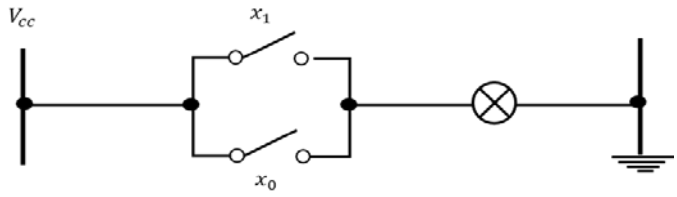


Figure 1.3. Interprétation électrique de la fonction OU de deux variables

Si les deux interrupteurs x_1 et x_0 sont ouverts, la lampe est éteinte, elle s'allume lorsqu'un des deux interrupteurs x_1 ou x_0 est fermé.

Pour forcer sélectivement des bits à un dans un mot binaire, il suffit d'utiliser un mot binaire masque OU, comme montré sur la figure 1.4, sur un mot de huit bits.

$x_7\ x_6\ x_5\ x_4\ x_3\ x_2\ x_1\ x_0$ $0\ \ 0\ \ 1\ \ 0\ \ 0\ \ 1\ \ 0\ \ 0$	← Mot binaire ← Mot binaire masque OU
$x_7\ x_6\ 1\ x_4\ x_3\ 1\ x_1\ x_0$	← Résultat

Figure 1.4. Mot binaire masque OU

Fonction OU_{ex}(XOR)

La table de vérité de la fonction logique OU_{ex} de deux variables (fonction différence) est donnée par le tableau 1.3.

x_1	x_0		$z = x_1 \oplus x_0$
0	0		0
0	1		1
1	0		1
1	1		0

Tableau 1.3. Table de vérité de la fonction OU_{ex}

L'interprétation électrique de la fonction OU_{ex} de deux variables est donnée par la figure 1.5.

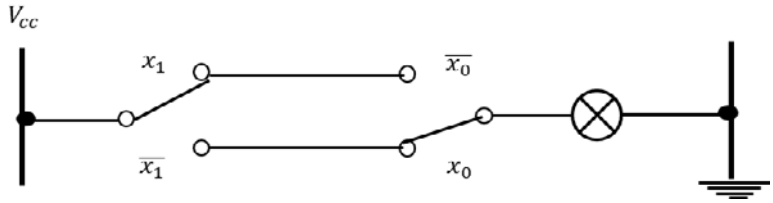


Figure 1.5. Interprétation électrique de la fonction OU_{ex} de deux variables

Si les deux interrupteurs x_1 et x_0 ont la même position fermée (x_1, x_0), ou ouvert $(\bar{x}_1, \bar{x}_0)$, la lampe est éteinte, elle s'allume lorsque les deux interrupteurs x_1 et x_0 ont des positions différentes $(\bar{x}_1, x_0)$, ou $(x_1, \bar{x}_0)$. Pour inverser sélectivement des bits dans un mot binaire, il suffit d'utiliser un mot binaire masque OU_{ex}, comme montré sur la figure 1.6, sur un mot de huit bits.

$x_7\ x_6\ x_5\ x_4\ x_3\ x_2\ x_1\ x_0$ $0\ \ 1\ \ 0\ \ 0\ \ 0\ \ 1\ \ 0\ \ 0$	← Mot binaire ← Mot binaire masque OU_{ex}
$x_7\ \bar{x}_6\ x_5\ x_4\ x_3\ \bar{x}_2\ x_1\ x_0$	← Résultat

Figure 1.6. Mot binaire masque OU_{ex}

Fonction *NON* (*NOT*)

La fonction NON d'une variable binaire est définie par : $NON(x) = \bar{x}$

Décalage binaire

Nous introduisons ci-dessous quatre opérations de décalage binaire sur un mot de huit bits, soit : $(N)_2 = (0\,0\,1\,0\,1\,1\,1\,0)_2 = (46)_{10}$

- **Décalage ouvert à droite** : $introduction\ 0 \rightarrow 0\,0\,1\,0\,1\,1\,1\,0 \rightarrow 0\ perdu \Longrightarrow (0\,0\,0\,1\,0\,1\,1\,1)_2 = (23)_{10}$

  Ce type de décalage a pour résultat de diviser le nombre binaire $(N)_2$ par la base $b = 2$.

  Si l'équivalent décimal du nombre $(N)_2$ est impair : $(N)_2 = (47)_{10}$ par exemple, alors le résultat de ce décalage sera $((N)_2 - 1)/2$.

- **Décalage ouvert à gauche** : $perdu\ 0 \leftarrow 0\ 0\ 1\ 0\ 1\ 1\ 1\ 0 \leftarrow 0\ introduction \Longrightarrow (0\ 1\ 0\ 1\ 1\ 1\ 0\ 0)_2 = (92)_{10}$. Ce type de décalage a pour résultat de multiplier le nombre binaire $(N)_2$ par la base $b = 2$.
- **Décalage cyclique à droite** : $0\ 0\ 1\ 0\ 1\ 1\ 1\ 0 \Longrightarrow 0\ 0\ 0\ 1\ 0\ 1\ 1\ 1$: le bit LSB sortant « 0 » est réinjecté de l'autre côté en tant que bit MSB entrant.
- **Décalage cyclique à gauche** : $0\ 0\ 1\ 0\ 1\ 1\ 1\ 0 \Longrightarrow 0\ 1\ 0\ 1\ 1\ 1\ 0\ 0$: le bit MSB sortant « 0 » est réinjecté de l'autre côté en tant que bit LSB entrant.

1.4. Représentation des nombres binaires signés (entiers relatifs)

1.4.1. Représentation sous forme module plus signe

Dans le cas d'une écriture sur n bits, nous avons :

$$(N)_2 = a_{n-1}2^{n-1} + a_{n-2}2^{n-2} + \cdots\cdots + a_1 2^1 + a_0 2^0 \quad (1.6)$$

Où :

a_{n-1} (le bit de poids fort) est le bit de signe : $s = 0$, si le nombre est positif, et $s = 1$, si le nombre est négatif.

$a_{n-2}\ \cdots\cdots\ a_1\ a_0$: représente le module exprimé en binaire naturel.

La plage de valeurs décimales codables sur n bits est :

$$-(2^{n-1} - 1) \leq (N)_2 \leq (2^{n-1} - 1) \quad (1.7)$$

Exemple : $n = 4$, d'où : $-(2^3 - 1) \leq (N)_2 \leq (2^3 - 1) \Longrightarrow -7 \leq N \leq 7$

Pour cet exemple, le tableau 1.4, montre le résultat des valeurs décimales codables.

Nombres positifs				Nombres négatifs		
s	$Module$	EqD		s	$Module$	EqD
0	0 0 0	+0		1	0 0 0	−0
0	0 0 1	+1		1	0 0 1	−1
0	0 1 0	+2		1	0 1 0	−2
0	0 1 1	+3		1	0 1 1	−3
0	1 0 0	+4		1	1 0 0	−4
0	1 0 1	+5		1	1 0 1	−5
0	1 1 0	+6		1	1 1 0	−6
0	1 1 1	+7		1	1 1 1	−7

Tableau 1.4. Valeurs décimales représentables en binaire signé sur $n = 4$ bits en format module plus signe

L'avantage de cette représentation en module plus signe est sa simplicité, son inconvénient est double :

- la valeur 0 a deux représentations différentes $+0 \rightarrow (0\ 0\ 0\ 0)$ et $-0 \rightarrow (1\ 0\ 0\ 0)$;
- le calcul arithmétique est impossible.

Exemple :

$(+1) + (-1) = 1\ 0\ 1\ 0 = -2$?

$(+2) + (-2) = 1\ 1\ 0\ 0 = -4$?
$(+3) + (-3) = 1\ 1\ 1\ 0 = -6$?

À cause de cette faiblesse, cette représentation n'est pas utilisée par les machines informatiques (calculateurs, ordinateurs, microprocesseurs, etc.).

1.4.2. Représentation sous forme de complément à deux $C2(N)_2$

Par définition, le complément vrai d'un nombre $(N)_b$ exprimé par n symboles (chiffres) dans la base b est :

$$CV(N)_b = b^n - (N)_b = -(N)_b \quad (1.8)$$

En effet, $b^n = 0$, car il est hors format sur n bits

$(a_{n-1}b^{n-1} + a_{n-2}b^{n-2} + \cdots\cdots + a_1b^1 + a_0b^0)$, (interprétation calculateur).

En base binaire $b = 2$, la relation (1.8) s'écrit :

$$C2(N)_2 = 2^n - (N)_2 = -(N)_2 \quad (1.9)$$

En pratique, le calcul du complément à deux d'un nombre est effectué plus simplement que la relation (1.9), en utilisant le complément à « 1 » d'un nombre binaire en base $b = 2$, $C1(N)_2$. En effet, nous montrons ci-dessous le résultat suivant :

$$C2(N)_2 = C1(N)_2 + 1 = -(N)_2 \quad (1.10)$$

Le complément à « 1 » d'un nombre binaire, $C1(N)_2$ est obtenu en complémentant tous les bits constituant le nombre binaire, donc : $C1(N)_2 = (\bar{N})_2$.

L'addition $(N)_2 + (\bar{N})_2$ donne le résultat suivant :

$(N)_2$	→	$a_{n-1}\ a_{n-2}\ \cdots\cdots\ a_0$
$+\ (\bar{N})_2$	→	$\bar{a}_{n-1}\ \bar{a}_{n-2}\ \cdots\cdots\ \bar{a}_0$
$2^n - 1 = -1$	=	$1\quad 1\quad \cdots\cdots 1$

D'où : $(N)_2 + C1(N)_2 = 2^n - 1 = -1 \implies -(N)_2 = C1(N)_2 + 1 = C2(N)_2$

La plage de valeurs décimales codables sur n bits en $C2(N)_2$ est :

$$-2^{n-1} \leq (N)_2 \leq 2^{n-1} - 1 \quad (1.11)$$

Cette plage est dissymétrique à cause du zéro qui n'apparaît que parmi les nombres positifs (codage unique du zéro)

Exemple : $n = 4$, d'où : $-2^3 \leq (N)_2 \leq 2^3 - 1 \implies -8 \leq N \leq 7$

Pour cet exemple, le tableau 1.5, montre le résultat des valeurs décimales codables.

Nombres positifs				$C1(N)_2$		Nombres négatifs $C2(N)_2 = C1(N)_2 + 1 = -(N)_2$		
s		EqD				*s*		EqD
0	**0 0 0**	**+0**		1 1 1 1		**0**	**0 0 0**	**+0**
0	0 0 1	+1		1 1 1 0		1	1 1 1	−1
0	0 1 0	+2		1 1 0 1		1	1 1 0	−2
0	0 1 1	+3		1 1 0 0		1	1 0 1	−3
0	1 0 0	+4		1 0 1 1		1	1 0 0	−4

0	1 0 1	+5		1 0 1 0		1	0 1 1	−5
0	1 1 0	+6		1 0 0 1		1	0 1 0	−6
0	1 1 1	+7		1 0 0 0		1	0 0 1	−7

Tableau 1.5. Valeurs décimales représentables en binaire signé sur $n = 4$ bits en format complément à deux

On note que le code 1000 n'apparaît pas. C'est normal puisque ce codage correspond à la valeur -8 dont l'opposé $+8$ ne peut s'écrire sur 4 bits en format $C2(N)_2$.

Le code 1000 est cependant tout à fait valide. On peut s'en convaincre en effectuant une opération (sans débordement, faute de quoi le résultat ne pourrait être que faux) qui l'utilise, comme dans l'exemple ci-dessous :

1 0 0 0	$-(8)$
0 1 0 1	$+(5)$
1 1 0 1	$= -(3)$ en $C2(N)_2$

L'avantage de la représentation des nombres entiers en complément à deux par rapport à la représentation module plus signe est :

- le zéro n'a qu'une seule représentation « **0 0 0 0** » quel que soit le signe du zéro ;
- le calcul arithmétique se fait correctement. Si on fait l'addition de $+3$ avec -3 le résultat est 0 :

3	0 0 1 1
+	
−3	1 1 0 1
carry in →	1 1 1
carry out → 1	0 0 0 0 = 0 OK

Remarque importante : dans les machines informatiques, l'opération de soustraction de deux nombres binaires est remplacée par l'opération d'addition du premier nombre avec le complément à deux du second nombre :

$$N = N_1 - N_2 = N_1 + C2(N_2) \tag{1.12}$$

Règle : Le résultat de l'addition de deux nombres, complément à deux de signes opposés est toujours correct. Par contre le résultat de l'addition de deux nombres, complément à deux de même signe, peut être correct ou non, cela dépend si le résultat présente un débordement (overflow) ou non.

Pour savoir si le résultat est correct ou non, il suffit d'opérer un XOR logique entre le bit de retenue sur le bit de signe « *carry in into the bit of sign* » et le bit retenu sortant du bit de signe « *carry out going out of the bit sign* ». Le résultat V du XOR indique s'il y débordement ou non, comme montré ci-dessous.

$V = XOR(carry\ in, carry\ out\ )$

$$\Longrightarrow \text{Si} \begin{cases} V = 1 & \Longrightarrow KO, \text{ débordement } (overflow) \\ V = 0 & \Longrightarrow OK, \text{ pas de débordement } (no\ overflow) \end{cases} \quad (1.13)$$

Exemple :
Si on fait l'addition de -6 avec -5 le résultat est -11, et on aura un débordement

-5	1 0 1 1
+	
-6	1 0 1 0
carry in →	0 1
carry out → 1	0 1 0 1 = 5 *KO* car $V = 1$

Si on fait l'addition de -4 avec -3 le résultat est -7, et on n'aura pas un débordement

-4	1 1 0 0
+	
-3	1 1 0 1
carry in →	1
carry out → 1	1 0 0 1 = -7 *OK* car $V = 0$

Ici, le *carry out* est ignoré car le résultat est correct.

1.5. Codage de l'information

Définition générale : Le codage de l'information est une opération de correspondance biunivoque entre les éléments de deux ensembles, comme montre la figure 1.7.

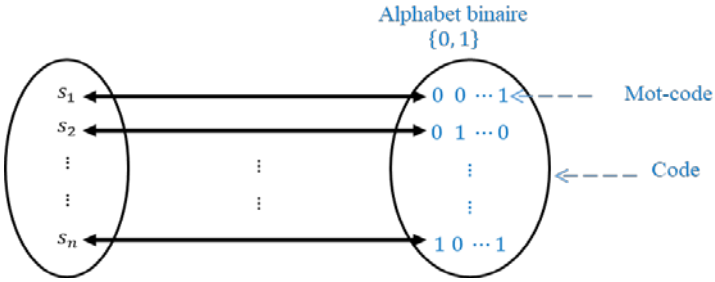


Figure 1.7. Codage de l'information

Le but de cette opération dépend du domaine d'utilisation :

En télécommunications, nous avons :

- le codage de source pour canaux sans perturbations permet d'adapter le débit binaire de la source à la capacité du canal de transmission ;
- le codage de canal pour canaux à perturbations sert d'une part à se protéger contre les erreurs de transmission par l'introduction d'une redondance voulue, et d'autre part à faire correspondre un signal électrique (à deux niveaux 0 et 5 V par exemple) à chacun des bits (symboles) d'information ;
- le codage peut être aussi utilisé dans une optique de chiffrement de l'information.

En électronique numérique, l'opération de codage sert à réduire le nombre des variables à traiter. Par exemple, avec 8 bits seulement on peut coder l'ensemble des touches du clavier d'un ordinateur.

1.5.1. Codes couramment utilisés en électronique numérique

Nous donnons ci-après les différents codes couramment utilisés en électronique numérique.

1.5.1.1. Codes décimaux utilisés pour la représentation des chiffres du système décimal

Pour représenter les chiffres décimaux de zéro à neuf, il suffit d'utiliser un mot binaire de quatre bits. Trois codes sont généralement utilisés : le code DCB (Décimal Codé Binaire), (BCD : *Binary Coded Decimal*), le code DCB+3, (DCB excess3) et le code Aïken. Ce sont des codes pondérés, comme montré par la table de vérité du tableau 1.6.

$(N)_{10}$	DCB 8 4 2 1	DCB+3 8 4 2 1	Aïken 2 4 2 1
0	0 0 0 0	0 0 1 1	0 0 0 0
1	0 0 0 1	0 1 0 0	0 0 0 1
2	0 0 1 0	0 1 0 1	0 0 1 0
3	0 0 1 1	0 1 1 0	0 0 1 1
4	0 1 0 0	0 1 1 1	0 1 0 0
5	0 1 0 1	1 0 0 0	1 0 1 1
6	0 1 1 0	1 0 0 1	1 1 0 0
7	0 1 1 1	1 0 1 0	1 1 0 1
8	1 0 0 0	1 0 1 1	1 1 1 0
9	1 0 0 1	1 1 0 0	1 1 1 1

Tableau 1.6. Codes décimaux généralement utilisés

Les codes DCB+3 et Aïken sont des codes auto-complémentés, cela veut dire :

- l'axe médian est un axe de symétrie, donc de part et d'autre de cet axe les mots-codes sont complémentés ;
- le complément à 1 d'un mot-code représente le complément à 9 dans l'ensemble source.

Exemple : en DCB+3, le $C1(6) = C1(1001) = 0110 = 3 = C9(6)$.

1.5.1.2. Code de Gray (code à distance unité) ou code binaire réfléchi

Le code de Gray est un code continu (les combinaisons consécutives sont adjacentes) et strict (la première combinaison et la dernière sont aussi adjacentes). La figure 1.8 montre comment construire un code de Gray à $(n+1)$ variables binaires à partir de code d'un Gray à n variables binaires.

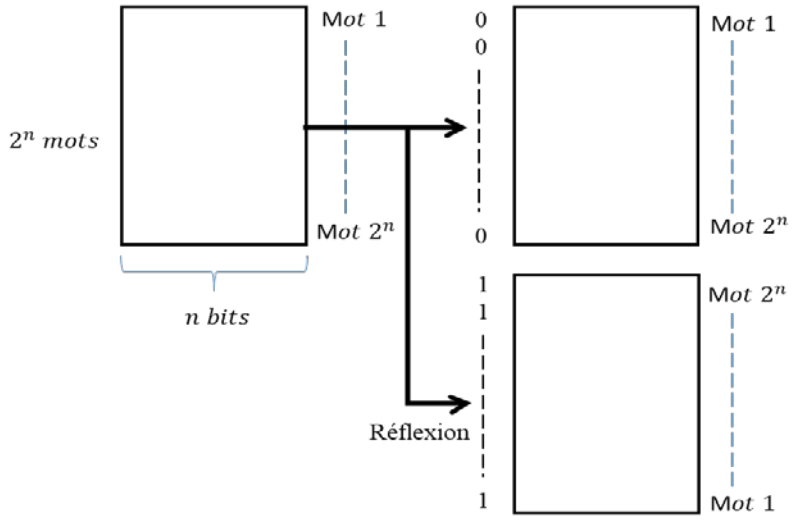


Figure 1.8. Construction du code de Gray à $(n + 1)$ variables binaires à partir d'un code de Gray à n variables binaires.

Sur la figure 1.9, nous montrons la construction du code de Gray à 2, 3 et 4 variables binaires par étapes successives à partir du code élémentaire à une variable binaire. Ainsi nous pouvons construire un code de Gray pour un nombre de variables binaires donné.

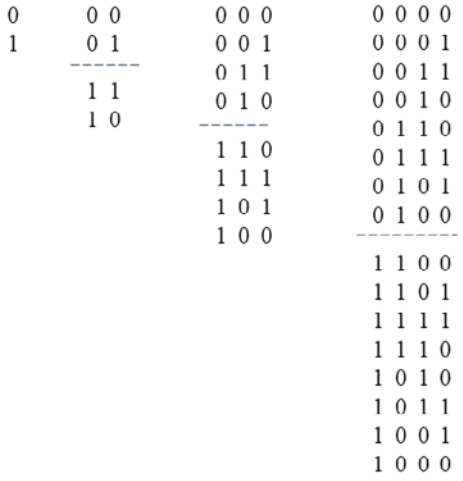


Figure 1.9. Construction du code de Gray à 2, 3 et 4 variables binaires par étapes successives

Remarque importante : intérêt de l'adjacence des codes

Avec un code non adjacent, le passage d'une combinaison à la suivante peut entraîner la commutation de plusieurs bits (sorties d'un compteur binaire par exemple). Par exemple, le passage de la valeur 7 à la valeur 8 à la sortie d'un compteur 4 bits, entraîne la commutation de tous les bits car on passe de la combinaison 0111 à la combinaison 1000.

La commutation des bits n'est pas instantanée, d'où possibilité de passage par des états intermédiaires :
$0111 \rightarrow 0110 \rightarrow 0100 \rightarrow 0000 \rightarrow 1000$.

Ce phénomène porte le nom d'aléas de commutation (rien à voir avec un « parasite »).

La séquence ci-dessus est donnée à titre d'exemple, mais d'autres sont tout aussi envisageables, par exemple : $0111 \rightarrow 0101 \rightarrow 1101 \rightarrow 1100 \rightarrow 1000$.

Conséquences éventuelles

- suivant l'instant d'observation, le résultat change ;
- si un circuit, placé en aval, décode la valeur 0110 (ou 0100, ou 0000) il peut se produire une détection erronée de cette valeur de manière transitoire mais potentiellement source de dysfonctionnement.

Avec le code Gray, ce problème disparaît naturellement puisque seulement 1 bit change d'état entre 2 combinaisons successives.

1.5.1.3. Code ASCII : *American Standard Code for Information Interchange*

A l'origine, ce code utilisant 7 bits associe une valeur numérique à chacun des caractères (latins) alphanumériques affichables et/ou imprimables. Une version étendue sur 8 bits est apparue ensuite afin de coder les caractères accentués et bon nombre de caractères dits « spéciaux » (caractères de contrôle, de mise en forme, etc.).

Le code ASCII est utilisé dans les réseaux informatiques assurant les connexions entre des ordinateurs et des organes périphériques (claviers, écrans, imprimantes, etc.). Le tableau 1.7, donne la table du code ASCII 7 bits.

b_7 b_6 b_5 → / Bits					000	001	010	011	100	101	110	111
b_4	b_3	b_2	b_1	Column → / Row ↓	0	1	2	3	4	5	6	7
0	0	0	0	0	NUL	DLE	SP	0	@	P	`	p
0	0	0	1	1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	2	STX	DC2	"	2	B	R	b	r
0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
0	1	0	0	4	EOT	DC4	$	4	D	T	d	t
0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	6	ACK	SYN	&	6	F	V	f	v
0	1	1	1	7	BEL	ETB	'	7	G	W	g	w
1	0	0	0	8	BS	CAN	(	8	H	X	h	x
1	0	0	1	9	HT	EM	)	9	I	Y	i	y
1	0	1	0	10	LF	SUB	*	:	J	Z	j	z
1	0	1	1	11	VT	ESC	+	;	K	[	k	{
1	1	0	0	12	FF	FS	,	<	L	\	l	|
1	1	0	1	13	CR	GS	—	=	M	]	m	}
1	1	1	0	14	SO	RS	.	>	N	^	n	~
1	1	1	1	15	SI	US	/	?	O	_	o	DEL

Tableau 1.7. Table du code ASCII 7 bits