PolyPublie
Polytechnique Montréal

POLYTECHNIQUE MONTRÉAL
UNIVERSITÉ D'INGÉNIERIE

Titre: Title:	Conception d'une plateforme d'Animats destinée à l'étude d'aglgorithmes d'apprentissage appliqués à la survie en environnement réel
Auteur: Author:	Jean-François Pons
Date:	2012
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Pons, J.-F. (2012). Conception d'une plateforme d'Animats destinée à l'étude d'aglgorithmes d'apprentissage appliqués à la survie en environnement réel [Master's thesis, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/878/

Document en libre accès dans PolyPublie
Open Access document in PolyPublie

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/878/
Directeurs de recherche: Advisors:	Jean-Jules Brault, & Yvon Savaria
Programme: Program:	génie électrique

Ce fichier a été téléchargé à partir de PolyPublie, le dépôt institutionnel de Polytechnique Montréal
This file has been downloaded from PolyPublie, the institutional repository of Polytechnique Montréal

https://publications.polymtl.ca

UNIVERSITÉ DE MONTRÉAL

CONCEPTION D'UNE PLATEFORME D'*ANIMATS* DESTINÉE À L'ÉTUDE D'ALGORITHMES D'APPRENTISSAGE APPLIQUÉS À LA SURVIE EN ENVIRONNEMENT RÉEL

JEAN-FRANÇOIS PONS

DÉPARTEMENT DE GÉNIE ÉLECTRIQUE

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION

DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES

(GÉNIE ÉLECTRIQUE)

JUIN 2012

© Jean-François Pons, 2012.

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé:

CONCEPTION D'UNE PLATEFORME D'*ANIMATS* DESTINÉE À L'ÉTUDE D'ALGORITHMES D'APPRENTISSAGE APPLIQUÉS À LA SURVIE EN ENVIRONNEMENT RÉEL

présenté par : PONS Jean-François

en vue de l'obtention du diplôme de : Maîtrise ès Sciences Appliquées

a été dûment accepté par le jury d'examen constitué de :

M. DAVID Jean-Pierre, Ph.D., président

M. BRAULT Jean-Jules, Ph.D., membre et directeur de recherche

M. SAVARIA Yvon, Ph.D., membre et codirecteur de recherche

M. FRIGON Jean-François, Ph.D., membre

DÉDICACE

A mia Nonna, A mia Mamma

REMERCIEMENTS

Mes remerciements s'adressent à l'ensemble des personnes qui m'ont, parfois sans le savoir, permis de mener à bien cette maîtrise. Tout d'abord, j'aimerais remercier mon directeur de recherche, le professeur Brault, pour m'avoir écouté et m'avoir accordé son temps lorsque j'en avais besoin. Je le remercie aussi pour m'avoir fait confiance en m'accordant des charges de répétition. J'aimerais aussi remercier mon co-directeur de recherche, le professeur Savaria, pour son exceptionnelle disponibilité malgré un emploi du temps chargé, ainsi que son aide aussi bien intellectuelle que matérielle. Ils m'ont permis, à partir d'une idée floue, de parvenir à un projet de recherche intéressant.

J'aimerais adresser mes remerciements aux professeurs Jean-Pierre David et Jean-François Frigon pour leur participation au jury en charge de l'évaluation de la pertinence et de la qualité de ma recherche.

Par ailleurs, j'aimerais remercier toutes les personnes qui m'ont donné des conseils ou qui m'ont permis d'accéder à des moyens techniques. Je pense notamment à Normand Bélanger qui m'a donné de précieux conseils concernant les FPGA, aux techniciens Maxime Thibault qui a toujours pris le temps de répondre à mes requêtes même s'il n'y était pas tenu et Jaques Girardin pour ses conseils et sa disponibilité. Je remercie plus spécialement Bryan Tremblay qui m'a vraiment aidé sur le plan technique dans les phases de prototypage et qui m'a appris beaucoup plus qu'il ne le croit et avec qui, sur le plan humain, j'ai eu beaucoup de plaisir à passer du temps.

J'ai une pensée pour les étudiants avec lesquels j'ai partagé mon quotidien, que ce soit dans le laboratoire, je pense notamment à « Papy », ou lors des charges de laboratoire. Je remercie à ce titre Édouard Martin-Haas avec qui j'ai beaucoup apprécié de travailler.

Enfin, je remercie ma famille notamment mes frères et sœurs qui me soutiennent malgré la distance et à qui je pense constamment. Je remercie aussi ma mère qui est pour moi un exemple en tout point et qui m'a toujours compris. Je remercie bien évidemment Vanessa qui a bien voulu m'accompagner pendant ces deux dernières années, sans qui cette expérience aurait certainement été plus difficile.

RÉSUMÉ

Le présent mémoire résume les travaux de maîtrise réalisés dans l'optique de proposer une plateforme de déploiement et d'analyse d'algorithmes d'apprentissage appliqués aux *animats*. Les *animats* désignent des agents minimalement équipés aussi bien sur le plan mécanique qu'en termes de puissance de calcul. Néanmoins, la théorie gravitant autour de l'étude de ces entités vise à mettre en avant des méthodes d'apprentissages permettant à ceux-ci d'évoluer à partir de ce qu'ils auront appris de leur expérience dans l'environnement. À l'image des êtres vivants, les différences de perception du milieu environnant devraient permettre de voir naître des particularités et des comportements différents pour chacun d'entre eux. Par ailleurs, ces *animats* devraient avoir la faculté de pouvoir communiquer entre eux dans le but de permettre, s'ils le désirent, de pouvoir partager de l'information et de se servir d'une connaissance et d'un apprentissage communs pour assouvir ce qui devrait être leur principal objectif : survivre dans leur environnement.

Le domaine de recherche des *animats* comprend aussi bien l'étude des *animats* simulés que celle des *animats* réels. Alors que le premier permet un déploiement facilité par l'abstraction rendue possible par les langages de haut niveau, la nécessité de devoir simuler un environnement avec l'ensemble de ses singularités induit une erreur de modélisation qui n'existe pas lorsque les *animats* sont physiquement réalisés. Par conséquent, si une telle plateforme était disponible, il serait alors seulement nécessaire de se concentrer sur les algorithmes d'apprentissage plutôt que sur des problèmes de modélisation. Par ailleurs, bien souvent, des algorithmes puissants lors des simulations se retrouvent inadaptés pour des problèmes réels de par le manque de fidélité entre l'environnement simulé et réel. Notamment, qu'adviendrait-il si un capteur devenait défectueux? Si une situation inconnue était rencontrée? Si le bruit ou la précision des capteurs avait mal été modélisés?

Dans ce contexte, nous proposons d'étudier et de concevoir ce qui sera la base d'une plateforme de développement et d'analyse d'algorithmes d'apprentissage. À la différence des plateformes existantes, la principale originalité de cette plateforme réside dans la prise en compte des problèmes énergétiques. En effet, pour pouvoir survivre l'*animat* devrait être capable d'estimer et de prévoir ses dépenses énergétiques présentes et futures et de les prendre en considération dans le choix des tâches à effectuer. Par ailleurs, l'objectif était de concevoir un *animat* suffisamment

équipé pour réaliser des tâches simples mais relativement primitif pour profiter de ses réelles limitations afin de mettre en place des parades algorithmiques comme la mise en commun d'information. De ce fait, la puissance de calcul embarquée est réduite mais l'*animat* devrait pouvoir, dépendamment de l'application, délocaliser les calculs. D'autre part, une taille réduite et un prix raisonnable constitueront à la fois des objectifs nécessaires à la mise en place de colonies d'*animats* et nous permettront de nous démarquer de ce qui se trouve dans la littérature.

Par conséquent, nous proposons ici de présenter une plateforme destinée à l'étude des algorithmes d'apprentissages dans des colonies d'*animats*. Nous détaillons l'ensemble des choix de conception réalisés pour atteindre les objectifs de l'application, notamment ceux concernant l'efficacité énergétique. Dans cette optique, nous traitons plus particulièrement des problèmes énergétiques liés à l'utilisation des moyens de communications. La prise en compte de cette problématique nous amène à réfléchir puis à proposer un circuit de réveil de la radio principale réalisé en logique asynchrone. L'utilisation même de ce type de logique est discutée, puis une amélioration d'une technique spécifique de conception est proposée puis analysée. Enfin, nous présentons une plateforme et détaillons les choix réalisés dans le but de répondre aux objectifs et à l'application visés.

En plus, de l'utilisation susmentionnée, cette plateforme peut trouver une utilité dans différentes disciplines du génie et à différents niveaux. Tout d'abord, de par sa faible complexité et son prix modéré celle-ci peut aisément constituer une plateforme d'introduction aux disciplines du génie aussi bien sous l'angle de l'électronique, de l'informatique, de la robotique que des télécommunications et des problèmes énergétiques. D'autre part, cette plateforme, bien qu'étant suffisamment avancée pour pouvoir implémenter des algorithmes, n'a pas la prétention de se décrire comme immuable. Au contraire, plusieurs points peuvent être améliorés et ce à plusieurs niveaux. De ce fait, cette plateforme peut aussi constituer un sujet intéressant pour des projets intégrateurs. Enfin et surtout, cette plateforme trouvera parfaitement sa place en recherche et plus spécifiquement en apprentissage machine, permettant de se différencier des autres contributions par l'utilisation de données réelles d'environnements non-simulés. Les résultats alors espérés pourront servir de nombreux domaines comme la robotique, la biologie, la gestion des flux routiers, les jeux-vidéo…

ABSTRACT

This master's thesis summarizes the achievements realized towards proposing a platform for deployment and analysis of machine learning algorithms applied to animats. Animats are agents minimally equipped on the mechanical plan as well as in terms of computing power. Nevertheless, the theory describing the study of these entities aims at discovering learning methods allowing them to evolve using what they learnt from their experience in the environment. Just like human beings, differences in perception of the surrounding environment should allow peculiarities and several behaviors for each of them. Besides, these animats should be able to communicate between them so as to allow, if needed, to share information and to use common knowledge and learning in order to succeed in what should be their main objective: survive in their environment.

The research field of animats includes the study of simulated and real animats. While the former allows a facilitated deployment due to the abstraction made possible by high-level languages, the necessity to emulate an environment with all peculiarities can lead to modeling errors that may be avoided when animats are physically built. Consequently, if such a platform was available, it would only be necessary to focus on the learning algorithms rather than on the modelling problems. Besides, very often, powerful algorithms tested on simulation may prove themselves unsuitable for real problems in real environments. For instance, it is hard to predict what would happen if a sensor became defective, if an unexpected situation was met, or if the noise or the precision of the sensors was not properly modeled.

In this context, we suggest studying and designing what will be the base of a platform for development and analysis of learning algorithms. Unlike the existing platforms, the main originality of this platform lies in the consideration of energy constraints. Indeed, to enhance its survivability, the animat should be able to estimate and to plan its present and future energy consumption and to consider it in the choices made. Besides, the objective was to design an animat sufficiently equipped to realize simple tasks but relatively primitive to make it necessary to take advantage of algorithmic parades, like for example the pooling of information. Therefore, the computing power embedded in the animat was purposely reduced to force the relocation of processing that proves too complex. On the other hand, a reduced size and a moderate price are necessary features to enable deployment of a colony of animats.

Consequently, we present here a platform that enables to study learning algorithms in colonies of animats. The various design choices made to handle the objectives of the application, especially energy efficiency, are exposed. As a step towards that goal, we also handle the energy related problems associated with the use of means of communication. The consideration of this problem brings us to propose a wake-up receiver of the main radio implemented using asynchronous logic. The use of this type of logic is discussed, and an improvement of a specific design technique is proposed and analyzed.

In addition to the objectives stated above, this platform can be useful in various engineering disciplines and at several levels. First of all, due to its low complexity and its moderate price, it can easily constitute an introductive platform to explore electronics, computing, robotics as telecommunications and energy problems related to the design of *animats* and wireless platform. On the other hand, although this platform is sufficiently complex to implement learning algorithms, we do not consider it definitive. On the contrary, several points can be improved at several levels. Therefore, this platform can also constitute an interesting base for team projects. Last but not least, this platform is very well suited for research and more specifically in the field of machine learning, allowing validation of a wide range of contributions using realistic experimental environments. The expected results can contribute to numerous research areas such as robotics, biology, management of traffic flows, video-games…

TABLE DES MATIÈRES

DÉDICACE III

REMERCIEMENTS IV

RÉSUMÉ V

ABSTRACT VII

TABLE DES MATIÈRES IX

LISTE DES TABLEAUX XIV

LISTE DES FIGURES XVI

LISTE DES SIGLES ET ABRÉVIATIONS XX

LISTE DES ANNEXES XXII

INTRODUCTION 1

CHAPITRE 1 PRÉSENTATION DU PROJET ET DÉFINITION DE L'APPLICATION GÉNÉRALE 4

1.1 Description de l'application « *animats* » 4

1.2 Problématique du projet 4

1.3 Exemples d'applications visées 5

1.4 Projets existants 6

1.5 Objectifs et présentation du projet réalisé 7

1.6 Réseaux de capteurs sans-fil et classes d'applications 9

1.7 Protocole de communication ZigBee 10

CHAPITRE 2 CONCEPTION DE L'ANIMAT 15

2.1 Architecture Générale et Justifications 15

2.1.1 Module Coordinateur 16

2.1.2 Animats 17

2.2 Cerveau....18
2.3 Communication et Réveil....20
2.4 Capteurs....21
2.4.1 Capteurs de courant et niveau de batterie....21
2.4.2 Capteurs de température....23
2.4.3 Capteurs de lumière....24
2.4.4 Détecteurs Infrarouges....24
2.4.5 Microphone....25
2.4.6 Accéléromètre....26
2.5 Déplacements....27
2.6 Énergie....29
2.7 Produit Final....32
CHAPITRE 3 LOGIQUE ASYNCHRONE ET NCL™....35
3.1 Généralités....35
3.1.1 Intérêts pour les circuits asynchrones....35
3.1.2 Avantages des circuits asynchrones....35
3.1.3 Inconvénients des circuits asynchrones....37
3.2 Types de design....37
3.2.1 Les circuits insensibles aux délais (DI)....38
3.2.2 Les circuits quasi-insensibles aux délais (QDI)....38
3.2.3 Les circuits indépendants de la vitesse (SI)....39
3.2.4 Les circuits à délais bornés....39
3.3 Choix du type de design....40
3.4 NULL Convention Logic....41

3.4.1 Présentation 41
3.4.2 Modules de base 44
3.4.3 Protocole de communication 50
3.4.4 Exemples de circuits 54
CHAPITRE 4 NULL CONVENTION LOGIC SIMPLIFIÉE : SHF-NCL 55
4.1 Motivations 55
4.1.1 Observations 55
4.1.2 Objectifs 56
4.1.3 Extrapolation ASIC – FPGA 57
4.2 Idée Générale 59
4.2.1 Principe de la simplification 59
4.2.2 Détails de la simplification 59
4.2.3 Modification du pipeline 60
4.2.4 Registre SHF-NCL 62
4.3 Validité de l'approche 63
4.3.1 Exemple : Cas du XOR non-optimal 63
4.3.2 Analyse des aléas 66
4.3.2.1 Différents types d'aléas susceptibles de survenir 66
4.3.2.2 Parades du NCL 69
4.3.3 Fan-out Reconvergeant 72
4.3.4 Validité de l'approche 79
4.4 Étude du Gain 83
4.4.1 Relevés sur différents exemples 84
4.4.1.1 Surplus 84

4.4.1.2 Porte OU-Exclusif 86
4.4.1.3 Additionneur 1-bit avec retenue 87
4.4.1.4 Circuit d'incrémentation 87
4.4.1.5 Unité Arithmétique et Logique 88
4.4.2 Mesures des performances 88
4.4.2.1 Circuit d'incrémentation 89
4.4.2.2 Unité Arithmétique et Logique 92
4.4.3 Gain Maximal Théorique et Limite de l'Approche 93
4.5 Conclusion 96
CHAPITRE 5 MODULE DE RÉVEIL ASYNCHRONE 98
5.1 Présentation Générale 98
5.1.1 Motivations 98
5.1.2 Fonctionnement Général 100
5.2 Présentation Générale 101
5.2.1 Signal de réveil 102
5.2.2 Démodulation 106
5.2.3 Sensibilité 110
5.3 Décodage Numérique 114
5.3.1 Consommation dans les réseaux de capteurs sans-fil 114
5.3.2 Module de réveil 114
5.3.3 Circuit proposé 115
5.3.3.1 Fonctionnement 115
5.3.3.2 Architecture Générale 116
5.4 Conception sur FPGA 119

5.4.1 Motivation du choix du FPGA et d'un FPGA en général 119
5.4.2 Caractéristiques et éléments de base 120
5.4.3 Design Asynchrone sur FPGA 121
5.4.4 Placement / Routage et Implémentation 125
5.4.5 Consommation 126
5.5 Alimentation Solaire 128
5.6 Discussion 129
5.6.1 Effets du SHF-NCL 129
5.6.2 Comparaison avec un circuit synchrone 130
CHAPITRE 6 EXPÉRIMENTATIONS 133
6.1 Environnement Graphique (GUI) et utilisation 133
6.2 Caractéristiques de l'animat 134
CHAPITRE 7 DISCUSSION 137
CONCLUSION 141
BIBLIOGRAPHIE 143
ANNEXES 147

LISTE DES TABLEAUX

Tableau 1.1. Principales caractéristiques des plateformes présentées 8

Tableau 1.2 Caractéristiques physiques du protocole ZigBee 12

Tableau 1.3. Format de la trame....14

Tableau 2.1. Principales Caractéristiques du PIC24FJ256GA108....20

Tableau 2.2. Caractéristiques de la batterie....29

Tableau 2.3. Consommation des grands ensembles de l'*animat* 29

Tableau 2.4. Estimation de l'autonomie pour différents cas d'utilisation sans recharge 30

Tableau 2.5. Prix des principaux composants....34

Tableau 3.1. Définitions des 27 portes du NCL....45

Tableau 4.1. Exemple d'évolution du prix et de la consommation statique en fonction du nombre de portes pour les FPGA 58

Tableau 4.2. Différents types d'aléas susceptibles de se produire....67

Tableau 4.3. Comparaison des ressources pour un registre de 1-bit....84

Tableau 4.4. Éléments utilisés pour divers N....85

Tableau 4.5. Surplus en fonction du nombre N de bits....85

Tableau 4.6. Gain obtenu sur l'exemple de la porte XOR....87

Tableau 4.7. Gains obtenus sur l'exemple du Full-adder 87

Tableau 4.8. Gains obtenus pour le circuit d'incrémentation pour différents nombre de bits N....88

Tableau 4.9. Relevé de performance pour le circuit d'incrémentation NCL (à 1,5V) 89

Tableau 4.10. Relevés de performance pour le circuit d'incrémentation SHF-NCL (à 1,5V) 89

Tableau 4.11. Gains obtenus 90

Tableau 4.12. Comparaison de mesures de performances sur l'ALU 93

Tableau 4.13. Gain réalisé sur la logique du circuit d'incrémentation 93

Tableau 4.14. Ressources utilisées par les registres du circuit d'incrémentation 94

Tableau 4.15. Comparaison du gain calculé avec le gain mesuré 94

Tableau 5.1. Caractéristiques de différents émetteur-récepteurs ZigBee 99

Tableau 5.2. Code d'opérations du module de communication SPI 118

Tableau 5.3. Consommation de différents FPGA de la famille Igloo d'Actel 120

Tableau 5.4. Principales caractéristiques du FPGA choisi 120

Tableau 5.5. Relevés des délais maximaux 122

Tableau 5.6. Caractéristique de l'implémentation du module de réveil 125

Tableau 5.7. Puissance dissipée par le module de réveil 126

Tableau 5.8. Comparaison des ressources et des consommations du NCL et du SHF-NCL pour le design de module de réveil sur Actel Iglo AGLN250V2 à 1.2V pour un taux de transition du signal d'entrée de 724 kHz 129

Tableau 5.9. Comparaison NCL, SHF-NCL et Logique Synchrone pour un taux de transition du signal d'entrée d'environ 22 kHz et une tension d'alimentation de 1.2V 130

Tableau 6.1. Consommations mesurées sur l'*animat* 135

Tableau 6.2. Estimations de l'autonomie à partir des mesures réelles de consommations 135

Tableau 6.3. Caractéristiques physiques 136

Tableau 7.1. Comparaison finale des plateformes 140

LISTE DES FIGURES

Figure 1-1. Définition des couches pour le protocole ZigBee et la norme IEEE802.15.4.....11

Figure 1-2. Types d'applications pour le protocole ZigBee [16].....12

Figure 1-3. Différents types de réseau ZigBee.....13

Figure 2-1. Architecture Coordinateur.....16

Figure 2-2. Prototype #1 du module Coordinateur sans boîtier.....17

Figure 2-3. Architecture générale de l'*animat*.....19

Figure 2-4. Circuit de mesure de la tension de la batterie.....22

Figure 2-5. Multiplexage des capteurs.....23

Figure 2-6. Capteur de lumière.....24

Figure 2-7. Circuit de détection infrarouge.....25

Figure 2-8. Circuit d'amplification du microphone.....26

Figure 2-9. Schéma d'un pont en H.....27

Figure 2-10. Schéma de contrôle d'un moteur.....28

Figure 2-11. Circuit de recharge de la batterie.....30

Figure 2-12. Exemples d'utilisation du LTC3105: (a) Tension de sortie sans charge, (b) Tension de sortie chargeant un super-condensateur.....31

Figure 2-13. Circuit de recharge solaire.....32

Figure 2-14. Répartition des prix.....33

Figure 2-15. Photo de l'*Animat*.....33

Figure 3-1. Types de designs asynchrones.....37

Figure 3-2. Élément C de Müller: (a) Schéma de la porte (b) Table de vérité.....38

Figure 3-3. Schéma du protocole de type délais bornés.....40

Figure 3-4. Représentations complètes des données: (a) *Dual-railI,* (b) *Quad-rail*.....42

Figure 3-5. Tables de vérités de portes logiques standard en représentation sur deux fils: (a) ET, (b) OU, (c) NON-ET et (d) INV 42

Figure 3-6. Exemple de la porte TH33: (a) Schéma (b) Chronogramme 45

Figure 3-7. Éléments de mémoire pour les portes à hystérésis 46

Figure 3-8. Porte TH33 avec rétroaction 47

Figure 3-9. Additionneur 1-bit: (a) Table de vérité, (b) Symbole 47

Figure 3-10. Additionneur NCL 1-bit avec retenue d'entrée 50

Figure 3-11. Pipeline NCL 51

Figure 3-12. Registres NCL: (a) Registre 1-bit, (b) Registre N-bits 52

Figure 3-13. Exemple de réponse du protocole NCL 53

Figure 3-14. Schéma général d'un circuit séquentiel en logique NCL 54

Figure 4-1. Registre NCL à 1-bit 59

Figure 4-2. Utilisation de portes standard: (a) Avant la modification, (b) Après la modification 61

Figure 4-3. Schéma général du SHF-NCL 62

Figure 4-4. Registre SHF-NCL 1-bit 62

Figure 4-5. Schéma du OU-Exclusif non-optimal: (a) Logique NCL, (b) Logique combinatoire 64

Figure 4-6. Schéma du OU-Exclusif NCL avec délai: (a) délai sur une fourche, (b) délai sur une branche 64

Figure 4-7. Schéma du OU-Exclusif SHF-NCL: (a) délai sur une fourche, (b) délai sur une branche 65

Figure 4-8. Cas utilisé pour l'étude des aléas 66

Figure 4-9. Exemple de circuit n°1 69

Figure 4-10. Chronogramme du circuit n°1 70

Figure 4-11. Chronogramme du circuit n°1 modifié 70

Figure 4-12. Exemple de *fan-out* reconvergeant 73

Figure 4-13. Chronogramme mettant en évidence les problèmes de *fan-out* reconvergeant 73

Figure 4-14. *Ripple Carry Adder* 75

Figure 4-15. Module de base pour la détermination de l'égalité 75

Figure 4-16. Schéma NCL du module d'égalité 77

Figure 4-17. Exemple de protocole SHF-NCL n°1 79

Figure 4-18. Exemple de protocole SHF-NCL cas n°2 80

Figure 4-19. Schéma typique d'un circuit séquentiel SHF-NCL 81

Figure 4-20. Chronogramme typique d'un circuit séquentiel SHF-NCL 83

Figure 4-21. Évolution du surplus en % en fonction de N 85

Figure 4-22. Schéma du XOR représenté selon la convention NCL 86

Figure 4-23. Évolution de la fréquence de fonctionnement en fonction de N 90

Figure 4-24. Évolution de la consommation dynamique en fonction de N 90

Figure 4-25. Évolution du rapport puissance dynamique / fréquence d'opération en fonction de N 91

Figure 5-1. Émetteur-Récepteur ZigBee: (a) SPZB32W1x2.1 , (b) CC2531, (c) MRF24J40MA, (d) MRF24J40MB 99

Figure 5-2. Schéma de l'architecture générale de l'*animat* 101

Figure 5-3. Modulation O-QPSK 102

Figure 5-4. Chaîne de réception du message de réveil : (a) Schéma de principe de la démodulation du signal reçu, (b) Signal reçu par l'antenne pour un SNR = 30 dB, (c) Signal en sortie du détecteur d'enveloppe, (d) Information reçue après décodage 103

Figure 5-5. Format de la trame du message de réveil: (a) Adresse de 16-bits, (b) Adresse de 64-bits 105

Figure 5-6. Schéma général d'un détecteur d'enveloppe 106

Figure 5-7. Tensions dans le détecteur d'enveloppe et définition des intervalles 108

Figure 5-8. Illustration des « Ripples » et du « Negative Peak Clipping » 109

Figure 5-9. Schématisation du problème d'adaptation d'impédance 111

Figure 5-10. Évolution de P_{R2} en fonction de R_2 111

Figure 5-11. Circuits permettant l'amélioration de la sensibilité 114

Figure 5-12. Architecture générale du module de réveil 117

Figure 5-13. Logigramme de la partie décodage du module de réveil 117

Figure 5-14. Environnement de développement du module de réveil: (a) Igloo Nano Starter Kit, (b) Bread-Board 120

Figure 5-15. Configuration pour la mesure du plus grand délai 122

Figure 5-16. Chronogramme permettant la mesure du plus grand délai à 1.5V 122

Figure 5-17. Délais sur un fil dans un FPGA 125

Figure 5-18. Évolution de la puissance dynamique en fonction de la fréquence d'occurrence des messages de réveil 127

Figure 5-19. Évolution du courant statique en fonction de la tension d'alimentation 128

Figure 6-1. Vue du réseau 133

Figure 6-2. Panneau d'affichage des informations 134

Figure A4-1. Unité Arithmétique et Logique: (a) Schéma, (b) Codes d'opérations 156

Figure A4-2. Fonction ET 2-bits NCL Figure A4-3. Fonction OU 2-bits NCL 157

Figure A4-4. Fonction OU-Exclusif 2-bits NCL 157

Figure A4-5. Additionneur NCL 2-bits 160

Figure A4-6. Exemple d'Unité Arithmétique et Logique en logique NCL 162

Figure A4-7. Circuit de base de l'incrémentation en logique NCL 163

Figure A4-8. Étage de synchronisation des circuits séquentiels NCL 163

Figure A4-9. Circuit d'incrémentation N-bits en logique NCL 164

LISTE DES SIGLES ET ABRÉVIATIONS

ACK	:	Acknowledgment
ALU	:	Arithmetic Logic Unit
ASIC	:	Application-Specific Integrated Circuit
ASK	:	Amplitude-Shift Keying
CAO	:	Conception Assistée par Ordinateur
CRC	:	Cyclic Redundancy Check
DI	:	Delay Insensitive
ED	:	Enveloppe Detector
EEAM	:	Energy and Environment Aware Module
EMI	:	Electromagnetic Interference
ED	:	Envelope Detector
ES	:	End System
FCS	:	Frame Check Sequence
FFD	:	Full Function Device
FPGA	:	Field-Programmable Gate Array
GALS	:	Globally Asynchronous Locally Synchronous
GUI	:	Graphical User Interface
HID	:	Human Interface Device
IEEE	:	Institute of Electrical and Electronics Engineers
LRN	:	Laboratoire de Réseau de Neurones
LUT	:	Look-Up Table
MCU	:	Micro-Controller Unit Mutually Exclusive Assertion Groups
MEAG	:	Mutually Exclusive Assertion Groups
MIPS	:	Mega Instructions Par Seconde
NCL™	:	Null Convention Logic
NRE	:	Non-Recurring Engineering
OOK	:	On-Off Keying
O-QPSK	:	Offset Quadrature Phase-Shift Keying
OUI	:	Organizationally Unique Indentifier
PAN	:	Personal Area Network
PC	:	Personal Computer

PCB	:	Printed Circuit Board
PDN	:	Pull-Down Network
PHY	:	Physical Layer
PTC	:	Positive Temperature Coefficient
PUN	:	Pull-Up Network
QDI	:	Quasi Delay Insensitive
RBFN	:	Radial Basis Function Network
RF	:	Radio Frequency ou Radio Fréquence
RFD	:	Reduce Function Device
RFID	:	Radio Frequency IDentification
SFR	:	Special Function Registers
SHF-NCL	:	State-Holding Free NULL Convention Logic
SI	:	Speed Independent
SNR	:	Signal to Noise Ratio
SPI	:	Serial Peripheral Interface
SVM	:	Support Vector Machine
UNCLE	:	Unified NULL Convention Logic Environment
USB	:	Universal Serial Bus
WBAN	:	Wireless Body-Area Networks
WCOSN	:	Wireless Control-Oriented Sensor Networks
WDCN	:	Wireless Data Collection Networks
WLSN	:	Wireless Location-Sensing Networks
WMSN	:	Wireless Multimedia Sensor Networks
WSN	:	Wireless Sensors Network
WUR	:	Wake-Up Receiver
XOR	:	Ou Exclusif
µC	:	Microcontrôleur

LISTE DES ANNEXES

ANNEXE 1 – Classes d'applications des réseaux de capteurs sans-fil 147

ANNEXE 2 – Classes d'applications basées sur le type de communication 150

ANNEXE 3 – Modèles énergétiques 152

ANNEXE 4 – Exemples de circuits en logique NCL™ 155

ANNEXE 5 - Délais dans les éléments de base du FPGA 165

ANNEXE 6 - Dessins des masques des PCB 167

ANNEXE 7 - Photos des PCB réalisés 169

ANNEXE 8 - Conception de la coque de l'*Animat* 170

ANNEXE 9 - Photos de l'*Animat* 171

ANNEXE 10 - Interface graphique 172

ANNEXE 11 - Article NEWCAS 2012 174

ANNEXE 12 - Article MWSCAS 2012 178

INTRODUCTION

Le monde de la recherche en apprentissage machine ne cesse de se développer. Un des premiers pionniers de cette recherche fut Franck Rosenblatt qui en 1957 proposa le perceptron comme élément de base pour la classification linéaire [1]. Cette proposition d'éléments de base de réseau de neurones artificiels à apprentissage supervisé reposant principalement sur une loi d'apprentissage proche de la loi de Hebb, suscita un fort engouement dans la communauté des chercheurs en intelligence artificielle alors toute jeune. Néanmoins, en 1969, Marvin Minsky et Seymour Papert [2] vinrent proposer une preuve de l'utilité limitée du perceptron et mirent en avant l'inefficacité probante de ce dernier pour apprendre une fonction aussi simple que le XOR. Ce coup dur freina quelque peu les espoirs mis sur ce classificateur compte tenu de son incapacité à distinguer des ensembles non-linéairement séparables. Pour pallier cela, la combinaison de plusieurs niveaux de classificateurs linéaires paraissait la solution, mais aucune règle d'entraînement d'un tel réseau n'avait alors été proposée. En 1988, David Rumelhart [3] proposa une règle d'apprentissage, basée sur les travaux de Parker [4], nommée la rétro-propagation du gradient de l'erreur. Cette règle permit notamment d'entraîner des réseaux de perceptrons multicouches, ce qui relança l'intérêt suscité auparavant. Il en résulta alors une recherche frénétique et prolifique qui permit de découvrir d'autres types d'apprentissages machines comme par exemple les SVM de Vladimir Vapnik [5], les RBFN [6], les réseaux de mixture d'experts [7] de Geoffrey Hinton ou plus récemment, les machines à états liquides [8].

Bien que ces algorithmes deviennent toujours plus performants, la tendance actuelle est de trouver des ensembles de données d'entraînement non-biaisées permettant une quantification de l'erreur d'apprentissage et une détermination de la performance de telles machines dans des applications réelles. En effet, on observe bien trop souvent de bons résultats obtenus à partir de simulations qui n'ont aucune réalité physique. Dans cette optique, un récent domaine de recherche s'est développé dans les universités concernant l'utilisation d'algorithmes d'apprentissage pour la commande de robots. Ce faisant, les données recueillies proviennent directement du monde extérieur et sont, par conséquent, à la fois non-biaisées par une erreur de modélisation et, en même temps, naturellement bruitées et dépendantes de la précision des capteurs. Les *animats* constituent une classe de robots en lien avec ce domaine de recherche. Le terme proposé pour la première fois par Stewart Wilson dans [9] peut à la fois désigner des robots

physiques ou virtuels simulés. Dans ce mémoire on considèrera seulement les robots physiques. Ceux-ci étant minimalement équipés au niveau mécanique, ayant des capacités restreintes et s'apparentant souvent à des insectes vivant la plupart du temps en colonie. Les avantages principaux de cette classe de robots découlent du fait qu'ayant des capacités initiales restreintes, ils devront apprendre à survivre dans leur environnement en fonction de leur expérience et éventuellement de celle des autres membres de la colonie.

Dans ce contexte, le Laboratoire de Réseaux Neuronaux de l'École Polytechnique de Montréal, parmi d'autres recherches, vise le développement d'algorithmes d'apprentissage pour de telles colonies. L'emphase étant mise sur la possibilité d'individualité et de défaut sur chacun des *animats*, ces derniers doivent malgré tout être capables de survivre le plus longtemps possible dans l'environnement *a priori* inconnu dans lequel ils évoluent. Plusieurs cas de figure seront ensuite considérés, notamment l'exploration de l'environnement, le partage de connaissance et le scepticisme qui va avec, ou encore l'adaptation à des caractéristiques propres de l'environnement au sens large. Ces dernières comprennent notamment l'éventualité d'existence de chemins plus coûteux en termes de consommation d'énergie et les compromis qui en découlent. Dans un tel cas, on peut se demander s'il est préférable d'emprunter un chemin défavorable à court terme pour bénéficier d'une amélioration d'une quelconque fonction de satisfaction à long terme. Ces particularités de l'environnement comprennent aussi la distribution de « nourriture » ou toute autre caractéristique à découvrir et à prendre en compte. Mais aussi l'éventualité d'une panne ou d'un niveau d'énergie bas qui mènerait l'*animat* à une modification de son comportement.

Par conséquent, le travail qui suit vise à proposer une plateforme pour le déploiement et l'étude d'algorithmes d'apprentissage dans des colonies d'*animats*. La principale problématique étant de fournir une plateforme à la fois suffisamment ouverte et configurable pour qu'un ensemble d'*animats* distincts ayant chacun ses qualités et défauts puisse être étudié et en même temps suffisamment restreinte pour proposer une solution peu énergivore.

Pour ce faire, ce mémoire s'organise comme suit : au Chapitre 1, une présentation générale du projet dans son contexte sera donnée. Nous y discuterons notamment des solutions proposées dans la littérature, mais aussi d'aspects plus techniques comme le choix du protocole de communication. Conséquemment, le Chapitre 2 sera consacré à l'*animat* et aux choix de conception de ses différentes parties. Ceci nous amènera à considérer la conception d'un module

de réveil pour diminuer la consommation due aux communications. Pour cela, au Chapitre 3, nous introduirons les techniques existantes de design asynchrones et en particulier la *Null Convention Logic*™ pour laquelle une amélioration en termes d'utilisation des ressources sera proposée au Chapitre 4. L'utilisation de logique asynchrone, qui n'avait pas encore été mentionnée jusqu'alors, trouvera son utilité dans la conception du module de réveil à basse consommation de l'émetteur-récepteur de la radio principale présentée et justifiée au Chapitre 5. Au Chapitre 6, le fonctionnement de la plateforme et des relevés d'expérimentations seront présentés. Par la suite, le Chapitre 7 proposera une discussion générale sur le projet de recherche ainsi qu'une ouverture sur les futurs travaux. Finalement, nous terminerons par une Conclusion générale.

CHAPITRE 1 PRÉSENTATION DU PROJET ET DÉFINITION DE L'APPLICATION GÉNÉRALE

1.1 Description de l'application « *animats* »

Comme souligné dans l'introduction, le terme *animat* peut regrouper à la fois des animaux artificiels physiques ou simulés. Les *animats* constituent une classe d'applications intéressante où l'objectif initial pour chacun d'entre eux est de survivre dans un environnement inconnu *a priori*. L'étude de ces derniers peut mener à de vastes résultats dans des domaines relativement différents. Par exemple, dans le domaine de l'apprentissage machine, les *animats* permettent d'avoir une plateforme sur laquelle développer de nouveaux algorithmes d'apprentissage. De plus, on relate bien souvent des collaborations avec l'univers de la recherche en biologie car l'*animat* permet de comprendre les mécanismes des comportements primitifs [10]. D'autre part, les résultats pourraient aussi être utilisés dans l'industrie des jeux vidéo en ce sens que certains comportements ou autres réactions pourraient être appris dépendamment des situations et amélioreraient alors le comportement des personnages [11].

Pour ce projet, on considèrera des *animats* physiques matérialisés par de petits robots minimalement équipés en termes de mécanique et relativement limités dans leurs tâches. Néanmoins ceux-ci devront trouver le moyen de survivre le plus longtemps possible dans leur environnement. L'aspect collaboratif devra être pris en compte non pas comme une nécessité mais plutôt comme une possibilité en ce sens qu'un *animat* solitaire, méfiant ou plus simplement pragmatique, devrait pouvoir avoir le choix d'ignorer les connaissances des autres membres de la colonie. À l'inverse, un *animat* plus grégaire trouverait la possibilité de communiquer avec ses prochains s'il juge que cela lui est profitable.

1.2 Problématique du projet

Par conséquent, dans ce projet nous nous proposons de fournir une plateforme rendant facile le déploiement de telles colonies. La plateforme sera composée de deux parties principales : d'une part les *animats* à proprement parler et d'autre part un module permettant de recueillir les informations de la colonie sur un ordinateur. La partie chargée de récupérer les données est nécessaire puisqu'elle permet de pouvoir quantifier l'évolution de la colonie, mais aussi car elle

permettra d'indiquer à certains *animats* comment ils doivent agir. En effet, ceci permet d'avoir une puissance de calcul centralisée qui pourra servir par exemple à contrôler les animats aux moyens d'algorithmes demandant des ressources calculatoires puissantes venant compléter les ressources embarquées qui sont relativement limitées. De plus, les *animats* devront être conçus avec une emphase mise sur la réduction de leur consommation énergétique.

De manière générale, la consommation d'énergie devra être au cœur du comportement de l'*animat* en ce sens qu'il devra s'adapter à d'éventuelles périodes de disette et faire en sorte de s'en prémunir pour les événements à venir.

1.3 Exemples d'applications visées

Les types d'applications peuvent-être très variés, le plus commun étant la survie dans un environnement où la distribution de nourriture reste à déterminer et peut être variable. Concrètement, la lumière pourrait être utilisée comme source d'énergie pour recharger des batteries et les *animats* devraient apprendre à évoluer dans l'environnement tout en anticipant leurs dépenses futures.

Un autre exemple consiste à demander aux *animats* d'aller d'un point à un autre en s'adaptant à d'éventuels changements de l'environnement à l'instar des colonies de fourmis capables après exploration de se focaliser sur un chemin minimisant leur trajet [12].

De plus, un exemple aussi très largement utilisé est celui du modèle proie-prédateur dans lequel chacun des protagonistes doit apprendre à contrer les parades de l'autre [13].

Plus récemment, les débouchés de ce projet semblaient intéresser des chercheurs dans le domaine des transports. En effet, si l'*animat* n'est plus un robot mais un ensemble de véhicules, il serait alors possible de connaître l'état du trafic routier sans avoir à le modéliser puis d'utiliser des algorithmes de traitement et d'apprentissage sur ces flux pour pouvoir informer les conducteurs de meilleurs itinéraires. Ceci aurait un double impact : une satisfaction accrue chez les conducteurs et un désengorgement des axes routiers qui conduirait à la réduction des émissions de gaz à effet de serre dans de telles situations.

Enfin, dans le domaine militaire, le déploiement d'*animats* sur terrains minés permettrait à des soldats d'être avertis de zones potentiellement périlleuses. La tâche de l'*animat* serait alors de se

promener dans l'environnement jusqu'à trouver une charge explosive, puis d'y demeurer suffisamment longtemps pour pouvoir avertir du danger lorsque les soldats passeront.

1.4 Projets existants

Dans cette partie, nous nous proposons de fournir une revue succincte de la littérature concernant les plateformes d'*animats*. En effet, compte tenu de l'intérêt croissant de la recherche pour cette discipline, plusieurs universités désireuses de développer des algorithmes d'apprentissage ou de mettre à profit leurs compétences en robotique, se sont lancées dans l'aventure. Nous présentons alors ces différents projets en les regroupant par affiliation.

L'École Polytechnique Fédérale de Lausanne a un important centre de recherche qui traite depuis une vingtaine d'années de ce genre de problématique. Au départ centralisé sur le projet Khepera [14], le Laboratoire de Système Robotique a peu à peu développé une réelle expertise en la matière en proposant divers exemples de robots notamment les versions successives II et III de Khepera mais aussi d'autres micro-robots comme le plus récent Swarm-bots [15] et la plateforme S-bots [16]. Le robot Khepera a connu un réel engouement de la part des chercheurs qui l'ont utilisé pendant près d'une décennie. Néanmoins son prix élevé a eu raison de lui. D'autres laboratoires de la même école développent et proposent de telles plateformes, notamment l'Autonomous Systems Lab. Ce laboratoire travailla à partir de 1998 sur le projet Alice [17]. Ce projet visait à proposer un robot de petite taille et peu coûteux permettant d'étudier les comportements collaboratifs. La réalisation finale fut un robot de 2x2x2 cm^3 pouvant être complété par des cartes d'extensions. Puis, ce même laboratoire ainsi que le Laboratory of Intelligent Systems et le Swarm-intelligent Systems group ont travaillé de pair sur le projet nommé « e-puck »[18] qui a pour but d'être une plateforme pour l'utilisation et la recherche académique. Malgré son design matériel libre de droits, cette plateforme se voit pénalisée, par exemple pour l'étude de colonie d'*animats*, par un coût prohibitif de près de 1000$ pour un robot simple excluant le système de recharge ou tout autre périphérique. Il est à noter que d'autres micro-robots provenant de cette même institution ont vu le jour mais ont connu beaucoup moins de succès. On peut notamment citer *Jemmy* ou *Inchy* [19].

L'université de Stuttgart proposa dans le courant de l'année 2005, une plateforme libre de droits nommée Jasmine [20]. Cette plateforme comprend aussi bien le design de micro-robots de moins de trois centimètres cube que le développement du logiciel qui l'accompagne. Plusieurs

publications présentes sur la page web du projet relatent des études algorithmiques réalisées à partir de cette plateforme notamment certaines inspirées de la recherche biologique.

L'université Harvard a récemment proposé un *animat* très intéressant : le Kilobot [21]. Ce micro-robot de la taille d'une pile bouton permet de réaliser des tâches basiques comme se déplacer et communiquer sur de courtes portées. Il est intéressant de voir qu'un ensemble de capacités si restreint permet d'avoir un comportement collectif relativement complexe. Un autre avantage principal de ce projet réside dans le fait que le coût de revient est d'environ 14$, bien que ce dernier soit vendu à un prix avoisinant 130$. Néanmoins, les capacités limitées et le type de communication très particulier risquent à terme de limiter l'ensemble des applications que l'on peut déployer sur la plateforme.

Enfin, d'autres projets isolés comme Robomote [22] ou le projet commercial Moway® [23] constituent des réalisations intéressantes. Néanmoins, Moway® est particulièrement destiné à un public qui découvre la robotique et tend par conséquent à simplifier la programmation ou à limiter la différentiation des *animats*. La plateforme Robomote ressemble, quant à elle, plus à une miniaturisation d'un robot et embarque des ressources superflues pour l'application visée.

À des fins de clarté et de comparaisons, le Tableau 1.1 récapitule les principales caractéristiques des différentes plateformes ou projets précédemment présentés lorsque celles-ci ont été fournies.

1.5 Objectifs et présentation du projet réalisé

Comme nous l'avons exposé précédemment, la littérature concernant notre projet est relativement dense. Il est alors nécessaire de se positionner et de statuer sur nos objectifs. La présente étude a donc pour but de doter le Laboratoire de Réseaux de Neuronaux d'une plateforme d'*animats* permettant de développer et d'analyser des algorithmes d'apprentissage. La nécessité de conception découle du fait que sur les plateformes précédemment présentées, bien qu'ayant chacune des propriétés intéressantes, aucune d'entre elles ne regroupe l'intégralité de celles qui nous paraissent nécessaires pour le type d'application visée.

Tableau 1.1. Principales caractéristiques des plateformes présentées

Nom	Année	Autonomie	Masse	Vitesse	Prix	MCU	Communication
Khepera	1996	45 min	80g	1.0 m/s	~1800$	Motorola 68331 @ 16 MHz	Radio (extension)
Khepera II	2002	1 heure	80g	0.5 m/s	-	Motorola 68331 @ 25 MHz	Port série
Khepera III	2006	8 heures	690g	0.5 m/s	3800$	DsPIC30F5011 @ 60MHz	Port série USB,WIFI et Ethernet (en extension)
Alice	2004	10 heures	5g	40 mm/s	>50$	PIC16LF877	Infrarouge et Radio
e-puck	2004	2 heures	200g	13cm/s	Open Source Vendu 850$	DsPIC30F6014A @ 30 MHz	RS232 et Bluetooth ZigBee en extension
Jasmine	2005	1 – 2 heures	~100g	-	~150$	ATMEGA168-AI	Infrarouge et UART
Kilobot	2011	3 – 10 heures	-	~1 cm/s	~14$ Vendu : 130$	ATMEGA328 @8Mhz	Infrarouge
Robomote	2002	20 – 30 min	90 g	~20 cm/s	~150$	ATMEGA8535 @ 8MHz	Infrarouge et Radio
Moway®	2007	2 heures	96 g	~18 cm/s	~200$	PIC18F87J50 @ 4MHz	Infrarouge Radio en extension

Ces propriétés sont résumées ci-dessous et font office d'objectifs de conception :

- Faible coût de revient (~100$)
- Plateforme modulaire laissant un vaste choix de personnalisation
- Prise en compte des problèmes énergétiques
- Ayant une autonomie d'au moins quelques heures
- Facile à prendre en main

Le troisième point est particulièrement important; considérant qu'une des tâches principales des *animats* est de survivre, ces derniers doivent être en mesure de connaître leur consommation et niveau d'énergie pour pouvoir au mieux les gérer. Les plateformes présentées précédemment se contentent, pour les meilleures, de relever uniquement le niveau de batterie. Ceci est insuffisant pour qu'un *animat* puisse apprendre quelles sont les tâches qui requièrent de l'énergie et comment en désactiver certaines compte tenu de la situation dans laquelle il se trouve.

Par ailleurs, il est important de noter que les projets de la littérature ont été réalisés par des équipes de plusieurs chercheurs et que cette étude n'a pas la prétention d'atteindre le niveau de perfectionnement des solutions proposées. Néanmoins, elle constitue une première étape à la réalisation d'une plateforme compétitive. Ceci justifie en partie l'aspect qualitatif des objectifs établis ci-dessus mais qui n'en demeurent pas moins le fil conducteur et la philosophie du projet.

1.6 Réseaux de capteurs sans-fil et classes d'applications

Avant de poursuivre, il semble nécessaire de noter que l'application « *animat* » peut apparaître dans un premier temps comme étant un cas particulier des réseaux de capteurs sans-fil (WSN). Le principal intérêt de rapprocher notre application à de tels réseaux est de pouvoir bénéficier de l'importante littérature qui les accompagne. De nombreux projets sont proposés et étudiés dans différents domaines comme la domotique, la sécurité, l'environnement, l'industrie, le médical, l'agriculture ou encore le militaire. La plupart sont caractérisés par un bas débit et une faible consommation. Un des principaux intérêts est de pouvoir distribuer aussi bien la puissance de calcul que la mesure tout en maintenant des dispositifs relativement simples et peu coûteux. C'est par ailleurs ce faible coût qui permet de déployer un grand nombre de ces dispositifs de mesure dont les tâches se limitent souvent à acquérir les données et les transmettre vers un ou plusieurs points de collecte. Dans la plupart des applications, les capteurs sont alimentés par des batteries mais leur temps de vie doit être suffisamment long pour transmettre la ou les informations qu'ils détiennent. Il est possible de distinguer ces différentes classes d'applications de la sorte :

- Wireless Body-Area Networks (WBAN)
- Wireless Data Collection Networks (WDCN)
- Wireless Location-Sensing Networks (WLSN)
- Wireless Multimedia Sensor Networks (WMSN)
- Wireless Control-Oriented Sensor Networks (WCOSN)

Dans l'ANNEXE 1 nous donnons une description sommaire de chacune de ces différentes classes visant à permettre une meilleure compréhension des enjeux et caractéristiques de chacune. Ceci nous permet par la même occasion de pouvoir situer l'application « *animat* » dans cet ensemble. Une description plus détaillée peut être trouvée dans [24].

Cette classification permet par conséquent d'organiser l'information et de pouvoir situer chaque projet par rapport à un autre. D'autre part, d'après les caractéristiques et fonctionnalités communes à l'intérieur de chaque classe, on peut entrevoir de manière générale quelles seront les implications techniques. Bien que cette classification soit largement acceptée dans la communauté scientifique, on constate la difficulté de positionnement de l'application « *animat* ». En effet, celle-ci serait à la frontière des classes WCOSN et WDCN, et dans une moindre mesure WMSN et WLSN. Notamment, l'établissement d'un modèle énergétique basé sur les classes

d'applications n'est pas possible. Pour ce faire, nous proposons une classification différente de tels réseaux basée sur le type d'utilisation des communications. Celle-ci se divise alors en trois classes énoncées ci-dessous :

- Communications continues
- Communications périodiques
- Communications lorsque nécessaire

Le détail de ces nouvelles classes est donné dans l'ANNEXE 2. Ce faisant, les modèles énergétiques seront alors plus faciles à établir et permettront à un concepteur de pouvoir jauger les différents paramètres qui influenceront la consommation d'énergie. Une proposition de définition concernant ces modèles énergétiques est fournie en ANNEXE 3.

1.7 Protocole de communication ZigBee

Avant de poursuivre ce mémoire, il semble important de clarifier de quel type de communication les *animats* seront dotés. Bien que cela puisse paraître prématuré dans l'analyse, nous avons choisi d'utiliser une communication sans fil basée sur la norme 802.15.4 et utilisant la *stack* ZigBee-Pro. Ce protocole de communication a été choisi principalement pour les trois raisons suivantes : tout d'abord celui-ci a été conçu avec comme objectif de proposer une solution peu énergivore. D'autre part, le protocole ZigBee revêt une notoriété grandissante en ce sens que de plus en plus de périphériques l'utilisent directement ou de façon dérivée. Ceci constitue une caractéristique intéressante pour notre projet car les *animats* ne devraient pas être cantonnés à une « espèce » que l'on aurait bien voulu concevoir mais la colonie devrait pouvoir bénéficier d'une multitude de singularités. Par suite, le fait de pouvoir utiliser n'importe quel périphérique utilisant le protocole ZigBee est une richesse pour notre application en ce sens qu'un capteur de température pourra être vu comme un *animat* à part entière ayant comme particularité une incapacité à se mouvoir contrebalancée par une forte précision de mesure de la température. Enfin, la dernière des trois raisons les plus importantes nous ayant conduit à choisir ce protocole est la possibilité de pouvoir avoir plusieurs types de réseaux comme nous allons le voir. Notamment cela pourra permettre par exemple à un *animat* de pouvoir communiquer avec un autre sans avoir à passer par un routeur ou autre coordinateur de réseau. Parmi d'autres, ces trois raisons nous ont conduits à choisir ZigBee comme protocole de communication. Dans ce qui suit, nous proposons de donner une brève description de ce protocole.

Qu'est-ce que ZigBee?

C'est un standard de communication permettant la mise en place et la gestion de réseaux sans-fil. Pour ce faire, ZigBee utilise la norme IEEE 802.15.4 [25] qui opère dans les bandes radio *Industrial, Scientific and Medical* (ISM) de fréquences 868 MHz, 915 MHz et 2.4 GHz. Un amalgame est souvent fait entre ZigBee et la norme IEEE 802.15.4. Alors que la première définit principalement tout ce qui a un rapport avec la gestion du réseau et l'application, la norme IEEE.802.15.4 définit quant à elle les couches de bas niveau comme la gestion de l'adressage et celle de l'émetteur-récepteur radio. La Figure 1-1 ci-dessous, résume les différentes couches :

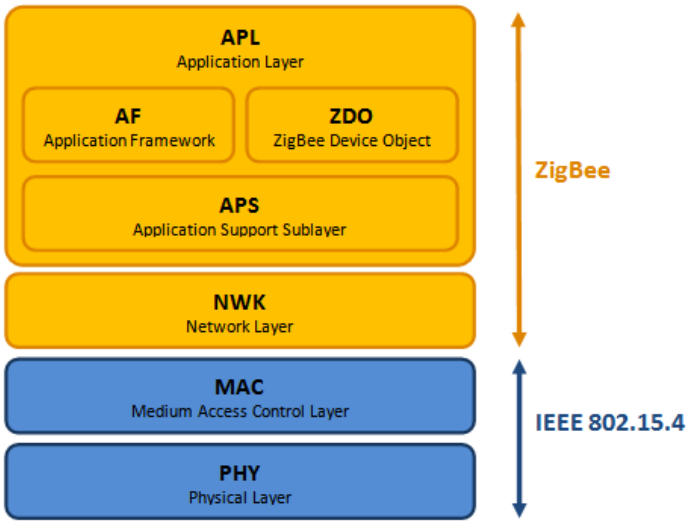


Figure 1-1. Définition des couches pour le protocole ZigBee et la norme IEEE802.15.4

Certains des objectifs principaux de ZigBee sont de proposer un protocole stable, peu couteux et à basse consommation qui puisse être facilement implémenté dans des applications embarquées que ce soit pour la domotique ou pour des applications à caractère plus industriel. Par ailleurs, bien qu'ayant connu des débuts difficiles, ce protocole semble aujourd'hui s'imposer dans des applications de la vie de tous les jours comme le suggère la Figure 1-2 disponible sur le site de la *ZigBee Alliance* [26].

Figure 1-2. Types d'applications pour le protocole ZigBee [16]

En revanche, contrairement au Wi-Fi par exemple, le protocole ZigBee ne permet que de faibles bandes passantes. Les taux de transfert dépendent de la fréquence de la porteuse et de la modulation comme rappelé dans le Tableau 1.2 inspiré du travail de [27]. Il est important de noter que l'utilisation des fréquences ci-dessous n'est autorisée que sur certaines parties de la planète et qu'il est donc nécessaire de se renseigner sur la législation en vigueur.

Tableau 1.2 Caractéristiques physiques du protocole ZigBee

PHY (MHz)	Bande de fréquences (Mhz)	Paramètres de diffusion		Paramètres de données		
		Taux de pulse (kchip/s)	Modulation	Taux de transfert (kb/s)	Taux de symboles (ksymboles/s)	Symboles
868/915	868 – 868.6	300	BPSK	20	20	Binaire
	902 - 928	600	BPSK	40	40	Binaire
868/915 (optionnel)	868 – 868.6	400	ASK	250	12.5	20-bit PSSS
	902 - 928	1600	ASK	250	50	5-bit PSSS
868/915 (optionnel)	868 – 868.6	400	O-QPSK	100	25	16-ary Orthogonal
	902 - 928	1000	O-QPSK	250	62.5	16-ary Orthogonal
2450	2400 – 2483.5	2000	O-QPSK	250	62.5	16-ary Orthogonal

Quels sont les types de nœuds et de réseaux?

La norme IEEE802.15.4 définit deux types de nœuds dépendamment de leur capacité ou non à pouvoir démarrer un réseau personnel : les FFD et RFD. Le protocole ZigBee, quant à lui, définit trois différents types de nœuds que l'on peut retrouver dans un réseau : le coordinateur, les routeurs et les ES (dispositifs à l'extrémité du réseau dans le cas d'un arbre). La documentation détaillée peut être trouvée dans [25] et [26]. Le coordinateur est chargé de mettre en place et de gérer le réseau. Les routeurs récupèrent et redistribuent l'information. Enfin les ES sont les dispositifs chargés de générer l'information utile (capteurs…). En théorie, chaque nœud pourrait passer d'un type à l'autre dynamiquement. Néanmoins en pratique, on profite du fait que les ES réalisent des tâches plus simples que les routeurs, qui eux-mêmes réalisent des tâches plus

simples que le coordinateur, pour pouvoir diminuer l'utilisation de la mémoire programme. Il n'est donc plus nécessaire de charger l'intégralité du code mais le réseau perd en adaptabilité.

Le protocole ZigBee permet une grande diversité de type de réseau. En effet, il est possible de mettre en place des réseaux de type point-à-point (*peer-to-peer)*, en étoile (*star*), en grappes (*clusters*) ou étoilé (*mesh*). Compte tenu de leur programme plus complet, seuls les FFD peuvent communiquer entre eux. Par conséquent, le type de réseau est lié aux types de nœuds utilisés. La Figure 1-3 donne un exemple de réseau que l'on peut obtenir.

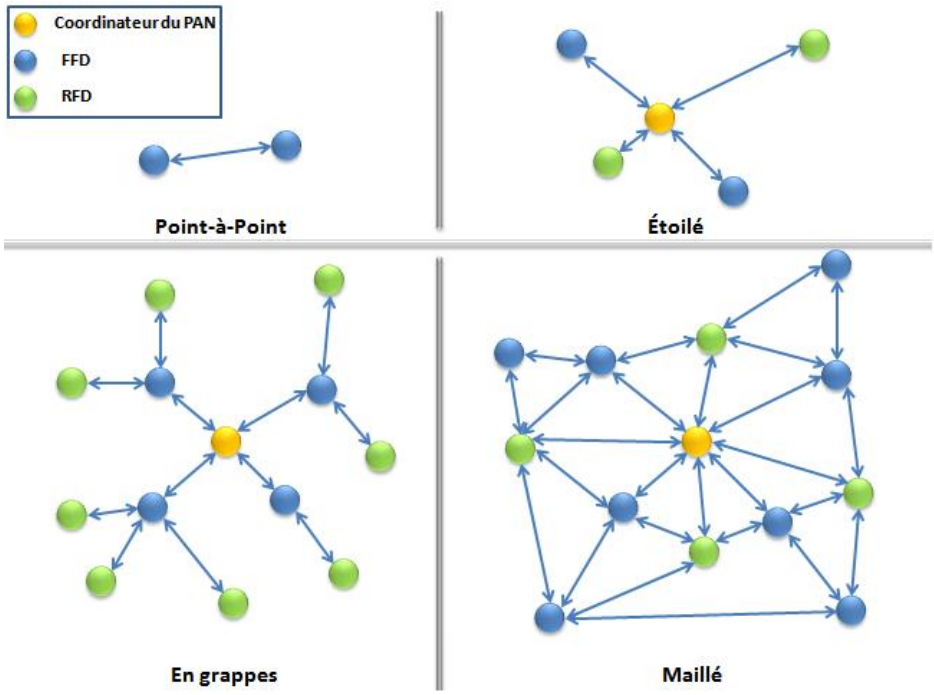


Figure 1-3. Différents types de réseau ZigBee

Certaines limitations s'appliquent quant à l'interruption de l'alimentation des routeurs et du coordinateur. Pour notre application, le coordinateur sera directement connecté à un PC et il est acceptable de considérer qu'il sera alimenté par une source de tension continue et inépuisable. Pour que les *animats* puissent communiquer entre eux, il semble nécessaire qu'ils soient des routeurs. Par conséquent, les contraintes d'alimentations de ces derniers devront être prises en compte dans la suite de l'étude.

Quelle est la forme de la trame? Comment sont adressés les nœuds?

Le format standard d'une trame est donné dans le Tableau 1.3. Celui-ci peut se décomposer en trois parties distinctes dont le détail est donné dans [25]:

- L'entête
- Le contenu
- Le contrôle : code de détection d'erreurs

Tableau 1.3. Format de la trame

Partie	Entête							Contenu	Contrôle
Octets	2	1	0/2	0/2/8	0/2	0/2/8	0/5/6/10/14	Variable	2
Détails	Contrôle de la trame	Numéro de Séquence	Identifiant du PAN de destination	Adresse de destination	Identifiant du PAN source	Adresse de la source	Entête de sécurité auxiliaire	Contenu de la trame	FCS

Au niveau du réseau, les nœuds peuvent être identifiés individuellement par différentes adresses. Tout d'abord chaque nœud a une adresse MAC de 64 bits unique. Les 24 premiers bits constituent le OUI et sont vendus par IEEE. Les 40 bits restant sont fournis par le manufacturier du nœud ZigBee qui doit s'assurer que ce dernier n'a pas déjà été attribué. Cette adresse est généralement utilisée pour rejoindre le réseau. Par la suite, c'est une adresse réseau de 16 bits qui est utilisée. Initialement celle-ci était choisie dépendamment de la position du nœud dans le réseau. Avec le protocole ZigBee Pro, cette adresse est choisie aléatoirement par le nœud qui cherche à joindre le réseau. Si celle-ci est déjà prise, on résout le conflit en utilisant l'adresse MAC. D'autre part, le protocole ZigBee permet d'établir des communications d'un nœud vers un autre (*unicast*), vers tous les nœuds d'un réseau (*broadcast*) et vers des sous-groupes du réseau (*multicast*). De plus amples détails peuvent être trouvés dans [25, 26] et [28, 29].

CHAPITRE 2 CONCEPTION DE L'ANIMAT

D'un point de vue théorique, le but initial d'un *animat* est d'évoluer le plus longtemps possible dans son environnement. Cela implique, entre autre, d'apprendre à connaître le milieu environnant et d'autre part d'apprendre à se connaître notamment en termes de points forts et points faibles. Ceci passe notamment par la découverte des ressources nécessaires pour effectuer une certaine tâche. La grande différence des *animats* par rapport à des robots classiques, notamment ceux présentés dans le Chapitre 1, se situe dans l'absence de connaissance *a priori*. En effet, bien que l'on puisse supposer que certains modules consomment plus que d'autres, comme par exemple pour le déplacement, il faut laisser la possibilité à l'*animat* de le découvrir par lui-même et de prendre des décisions en conséquence. Deux intérêts en découlent : premièrement cela permet un environnement de test pour les algorithmes d'apprentissage et cela permet à l'*animat* de s'adapter à son milieu. Par exemple, supposons que celui-ci se déplace de telle sorte qu'il descend constamment des pentes alors peut-être que l'énergie propre aux déplacements sera inférieure à celle pour la communication par exemple. Dans ce cas-ci l'*animat* pourra adapter au mieux ses activités contrairement au cas où les consommations relatives auraient été fixées. Par conséquent, ce chapitre présente les choix de conception retenus pour la plateforme en prenant en considération les trois principaux objectifs suivants:

- Réduction de la consommation énergétique
- Capacité à gérer ses propres ressources
- Coût modéré

2.1 Architecture Générale et Justifications

La plateforme se décompose en deux sous parties : la première est un module directement relié à un PC, l'autre est l'*animat* à proprement parler. Bien que la première ne soit pas le cœur de notre projet, elle demeure nécessaire pour pouvoir recevoir les informations des différents *animats* et éventuellement les traiter. En effet, un des objectifs du projet était d'avoir la possibilité de délocaliser la puissance de calcul. Par conséquent, dans les applications futures, on pourrait très bien imaginer qu'un *animat* estime qu'il est plus rentable pour lui de communiquer les données à traiter plutôt que de réaliser les calculs localement. De ce fait, nous présentons ici le module relié au PC puis l'architecture générale de l'*animat*.

2.1.1 Module Coordinateur

L'objectif de ce module est multiple. Premièrement, du point de vue réseau ce module est chargé de mettre en place le réseau ZigBee auquel chacun des *animats* pourra se connecter. Ce module permet donc la mise en place du réseau et l'autorisation de rejoindre celui-ci. D'autre part, ce module devrait pouvoir être utilisé pour recevoir les données des *animats*, éventuellement les traiter, et renvoyer soit des informations ou autres résultats, soit des ordres directs pour effectuer certaines tâches. Pour ce faire, nous avons conçu un dispositif se connectant au port USB et disposant d'une antenne ZigBee. Le schéma de l'architecture générale est fourni dans la Figure 2-1.

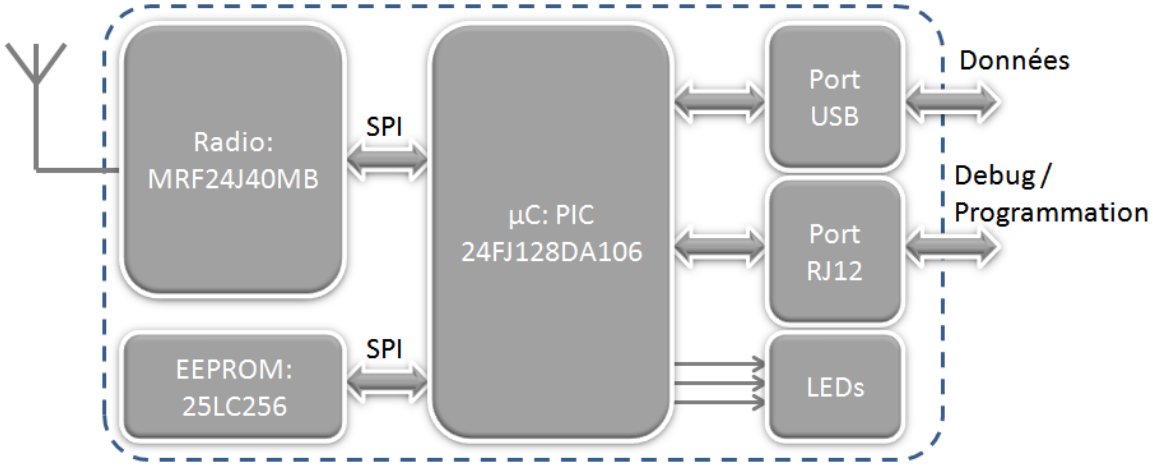


Figure 2-1. Architecture Coordinateur

L'architecture de ce module est relativement simple. Le microcontrôleur permet de gérer les piles ZigBee et USB dans le but de pouvoir à la fois créer et gérer le réseau sans-fil, et de permettre la communication avec le PC. Plusieurs remarques sont à faire :

- Pour permettre une facilité d'utilisation et ne pas restreindre l'éventuelle mobilité de l'utilisateur, le module est autoalimenté via le port USB.
- Un port RJ12 a été ajouté dans le but de pouvoir reprogrammer ou debugger le microcontrôleur. Cela permet entre autre de changer les fonctionnalités du coordinateur et de ne pas restreindre le champ d'application de la plateforme
- La mémoire EEPROM permet de stocker des données concernant le réseau ZigBee
- Les LED permettent une signalisation par rapport au fonctionnement de la carte. On utilise ici trois LED indiquant la mise sous tension, l'activité USB et l'activité sans-fil.

Le code du microcontrôleur gère actuellement dans sa boucle principale les tâches relatives au réseau puis les tâches propres à la connexion USB. Pour ne pas perturber l'activité réseau, les tâches USB sont toutes non-bloquantes et un dispositif de FIFO à haute et basse priorité a été mis en place pour stocker les messages qui ne pourraient pas être immédiatement traités.

Par ailleurs, le profil USB utilisé est le profil HID. Celui-ci nous permet une utilisation facilitée du module sans nécessité de *driver* tout en offrant une bande passante suffisante allant jusqu'à 12 Mbits/s. La Figure 2-2 représente le premier prototype du module réalisé sans son boîtier. Le lecteur intéressé pourra trouver le dessin du PCB en ANNEXE 6 réalisé à l'aide de Design Spark. Finalement pour intéragir avec l'utilisateur, un GUI a été développé sous JAVA et sera présenté dans le Chapitre 6.

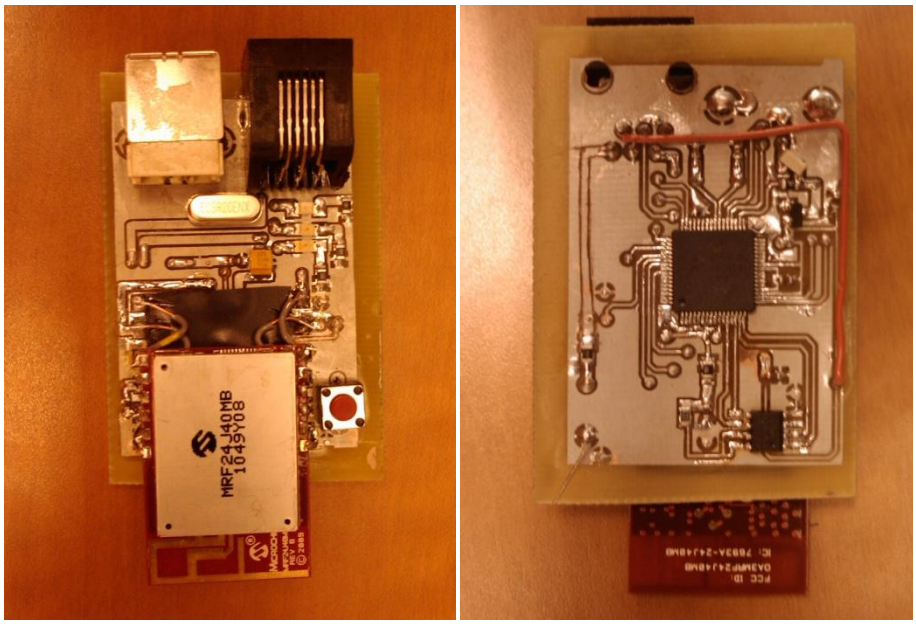


Figure 2-2. Prototype #1 du module Coordinateur sans boîtier

2.1.2 Animats

Contrairement à la simplicité apparente du module précédent, la conception de l'animat est plus étoffée. C'est pourquoi dans cette partie nous présenterons l'architecture générale de ce dernier avant de détailler chacun des choix de conception dans les parties subséquentes. Le cœur de l'architecture est comme précédemment un microcontrôleur autour duquel se greffent plusieurs modules complémentaires. Parmi eux, on compte l'émetteur-récepteur ZigBee, le module de gestion de la batterie, les différents capteurs et la gestion des moteurs. On notera de plus, que le

design devrait permettre de rajouter autant que faire se peut, d'autres capteurs que ceux initialement disponibles dans le but de laisser émerger des particularités dans une colonie. Ceci implique d'une part de doter l'*animat* de base de capteurs que nous jugeons indispensables et d'autre part de permettre l'ajout d'un maximum d'autres capteurs. L'architecture générale alors retenue est dépeinte sur la Figure 2-3.

Les capteurs de base utilisent un nombre non négligeable de broches notamment analogique. Néanmoins, les 34 broches restantes peuvent être utilisées pour connecter un grand nombre de capteurs en utilisant par exemple des multiplexeurs analogiques et numériques externes. Par ailleurs on notera la présence de capteurs de consommation qui sont soit des capteurs de courants, soit des capteurs de tensions placés à des endroits stratégiques de l'*animat* et fonctionnant comme des capteurs internes de celui-ci.

La communication infrarouge est un supplément à la communication radio permettant une alternative moins gourmande en énergie lorsque les animats sont proches.

Enfin le circuit de contrôle des ressources permet de désactiver certains ensembles de périphériques dans le but de diminuer la consommation. Par exemple, l'*animat* pourrait choisir de déconnecter l'ensemble des capteurs, la radio ou encore les moteurs.

2.2 Cerveau

Le domaine de recherche des *animats* s'oppose à l'univers de la robotique plus général notamment par la différence de capacité de calculs embarquée. En effet, des robots comme *Khepera* dont nous parlions au Chapitre 1, ou des robots plus évolués comme *Asimo*[1] ou B*igDog*[2], embarquent d'importants dispositifs de calcul pour le contrôle des déplacements ou encore l'apprentissage machine.

[1] http://fr.wikipedia.org/wiki/ASIMO

[2] http://fr.wikipedia.org/wiki/BigDog

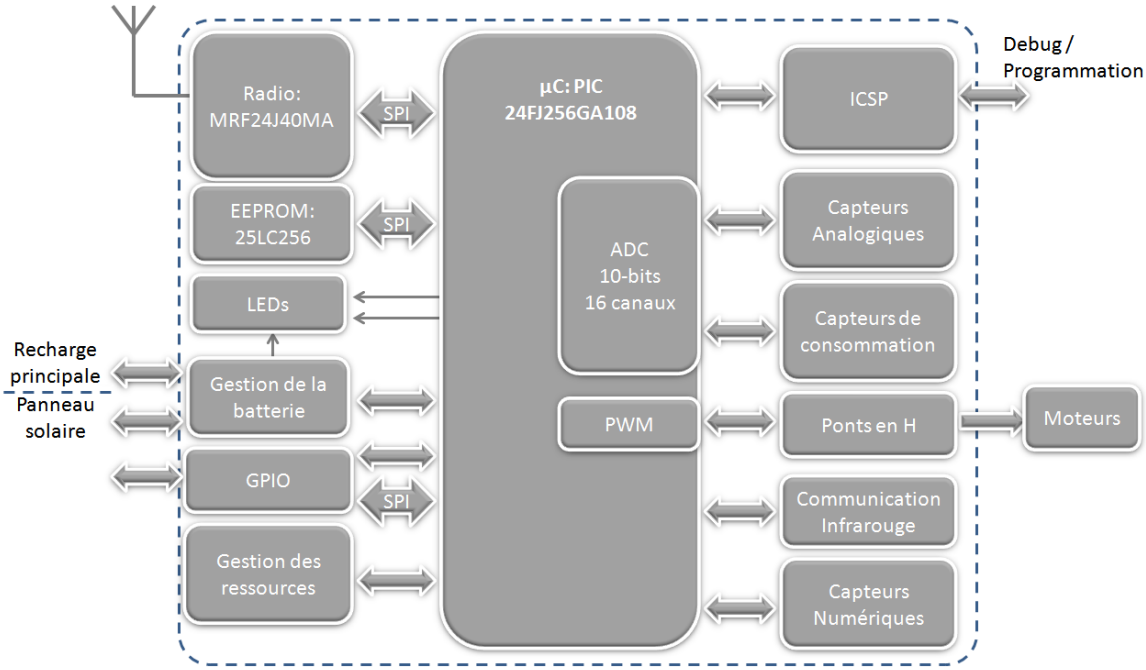


Figure 2-3. Architecture générale de l'*animat*

À l'inverse, les *animats* devraient être dotés d'une capacité de calculs modérée voire faible impliquant certaines contraintes sur les calculs réalisables localement. De manière plus générale, l'ensemble des choix de design devrait être réalisé en tenant compte du fait que toutes les capacités des *animats* devraient être réduites; aussi bien celle de calcul que celle à percevoir l'environnement. En effet, de ces contraintes devraient naître des techniques pour qu'à partir de données peu précises voire fortement bruitées ou biaisées par le contexte d'un *animat* (notion de subjectivité), celui-ci parvienne tout de même à obtenir une connaissance précise. Cela peut émerger de la collecte d'une multitude de données de capteurs différents mais dont l'information est redondante ou encore par la mise en commun des données perçues par la colonie. À l'inverse, comme nous le mentionnions, émettre des messages radio demande une quantité substantielle d'énergie comparativement à celle requise par le reste des tâches. Il est donc intéressant de laisser l'*animat* choisir de traiter localement les données ou de délocaliser son cerveau vers, par exemple, un PC ou un autre *animat* qui serait plus à même de calculer.

Par conséquent, la puissance de calcul d'un *animat* devrait se situer quelque part entre ces deux contraintes somme toute très qualitatives. De ce fait, nous avons choisi d'utiliser un microcontrôleur pour diverses raisons. Premièrement, le prix d'une telle puce est relativement faible. D'autre part, en embarquant toujours plus de périphériques, les microcontrôleurs

répondent à une large gamme d'applications et sont parfaitement bien adaptés à la robotique. Enfin, leur consommation décroit toujours plus et certain d'entre eux permettent des courants de veille aussi faible que quelques dizaines de nano-ampères. Le seul défaut que revêt l'utilisation d'un microcontrôleur se situe dans l'incapacité de réaliser des calculs en parallèle. Cet aspect peut néanmoins être résolu en émulant un calcul parallèle à partir d'une machine séquentielle ou bien en ajoutant des cartes d'extension pour des calculs spécialisés.

Toutes ces caractéristiques nous ont conduits à choisir le PIC24FJ256GA108. Ce microcontrôleur a la particularité d'être compatible avec la stack ZigBee mise à notre disposition. Par ailleurs, son espace mémoire est suffisamment important pour permettre le développement d'applications relativement complexes. Enfin, en termes de consommation celui-ci affiche 1 mA/MIPS en mode actif et entre 100 nA et 2.5µA selon le mode de veille choisi. Enfin, compte tenu que la stack ZigBee requière une EEPROM externe et une antenne externe et que celles-ci utilisent le protocole de communication SPI, nous avons opté pour un microcontrôleur ayant au moins trois ports de ce type, nous permettant ainsi de figer deux de ces ports pour les ressources réseau et de laisser le troisième libre pour l'utilisateur. Les caractéristiques les plus importantes de ce microcontrôleur sont résumées dans le Tableau 2.1.

Tableau 2.1. Principales Caractéristiques du PIC24FJ256GA108

Architecture 16 bits	16 bits
Performance	16 MIPS
Mémoire Programme	256kB
Mémoire RAM	16384 bytes
Tension d'alimentation	Entre 2 et 3.6V
GPIO	69
Ports de Communication	4-UART, 3-SPI, 3-I2C
Convertisseur Analogique-Numérique	1 – 16 canaux, 10 bits, 500 kS/s
PWM	9
Multiplication, Division	Hardware
Timers	5 x 16 bits
Consommation	Normal : 1 mA/MIPS @ 2V Sleep : 100 nA Idle : 2.5 µA @ 2V

2.3 Communication et Réveil

Contrairement au module coordinateur, le module de communication choisi pour l'*animat* est le MRF24J40MA. La principale différence réside dans le fait que ce module-ci est dépourvu d'amplificateur d'où des communications sur de moins grande portée. L'avantage d'un tel choix est la diminution de la consommation nécessaire pour l'envoi et la réception de messages. Là où

le module radio du coordinateur consomme respectivement 25 mA et 130 mA en moyenne pour la réception et l'émission de messages, le module radio de l'*animat* ne consomme plus que 19 mA et 23 mA. C'est la contrainte en consommation d'énergie qui a motivé ce choix en prenant en compte que le module coordinateur est alimenté via le port USB alors que l'*animat* l'est sur batterie. Des tests en environnement intérieur nous ont permis d'atteindre des communications entre l'*animat* et le coordinateur sur une vingtaine de mètres au travers de quelques murs.

D'autre part, compte tenu des contraintes de temps de développement, le module de réveil ne fait pas partie des capteurs de base de l'*animat* mais devra être ajouté dans des travaux futurs. On notera tout de même que le présent mémoire présente une preuve de concept de ce module et a de plus détaillé la conception de ce dernier. Celui-ci sera alors présent comme carte d'extension.

2.4 Capteurs

Une des contraintes de conception de l'*animat* fut de pouvoir doter ce dernier d'un grand nombre de capteurs. Notamment, cela permet comme nous l'expliquions plus haut d'avoir différents points de vue d'une même grandeur de sorte que la mise en commun de ces données, éventuellement peu précises, puisse fournir une information qui l'est davantage. Par conséquent, l'*animat* devrait pouvoir laisser un maximum de ports disponibles pour que l'utilisateur puisse ajouter les capteurs qui lui semblent nécessaires. Néanmoins, nous considérons que certains capteurs sont primordiaux et dotons de ce fait l'*animat* de capteurs de base nécessaires à son évolution minimum dans un environnement. Dans la suite, nous détaillons chaque grand groupe de capteurs disponible sur l'*animat*.

2.4.1 Capteurs de courant et niveau de batterie

Une des grandes différences proposée par rapport aux diverses plateformes présentées dans le Chapitre 1, réside dans la capacité de l'*animat* à pouvoir estimer les dépenses énergétiques de ses actions. En effet, la problématique centrale de l'application étant la survie dans un environnement inconnu, l'*animat* devrait pouvoir apprendre et prévoir quelles seront les dépenses énergétiques associées à ses actes. Pour ce faire, nous avons opté pour trois stratégies complémentaires.

Tout d'abord, la tension de la batterie est souvent corrélée de manière non-linéaire à sa capacité restante comme le montrent certaines équations comme celle de Butler–Volmer. Par conséquent, connaître la tension de la batterie est un indicateur précieux pour que l'*animat* sache combien de

temps il lui reste avant de ne plus pouvoir fonctionner ou pour restreindre son activité. Certains dispositifs sur le marché permettent une gestion de la batterie, de sa charge, sa décharge et de son état. Malheureusement ceux-ci sont souvent trop onéreux. Pour notre application, nous décidons d'utiliser une simple mesure de tension. Malheureusement, sur le plan conceptuel, on ne peut *a priori* pas mesurer le V_{dd} qui est généralement une tension de référence de l'ADC. Par conséquent, ici on choisi de relever la tension avant le régulateur de tension au travers d'un diviseur de tension. De cette manière, la tension de référence est fixe à 3.3V dans notre cas alors que l'on peut suivre l'évolution de la tension de la batterie avant que celle-ci arrive dans une zone de « sur-décharge ». Le circuit ainsi résultant est représenté sur la Figure 2-4.

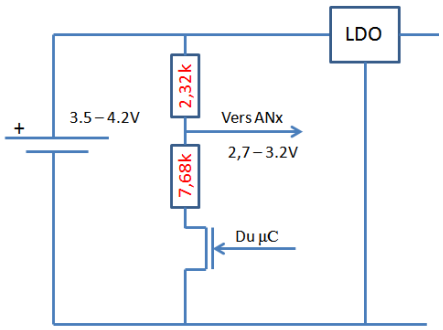


Figure 2-4. Circuit de mesure de la tension de la batterie

Sur ce schéma, on notera l'utilisation d'un interrupteur. Ceci constitue la deuxième stratégie employée dans l'architecture de l'*animat* nommée « Gestion des ressources » dans la Figure 2-3. En effet, l'*animat* devrait pouvoir être en mesure d'éteindre les modules externes ou internes qu'il n'utilise pas. Pour les modules externes au microcontrôleur, nous utilisons cette technique d'interrupteur pour limiter les pertes statiques. Pour cela, nous utilisons un jeu de multiplexage/démultiplexage pour pouvoir activer les modules d'intérêt. Le principal désavantage de cette technique réside dans l'incapacité de pouvoir activer plusieurs périphériques en même temps. Néanmoins, la volonté de permettre à l'utilisateur d'ajouter un maximum de capteurs impose ce genre de stratégie. Par conséquent, un ensemble de modules externes sont multiplexés comme représenté sur la Figure 2-5.

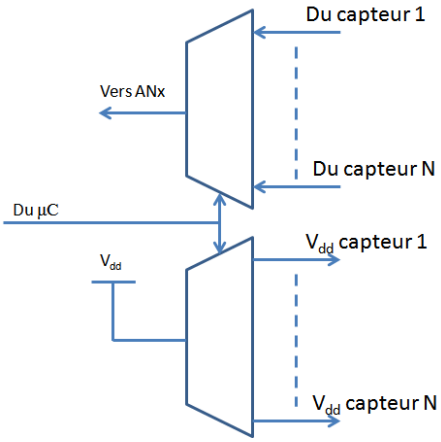


Figure 2-5. Multiplexage des capteurs

Enfin, on notera la présence de capteurs de courant. Ceux-ci incarnent la troisième stratégie mise en place pour réduire la consommation d'énergie. En effet, plutôt que de fixer *a priori* des relations entre les consommations de divers sous-modules, en dotant l'*animat* de capteurs de courant, celui-ci aura la capacité de connaître ses dépenses personnelles et de les adapter à partir d'un algorithme d'apprentissage par exemple. Pour ce faire, nous plaçons sur la carte de base de l'*animat* six de ces capteurs aux endroits stratégique suivants :

- Courant débité par la batterie
- Courant consommé par les capteurs
- Courant consommé par le microcontrôleur
- Courant consommé par le moteur 1
- Courant consommé par le moteur 2
- Courant consommé par la radio

Comme précédemment, ces informations sont multiplexées dans le but de ne requérir qu'un seul canal analogique du microcontrôleur.

2.4.2 Capteurs de température

Parmi les capteurs de base et dans le but de permettre à l'*animat* de connaître son environnement, celui-ci est doté d'un capteur de température fournissant une tension proportionnelle à la température environnante avec une précision de ± 1°C. Par ailleurs, la consommation des semi-conducteurs étant dépendante de la température, l'*animat* pourra par exemple apprendre que la consommation augmente avec la température et fuir en conséquence les endroits chauds. Ceci est alors rendu possible par la mise en commun des informations de ce capteur et de ceux de courant.

2.4.3 Capteurs de lumière

Dans le but de répondre aux applications de type « survie », l'*animat* a souvent besoin de trouver de la nourriture. Une des approches pour cela est, par exemple, de rechercher les endroits illuminés. On imagine aisément le cas où un animat serait doté d'un panneau solaire et pourrait recharger ses batteries. Dans ce cas là, la recherche de la nourriture ne serait plus théorique mais permettrait physiquement à l'*animat* de pouvoir évoluer plus longtemps. Le montage utilisé pour ce capteur de lumière est alors représenté sur la Figure 2-6.

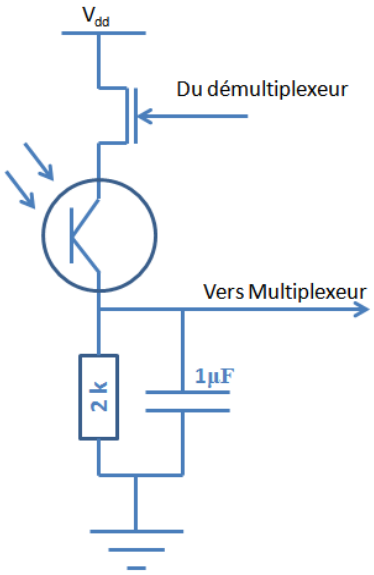


Figure 2-6. Capteur de lumière

2.4.4 Détecteurs Infrarouges

L'évolution dans un milieu inconnu requiert de pouvoir se situer et d'éviter des obstacles éventuels. Pour cela, plusieurs techniques sont possibles. Parmi elles, on compte les ultrasons, les télémètres laser ou encore l'émetteur/récepteur infrarouge. Pour des raisons de facilité de mise en œuvre, de coût et de consommation, nous avons choisi d'utiliser des détecteurs infrarouges. Par ailleurs, ce dispositif peut aussi être utilisé pour de la communication locale entre deux *animats* présentant l'avantage d'être moins énergivore que celle nécessaire pour l'émetteur/récepteur ZigBee.

Le principe de tels détecteurs est relativement simple : l'émetteur émet une onde infrarouge qui se propage dans l'environnement. Cette onde est éventuellement réfléchie par l'environnement

jusqu'au récepteur infrarouge. En se propageant dans un milieu, une onde lumineuse perd de manière exponentielle en intensité en fonction du trajet parcouru : c'est la loi de Beer-Lambert[3]. Par conséquent, le récepteur infrarouge étant capable de « mesurer » cette intensité lumineuse, il est alors théoriquement possible de connaître la longueur du trajet parcouru par l'onde.

L'émetteur infrarouge n'est autre qu'une diode électroluminescente. Le récepteur quant à lui est un phototransistor dont la base est remplacée par un détecteur d'intensité lumineuse. De ce fait, le transistor devient plus ou moins passant en fonction de l'onde incidente. En utilisant le montage de la Figure 2-7, nous sommes capables d'obtenir un détecteur infrarouge efficace ne consommant que 12 mA lorsqu'il est actif. On notera que le récepteur consomme moins d'un micro-ampère dans l'obscurité et 8 µA lorsque l'onde est la mieux réfléchie.

2.4.5 Microphone

Pour couvrir divers cas d'applications, les *animats* sont dotés de microphone de type électret. Ces microphones ont la particularité d'être petits, peu coûteux et faciles à utiliser. D'autre part, en utilisant la propriété de leur matériau, ceux-ci conservent une charge permanente permettant ainsi de ne pas recourir à un circuit de polarisation.

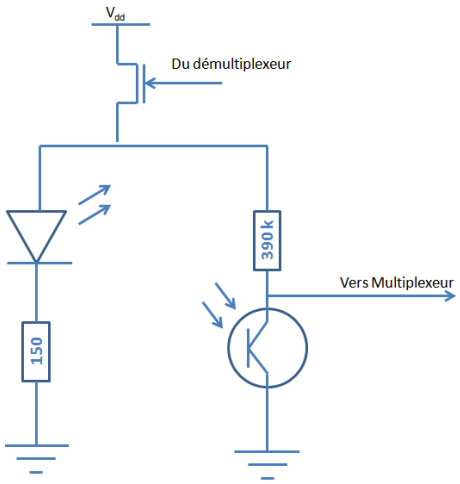


Figure 2-7. Circuit de détection infrarouge

[3] http://fr.wikipedia.org/wiki/Loi_de_Beer-Lambert

Ce microphone permettra par exemple de sonder l'environnement pour le caractériser par exemple, mais on peut aussi imaginer des applications de fuite des endroits bruyants comme le font la plupart des animaux ou encore un autre moyen de communication si un émetteur sonore est ajouté. Néanmoins, ceci nécessite un circuit d'amplification que nous représentons sur la Figure 2-8. Ce circuit permet d'avoir un gain d'amplification d'environ 100 en amplitude et de centrer le signal résultant.

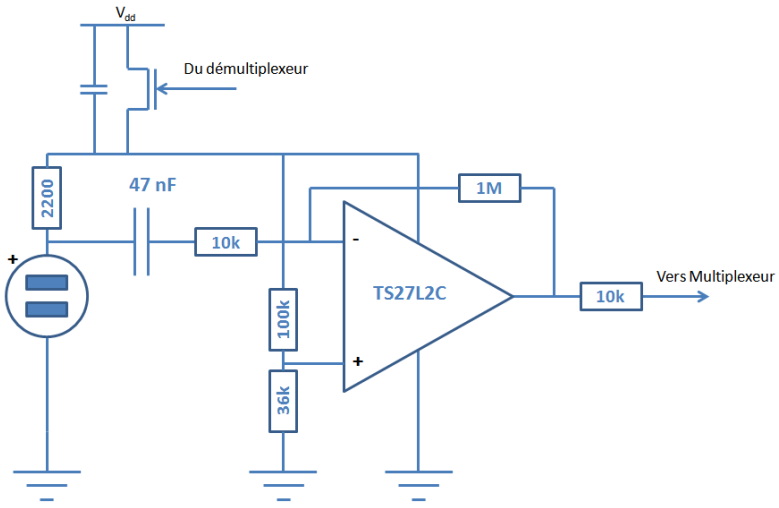


Figure 2-8. Circuit d'amplification du microphone

2.4.6 Accéléromètre

Enfin, un dispositif couramment embarqué dans les plateformes mobiles est un accéléromètre. Ce dernier permet d'obtenir l'accélération de l'*animat* selon trois axes. Cette information trouve une multitude d'applications : en intégrant cette information, il est possible de connaître la vitesse puis la position relative de l'*animat*. Ce type d'application est d'ailleurs un bon exemple sur le plan pédagogique pour mettre en œuvre des filtres prédictifs comme celui de Kalman ou des algorithmes d'apprentissage et de modélisation. D'autre part, un tel capteur peut permettre à un *animat* de s'endormir pour économiser son énergie ou se recharger et être réveillé si un élément extérieur le bouscule par exemple. L'accéléromètre utilisé est le LIS331DL de STMicroelectronics et celui-ci communique avec le microcontrôleur via un port SPI qui sera partagé avec l'ensemble des capteurs, utilisant ce protocole, ajoutés par l'utilisateur.

2.5 Déplacements

Une des principales différences de l'application *animat* par rapport à un réseau de capteurs sans-fil est la capacité qu'a chaque nœud à se déplacer. Néanmoins, il existe une grande diversité de moyens de déplacement et le nombre élevé d'entrées/sorties laissées à la disposition de l'utilisateur lui permette de choisir le plus adéquat pour son application. Néanmoins, de manière récurrente, l'utilisation de deux moteurs apparaît comme étant une des plus utilisée. Par exemple, les prototypes conçus utilisent deux modes de déplacement totalement différents mais utilisant tous deux des moteurs : des roues ou des tiges vibrantes. Par conséquent, nous avons dotés l'*animat* d'un circuit permettant de contrôler deux moteurs. Ces circuits sont des ponts en H et leur représentation schématique est proposée à la Figure 2-9.

Il existe des circuits intégrés conçus pour implémenter ces circuits comme le L293 de STMicroelectronics. Néanmoins, ceux-ci consomment généralement plus que la technique suivante : pour des moteurs de petites tailles dont la consommation est faible (jusqu'à 300 mA), il est plus intéressant d'utiliser quatre interrupteurs et de les connecter comme sur la Figure 2-9. Des circuits comme le ADG811 ou le MAX4678 intègrent quatre interrupteurs et des diodes internes protègent des surtensions. L'utilité de telles diodes, appelées diodes de roue libre, est d'absorber la surtension créée aux bornes de la bobine lorsque le courant est coupé.

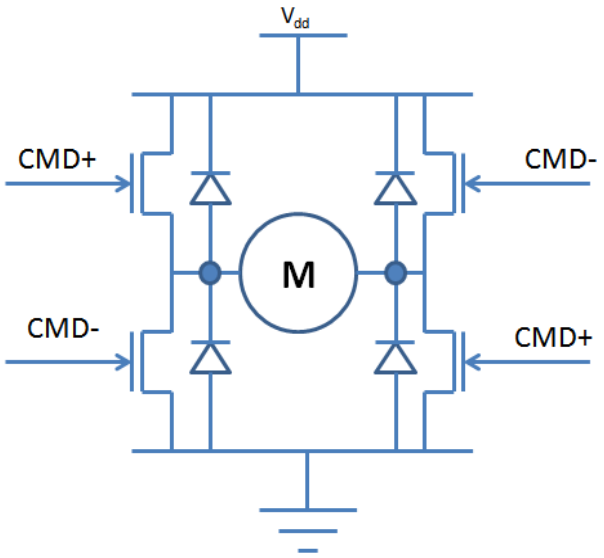


Figure 2-9. Schéma d'un pont en H

Ceci s'explique en rappelant l'expression de la tension aux bornes d'une bobine de réactance L:

$$U_L = L.\left(\frac{di_L}{dt}\right)$$

Lors de nos tests avec ce genre de circuits, nous avons relevé que la consommation du circuit est de l'ordre de 220 nA réduisant ainsi la consommation au repos de deux ordres de grandeur par rapport à un circuit spécialisé. Pour des raisons de diminution de consommation, nous avons choisi d'utiliser cette technique sur l'*animat*. Néanmoins, il est conseillé à l'utilisateur désireux d'utiliser des moteurs plus puissants d'ajouter une carte d'extension spécialisée pour la gestion des moteurs. On notera de plus la possibilité de couper l'alimentation du circuit de commande des moteurs à l'aide d'un transistor de commande. Par ailleurs, afin d'éviter les courts-circuits, de la logique supplémentaire a été ajoutée en entrée pour éviter à l'utilisateur de demander au moteur à tourner dans les deux sens en même temps. En considérant les temps de fermeture et d'ouverture des interrupteurs, toujours afin d'éviter les courts-circuits, l'utilisateur devra laisser un laps de temps entre la commande dans une direction puis dans l'autre. Le schéma final de contrôle des moteurs est alors représenté sur la Figure 2-10.

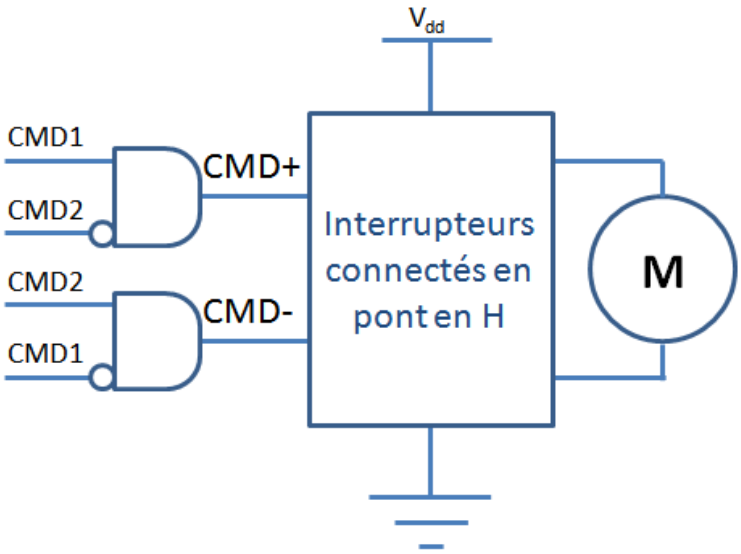


Figure 2-10. Schéma de contrôle d'un moteur

2.6 Énergie

Finalement, pour pouvoir alimenter l'*animat* nous avons opté pour une pile au Lithium-Ion de 1000 mAh de capacité. Le choix de ce type de pile est principalement dû à une masse relativement faible par rapport à la capacité de la pile ainsi qu'une taille raisonnable pour notre application. Ses principales caractéristiques sont résumées dans le Tableau 2.2.

Tableau 2.2. Caractéristiques de la batterie

Capacité	1000 mAh
Tension nominale	3.7 V
Dimensions	53 x 33 x 5.7 mm
Masse	22 g
Température	-25 à 60°C
Décharge Maximale	2C
Autodécharge	< 8% / mois

D'autre part, la capacité choisie permet d'atteindre une autonomie relativement élevée. En effet, en considérant les consommations moyennes de chaque sous-ensemble de l'*animat* telles que décrites dans le Tableau 2.3, on obtient une autonomie suffisamment longue pour des expérimentations de type apprentissage et collecte d'informations. Le Tableau 2.4 évalue des autonomies attendues pour différents cas d'utilisation des ressources à partir des données présentées dans le Tableau 2.3. L'autonomie est de l'ordre de ce que l'on trouve dans la littérature. Néanmoins dans notre cas, nous profitons de plusieurs périphériques nécessaires pour la mise en place d'applications de type *animat*.

Tableau 2.3. Consommation des grands ensembles de l'*animat*

Composant	Consommation Active (mA)	Consommation Veille (µA)	Consommation Veille profonde (µA)
Microcontrôleur	16 @ 32MHz	2.5 @ 2V	0.1
Radio	Émission : 23 Réception : 19	2	-
Moteurs vibratoires (pour les deux)	104	10 (Circuit de commande)	-
Moteur-Réducteurs (pour les deux)	100	10 (Circuit de commande)	-
Ensemble des capteurs	15 (seulement un actif à la fois, cf. multiplexage)	~ 1	-

Tableau 2.4. Estimation de l'autonomie pour différents cas d'utilisation sans recharge

Composant	Taux d'utilisation (%)					
	Cas 1	Cas 2	Cas 3	Cas 4	Cas 5	Cas 6 (WSN)
Microcontrôleur	100	100	100	75	10	1
Radio	100	100	20	20	10	1
Moteurs	100	75	75	50	10	0
Ensemble des capteurs	100	100	50	20	10	10
Autonomie	**5h 46min**	**6h 45min**	**9h 21min**	**13h 37 min**	**57h 45 min**	**46j 12h 46min**

Par ailleurs, bien que les batteries au lithium aient des caractéristiques très intéressantes pour des robots de taille réduite pour lesquels la masse embarquée est critique, celles-ci engendrent certaines contraintes. Notamment, la charge et la décharge de la batterie devraient être effectuées attentivement pour éviter tout risque d'explosion ou d'incendie. Pour ce faire, nous utilisons des batteries comportant des circuits de protection et ajoutons à cela un circuit dédié pour la charge comme représenté sur la Figure 2-11. De plus, pour limiter le courant débité nous utilisons un fusible de type PTC. Ces fusibles ont la particularité de pouvoir déconnecter la charge lorsque le courant est trop élevé et de la reconnecter lorsque la température redescend.

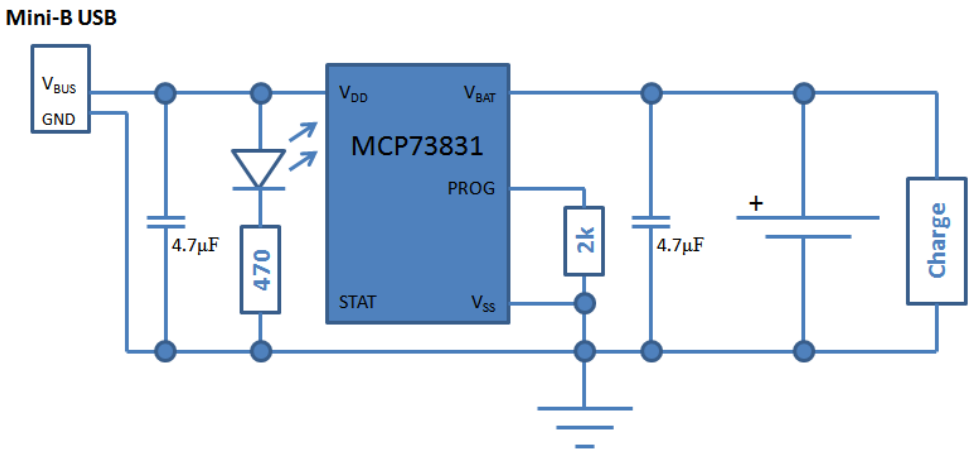
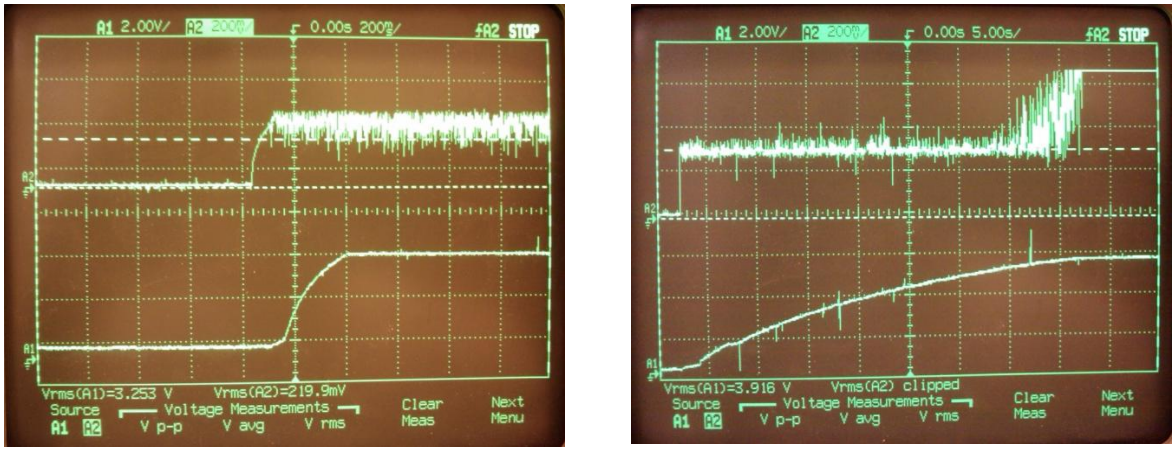


Figure 2-11. Circuit de recharge de la batterie

Enfin, la gestion des ressources énergétiques étant au cœur des problématiques de l'*animat*, nous l'avons doté d'un panneau solaire de petite taille. En pleine illumination solaire, ce panneau a à

ses bornes une tension de plus de 3V en circuit ouvert et débite un courant de 21mA en circuit fermé. Néanmoins, contrairement à une source de tension idéale, un panneau solaire a une impédance élevée. De ce fait, la tension aux bornes du panneau solaire dépend fortement de la charge qui lui est connectée. Il faudrait alors réaliser une adaptation d'impédance. D'autre part, la tension aux bornes du panneau est insuffisante pour charger une batterie au lithium. Par conséquent, nous utilisons un convertisseur DC-DC de type *step-up* avec adaptation de l'impédance vue par le panneau solaire. Ce circuit est le LTC3105. Lorsque la charge connectée est faible, pour maintenir la tension minimale aux bornes du panneau solaire, le circuit va réguler le courant au travers d'une bobine pour atteindre le transfert de puissance maximal. Par ailleurs, ce circuit a la particularité de fonctionner pour une tension d'entrée aussi basse que 250mV.

Des résultats expérimentaux sont montrés sur la Figure 2-12 où la tension du haut est celle aux bornes du panneau solaire, alors que celle du bas est celle aux bornes de la charge. Lors de nos tests, nous nous sommes aperçus qu'en environnement intérieur, le courant débité par le panneau solaire n'était pas suffisant pour charger une batterie tout en alimentant le reste du circuit. Sur le plan théorique, cela représente un avantage car l'*animat* devra « s'endormir » pour pouvoir recharger ses batteries. Par conséquent, nous pourrions aussi ajouter un super-condensateur qu'il serait possible de recharger en quelques secondes et pourrait servir de source d'alimentation temporaire.



(a) (b)

Figure 2-12. Exemples d'utilisation du LTC3105: (a) Tension de sortie sans charge, (b) Tension de sortie chargeant un super-condensateur

Par ailleurs, avec l'illumination intérieure normale, le courant débité par ce petit panneau solaire est relativement faible (environ 1 mA). Il sera donc nécessaire de positionner des zones de forte illumination pour que ces panneaux soient utilisables. Sur le plan théorique, cette contrainte est encore une fois intéressante et permet de forcer physiquement les *animats* en quête de nourriture à se déplacer pour trouver des sources d'énergie plutôt que de devoir le simuler. Le circuit résultant est alors représenté sur la Figure 2-13.

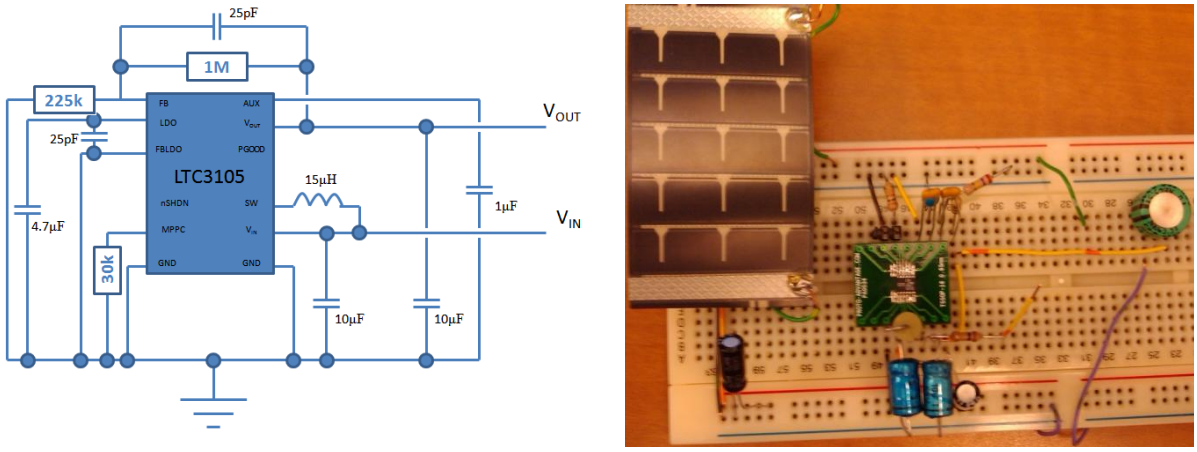


Figure 2-13. Circuit de recharge solaire

2.7 Produit Final

Finalement, le circuit proposé est relativement complexe et intègre plusieurs sous-ensembles répondant tous à une contrainte de l'application *animats* ou, de manière générale, d'applications où la prise en compte et la gestion des ressources énergétiques est primordiale. De ce fait, le prix est relativement élevé mais justifié par rapport aux attentes. Ci-dessous, nous fournissons dans le Tableau 2.5 une liste des principaux composants avec leur prix unitaire et leur prix de gros. À titre indicatif, lors de commandes réalisées pour la conception de 10 *animats*, le prix total de revient, incluant les PCB et les composants passifs (résistances et condensateurs non mentionnés dans le Tableau 2.5) était de l'ordre de 150$. Une commande plus volumineuse pourrait aisément faire chuter les coûts à une centaine de dollars. Par ailleurs, la Figure 2-14 représente la répartition du prix en fonctions de sous-ensembles caractéristiques. On constate alors que la motorisation et les capteurs occupent deux tiers du coût total des pièces de l'*animat* et que le reste

est occupé par les moyens de communication, de calculs et par les ressources énergétiques. Une réduction future du prix de l'*animat* devrait prendre en compte cette répartition des prix en faisant apparaître que, par exemple, alors que la motorisation représente un tiers du prix total, en termes de nombre de pièces cela représente qu'un pourcentage beaucoup plus faible. Ce genre d'exercice conduirait alors à l'adoption de moyen de locomotion moins coûteux. Finalement, le produit final est représenté sur la Figure 2-15. Les PCB utiles à cette conception sont fournis en ANNEXE 6 et en ANNEXE 7.

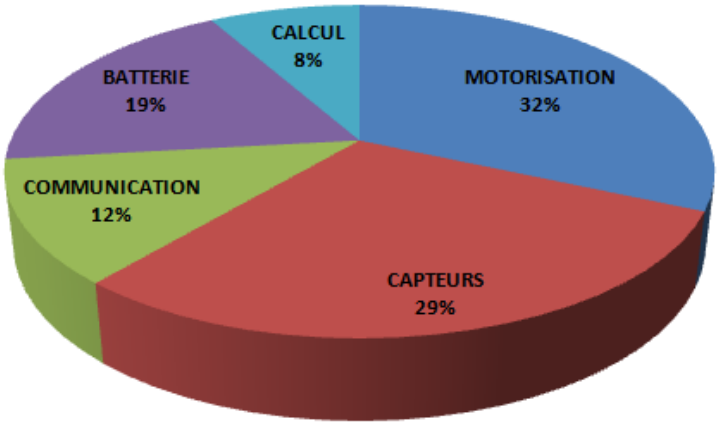


Figure 2-14. Répartition des prix

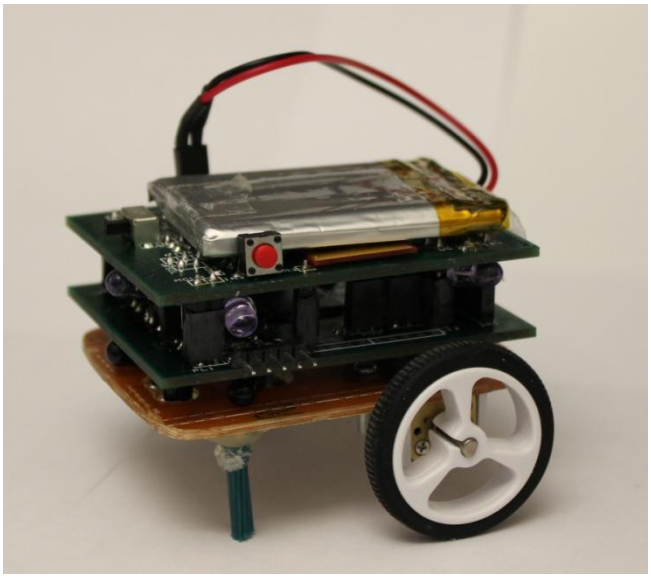


Figure 2-15. Photo de l'*Animat*

Tableau 2.5. Prix des principaux composants

Pièce	Fonction	Prix Unitaire	Prix de gros	Quantité	Totaux			
					M1 Unité	M2 Unité	M1 En gros	M2 En gros
MAX9634	Current sense	1,96	1,4804	6	**94,9**	**130,1**	**82,3**	**113,5**
LTC3105	Step-Up Conveter	6,17	6,17	1				
ADG811	Switch (pour H-bridge)	3,75	3,75	2				
MCP9700A	Temperature	0,41	0,332	1				
1080-1244	Light Sensor	0,65	0,504	1				
NC7SZ58	Porte Logique Configurable	0,48	0,345	4				
MCP73843	Li-Ion Controller	1,59	1,319	1				
25LC256	EEPROM	1,82	1,308	1				
102-1722	Micro Electret	1,2	0,929	1				
MRF24J40MA	Radio ZigBee	11,66	10,594	1				
TC1265	LDO	0,79	0,729	1				
PIC24FJ256GA108	Micro-contrôleur	7,96	6,262	1				
497-8327	Acceleromètre	3,35	2,999	1				
475-1080	Photo-transistor	0,36	0,292	10				
475-1468	Émetteur IR	0,59	0,474	8				
DG4051A	Multiplexeur 1:8	1,83	1,4468	2				
MC14067	Multiplexeur 1:16	1,59	1,2672	2				
455-2247	JST Connector	0,15	0,144	3				
CKN9560	Interrupteur	0,5	0,473	1				
ED2992CT	Mini-B USB	0,73	0,658	1				
TS27L2	Ampli-Op	0,23	0,23	3				
PRT-00339	Li-ion 1000mAH battery	11,95	10,76	1				
P14793CT	Moteur Vibratoire (M1)	4,34	4,111	2				
ROB-08896	Maintient moteur (M2)	4,95	4,46	1				
ROB-08901	Roues (M2)	6,95	6,26	1				
ROB-08910	Moteur Pololu 100:1 (M2)	15,95	14,36	2				

Néanmoins, l'utilisation d'un émetteur-récepteur compatible avec le protocole ZigBee nécessite une consommation importante aussi bien pour recevoir que pour émettre des messages comme le montre le Tableau 2.3. Il est donc primordial de profiter au maximum des modes de veille de ces modules. Pour cela, nous proposons dans les chapitres suivants de concevoir un circuit de réveil capable de déterminer quand réveiller l'émetteur-récepteur principal. Compte tenu que celui-ci est réalisé en logique asynchrone, il convient de présenter auparavant ce type de logique.

CHAPITRE 3 LOGIQUE ASYNCHRONE ET NCL™

3.1 Généralités

3.1.1 Intérêts pour les circuits asynchrones

Avec la diminution des dimensions des transistors et la difficulté de propager un signal d'horloge à travers tout un circuit sans souffrir de problème de déphasage d'horloge excessif, les concepteurs en microélectronique se posent de plus en plus la question de la viabilité de l'utilisation d'un tel signal de contrôle. Nombreux voient dans les circuits asynchrones une alternative concrète à ces problèmes que ce soit partiellement (GALS) ou totalement. L'avantage principal de tels circuits réside dans le fait que, sous-certaines conditions, ils assurent l'intégrité indispensable des signaux sans avoir à mettre en place des efforts importants pour propager correctement le signal d'horloge et sans avoir à souffrir d'une consommation dynamique élevée et parfois inutile. Avant d'entrer dans les détails, il est important de signaler que les circuits asynchrones, bien qu'ayant des propriétés attrayantes, ne sont pas la panacée pour toutes les applications. En effet, pour des applications intimement liées avec des intervalles de temps réguliers, comme l'échantillonnage de signaux usuel, l'utilisation d'un signal d'horloge faisant office de base temporelle semble nécessaire. Il est toutefois possible d'imaginer un rapprochement des travaux d'Emmanuel Candès sur le *Compressed Sensing* [30] et un circuit d'échantillonnage asynchrone [31].

3.1.2 Avantages des circuits asynchrones

Avant de poursuivre, nous donnons ici quelques-uns des avantages et inconvénients des circuits asynchrones par rapport aux circuits synchrones pour que le lecteur puisse se faire sa propre idée des exemples d'applications dans lesquels l'un ou l'autre devrait être utilisé. Tout d'abord, un des avantages le plus usité est celui selon lequel la consommation d'un circuit asynchrone est moindre. En effet, compte tenu du fait que les circuits asynchrones ne comportent pas de signal global oscillant à une fréquence élevée fixe tel un signal d'horloge, on peut espérer avoir une consommation dynamique plus faible. D'autre part, de tels circuits semblent plus efficaces en ce sens qu'il n'y aura pas de transitions des signaux internes lorsque ce n'est pas nécessaire. Ceci s'oppose radicalement au circuit synchrone où, en dehors des modes de veille ou d'autre cas

particulier, l'horloge oscille en permanence aussi bien pour réaliser des opérations arithmétiques et logiques ou autres accès à la mémoire que lorsqu'un processeur attend par exemple. Ainsi les circuits asynchrones paraissent plus efficaces sur le plan énergétique. Néanmoins, il faut être prudent quant à l'affirmation absolue concernant l'efficacité énergétique. En effet, comme on le verra plus loin, les circuits asynchrones ont tendance à utiliser plus de ressources. De ce fait, pour une technologie donnée, essentiellement à cause des fuites, les circuits asynchrones accuseront une consommation statique plus élevée. Par conséquent, ils seront plus efficaces sur le plan énergétique dès lors que le surcoût en consommation statique de ces derniers ne contrebalance pas le gain en consommation dynamique. En plus des justifications purement physiques, au niveau architectural, le fait de pouvoir se passer d'un signal d'horloge permet de se séparer des modules de génération et de propagation de l'horloge réduisant ainsi la consommation totale. D'autre part, en ce qui concerne les performances, les circuits synchrones sont limités par le chemin critique et donc le pire cas. Comme on le verra plus loin, certains types de designs asynchrones ont l'avantage de pouvoir s'adapter aux délais dans les fils. En d'autres termes, chaque donnée est propagée avec le délai propre au chemin par lequel elle passe et n'a pas à être contrainte par un timing global comme l'impose la fréquence d'horloge. On passe alors d'un schéma de type pire cas à un schéma de type cas moyen ce qui se traduit par un gain en performance. Ces mêmes circuits ont aussi l'énorme avantage de pouvoir s'adapter aux changements des variables d'environnements (température, source de tension, fabrication…) dès lors que celles-ci se traduisent par une variation de délai. Cette propriété a au moins deux impacts majeurs : tout d'abord elle permet de réduire le prix total d'une solution en ce sens que l'on peut diminuer la qualité du régulateur de tension voire l'omettre par exemple ou même diminuer la qualité des tests de fabrication. L'autre intérêt réside dans le fait que la consommation globale diminue aussi avec la diminution des composants externes. De plus, utiliser de la logique asynchrone améliore la réutilisabilité et la modularité car les protocoles de communication entre modules ainsi que ceux de validité de l'intégrité des données (*handshaking protocol*) sont naturellement asynchrones dans de tels circuits. Enfin, en évitant d'avoir l'ensemble du circuit qui commute à une fréquence donnée mais en privilégiant des traitements locaux, les circuits asynchrones proposent une caractéristique intéressante de réduction des interférences électromagnétiques (EMI).

3.1.3 Inconvénients des circuits asynchrones

En contrepartie, les inconvénients se résument principalement à un certain scepticisme et une absence de volonté de changement de la part de l'industrie microélectronique mais pas seulement. En effet, un tel changement nécessiterait d'une part de former des ingénieurs, déjà très expérimentés dans les circuits synchrones, à de nouvelles techniques de design parfois déroutantes. D'autre part, c'est toute l'industrie qu'il faut entraîner dans cette aventure notamment les développeurs d'outils de design qui à l'heure actuelle semblent assez timides. En dehors de ces aspects financiers, les designs asynchrones possèdent des inconvénients propres à leur nature : la vérification de ces circuits est beaucoup plus complexe, la surface occupée pour une fonctionnalité et une technologie données peut être beaucoup plus grande et dans certains cas spécifiques, les performances peuvent être dégradées.

3.2 Types de design

C'est donc dans ce contexte et avec ces caractéristiques que les circuits asynchrones tentent de trouver leur place dans le monde de la microélectronique. Les techniques de design asynchrones peuvent être classées en quatre grandes catégories que nous allons détailler subséquemment. Celles-ci se différencient notamment par des hypothèses temporelles plus ou moins fortes et de ce fait une robustesse plus ou moins élevée. Graphiquement, on pourrait les représenter comme sur la Figure 3-1.

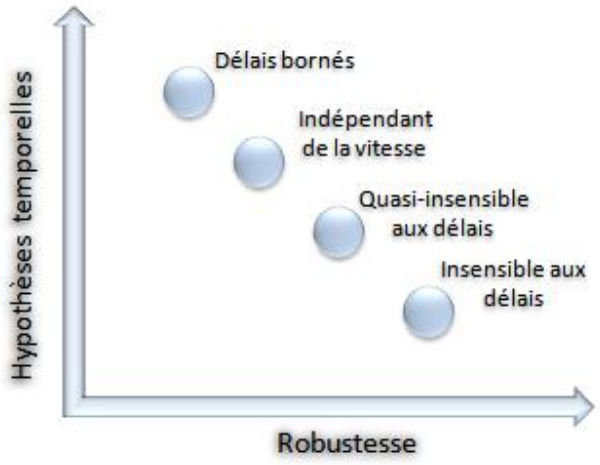


Figure 3-1. Types de designs asynchrones

3.2.1 Les circuits insensibles aux délais (DI)

Cette première classe de circuits asynchrones regroupe les circuits qui ne font aucune hypothèse sur les délais. Bien que ces circuits puissent paraître les plus robustes, ils n'en demeurent pas moins purement théoriques ou très limités en pratique. En effet, Alain Martin [32] montra qu'un tel cadre théorique se traduisait en pratique par la seule utilisation d'éléments C de Müller à une sortie et plusieurs entrées. La Figure 3-2 donne la représentation ainsi que la table de vérité pour une telle porte à deux entrées.

(a)

A	B	S(n)
0	0	0
0	1	S(n-1)
1	0	S(n-1)
1	1	1

(b)

Figure 3-2. Élément C de Müller: (a) Schéma de la porte (b) Table de vérité

3.2.2 Les circuits quasi-insensibles aux délais (QDI)

Alors que les circuits DI restent très limités dans leur utilisation pratique, les circuits quasi-insensibles aux délais proposent d'assouplir légèrement les hypothèses temporelles pour pouvoir se défaire des limitations précédentes. Contrairement au premier cas pour lequel aucune hypothèse sur les délais n'était faite, ici des contraintes sur certaines fourches du circuit sont imposées. Cette hypothèse s'appelle en anglais l'*isochronic fork* ou fourche isochrone.

On trouve souvent à tort dans la littérature la définition simplifiée de l'hypothèse sur de telles fourches comme ayant des délais égaux. D'autres se risquent même à dire que cela serait équivalent à négliger les délais dans les fils à l'intérieur de modules de base. Cette dernière tend à devenir de plus en plus fausse avec la diminution des dimensions dans les circuits intégrés. Et bien qu'elle puisse sembler acceptable dans les circuits ASIC où le routage peut être contrôlé localement, ceci n'est plus nécessairement vrai dès lors que le routage est automatisé ou dans un environnement FPGA. En réalité, une définition plus juste et respectant celle donnée dans [32] serait de dire que :

> Pour les fils se divisant en plusieurs branches, si une transition sur l'une des branches a induit un effet visible au niveau des sorties et que cet effet a reçu un accusé de réception alors on suppose que dans le même temps la transition sur les autres branches est aussi survenue.

En respectant cette contrainte, on s'assure de la validité et de l'intégrité des données et on évite la limitation imputée aux circuits DI. Parmi les circuits QDI on compte notamment les circuits de David [33], la proposition de Sparso [34] d'utiliser des boucles ce qui peut par la suite s'adapter aux autres techniques pour permettre un certain séquencement des tâches, les travaux de Singh[35] ou encore le NULL Convention Logic [36, 37].

3.2.3 Les circuits indépendants de la vitesse (SI)

Pour cette classe de circuits asynchrones, l'hypothèse temporelle est encore plus forte en ce sens qu'ici les délais dans les fils seront tout bonnement négligés. Comme nous l'avons mentionné plus haut, ce type d'hypothèse devrait se limiter à des circuits très spécifiques sur lesquels le concepteur a un contrôle à très bas niveau. Contrairement aux circuits QDI, ici c'est l'ensemble des branches sur lequel des hypothèses sont émises.

3.2.4 Les circuits à délais bornés

Finalement, la dernière classe de circuits asynchrones rassemble l'ensemble des circuits pour lesquels l'hypothèse temporelle se résume à dire que les délais dans les parties combinatoires sont connus et bornés. On peut alors rajouter des délais physiques sur les lignes de synchronisation. Un des gros désavantages de tels circuits est la limitation de la performance au pire cas. En effet, le délai dans la boucle de retour, comme sur la Figure 3-3, est fixé et égal au pire temps dans la partie combinatoire correspondante. La Figure 3-3 est une vue schématique des éléments mis en jeu et ne rend pas compte de l'agencement judicieux des éléments C de Müller dans la partie de synchronisation entre les registres, comprenant notamment le délai. Le gain par rapport à des circuits synchrones c'est que le pire cas est calculé localement contrairement au pire cas global imposé par le signal d'horloge. À l'inverse, contrairement aux circuits QDI qui nécessitent une représentation des données particulières, comme on le verra dans le cas particulier du NCL détaillé dans la partie 3.4, les circuits à délais bornés peuvent se

contenter de réutiliser les parties combinatoires traditionnelles avec une représentation booléenne. Parmi cette classe de circuits on compte les *micropipelines* de Sutherland [38].

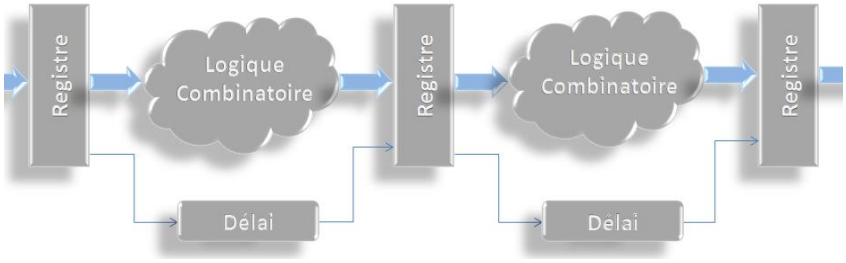


Figure 3-3. Schéma du protocole de type délais bornés

3.3 Choix du type de design

Pour la suite de ce projet, nous avons choisi d'utiliser les circuits asynchrones de types QDI pour les raisons exposées ci-après. Comme nous le verrons dans le Chapitre 5, lorsque nous serons amenés à concevoir un module répondant à un besoin spécifique, nous opterons pour une réalisation sur FPGA. Les choix d'un tel support seront détaillés dans les parties correspondantes. Bien qu'ayant une multitude d'avantages par rapport à notre besoin, l'utilisation de FPGA nous empêche d'utiliser certains types de design asynchrone ou bien rend la tâche plus complexe. Tout d'abord, les circuits DI peuvent être écartés car nous avons vu que leur champ d'application était trop limité. D'autre part, dans un tel environnement, l'hypothèse selon laquelle les délais dans les fils sont négligeables se révèle être fausse. Pour justifier l'affirmation précédente, un exemple de mesures prenant en compte les délais de placement/routage sera donné dans la sous-partie 5.4.3 du Chapitre 5 concernant la faisabilité. Enfin, générer des délais précis dans un tel environnement n'est pas chose aisée. En effet, compte tenu des variations dans le placement/routage et l'impact de ces variations sur les délais, une telle approche, bien que réalisable, demande un effort élevé de la part du concepteur. À cela s'ajoute le fait que pour générer des délais précis, il est nécessaire d'utiliser une quantité non-négligeable de ressources. Par conséquent, les circuits QDI semblaient s'imposer d'eux-mêmes.

Parmi l'ensemble des techniques proposées, nous avons choisi le NULL Convention Logic car il fait partie des techniques de design les plus utilisées et documentées mais aussi car il facilite la conception et, en définissant un ensemble de fonctions de base, permet des optimisations que les autres techniques ne permettent pas. Par conséquent, il convient de présenter le NCL.

3.4 NULL Convention Logic

3.4.1 Présentation

Le NCL est un paradigme qui regroupe des règles de design permettant la conception de circuits asynchrones. Une des volontés de ce dernier est de clairement détailler les tenants et aboutissants qui permettent de concevoir des circuits asynchrones fiables. L'autre dessein du NCL est de faciliter et de promouvoir l'accès au monde, parfois obscur, de l'asynchrone en définissant un ensemble de règles de conception simples. Ces règles découlent directement du fait que, comme mentionné antérieurement, le NCL se situe dans la classe des circuits asynchrones de type QDI. Les faibles hypothèses temporelles impliquent un ensemble de mesures nécessaires pour connaître la validité d'un résultat et assurer son intégrité.

Une des premières étapes est de comprendre que la représentation booléenne des nombres telle que nous la connaissons n'est pas complète. C'est à dire qu'en l'absence d'un signal de synchronisation on ne peut pas savoir si le résultat que l'on serait amené à lire est valide ou non. En effet, c'est exactement le rôle de l'horloge pour les circuits synchrones que de définir un signal global indiquant à chaque étage que l'étage précédent lui fournit des données valides. Pour les circuits asynchrones ce signal global est supprimé au profit de synchronisation locale. Pour les circuits à délais bornés, cela est immédiat car le délai indique localement le temps à partir duquel les données en entrée d'un registre peuvent être prises en compte. Pour les circuits QDI, les délais peuvent être supposés infinis sans que cela ne doivent compromettre le fonctionnement d'un module. Il est donc nécessaire que les données elles-mêmes comportent l'information de validité. La représentation des données doit par conséquent être changée. Une explication détaillée du choix de telles représentations est disponible dans [39]. Plusieurs représentations complètes ou insensibles aux délais peuvent être imaginées. Parmi ces représentations, on compte la représentation sur deux fils (*dual rail*), sur quatre fils (*quad rails*) et l'ensemble des représentations de types 1 fils sur N (*1-out-of-*N) et d'autres représentations appelées MEAG. L'avantage des trois premières réside dans le fait qu'elles sont optimales du point de vue du nombre de bits physiques pour représenter un bit logique. La Figure 3-4 décrit les représentations sur deux et quatre fils. Ce sont notamment les deux représentations utilisés avec le NCL. Par la suite nous privilégierons la représentation sur deux fils pour des raisons de réduction de la complexité. Il faut néanmoins remarquer que la représentation sur quatre fils est susceptible de

permettre une réduction de la consommation dynamique en ce sens que le nombre de bits qui commutent pour représenter un même nombre est inférieur.

Signification	Bit 1	Bit 0
NULL	0	0
'0' Valide	0	1
'1' Valide	1	0
Interdit	1	1

(a)

Signification	Bit 3	Bit 2	Bit 1	Bit 0
NULL	0	0	0	0
'0' Valide	0	0	0	1
'1' Valide	0	0	1	0
'2' Valide	0	1	0	0
'3' Valide	1	0	0	0
Interdit	*Les autres combinaisons*			

(b)

Figure 3-4. Représentations complètes des données: (a) *Dual-railI*, (b) *Quad-rail*

Changer de représentation implique un nouveau raisonnement lors du design qui peut parfois sembler contre-intuitif ou très différent de la logique booléenne traditionnelle. Notamment, il n'est plus question d'utiliser une porte pour inverser un signal, il suffit d'affecter au *Bit0* de la sortie la valeur du *Bit1* de l'entrée et de même le *Bit1* de la sortie reçoit le *Bit0* de l'entrée. La Figure 3-5 donne les tables de vérité de quelques portes logiques standard.

	0	1	N
0	0	0	N
1	0	1	N
N	N	N	N

(a)

	0	1	N
0	0	1	N
1	1	1	N
N	N	N	N

(b)

	0	1	N
0	1	1	N
1	1	0	N
N	N	N	N

(c)

0	1
1	0
N	N

(d)

Figure 3-5. Tables de vérités de portes logiques standard en représentation sur deux fils: (a) ET, (b) OU, (c) NON-ET et (d) INV

En utilisant cette représentation, il est alors possible de retrouver l'ensemble des fonctions logiques que nous connaissons. Le NCL définit toutefois un ensemble de 27 portes standard que nous détaillerons dans la sous-partie 3.4.2.

Un des grands défis de tels circuits asynchrones consiste à se prémunir des aléas pouvant survenir dans les fils. En effet, comme nous le verrons dans la sous-partie 3.4.3 sur le protocole de communication du NCL, à la différence des circuits synchrones ou chaque signal peut passer par des états intermédiaires avant de se stabiliser, pour les circuits QDI chaque valeur des signaux de sorties est considérée comme étant une valeur ayant un sens. À des fins de prévention de ces états transitoires, le NCL, en plus de l'hypothèse de fourches isochroniques, impose deux règles qu'il est nécessaire de respecter. Ces deux règles sont décrites ci-après. On désignera dans le reste de ce document par symbole tout ensemble de deux fils physiques constituant les deux bits utilisés pour la représentation en *dual-rail*. De plus, un groupe de N symboles sera désigné comme étant un symbole de N bits et sera constitué de 2*N fils. Chaque symbole peut alors prendre la valeur NULL (les deux fils au zéro logique) ou la valeur DATA (un seul des deux fils est actif). En reprenant les notations de la Figure 3-4, dans le cas où le *bit0* est actif, on parle de DATA0 qui représente un « zéro valide », et dans le cas où le *bit1* est actif, on parle de DATA1 qui représente un « un valide ».

Complétude par rapport aux entrées :

- Par rapport au type DATA:

  L'ensemble des sorties ne peut pas passer du type NULL à DATA si l'ensemble des entrées n'est pas de type DATA.

- Par rapport au type NULL:

  L'ensemble des sorties ne peut pas passer du type DATA à NULL si l'ensemble des entrées n'est pas de type NULL.

Autrement dit, une certaine forme de synchronisation est imposée en ce sens que le type des sorties doit correspondre au type de l'ensemble des entrées. La synchronisation vient du fait que, comme nous le verrons dans la sous-partie 3.4.3, le protocole de communication proposé par le NCL implique l'attente du passage de la sortie d'une partie combinatoire d'un type à un autre.

Observabilité :

Aucun signal *orphelin* ne peut traverser une porte à l'intérieur de la logique contenue entre deux rangs de registres. Ceci implique que chaque transition de portes est visible sur au moins une des sorties. On qualifiera d'*orphelin* tout signal qui ne joue aucun rôle dans la détermination d'une sortie. Cette propriété n'est pas intrinsèque à un signal, qui n'aurait alors aucune utilité et pourrait être supprimé, mais dépend de l'ensemble des symboles fournis à l'entrée du module considéré.

3.4.2 Modules de base

En plus des deux règles énoncées précédemment, le NCL définit un ensemble de 27 portes de base dont l'agencement sert à définir des modules plus complexes à l'image des cellules standard des bibliothèques utilisées pour la conception de circuit VLSI. Ces 27 portes sont la réalisation de l'ensemble des fonctions positives de deux à quatre entrées. Par la suite on appellera fonction positive toute fonction dont l'activation d'une entrée ne peut entraîner que l'activation de la sortie ou la laisser inchangée. De même, la désactivation d'une entrée ne peut entraîner que la désactivation de la sortie ou la laisser inchangée. Le Tableau 3.1 donne la définition des fonctions d'activation de ces 27 portes. Il est important de noter que ces portes sont des portes à hystérésis ou à effet mémoire dans le sens où celles-ci sont activées par les fonctions décrites par le tableau susmentionné mais que leur désactivation est occasionnée par la désactivation de l'ensemble des données d'entrées tout comme l'élément C de Müller déjà mentionné. Ici, les entrées dénommées par {A, B, C, D} dans le Tableau 3.1 font référence à des bits et non à des symboles.

La limitation à seulement quatre entrées provient vraisemblablement de la volonté de définir une bibliothèque de portes définie au niveau transistor. En technologie CMOS l'empilement de transistors limite le nombre d'entrées pour respecter les contraintes sur les tensions de seuil. On remarquera de plus que la porte TH22 n'est autre que l'élément C de Müller.

Le NCL définit sa propre symbolique et ses propres notations pour chacune des portes. À l'exception des portes THXOR0, THAND0 et TH24COMP la notation est la suivante pour une porte à N entrées: $TH_{MNW_{w_1 w_2 \ldots w_K}}$. M est le seuil d'activation, autrement dit le nombre d'entrées qui doivent être actives pour que cet état se propage à la sortie. Les coefficients $W_1 W_2 \ldots W_K$ définissent les poids des entrées 1 à K. Les N-K entrées non représentées ont un poids égal à 1. Il

est alors aisé de faire le rapprochement entre un réseau de portes NCL et un réseau de neurones avec fonction d'activation à seuil.

Tableau 3.1. Définitions des 27 portes du NCL

PORTES NCL	EQUIVALENT BOOLEEN
TH12	*A + B*
TH22	*AB*
TH13	*A + B + C*
TH23	*AB + AC + BC*
TH33	*ABC*
TH23W2	*A + BC*
TH33W2	*AB + AC*
TH14	*A + B + C + D*
TH24	*AB + AC + AD + BC + BD + CD*
TH34	*ABC + ABD + ACD + BCD*
TH44	*ABCD*
TH24W2	*A + BC + BD + CD*
TH34W2	*AB + AC + AD + BCD*
TH44W2	*ABC + ABD + ACD*
TH34W3	*A + BCD*
TH44W3	*AB + AC + AD*
TH24W22	*A + B + CD*
TH34W22	*AB + AC + AD + BC + BD*
TH44W22	*AB + ACD + BCD*
TH54W22	*ABC + ABD*
TH34W32	*A + BC +BD*
TH54W32	*AB + ACD*
TH44W322	*AB + AC + AD + BC*
TH54W322	*AB + AC + BCD*
THXOR0	*AB + CD*
THAND0	*AB + BC + AD*
TH24COMP	*AC + BC + AD + BD*

Comme nous le mentionnions plus haut, les portes NCL sont des portes à mémorisation de l'état. Autrement dit, une fois la sortie activée, la désactivation de cette dernière ne survient que lorsque l'ensemble des entrées est désactivé. La Figure 3-6 fournit la représentation et le chronogramme de la porte TH33 pour un jeu d'entrées particulier afin d'illustrer cette caractéristique.

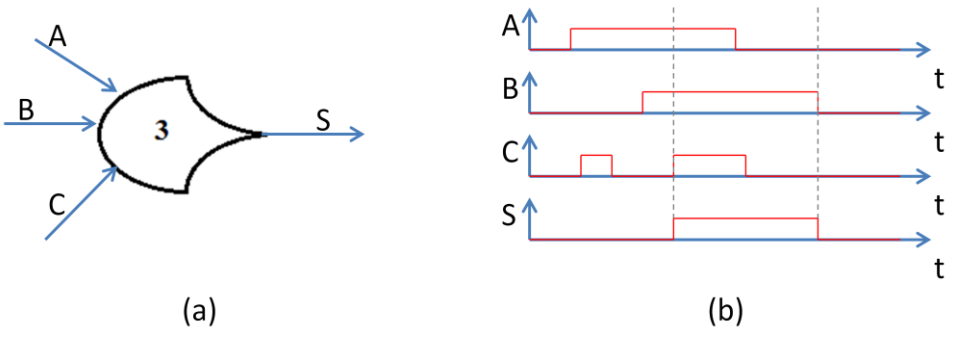


Figure 3-6. Exemple de la porte TH33: (a) Schéma (b) Chronogramme

Hormis pour les portes de la forme TH1N, cette propriété de mémorisation se traduit physiquement par l'ajout d'un élément de mémoire. En effet, ces dernières ont une fonction d'activation telle que la sortie reste active tant qu'au moins une des entrées est active. Par conséquent un élément de mémoire serait superflu. Pour les autres, il est possible de procéder de plusieurs manières pour parvenir à mémoriser l'état de la sortie : on peut soit ajouter un élément de mémoire ou bien inclure des boucles de rétroaction. La première solution peut se traduire dans un environnement ASIC par l'ajout de deux inverseurs tête-bêche dont l'un est faible comme représenté sur la Figure 3-7a ou bien, dans un environnement FPGA, par l'utilisation de loquets comme représenté sur la Figure 3-7b.

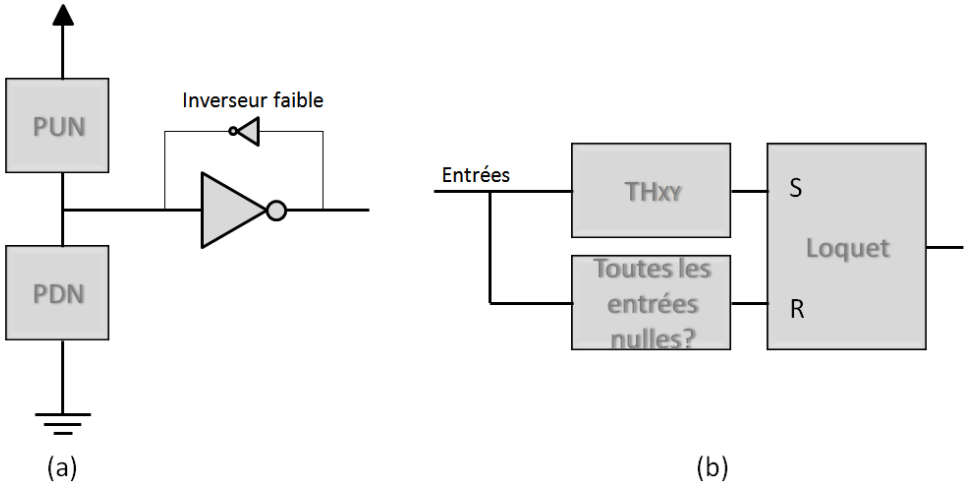


Figure 3-7. Éléments de mémoire pour les portes à hystérésis

Bien que schématique, on peut d'ores et déjà supposer au vu de la Figure 3-7 qu'une implémentation sur FPGA sera gourmande en termes de ressources. On notera que l'exemple de la Figure 3-7a représente une version semi-statique de la mémorisation. Le lecteur intéressé pourra trouver son pendant en version statique dans [37].

L'autre alternative est d'utiliser des boucles de rétroaction bien que cela soit généralement déconseillé dans des environnements pour lesquels les délais dans les fils sont dépendants du placement/routage. Un exemple est donné pour la porte TH33 dans la Figure 3-8.

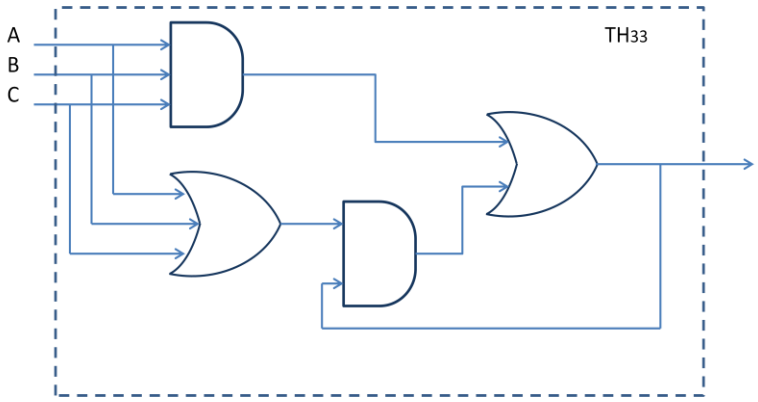
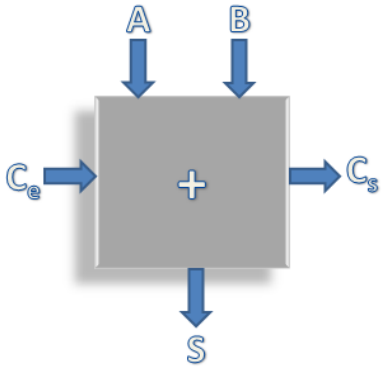


Figure 3-8. Porte TH_{33} avec rétroaction

L'idée à l'origine de l'utilisation de portes à hystérésis réside dans le fait qu'en utilisant de telles portes, la complétude par rapport aux entrées de type NULL est assurée localement. En effet, chaque porte s'assure que sa sortie se désactive uniquement quand l'ensemble de ses entrées est de type NULL. De ce fait, par application de cette règle de proche en proche, on s'assure que le vecteur de sortie passera de DATA à NULL uniquement si l'ensemble des entrées est de type NULL. Il n'en demeure pas moins que le concepteur doit encore s'assurer de la complétude par rapport aux entrées de type DATA et de l'observabilité de chacun de ces modules. Pour illustrer ce genre de contraintes, étudions la conception d'un additionneur 1-bit avec retenue ou *full-adder* comme représenté sur la Figure 3-9. Les notations employées sont A, B et C_e pour les deux entrées à additionner et la retenue d'entrée puis S et C_s pour le résultat et la retenue de sortie. Les valeurs de type NULL ont été omises pour ne pas surcharger inutilement la table de vérité.

C_e	A	B	S	C_s
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

(a)



(b)

Figure 3-9. Additionneur 1-bit: (a) Table de vérité, (b) Symbole

Pour concevoir un module en logique NCL, on commence d'abord par trouver la fonction booléenne d'un tel module. Dans ce cas on a les équations suivantes :

$$S = A \oplus B \oplus C_e$$

$$C_s = A.B + A.C_e + B.C_e$$

La première étape consiste à changer de représentation et à passer, dans notre cas, à une représentation sur deux fils. Avec cette représentation, à chaque grandeur booléenne X on associe une nouvelle représentation représenté par $\{X_0, X_1\}$ où ces deux grandeurs prennent des valeurs telles qu'explicitées dans la Figure 3-4 concernant cette représentation. On obtient alors l'ensemble de quatre équations suivant :

$$S_1 = A_1 \oplus B_1 \oplus C_{e1} = A_1B_0C_{e0} + A_0B_1C_{e0} + A_0B_0C_{e1} + A_1B_1C_{e1}$$

$$S_0 = \overline{A_1 \oplus B_1 \oplus C_{e1}} = A_1B_1C_{e0} + A_0B_1C_{e1} + A_1B_0C_{e1} + A_0B_0C_{e0}$$

$$C_{s1} = A_1B_1 + A_1C_{e1} + B_1C_{e1}$$

$$C_{s0} = \overline{A.B + A.C_e + B.C_e} = A_0B_0 + A_0C_{e0} + B_0C_{e0}$$

Dans cette exemple, on peut vérifier que chacune des équations est complète par rapport aux entrées de types DATA. En effet, il suffit de s'assurer que pour des entrées non toutes égales à DATA, l'ensemble des sorties ne peut être égal à DATA. Une démonstration rigoureuse pourrait se faire en regardant les dépendances de l'ensemble des sorties par rapport aux entrées et s'assurer que chacune des entrées (bit_0 ou bit_1) joue un rôle déterminant dans l'évaluation de celles-ci. Prenons un cas particulier, pour les entrées $\{A, B, C_e\}$ passant de {NULL, NULL, NULL} à {DATA1, DATA1, NULL} définissant un ensemble d'entrées qui, par définition, ne contient pas que des éléments de type DATA. Ce faisant, selon le principe de complétude par rapport aux entrées de type DATA, on devrait s'attendre à un ensemble de sortie pour lequel l'ensemble de ces éléments ne sont pas de type DATA. Dans ce cas-ci, on obtiendrait :

$$\{S, C_s\} = \{\text{NULL, DATA1}\}$$

Ce qui respecte donc bien la règle. On constate que plusieurs des sorties peuvent être de type DATA lorsque l'ensemble d'entrées ne contient pas que des éléments de type DATA sans violer la règle de complétude dès lors qu'au moins une des sorties reste à NULL. La règle de complétude pour des entrées de type NULL est quant à elle respectée en supposant l'utilisation de portes à hystérésis. À ce propos, les dernières étapes consistent en la traduction des équations en termes de portes logiques (*mapping*) et en la vérification de la règle d'observabilité. On rappelle

ici que ces équations devront être réalisées en utilisant les 27 portes définies par le NCL. Une réalisation naïve dans ce cas-ci consisterait alors à remplacer chaque ET à trois ou deux entrées par une portes TH_{33} ou TH_{22}, chaque OU à quatre ou trois entrées par une portes TH_{14} ou TH_{13}. Néanmoins cette solution mène à une utilisation excessive des ressources et donc sous-optimale. Une autre façon de faire et de remarquer que :

$$S_1 = C_{s0}.(A_1 + B_1 + C_{e1}) + A_1 B_1 C_{e1}$$

$$S_0 = C_{s1}.(A_0 + B_0 + C_{e0}) + A_0 B_0 C_{e0}$$

Car pour toute entrée X, on a $X_0.X_1 = 0$. On peut alors établir les relations portes – équations en se référant au Tableau 3.1 en écrivant de la sorte :

$$C_{s1} = TH_{23}(A_1, B_1, C_{e1})$$

$$C_{s0} = TH_{23}(A_0, B_0, C_{e0})$$

$$S_1 = TH_{34W2}(C_{s0}, A1, B1, C1)$$

$$S_0 = TH_{34W2}(C_{s1}, A0, B0, C0)$$

Cette configuration-là utilise seulement quatre portes NCL comme représenté sur la Figure 3-10. Néanmoins, en utilisant en cascade deux portes, on introduit des signaux orphelins notés k_1 et k_2. Il est donc nécessaire de s'assurer que ceux-ci ne violent pas la règle d'observabilité. En effet, lorsque k_1 ou k_2 devient actif, celui-ci n'est pas directement vu à la sortie. Il faut donc se demander s'il est possible que l'un ou l'autre de ces signaux soit actif sans que cela n'ait d'impact sur la sortie. Le problème étant totalement symétrique, nous nous contenterons d'analyser le cas où k_1 passe à 1. Dans ce cas-là, il est possible que S_1 reste inactif. Néanmoins, k_1 est directement égal à C_{s0}, par conséquent si le signal k_1 est actif, celui-ci est vu à la sortie et par conséquent ce signal n'est pas orphelin. Bien que ces étapes puissent paraître fastidieuses, elles sont très facilement automatisables comme le propose le projet UNCLE [40].

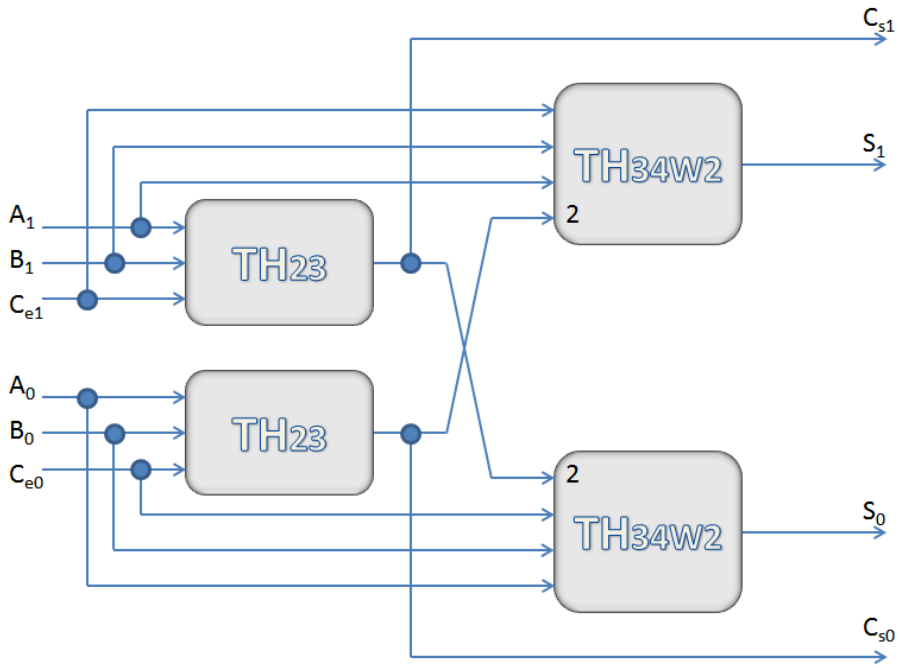


Figure 3-10. Additionneur NCL 1-bit avec retenue d'entrée

3.4.3 Protocole de communication

Pour que l'intégrité des données puisse être assurée le NCL propose de respecter un certain *handshake* protocole ce qui est courant, sinon nécessaire, pour les circuits de types QDI. Celui-ci demande à chaque étage d'attendre que l'ensemble de ses sorties soit de type NULL pour pouvoir accepter un nouvel ensemble d'entrées de type DATA. De manière complémentaire, chaque étage doit informer l'étage précédent du bon enregistrement des dernières valeurs calculées et de la possibilité de recevoir un nouvel ensemble d'entrées de type NULL. Le protocole de communication se décompose alors en deux fronts : un front de type NULL et un autre de type DATA. L'agencement des étages est alors très similaire au pipeline des circuits synchrones mis à part le fait que le signal de synchronisation soit localement généré par chaque étage et dépende de l'activité des données. La Figure 3-11 représente un schéma typique de plusieurs étages.

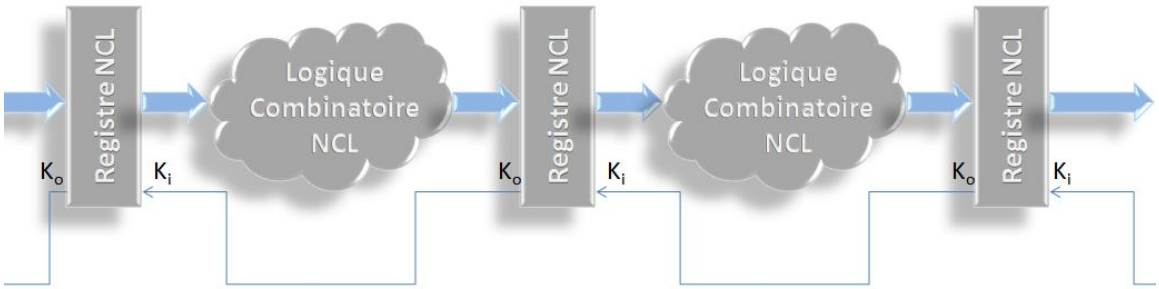


Figure 3-11. Pipeline NCL

Les parties combinatoires sont réalisées en utilisant de la logique telle que présentée dans la sous-partie précédente. Chacune de ces parties est entourée par des registres asynchrones qui sont le cœur de la régulation des flux de données. Un tel registre peut être réalisé en utilisant plusieurs éléments C de Müller ou portes TH_{22} comme représenté sur la Figure 3-12 ainsi qu'un circuit capable de calculer le statut de l'ensemble de sortie : DATA ou NULL. C'est cette information propagée vers l'étage précédent qui va contribuer à la régulation du flux de données. La Figure 3-12 ne représente pas le signal de *reset* qui initialise les registres à l'état NULL ou bien à une autre valeur de type DATA prédéfinie. De plus, la porte $TH_{(N)(2N)}$ n'est pas une porte définie par le NCL mais symbolise ici l'association de plusieurs portes NCL permettant de savoir quand, pour toutes les sorties, un des deux fils est à un. Autrement dit, ce circuit fournit une sortie égale à 1 lorsque l'ensemble des sorties est de type NULL et une sortie égale à 0 lorsque l'ensemble des sorties du registre est de type DATA. C'est cette information qui va permettre à chaque registre de se synchroniser localement avec ses voisins.

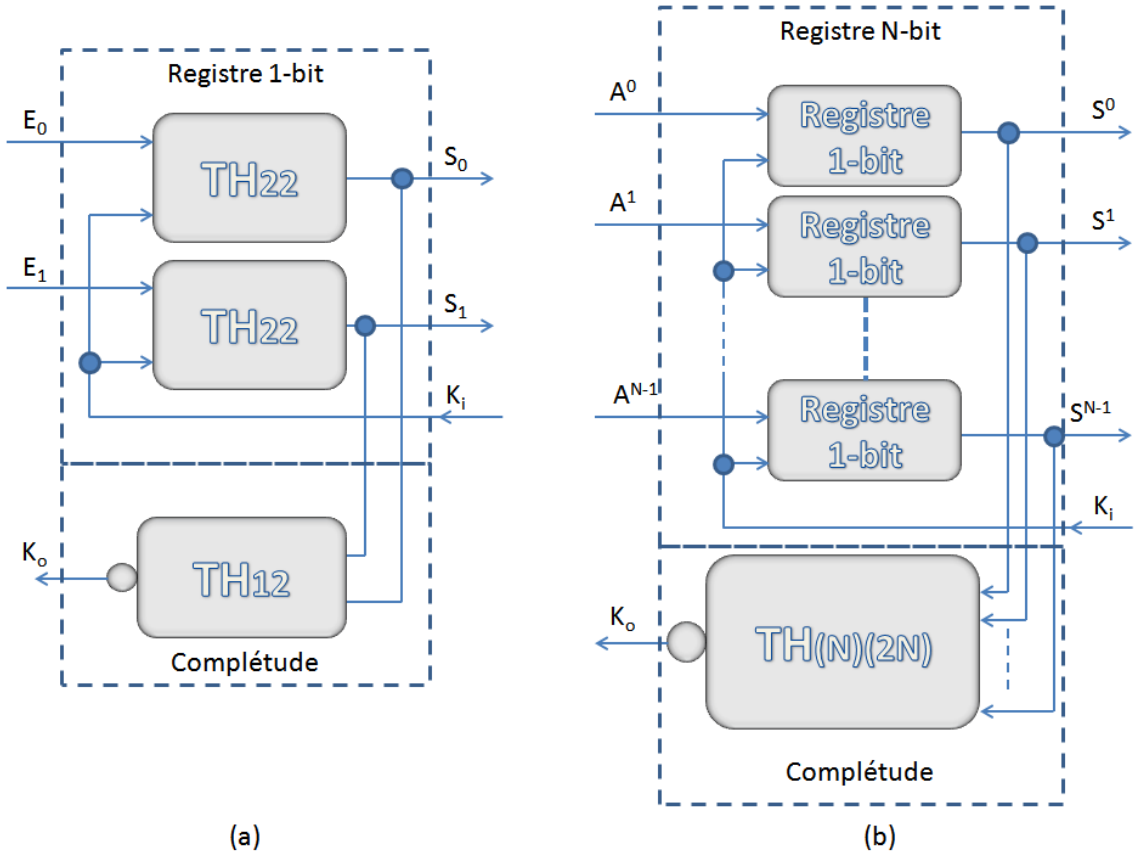


Figure 3-12. Registres NCL: (a) Registre 1-bit, (b) Registre N-bits

Comme représenté sur la Figure 3-12, chaque registre reçoit un signal provenant de l'étage suivant noté K_i et génère un signal en direction de l'étage précédent noté K_o. Lorsque les entrées d'un registre sont valides et que l'étage suivant est prêt à accepter de nouvelles données (K_i = 1) alors, les données sont propagées vers la sortie et le signal K_o passe à 0 signalant à l'étage précédent que les données ont bien étaient prises en compte et qu'il peut envoyer des données de type NULL. Si l'étage précédent n'est pas prêt à accepter des données de type DATA (K_i = 0) ou bien que la partie combinatoire ne fournit pas un ensemble de données de type DATA, le signal K_o reste à 1 signalant à l'étage précédent soit qu'il doit continuer à fournir des données de type DATA soit qu'il doit en envoyer. En connectant plusieurs registres fonctionnant de la sorte et en respectant les règles de conception précédemment énoncées, le concepteur s'assure de la validité des données prises en compte. La Figure 3-13 représente un ensemble de situations pouvant survenir et comment le protocole répond à ces dernières. Celle-ci n'est pas exhaustive mais permet de comprendre l'essentiel du protocole. On remarquera que ce protocole est bloquant, c'est-à-dire que lorsqu'une condition n'est pas satisfaite sur les signaux de synchronisation, le

pipeline va être bloqué jusqu'à ce que l'étage précédent envoie le type de donnée requis. Pour la Figure 3-13, nous employons la convention selon laquelle la couleur bleue représente soit une donnée de type NULL soit un booléen nul dépendamment si une donnée est considérée ou un signal de synchronisation, et la couleur rouge pour le type DATA ou un booléen égal à 1.

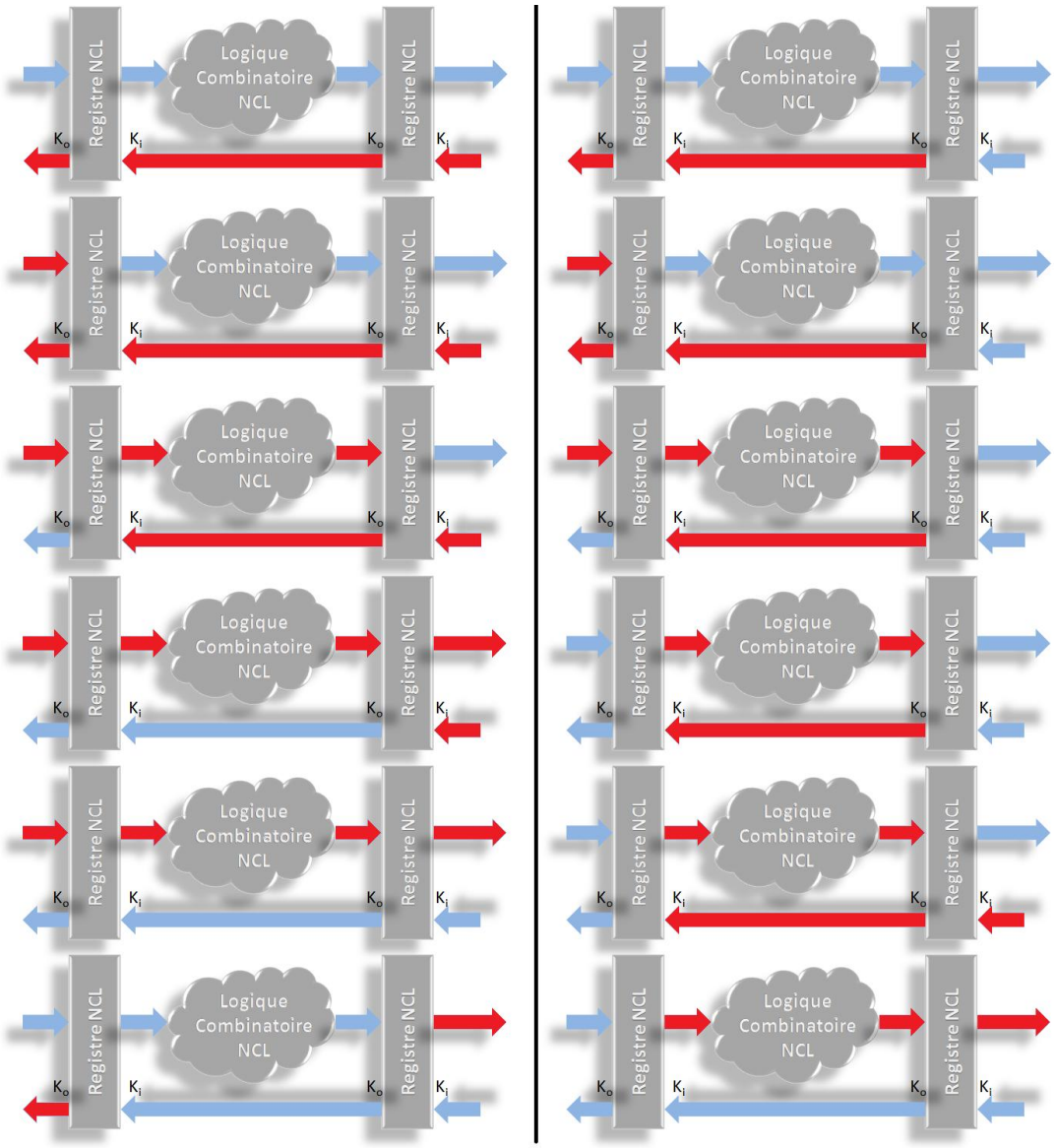


Figure 3-13. Exemple de réponse du protocole NCL

Il est alors aussi possible de concevoir des circuits séquentiels en réinjectant la sortie sur l'entrée. Malheureusement ceci ne peut pas être réalisé directement car il est nécessaire d'assurer l'alternance des fronts DATA et NULL. Une solution est alors de cascader en série au moins trois

registres asynchrones comme représenté sur la Figure 3-14. Une analyse des différents cas susceptibles de survenir similaire à celle de la Figure 3-13 montrerait le bon fonctionnement de la solution proposée. À des fins de vérification, nous avons implémenté sur FPGA un compteur à incrément de un et avons vérifié son bon fonctionnement en respectant ce protocole. Cet exemple sera détaillé dans le Chapitre 5 lors de l'étude de la faisabilité sur FPGA.

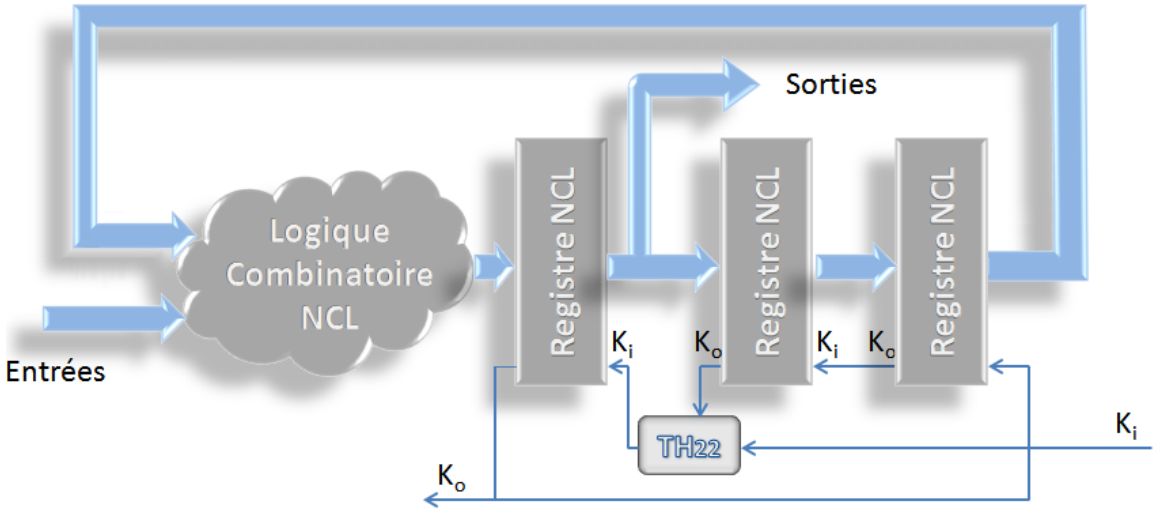


Figure 3-14. Schéma général d'un circuit séquentiel en logique NCL

3.4.4 Exemples de circuits

Afin de pouvoir appréhender la technique de conception proposée par le NCL, un ensemble d'exemples de circuits de base sont présentés dans l'ANNEXE 4. Notamment une unité arithmétique et logique de même qu'un circuit d'incrémentation y sont détaillés. Ces deux exemples seront par la suite réutilisés pour mesurer les performances obtenus grâce à la méthode proposée dans le Chapitre 4.

CHAPITRE 4 NULL CONVENTION LOGIC SIMPLIFIÉE : SHF-NCL

4.1 Motivations

4.1.1 Observations

En appliquant les règles de conception proposées par le NCL, nous nous sommes tout d'abord aperçus qu'il était possible d'obtenir des circuits asynchrones robustes. Dans le cadre de ce projet, nous avons opté pour des implémentations sur FPGA comme nous le justifierons dans la Chapitre 5. Nous avons alors implémenté, sur une telle plateforme, divers circuits allant du plus simple, comme une porte OU-Exclusif, à des circuits plus complexes comme des Unités Arithmétiques et Logiques ou encore un module de réveil comme celui que nous présenterons aussi dans le Chapitre 5. Néanmoins, bien que le NCL fournisse un cadre complet dans lequel, en suivant les règles de conception et en fournissant un effort minime de vérification temporelle, il est possible de concevoir des circuits asynchrones, plusieurs contraintes surviennent. Dans une moindre mesure, bien qu'un projet d'automatisation soit en cours de développement [40], sans de tels outils la conception et la vérification deviennent des tâches relativement contraignantes et parfois peu intuitives. En effet, le concepteur doit tout d'abord changer de façon de concevoir en adoptant la logique sur deux ou plusieurs fils. D'autre part, en ce qui concerne la vérification, dès qu'une erreur est trouvée, changer la logique devient très ardu notamment lorsqu'il s'agît de modifier une machine à états. Dans ce cas, toutes les équations doivent être recalculées et toute la correspondance avec les portes NCL doit être ré-établie.

D'autre part, nous nous sommes aperçus que l'utilisation de ressources pour des circuits très simples devenait rapidement importante. Par exemple, une simple porte ET en logique NCL peut nécessiter jusqu'à 9 éléments de base sur un FPGA comme ceux de la famille Igloo d'Actel. Ceci est principalement dû au fait que l'on utilise une représentation différente de la représentation booléenne qui, à première vue, aurait tendance à doubler la complexité, mais aussi car des portes à mémoire sont requises. Le changement de représentation, de booléen standard à une représentation complète sur plusieurs fils, ne peut pas être supprimé lorsque l'on utilise de la logique quasi-insensible aux délais (*QDI)*. En effet, l'information de validité doit être transportée par les données elles-mêmes. Le seul choix laissé au concepteur est celui d'une représentation efficace : c'est-à-dire que le ratio entre le nombre de fils utilisés par rapport au nombre de

symboles représentés doit être le plus petit possible. Par exemple, les représentations *quad rail* et *dual rail* sont optimales car ce ratio est de 2. D'autre part, tel que mentionné dans le Chapitre 3, les 27 portes NCL, à l'exception de celles de type TH1N, mémorisent l'état de la sortie jusqu'à ce que l'ensemble des entrées soit nul. Et comme nous l'avons vu, pour réaliser cette caractéristique, il est possible soit d'utiliser des chemins de rétroaction, soit d'utiliser des éléments de mémoire comme des loquets. Dans les deux cas, un surplus de ressources est nécessaire.

Ces deux constats nous ont amenés à nous questionner sur la nécessité d'utiliser les 27 portes NCL au lieu de portes booléennes standard et plus particulièrement leur caractéristique de mémorisation. Si tel n'était pas le cas, il serait possible de profiter d'un gain aussi bien en termes d'utilisation de ressources que de facilité de conception et de modification.

4.1.2 Objectifs

L'objectif principal de cette amélioration est donc de diminuer le nombre d'éléments logiques utilisés pour réaliser un design en logique asynchrone. De manière générale, cette simplification peut conduire à quatre principaux avantages détaillés ci-dessous.

Le premier avantage est dû à l'observation suivante : plus on utilise de ressources, plus la consommation de notre design est susceptible d'augmenter. En effet, pour des réalisations de type ASIC, un nombre élevé de ressources augmente à la fois la consommation statique (plus de fuites car plus de transistors) et dynamiques (plus d'éléments qui commutent). Sur un FPGA, si on réduit le nombre d'éléments, la consommation dynamique devrait aussi diminuer. En revanche, le nombre total d'éléments alimentés, qu'ils soient utilisés ou non, est fixe donc la consommation statique ne devrait pas être affectée par cette amélioration. Toutefois, si le gain en nombre d'éléments logiques libérés par l'amélioration du NCL est suffisant, on pourrait alors se trouver dans le cas où le choix d'un FPGA plus petit en termes d'éléments logiques puisse être envisagé. Par conséquent, la consommation statique pourrait aussi être diminuée. En conclusion, cette amélioration devrait pouvoir permettre une réduction de la consommation.

D'autre part, pour l'une ou l'autre des solutions (ASIC ou FPGA), réduire le nombre de ressources utilisées peut mener à la réduction de la surface et par conséquent à la réduction du coût de la solution. En effet, pour les ASIC, limiter le nombre d'éléments logiques réduit la surface de silicium utilisée et par la même le coût. Dans un environnement FPGA, comme

précédemment, réduire le nombre d'éléments logiques utilisés peut mener à choisir un FPGA plus petit en termes de portes logiques ce qui réduit généralement le coût de la solution.

Par ailleurs, l'utilisation de portes NCL nécessite soit un synthétiseur capable de faire correspondre le code avec des portes de la bibliothèque NCL, soit un prétraitement du code comme dans le projet UNCLE [40] ou bien de devoir instancier à la main chacune des portes NCL. Actuellement, le synthétiseur discuté n'est pas disponible notamment car l'industrie de la microélectronique semble timide quant à l'adoption de la logique purement asynchrone. Le projet UNCLE semble quant à lui très prometteur et a certainement été conçu pour faire face à la difficulté de conception de tels circuits. Néanmoins, il faut encore attendre que l'outil soit développé pour plusieurs plateformes et plusieurs langages. Actuellement seulement Linux i686 et Verilog sont supportés. Enfin, lorsque la tâche est laissée au concepteur, celle-ci s'avère particulièrement contraignante et demande un travail fastidieux. Une des grandes faiblesses demeure l'impossibilité de modification du code en changeant une équation. Généralement, c'est l'instanciation de plusieurs portes qui doit être modifiée. L'utilisation de logique combinatoire standard aurait alors cet avantage de faciliter à la fois la conception et la modification du code.

Enfin, une réduction de la complexité implique une diminution du nombre d'éléments logiques entre les entrées et les sorties du système. Par conséquent, il est possible d'envisager une amélioration des performances en termes de latence et de débit. Ce qui est un autre avantage par rapport au système synchrone pour lesquels l'implication n'aurait pas été aussi évidente. Ici, le temps de passage d'un étage à un autre est directement dépendant des délais à l'intérieur de l'étage. Dans le cas d'un circuit synchrone, celui-ci aurait été dépendant du délai dans le chemin critique. Il est donc envisageable de penser que cette diminution de la complexité puisse s'accompagner d'une amélioration des performances plus prononcée.

4.1.3 Extrapolation ASIC – FPGA

Comme nous l'avons mentionné précédemment, l'impact qu'aura cette simplification dépend de l'architecture visée. Notamment, en termes de consommation statique, de réduction de la surface utilisée ou encore du prix de la solution finale. Une manière simplifiée d'appréhender le problème est de considérer que l'environnement ASIC est continu et celui des FPGA est discret. En effet, pour une conception ASIC, le prix, la consommation statique et la surface utilisée dépendent directement du nombre d'éléments logiques utilisés. À l'inverse, pour les FPGA, un

ensemble de portes est acheté qu'elles soient utilisées ou non. Par conséquent, changer pour un FPGA plus ou moins complexe s'accompagne généralement par un écart substantiel aussi bien au niveau du prix, de la consommation dynamique et de la surface utilisée. À titre d'exemple, le Tableau 4.1 compare trois exemples de FPGA de la famille Igloo d'Actel. Par conséquent, l'impact global de la simplification dépendra du support physique d'implémentation.

Tableau 4.1. Exemple d'évolution du prix et de la consommation statique en fonction du nombre de portes pour les FPGA

FPGA	Nombre de portes	Consommation statique (µw)	Prix moyen ($)
AGLN010	10000	2	7
AGLN060	60000	10	12
AGLN125	125000	16	18
AGLN250	250000	24	20

À un niveau d'abstraction moins élevé, le retrait des éléments de mémoire des portes NCL se quantifie différemment sur l'une ou l'autre des plateformes. Pour les FPGA, il est généralement préférable d'utiliser des éléments de mémoire comme des loquets pour les portes NCL. Malheureusement ceux-ci ne sont pas nécessairement disponibles. Dans ce cas, il faudra se tourner vers une solution avec rétroaction. On peut considérer que les deux solutions sont quasiment équivalentes en termes de complexité. Quantitativement cela représente l'utilisation d'environ un à deux éléments de base dépendamment des architectures, ce qui en termes de transistors est relativement conséquent. Pour les ASIC, la modification survient au niveau des transistors mêmes. Il est alors nécessaire de différencier deux cas de réalisation de l'élément de mémoire : le cas statique et celui semi-statique présentés dans [37].

En analysant, l'impact qu'aurait la simplification sur chacune des portes NCL, on obtient qu'en moyenne, pour une réalisation statique le gain par porte est d'environ 9,2 transistors soit environ 52% de la taille moyenne d'une porte et dans le cas d'une réalisation semi-statique, le gain est de l'ordre de 3,6 transistors soit environ 29,4% de la taille moyenne des portes.

Par conséquent, dépendamment du support utilisé pour réaliser l'implémentation visée, on peut constater certaines fluctuations qui, au demeurant, n'altèrent pas l'objectif qualitatif attendu.

4.2 Idée Générale

4.2.1 Principe de la simplification

L'idée principale de cette simplification réside dans la suppression des éléments de mémoire qui sont instanciés à l'intérieur même des 27 portes NCL. Néanmoins, cette propriété était utilisée par le NCL pour assurer au niveau local la complétude par rapport aux entrées de type NULL. D'autre part, elle assure une sécurité redondante pour prévenir des aléas. Par conséquent, s'il est possible de retirer ces éléments de mémoire sans compromettre la règle de complétude et sans rajouter une quantité importante de portes pour que celle-ci soit respectée, alors un gain substantiel en termes de ressources peut être espéré.

4.2.2 Détails de la simplification

L'objectif est donc de trouver une façon peu coûteuse en termes de ressources qui nous permette de s'assurer que l'ensemble des sorties ne deviennne pas NULL tant que l'ensemble des entrées n'est pas NULL. Comme nous l'avons vu lors de la présentation du NCL, un circuit NCL se décompose sous la forme d'étages de logique encadrés par des registres asynchrones, l'ensemble étant très ressemblant à un pipeline classique. Pour des raisons de clarté et de manière à introduire brièvement la symbolique NCL, le schéma d'un registre NCL à un seul bit est représenté sur la Figure 4-1. Le nombre à l'intérieur des symboles représentant les portes correspond au seuil d'activation. Par exemple, lorsque celui-ci vaut deux, pour que la sortie de la porte vaille 1, il faut qu'au moins deux des entrées soient actives. Dans l'exemple de la Figure 4-1, les deux portes du haut ont deux entrées chacune et un seuil d'activation de deux, se sont donc des éléments C de Müller comme nous l'avions déjà mentionné dans le chapitre précédent. La porte du bas est une porte de complétion permettant de connaître l'état la sortie du registre : DATA ou NULL. Les éléments C de Müller fonctionnent de la manière suivante dans ce schéma : si le signal d'*acknowledge* provenant de l'étage suivant est actif et que l'une des entrées (E_0 ou E_1) est active, alors le signal correspondant (respectivement

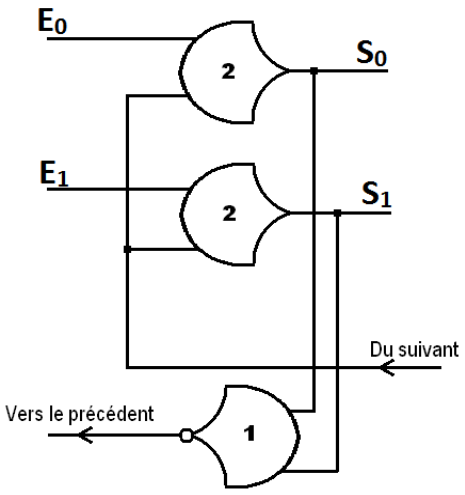


Figure 4-1. Registre NCL à 1-bit

S_0 ou S_1) sera activé. Si le signal d'*acknowledge* provenant de l'étage suivant est inactif et que l'une ou l'autre des entrées se désactive alors le signal de sortie correspondant sera désactivé. Dans tous les autres cas, les signaux de sorties restent inchangés. Le signal d'*acknowledge* n'est autre que le signal de complétion du registre de l'étage suivant. En d'autres termes, ce registre agît comme un tampon qui permet d'attendre que l'étage suivant soit prêt à recevoir des données ou un front NULL. Ceci implique notamment que le flot soit tel qu'il y ait une alternance de fronts de type DATA et de fronts de type NULL. D'autre part, pour des registres de plusieurs bits, il suffit de mettre en parallèle plusieurs registres 1 bit et de fusionner les signaux de complétion de telle sorte que le signal résultant soit actif lorsque tous les symboles sont de type NULL et soit inactif lorsque tous les symboles sont de type DATA.

Comme mentionné précédemment, ce qui permet au NCL de s'assurer de la propriété de complétude par rapport aux entrées sur front NULL, réside dans le fait que pour que les portes de la logique fournissent une sortie qui repasse à NULL, il faut que l'ensemble de leurs entrées soit NULL. Autrement dit, on vérifie au niveau local cette propriété de complétude. Ce qu'on propose ici, c'est d'effectuer cette vérification non plus au niveau local mais de manière globale à l'échelle d'un étage du pipeline. Ceci devant être fait sans surplus excessif de ressources dans le but d'assurer une diminution de la complexité.

4.2.3 Modification du pipeline

La première étape de la simplification consiste à ne plus utiliser les 27 portes définies par le NCL. Les portes utilisées sont alors des portes combinatoires standard. Il est néanmoins toujours nécessaire d'assurer la complétude par rapport aux entrées de type DATA. En reprenant l'exemple de la porte ET discutée dans l'ANNEXE 4, on obtiendrait alors un nombre de portes égal à 3 sur une architecture telle que celle des Igloo de la compagnie Actel. Cette réduction étant principalement due au retrait des éléments de mémoire. Cette modification peut se traduire schématiquement comme sur la Figure 4-2.

Néanmoins, remplacer une porte TH22 par un ET logique change le comportement de l'ensemble : une porte TH22 dont la sortie est active garde son état lorsque ses entrées sont par exemple {0,1} alors que la sortie d'un ET logique passerait à 0.

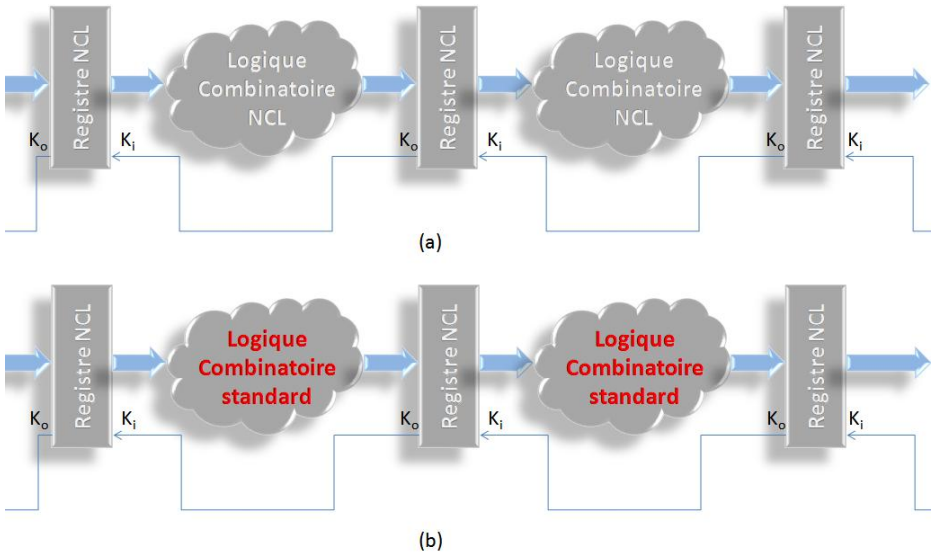


Figure 4-2. Utilisation de portes standard: (a) Avant la modification, (b) Après la modification

Par conséquent, pour un ensemble d'entrée prenant les valeurs {(NULL, NULL), (DATA1, DATA1), (NULL, DATA1)} les sorties de la porte ET en logique NCL prendrait les valeurs {NULL, DATA1, DATA1} alors que la porte ET en logique combinatoire standard prendrait les valeurs {NULL, DATA1, NULL}.Il est donc nécessaire de s'assurer que l'ensemble des sorties ne peut passer au type NULL que si l'ensemble des entrées est de type NULL. Ce que nous proposons ici est alors de ne plus vérifier cette propriété à la sortie de la logique mais à la sortie du registre de sortie. Pour cela, il nous faudrait un indicateur nous permettant de savoir que toutes les entrées de la partie combinatoire sont à NULL. Le principal intérêt de notre amélioration réside dans le fait que ce signal est déjà calculé : il s'agît de réutiliser le signal de complétion de l'étage précédent. En effet, lorsque celui-ci est actif cela signifie que l'ensemble des entrées de la partie combinatoire considérée est NULL. Par suite, la seule modification à apporter se situe au niveau du registre asynchrone détaillé plus haut. Tout d'abord, un nouveau signal de synchronisation est ajouté au schéma principal d'un circuit NCL comme représenté sur la Figure 4-3. On notera ce signal *e*, comme *enhanced*, et l'appellation de cette amélioration : SHF-NCL (*State-Holding Free NCL*).

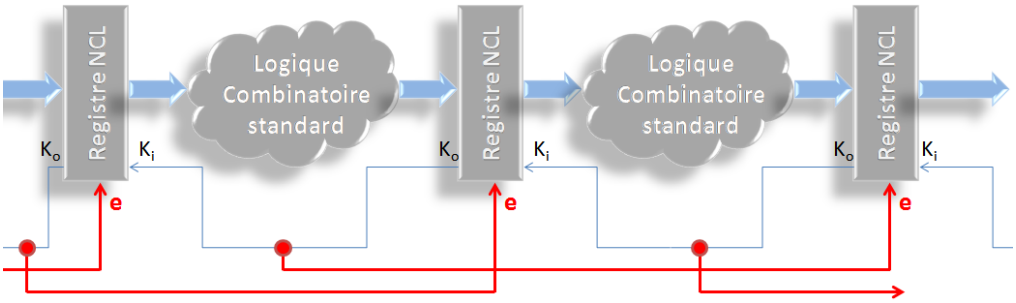


Figure 4-3. Schéma général du SHF-NCL

4.2.4 Registre SHF-NCL

Il est donc nécessaire de modifier les registres pour que ceux-ci prennent en compte le signal *e*. Pour les besoins de la cause, nous représentons sur la Figure 4-4 un registre 1-bit SHF-NCL et utilisons les notations employées sur cette dernière. Le registre SHF-NCL se comporte alors de la manière suivante :

Activation des sorties :

- Si $K_i = 1$ et $E_0 = 1$ et $e = 0$ alors S_0 est activée
- Si $K_i = 1$ et $E_1 = 1$ et $e = 0$ alors S_1 est activée

Désactivation des sorties :

- Si $K_i = 0$ et $E_0 = 0$ et $e = 1$ alors S_0 est désactivée
- Si $K_i = 0$ et $E_1 = 0$ et $e = 1$ alors S_1 est désactivée

La génération du signal K_o reste inchangée.

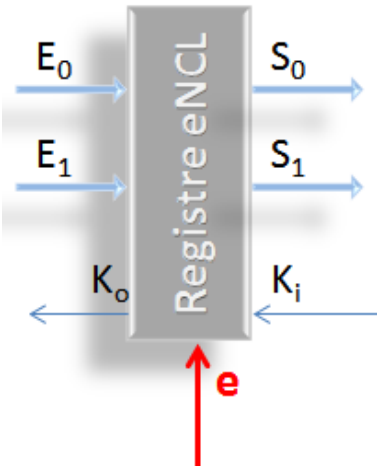


Figure 4-4. Registre SHF-NCL 1-bit

En d'autres termes, l'activation et la désactivation des sorties doivent respecter une nouvelle condition qui dépend du signal e, ce dernier indiquant que l'étage précédent envoie des données de type NULL lorsqu'il est actif ou DATA lorsqu'il est inactif. De plus, un des grands intérêts du signal *e* réside dans le fait qu'aucun matériel supplémentaire n'est nécessaire pour le générer. En revanche pour le prendre en compte, il suffit de le combiner avec le signal K_i ce qui requiert deux portes logiques ET. Pour un registre de plusieurs bits, on peut définir des signaux internes d'autorisation d'activation (A) et d'autorisation d'effacement (C) des sorties de la manière suivante :

$$\begin{cases} A = ack_{in}.\bar{e} \\ C = \overline{ack_{in}}.e \end{cases}$$

On a alors que chacun des registres d'un bit respecte les règles suivantes :

Activation des sorties :

- Si A = 1 et E_0 = 1 alors S_0 est activée
- Si A = 1 et E_1 = 1 alors S_1 est activée

Désactivation des sorties :

- Si C = 1 et E_0 = 0 alors S_0 est désactivée
- Si C = 1 et E_1 = 0 alors S_1 est désactivée

De cette manière, il est possible de modifier le comportement de la synchronisation locale en n'ajoutant qu'un nombre limité de portes.

4.3 Validité de l'approche

Avant de poursuivre, il est important de constater que la simplification proposée modifie grandement le protocole proposé par le NCL. Ce faisant, il est impératif de contrôler que l'intégrité des données n'a pas été altérée. Pour ce faire, nous proposons d'étudier un cas simple pour mettre en évidence le genre de problèmes soulevé avant de détailler la validité de l'approche notamment en termes d'aléas.

4.3.1 Exemple : Cas du XOR non-optimal

Le cas que nous allons utiliser ici est celui d'une porte OU-Exclusif non-optimisée. Par non-optimisée nous sous-entendons que la version utilisée n'est pas celle présentée dans l'ANNEXE

4 où le nombre de portes était réduit à son strict minimum. Dans ce cas-ci, nous réaliserons la porte en évaluant chacun des mintermes. La Figure 4-5 donne une représentation schématique de cette porte en logique NCL et en logique combinatoire standard.

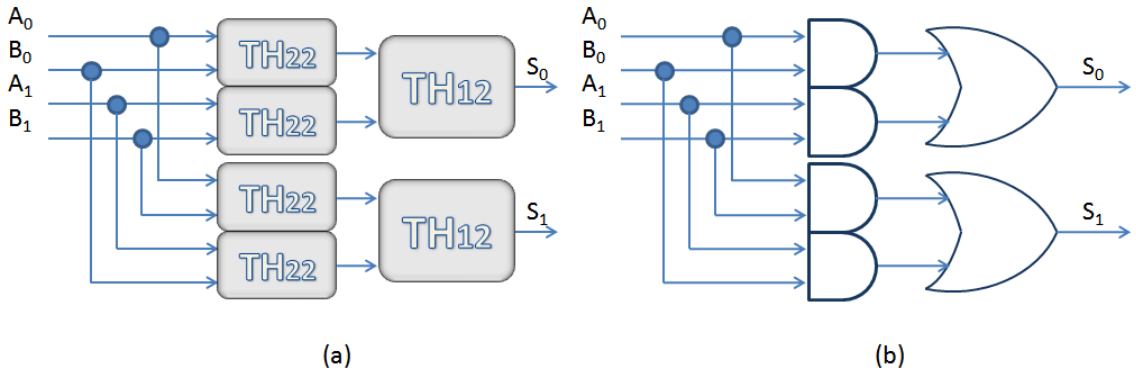


Figure 4-5. Schéma du OU-Exclusif non-optimal: (a) Logique NCL, (b) Logique combinatoire

Supposons alors qu'un délai significatif apparaisse sur l'une des branches. On considère les deux cas de la Figure 4-6 et de la Figure 4-7 pour lesquels les délais sont respectivement notés d_1 et d_2.

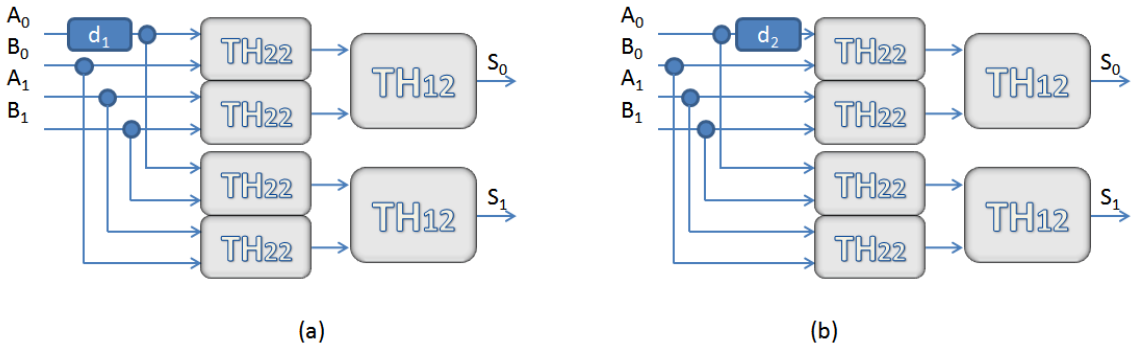


Figure 4-6. Schéma du OU-Exclusif NCL avec délai: (a) délai sur une fourche, (b) délai sur une branche

Supposons alors que ces circuits sont soumis aux deux successions d'entrées suivantes :

E1 : (A, B) = {(N, N), (DATA0, DATA1), (N, N), (DATA1, DATA0)}

E2 : (A, B) = {(N, N), (N, DATA1), (DATA0, DATA1), (DATA0, N)}

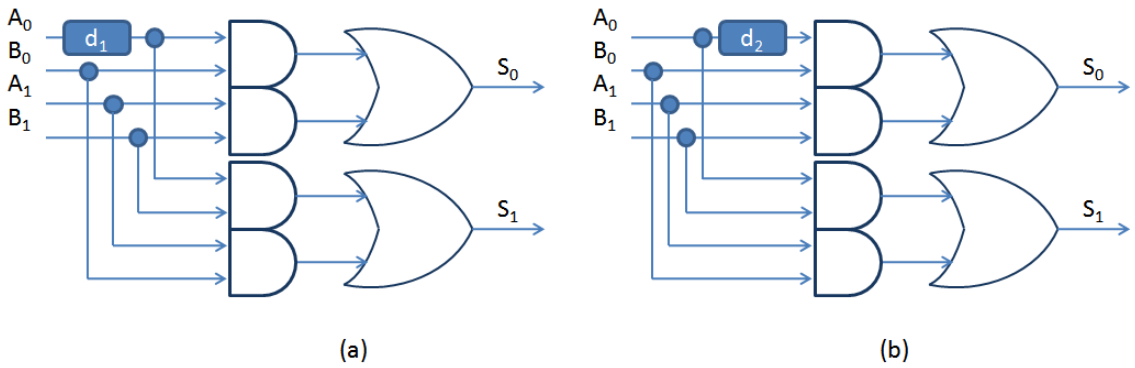


Figure 4-7. Schéma du OU-Exclusif SHF-NCL: (a) délai sur une fourche, (b) délai sur une branche

Et supposons de plus que les délais mis en jeu sont arbitrairement grands. Le circuit de la Figure 4-6a répond parfaitement aux stimuli d'entrées en ce sens que les sorties sont :

Q1 : (S) = {N, DATA1, N, DATA1}

Q2 : (S) = {N, N, DATA1, DATA1}

En revanche le circuit de la Figure 4-6b peut entrer dans un état invalide en ce sens que lorsque (A, B) = (DATA0, DATA1), le signal délayé étant orphelin, celui-ci peut être « coincé » de telle sorte qu'au front suivant de données, la porte TH_{22} la plus haute voit deux de ces entrées actives de même que la porte TH_{22} la plus basse. Il en résulte alors une sortie invalide :

Q1 : (S) = {N, DATA1, N, INVALIDE}

L'ensemble d'entrées E2, ne pose quant à lui aucun problème. En revanche, lorsque de la logique standard est utilisée il est possible de constater des problèmes pour les différents cas. Tout d'abord, considérons la situation de la Figure 4-7a pour les différents ensembles d'entrées. Celui-ci est moins robuste en ce sens que les portes ET ayant des entrées « invalidantes », le passage de l'entrée B à NULL va inhiber l'action de l'entrée A retardée conduisant ainsi à une sortie ne respectant pas la règle de complétude par rapport aux entrées de type NULL. On pourrait alors avoir les ensembles de sorties suivants :

Q1 : (S) = {N, DATA1, N, INVALIDE}

Q2 : (S) = {N, N, DATA1, N}

Dans les deux cas, le comportement n'est pas celui souhaité. Le circuit de la Figure 4-7b souffre des mêmes problèmes et génère les mêmes ensembles de sortie que dans le cas de la Figure 4-7a. Cet exemple introduit le type de problèmes pouvant survenir. Par conséquent, il est utile d'analyser quels sont les cas mettant *a priori* en défaut le bon fonctionnement du système et de considérer quels sont les parades proposées par le NCL. En effet, la simplification respectant l'ensemble des règles imposées par le NCL, il restera à savoir si celles-ci suffisent lorsque la logique utilisée n'est plus DI. Pour cela, nous verrons deux aspects dont il faut tenir compte : les aléas dus à des délais dans la logique et l'extrapolation du *fan-out* reconvergeant aux représentations complètes.

4.3.2 Analyse des aléas

Dans cette partie, nous allons considérer une certaine partie combinatoire en logique NCL et analyser les différents aléas qui pourraient survenir à ses sorties, celles-ci étant supposément reliées à un registre de sortie en accord avec le schéma général du pipeline NCL. Nous nous servirons alors des notations de la Figure 4-8. Le but étant de relever quelles sont les règles de conception ou autres hypothèses qui protège le NCL des aléas.

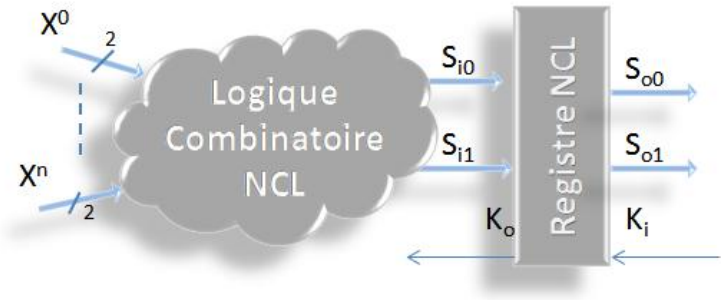
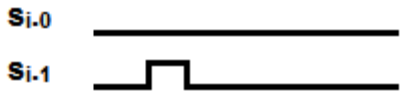
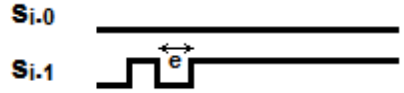
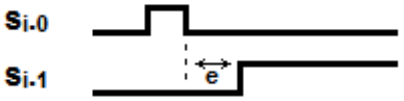
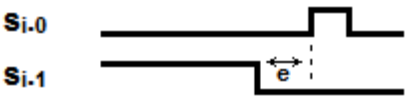
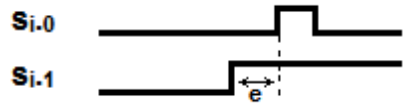
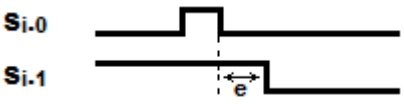
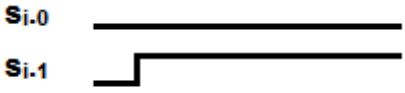
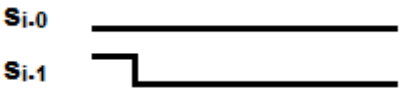


Figure 4-8. Cas utilisé pour l'étude des aléas

4.3.2.1 Différents types d'aléas susceptibles de survenir

On distingue alors plusieurs cas de transitions sur la sortie s_i que l'on résume dans le Tableau 4.2. On suppose que la sortie désirée est (S_{i0}, S_{i1}) = (0, 1) dans le cas d'un front de données et (0, 0) dans le cas d'un front NULL.

Tableau 4.2. Différents types d'aléas susceptibles de se produire

FRONT DATA	FRONT NULL
1. 	6.
2. 	7.
3. 	8. 
4. 	9. 
5. 	10. 

On remarque que les cas 5 et 10 sont les cas normaux attendus. On a aussi supposé que la combinaison des entrées devrait donner (S_{i0}, S_{i1}) = (0, 1) dans le cas d'un front de données et (0, 0) dans le cas d'un front NULL. Par conséquent les cas 1 et 6 sont invalides en ce sens qu'en régime permanent (lorsqu'on laisse suffisamment de temps aux sorties pour se stabiliser), celles-ci ne prennent pas la bonne valeur. Dans tous les cas, on supposera que le temps d'un état (haut ou bas) est suffisamment long pour être détecté et enregistré par le registre asynchrone de sortie. Si ce n'est pas le cas, cela revient à négliger les transitions concernées (aléas). On supposera aussi que pour le front DATA, le signal K_i est initialement à 1 et pour le front NULL celui-ci est à 0. Détaillons alors, les effets de ces variations sur les sorties :

- **Cas 1 :** L'état haut étant suffisamment long pour être enregistré par le registre de sortie on a (S_{o0}, S_{o1}) = (0, 1)
- **Cas 2** : Si *e* est plus grand que le temps dit « de retour » noté t_{ack}, tel que les changements sur S_o soient pris en compte par le module suivant et modifient le signal K_i, alors le passage à zéro sera pris en compte comme un passage à NULL et on aura alors

l'impression que deux fronts de données de valeurs (0, 1) et (0, 1) séparés par un front NULL auront eu lieu. Ce qui n'est évidemment pas le comportement souhaité. Si *e* est plus petit que $t_{ack'}$ alors cet aléa sera tout simplement ignoré.

- **Cas 3** : Si $e > t_{ack'}$, alors deux fronts de données de valeurs (S_{i0}, S_{i1}) = (1, 0) et (0, 1) séparés d'un front NULL peuvent être pris en compte. Sinon, la donnée invalide (S_{i0}, S_{i1}) = (1, 1) peut être enregistrée.
- **Cas 4** : Si $e < t_{ack'}$, la donnée invalide (S_{i0}, S_{i1}) = (1, 1) risque d'être enregistrée. Sinon, la valeur de S_{i0} sera ignorée car K_i sera passée à 0.
- **Cas 5** : C'est le cas normal de fonctionnement.
- **Cas 6 :** L'état bas étant suffisamment long pour être enregistré par le registre de sortie on a (S_{o0}, S_{o1}) = (0, 0). Néanmoins dès que celui-ci aura été pris en compte, une nouvelle valeur (S_{i0}, S_{i1}) = (0, 1) risque d'être prise en compte à tort.
- **Cas 7** : Si $e > t_{ack'}$ alors un front de données supplémentaires égal à (S_{i0}, S_{i1}) = (0, 1) va être pris en compte. Sinon celui-ci est ignoré.
- **Cas 8** : Si $e > t_{ack'}$ alors un front de données supplémentaires égal à (S_{i0}, S_{i1}) = (1, 0) sera pris en compte. Sinon celui-ci sera ignoré et le fonctionnement sera normal.
- **Cas 9** : Le signal K_i étant à zéro, indépendamment de la valeur de *e* le fonctionnement est normal.
- **Cas 10** : C'est le cas normal de fonctionnement.

On distingue alors trois types de fonctionnements :

- Les cas toujours valides (5, 9 et 10)
- Les cas valides ou invalides dépendamment de paramètres temporels (1, 2, 4, 7, 8)
- Les cas toujours invalides (3, 6)

Remarque : L'étude précédente prend en compte le cas où une seule sortie est générée par la logique NCL. Dans le cas de plusieurs sorties, tous les cas restent les mêmes pour chacune des sorties prise séparément, mais les conséquences sont diverses. Il serait alors possible d'étudier toutes les combinaisons de cas sur chacune des sorties mais les conclusions resteraient les mêmes. On peut néanmoins remarquer que dès lors qu'un cas « toujours invalide » survient sur l'une des sorties alors l'ensemble est invalide.

4.3.2.2 Parades du NCL

Nous cherchons ici à mettre en exergue les parades proposées par le NCL visant à se prémunir de ces aléas. Le but étant de savoir comment la simplification pourra ou devra adopter des parades si celles-ci ne sont pas conservées.

- **Cas 1 :** L'utilisation de la logique positive[4] et l'observabilité empêchent ce cas de se produire. En effet, raisonnons par l'absurde et supposons que ce cas puisse survenir. Pour cela il faudrait qu'une combinaison des signaux d'entrées puisse activer puis interdire indéfiniment un signal de sortie. La logique positive empêche que l'effet direct de l'activation d'un signal d'entrée soit la désactivation d'un autre. Autrement dit on ne peut pas avoir $x\uparrow \rightarrow y\downarrow$. L'observabilité assure que toute transition interne des entrées devrait êtres vue sur les sorties. Manifestement ici, après stabilisation des sorties, aucun des rails n'est actif. Ce qui est en contradiction avec cette règle d'observabilité.
- **Cas 2** : Ce cas respecte la propriété d'observabilité, car après stabilisation, la sortie s_{i-1} prend la valeur 1. Le fait que la sortie se désactive ne peut pas être du à une combinaison des entrées car on utilise de la logique positive comme mentionné plus haut. Néanmoins cet aléa peut-être dû à un délai comme montré dans l'exemple de la Figure 4-9.

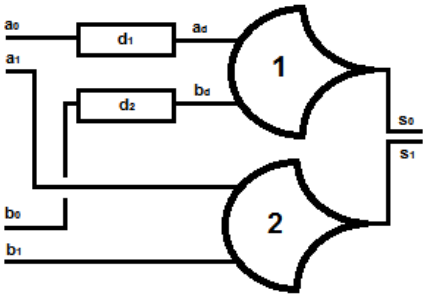


Figure 4-9. Exemple de circuit n°1

Dans cet exemple, on suppose que $d_1 >> d_2$ et que $d_1 > 2\ t_{ack'}$. Un chronogramme dans le cas considéré pourrait alors être celui de la Figure 4-10.

[4] On appellera logique positive toute combinaison de portes telle que pour chacune de ces portes l'activation (passage de 0 à 1) d'une entrée puisse avoir comme effet sur la sortie soit une activation soit aucun effet. Cette logique est utilisée lorsque l'on utilise une représentation complète. Réciproquement, la désactivation des entrées ne peut que désactiver les sorties ou les laisser inchangées.

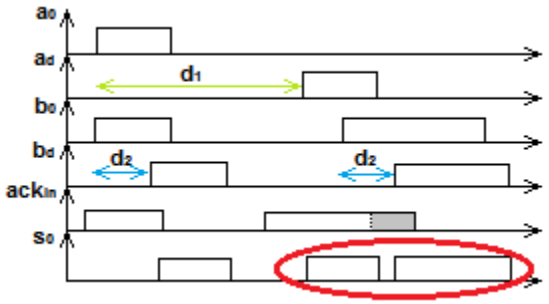


Figure 4-10. Chronogramme du circuit n°1

Néanmoins en écrivant les équations de s_0 et s_1 on se rend compte que l'ensemble n'est pas complet par rapport aux entrées de type DATA. En effet, on a :

$$s_0 = a_0 + b_0$$
$$s_1 = a_1.b_1$$

Une façon de le rendre complet par rapport aux entrées est, par définition, d'empêcher que la sortie soit valide si l'ensemble des entrées n'est pas valide. Or on constate que s_0 peut s'activer si l'une ou l'autre des entrées a_0 ou b_0 est active. On peut alors réécrire l'équation complète en rajoutant les termes suivants :

$$s_0 = a_0.(b_0 + b_1) + b_0.(a_0 + a_1)$$

$$s_0 = a_0.b_0 + a_0.b_1 + a_0.b_0 + a_1.b_0$$

$$s_0 = a_0.b_0 + a_1.b_0 + a_0b_1$$

Ici cela revient à énumérer tous les mintermes, ce qui n'est évidemment pas toujours le cas, notamment pour des modules à plusieurs sorties. Dans ce cas-ci, la sortie ne pourra pas prendre de valeur valide avant que l'ensemble des signaux traités ne soit valide. Par conséquent, le chronogramme devient celui de la Figure 4-11.

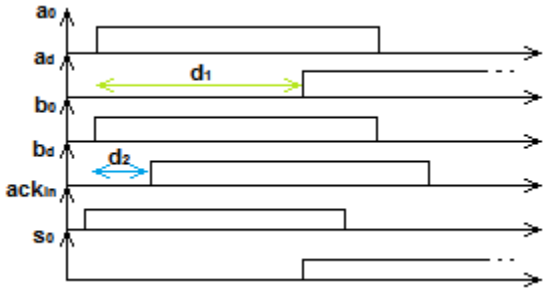


Figure 4-11. Chronogramme du circuit n°1 modifié

Le circuit attend donc que tous les signaux soient valides avant de fournir un ensemble, ici réduit à un élément, de sorties valide. On remarque que pour la suite du traitement, c'est la capacité de *state-holding* qui permet de s'assurer que le signal délayé a_d est bien repassé à 0 avant de redemander un nouveau front de données. D'où la remarque suivante.

Remarque importante : La complétude par rapport à NULL est induite par la mémorisation au niveau des portes. Dans la simplification proposée on cherchera à supprimer cette mémorisation ce qui rendra le système plus fragile notamment au genre d'aléas mentionné dans l'exemple précédent.

- **Cas 3** : En reprenant le cas de la figure 2 et en changeant les vecteurs d'entrées appliqués pour {(N, N), (0, 0), (N, N), (1, 1)}, on se retrouverait exactement dans le cas 3. Or la même remarque pourrait être faite en soulignant le fait que, lorsque l'on rend le circuit complet par rapport aux entrées alors ce genre de cas ne peut plus survenir toujours en faisant l'hypothèse de fourche isochrone.
- **Cas 4** : On se retrouve essentiellement dans la même situation que le cas précédent en modifiant la dépendance par rapport au délai d_1.
- **Cas 5** : C'est le cas normal de fonctionnement.
- **Cas 6 :** En utilisant ce qui a été souligné dans la remarque, on sait que pour que l'ensemble des sorties passe à NULL il faut que l'ensemble des entrées soit NULL. Par conséquent, pour que le signal S_{i1} repasse à une valeur active ceci ne peut pas être le résultat de la combinaison des entrées car celles-ci sont toutes nulles et que l'on utilise de la logique positive. Par conséquent, cela doit être induit par un retard sur une des branches. Une fois de plus, la complétude permet de se prémunir de ce genre d'aléas dans la majorité des cas. Néanmoins, comme nous l'avons vu en introduction de cette sous-partie avec le cas du circuit OU-Exclusif non-optimal, ce cas pourrait survenir s'il on ne faisait pas l'hypothèse de fourche isochrone.
- **Cas 7, 8 et 9** : Comme précédemment, la combinaison d'une logique incomplète par rapport aux entrées et des délais différents sur les différents chemins peut mener à ce

genre d'aléas. En revanche, en rendant la logique complète ceux-ci ne subsistent plus sauf dans des cas pour lesquels l'hypothèse de fourche isochrone n'est pas respectée.

- **Cas 10** : C'est le cas normal de fonctionnement.

Remarque importante 2 : Il est important de noter que la mémorisation réalisée au niveau local par chaque porte, celle-là même qui donne la caractéristiques de complétude par rapport aux entrées sur front NULL, prévient le système de tomber dans des états invalides. En effet, sans cela, on pourrait imaginer le cas où une entrée reste valide alors qu'une autre devient « invalidante », comme dans le cas de la porte ET standard, ce qui entraînerait un ensemble de sortie NULL alors qu'au moins une des entrées n'est pas de type NULL. C'est notamment cette remarque qui nous a conduit à ajouter le signal *e* pour maintenir les règles de conception qui se trouvent être directement dictées par des considérations utiles pour empêcher les aléas.

4.3.3 Fan-out Reconvergeant

Comme nous l'avons souligné précédemment, la complétude par rapport aux entrées ainsi que l'observabilité semblent, dans la majorité des cas, permettre de se prémunir des différents problèmes dus aux aléas. Néanmoins, l'exemple introductoire de la porte OU-Exclusif soulignait le fait qu'il est possible que certains de ces circuits puissent encore être invalides. L'argument alors utilisé par les concepteurs du NCL fut de supposer que les fourches étaient isochrones. Ici, on propose d'étudier le problème en faisant le parallèle avec les problèmes de *fan-out* reconvergeant des circuits combinatoires usuels.

On désigne par « *fan-out* reconvergeant » tout signal se divisant en deux ou plusieurs branches, nécessaires à l'évaluation de signaux intermédiaires, dont au moins deux de ces résultats intermédiaires se recombinent pour définir un autre signal. Lorsque les signaux intermédiaires réutilisés comprennent une copie du signal d'origine et son inverse, l'évaluation de la variable finale peut être sujette à des aléas. En effet, le fan-out reconvergeant est une condition nécessaire mais non-suffisante pour avoir des aléas dus à la logique.

Dans l'exemple de la Figure 4-12 ci-dessous tiré de [41], une copie du signal *x* est utilisée pour évaluer *s* et *t* qui eux-mêmes permettent de déterminer *h*.

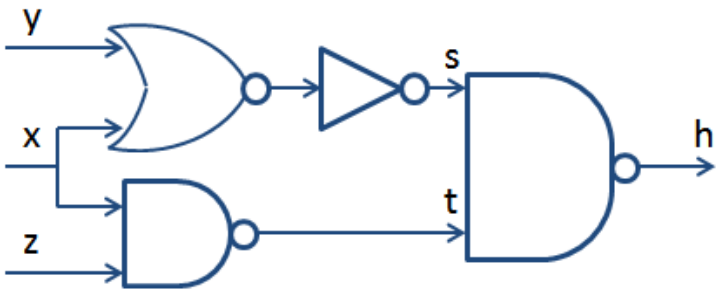


Figure 4-12. Exemple de *fan-out* reconvergeant

Dans cet exemple, la différence de charge sur chacune des branches implique que l'évaluation des signaux *s* et *t* se fait à des instants différents. Supposons alors que pour un changement d'une entrée (*x*, *y* ou *z*) le signal *s* se stabilise avec un retard *d* par rapport à *t*. Alors, pour un jeu d'entrées passant de (*x*, *y*, *z*) = (1,0,1) à (0,0,1) et en supposant que toutes les autres transitions sont instantanées on obtiendrait le chronogramme de la Figure 4-13.

Figure 4-13. Chronogramme mettant en évidence les problèmes de *fan-out* reconvergeant

Par conséquent, bien que $h = \bar{x}.\bar{y} + x.z$, et que $h(1,0,1) = h(0,0,1) = 1$, la convergence de deux copies complémentaires du signal *x* couplée à des délais de propagation nous donne un résultat qui passe par zéro. Dans les systèmes synchrones, ce genre d'aléas ne pose pas de problème dès lors qu'il est résolu à l'intérieur d'un cycle d'horloge. En revanche pour des circuits purement combinatoires, ce « *glitch* » peut avoir un effet indésirable comme par exemple pour des contrôleurs d'affichage.

Pour les circuits NCL, le fait d'utiliser de la logique positive, nous empêche de voir concrètement ce genre de problèmes. En revanche, on pourrait étendre la définition de chemin reconvergeant à la représentation sur deux fils de la manière suivante :

> *On appelle branche orpheline, toute branche d'un signal qui n'influe pas la détermination d'un signal de sortie. Alors pour tout circuit comportant une division d'un signal en au moins deux branches dont une est orpheline et que celle-ci sert à la détermination de l'un des deux signaux s_0 ou s_1 d'une sortie s et qu'une autre sert à la détermination de l'autre signal définissant cette sortie (s_1 ou s_0) alors ce circuit est susceptible de souffrir d'aléas.*

En d'autres termes, pour de la logique combinatoire traditionnelle, les aléas de la logique sont donnés par le fait qu'une entrée et son inverse sont recombinés pour le calcul d'une sortie et que celles-ci souffrent de délais de propagation différents. Pour de la logique positive en représentation sur deux fils, le *fan-out* reconvergeant est défini par l'utilisation d'un même signal pour définir à la fois une sortie et son complémentaire. Compte tenu de la propriété d'observabilité, ces branches ne peuvent se trouver qu'avant le premier niveau de portes logiques ou alors une de ces branches doit nécessairement être une sortie. D'autre part, compte tenu de la propriété de complétude la quasi-totalité des circuits NCL comporte ce type de branches orphelines. Ce qui « sauve » le NCL c'est l'utilisation de l'hypothèse de fourches isochrones. En effet, le cas mentionné plus haut ne génère des aléas que dépendamment des délais de propagation dans la branche orpheline (cf. Figure 4-6b). Cette dépendance par rapport aux délais de propagation se retrouve aussi dans le cas du *fan-out* reconvergeant standard. Ici, il faut s'assurer que le délai sur la branche orpheline est plus court que le temps entre deux fronts de types DATA.

L'hypothèse de fourche isochrone utilisée dans le cas de circuits NCL nous empêche de supposer qu'un délai sur une des branches d'une séparation d'un même fil soit suffisamment long pour causer des erreurs de validité. Néanmoins, il est intéressant de savoir s'il est possible, tout en respectant cette hypothèse, de construire un circuit pour lequel les délais sur des branches se divisant successivement s'accumulent. Pour ce faire, considérons deux exemples avant de tenter de généraliser. Les exemples utilisés seront ceux d'un additionneur N bits avec propagation de la retenue simple comme présenté dans le chapitre précédent ainsi qu'un comparateur d'égalité itératif.

Additionneur 1-bit :

Pour cet exemple, on reprend comme module de base noté FA le circuit de la Figure 3-10. On construit alors un circuit d'addition N-bits en cascadant plusieurs de ces modules de base comme représenté sur la Figure 4-14: c'est le *Ripple Carry Adder*.

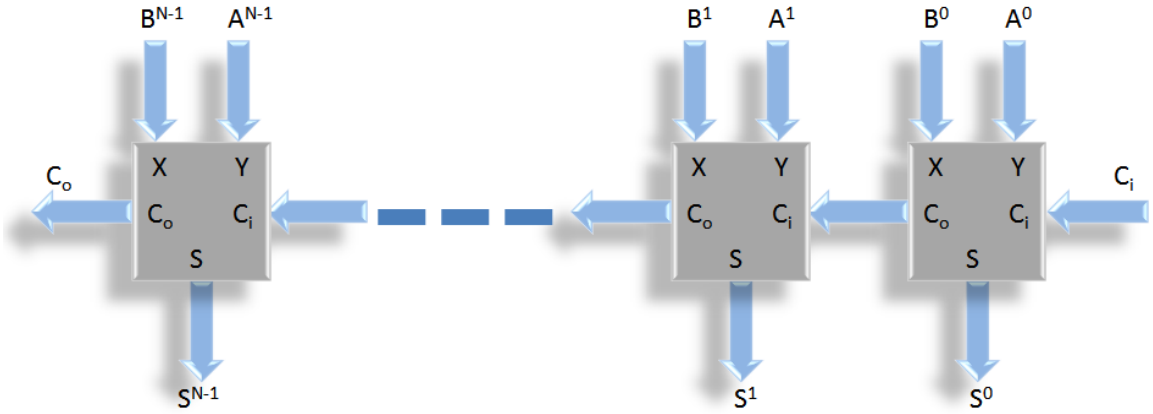


Figure 4-14. *Ripple Carry Adder*

En analysant la Figure 3-10 ainsi que le montage de la Figure 4-14, on constate qu'il n'existe pas de signal se divisant pour affecter à la fois un fil d'une sortie et de son complémentaire et que cette sortie se propage d'une porte à l'autre. En effet, seul le signal de retenue se propage et pour celui-ci les signaux utilisés pour l'évaluation des deux bits sont dissociés. Par conséquent, tentons de concevoir un exemple dans lequel on puisse rencontrer ces contraintes.

Comparateur d'égalité :

On considère dans cet exemple un module dont le but est de déterminer l'égalité entre deux nombres. Comme précédemment, supposons une réalisation itérative d'un tel module dont l'élément de base serait tel que celui de la Figure 4-15.

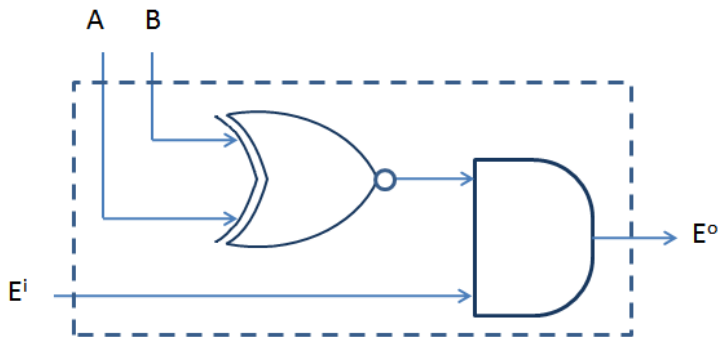


Figure 4-15. Module de base pour la détermination de l'égalité

Une traduction directe de ce schéma en logique en représentation sur deux bits, donnerait les équations suivantes pour la sortie E^o :

$$E_1^o = E_1^i.(A_0.B_0 + A_1.B_1)$$

$$E_0^o = E_0^i + A_1.B_0 + A_0.B_1$$

Manifestement, celles-ci ne sont pas complètes par rapport aux entrées sur front DATA. Par exemple, E^o peut prendre la valeur DATA0 si (A, B, E^i) = (N, N, DATA0). Il serait alors envisageable de rajouter des termes dans la seconde équation pour arriver au bon résultat. Pour palier cela, on préfère ici rajouter une sortie intermédiaire notée S qui en plus de résoudre partiellement le problème de complétude, nous permettra de connaître aussi le résultat bit à bit de la fonction NXOR appliquée aux deux vecteurs d'entrées. De cette manière les équations deviennent :

$$E_1^o = E_1^i.(A_0.B_0 + A_1.B_1)$$

$$E_0^o = E_0^i + A_1.B_0 + A_0.B_1$$

$$S_1 = A_0.B_0 + A_1.B_1$$

$$S_0 = A_1.B_0 + A_0.B_1$$

L'ensemble des sorties n'est toujours pas complet par rapport aux entrées sur front DATA. Néanmoins il suffit de modifier légèrement la seconde équation pour parvenir à un système complet :

$$\begin{cases} E_1^o = E_1^i.(A_0.B_0 + A_1.B_1) \\ E_0^o = E_0^i + (A_1.B_0 + A_0.B_1).E_1^i \\ S_1 = A_0.B_0 + A_1.B_1 \\ S_0 = A_1.B_0 + A_0.B_1 \end{cases}$$

Finalement, l'ensemble des sorties (E^o, S) ne peut pas passer du type NULL à DATA si l'ensemble des entrées (E_i, A, B) n'est pas de type DATA. Une représentation de ce circuit est alors donnée sur la Figure 4-16.

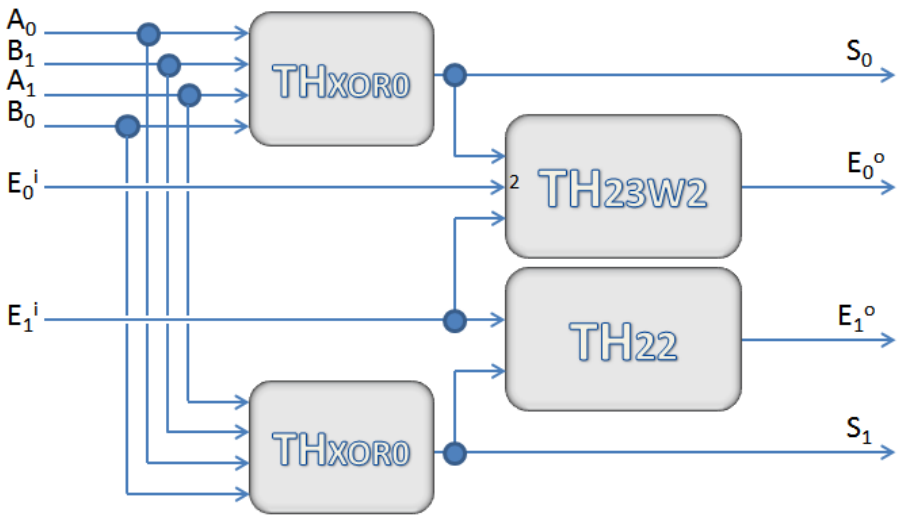


Figure 4-16. Schéma NCL du module d'égalité

En couplant K modules de base à l'image de ce qui était proposé sur la Figure 4-14 où E joue le rôle de C, on obtient un comparateur de nombre itératif. Le circuit ainsi formé est observable et complet par rapport aux entrées. En revanche celui-ci comporte un signal E_1^i servant à la détermination de E_o^0 et de E_o^1. Cela correspond donc bien au cas de *fan-out* reconvergeant décrit pour les circuits en représentation sur deux fils. Supposons maintenant qu'il existe un délai *d* sur la branche $E_1^i \rightarrow TH_{22}$ et supposons que les nombres A et B sont tels que :

$$\begin{cases} A_i = B_i, & \forall i \in [1; K-2] \\ A_{K-1} = \overline{B_{K-1}} \end{cases}$$

Le signal E^i se propage donc au travers de *K-1* délais *d*. On a donc que $E_1^{i,K-1}$ passera à 1 après un temps (*K-1*)**d* après que $E_1^{i,1}$ eut été activé. Et d'après le format de A et B, on a que $E_1^{i,K-1}$ est un signal ayant une branche orpheline. Par conséquent, il semblerait qu'on soit en mesure de sommer des délais. Cela peut-il avoir un impact sur le fonctionnement du circuit?

Dans le cas de la logique NCL, cela n'a pas d'impact sur la validité des résultats : certes le débit va être réduit car il faut attendre que le signal se propage avec un délai au moins égal à (*K-1*)**d* mais pour la suite, il faudra attendre que tous les signaux E_i intermédiaires qui ont été mis à 1 repassent à 0 pour que l'ensemble des sorties soit NULL.

En revanche, quand est-il du circuit simplifié? Pour que l'ensemble des sorties soit NULL, il suffit que A et B soient NULL. Supposons que cela prenne un délai d_N très petit devant (*K-1*)**d*.

Ceci est possible car K peut être choisi arbitrairement grand pour que $(K\text{-}1)*d >> d_N$. On pourrait alors penser que la désactivation de $E^{o,\,K-1}$ prend un temps $(K\text{-}1)*d$. Mais ce n'est pas le cas car la porte TH22 est elle aussi transformée en une porte sans mémorisation de type porte ET. Par conséquent, la désactivation de $E^{0,\,K-1}$ et par la même de tous les $E^{i,\,j}$ se fait en un temps négligeable par rapport au temps entre deux fronts DATA.

Généralisation :

Au vu de ces deux exemples, peut-on conclure quant à l'impossibilité d'accumuler des délais? Notons tout d'abord ce qu'il manque pour qu'un exemple puisse mettre en défaut la solution et tentons de vérifier si cela est compatible avec les règles d'observabilité et de complétude par rapport aux entrées. Pour pouvoir correctement additionner les délais, il faudrait une ligne de propagation qui prenne le signal d'entrée déjà retardé et qui le propage en ajoutant un petit délai *d* indépendamment des autres signaux. En connectant *K* de ces circuits, on aurait un signal retardé de $K*d$. Pour que cela puisse avoir un effet néfaste, il faudrait que cette ligne à délais soit close par une porte qui dépendrait d'un autre signal et qui inhiberait ce signal propagé. Ce signal inhibiteur devrait avoir un délai de propagation, entre les entrées et la recombinaison avec le signal délayé d'intérêt, négligeable par rapport à $K*d$. Un tel signal entre en contradiction directe avec la propriété d'observabilité. Autrement dit la complétude par rapport aux entrées impose que différents signaux soient minimalement couplés et l'observabilité empêche qu'un signal orphelin ne se propage au-delà d'une porte.

Par conséquent, en supposant que le délai dans la branche la plus retardée d'une fourche est plus petit que le temps qu'il a fallu pour que l'ensemble des sorties soit valide et reconnu comme tel (hypothèse de fourche isochrone) et en appliquant les règles de complétude et d'observabilité, la validité des signaux est assurée pour les circuits NCL. La simplification quant à elle bénéficie des mêmes propriétés dès lors que le délai dans le chemin le plus long est inférieur à celui entre un front NULL et un front DATA. Dans le Chapitre 5, on discutera de cette hypothèse et de la faisabilité de tels circuits sur FPGA.

4.3.4 Validité de l'approche

L'ajout du signal *e* peut rendre complexe la vérification d'un tel circuit et on peut craindre que certains problèmes de course se manifestent. Ici on propose d'étudier les différents cas qui peuvent survenir pour s'assurer de la validité de notre approche. Pour cela, on représentera sous forme de schémas ces différents cas en utilisant la couleur bleu pour les signaux de type NULL ou 0 et la couleur rouge pour les signaux de type DATA ou 1. Les différents cas sont représentés sur les Figure 4-17 et Figure 4-18.

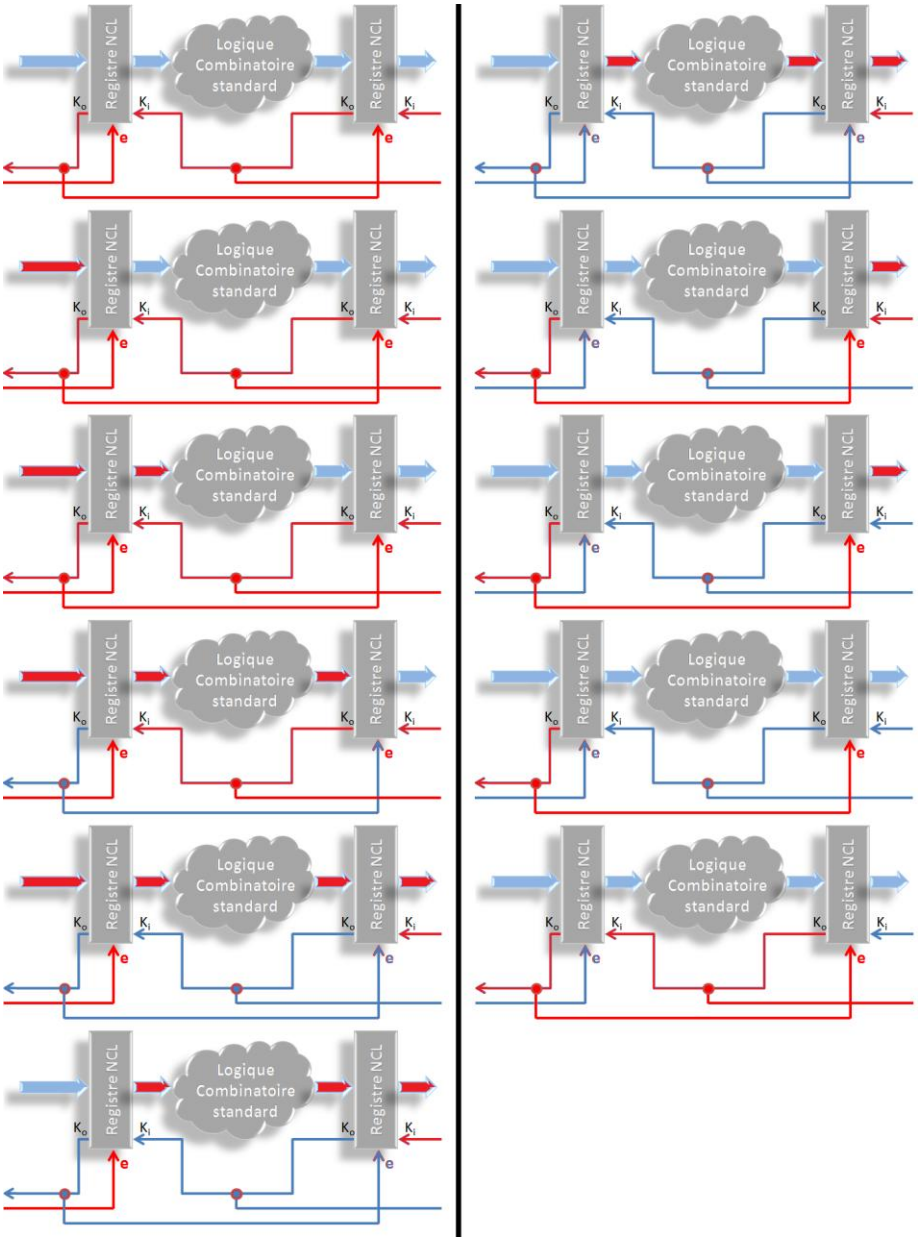


Figure 4-17. Exemple de protocole SHF-NCL n°1

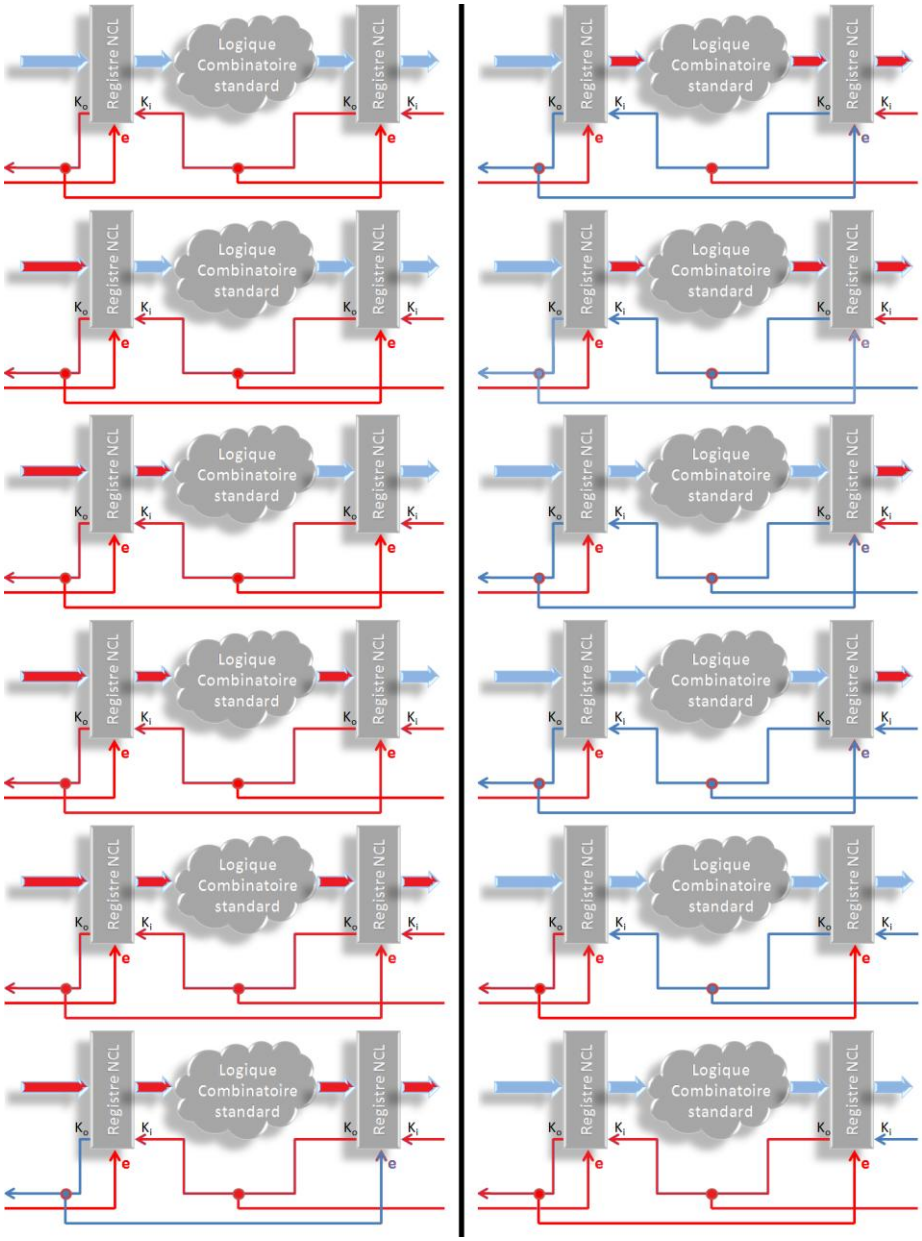


Figure 4-18. Exemple de protocole SHF-NCL cas n°2

De plus, il convient de traiter le cas d'un circuit séquentiel. Dans ce type de circuit, la partie combinatoire est suivie de trois ou quatre registres qui servent à la mémorisation de l'état précédent tout en respectant l'alternance des fronts DATA et NULL. L'utilisation de quatre registres plutôt que trois permet notamment d'augmenter le débit traité. Dans notre cas, ce qui importe est la communication entre ces registres par conséquent, on peut étudier indifféremment l'un ou l'autre des montages. Nous choisissons alors d'étudier le cas à trois registres comme

représenté sur la Figure 4-19. Le code de couleurs utilisé est le suivant : les données sont représentées en bleu, les signaux de type K_i/K_o en noir et les signaux de type e en rouge.

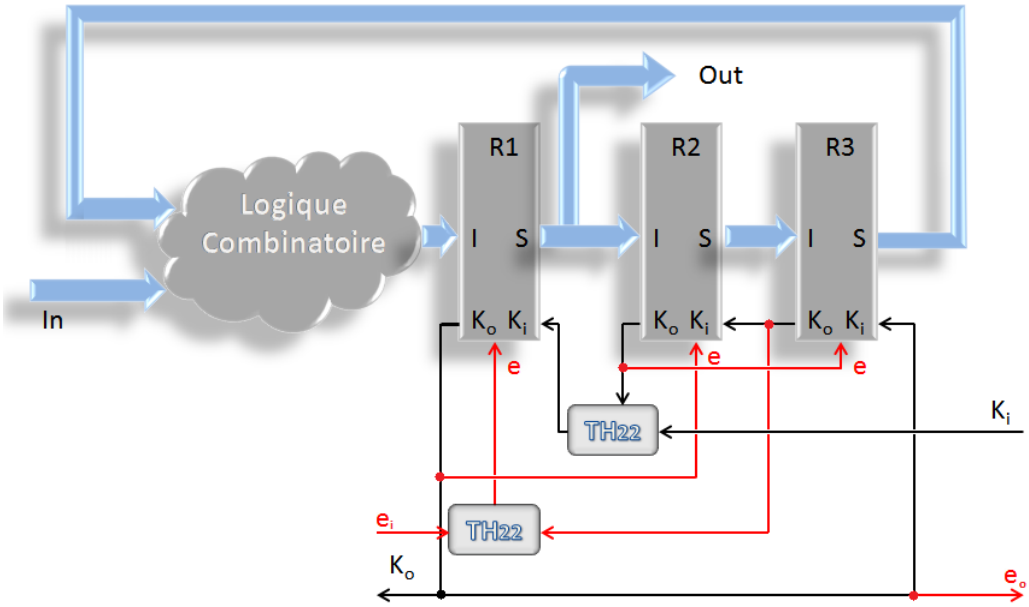


Figure 4-19. Schéma typique d'un circuit séquentiel SHF-NCL

Dans ce schéma, les données passent d'un registre à l'autre puis passent au travers de la logique. Pour un fonctionnement correct, il faut que le registre du centre soit initialisé à une donnée de type DATA et que les deux autres soient initialisés à NULL. On constate alors qu'une nouvelle donnée est calculée seulement si l'étage suivant (non représenté dans la Figure 4-19) requiert une nouvelle donnée. Pour faire l'analyse de ce circuit on utilisera le chronogramme de la Figure 4-20 sur lequel certains événements numérotés seront détaillés subséquemment. On adopte la convention selon laquelle tous les signaux propres aux registres seront indexés avec le numéro du registre correspondant et les signaux qui ne sont pas numérotés sont des signaux qui communiquent directement avec l'extérieur.

1 : À cet instant, le signal de *reset* est relâché. L'entrée *In* aurait pu passer à DATA. Ceci n'aurait rien changé car la logique étant complète par rapport aux entrées *In* et S_3, l'entrée I_1 serait restée NULL.

2 : À cet instant, le système et stable et attend que l'environnement extérieur fournisse une donnée *In* de type DATA.

3 : Le système est stable et attend qu'au travers du signal K_i, l'environnement extérieur signal que la donnée *Out* a bien été prise en compte.

4 : Quand K_i se désactive, le processus continue et bloque tant que les données en entrée, *In*, ne sont pas remises à NULL. On voit ici l'intérêt du signal *e*. En effet, la logique étant de la logique standard, on a supposé être dans un cas où la présence de données de type NULL à la sortie du registre R3 a fait en sorte que la sortie de la logique, *I1*, soit devenue NULL alors que l'entrée *In* est encore de type DATA. Mais, étant donné que l'étage précédent (non représenté) n'a pas encore validé le fait que *In* est de type NULL (au moyen du signal e_{in}), le registre R1 ne peut pas encore considérer son signal d'entrée comme étant NULL. Ceci découle directement des formules de *A* et *C* fournies plus haut dans ce chapitre.

5 : Lorsque le signal *In* passe finalement à NULL et que ceci est confirmé par l'activation du signal e_{in}, alors le registre R1 peut propager ce front NULL.

6 : Puis le système se stabilise jusqu'à ce que l'étage suivant (non représenté) signale la prise en compte de l'état NULL du signal *Out*. Si cela était arrivé entre temps, le système serait alors directement passé à l'étape 7 (moyennant les temps de propagation et le respect du protocole).

7 : Le processus se stabilise et attend un nouveau front DATA sur l'entrée *In*.

Dans cet exemple, on constate que le protocole est stable et que chaque front de données est attendu et son intégrité est préservée. On constate de plus que l'ordre d'arrivée ainsi que le temps d'arrivé des signaux de conditions (K_i et *e*) n'importe pas.

Enfin, pour parfaire l'étude de conservation de l'intégrité des données, il convient de revenir sur le cas de délais excessif entre le registre d'entrée et le premier rang de portes de la logique comme présenté sur la Figure 4-7a. Dans ce cas, nous avons vu que le bon fonctionnement du circuit pouvait être mis à mal. Mais un tel cas est-il envisageable physiquement? Dans la pratique, pour avoir un délai considérable sur un fil, il faudrait que le sous-ensemble du registre d'entrée correspondant au fil considéré soit situé suffisamment loin sur le FPGA par rapport au reste de la logique. Si tel était le cas, le signal *e* serait alors lui aussi retardé d'un délai comparable à celui noté d_1. Par la suite, les délais relatifs peuvent être négligés par rapport aux délais dans la logique interne et dans le chemin de rétroaction. En d'autres termes, mêmes si des délais importants apparaissent sur certains fils, les signaux *e* et *K* feront en sorte que les signaux retardataires soient toujours attendus. De plus, l'observabilité nous permet de nous prémunir des

éventuels effets des retards sur les signaux situés après le premier rang de logique. Dans le Chapitre 5, nous reviendrons plus en détails sur les délais dans les fils pour un environnement FPGA.

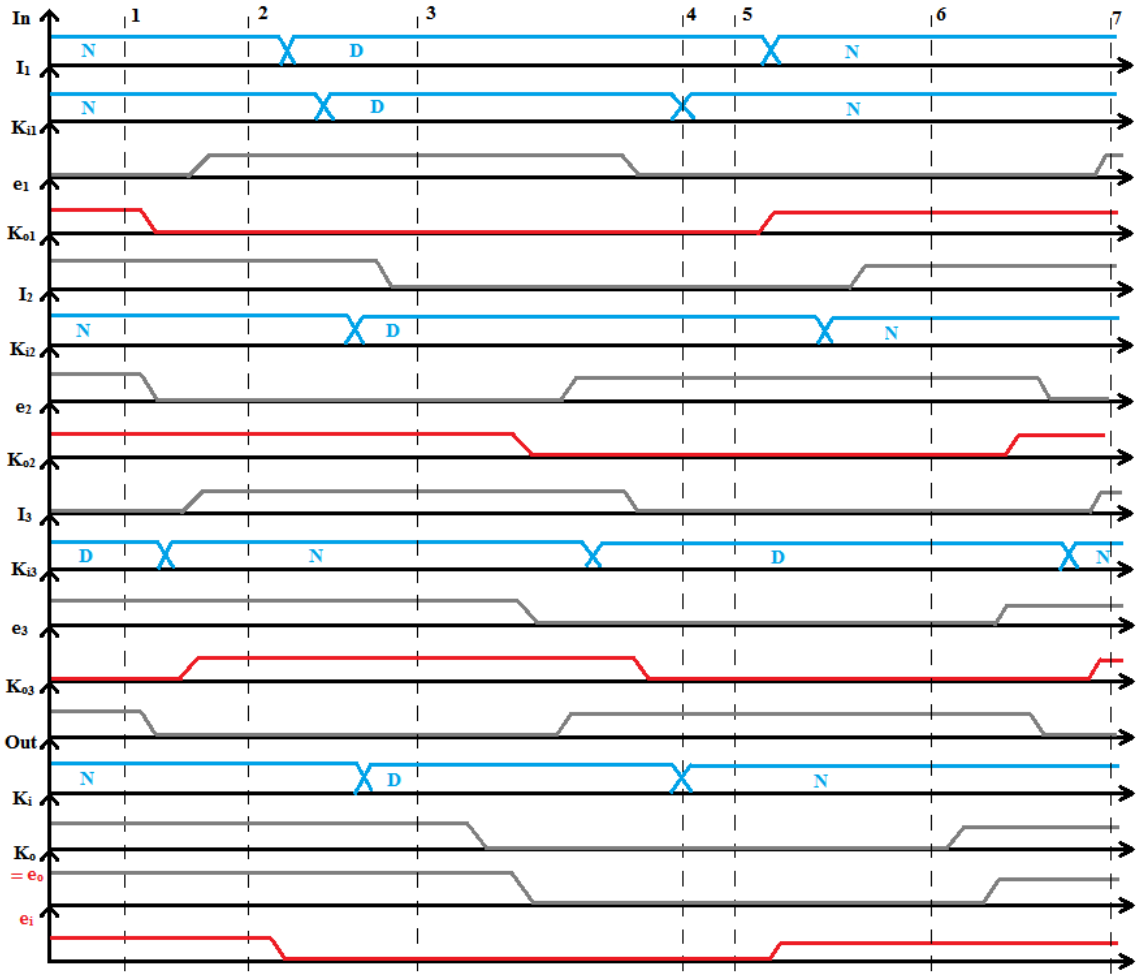


Figure 4-20. Chronogramme typique d'un circuit séquentiel SHF-NCL

4.4 Étude du Gain

Après avoir discuté de la faisabilité d'une telle simplification, nous allons mettre en avant quels sont les principaux gains attendus. Nous commencerons par présenter des relevés réalisés sur divers exemples puis nous conclurons par une analyse du gain maximal théorique.

4.4.1 Relevés sur différents exemples

Dans un premier temps, nous quantifions le surplus engendré par l'ajout du signal *e*. Ce surplus devrait être inférieur au gain réalisé sur les circuits de manière à observer un gain total. Il est important de noter que le surplus est observé au niveau des registres alors que le gain est observé au niveau de la logique. Par conséquent, dans les exemples suivants, nous considérerons que chacun des modules étudiés est encadré par deux registres sauf mention contraire.

4.4.1.1 Surplus

Premièrement, intéressons-nous au surplus de logique que cette modification implique. Dans un environnement ASIC, pour chaque porte avec mémorisation de l'état interne cette modification revient à supprimer entre 2 et 12 transistors dépendamment de la porte et du type de design considérés. Le type de design faisant référence aux types statique ou semi-statique discutés dans la sous-partie 4.1.3 de ce chapitre. Dans un environnement FPGA, les ressources nécessaires pour mener à bien cette modification vont dépendre de l'architecture visée. Cela dépend notamment du nombre d'entrées des LUT ainsi que des éléments logiques et de mémoire disponibles. Le Tableau 4.3 présente des relevés d'utilisation de ressources pour un registre NCL et SHF-NCL instanciés sur différentes architectures.

Tableau 4.3. Comparaison des ressources pour un registre de 1-bit

FPGA	Type	# LUT	# Éléments de mémoire
Virtex5 Virtex4	NCL	3	2
	SHF-NCL	5	2
Igloo	NCL	7	2
	SHF-NCL	8	2
ProAsic3	NCL	7	2
	SHF-NCL	8	2

En moyenne cela correspond à un surplus de ressources de 26% par rapport au registre 1-bit NCL. Pour un registre de plusieurs bits, nous avons vu que l'on peut utiliser des signaux (A, C) uniques pour les N bits du registres. De cette manière, le surplus de ressources devient de plus en plus petit avec l'augmentation du nombre de bits N du registre comparativement au nombre total d'éléments. Les tableaux Tableau 4.4 et Tableau 4.5 ainsi que la Figure 4-21, permettent de quantifier cette évolution sur quelques architectures de FPGA. Pour le Tableau 4.4, les résultats

sont exprimés sous la forme « nombre de LUT – nombre de loquets ». Pour l'ensemble des résultats le paramètre N représente le nombre de bits du registre considéré.

Tableau 4.4. Éléments utilisés pour divers N

FPGA	Type	N = 1	N = 2	N = 3	N = 4	N = 5	N = 10	N = 20
Virtex5/4	NCL	3 - 2	5 - 5	9 - 7	13 - 9	15 - 11	30 - 21	59 - 41
	SHF-NCL	5 - 2	5 - 5	9 - 7	13 - 9	15 - 11	30 - 21	59 - 41
Igloo	NCL	7 - 2	16 - 5	24 - 7	32 - 9	40 - 11	82 - 21	160 - 41
	SHF-NCL	8 - 2	17 - 5	25 - 7	33 - 9	41 - 11	82 - 21	163 - 41
ProAsic3	NCL	7 - 2	16 - 5	24 - 7	32 - 9	40 - 11	82 - 21	160 - 41
	SHF-NCL	8 - 2	17 - 5	25 - 7	33 - 9	41 - 11	82 - 21	163 - 41

Tableau 4.5. Surplus en fonction du nombre N de bits

FPGA	N = 1	N = 2	N = 3	N = 4	N = 5	N = 10	N = 20
Virtex5/4	40%	0%	0%	0%	0%	0%	0%
Igloo	11%	4,8%	3,2%	2,4%	2%	0%	1,5%
ProAsic3	11%	4,8%	3,2%	2,4%	2%	0%	1,5%
Moyenne	**20,67 %**	**3,20 %**	**2,13 %**	**1,60 %**	**1,33 %**	**0,00 %**	**1,00 %**

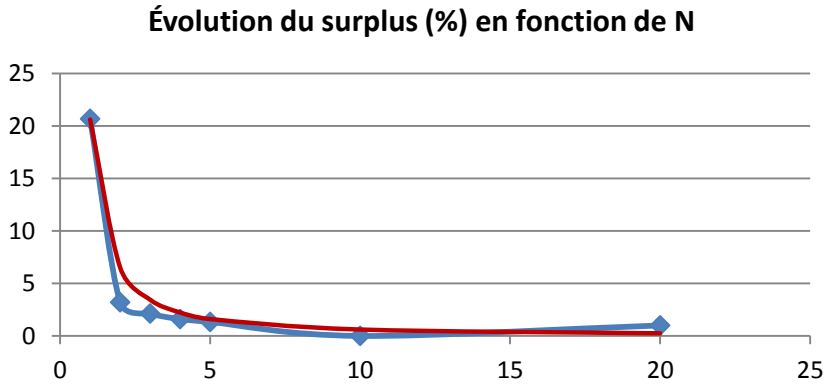


Figure 4-21. Évolution du surplus en % en fonction de N

La Figure 4-21 permet pleinement d'observer une dépendance inversement proportionnelle à N*log(N). En effet, la courbe bleue représente les mesures effectuées alors que la courbe rouge montre une évolution en 1/(N*log(N)). Ceci s'explique par le fait que pour chaque bit ajouté, la complexité des portes de type TH22 augmente de manière proportionnelle à N, de même que la complexité du premier étage de la partie de calcul de la complétude. En revanche pour combiner ces N premiers signaux intermédiaires de calcul de la complétude, il faut un nombre de portes en O(N*log(N)). Le surplus étant un nombre de portes indépendant de N, on obtient la complexité

mentionnée. Ces résultats permettent donc d'appréhender l'évolution du surplus par rapport au nombre et à la taille des registres. On constate de plus que cette évolution dépend du type d'architecture choisi. Pour pouvoir conclure par rapport au potentiel de cette simplification, il est alors nécessaire de comparer ce surplus au gain réalisé dans la partie combinatoire. Pour cela prenons quelques exemples avant de statuer sur le gain maximal théorique. Comme souligné plus haut, les circuits étudiés ci-dessous prennent en compte le surplus induit par un registre d'entrée et un registre de sortie ayant le nombre convenable de bits. De la sorte, les résultats présentés seront pessimistes car le surplus des registres devrait alors être partagé sur les deux étages de logique desquels ils réalisent la jonction. Néanmoins cela nous permet d'obtenir une valeur de gain minimal pouvant être espéré pour chacun des exemples.

4.4.1.2 Porte OU-Exclusif

Le premier exemple que nous utiliserons est celui de la porte OU-Exclusif représentée sur la Figure 4-22 pour des raisons de commodité.

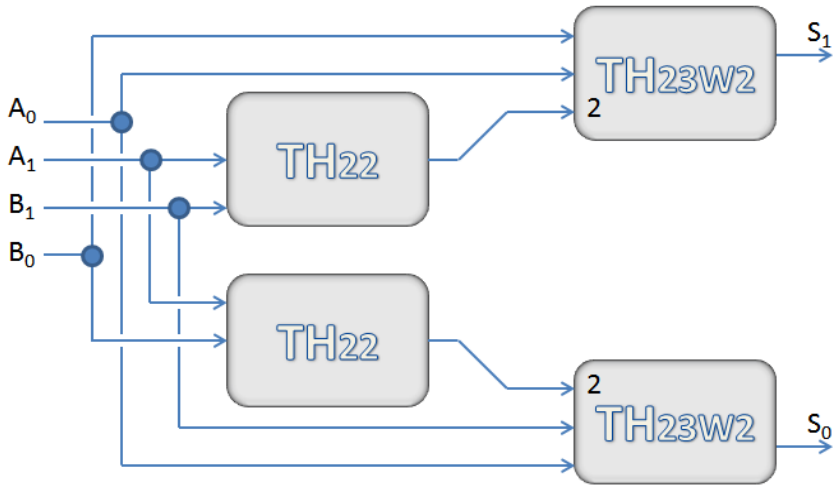


Figure 4-22. Schéma du XOR représenté selon la convention NCL

Les résultats en termes de complexité pour différentes architectures sont donnés dans le Tableau 4.6. La notation employée représente le nombre total d'éléments utilisés puis, entre parenthèses, respectivement le nombre d'éléments logiques et le nombre d'éléments de mémoire. On notera que pour une architecture telle que celle des Igloo d'Actel, un loquet requiert deux éléments de base. On observe alors que pour un circuit aussi peu complexe qu'un porte OU-Exclusif, le gain en termes de ressources peut être aussi élevé que **17%**.

Tableau 4.6. Gain obtenu sur l'exemple de la porte XOR

FPGA	NCL	SHF-NCL	Gain
Virtex5/4	28(11 + 17)	22 (7 + 15)	21,4%
Igloo	49(11*2 + 27)	43(7*2 + 29)	12,2%
Moyenne	**-**	**-**	**16,80 %**

4.4.1.3 Additionneur 1-bit avec retenue

Le second circuit étudié est celui d'un additionneur 1-bit avec retenue présenté dans le Chapitre 3. Les résultats en termes de complexité sur ce circuit sont regroupés dans le Tableau 4.7 pour lequel la notation employée est la même que précédemment.

Tableau 4.7. Gains obtenus sur l'exemple du Full-adder

FPGA	NCL	SHF-NCL	Gain
Virtex5/4	42(16+26)	39(12+27)	7,1%
Igloo	77(16*2+35)	69(12*2 + 45)	10,4%
Moyenne	**-**	**-**	**8,75 %**

Le gain moyen observé en termes de ressources utilisées est aux alentours de **9%** dépendamment du type d'architecture visée.

4.4.1.4 Circuit d'incrémentation

Le cas du circuit d'incrémentation est particulièrement intéressant car il permet de mettre en jeu une succession de registres sans logique entre ceux-ci. Le surcoût dû à la génération des signaux *A* et *C* devrait alors être plus significatif. D'autre part, étant un circuit itératif, il permet d'observer l'impact de l'augmentation du nombre de bits noté N sur le gain. Le circuit utilisé est celui de la Figure A4-9. Concernant l'utilisation des ressources, les résultats pour plusieurs architectures sont présentés dans le Tableau 4.8 pour un nombre variable N de bits.

On peut alors constater que plus le nombre de bits est élevé plus le gain est important. Ce qui semble intuitif en ce sens qu'augmenter le nombre de bits accroît le nombre d'éléments de mémoires dans les registres dans les mêmes proportions pour les deux types de designs mais diminue fortement le nombre de ces éléments dans la logique. De plus le surplus étant fixe, cela implique que l'on peut espérer un gain plus important avec l'augmentation du nombre N.

Tableau 4.8. Gains obtenus pour le circuit d'incrémentation pour différents nombre de bits N

FPGA	Type	N = 1	N = 2	N = 3	N = 4	N = 5	N = 8	N = 16	Gain Moyen
Virtex5/4	NCL	73	107	131	162	186	273	517	8,5%
	SHF-NCL	74	99	122	144	167	240	449	
	Gain	-1,4%	7,5%	6,9%	11,1%	10,2%	12,1%	13,2%	
Igloo	NCL	114	159	202	247	294	427	773	11,7%
	SHF-NCL	114	149	180	215	248	351	629	
	Gain	0%	6,3%	10,9%	13,0%	15,6%	17,8%	18,6%	
Gain moyen	-	-0,7%	6,9%	8,9%	12,5	12,9%	15,0%	15,9%	**10,1%**

4.4.1.5 Unité Arithmétique et Logique

Finalement, le dernier circuit considéré est celui de l'unité arithmétique et logique présentée dans l'ANNEXE 4 et décrite en détails dans [37]. Hormis les deux registres d'entrée et de sortie respectivement de six et trois bits, on peut constater que dans ce design la majeure partie des ressources utilisées l'est dans la logique. Il est donc pertinent de s'attendre à obtenir un gain élevé en appliquant l'amélioration proposée. Les relevés des rapports de synthèse pour ce même design réalisé avec et sans l'amélioration nous permette d'avoir un gain de l'ordre de **49%** sur les deux architectures précédemment utilisées.

En conclusion, au travers de ces quatre exemples, nous avons pu constater que le surplus dû au passage du NCL au SHF-NCL est amplement compensé par le gain réalisé par la suppression des éléments de mémoire dans la logique. On obtient alors un gain substantiel susceptible d'améliorer les performances en termes de débit mais aussi en termes de consommation comme nous tentons de l'observer dans la suite. On notera toutefois que le gain peut varier grandement entre un circuit et un autre. L'analyse du gain maximal théorique présentée dans la sous-partie 4.4.3 tentera d'apporter des réponses à cette observation.

4.4.2 Mesures des performances

Comme nous le mentionnions plus haut, il est envisageable de considérer que la réduction du nombre d'éléments logiques entre deux étages de registres puisse diminuer les temps de

propagation et par conséquent augmenter le débit. D'autre part, en diminuant le nombre d'éléments, la charge est elle aussi susceptible de diminuer et par la même la consommation énergétique pourrait s'en trouver améliorée. Cette partie tente, au travers des deux exemples de l'ALU et du circuit d'incrémentation, d'observer ces allégations.

4.4.2.1 Circuit d'incrémentation

Le premier environnement de test est celui du circuit d'incrémentation. Pour ces relevés, nous implémenterons successivement un design de type NCL puis un design de type SHF-NCL sur le FPGA dont nous disposons : le Actel Igloo AGLN250V2. Nous utiliserons la carte de développement *Igloo nano Starter Kit* d'Actel qui utilise ce FPGA. Cette carte nous permet facilement de relever la consommation des éléments logiques. D'autre part, pour relever les performances en termes de débit, plusieurs techniques sont possibles. Nous choisissons ici de relever directement à l'oscilloscope la fréquence d'oscillation du bit de poids faible du registre de sortie. Les mesures effectuées sont représentées dans les tableaux Tableau 4.9 à Tableau 4.11 ainsi que sur les figures Figure 4-23 à Figure 4-25 pour apprécier les résultats de manière graphique.

Tableau 4.9. Relevé de performance pour le circuit d'incrémentation NCL (à 1,5V)

	Unité	**N = 8**	**N = 12**	**N = 16**	**N = 24**	**N = 32**	**N = 48**	**N = 64**
Fréquence	MHz	8,447	7,249	6,874	5,463	5,413	4,166	3,518
Courant statique	µA	22,1	22,4	22,1	22,1	22,0	22,3	22,1
Puissance statique	µW	33,2	33,6	33,2	33,5	33,0	33,5	33,2
Courant dynamique	µA	1116	1210	1252	1315	1360	1413	1420
Puissance dynamique	µW	1674	1815	1878	1973	2040	2120	2130

Tableau 4.10. Relevés de performance pour le circuit d'incrémentation SHF-NCL (à 1,5V)

	Unité	**N = 8**	**N = 12**	**N = 16**	**N = 24**	**N = 32**	**N = 48**	**N = 64**
Fréquence	MHz	7,575	6,975	7,036	5,988	5,458	4,701	3,760
Courant statique	µA	22,4	22,6	22,6	22,6	22,6	22,5	22,6
Puissance statique	µW	33,6	35,0	33,9	35,85	33,9	34,2	33,9
Courant dynamique	µA	1041	1153	1220	1340	1365	1436	1433
Puissance dynamique	µW	1562	1730	1830	2010	2048	2154	2150

Tableau 4.11. Gains obtenus

	N = 8	N = 12	N = 16	N = 24	N = 32	N = 48	N = 64
Gain en fréquence	-10%	-4%	2%	10%	1%	13%	6%
Diminution de la puissance dynamique	7%	5%	3%	-2%	-0%	-2%	-1%
Diminution en P/f	-4%	1%	5%	7%	>1%	10%	6%
Diminution des ressources	15%	15%	16%	18%	18%	19%	18%

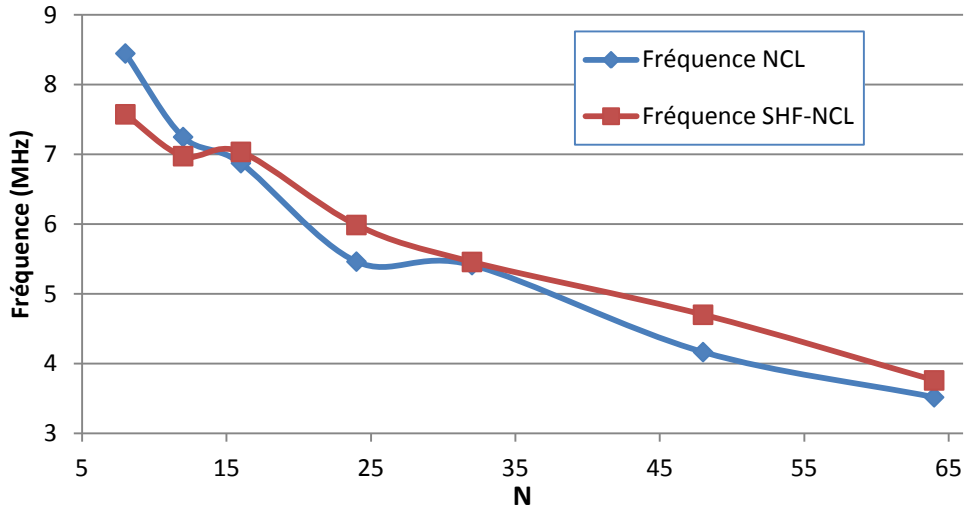


Figure 4-23. Évolution de la fréquence de fonctionnement en fonction de N

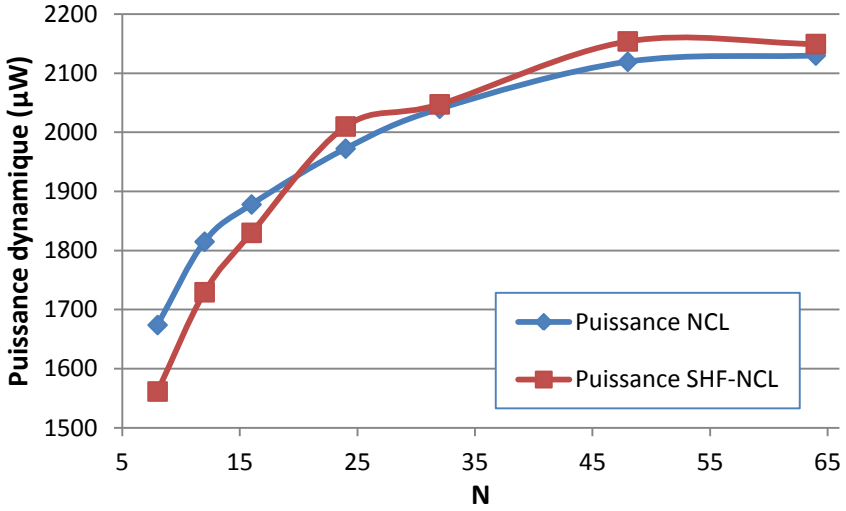


Figure 4-24. Évolution de la consommation dynamique en fonction de N

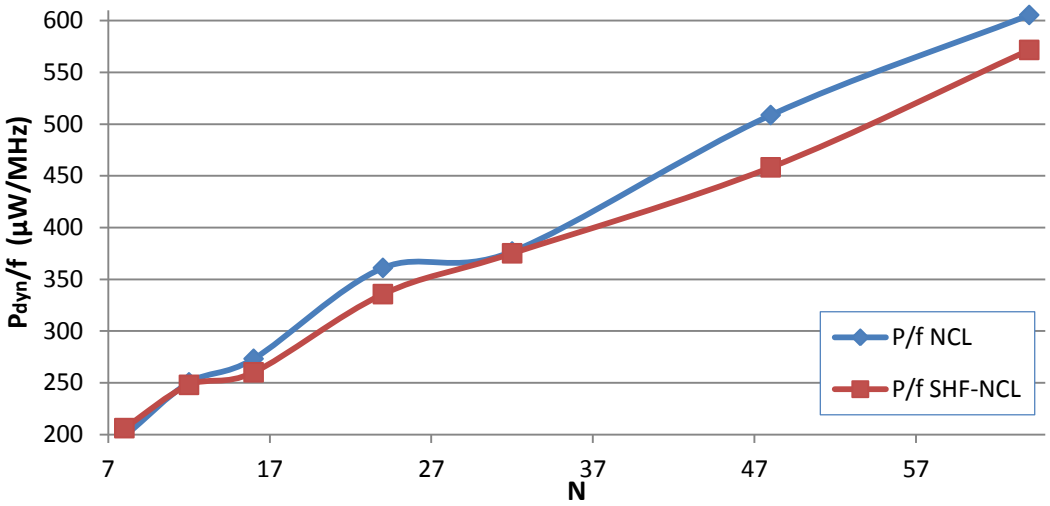


Figure 4-25. Évolution du rapport puissance dynamique / fréquence d'opération en fonction de N

Au vu de ces résultats, il est possible de dire que bien que la réduction de ressources soit franche et facilement observable, les performances dynamiques telles que le débit et la consommation ne respectent pas nécessairement nos prévisions.

Tout d'abord concernant la fréquence d'opération, il est possible d'observer un gain très faible pour des valeurs élevées de N et suffisamment faible pour supposer qu'il est seulement dû aux variations de placement/routage. D'autre part, pour des valeurs de N plus petites, la tendance peut même s'inverser. Quels sont alors les facteurs qui font que la diminution attendue ne peut être observée? En enlevant les éléments de mémoire de la logique NCL, le SHF-NCL diminue le nombre d'éléments à traverser. En contre partie, le protocole de synchronisation local est changé et, pour un étage particulier, celui-ci doit attendre comme précédemment que l'étage suivant l'informe de sa disposition à recevoir un nouveau front de données. Mais, l'étage considéré doit dorénavant attendre de plus que l'étage précédent l'informe de la validité ou de la nullité des informations envoyées. Par conséquent, un certain retard est induit ce qui en moyenne vient contrebalancer le gain supposément obtenu par la diminution des ressources.

Concernant la fréquence d'opération, le même type de phénomène entre en jeu : la diminution des ressources implique une diminution du nombre d'éléments qui commutent et de la charge ce qui devrait impliquer une diminution de la consommation dynamique. En revanche, les observations montrent que cette diminution de la consommation dynamique n'est pas aussi importante que ce que l'on pourrait espérer. Avant de poursuivre, il est utile de souligner que la consommation dynamique dépend de manière linéaire de la fréquence d'opération. Par

conséquent, étant donné que la modification du circuit modifie la fréquence d'opération, nous rajoutons une grandeur calculée à partir des mesures et qui exprime le rapport de la puissance dynamique et de la fréquence d'opération. Comme pour la fréquence d'opération, les résultats ne sont pas concluants. Une explication à cela vient probablement du fait que les éléments de mémoire à seuil de la logique NCL agissent comme des tampons aux variations des signaux. En effet, imaginons une portes ET à deux entrées pour laquelle initialement ses deux entrées sont actives. Si l'une d'entre elle reste fixe alors que l'autre passe par une succession d'états intermédiaires avant de se stabiliser à 0, la sortie commutera en suivant l'entrée instable. L'homologue de la porte ET en logique NCL est la porte TH22. Dans la même situation, la sortie restera fixée à la valeur 1 jusqu'à ce que les deux sorties soient désactivées en même temps. De ce fait, les éléments de logique agissent comme des tampons par rapport aux signaux transitoires réduisant le nombre de commutations et donc la consommation dynamique. À cela s'ajoute le fait que les signaux *A* et *C* ajoutés à chaque registre pour prendre en compte l'état du signal *e* sont deux signaux commutant à la fréquence d'opération du circuit auquel est connectée une charge importante. Ces deux observations compensent en moyenne le gain attendu, résultant en une consommation dynamique quasiment inchangée.

4.4.2.2 Unité Arithmétique et Logique

Nous réitérons les mesures sur un circuit de type unité arithmétique et logique pour lequel nous avons vu que la diminution des ressources est beaucoup plus importante que dans le cas précédent. Ici, il n'est plus possible de mesurer la fréquence d'opération en relevant l'activité du bit de poids le plus faible. Par conséquent, le mode opératoire adopté consiste à fournir des vecteurs de tests aux entrées de l'ALU et à relever le temps entre la prise en compte d'un nouveau front et l'événement correspondant sur le signal K_o. On supposera de plus que le signal K_i répond suffisamment rapidement pour ne pas ralentir le processus et fausser la mesure. Par ailleurs, pour ne pas que les résultats mesurés dépendent du jeu de vecteurs d'entrées, un test exhaustif pour les 16 entrées possibles et pour les quatre opérations différentes a été réalisé plusieurs fois et la moyenne de ces résultats est exprimée dans le Tableau 4.12 ci-dessous. Un test plus poussé consisterait à réitérer le processus pour diverses configurations de placement/routage.

Tableau 4.12. Comparaison de mesures de performances sur l'ALU

	NCL	SHF-NCL	Gain
Délai (ns)	~79	~79	~0%
Fréquence d'opération (MHz)	~6,3	~6,3	~0%
Courant (µA)	22,1	21,9	> 1%

4.4.3 Gain Maximal Théorique et Limite de l'Approche

Comme nous l'avons vu pour les exemples du XOR, du *full-adder* ou du circuit d'incrémentation, le gain en termes de ressources, bien qu'existant, peut paraître limité. On peut alors se demander s'il existe un gain maximal théorique et, dans l'affirmative, comment l'atteindre. Avant de comprendre, comment appréhender le gain maximal théorique reprenons l'exemple du circuit d'incrémentation. Pour ce circuit, on constate que le gain maximal obtenu est inférieur à 20%. Néanmoins, regardons ce qu'il en est en relevant le gain réalisé sur la logique elle-même.

Tableau 4.13. Gain réalisé sur la logique du circuit d'incrémentation

	NCL	SHF-NCL	Gain
N = 8	112	40	64,3%
N = 16	223	80	64,2%
N = 32	448	145	67,6%
N = 64	894	300	66,4%

Le Tableau 4.13 résume le nombre d'éléments logiques utilisés dans les différentes configurations et exprime le gain réalisé. On obtient alors un gain moyen de **65,6%**, soit trois fois plus que le gain obtenu sur l'ensemble. En supposant dans un premier temps que le surplus est négligeable, la principale raison de la diminution du gain est due au fait que l'amélioration proposée diminue l'utilisation de ressources à l'intérieur de la logique mais laisse tels quels les étages de registres. On se retrouve alors avec un équivalent de la loi d'Amdahl que l'on pourrait résumer comme suit : « le gain maximal que l'on atteint en diminuant les ressources dans la logique ne donnera un gain global jamais supérieur au rapport entre le nombre d'éléments dans la logique sur le nombre d'éléments total du circuit ». En effet, considérons les ressources utilisées par un registre pour différentes valeurs de N comme représenté dans le Tableau 4.14:

Tableau 4.14. Ressources utilisées par les registres du circuit d'incrémentation

	Nombre d'éléments
N = 8	100
N = 16	194
N = 32	387
N = 64	774

On peut alors calculer grossièrement le gain réalisé en utilisant la formule suivante :

$$G_{calculé} = \frac{L_{NCL} - L_{eNCL} + Nb_{REG}.surplus}{L_{NCL} + Nb_{REG}.L_{REG}}$$

Dans cette expression, L_i désigne le nombre d'éléments utilisés par la logique en NCL lorsque i = NCL, la logique en SHF-NCL lorsque i = SHF-NCL et les registres lorsque i = REG. Et Nb_{REG} désigne le nombre de registres utilisés. Le surplus est dû à la logique supplémentaire rajoutée dans chaque registre pour prendre en compte le signal *e*. Pour le FPGA AGLN205V2, ce surplus vaut 4. On obtient un gain $G_{calculé}$ qui s'approche de ce que l'on mesure tel que le montre le Tableau 4.15. L'écart entre les deux gains est principalement dû au circuit de combinaison du signal e_i avec un signal interne qui devient négligeable avec l'augmentation de N.

Tableau 4.15. Comparaison du gain calculé avec le gain mesuré

	$G_{calculé}$	$G_{mesuré}$
N = 8	20,4%	15%
N = 16	19,3%	16%
N = 32	19,6%	18%
N = 64	18,8%	18%

Exprimons alors notre problème de manière à ce qu'il s'assimile à l'énoncé de la loi d'Amdahl en définissant les grandeurs suivantes :

- $K_{logique}$: le nombre d'éléments utilisés pour la logique
- $K_{registres}$: le nombre d'éléments utilisés pour les registres
- s : le rapport $K_{logique}/(K_{logique} + K_{registres})$ où $K_{logique} + K_{registres}$ est supposé non nul
- A : le gain réalisé sur la logique

En notant, K_{avant} et $K_{après}$ respectivement les nombres d'éléments utilisés avant et après l'amélioration on a :

$$K_{avant} = K$$

$$K_{après} = (1 - s).K + s.K.(1 - A)$$

On a alors que le gain global et le gain maximal sont donnés par les formules suivantes :

$$G = \frac{K_{avant} - K_{après}}{K_{avant}} = s.A \qquad (1)$$

$$G_{max} = s \text{ lorsque } A = 1 \qquad (2)$$

Dans l'exemple du circuit d'incrémentation, on a $G_{max} \approx 27{,}6\%$. Par conséquent, en ayant un gain moyen mesuré de 16,8% on a atteint plus de 60% du gain maximal. Ce qui est un résultat relativement intéressant. Il est important de noter qu'il est impossible d'obtenir un gain égal à G_{max} dans un cas non trivial puisque cela équivaudrait à avoir une fonction logique réalisée avec aucune porte ($A = 1$). Alors comment faire pour avoir un gain total élevé?

En analysant les équations (1) et (2) on peut en déduire qu'il faut :

- Augmenter *s*
- Augmenter *A*

Avoir un coefficient *s* grand (proche de 1) revient à avoir des circuits pour lesquels le nombre d'éléments dans la logique est très grand par rapport au nombre d'éléments dans les registres. Un mauvais exemple est typiquement le cas du circuit d'incrémentation pour lequel à un étage de logique sont associés trois registres et pour lequel la complexité de ces derniers est en O(N*log(N)) et celle dans la logique est proportionnelle à N.

Pour augmenter A, plusieurs paramètres sont à prendre en compte. Premièrement, l'amélioration permet de supprimer les éléments de mémoires de la logique. Or la norme NCL définit trois portes (TH12, TH13 et TH14) qui n'utilisent pas d'éléments de mémoire. Par conséquent, pour que l'amélioration ait un gain plus important, il est trivial de dire qu'il faut que les circuits initiaux, avant la simplification, utilisent des portes à mémoire. Un autre paramètre important à considérer, est le surplus dû à la prise en compte du signal *e*. Appelons ce nombre α et notons qu'il est dépendant de l'architecture visée. Dans le cas des FPGA de la famille Igloo d'Actel, ce paramètre vaut 4. Et pour chaque étage de registre, on paye un surplus de α. Autrement dit, il est

important de compenser ce surplus par un gain dans de la logique. Ce qui, dans l'exemple du circuit d'incrémentation n'est pas directement réalisé puisque trois étages de registres se succèdent sans logique entre ceux-ci.

Un design plus à même de bénéficier de cette amélioration serait un design où le nombre d'opérations réalisées par rapport au nombre de bits des registres est important, comme par exemple le cas de l'ALU étudié plus haut. En effet, hormis les deux registres d'entrée et de sortie respectivement de six et trois bits, on peut constater que dans ce design la majeure partie des ressources utilisées l'est dans la logique. On peut donc s'attendre à obtenir un gain élevé en appliquant l'amélioration proposée. En mesurant les ressources utilisées par les registres et les ressources utilisées par la logique pure, on obtient un gain maximal théorique de l'ordre de 67%. Les relevés des rapports de synthèse pour ce même design réalisé avec et sans l'amélioration nous permettent d'avoir un gain de **49%** soit 73% du gain maximal théorique.

4.5 Conclusion

En conclusion, l'observabilité, la complétude et l'utilisation de logique positive semblent suffire à obtenir un comportement stable. L'utilisation d'éléments à mémoire fournit cependant un moyen intrinsèque de respecter la complétude par rapport au front NULL et, d'autre part, cela permet de le faire au niveau local diminuant ainsi les problèmes qui pourraient avoir lieu de par l'accumulation de délais par exemple. En supprimant les éléments de mémoire de la logique NCL et en utilisant de la logique combinatoire positive usuelle à la place, la complétude par rapport au front NULL n'est plus respectée. Pour palier ce problème, un signal noté *e* a été ajouté. De ce fait, la complétude par rapport au front DATA n'est plus respectée au niveau local. Comme nous l'avons souligné, ce changement peut affaiblir la robustesse temporelle du circuit sur le plan théorique. Néanmoins, en considérant les contraintes physiques de placement routage ainsi que le temps nécessaire pour générer le signal *e* et fournir un nouveau front DATA, nous avons montré que ce type de problème était peu probable.

Concernant les objectifs de cette simplification, nous avons montré qu'un gain substantiel en termes de ressources pouvait être espéré. De plus, en proposant une analyse théorique du gain maximal, nous avons mis en évidence que cette approche était efficace en ce sens que ce dernier était approché de 60 à 70% près dépendamment des cas. De plus, la conception ainsi que la vérification du code ont été améliorées de par l'utilisation de méthodes et d'outils

conventionnels. Toutefois, alors que des améliorations en termes de consommation et de fréquence d'opération étaient attendues, nous avons vu que celles-ci n'avaient pu être observées et en avons détaillé les causes. À défaut de les améliorer, la simplification proposée n'altère pas ces caractéristiques. Néanmoins, il serait intéressant de compléter ces observations par des expérimentations sur ASIC dans le but d'évaluer un éventuel gain en termes de consommation statique.

On notera que la simplification présentée dans ce chapitre a constitué la base d'un article de conférence qui a été accepté à MWSCAS 2012 et qui est reproduit en ANNEXE 12.

CHAPITRE 5 MODULE DE RÉVEIL ASYNCHRONE

Le Chapitre 3 a introduit la logique asynchrone et plus particulièrement le NULL Convention Logic. Par la suite, dans le Chapitre 4, nous avons détaillé une amélioration de ce paradigme visant à réduire les ressources utilisées pour un circuit donné. Dans ce chapitre, nous utilisons la logique alors présentée afin de réaliser un module de réveil de l'émetteur-récepteur ZigBee et détaillons chaque étape de sa conception. Les résultats présentés ici ont donné naissance à un article de conférence accepté à NEWCAS 2012 et reproduit en ANNEXE 12.

5.1 Présentation Générale

5.1.1 Motivations

Une des caractéristiques importantes d'un *animat* est la capacité de pouvoir communiquer avec ses pairs lorsque ce dernier estime que ceci lui est profitable. Il peut alors décider de former une colonie et, par exemple mettre en commun de l'information et des connaissances. Un des avantages de ce genre de stratégie est de pouvoir, à partir de mesures peu précises faites localement, bénéficier d'autres mesures faites dans d'autres contextes permettant éventuellement une vision du problème moins biaisée. En reprenant l'exemple des colonies de fourmis, on peut constater que parmi ces dernières, seule une faible proportion explore l'environnement à la recherche de nourriture et les autres se contentent de suivre le chemin découvert. Par conséquent, il y a un échange de connaissances, ici chimique, pour réaliser une tâche commune.

Pour la plateforme que nous voulons concevoir, nous avons choisi de doter chacun des *animats* d'un émetteur-récepteur radio utilisant le protocole ZigBee. Ce dernier nous permettant de mettre en place et gérer aisément un réseau au travers duquel les *animats* pourront communiquer. D'autres moyens de communications auraient pu être utilisés comme nous l'avons détaillé dans le Chapitre 2. Néanmoins, ces modules de communications sont très gourmands en termes de consommation comme le résume le Tableau 5.1 pour quelques modules actuellement sur le marché. Ces derniers sont aussi représentés sur la Figure 5-1 à titre d'information.

Tableau 5.1. Caractéristiques de différents émetteur-récepteurs ZigBee

Module	Consommation			Sensibilité
	Émission	Réception	Veille	
SPZB32W1x2.1 STMicroelectronics	32 mA	28 mA	1,3 µA	-95 dBm
CC2531 Texas Instruments	29 mA	24 mA	0,4 µA	-97 dBm
MRF24J40MA Microchip	23 mA	19 mA	2 µA	-94 dBm
MRF24J40MB Microchip	130 mA	25 mA	5 µA	-102 dBm

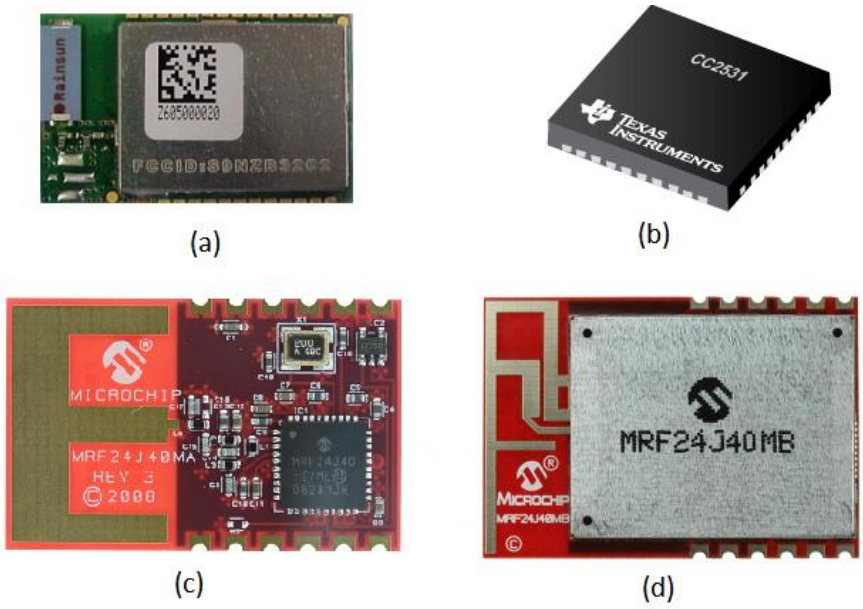


Figure 5-1. Émetteur-Récepteur ZigBee: (a) SPZB32W1x2.1 , (b) CC2531, (c) MRF24J40MA, (d) MRF24J40MB

En considérant la faible taille des *animats* ainsi que leur masse réduite, ce type de consommation est comparable à la consommation nécessaire à leur déplacement. Par conséquent, pour ne pas dégrader leur autonomie, il est nécessaire de proposer un moyen efficace d'utilisation des modes de veille de tels émetteur-récepteurs. Dans le Chapitre 1, nous présentions différents modèles d'énergie relatifs à l'utilisation de la communication. Ici, il est clair que pour minimiser la consommation, les deux modes adéquats sont les modes de communication périodique ou le mode de communication lorsque nécessaire. Le choix entre l'un ou l'autre est dépendant de l'application : pour des *animats* très grégaires et nécessitant de ce fait de partager une quantité

raisonnable d'informations avec leurs proches, le premier modèle pourrait être justifié. Pour des *animats* plus enclin à explorer leur environnement et à se forger leur propre vision de ce dernier, le deuxième modèle serait plus approprié. Dans cette deuxième situation, il est alors nécessaire de doter chaque *animat* d'un module de réveil capable d'écouter le réseau en utilisant un minimum d'énergie et capable de réveiller lorsque nécessaire l'émetteur-récepteur principal.

5.1.2 Fonctionnement Général

Pour pouvoir remplacer un émetteur-récepteur qui écoute périodiquement le réseau par un module moins énergivore écoutant en permanence le réseau, il est important de comprendre pourquoi les émetteur-récepteurs traditionnels consomment autant. On peut distinguer trois parties principales responsables de cette consommation.

Tout d'abord, les modules du Tableau 5.1 revendiquent certaines spécifications sur le seuil de puissance entrante détectée : c'est la sensibilité. Généralement, pour être en mesure de détecter des signaux entrants ayant une puissance aussi faible que -99dBm, les constructeurs de tels modules utilisent des amplificateurs.

Dans un second temps, pour démoduler l'onde entrante et la ramener dans la bande de base des multiplicateurs de signaux sont utilisés. En effet, sans rentrer dans les détails, les équations simplifiées de la modulation de fréquence sont :

$$y_m(t) = y_t(t).e^{if_p t}$$

$$y_d(t) = y_m(t).e^{-if_p t} = y_t(t)$$

On représente ici par y_t le signal à transmettre, par y_m le signal modulé, par y_d le signal démodulé et enfin par f_p la fréquence de la porteuse. Ces équations sont simplifiées et ne tiennent pas en compte des problèmes de déphasage ou encore des stratégies pour ramener le signal non pas directement en bande de base mais à des bandes plus élevées pour poursuivre le traitement numériquement. Néanmoins, cela nous permet de constater que deux multiplieurs sont nécessaires aussi bien pour la modulation que pour la démodulation. Dans notre cas, la fréquence de la porteuse est de l'ordre de 2,4 GHz. Ceci conduit à des consommations dynamiques, nécessaires par ces modules, relativement élevées.

Enfin, la troisième partie susceptible de consommer de l'énergie est le contrôleur lui-même. En effet, une fois ramenée dans une bande permettant le traitement numérique, plusieurs traitements sont réalisés sur l'onde. Notamment, celle-ci doit vérifier les contraintes imposées par la norme IEEE 802.15.4 pour pouvoir être traitée par les couches plus abstraites du protocole ZigBee. On constate néanmoins que devant la consommation des deux premières parties, la consommation des parties numériques est faible.

Par conséquent, un module de réveil devrait, autant que faire se peut, minimiser les dépenses énergétiques selon ces trois axes notamment en utilisant des techniques aux performances dégradées mais qui seront compensées une fois l'émetteur-récepteur principal mis en route. De plus, schématiquement un tel circuit de réveil se situe dans l'architecture générale comme sur la Figure 5-2 où celui-ci est situé par rapport aux autres modules de la plateforme.

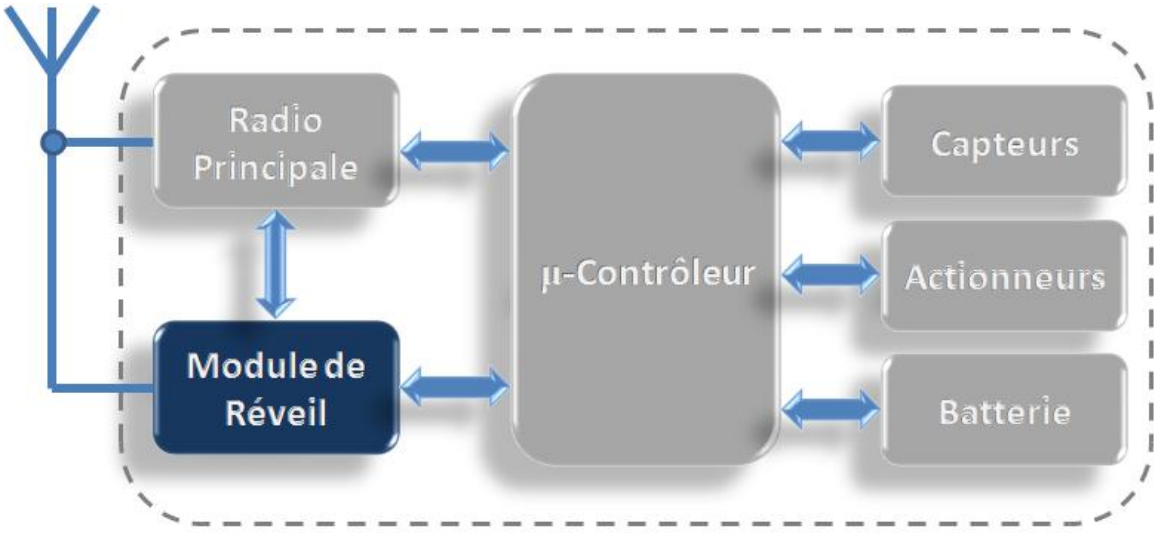


Figure 5-2. Schéma de l'architecture générale de l'*animat*

5.2 Présentation Générale

Avant d'aller plus loin dans le détail de la partie numérique qui nous intéresse, il convient de discuter de la partie analogique d'un tel module. En effet, pour les émetteur-récepteurs standards comme ceux présentés dans le Tableau 5.1, une grande partie de la consommation se trouve être nécessaire pour alimenter d'une part la circuiterie de démodulation et d'autre part pour alimenter les amplificateurs utiles pour assurer une certaine sensibilité. Pour notre part, la principale caractéristique d'un module de réveil devrait résider dans l'effort particulier réalisé dans le but de

diminuer la consommation. De ce fait, il est nécessaire de modifier le circuit et plus particulièrement les deux parties susmentionnées.

5.2.1 Signal de réveil

Le protocole ZigBee définit plusieurs fréquences d'émission dépendamment du pays dans lequel il est utilisé et des choix du constructeur comme présenté dans le Chapitre 1. Pour ce projet, nous avons choisi d'utiliser une porteuse autour de 2,4GHz principalement car elle est autorisée partout sur le globe. A cette fréquence, l'onde est modulée en O-QPSK : *Offset Quadrature Phase-Shift Keying*. L'information est donc codée sur la phase de la porteuse et on distingue, pour cette modulation, quatre valeurs de phases distinctes.

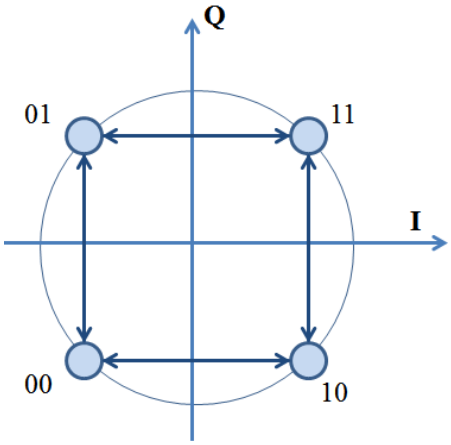


Figure 5-3. Modulation O-QPSK

Le principal avantage de cette modulation est, en plus d'augmenter la bande passante, de minimiser les fluctuations d'amplitude à la réception en n'acceptant de pouvoir passer que d'une phase à une autre par une différence de 90°. Ceci implique que les composantes I et Q ne changent jamais en même temps comme représenté sur la Figure 5-3.

Pour le signal de réveil, nous utiliserons une modulation différente pour diminuer la complexité de démodulation. Nous choisissons alors une modulation de type *Amplitude-Shift Keying* (ASK). Ce type de modulation à l'énorme avantage d'être facile à mettre en œuvre et d'être simple à démoduler, comme nous le verrons plus bas. En effet, l'information est codée sur des variations de l'amplitude et il suffira dans notre cas de détecter les niveaux de tensions reçues. Nous avons choisi une des modulations d'amplitude la plus simple : le *On-Off Keying* (OOK). Ici l'information est simplement codée sur le temps de détection d'un niveau logique. Par exemple, imaginons que l'on définisse un temps t_0 avec une précision ε_0 et un temps t_1 avec une précision ε_1 de telle sorte que lorsque l'on détecte une onde on mesure sa durée t et on obtient l'information de la manière suivante :

$$\begin{cases} si\ |t - t_0| \leq\ \varepsilon_0\ alors\ un\ '0'\ est\ reçu \\ si\ |t - t_1| \leq\ \varepsilon_1\ alors\ un\ '1'est\ reçu \\ sinon, on\ ignore\ la\ donnée \end{cases}$$

En choisissant t_0, ε_0, t_1 et ε_1 de telle manière que $t_0 \pm \varepsilon_0$ soit très différent de $t_1 \pm \varepsilon_1$ il devient possible d'utiliser une référence de temps peu précise qui suffit à distinguer correctement les deux symboles. C'est par exemple le même principe qui est utilisé pour le code Morse. Le fait d'ignorer les symboles pour lesquels les contraintes temporelles ne sont pas respectées permet de se prémunir de fausses alarmes qui pourraient être générées par le trafic normal.

Supposons que l'on choisisse une période T d'état bas et qu'un '0' soit représenté par un état haut d'une durée T, alors qu'un '1' soit représenté par un état haut d'une durée 2T, où T devrait être choisie en tenant compte des considérations physiques présentées plus bas. Alors il suffit de mesurer le temps de réception d'un état haut pour savoir quel symbole a été reçu. La Figure 5-4 représente le schéma de principe de la démodulation ainsi qu'un exemple de signal reçu et de déchiffrage de l'information.

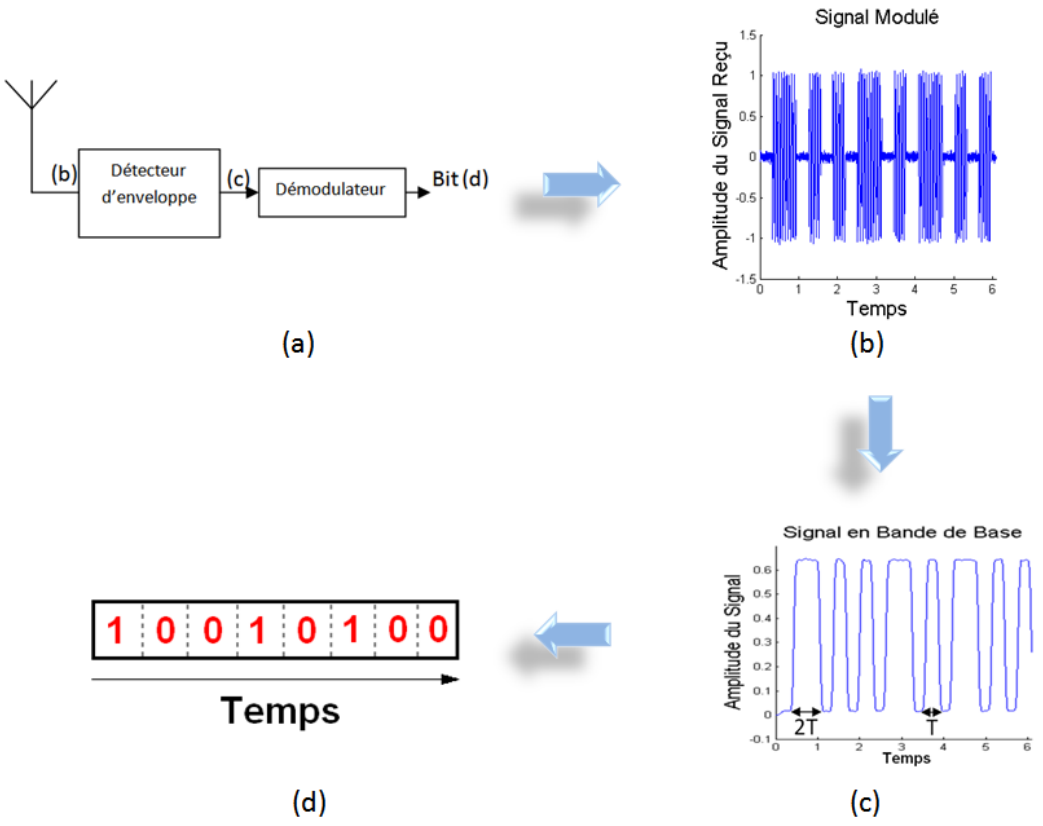


Figure 5-4. Chaîne de réception du message de réveil : (a) Schéma de principe de la démodulation du signal reçu, (b) Signal reçu par l'antenne pour un SNR = 30 dB, (c) Signal en sortie du détecteur d'enveloppe, (d) Information reçue après décodage

Plusieurs paramètres et compromis doivent être pris en compte pour un tel module de réveil. En plus de devoir recevoir les messages de réveil, le module doit être capable de les démoduler correctement, limiter les fausses alarmes et minimiser le taux de non-détection de messages de réveil. En effet, les deux premières remarques constituent la fonction même du module. De plus, celui-ci devrait pouvoir permettre de minimiser la consommation totale du système en générant une interruption capable de réveiller la radio principale, par conséquent un taux de fausses alarmes élevé provoquerait un réveil de la radio principale plus fréquent ce qui irait à l'encontre de l'objectif de basse consommation du système. D'autre part, il faut aussi éviter une non-détection des messages pour éviter que le nœud à l'origine de l'émission du message de réveil ne réitère sa requête et consomme par la même plus d'énergie que nécessaire. Un taux de non-détection élevé pourrait aussi induire une perte d'information dans le réseau dès lors que l'on ne peut pas atteindre le destinataire visé.

De plus, la latence constitue un autre paramètre à prendre en compte. En effet, le fait de rajouter un tel module et un autre protocole de communication en amont de la transmission normale peut amener à une augmentation de la latence. Premièrement car, dorénavant, avant d'émettre il faut au préalable avoir réveillé son destinataire. Deuxièmement, le décodage prend un certain temps contribuant aussi à une augmentation de la latence. Il faut néanmoins tenir compte que dans le schéma classique de communication que l'on retrouve communément dans les WSNs, les RFD se réveillent périodiquement et questionne leur parent direct pour savoir s'ils ont reçu des messages pendant leur période d'inactivité. Ceci implique aussi une certaine latence entre l'émission d'un message et sa prise en compte effective. Néanmoins, de manière à limiter celle-ci, le choix de T devrait être minutieusement réalisé. Pour ce faire, nous présentons tout d'abord le choix du format utilisé pour le message de réveil en gardant comme point de vue l'objectif de minimisation de la consommation d'énergie. Par conséquent, nous ne chercherons pas dans un premier temps à effectuer des vérifications de CRC ou toute autre vérification de l'intégrité du message. Ce qui limite par ailleurs, la quantité de matériel nécessaire et diminue la latence. En revanche pour éviter d'avoir un taux de fausses alarmes élevé, nous ajouterons en début de message un préambule particulier qui servira uniquement à s'assurer du type de message, limitant ainsi la confusion avec le trafic normal. Nous choisissons ainsi un préambule d'un octet ce qui présente l'avantage d'être suffisamment long pour éviter les fausses alarmes et relativement court, évitant ainsi d'accroître démesurément la latence. Ensuite, un bit supplémentaire

permettant de savoir si l'adresse qui sera transmise par la suite est courte (16 bits) ou longue (64 bits) est ajouté. Rappelons que le protocole ZigBee propose ces deux types d'adressage la première ayant une validité à l'intérieur d'un réseau donné alors que la deuxième est globale. Enfin, pour déterminer le début et la fin d'un message de réveil, toute trame devra être espacée de la trame précédente d'une période d'inactivité de 5T. Le format de la trame retenu est représenté sur la Figure 5-5.

Préambule	Sélecteur	Adresse de destination	
10110010	1	Adresse de 16-bits	(a)
10110010	0	Adresse de 64-bits	(b)

Figure 5-5. Format de la trame du message de réveil: (a) Adresse de 16-bits, (b) Adresse de 64-bits

On s'aperçoit alors que la taille maximale du message transmis est de 73 bits plus les 5T d'*Interframe Gap*. On remarquera le choix d'un sélecteur nul pour les adresses de 64 bits a été fait dans l'optique de diminuer la latence car les '0' sont codés par un niveau haut pendant T. Le pire cas en termes de latence est alors donné pour une adresse de 64 bits constituée seulement de '1' donnant alors :

$$T_{rec} = T_{préambule} + T_{\overline{sél}} + T_{adr=\overline{0}} + T_{GAP}$$

$$\Leftrightarrow T_{rec} = 20 * T + 2 * T + 64 * 3 * T + 5 * T$$

$$\Leftrightarrow T_{rec} = 219 * T$$

En imposant que la transmission dans le pire cas prenne 10 ms, cela donnerait $T = 45.7\ \mu s$.

En utilisant un oscillateur pour mesurer le temps d'un état haut avec une résolution de 10 points pendant T, ceci conduirait à choisir un oscillateur de fréquence minimum d'environ 220 kHz. Une deuxième idée est d'utiliser des composants externes pour mesurer le temps comme, par exemple, la tension aux bornes d'un condensateur qui se charge au travers d'une résistance ou encore une ligne à délais, le tout commandé par un module asynchrone. Ces deux solutions synchrones et asynchrones seront détaillées et comparées sur le plan énergétique lors de la discussion.

5.2.2 Démodulation

Considérons le problème de démodulation : par cela nous entendons le fait de ramener l'onde modulée à la fréquence de la porteuse dans la bande de base puis la conversion de l'information sous forme binaire. Le fait d'utiliser une fréquence plus élevée, appelée fréquence de porteuse, pour moduler le signal permet d'atteindre de meilleures caractéristiques de propagation de l'onde notamment en termes de réflexion et d'interférences. Néanmoins, l'utilisation directe d'ondes modulées est à proscrire. Par exemple, pour le cas d'ondes ZigBee à 2,4GHz, ne pas démoduler l'onde avant de l'échantillonner impliquerait selon le *théorème d'échantillonnage de Nyquist-Shannon* d'utiliser une fréquence d'échantillonnage strictement supérieure à 4,8 GHz! À une telle fréquence d'opérations une forte contrainte sur le débit de calcul est imposée et la consommation dynamique augmente de plusieurs ordres de grandeurs prohibant ainsi l'utilisation directe de l'onde modulée.

Par conséquent, considérons tout d'abord la technique utilisée pour ramener l'onde modulée en bande de base. Dans les circuits traditionnels, nous avons vu que la technique communément utilisée revient à multiplier au sens complexe l'onde reçue par le complémentaire de la porteuse. Pour éviter d'utiliser ce type de circuiterie généralement trop énergivore pour nos contraintes, nous décidons d'utiliser un circuit beaucoup plus simple qu'est le détecteur d'enveloppe. L'utilisation de ce dernier est rendue possible par le choix d'une modulation OOK. La Figure 5-6 représente le schéma général d'un tel détecteur.

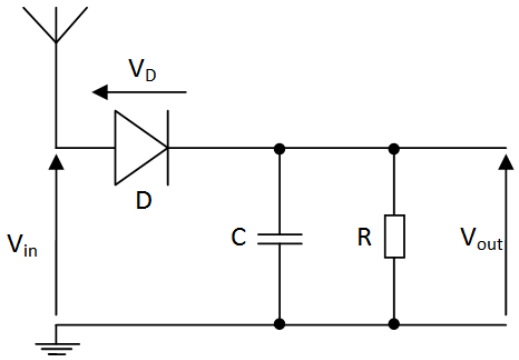


Figure 5-6. Schéma général d'un détecteur d'enveloppe

L'avantage principal de ce type de circuit vient du fait qu'il n'est constitué que d'éléments passifs le rendant très intéressant sur le plan énergétique.

Lorsqu'une onde est reçue par l'antenne, le champ électromagnétique induit une tension V_{in} qui apparait aux bornes du circuit. Mis à part le bruit dû au canal de communication, V_{in} est une onde sinusoïdale de fréquence de porteuse 2.4GHz dont l'amplitude est modulée par l'information que l'on cherche à transmettre. Nous pouvons donc écrire :

$$V_{in}(t) = A(t).\sin(2\pi ft), avec\ f = 2.4GHz$$

Dans notre cas, la modulation étant de type OOK, nous avons :

$$A(t) \in \{0\ ;1\}$$

Le but du circuit RC est d'agir comme une mémoire de la valeur maximale à basse fréquence du signal d'entrée de ce dipôle, ce qui représente finalement l'amplitude du signal ou encore l'information que l'on cherche à extraire. Néanmoins, on convient que sans la diode, des valeurs positives et négatives apparaissent à l'entrée de ce dipôle et V_{out} suivrait V_{in}. Par conséquent, l'ajout de celle-ci permet de considérer uniquement la partie positive du signal modulé d'entrée. Par conséquent, sur le premier lobe du sinus le condensateur C se charge puis lorsque la tension diminue puis devient négative, celui-ci agît comme une mémoire de la valeur maximale du signal passé. La décharge du condensateur peut être vue comme une certaine fuite de mémoire permettant de détecter de faibles amplitudes sur le long terme.

En effet, supposons que A(t) = 1, alors pendant une durée k_1, V_{out} suit V_{in} à la tension de seuil de la diode près et le condensateur se charge jusqu'à ($V_{in} - V_{th}$). Lorsque cette tension est atteinte, si le condensateur ne se décharge pas suffisamment vite, alors la diode va être polarisée en inverse lorsque V_{in} diminue et sera de ce fait bloquée. Par conséquent, le condensateur se décharge progressivement au travers de la résistance. Par la suite, V_{in} continue son oscillation jusqu'à ce qu'elle devienne plus grande que V_{out}. La diode redevient alors polarisée en sens direct ce qui conduira à recharger le condensateur.

Considérons alors les équations régissant le fonctionnement dans le but de pouvoir dimensionner le circuit. Pour ce faire on distinguera trois phases correspondant à l'intervalle k_1 qui est le premier intervalle de temps théorique pendant lequel le condensateur est déchargé et la tension V_{in} commence à croître, l'intervalle de temps pendant lequel V_{in} est plus petit que V_{out} que l'on notera k_2 et finalement l'intervalle k_3. La Figure 5-7 propose une représentation graphique de ces

intervalles. Une expression analytique peut être fournie pour ces différents intervalles en fonction de la fréquence du signal d'entrée et des composants choisis.

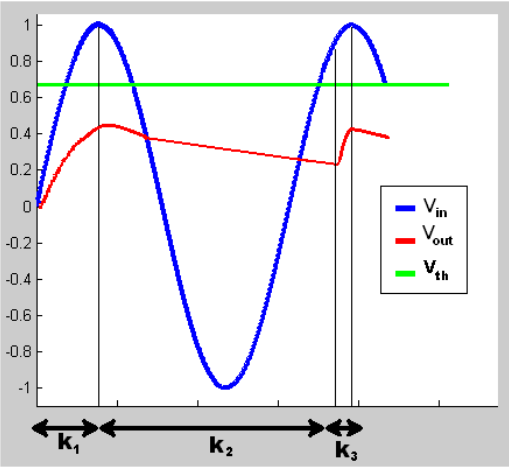


Figure 5-7. Tensions dans le détecteur d'enveloppe et définition des intervalles

Nous représenterons le fonctionnement de la diode par l'équation de Shockley :

$$i_D = I_s.(e^{V_D/nV_T} - 1)$$

Où i_D, I_S, V_D, V_T et n représentent respectivement le courant qui traverse la diode, le courant de saturation en polarisation inverse, la tension aux bornes de la diode, le potentiel thermique définit en thermodynamique par kT/q et le facteur de qualité qui dépend essentiellement du procédé de fabrication.

En notant i_R et i_C les courants dans R et C et en utilisant la loi des mailles et la loi des nœuds on arive à l'équation suivante :

$$\frac{di_R}{dt} = \frac{1}{RC}\left(I_s.\left(e^{(V_{in}(t)-R.i_R)/nV_T} - 1\right) - i_R\right), \forall t$$

Connaissant i_R on obtiendrait directement V_{out} en multipliant par R. Néanmoins, comme on peut le constater, cette équation est particulièrement complexe à résoudre notamment à cause de non-linéarités. On pourrait chercher à linéariser au premier ordre l'exponentielle et résoudre l'équation. Ici, on préfère utiliser une autre modélisation : celle de la diode parfaite. On dira alors que si $V_D > V_{th}$, avec V_{th} une certaine tension de seuil, alors la diode est passante, et $V_D \approx V_{th}$. Sinon la diode est bloquée, c'est-à-dire que i_D est nul. On fait comme dernière hypothèse que C est complètement déchargé à t = 0 et que A(t) = 1 pour tout t.

Sur l'intervalle k_1 : Tant que V_{in} est inférieur à V_{th} la diode est bloquée et rien ne se passe dans notre modèle. Dans la réalité, à cause du faible courant qui passe au travers de la diode (cf. équation de Shockley), le condensateur commence peu à peu à se charger. Notons t_{th} l'instant à partir duquel V_{in} devient supérieur à V_{th}. Entre cet instant et la borne maximale de k_1, le condensateur se charge très rapidement au travers de la faible résistance interne de la diode, ce qui donne l'impression que V_{out} suit ($V_{in} - V_{th}$).

Sur l'intervalle k_2 : Le condensateur se décharge dans la résistance R avec un temps caractéristique donné par $\tau = RC$. En écrivant les équations de base d'un tel montage on obtient qu'après un intervalle de temps de longueur $|k_2|$ la tension aux bornes du condensateur C est donnée par :

$$V_{out} = V_{max}.e^{-\frac{|k_2|}{\tau}}$$

où $V_{max} = (V_{in,max} - V_{th})$.

Sur l'intervalle k_3 : Les équations restent les mêmes que sur k_1, sauf que le condensateur doit se charger entre $V_{max}.e^{-\frac{|k_2|}{\tau}}$ et V_{max}.

On obtient donc qu'en moyenne, lorsque A(t) = 1, V_{out} est environ égal à $\frac{V_{max}}{2}.(e^{-\frac{|k_2|}{\tau}} + 1)$ que l'on peut approximer à V_{max} lorsque $\frac{|k_2|}{\tau} \ll 1$.

Pour assurer le bon fonctionnement du montage, il faut prendre en compte deux phénomènes : les « Ripples » et le « Negative Peak Clipping » qui sont illustrés sur la Figure 5-8.

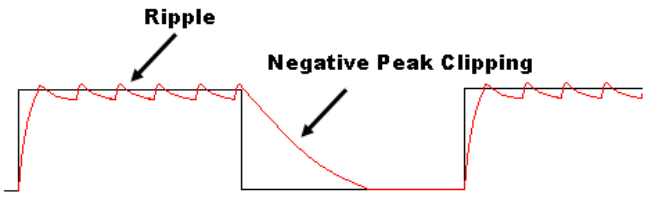


Figure 5-8. Illustration des « Ripples » et du « Negative Peak Clipping »

Ces deux phénomènes sont dominés par le temps caractéristique du montage RC comme nous l'avons vu dans les équations précédentes. Par conséquent, il faut choisir $\tau = RC$ de telle manière que : $\frac{1}{f} \ll \tau \ll T$ où f est la fréquence de la porteuse et T est la constante de temps choisie pour la modulation OOK. En effet, la partie de gauche de l'inéquation assure que, lorsque la diode est

passante, la tension V_{out} ne suit pas V_{in} mais l'enveloppe de celle-ci. Ceci est permis par un temps de décharge relativement long par rapport à la période de la porteuse. La deuxième partie de l'inéquation assure que durant l'état bas de l'onde modulée, on a eu le temps de décharger le condensateur. Dans notre cas, on a $f = 2.4GHz$ et $T = 45.7\mu s$, ce qui donne :

$$4.2 * 10^{-10} \ll \tau \ll 45.7 * 10^{-6}$$

Nous choisissons, comme critère de sélection, de prendre τ tel qu'il existe un unique coefficient de proportionnalité qui le sépare à la fois de sa borne inférieure et de sa borne supérieure :

$$\frac{\tau}{4.2 * 10^{-10}} = \frac{45.7 * 10^{-6}}{\tau}$$

Ce qui donne $\boldsymbol{\tau \approx 1.39 * 10^{-7} s}$.

Avec des valeurs normalisées de résistance et de condensateur et en considérant la taille des composants que l'on souhaite minimiser, on peut choisir :

$$R = 8.66\Omega \; et \; C = 0.016\mu F$$

Respectivement avec les technologies suivantes pour la résistance et le condensateur.

On constate aussi que le condensateur se charge à la tension V_{max}. Or en choisissant une diode standard la tension de seuil est habituellement comprise entre 0.6 et 0.8V ce qui nécessiterait que la tension générée par l'antenne soit supérieure à celle-ci. Ceci est d'autant plus difficile sur de longue distance. Un choix technologique serait alors d'utiliser une diode ayant une faible tension de seuil comme une diode Schottky. Une autre possibilité consiste à ajouter un amplificateur qui dégraderait les propriétés énergétiques attrayantes du circuit.

5.2.3 Sensibilité

Le problème introduit plus haut sur la capacité de l'antenne de générer des signaux d'amplitudes suffisamment élevées est déterminant dans l'étude de la sensibilité. Tout d'abord, dans l'optique de transférer un maximum de puissance entre l'antenne et le reste du circuit, une adaptation d'impédance doit être réalisée. Prenons l'exemple simple de la Figure 5-9 pour illustrer ceci.

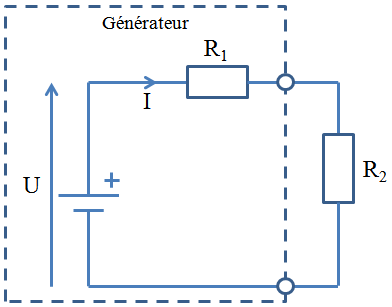


Figure 5-9. Schématisation du problème d'adaptation d'impédance

On peut considérer l'antenne utilisée comme un générateur de tension ayant une résistance interne de valeur R_1 = 50 Ω. Alors, en exprimant l'expression de la puissance transmise à R_2 on obtient l'équation suivante :

$$P_{R_2} = R_2 . I^2 = \frac{R_2}{(R_1 + R_2)^2} . U^2$$

Pour U et R_1 constantes, l'expression de P_{R_2} est une fonction quadratique concave, par conséquent, son maximum est donné en annulant sa dérivée :

$$\frac{dP_{R_2}}{dR_2} = \frac{(R_1 + R_2)^2 - 2.R_2.(R_1 + R_2)}{(R_1 + R_2)^4} . U = \frac{R_1^2 - R_2^2}{(R_1 + R_2)^4} . U = 0$$

R_2 acceptant des valeurs positives ou nulles, l'unique solution de ce problème lorsque U ≠ 0 est donnée par : $\boldsymbol{R_2 = R_1}$. La Figure 5-10 représente l'évolution de P_{R_2} pour U = 1 V et R_1 = 50 Ω.

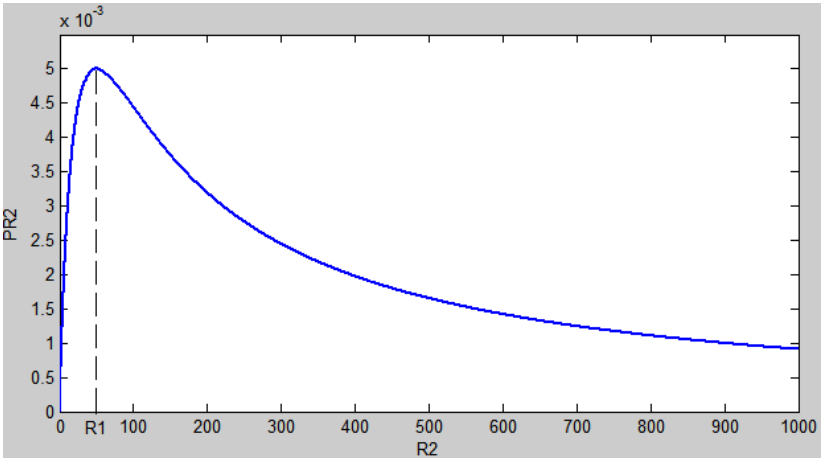


Figure 5-10. Évolution de P_{R2} en fonction de R_2

Par conséquent, il est nécessaire que l'antenne « voit » à ses bornes une charge égale à 50 Ω. Pour ce faire, il est possible d'utiliser des composants discrets passifs tels que des condensateurs

et des bobines pour adapter l'impédance. Pour le reste du calcul, appelons, r_{ant} la résistance interne de l'antenne, Z_{dec} l'impédance du détecteur d'enveloppe, Z_L et Z_C les impédances de la bobine et du condensateur utilisés pour réaliser l'adaptation d'impédance. Afin de ne pas surcharger le reste de la discussion de développements mathématiques, nous nous contentons ici de donner l'idée générale du développement ainsi que le résultat. Pour réaliser l'adaptation on peut choisir une bobine et un condensateur de valeurs telles que l'impédance équivalente $Z_{éq}$ du circuit connecté à l'antenne soit égale à r_{ant} pour assurer une transmission de puissance maximale. Dans le cas, où l'antenne aurait une impédance complexe notée Z_{ant}, on chercherait $Z_{éq} = \overline{Z_{ant}}$. Tous calculs faits, on obtient :

$$Z_{ant} = r_{ant} = \frac{Z_{\tilde{c}}(Z_L + Z_{dec})}{Z_{\tilde{c}} + Z_L + Z_{dec}} \qquad (A)$$

Deux choix sont généralement possibles pour une telle adaptation : mettre un condensateur en série avec le circuit à adapter et une bobine en parallèle, ou bien faire l'inverse. Il en résulte que le premier circuit agît comme un filtre passe-haut et le second comme un filtre passe-bas. Compte tenu du fait que l'objectif du détecteur d'enveloppe est de supprimer les hautes fréquences, il est préférable d'utiliser le montage de type passe-bas.

$$Z_{dec} = r_{diode} + \frac{R(1 - jRC\omega)}{1 + (RC\omega)^2} \qquad (B)$$

La résistance interne de la diode peut être négligée (mesurée à 0.2Ω). En remplaçant l'expression de Z_{dec} donnée dans (B) dans l'équation (A) et pour $\omega = 2 * \pi * 2.45e9$, on obtient avec Matlab:

$$L \approx 0.84\, nH$$

$$\tilde{C} \approx 5\, pF$$

L'adaptation d'impédance est une des premières étapes de l'amélioration de la sensibilité. Néanmoins, les signaux perçus aux bornes de l'antenne sont faibles et bruités. Par conséquent, il peut être envisageable d'ajouter des circuits dans le but d'augmenter la sensibilité. Dans le reste de la discussion on supposera toujours qu'un circuit d'adaptation d'impédance est utilisé mais ne sera pas représenté. Nous présentons donc certains de ces circuits en insistant sur les compromis et surcoûts que ceux-ci impliquent. Dans la suite, nous désignerons par ED le circuit de détection d'enveloppe présenté plus haut. La Figure 5-11 représente l'ensemble des circuits dont nous

allons discuter. L'ajout du filtre passe bas de la solution (a) permet de limiter l'amplitude des «Ripples» présentés plus haut. Le principal inconvénient est le suivant : si le filtre est mal dimensionné, cela peut augmenter le phénomène de «Negative Peak Clipping». Le montage (b) propose l'ajout d'un filtre passe bande en entrée de l'ED. Ceci permet essentiellement de filtrer le signal provenant de l'antenne de manière à s'affranchir des interférences qui pourraient survenir en présence de périphériques communiquant sur d'autres bandes de fréquences. Généralement, l'antenne est conçue pour réaliser ce filtrage. Si tel n'est pas le cas ce circuit permet de filtrer le signal d'entrée et il est possible de choisir les valeurs des composants R, C_1 et C_2 de manière à laisser passer la plage de fréquences qui nous intéresse à savoir 2.4GHz – 2.48GHz. Le circuit (c) permet de rétablir le niveau logique détecté par le module ED. En effet, rappelons que la partie analogique que nous concevons doit pouvoir générer un signal qui puisse être pris en compte par la partie numérique du design. Dans ce cas, il faut que la tension de sortie soit suffisamment élevée pour être traitée numériquement. Ce montage permet alors, dans la mesure où le signal peut être distingué du bruit, de comparer la valeur de sortie du détecteur d'enveloppe à une tension de référence et de générer un résultat qui puisse être pris en compte. Le principal inconvénient de ce montage, tout comme le montage (d), c'est qu'il consomme de l'énergie alors que les autres sont réalisés avec des composants passifs. Le montage (d) propose une solution qui a pour but d'amplifier le signal d'entrée. Ceci permet notamment de pouvoir traiter des signaux plus faibles en entrée, autrement dit cela permet d'augmenter la portée de détection des messages de réveil. Pour les montages (c) et (d), il y a donc un compromis à faire entre performance et consommation d'énergie. Finalement le montage (f), qui n'est autre qu'une pompe à charge de Dickson, utilise comme module de base le montage (e). Ce montage permet d'amplifier le signal d'entrée de manière passive en stockant progressivement de l'énergie. Néanmoins, contrairement à ses homologues actifs (c) et (d), le gain en consommation d'énergie doit être pondéré par une latence accrue. En effet, dépendamment du nombre de cellule de base que l'on cascade, le temps pour charger l'ensemble des condensateurs augmente. D'autre part ce circuit est efficace si les diodes sont des diodes Schottky idéales ce qui, dans la pratique, n'est pas le cas. La présentation réalisée ici se veut informative et doit permettre à quiconque désireux d'augmenter les performances énergétiques du circuit de pouvoir faire un choix éclairé. Pour la suite, nous pensons qu'il est préférable d'utiliser une combinaison des solutions (c) et (d) en utilisant des amplificateurs peu énergivores.

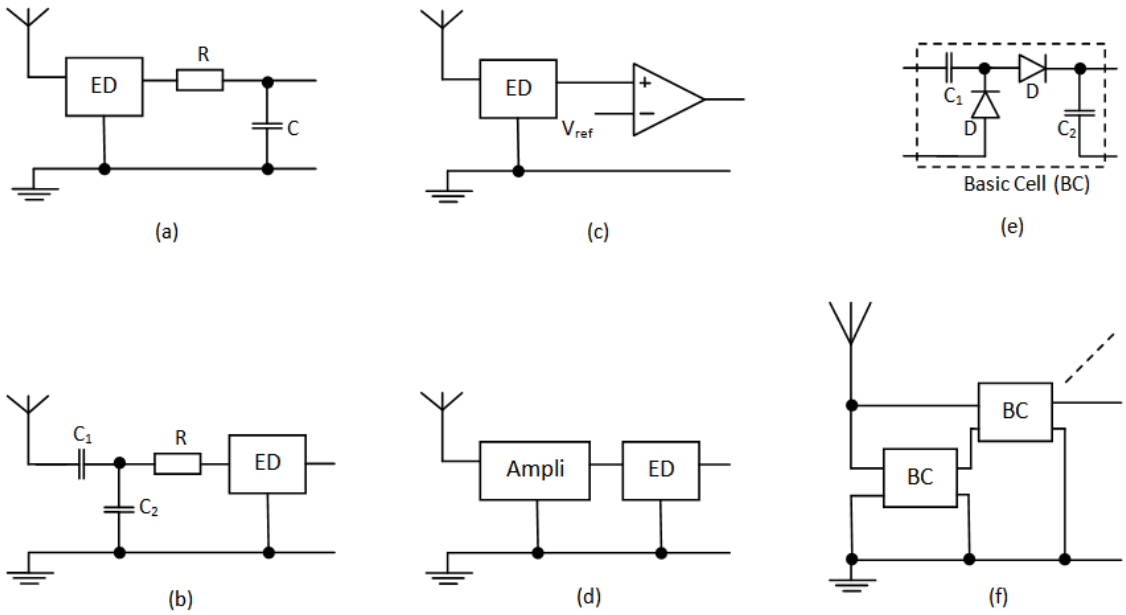


Figure 5-11. Circuits permettant l'amélioration de la sensibilité

5.3 Décodage Numérique

5.3.1 Consommation dans les réseaux de capteurs sans-fil

Les problèmes de consommation dans les WSN ont été à l'origine d'une multitude de contributions. Un des aspects revenant souvent est la consommation due à l'émetteur-récepteur radio. Pour diminuer cette consommation, plusieurs possibilités sont explorées. La plupart du temps, la technique utilisée consiste à profiter au maximum des modes de veille des émetteur-récepteurs ainsi que des contrôleurs de manière à diminuer l'activité et la consommation. Par exemple, dans [42], l'emphase est mise sur une activité cyclique adaptative en fonction des besoins et de l'utilisation du réseau. D'autres recherches, visent, en plus d'une gestion de la synchronisation des communications, à mettre en avant quels sont les choix architecturaux, algorithmiques et technologiques permettant de réduire la consommation dans les réseaux de capteurs sans-fil comme par exemple dans [43].

5.3.2 Module de réveil

Pour des applications où la latence n'est pas un facteur critique et pour lesquelles l'activité réseau est relativement faible, une alternative aux réveils périodiques est l'émission de messages de réveil, encore appelée *signalisation* dans la littérature. Ceci désigne le modèle de communication

décrit dans le Chapitre 1 comme « communication lorsque nécessaire ». De nombreux articles comparent cette approche à celle de réveils périodiques [44], alors que d'autres tentent de proposer des protocoles les combinant comme par exemple [45]. La littérature concernant les modules de réveils appelés WUR pour *Wake-Up Receiver* est abondante. On peut citer, par exemple, les contributions suivantes : dans [46] un design ainsi qu'une étude de l'efficacité d'un module de réveil sont évalués de même que dans [47], d'autres architectures plus exotiques peuvent être trouvées dans [48-52]. De manière générale, l'ensemble de ces circuits proposent des architectures synchrones. Certaines privilégient le traitement numérique pour le décodage alors que d'autres comme [50] s'appuient sur un « calcul analogique » en accumulant des niveaux de tensions. L'ensemble de ces références utilise une modulation de type OOK pour pouvoir bénéficier d'une démodulation facilitée par un détecteur d'enveloppe bien que [49] utilise tout de même un mixeur.

5.3.3 Circuit proposé

Dans ce contexte, nous proposons de concevoir un module de réveil ayant les particularités d'être conçu en logique asynchrone et sur FPGA. Avant de discuter des avantages d'un tel circuit par rapport à un traitement synchrone dans la sous-partie 5.6.2, nous présentons ici son fonctionnement ainsi que son architecture générale.

5.3.3.1 Fonctionnement

Ayant présenté la démodulation analogique du signal, on supposera dans cette partie que le signal arrivant en entrée de la partie numérique considérée a été ramené en bande de base et à un niveau logique suffisant pour être détecté. L'objectif du circuit à concevoir est alors le suivant :

Vérifier que le message reçu correspond à un message de réveil et qu'il concerne l'animat sur lequel est embarqué le module de réveil. Si tel est le cas, générer un signal d'interruption.

La reconnaissance du type de message de réveil est évaluée à partir de l'entête et du respect de l'espace entre deux trames. La vérification du nœud de destination se fait par comparaison de chaque symbole reçu avec ceux de trois adresses de références. Ces adresses peuvent être modifiées par l'utilisateur via un protocole SPI simplifié. La simplification réside dans le fait que le module conçu ne peut être utilisé que comme esclave de ce protocole et qu'il ne met pas à jour certains registres d'états. De plus, seule l'écriture est possible. Sur les trois adresses de références

deux contiennent 16 bits et la dernière en contient 64. L'utilisation suggérée est d'utiliser l'adresse de 64 bits pour l'adresse MAC du nœud porteur de ce module de réveil et une des adresses de 16 bits pour son adresse réseau. La troisième adresse a été ajoutée dans le but de permettre un réveil collectif ou de cibler certains groupes.

À chaque symbole reçu, une comparaison est effectuée avec le symbole correspondant des trois adresses de références et le résultat généré est combiné aux résultats des comparaisons des symboles antérieurs du message. Si à la fin de la réception, il y a une concordance avec au moins une des adresses de référence, alors une interruption est générée.

5.3.3.2 Architecture Générale

L'architecture générale se décompose en trois sous-parties internes notées *Front-End*, *Comparateur* et *Communication SPI*. En plus de la partie analogique de démodulation, ce circuit requiert une partie externe lui permettant de fournir une base de temps afin de retrouver l'information codée avec la modulation OOK. Cette partie peut-être constituée de ligne à délais (et par conséquent être éventuellement conçue à l'intérieur du *chip*) ou des circuits de types *RC*. L'architecture générale est alors représentée sur la Figure 5-12.

Front-End :

Ce premier bloc a deux principales fonctions : recevoir les signaux du monde extérieur et les convertir en représentation sur deux fils, ainsi que de décoder l'information temporelle contenue dans chacun des signaux d'entrée. Cela comprend notamment la gestion de la charge/décharge des lignes à délais. La Figure 5-13 résume sous-forme de logigramme le fonctionnement de la partie décodage.

Comparateur :

Ce second bloc reçoit les informations décodées du module précédent et par un jeu de pointeur compare chaque symbole reçu avec le symbole correspondant pour les trois adresses de référence en parallèle. Le résultat de ces comparaisons est alors accumulé dans des fonctions ET à mémoire de telle manière qu'à la fin de la réception la seule possibilité pour qu'au moins une des sorties de ces trois portes soit active est qu'il y ait eu une correspondance entre le message reçu et au moins une des trois adresses de référence.

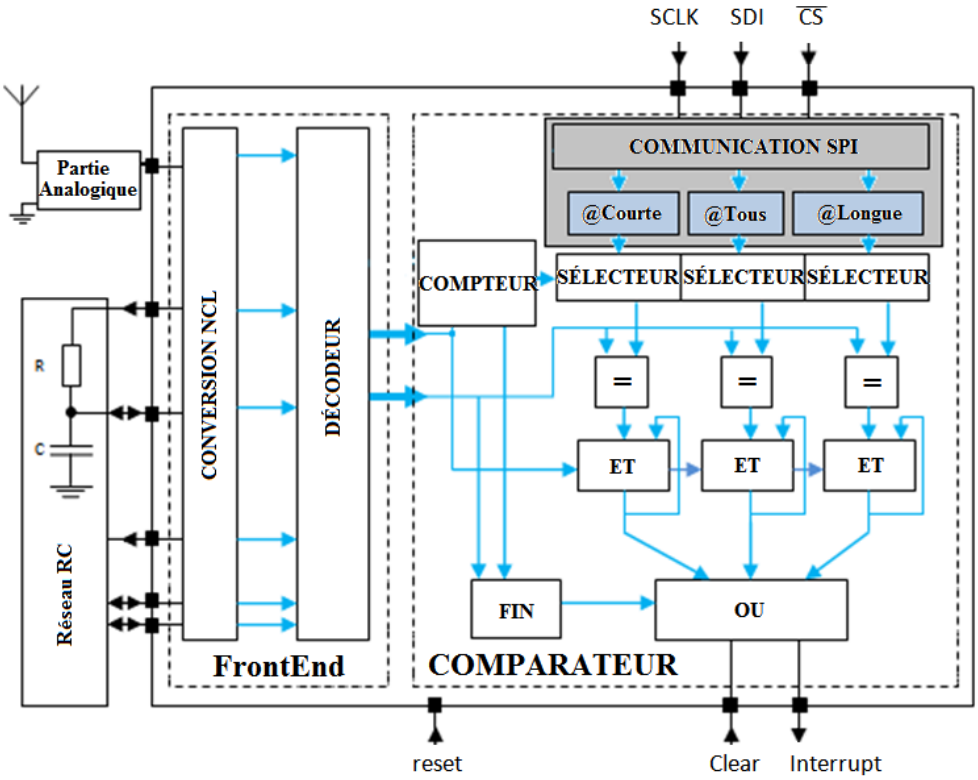


Figure 5-12. Architecture générale du module de réveil

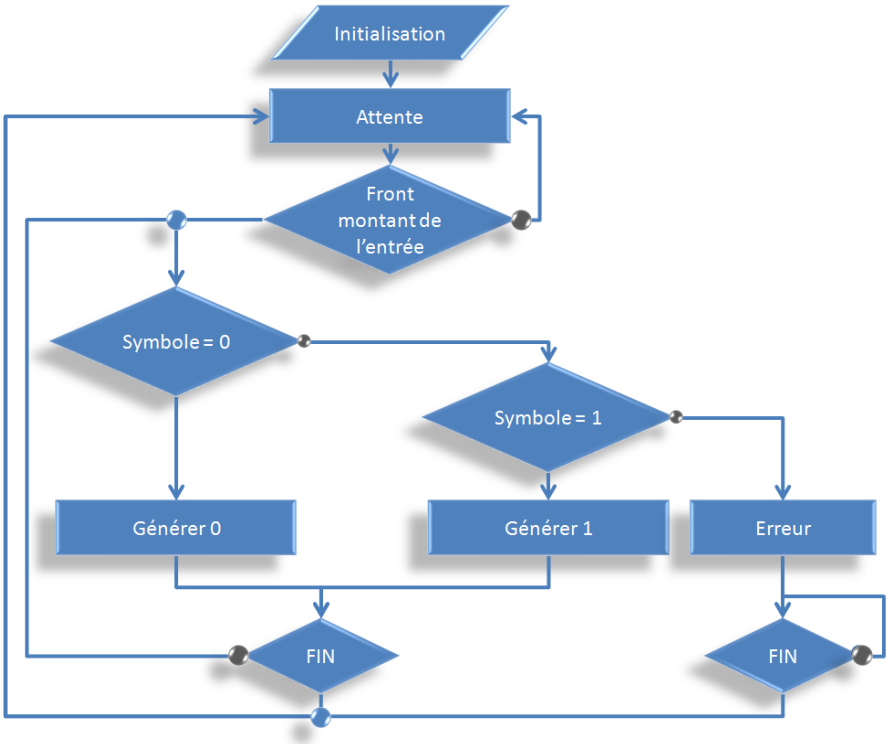


Figure 5-13. Logigramme de la partie décodage du module de réveil

On remarquera que tout comme l'adresse de 64 bits, les adresses de 16 bits sont configurables par l'utilisateur. Par conséquent, il n'est pas impossible que ces deux soient identiques conduisant ainsi à deux résultats de correspondance actifs à la fin de la réception. La porte OU récupérant les résultats de comparaison est inhibée par un signal provenant du module précédent et qui reste inactif tant que la fin du message n'a pas été reçue. Cela permet d'éviter les interruptions avant la fin de la totalité de la comparaison.

Communication SPI :

Finalement, ce dernier bloc permet à l'utilisateur, via un microcontrôleur par exemple, de configurer les trois adresses de référence. On notera que, bien que le protocole SPI requiert une horloge, celle-ci sera fournie par le microcontrôleur à l'initialisation et normalement plus jamais après. De ce fait, nous pouvons considérer en régime permanent que ce module est aussi asynchrone. Pour le reste, le fonctionnement de ce module est relativement simple. Lorsque la pin $\overline{CS}$ est désactivée le module écoute les informations provenant de l'extérieur sur sa pin SDI lorsque des fronts d'horloge sur la pin $SCLK$ sont détectés. Sur les deux premiers fronts d'horloge, les valeurs de SDI permettent de déterminer le champ à écrire comme exprimé dans le Tableau 5.2. Par ailleurs, les 8, 16 ou 64 valeurs suivantes seront placées dans des registres NCL correspondant et converties en représentation sur deux fils à la volée. On notera que tant que les trois adresses de référence et le préambule ne sont pas initialisés, le reste du circuit est placé dans un mode inactif.

Tableau 5.2. Code d'opérations du module de communication SPI

Code	Registre modifié
00	Préambule
01	Adresse réseau 16 bits
10	Adresse de groupe 16 bits
11	Adresse MAC 64 bits

5.4 Conception sur FPGA

5.4.1 Motivation du choix du FPGA et d'un FPGA en général

Pour implémenter le module de réveil, nous avons choisi d'utiliser un FPGA. Ce choix est motivé par plusieurs paramètres dont nous résumons ici les plus importants. Tout d'abord, les FPGA permettent un prototypage rapide est généralement modifiable. En effet, comparée à celle sur ASIC, la conception sur FPGA bénéficie en termes d'implémentation de la rapidité de la programmation en contraste avec les délais de fonderie de plusieurs mois. De plus, hormis les FPGA en technologie *anti-fuse*, les FPGA dans des technologies plus courantes, notamment *SRAM* et *Flash*, sont reprogrammables. Cette propriété confère une plus grande souplesse au projet dans le sens où des modifications postérieures peuvent être apportées à volonté. D'autre part, pour des applications à faible volume comme dans notre cas, en termes de coût de fabrication les FPGA sont très largement moins onéreux comparativement aux larges NRE d'une conception sur ASIC ainsi qu'aux coûts propres de fabrication.

En contrepartie, l'utilisation de FPGA ne permet pas toujours de bénéficier des améliorations apportées en termes de ressources utilisées notamment. En effet, le nombre d'éléments dans un FPGA est fixe indépendamment de l'utilisation de ces derniers. Par conséquent, une diminution des ressources qui s'accompagnerait dans un environnement ASIC d'une diminution de la consommation statique, ne serait perçue dans un environnement FPGA que si celle-ci permet de changer de FPGA. Le Tableau 5.3, montre l'évolution des pertes statiques en fonction du nombre d'éléments de plusieurs FPGA de la famille Igloo d'Actel.

Le FPGA que nous avons alors choisi pour implémenter le module de réveil présenté précédemment est l'Igloo Nano *AGLN250V2* d'Actel. Le Tableau 5.4 résume ses principales caractéristiques.

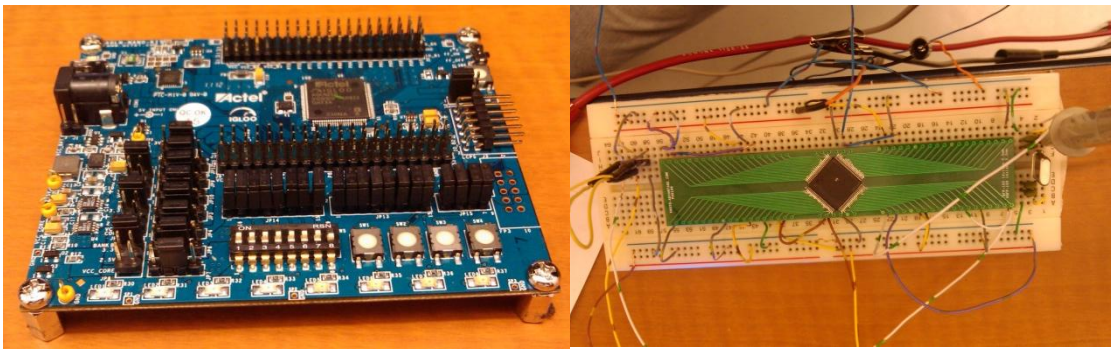
Le choix de ce FPGA a été motivé par une des plus faibles consommations du marché et un ensemble de portes suffisant pour pouvoir supporter le design proposé. Par ailleurs, celui-ci équipe les plateformes de développement *Actel Igloo Nano Starter Kit*. Nous avons alors pu tester rapidement certaines caractéristiques sur cette plateforme avant de concevoir un prototype sur *breadboard* afin de maîtriser exactement les consommations mesurées. La Figure 5-14 représente les deux environnements de prototypage discuté.

Tableau 5.3. Consommation de différents FPGA de la famille Igloo d'Actel

FPGA	Consommation statique (µW) (avec Flash*Freeze Mode)	Nombre de portes
AGLN010	2	10000
AGLN015	4	15000
AGLN020	4	20000
AGLN030Z	5	30000
AGLN060	10	60000
AGLN125	16	125000
AGLN250	24	250000

Tableau 5.4. Principales caractéristiques du FPGA choisi

Nombre de portes	Consommation statique (µW)	D-flip-flops	RAM (kbits)	ROM (bits)	Banques I/O	Nombres de GPIO	Package choisi
AGLN010	30	6144	36	1024	4	68	VQ100



(a) (b)

Figure 5-14. Environnement de développement du module de réveil: (a) Igloo Nano Starter Kit, (b) Bread-Board

5.4.2 Caractéristiques et éléments de base

L'analyse détaillée du FPGA sélectionné n'est pas réellement pertinente pour la suite. En revanche, il est important de signaler que l'*AGLN250V2* est un FPGA dont les cellules sont réalisées avec des transistors *Flash* ce qui permet au circuit de réveil de ne pas être alimenté tout

en gardant en mémoire le design implémenté. Cette propriété est souvent utile pour les designs pour lesquels la préservation de la propriété intellectuelle est importante en ce sens qu'il n'est pas nécessaire que le design soit stocké dans une mémoire externe non-volatile pour configurer le FPGA à chaque mise sous-tension. La protection de la propriété intellectuelle n'est pas une des caractéristiques importantes de ce projet académique. Néanmoins, compte tenu que l'objectif du système *animat* est de minimiser la consommation d'énergie totale, la technologie *Flash* procure la possibilité de couper l'alimentation du module de réveil tout en préservant le design pour un futur redémarrage.

De plus, il est important de noter que chaque élément de base de l'architecture de ce FPGA peut être configuré soit comme une LUT à trois entrées et une sortie, soit comme une bascule ou soit comme un loquet. La possibilité de pouvoir implémenter des loquets est primordiale pour pouvoir réaliser des circuits asynchrones. D'autre part, pour pouvoir profiter pleinement des 27 portes décrites par le NCL, l'utilisation de LUT à quatre entrées aurait été préférable. Néanmoins ceci aurait requis un FPGA différent consommant beaucoup plus. La contrainte de consommation ayant une priorité supérieure à celle du nombre de ressources utilisées, nous avons tout de même choisi d'utiliser ce FPGA.

5.4.3 Design Asynchrone sur FPGA

Avant de pouvoir présenter les résultats de l'implémentation sur FPGA du module de réveil, il semble important de se demander si la réalisation de circuit asynchrone sur FPGA est possible. Notamment, nous avons vu que le NCL a été proposé dans l'optique d'implémentation ASIC avec une bibliothèque de 27 portes spécifiques. Est-il alors possible de former ces portes en utilisant des cellules standard tout en espérant obtenir un fonctionnement correct? Peut-on toujours faire l'hypothèse de fourches isochrones?

Pour tenter de répondre à ces questionnements, nous avons tout d'abord tenté d'évaluer le pire délai de propagation d'un signal dans un fil. Pour cela, nous avons supposé que ce cas adviendrait lorsque le fil aurait une longueur égale à celle de la diagonale du FPGA. Une implémentation d'un circuit de test comme représenté sur la Figure 5-15, nous a permis de quantifier ce délai dans les conditions normales de fonctionnement. Les résultats on été obtenus par simulation pour deux tensions d'alimentation différentes. Les relevés sont résumés dans le Tableau 5.5 alors que la Figure 5-16 représente le type de chronogrammes observé.

Tableau 5.5. Relevés des délais maximaux

V_{core}	Délai Diagonal
1.5 V	~ 4.2 ns
1.2V	~ 6.2 ns

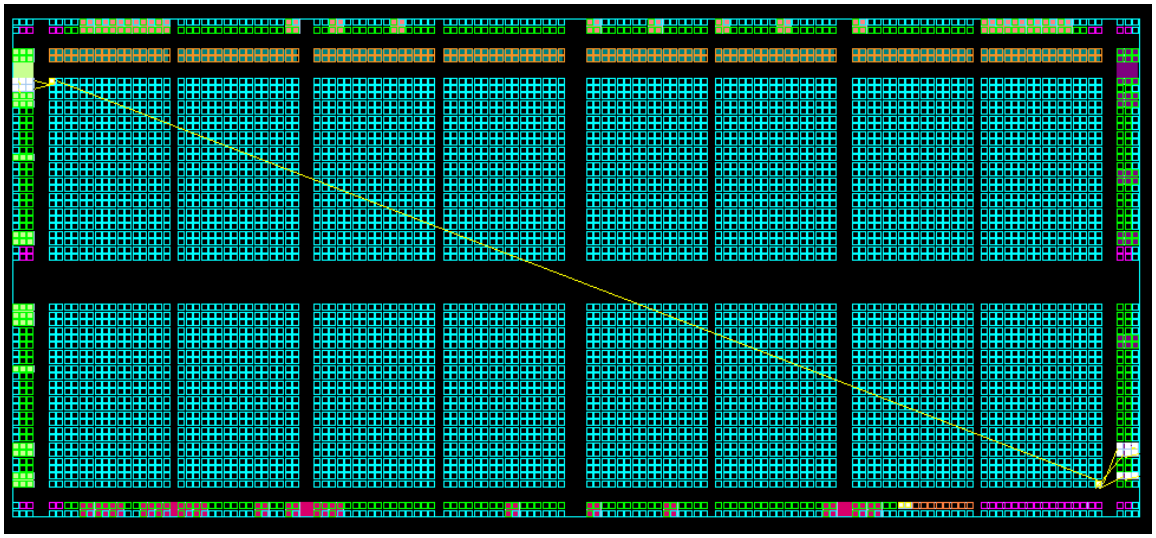


Figure 5-15. Configuration pour la mesure du plus grand délai

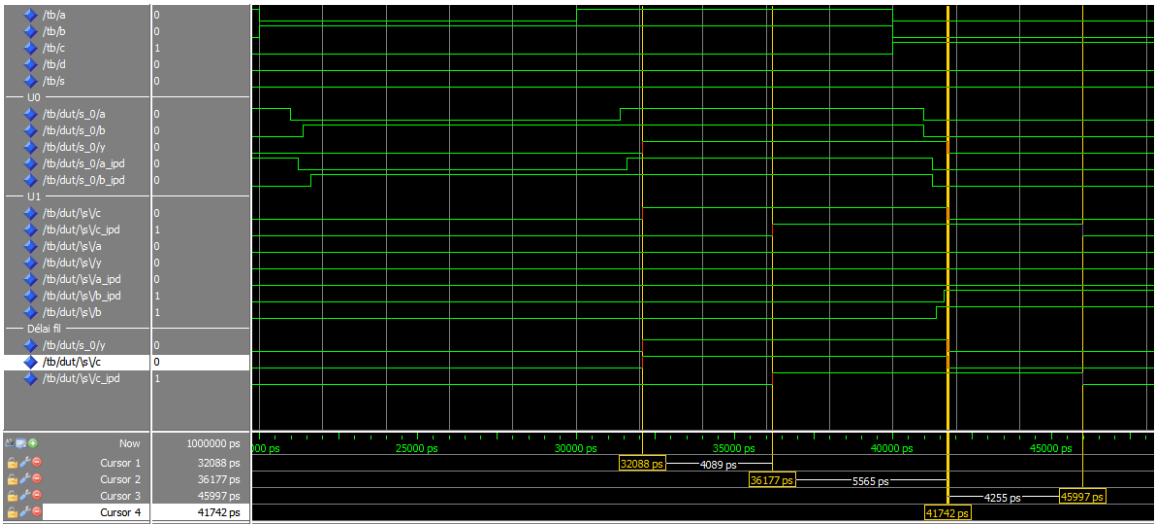


Figure 5-16. Chronogramme permettant la mesure du plus grand délai à 1.5V

En considérant les délais dans les éléments de base du FPGA rappelés dans l'ANNEXE 5, on justifie l'affirmation selon laquelle les délais dans les fils ne peuvent pas être considérés

négligeables devant ceux dans la logique. Ceci permet donc d'éviter une mauvaise définition de l'hypothèse de fourches isochrones trop souvent rencontrée dans la littérature.

Comme nous l'avons déjà vu plus haut, la condition d'observabilité empêche les cas où une somme de délais puisse avoir un impact sur l'intégrité des données. Par conséquent, le seul cas qui devrait être considéré pour s'assurer que l'hypothèse de fourches isochrones est respectée est celui d'un délai avant un niveau de logique bloquant comme par exemple une porte ET. Dans ce cas, en reprenant l'exemple de la Figure 4-7b et en supposant que d_2 est égal au délai maximal exprimé plus tôt, essayons de montrer que l'hypothèse est respectée.

Supposons le cas où il existe un délai très long sur une branche orpheline, et que sur les autres branches il n'y ait ni porte ni délai dans les fils. On sait que la règle de complétude empêche un tel cas, néanmoins cela nous permet d'estimer la pire configuration. Appelons t_f et t_p respectivement les délais dans le fil et la porte de la branche concernée. D'après l'hypothèse, il faut donc que cette information soit « résolue » avant qu'un nouvel ensemble de données ne puisse arriver. En notant t_e le temps nécessaire pour que le signal *e* soit généré puis propagé et t_{Ki} le temps nécessaire pour que le signal K_i soit généré puis pris en compte, on peut traduire l'hypothèse de fourches isochroniques par l'inéquation suivante :

$$t_f + t_p < t_\varepsilon + t_{ack} \qquad (1)$$

Or, minimalement, pour que *e* soit pris en compte, il faut que celui-ci soit généré, qu'il soit propagé puis pris en compte. Dans le cas minimal, la sortie du registre NCL comporte un seul signal (codé sur 2 fils). Le signal *e* doit donc passer par une porte NOR2 (génération), par un fil (transport) et par un loquet SR2. Pour que le signal K_i soit pris en compte, il faut que les données franchissent le registre de sortie (mémorisation), que le signal K_i soit généré grâce à une porte NOR2 (complétion), puis que celui-ci soit acheminé vers le registre d'entrée (retour) et finalement qu'il soit enregistré dans le loquet SR1. En supposant que les délais dans les fils sont négligeables sauf pour la branche considérée, on a alors les égalités suivantes :

$$t_{e,min} = t_{NOR2} + t_{transport} + t_{SR2} \approx t_{NOR2} + t_{SR2} \qquad (2)$$

$$t_{Ki,\,min} = t_{mém} + t_{OR2} + t_{retour} + t_{SR1} \approx t_{mém} + t_{NOR2} + t_{SR1}$$

Par conséquent, on obtient que l'inégalité suivante doit être vérifiée :

$$t_f + t_p < t_{NOR2} + t_{SR2} + t_{mém} + t_{OR2} + t_{SR1} \qquad (3)$$

$$\Leftrightarrow t_f < t_{NOR2} + t_{SR2} + t_{mém} + t_{OR2} + t_{SR1} - t_p$$

Pour les différentes valeurs de la tension V_{core}, l'application numérique partielle donne :

1.5V : $t_{mém} \approx 3.07$ ns (mesuré sur waveform)

$t_f < 1.10 + 0.62 + 3.07 + 1.19 + 0.6 - t_p \approx 6.58 - t_p$

1.2V : $t_{mém} \approx 5.26$ ns (mesuré sur waveform)

$t_f < 2.07 + 0.89 + 5.26 + 2.3 + 0.87 - t_p \approx 11.39 - t_p$

Si pour t_p on choisit de prendre le plus grand délai dans une porte, trouvé pour une porte XOR à trois entrées, on obtient dans le cas le plus défavorable que les délais dans les fils doivent vérifier les inéquations suivantes. On notera que le choix d'une porte XOR nous permet d'obtenir un cas pire que celui permis par l'utilisation de la logique positive.

À 1.5V : **$t_f < 4.79$ ns**

À 1.2V : **$t_f < 8.27$ ns**

En reprenant les valeurs mentionnées dans le Tableau 5.5, on montre que l'hypothèse de fourches isochrones est bien respectée. On remarquera toutefois que l'étude réalisée a été faite pour un placement peu probable et en négligeant certains délais notamment dans les autres fils que la branche orpheline. Par conséquent, dans un cas réel, les conditions seraient plus largement vérifiées.

Par ailleurs, dans le Chapitre 4, nous soulevions qu'un autre type de problème de délais pouvait survenir dans les circuits de type SHF-NCL et qui n'apparait pas pour les circuits NCL. Ce cas particulier était mentionné sur la Figure 4-7a. Considérons alors, comment un tel cas pourrait survenir dans un FPGA. On peut alors décliner le problème selon trois cas : la porte du registre de sortie générant le signal retardé est éloignée des autres portes de l'ensemble considéré (comme sur la Figure 5-17), les portes recevant le signal retardé sont éloignées du reste ou, enfin, l'ensemble des portes est compact mais seul le routage du signal considéré est excessivement grand. La Figure 5-17 illustre le premier cas avec la convention suivante : en vert sont représentées les portes utilisées pour le registre d'entrée, en orange celles pour la logique et en bleu celles pour le registre de sortie. Les fils ne sont pas tous représentés et l'ensemble reste schématique. Néanmoins, les fils rouges sont les fils sur lesquels un important délai de

propagation est induit par le placement. On observe alors dans ce cas spécifique que le signal *e* est lui aussi retardé préservant ainsi l'intégrité des données. Pour le second cas le signal *e* sera en avance mais de par l'éloignement des portes recevant le signal retardé, les sorties seront retardées et de même l'intégrité des données sera préservées. Enfin, le seul cas *a priori* problématique est celui d'un très mauvais routage. Cependant, l'analyse des délais précédente montre que même avec un routeur exécrable, l'intégrité des données est préservée.

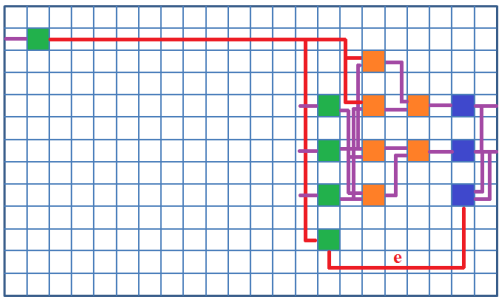


Figure 5-17. Délais sur un fil dans un FPGA

5.4.4 Placement / Routage et Implémentation

Après avoir codé notre design, nous avons suivi le flot de conception standard : synthèse placement-routage, génération des fichiers avec délais back-annotés, génération des fichiers de configurations et implémentation sur puce. Pour chacune de ces étapes, des phases de vérifications ont été réalisées. Le Tableau 5.6 résume les principales caractéristiques du design en respectant le protocole NCL. Une discussion sur les avantages d'utiliser le SHF-NCL sera faite plus bas dans ce chapitre.

Tableau 5.6. Caractéristique de l'implémentation du module de réveil

	Description
FPGA	IGLOO AGL250V2 1.2V Package: 100 VQFP
Portes	4275 (69.58%): Combinatoire = 2728, Séquentiel = 1547
I/O	15 (22.06%): Entrées = 6, Sorties = 4, Bidirectionnel = 5

5.4.5 Consommation

Afin de pouvoir mesurer l'apport d'un tel circuit, nous avons évalué sa consommation aussi bien statique que dynamique. Pour ce faire, nous avons tout d'abord utilisé le logiciel Smart Power fourni avec l'IDE, Libero, d'Actel qui permet à partir de configurations d'activités internes d'évaluer la puissance consommée. Le démarche fut la suivante : lors de la vérification sous ModelSim des fichiers de type *.vcd* ont été créés. Ces fichiers enregistrent l'activité interne de chaque *nets* dans le circuit pour les stimuli utilisés. L'application Smart Power utilise alors ces informations concernant l'activité du FPGA, les combine avec les données de placement routage puis évalue la consommation statique et dynamique du design. Ces résultats sont présentés dans le Tableau 5.7 pour différentes fréquences, que ce soit celle pour le SPI ou celle du signal de réveil.

Tableau 5.7. Puissance dissipée par le module de réveil

Mode	**Fréquence ou taux de transition**	**Puissances obtenues avec Smart Power À 25°C pour V_{dd} = 1.2V exprimées en µW**		
		Puissance Statique	**Puissance Dynamique**	**Total**
Configuration	32 kHz	29.8	3.5	**33.3**
	1 MHz		90.0	**119.7**
	2 MHz		179.2	**208.9**
Normal	22 kHz		22.5	**52.3**
	724 kHz		776.9	**806.7**

On constate que la consommation dynamique est relativement élevée. Néanmoins, celle-ci n'est effective que si le module de réveil est constamment utilisé. Conformément à ce que nous mentionnions lors de l'étude des modèles d'énergie, un tel circuit trouve son utilité lorsque la fréquence d'occurrence des messages de réveil, notée f_{WU}, est faible. La Figure 5-18 représente alors la consommation totale du circuit lorsque la fréquence d'occurrence des messages de réveil diminue. Les résultats présentés utilisent les valeurs du Tableau 5.7.

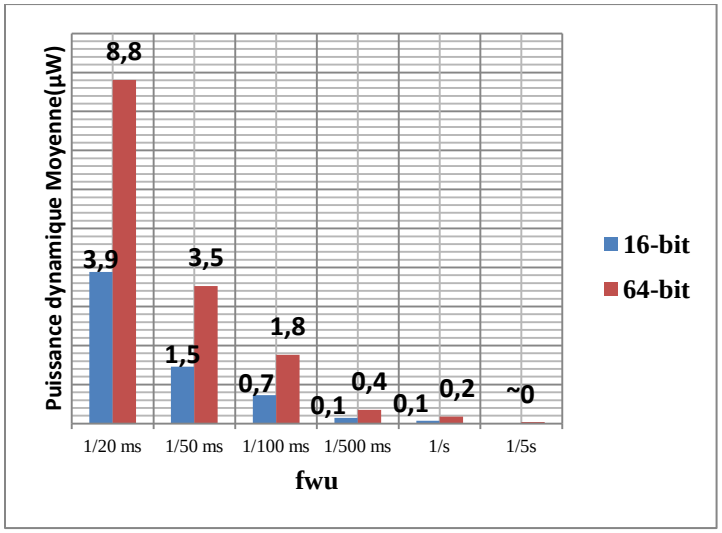


Figure 5-18. Évolution de la puissance dynamique en fonction de la fréquence d'occurrence des messages de réveil

On constate alors que très rapidement, pour de faibles fréquences d'occurrence des messages de réveil, la consommation totale est dominée par la consommation statique. Pour diminuer cette consommation statique, une des possibilités est de changer de FPGA. Compte tenu de la taille de notre design, une telle modification n'est pas permise. Nous avons alors cherché à diminuer la tension d'alimentation du FPGA de manière à faire chuter cette consommation. Les résultats sont résumés sur la Figure 5-19. On constate qu'avec une tension d'alimentation de 850mV la consommation statique chute à environ 5µW. Ce résultat est d'autant plus remarquable que le design le moins énergivore pouvant être trouvé dans la littérature [50] utilisant un circuit dédié affiche une consommation de l'ordre de 2.4µW. On notera tout de même que certains des résultats présentés dans cet article sont discutables. Notamment, il semblerait que seulement la consommation nécessaire à fournir l'horloge au circuit dépasse les résultats annoncés.

On constate alors ici que le principal avantage d'un module de réveil en logique asynchrone se résume au fait que lorsqu'il y a de l'activité sur les entrées, alors le circuit fonctionne et consomme de l'énergie. Dès que l'activité sur les entrées s'interrompt le module est dans un mode veille inhérent aux circuits asynchrones. C'est cette propriété qui nous permet de considérer que la puissance totale tend à être dominée par la puissance statique pour des fréquences d'occurrence des messages de réveil faibles.

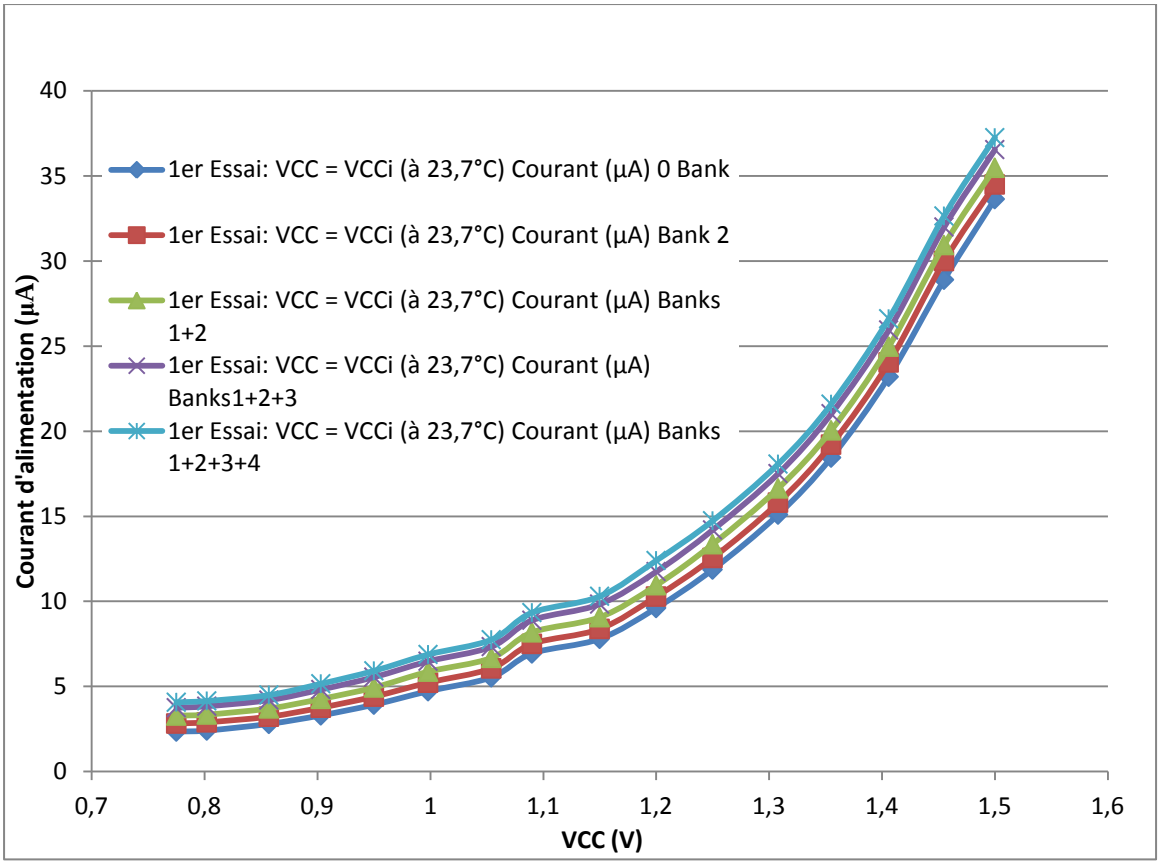


Figure 5-19. Évolution du courant statique en fonction de la tension d'alimentation

5.5 Alimentation Solaire

La consommation totale mesurée étant relativement faible, on notera qu'il est envisageable d'utiliser un module d'alimentation solaire pour fournir l'énergie nécessaire à ce circuit. Dans le Chapitre 2, nous avons détaillé un circuit d'alimentation solaire utilisé pour permettre la recharge de batterie Li-Ion. Pour ce module, la consommation d'énergie est généralement très faible et ne requiert une consommation plus élevée que sur de faibles laps de temps survenant de manière peu fréquente. Pour ce type de situation, l'utilisation d'un super-condensateur semble toute désignée. De ce fait, le module de réveil serait totalement autonome en termes d'énergie. D'autre part, ce type de design permettrait sur le plan théorique des *animats* une utilisation des moyens de communication seulement lorsque l'énergie récupérée pour ceux-ci est suffisante. Par conséquent, un *animat* désireux de communiquer souvent devrait, par exemple, apprendre à se déplacer de telle manière que sa recharge solaire le lui permette.

5.6 Discussion

Dans cette partie, nous revenons sur deux aspects que nous pouvons désormais discuter en utilisant l'exemple du module de réveil. Tout d'abord, nous nous servirons de cet exemple pour évaluer le gain apporté par la modification du protocole asynchrone sur un système complet. Dans un second temps, nous considérerons un circuit synchrone ayant les mêmes fonctionnalités que celui présenté dans ce chapitre et les compareront en termes de consommation.

5.6.1 Effets du SHF-NCL

Nous comparons donc ici les ressources utilisées pour le même circuit de réveil que celui présenté précédemment, mais refaisons le design en respectant le protocole SHF-NCL. Le Tableau 5.8 résume l'utilisation des ressources pour les deux méthodes.

Tableau 5.8. Comparaison des ressources et des consommations du NCL et du SHF-NCL pour le design de module de réveil sur Actel Iglo AGLN250V2 à 1.2V pour un taux de transition du signal d'entrée de 724 kHz

	NCL	**SHF-NCL**
Ressources Combinatoires	2728	2579
Ressources Séquentielles	1547	1334
Total des Ressources	4275	3913
Consommation Statique	29.8 µW	
Consommation Dynamique	776,2 µW	352,7 µW

On constate donc que sur cet exemple la réduction de complexité est de l'ordre de 10% alors que la consommation semble être réduite de moitié. Rappelons tout de même que ces résultats ont été obtenus à l'aide de l'outil de simulation Smart Power et que pour être exhaustive, l'analyse devrait être complétée par une implémentation physique accompagnée de relevés. Toutefois, contrairement au design ASIC les simulations temporelles et énergétiques sur FPGA sont très proches de ce que l'on peut observer physiquement car l'architecture est parfaitement connue. Par conséquent, nous pouvons conclure quant à l'utilité de l'approche SHF-NCL sur cet exemple

en disant que celle-ci permet aussi bien une réduction de la complexité qu'une réduction de la consommation.

5.6.2 Comparaison avec un circuit synchrone

Enfin, afin de clore ce paragraphe, nous avons décidé de comparer la solution proposée avec un équivalent synchrone. L'idée étant de préserver le plus possible l'architecture de comparaison par rapport à trois adresses de référence à chaque nouveau symbole reçu. La principale différence réside dans le fait qu'il n'est pas nécessaire d'utiliser une stratégie particulière pour retrouver une base de temps compte tenu de l'utilisation d'une horloge interne. Par ailleurs, pour différencier les états hauts des états bas, on supposera que la fréquence d'échantillonnage permet d'acquérir dix points sur la plus petite période $T = 45.7\mu s$ du signal d'entrée. Autrement dit la fréquence d'horloge du système est donnée par :

$$f_{hor} = \frac{10}{T} \approx 219\ kHz$$

Pour le respect des conditions temporelles des états haut et bas on utilisera des compteurs et une machine à états similaires à celle présentée pour le module de réveil. Sur le Tableau 5.9, nous présentons les principales caractéristiques de ce circuit notamment en termes de ressources utilisées mais aussi en termes de consommation.

Tableau 5.9. Comparaison NCL, SHF-NCL et Logique Synchrone pour un taux de transition du signal d'entrée d'environ 22 kHz et une tension d'alimentation de 1.2V

	NCL	**SHF-NCL**	**Synchrone**
Ressources Combinatoires	2728	2579	1048
Ressources Séquentielles	1547	1334	263
Total des Ressources	4275	3913	1311
Consommation Statique	29.8 µW		27.7 µW
Consommation Dynamique	22.5 µW	11.4 µW	223.7 µW

Ces résultats mettent en avant que, comme nous l'avions prévu, l'utilisation de logique synchrone utilise beaucoup moins de ressources. On mesure dans l'exemple du module de réveil proposé

une diminution d'environ 70% par rapport au NCL standard et de l'ordre de 66% par rapport au SHF-NCL. En décomposant l'utilisation des ressources pour la partie combinatoire et séquentielle, on observe que pour la partie combinatoire la réduction est de l'ordre de 60%. Ceci s'explique principalement par la non nécessité d'utiliser une représentation complète comme celle « sur deux fils » utilisée pour le design. Au niveau séquentiel, l'utilisation des ressources est divisée par un facteur quatre environ. En effet, la représentation booléenne utilisée induit une diminution d'un facteur deux du nombre de signaux à stocker dans des éléments de mémoire. Le second facteur deux est quant à lui propre à l'architecture du FPGA choisi qui utilise une porte de base pour implémenter une bascule alors qu'un loquet en utilise deux.

En analysant la consommation énergétique, on constate que la consommation statique a changé alors que le FPGA utilisé est resté inchangé. Ceci s'explique par une diminution des entrées/sorties et par conséquent, la possibilité de diminuer le nombre de banque d'I/O alimentées. D'autre part, concernant la consommation dynamique, pour avoir une précision suffisante sur le temps des états hauts nous avons arbitrairement choisi d'imposer une fréquence dix fois plus grande que celle du signal d'entrée. À une telle fréquence, on constate que la puissance dynamique consommée par le circuit synchrone est dix fois supérieure à celle d'un circuit conçu en logique NCL standard et presque vingt fois plus que celle d'un circuit SHF-NCL. En effet, bien que le nombre de ressources pour une conception synchrone soit inférieur en comparaison d'un circuit asynchrone, le signal d'horloge doit se propager dans tout l'arbre de distribution impliquant de ce fait une consommation dynamique importante. D'autre part, pour avoir une précision relativement bonne sur le temps des états haut et bas, nous avons choisi une fréquence d'horloge dix fois supérieure à celle du signal. Compte tenu que la consommation dynamique de tels circuits est proportionnelle à la fréquence d'horloge, il serait toujours possible de rétorquer que le choix cette fréquence *a priori* arbitraire vient appuyer le résultat que l'on désire mettre en évidence. Pour répondre à cela, considérons le cas limite où la fréquence d'horloge est la même que celle du signal d'entrée. Dans ce cas, les consommations dynamiques seraient du même ordre. Néanmoins, comme le montre la Figure 5-18 et la discussion qui l'accompagne la consommation d'un circuit en logique asynchrone est dominée par la consommation statique dès lors que l'activité est faible. Par ailleurs, on notera que l'utilisation d'une horloge interne et les problèmes de déphasage d'horloge associés empêchent de diminuer la tension d'alimentation de la même façon que pour un circuit asynchrone.

Par conséquent, au vu des résultats et des remarques qui les accompagnent, il est possible de conclure sur l'efficacité sur le plan énergétique d'utiliser de la logique asynchrone pour concevoir un module de réveil de la radio principale. D'autre part, ces remarques peuvent ouvrir un nouvel axe de recherche pour la suite du projet, en notant qu'il serait alors intéressant de concevoir d'autres modules en utilisant une telle logique, notamment un gestionnaire de l'utilisation des modules de l'*animat* en fonction de l'énergie disponible ou de l'activité des entrées.

CHAPITRE 6 EXPÉRIMENTATIONS

Après avoir mis en place l'ensemble des caractéristiques principales d'un *animat* et avoir présenté son architecture principale ainsi que les détails de réalisation, nous présentons brièvement ici quelques aspects pratiques d'utilisation de la plateforme.

6.1 Environnement Graphique (GUI) et utilisation

Pour faciliter l'utilisation de la plateforme, nous avons conçu un environnement graphique qui permet d'interagir plus facilement avec les *animats*. Le but d'un tel outil est de pouvoir aisément connaître les propriétés de la colonie, de recevoir les données des *animats* sélectionnés ainsi que de pouvoir prendre le contrôle de ces derniers. La première de ces trois tâches est à la fois utile et nécessaire pour prendre en note que les *animats* ont bien rejoint le réseau, mais permet aussi de savoir d'où proviennent les informations reçues. La partie de visualisation du réseau est représentée sur la Figure 6-1 et représente une sous-partie de l'interface totale détaillée dans l'ANNEXE 10. On notera que pour les développements futurs, il serait intéressant de représenter le réseau non plus sous forme d'arbre mais suivant un graphe qui permettrait de mettre en évidence les relations entre les nœuds.

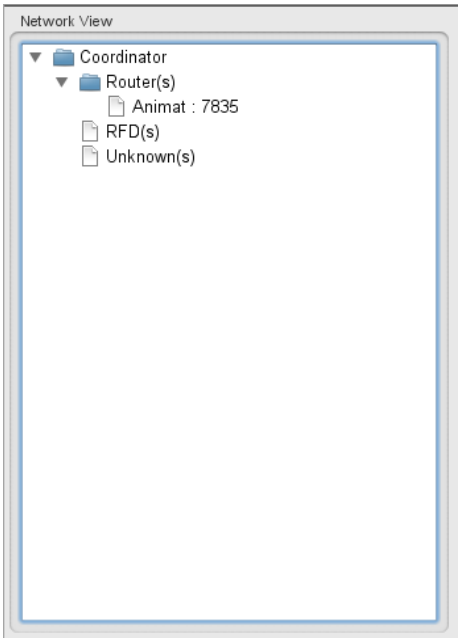


Figure 6-1. Vue du réseau

Les deux autres tâches de cette interface sont essentielles pour l'utilisation future de la plateforme. En effet, pour pouvoir appliquer des algorithmes d'apprentissage, l'utilisateur doit pouvoir être capable de récupérer les données et d'agir en conséquence. Pour ce faire, trois panneaux notés « Manual Communication », « Commands » et « Matlab » sont accessibles par l'utilisateur. Ceux-ci permettent respectivement d'effectuer des communications manuelles avec les nœuds du réseau mais aussi avec le dispositif USB, de contrôler les déplacements de l'*animat* et, dans une application future, l'ensemble de ses actionneurs, et finalement d'utiliser un code écrit sous Matlab pour contrôler l'*animat*. Cette dernière propriété permettra de profiter de la

puissance de ce calcul de la machine hôte pour pouvoir réaliser des calculs poussés de manière délocalisée. Le détail de ces différents panneaux est lui aussi fourni dans l'ANNEXE 10.

Par ailleurs, lorsqu'un *animat* est sélectionné, après que celui-ci ait effectué une phase d'initialisation avec le coordinateur du réseau, un panneau d'information devient disponible. Sur celui-ci, en plus des adresses courte et longue, l'utilisateur peut voir l'ensemble des capteurs, leurs valeurs et leurs états. De plus, il est possible de désactiver ou d'activer ces capteurs depuis l'interface. Concrètement, cette action envoie un message radio à l'*animat* concerné qui (dés)active l'échantillonnage des capteurs ciblés et qui, conséquemment, (n')émet (plus) les valeurs concernées lors de l'émission de rapport de capteurs. Un exemple de cette fenêtre est représenté sur la Figure 6-2. Il est intéressant de noter que c'est l'*animat* lui-même qui envoie la liste et la description de ses capteurs permettant ainsi à l'application de s'adapter à différents types d'*animats*.

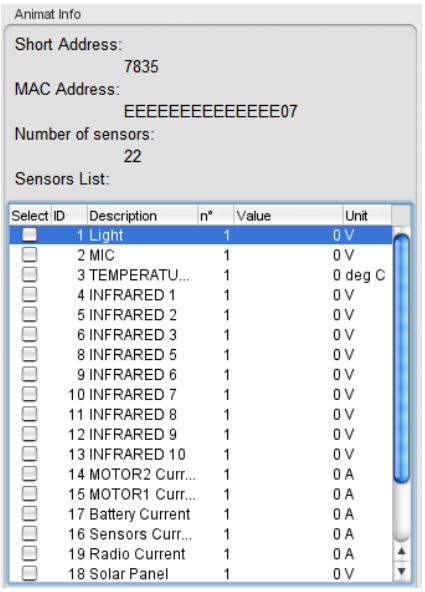


Figure 6-2. Panneau d'affichage des informations

Enfin, on notera qu'une première fenêtre permet la connexion au périphérique USB et réapparaît si celui-ci est déconnecté pendant l'application. Une vue de cette fenêtre est présenté dans l'ANNEXE 10.

6.2 Caractéristiques de l'animat

Dans cette partie, nous donnons une présentation détaillée des principales caractéristiques de l'*animat* conçu que ce soit en termes de dimensions ou de masse, ou encore en termes de consommation énergétique. Cette dernière information nous permettra de corroborer ou d'infirmer les estimations faites précédemment. Toutes les données de cette sous-partie ont été mesurées sur différents *animats*, celles-ci sont résumées dans le Tableau 6.1. Elles nous permettent d'évaluer à nouveau l'autonomie d'un tel *animat* en reprenant les différents cas du Chapitre 2. Les résultats sont alors représentés dans le Tableau 6.2 en gardant les valeurs estimées pour les modes de veilles car celles-ci n'ont pas été mesurées mais en les majorant des 5 mA consommés par la logique externe.

Tableau 6.1. Consommations mesurées sur l'*animat*

Ensemble	Courant (mA)
Microcontrôleur	62
Radio	20
Moteur-Réducteurs (pour les deux)	36
Ensemble des capteurs	4
Batterie (sans moteur)	91
Batterie (avec moteurs)	127

Tableau 6.2. Estimations de l'autonomie à partir des mesures réelles de consommations

Composant	Taux d'utilisation (%)					
	Cas 1	Cas 2	Cas 3	Cas 4	Cas 5	Cas 6 (WSN)
Microcontrôleur	100	100	100	75	10	1
Radio	100	100	20	20	10	1
Moteurs	100	75	75	50	10	0
Ensemble des capteurs	100	100	50	20	10	10
Autonomie	**7h 52min**	**8h 28min**	**10h 00min**	**13h 27min**	**2j 10h 6min**	**6j 16h 23min**

On notera que l'autonomie est alors meilleure lorsque l'*animat* est actif et en contrepartie, lorsqu'il est inactif, la consommation constante due aux composants externes qui jonchent le PCB, pénalise l'autonomie en mode d'hibernation. Pour améliorer cette dernière, il serait intéressant de cibler les composants responsables de cette consommation (par exemple le LDO) et d'opter pour des techniques de réduction de la consommation.

Enfin, le Tableau 6.3 résume les principales caractéristiques physiques de l'*animat* telles que mesurées sur les prototypes. Ces données seront réutilisées à des fins de comparaison.

Tableau 6.3. Caractéristiques physiques

Caractéristique	Valeur(s)
Dimensions (sans coque)	65mm x 70mm x 50 mm
Dimensions (avec coque)	80mm x 75mm x 65mm
Vitesse	12cm/s
Masse de la batterie	21g
Masse de la plateforme	42g
Masse socle moteurs + roues	38g
Masse de l'*animat* sans coque	101g
Masse de l'*animat* avec coque	191g
Masse moteurs et roues	33g
Volume de la partie haute de la coque	$21.3cm^3$
Volume de la partie basse de la coque	$49.3cm^3$

On notera la mention d'une coque permettant de protéger l'ensemble de l'*animat*. Celle-ci a été modélisée avec le logiciel de CAO SolidWorks et a été réalisée avec une imprimante 3D. Des photos et autres vues de cette coque sont disponibles en ANNEXE 8. On notera par ailleurs que le matériau utilisé par l'imprimante 3D disponible a une masse volumique relativement élevée et que l'utilisation d'un plastique permettrait de diminuer la masse totale de la coque et d'améliorer ses propriétés mécaniques, notamment sa résistance.

CHAPITRE 7 DISCUSSION

L'ensemble des chapitres précédents nous a permis de mettre en place ce que devrait être l'architecture générale d'un *animat* physique dans le but de répondre aux contraintes applicatives propres au domaine de recherche théorique sur le sujet. Avant de conclure, nous jugeons intéressant de discuter les atouts et faiblesses de la plateforme notamment par rapport à celles mentionnées dans le premier chapitre mais aussi par rapport aux objectifs fixés. Dans un second temps, nous ouvrirons la discussion avec des pistes susceptibles d'améliorer la plateforme.

Avant d'aller plus loin dans la discussion, nous rappelons ci-dessous, pour des raisons de lisibilité, les principaux objectifs que nous nous étions fixés :

- Faible coût de revient (~100$)
- Plateforme modulaire laissant un vaste choix de personnalisation
- Prise en compte des problèmes énergétiques
- Ayant une grande autonomie
- Facile à prendre en main

Le faible coût de revient est nécessaire pour permettre de concevoir de vastes colonies d'*animats* sans avoir à payer des sommes astronomiques. Actuellement, la solution proposée coûte aux alentours de 100$ par *animat*. Ceci reste encore relativement élevé car une colonie de 100 éléments reviendrait alors à 10000$. Néanmoins ces coûts peuvent être diminués de plusieurs manières. Tout d'abord, au niveau des prix des pièces elles-mêmes, nous n'avons considéré que les prix unitaires ou les prix pour une dizaine de pièces comme prix de gros. Pour une colonie d'une centaine d'*animats* ces coûts peuvent être réduits d'environ 35 à 40%. D'autre part, la plateforme proposée l'est pour un vaste ensemble d'applications. De ce fait, certains composants ne sont peut-être pas utile pour toutes et l'utilisateur peut choisir de ne pas les installer tout comme il est libre d'en rajouter. Supposons par exemple, que l'utilisateur choisisse de ne pas laisser communiquer les *animats* alors, il peut décider de ne pas connecter l'émetteur-récepteur ZigBee réduisant ainsi le coût d'une dizaine de dollars. Il en va de même pour les autres composants. Néanmoins, le but de la carte étant de permettre à l'utilisateur d'avoir une plateforme prête à être utilisée pour des applications de types *animats* standard, nous avons jugé utile de réserver une place sur la carte pour ces composants de base et de laisser à la discrétion de l'utilisateur le choix de les installer et de les utiliser.

D’autre part, tout comme l’utilisateur peut choisir de ne pas utiliser les composants de base, celui-ci peut ajouter des composants à l’aide des 34 broches laissées libres d’usage pour des cartes d’extensions. Il est suggéré de reproduire la technique employée ou une technique similaire afin de multiplexer les capteurs utilisés et connaître leur consommation si l’apprentissage de la gestion des ressources énergétiques est primordial.

Concernant la prise en compte des problèmes énergétiques, l’*animat* proposé a la particularité de pouvoir éteindre la quasi-totalité de ses composants externes en plus de connaître leur consommation. De ce fait, il est possible d’imaginer une large palette d’applications où ce dernier devrait apprendre à gérer ses ressources à l’aide d’algorithmes d’apprentissage machine par exemple. D’autre part, toujours dans cette optique, nous avons doté la plateforme d’une capacité de recharge solaire des batteries rendant ainsi l’*animat* capable de se recharger si cela est nécessaire. D’autre part, pour les problèmes spécifiques de consommation liés aux radiocommunications, nous avons proposé un module de réveil en logique asynchrone permettant de réduire la période d’activité de l’émetteur-récepteur. De plus, l’utilisation de logique asynchrone NCL ou SHF-NCL devrait être plus largement utilisée pour permettre la mise en place de circuits d’éveil pour un plus large panel de capteurs. Toutes ces caractéristiques mises ensemble font que la plateforme proposée est parfaitement adaptée pour les applications pour lesquelles les contraintes énergétiques sont fortes.

Par ailleurs, dépendamment de l’activité des sous-modules, nous avons évalué au Chapitre 2 et Chapitre 6 la consommation de la plateforme et en avons déduit l’autonomie. Comparativement aux solutions présentées dans le Chapitre 1, l’autonomie affichée est généralement supérieure à celle proposée par les plateformes existantes. Néanmoins, il est difficile de comparer toutes ces plateformes notamment car les batteries utilisées sont différentes, de même que la motorisation ou la masse. Dans notre cas, nous avons choisi de minimiser la taille et la masse du robot, nous empêchant ainsi d’utiliser une batterie de capacité plus grande et par conséquent plus lourde. Ceci réduit alors l’autonomie mais permet, par exemple, d’utiliser des moteurs moins puissants et qui consomment de ce fait moins d’énergie. Par ailleurs, pour comparer avec le plus de justesse les différentes plateformes il faudrait s’assurer que celles-ci sont utilisées dans les mêmes situations. Les autonomies rapportées dans le Chapitre 1, l’étaient généralement pour une utilisation à 100% des périphériques. Néanmoins, l’autonomie d’environ 6h évaluée pour cette plateforme est

relativement compétitive par rapport aux plateformes présentées et ne tient pas compte de la possibilité de recharge solaire.

Enfin, en plus de servir la communauté scientifique travaillant sur les algorithmes d'apprentissage, cette plateforme peut-être destinée à des étudiants en Génie Électrique à divers stades de leur formation. De ce fait, la prise en main devrait être facilitée. Pour cela, la plateforme proposée détient deux principaux avantages : en utilisant un microcontrôleur de type PIC, l'utilisateur peut utiliser l'environnement MPLAB de Microchip qui est gratuit. D'autre part, il est possible de programmer l'*animat* aussi bien en assembleur qu'en C permettant aux personnes désireuses de s'affranchir au maximum des contraintes matérielles de pouvoir programmer avec un langage de relativement haut niveau. D'autre part, un canevas de code est proposé pour n'avoir à modifier que quelques fonctions si cela est voulu. Enfin, concernant l'interactivité, nous avons conçu une interface graphique relativement simple mais permettant de connaître l'état du réseau, de communiquer avec l'*animat* et de connaître par exemple l'état de ses capteurs ou de le contrôler.

Pour parfaire la comparaison, nous résumons dans le Tableau 7.1 les principales caractéristiques de la plateforme proposée et les comparons à celles des plateformes présentées dans le Chapitre 1. Pour des raisons de lisibilité, nous complétons donc le Tableau 7.1 avec les données déjà énoncées. Les principales caractéristiques des autres plateformes sont disponibles sur la plateforme d'*animats* proposée mais que celles-ci propose en plus des dispositifs pour la gestion des ressources énergétiques. Par ailleurs, un effort particulier a été mis en œuvre pour avoir une plateforme aussi petite que possible tout en embarquant l'ensemble des dispositifs voulus.

D'autre part, bien que la conception de la plateforme soit complète, celle-ci ne représente que la première étape d'un plus vaste projet. En effet, sur le plan technique, compte tenu du temps restreint de tout projet, l'ensemble des capacités de la plateforme n'a pu être testé. D'autre par, sur le plan pratique, les applications mises en œuvres furent simples et avaient comme but principalement de vérifier le comportement de certains modules. Par conséquent, il reste maintenant à utiliser la plateforme, à exploiter au maximum ses capacités et à en relever les faiblesses et autres limitations dans le but d'améliorer cette dernière.

Tableau 7.1. Comparaison finale des plateformes

Nom	Année	Autonomie	Masse	Vitesse	Prix	MCU	Communication
Notre Travail	2012	~8 heures	101g	12cm/s	~100$	PIC24FJ256GA108 @ 32 MHz	Radio (ZigBee) et Infrarouge
Khepera	1996	45 min	80g	1.0m/s	~1800$	Motorola 68331 @ 16 MHz	Radio (extension)
Khepera II	2002	1 heure	80g	0.5m/s	-	Motorola 68331 @ 25 MHz	Port série
Khepera III	2006	8 heures	690g	0.5m/s	3800$	DsPIC30F5011 @ 60MHz	Port série USB,WIFI et Ethernet (en extension)
Alice	2004	10 heures	5g	40mm/s	>50$	PIC16LF877	Infrarouge et Radio
e-puck	2004	2 heures	200g	13cm/s	Open Source Vendu 850$	DsPIC30F6014A @ 30 MHz	RS232 et Bluetooth ZigBee en extension
Jasmine	2005	1 – 2 heures	~100g	-	~150$	ATMEGA168-AI	Infrarouge et UART
Kilobot	2011	3 – 10 heures	-	~1cm/s	~14$ Vendu : 130$	ATMEGA328 @8Mhz	Infrarouge
Robomote	2002	20 – 30 min	90 g	~20cm/s	~150$	ATMEGA8535 @ 8MHz	Infrarouge et Radio
Moway®	2007	2 heures	96 g	~18cm/s	~200$	PIC18F87J50 @ 4MHz	Infrarouge Radio en extension

Néanmoins, actuellement, la solution proposée permet de réaliser un vaste ensemble d'applications que ce soit en robotique traditionnelle, réseau de capteurs sans-fil ou encore pour l'apprentissage machine. Sur le plan académique, plusieurs disciplines peuvent être mise en œuvre comme par exemple, la réseautique, la programmation embarquée, la gestion d'énergie, l'apprentissage machine, le contrôle et dans une moindre mesure la mécanique. Les applications pouvant être déployées sont nombreuses notamment grâce à la multitude de capteurs internes, externes et à ceux que l'utilisateur peut rajouter. Parmi celles-ci, il est possible d'imaginer un *animat* ne sachant pas comment se déplacer et qui apprendrait grâce à l'accéléromètre et aux capteurs infrarouges à avancer en ligne droite en appliquant les bonnes tensions sur ces moteurs. De même, grâce à ces capteurs de courant, ce dernier peut apprendre un vaste ensemble de choses seulement limité par la créativité de l'utilisateur. Notamment, l'*animat* pourrait apprendre que dans certains environnements, comme par exemple les pentes, le ratio entre la consommation pour le déplacement et la communication peut changer, ce qui n'aurait pas été possible avec une information fixée *a priori*. Enfin, le capteur de lumière et le panneau solaire permettent, comme nous le mentionnions déjà auparavant, de mettre en place des applications au cœur de ce que ce sont les *animats* : des éléments tâchant de survivre le plus longtemps dans un environnement en optimisant leurs dépenses énergétiques et en cherchant de la « nourriture » pour se recharger.

CONCLUSION

Ce projet propose une plateforme d'*animats* qui ne sont autres que des robots de petites tailles ayant *a priori* peu de capacités mécaniques et évoluant dans un environnement inconnu. La tâche de ces derniers étant de parcourir leur environnement et de découvrir des façons d'y évoluer pour survivre le plus longtemps possible. D'autre part, ces derniers peuvent évoluer en colonies et partager leurs « points de vues » sur certaines choses comme la façon de répondre à des événements extérieurs de manière peu énergivore. Les autres éléments de la colonie peuvent alors choisir d'élire des experts par rapport à certains sujets, de leur faire confiance ou de se forger leur propre opinion.

Pour ce faire, nous avons tout d'abord énoncé le cadre théorique sous-jacent, présenté les principales plateformes existantes et en avons extrait certaines caractéristiques qui ont été les contraintes de conception.

Dans un second temps, nous avons exposé l'architecture à proprement parler de la plateforme d'*animats* et en avons justifié les choix de conception. Notamment, les principales caractéristiques de celle-ci sont une faible consommation (de l'ordre de 130 mA lorsque l'ensemble des sous-modules de base sont actifs), un nombre élevé de capteurs sur la plateforme standard, la possibilité d'ajouter un grand nombre de capteurs dépendamment des contraintes applicatives, mais surtout, la possibilité de délocaliser les calculs et celle de gérer les dépenses énergétiques. Cette dernière caractéristique, très importante pour les applications de type *animat*, permet à ce dernier de prendre conscience de ses propres dépenses et de les adapter, ce qui, au sens large du terme, peut être vu comme de l'introspection.

Ensuite, nous nous sommes préoccupés des problèmes liés à la consommation des moyens de radiocommunications. Pour cela, nous avons étudié puis proposé un module de réveil en logique asynchrone efficace sur le plan énergétique. Par ailleurs, nous avons proposé une technique pour diminuer la quantité de ressources utilisées en suivant les patrons de conception en logique NCL. En utilisant cette nouvelle technique de conception nous avons montré sur plusieurs exemples qu'une réduction de complexité de 65% pouvait être obtenue lorsqu'elle était comparée au gain maximal. Bien que l'utilisation de cette technique fût destinée, pour notre application, à une implémentation sur FPGA, nous avons aussi discuté de l'efficacité de celle-ci dans un environnement de conception de type ASIC. Le module de réveil ainsi proposé peut aussi bien

servir pour la plateforme d'*animats* que pour des réalisations plus générales comme les réseaux de capteurs sans-fil pour lesquels la consommation énergétique est cruciale. D'autre part, cela nous a permis de souligner le fait que la logique asynchrone pouvait être très appréciable pour des applications où l'activité est éparse et dépendante d'événements extérieurs. En effet, en plus de montrer sur cet exemple que la consommation dynamique avait était réduite de moitié en diminuant la complexité de l'implémentation de l'architecture, nous l'avons aussi comparé sur le plan énergétique avec un circuit équivalent réalisé en logique synchrone sur le même FPGA. Les résultats sont sans appel; profitant de l'état de veille inhérent propre aux circuits asynchrones, ceux-ci montrent une consommation dynamique moyenne 10 fois inférieure dans la version NCL et 20 fois inférieure dans la version SHF-NCL, lorsque l'activité des communications est supposée être faible.

Finalement, certains détails applicatifs et quelques expérimentations ont été détaillés, de même que l'environnement graphique proposé pour l'utilisation de la plateforme.

En conclusion, le travail réalisé a permis d'une part de proposer une colonie de robots qui seront utilisés aussi bien dans un cadre académique à plusieurs niveaux de l'enseignement de la robotique, de la programmation embarquée, de la réseautique ou de l'apprentissage machine que dans un cadre de recherche notamment au Laboratoire de Réseaux Neuronaux de l'École Polytechnique de Montréal afin d'analyser et de proposer de nouveaux algorithmes d'apprentissage dont l'efficacité pourra être prouvée dans des environnements réels. D'autre part, une amélioration d'un patron de conception en logique asynchrone a été proposée et devrait conduire à une réduction certaine des ressources utilisées.

BIBLIOGRAPHIE

[1] F. Rosenblatt, *The perceptron, a perceiving and recognizing automaton Project Para*: Cornell Aeronautical Laboratory, 1957.

[2] M. Minsky and S. Papert, *Perceptrons: an introduction to computational geometry*: MIT Press, 1969.

[3] D. E. Rumelhart, *et al.*, "Learning internal representations by error propagation," in *Neurocomputing: foundations of research*, ed: MIT Press, 1988, pp. 673-695.

[4] D. B. Parker, "Learning logic," Technical Report TR-47, Center for Computational Research in Economics and Management Science, MIT, 1985.

[5] B. E. Boser, *et al.*, "A training algorithm for optimal margin classifiers," presented at the Proceedings of the fifth annual workshop on Computational learning theory, Pittsburgh, Pennsylvania, United States, 1992.

[6] J. Park and I. W. Sandberg, "Universal Approximation Using Radial-Basis-Function Networks," *Neural Computation*, vol. 3, pp. 246-257, 1991/06/01 1991.

[7] R. A. Jacobs, *et al.*, "Adaptive Mixtures of Local Experts," *Neural Computation*, vol. 3, pp. 79-87, 1991/02/01 1991.

[8] W. Maass, *et al.*, "Real-Time Computing Without Stable States: A New Framework for Neural Computation Based on Perturbations," *Neural Computation*, vol. 14, pp. 2531-2560, 2002/11/01 2002.

[9] B. International Conference on Simulation of Adaptive, *et al.*, "From animals to animats proceedings of the first International Conference on Simulation of Adaptive Behavior," Cambridge, Mass.

[10] I. S. Technologies. (2005, *LEURRE : Artificial Life Control in Mixed Socities*. Available: http://leurre.ulb.ac.be/

[11] Animats, "Animats : Ragdolls and More," 2001.

[12] *MUTE Project*. Available: http://mute-net.sourceforge.net/howAnts.shtml

[13] S. Nolfi and D. Floreano, "Coevolving Predator and Prey Robots: Do “Arms Races” Arise in Artificial Evolution?," *Artificial Life*, vol. 4, pp. 311-335, 1998/10/01 1998.

[14] F. a. F. Mondada, E. and Guignard, A., "The Development of Khepera," presented at the Experiments with the Mini-Robot Khepera, Proceedings of the First International Khepera Workshop, Paderborn, 1999.

[15] M. a. T. Dorigo, E. and Mondada, Francesco and Nolfi, S. and Deneubourg, J.-L. and Floreano, D. and Gambardella, L.M., "The SWARM-BOTS Project," *Künstliche Intelligenz*, pp. 32-35, 2005.

[16] M. Dorigo. (2001, *Swarms-bots*. Available: http://www.swarm-bots.org/

[17] G. Caprari, *et al.*, "The autonomous micro robot “Alice”: a platform for scientific and commercial applications," in *Micromechatronics and Human Science, 1998. MHS '98. Proceedings of the 1998 International Symposium on*, 1998, pp. 231-235.

[18] Mondada F., *et al.*, "The e-puck, a robot designed for education in engineering," in *In Proceedings of the 9th Conference on Autonomous Robot Systems and Competitions*, 2009, pp. 59-65.

[19] MRSG. (1997, *Microrobots*. Available: http://diwww.epfl.ch/lami/mirobots/1cubes.html

[20] (2005, *Jasmine: swarm robot platform*. Available: http://www.swarmrobot.org/index.html

[21] Rubenstein M., *et al.*, "Kilobot: A Low Cost Scalable Robot System for Collective Behaviors," Harvard University2011.

[22] G. T. Sibley, *et al.*, "Robomote: a tiny mobile robot platform for large-scale ad-hoc sensor networks," in *Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on*, 2002, pp. 1143-1148.

[23] Minirobots S.L. (2007, *Moway - Bring it to life*. Available: http://www.moway-robot.com/

[24] B. Latré, *et al.*, "A survey on wireless body area networks," *Wirel. Netw.*, vol. 17, pp. 1-18, 2011.

[25] IEEE, "IEEE Standard for Local and metropolitan area networks," in *Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs)*, ed: IEEE, 2011.

[26] Z. Alliance. (2012. Available: http://www.zigbee.org/

[27] A. Technologies. (2010, *About ZigBee Baseband Verification Library*. Available: http://edocs.soco.agilent.com/display/sv201001/About+ZigBee+Baseband+Verification+Library

[28] D. P. Lattibeaudiere, "Microchip ZigBee-2006 Residential Stack Protocol," Microchip Technology Inc.2008.

[29] D. P. Lattibeaudiere, "Microchip ZigBee® PRO Feature Set Protocol Stack," Microchip Technology Inc.2009.

[30] E. Candes, *et al.*, "Compressed Sensing with Coherent and Redundant Dictionaries," *Applied and Computational Harmonic Analysis*, vol. 31, pp. 1-21, 2010.

[31] R. Agarwal, *et al.*, "Adaptive asynchronous analog to digital conversion for compressed biomedical sensing," in *Biomedical Circuits and Systems Conference, 2009. BioCAS 2009. IEEE*, 2009, pp. 69-72.

[32] A. J. Martin, "Limitations to Delay-Insensitivity in Asynchronous Circuits," California Institute of Technology, Technical Report1990.

[33] I. David, *et al.*, "An efficient implementation of Boolean functions as self-timed circuits," *IEEE Transactions on Computers*, vol. 41, pp. 2-11, 1992.

[34] J. Sparso, *et al.*, "Design of delay insensitive circuits using multi-ring structures," in *European Design Automation Conference -EURO-VHDL '92, September 7, 1992 - September 10, 1992*, Hamburg, Ger, 1992, pp. 15-20.

[35] N. P. Singh, "A DESIGN METHODOLOGY FOR SELF-TIME SYSTEMS," Massachusetts Institute of Technology1981.

[36] S. C. Smith, *et al.*, "Optimization of NULL convention self-timed circuits," *Integration, the VLSI Journal*, vol. 37, pp. 135-165, 2004.

[37] S. C. Smith and J. Di, *Designing asynchronous circuits using NULL convention logic (NCL)*: Morgan & Claypool Publishers, 2009.

[38] I. E. Sutherland, "Micropipelines," *Commun. ACM*, vol. 32, pp. 720-738, 1989.

[39] Karl M. Fant and S. A. Brandt, "NULL Convention Logic™," Theseus Logic Inc.1997.

[40] R. Reese. (2010, *Unified NULL Convention Logic Environment (UNCLE)*. Available: https://sites.google.com/site/asynctools/

[41] H. Kaeslin. (2008, *Digital integrated circuit design from VLSI architectures to CMOS fabrication*. Available: http://proxy2.hec.ca/login?url=http://library.books24x7.com/library.asp?B?&bookid=29297

[42] T. v. Dam and K. Langendoen, "An adaptive energy-efficient MAC protocol for wireless sensor networks," presented at the Proceedings of the 1st international conference on Embedded networked sensor systems, Los Angeles, California, USA, 2003.

[43] R. Min, *et al.*, "Low-power wireless sensor networks," in *VLSI Design, 2001. Fourteenth International Conference on*, 2001, pp. 205-210.

[44] Z. Yan, *et al.*, "An analytical model for energy efficiency analysis of different wakeup radio schemes," in *Personal, Indoor and Mobile Radio Communications, 2009 IEEE 20th International Symposium on*, 2009, pp. 1148-1152.

[45] F. Ashraf, *et al.*, "Synchronization vs. Signaling: Energy-Efficient Coordination in WSN," in *Wireless Mesh Networks (WIMESH 2010), 2010 Fifth IEEE Workshop on*, 2010, pp. 1-6.

[46] M. A. Ameen, *et al.*, "Design and analysis of a MAC protocol for wireless body area network using wakeup radio," in *Communications and Information Technologies (ISCIT), 2011 11th International Symposium on*, 2011, pp. 148-153.

[47] L. Gu and J. A. Stankovic, "Radio-Triggered Wake-Up for Wireless Sensor Networks," *Real-Time Systems*, vol. 29, pp. 157-182, 2005.

[48] Z. Yan, *et al.*, "A 3.72μW ultra-low power digital baseband for wake-up radios," in *VLSI Design, Automation and Test (VLSI-DAT), 2011 International Symposium on*, 2011, pp. 1-4.

[49] N. M. Pletcher, *et al.*, "A 2GHz 52μW wake-up receiver with -72dBm sensitivity using uncertain-IF architecture," in *2008 IEEE International Solid State Circuits Conference, ISSCC, Febrary 3, 2008 - Febrary 7, 2008*, San Francisco, CA, United states, 2008, pp. 524-525+633+521.

[50] C. Hambeck, *et al.*, "A 2.4μW Wake-up Receiver for wireless sensor nodes with -71dBm sensitivity," in *Circuits and Systems (ISCAS), 2011 IEEE International Symposium on*, 2011, pp. 534-537.

[51] M. S. Durante and S. Mahlknecht, "An Ultra Low Power Wakeup Receiver for Wireless Sensor Nodes," in *Sensor Technologies and Applications, 2009. SENSORCOMM '09. Third International Conference on*, 2009, pp. 167-170.

[52] P. Le-Huy and S. Roy, "Low-Power 2.4 GHz Wake-Up Radio for Wireless Sensor Networks," in *Networking and Communications, 2008. WIMOB '08. IEEE International Conference on Wireless and Mobile Computing*, 2008, pp. 13-18.

ANNEXE 1 – Classes d'applications des réseaux de capteurs sans-fil

Les réseaux de capteurs sans-fil sont généralement représentés par les cinq classes d'applications mentionnées ci-après. La description faite est un résumé de ce que l'on peut trouver dans la littérature. Notamment, de plus amples détails peuvent être trouvés dans [24].

Wireless Body-Area Networks :

La fonction principale de tels réseaux est d'assurer la surveillance et de faire l'historique de l'état de santé d'une personne. Par extension, ce modèle peut aussi être utilisé pour la surveillance de l'état de machines comme dans le domaine industriel. Les principales caractéristiques de cette classe sont les suivantes: une proximité entre les nœuds (lorsqu'il s'agit du corps humain, on peut considérer que les nœuds sont contenus dans une sphère de 1 à 2 m de rayon), un nombre relativement faible de nœuds, pas de redondance, les grandeurs mesurées sont physiques ou physiologiques, les nœuds sont distribués à des endroits stratégiques, la bande passante nécessaire est faible et les traitements sont généralement de type prioritaire. Étant donné le faible nombre de capteurs, leur proximité et le caractère prioritaire des traitements, le type de réseau utilisé sera souvent un réseau en étoile. Enfin, les dispositifs doivent d'une part travailler en temps réel et, d'autre part, être suffisamment résistants aux changements d'environnement que peut subir le porteur (intempéries, changement de température, accélération, dispositifs sous la peau, ...).

Wireless Data Collection Networks :

Celui-ci désigne les réseaux dont les nœuds distribués sur d'importantes surfaces doivent acquérir un grand nombre de données. Il est souvent fait allusion à des applications comme la surveillance d'incendie de forêts ou le contrôle de l'état de plantations (taux d'humidité, température, etc.). Généralement, indépendamment de la distribution préalable des capteurs, il existe une forte corrélation entre la donnée mesurée et la position du capteur. De plus, à l'instar de la température, étant donné cette corrélation et la multitude de capteurs, on constate souvent que l'information reçue est redondante. Cette redondance fait d'ailleurs partie intégrante de l'information et, par exemple, un écart par rapport à la moyenne peut signaler une anomalie. Pour ce genre d'application, compte tenu de la distribution des capteurs et du fait que l'on veuille minimiser la

maintenance, la durée d'opération ainsi que l'autonomie vont constituer deux paramètres importants. En ce qui concerne le type de réseau, les larges surfaces couvertes tendent à désigner une topologie maillée (*mesh*) avec des sous-structures. Ce qui impliquera notamment des communications « multi-sauts », des temps de réponses élevés ainsi qu'une probabilité de perte de paquet non-négligeable.

Wireless Location-Sensing Networks :

À l'image de l'engouement pour les dispositifs RFID, cette classe d'applications regroupe celles où l'identité du capteur ainsi que sa localisation physique ou relative constituent l'information à transmettre. La donnée physique mesurée, si elle existe, est considérée comme une information supplémentaire. Une des applications souvent citée concerne de petits écrans placés sur les chariots des supermarchés et qui pourraient orienter le client vers une promotion ou suggérer un complément en fonction de l'endroit où le client s'attarde. Une autre application est la gestion des stocks dans un entrepôt. Par conséquent, pour ces dispositifs, la localisation est un facteur très important de même que la consommation d'énergie. Naturellement, une topologie maillée ou hiérarchique est importante pour la localisation (triangulation…).

Wireless Multimedia Sensor Networks :

Compte tenu de l'apparition sur le marché de caméra CMOS à faible coût et à basse consommation d'énergie, les réseaux multimédia tendent de plus en plus à se développer pour des applications aussi bien de sécurité et surveillance, que de télé-présence (visite de musée à distance) ou encore du contrôle de l'activité que ce soit de l'activité animale (déplacement d'une espèce, etc.) ou industrielle (fonctionnement d'une chaîne de montage). Dépendamment de l'application, on cherche donc ici à mesurer généralement de manière redondante un flux vidéo, audio ou les deux. Cette fois-ci la redondance ne peut plus être utilisée pour détecter un écart, en revanche, elle peut servir à compléter une information manquante. Comme par exemple dans le cas de vidéosurveillance où deux caméras viseraient une même scène mais selon des angles différents. La distribution des capteurs est donc très importante et devrait être faite de manière stratégique. Conséquemment à la forte demande en bande passante, le nombre de nœuds est souvent limité. De plus, les informations devant être transmises à un même point de collecte, le type de réseau est dans la plupart des cas un réseau en étoile.

Wireless Control-Oriented Sensor Networks :

Cette dernière classe d'applications se démarque des précédentes en ce sens qu'en plus de recueillir des données, certains dispositifs permettent aussi d'agir sur l'environnement au travers d'actionneurs. L'exemple le plus cité est l'application interrupteur/lumière. Un nœud relève l'information sur l'interrupteur et l'envoie à l'autre nœud qui gère la lumière tout cela par onde radio : le but étant, par exemple, d'éviter d'avoir des interrupteurs fixes dans une maison mais de pouvoir les placer où l'on veut sans pour autant avoir à amener un câble électrique. Généralement, ce type d'application nécessite une faible bande passante et l'utilisation de routeur n'est pas nécessaire.

ANNEXE 2 – Classes d'applications basées sur le type de communication

Ici, nous proposons de différencier les différentes classes d'applications des réseaux de capteurs sans-fil dépendamment de l'utilisation de moyen de communication. On distingue alors trois cas qui sont détaillés ci-après.

Communications continues :

Cette classe d'applications regroupe principalement les modules qui sont utilisés comme des collecteurs reliés à une station de traitement comme un PC. Par conséquent, ces applications sont caractérisées par une contrainte temps réel plus importante que celle énergétique. Étant donnée la communication permanente à laquelle ces nœuds participent, leurs modules de communication devront être constamment actifs. Il est alors largement envisageable de considérer que ces nœuds seront alimentés directement par le réseau électrique.

Communications périodiques :

Ce type de communication est certainement le plus répandu dans les WSN car il permet aux nœuds d'atteindre des temps de vie de quelques mois voire quelques années moyennant une configuration correcte. Le principe étant de réveiller le nœud à intervalles de temps prédéterminés. Ces intervalles ne sont pas nécessairement réguliers bien que ce soit le cas le plus courant. Une fois réveillé, celui-ci écoute le canal de communication pour savoir s'il peut émettre ou bien si un message lui a été destiné. Dans le cas échéant il se rendort directement, sinon, il traite les communications puis se rendort. Le principal avantage est de bénéficier des modes de veille des différents composants durant les périodes d'inactivités.

Usuellement, les relevés sur l'environnement précèdent le passage en mode de veille réduisant ainsi le bilan énergétique. Malheureusement, lorsque les besoins de l'application requièrent une fréquence d'échantillonnage qui diffère de celle liée à la communication, le modèle de mise en veille précédent est dégradé. Cette classe regroupe donc principalement les applications pour lesquelles on cherche à avoir des relevés périodiques de différents capteurs dont l'importance des fluctuations temporelles des valeurs relevées est supposée être faible. Le cas typique étant le relevé de température.

Communications lorsque nécessaire :

Finalement, ce modèle représente un mode de communication différent des précédents en ce sens que les nœuds sont réveillés lorsqu'un message leur est destiné. Par conséquent, il est éventuellement possible de choisir des modes de veille profonds et plus longs que ceux pour des réveils périodiques. Cette classe comprend généralement les cas où un module doit émettre un signal lorsqu'un événement peu fréquent survient. Pour illustrer cette classe on peut citer les exemples suivants : les systèmes de sécurité comme les alarmes pour lesquels on souhaite que le dispositif reste en alerte jusqu'à la détection d'une intrusion, ou encore des dispositifs embarqués sur des animaux où l'information pertinente est l'évolution sur une longue période de temps.

ANNEXE 3 – Modèles énergétiques

En se basant sur la classification relative à l'utilisation des moyens de communications présentée dans l'ANNEXE **2**, il est possible de proposer un modèle énergétique pour chacun des groupes. Ce modèle permettra notamment de pouvoir quantifier les dépenses relatives à une application et de dimensionner celle-ci en conséquence. Pour ce faire, reprenons les différentes classes exposées dans l'ANNEXE **2**. Les différentes notations utilisées sont résumées dans le tableau ci-dessous. Pour l'analyse on utilisera la propriété additive de l'énergie.

Tableau A3.1. Notations utilisées pour les modèles énergétiques

Paramètres	Signification
E_{tot}	Énergie totale de l'*animat*
$E_{travail}$	Énergie utile pour traiter l'information et réaliser les différentes tâches
E_{RX}	Énergie nécessaire pour la réception de messages
E_{TX}	Énergie nécessaire pour l'émission de messages
E_{calc}	Énergie nécessaire pour la réalisation des calculs
$E_{mém}$	Énergie nécessaire pour l'utilisation de la mémoire
$E_{réveil}$	Énergie nécessaire pour sortir du mode de veille
E_{veille}	Énergie utilisée durant le mode de veille
E_{act}	Énergie utilisée par les actuateurs
$E_{c,i}$	Énergie utilisée par le capteur *i*
$f_{éch,i}$	Fréquence d'échantillonnage du capteur *i*
P_{RX}	Puissance consommée pour l'émission de messages
P_{TX}	Puissance consommée pour la réception de messages
$P_{réveil}$	Puissance consommée pour sortir du mode de veille
P_{veille}	Puissance consommée pendant la veille
$p_{mt}(l,t)$	Densité de probabilité de transmission d'un message en fonction de la taille et du temps
$p_{mr}(t,l)$	Densité de probabilité de réception d'un message en fonction de la taille et du temps
$p_{mt\|sync}$	Densité de probabilité de transmission d'un message sachant que l'*animat* est autorisé à communiquer en fonction de la taille et du temps
$p_{mr\|sync}$	Densité de probabilité de réception d'un message sachant que l'*animat* est autorisé à communiquer en fonction de la taille et du temps
n_c	Nombre de capteurs
T_{batt}	Durée de vie de la batterie

Communications continues :

Compte tenu de l'utilisation permanente des moyens de communications, l'énergie totale des nœuds utilisant ce modèle est égale à l'énergie utile à laquelle s'ajoute l'énergie pour les émissions et les réceptions de messages :

$$E_{tot} = E_{travail} + E_{RX} + \iint p_{mt}(l,t).P_{TX}(t).dl.dt + E_{act}$$

L'expression de l'énergie utile peut différer d'une application à une autre. Néanmoins, sans perte de généralité, on peut la décomposer comme étant la somme de l'énergie nécessaire à l'échantillonnage réalisé par les capteurs, avec celle pour réaliser les calculs et celle pour conserver les données en mémoire :

$$E_{travail} = \left(\sum_{i=0}^{n_c-1} f_{éch,i}E_{c,i}\right)T_{batt} + E_{calc} + E_{mém}$$

On notera que le fait de décrire l'énergie nécessaire pour réaliser les calculs par E_{calc} n'empêche pas la possibilité selon laquelle l'unité de calcul puisse se trouver dans des modes de veille. Il en va de même pour la mémoire. On obtient finalement :

$$\boxed{E_{tot} = \left(\sum_{i=0}^{n_c-1} f_{éch,i}E_{c,i}\right)T_{batt} + E_{calc} + E_{mém} + E_{RX} + \iint p_{mt}(l,t).P_{TX}(t).dl.dt + E_{act}}$$

Remarque : Bien que les intégrales utilisées semblent être définies sur des intervalles non bornés, il est nécessaire de rappeler que pour de telles applications, aussi bien la longueur des messages mis en jeu que la durée de vie de la batterie si elle est utilisée sont des grandeurs finies. Dans le cas d'un dispositif connecté au réseau électrique, il est possible que ces intégrales divergent bien qu'ayant aucune réalité physique.

Communications périodiques :

Ce modèle devrait différer du précédent en ce sens que la communication n'est pas constamment active mais est utilisée selon une certaine fréquence de synchronisation. On tient ici compte des surcoûts dus au passage du mode éveillé au mode de veille et vice-versa. De ce fait, l'énergie liée aux communications est modifiée de la sorte :

$$E_{RX} = \iint p_{mr|sync}(t,l).P_{RX}.dl.dt$$

$$E_{TX} = \iint p_{mt|sync}(t,l).P_{TX}.dl.dt$$

$$E_{réveil} = \int p_{sync}(t).P_{réveil}.dt$$

$$E_{veille} = \iint \left(1 - p_{sync}(t)\right).P_{veille}.dl.dt$$

L'expression sous forme compacte de l'énergie totale devient alors la suivante :

$$\boxed{E_{tot} = E_{RX} + E_{TX} + E_{veille} + E_{réveil} + \left(\sum_{i=0}^{nc-1} f_{éch,i}.E_{c,i}\right).T_{batt} + E_{calc} + E_{mém} + E_{act}}$$

Communications lorsque nécessaire :

Finalement, dans ce dernier modèle, la probabilité d'utilisation des moyens de communications dépend uniquement de la probabilité de réception ou d'envoi des messages et n'a donc pas à être corrélée avec une quelconque synchronisation. Par conséquent, les équations des énergies relatives à ces moyens de communications deviennent :

$$E_{RX} = \iint p_{mr}(t,l).P_{RX}.dl.dt$$

$$E_{TX} = \iint p_{mt}(t,l).P_{TX}.dl.dt$$

$$E_{réveil} = \iint \max\left(1, p_{mr}(t,l) + p_{mt}(t,l)\right).P_{réveil}.dl.dt$$

$$E_{veille} = \iint \left(1 - \max\left(1, p_{mr}(t,l) + p_{mt}(t,l)\right)\right).P_{veille}.dl.dt$$

L'expression générale de l'énergie totale sous forme compacte reste inchangée :

$$\boxed{E_{tot} = E_{RX} + E_{TX} + E_{veille} + E_{réveil} + \left(\sum_{i=0}^{nc-1} f_{éch,i}.E_{c,i}\right).T_{batt} + E_{calc} + E_{mém} + E_{act}}$$

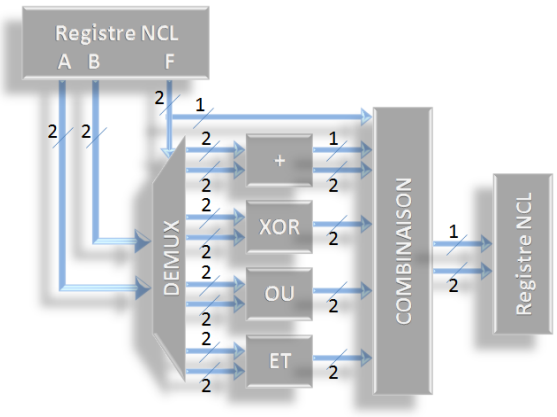
L'efficacité d'un modèle par rapport à un autre est très largement dépendante du type d'application. En effet, pour un nœud réveillé fréquemment, le surcoût de consommation dû au dispositif de réveil du récepteur principal peut devenir inefficace par rapport à un module qui se réveillerait de manière périodique. De même, ce modèle peut être inadapté dès lors que de faibles latences sont requises. Inversement, une application où la communication survient peu fréquemment et pour laquelle il y a une forte contrainte sur la consommation d'énergie, pourrait bénéficier d'un module de réveil. Enfin si la latence est le facteur limitant et que la consommation constitue une contrainte plus faible, il se peut que le modèle de communication continue soit le plus efficace. En d'autres termes, le choix de l'un ou l'autre de ces modèles dépend de l'application et de ses contraintes.

ANNEXE 4 – Exemples de circuits en logique NCL™

Nous nous intéressons ici à quelques exemples de circuit conçus en logique NCL. Cette présentation permettra au lecteur de pouvoir se familiariser un peu plus avec ce style de logique et d'appréhender les compromis qu'une telle conception soulève. Par conséquent cette annexe n'a pas la prétention de donner un aperçu de l'éventail des possibilités de conception offertes par le NCL mais nous permet de présenter des exemples simples dont nous nous resservirons lorsque nous devrons comparer les gains apportés par la simplification présentée au Chapitre 4. Pour ce faire, nous considérerons une unité arithmétique et logique (ALU) présentée dans [37] et un circuit d'incrémentation que nous détaillerons. Le lecteur intéressé pourra trouver des circuits plus complexes tels que la réalisation de processeur en logique asynchrone NCL dans la littérature sur le sujet notamment à propos du microcontrôleur 8-bit NCL08.

Unité Arithmétique et Logique (ALU)

Commençons par analyser le cas de l'ALU qui traite deux symboles de deux bits chacun et est capable de réaliser quatre opérations différentes : la fonction OU, la fonction ET, la fonction OU-Exclusif et une addition. Pour ce faire, ce circuit reçoit les nombres A et B ainsi que le code d'opération F également sur deux bits et réalise les calculs en fonction de ce dernier. En sortie, le résultat est retourné sur deux bits accompagné d'un bit supplémentaire lorsque l'addition déborde. La Figure A4-1 présente le schéma simplifié de l'ALU ainsi que la correspondance entre le code d'opération et l'opération réalisée. Néanmoins, comme nous l'avons vu lors de l'analyse du protocole, ce bit devra être forcé à DATA0 pour les opérations autres que l'addition. On peut alors décomposer cette ALU en trois sous-parties : le démultiplexeur d'entrée, les circuits de réalisations des calculs et le circuit de génération du résultat.



(a)

F		Fonction	Résultat	
F_1	F_0		C	OV
DATA0	DATA0	ET	A ET B	0
DATA0	DATA1	OU	A OU B	0
DATA1	DATA0	XOR	A⊕B	0
DATA1	DATA1	ADD	A + B	Débordement

(b)

Figure A4-1. Unité Arithmétique et Logique: (a) Schéma, (b) Codes d'opérations

Démultiplexeur :

Le circuit de démultiplexage n'a rien de particulier : il permet d'acheminer les entrées à la bonne unité de calcul. Dans un circuit synchrone, tous les calculs auraient été faits en parallèle et la sélection aurait été faite à la sortie. Ici, compte tenu de la règle d'observabilité, pour éviter de générer des signaux orphelins, les unités de calculs non-concernées ne doivent pas être sollicitées. Une des particularités tout de même de ce circuit pour ce cas spécifique, est qu'il propage l'information selon laquelle l'opération effectuée n'est pas l'addition ce qui permettra par la suite de mettre à DATA0 le signal de débordement. Ensuite viennent les unités de calculs que nous allons détailler.

Fonction ET :

En utilisant directement la formule de la fonction logique ET sur un bit, on obtiendrait en logique NCL l'expression de la sortie sous la forme suivante :

$$S_1 = A_1.B_1$$

$$S_0 = A_0 + B_0$$

Malheureusement, ces expressions ne forment pas un ensemble complet par rapport aux entrées de type DATA. En effet, si {A, B} = {DATA0, NULL} par exemple, on aura une sortie valide de type DATA0 ce qui est en contradiction avec la règle de complétude. Pour palier ce problème, il est nécessaire dans ce cas de calculer l'ensemble des mintermes ce qui n'est pas vrai en général avec le NCL. L'équation de S_0 se transforme alors de la sorte :

$$S_0 = A_0.(B_0 + B_1) + B_0.(A_0 + A_1) = A_0.B_0 + A_0.B_1 + A_1.B_0$$

On obtient alors en utilisant les portes NCL les équations suivantes :

$$S_1 = TH22(A_1, B_1)$$

$$S_0 = TH_{AND0}(A_0, B_0, A_1, B_1)$$

Le calcul de la fonction ET bit-à-bit pour N bits se réalise alors en mettant en parallèle N de ces unités de base. Pour le cas N = 2 qui nous intéresse, le schéma final est celui de la Figure A4-2.

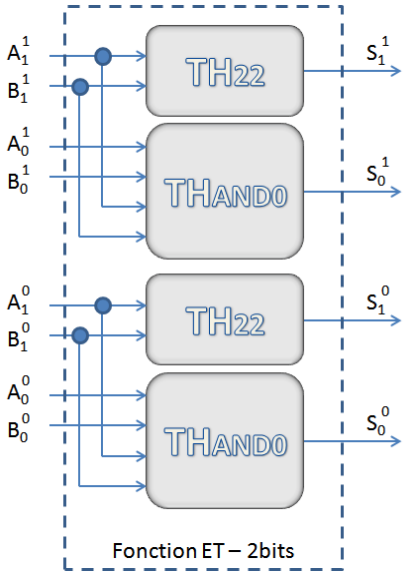


Figure A4-2. Fonction ET 2-bits NCL

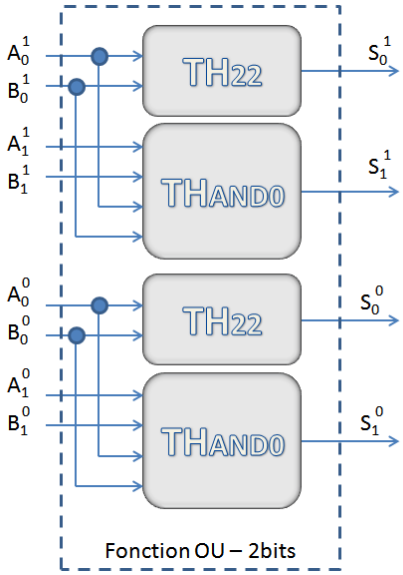


Figure A4-3. Fonction OU 2-bits NCL

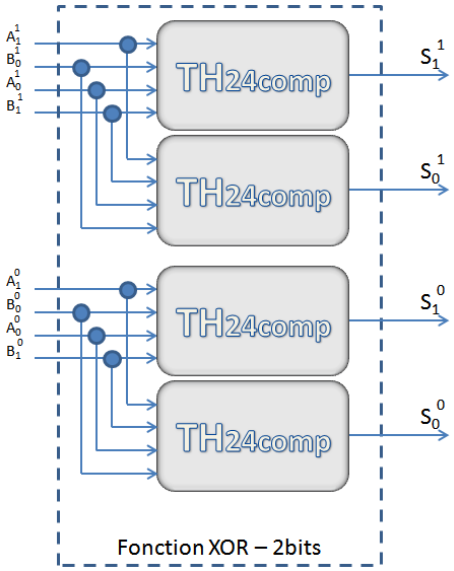


Figure A4-4. Fonction OU-Exclusif 2-bits NCL

Fonction OU :

La démarche est la même que celle décrite précédemment : il est nécessaire de rendre complète l'unité de calcul de base. En reprenant les mêmes étapes on arriverait aux équations suivantes :

$$S_1 = TH_{AND0}(A_1, B_1, A_0, B_0)$$

$$S_0 = TH_{22}(A_0, B_0)$$

De même, pour une opération bit-à-bit avec des symboles d'entrées de taille N = 2, il suffit de mettre en parallèle deux unités définies par les équations précédentes. Le schéma résultant est fourni dans la Figure A4-2.

Fonction OU-Exclusif (XOR) :

L'analyse de cette porte peut conduire à une solution naïve ou bien à une solution optimisée en termes d'utilisation de transistors. Commençons par écrire les équations de la fonction XOR en partant des équations booléennes :

$$S_1 = A_1.B_0 + A_0.B_1$$

$$S_0 = A_1.B_1 + A_0.B_0$$

Ceci ce traduirait directement en termes de portes NCL par l'utilisation de portes TH_{XOR0}. Néanmoins, comme il est possible de supposer que les termes X_0 et X_1 d'un symbole X ne peuvent pas être simultanément actifs, on peut modifier les équations précédentes en ajoutant des termes n'ayant aucun impact (*don't care*). On obtiendrait alors :

$$S_1 = A_1.B_0 + A_0.B_1 + A_0.A_1 + B_0.B_1$$

$$S_0 = A_1.B_1 + A_0.B_0 + A_0.A_1 + B_0.B_1$$

Par conséquent, cela ce traduirait par l'utilisation de deux portes de type TH_{24comp}. On peut alors constater que le nombre de portes NCL est inchangé, en revanche, comme détaillé dans [37], le nombre de transistors sera diminué. De même que précédemment, pour le cas où N = 2, on obtient le schéma de la Figure A4-4. Il est important de noter que pour une implémentation sur FPGA ce changement n'aurait probablement aucun impact compte tenu de l'utilisation de LUT, dans la majorité des cas, pour la réalisation de fonctions.

Additionneur :

En ce qui concerne le circuit de calcul de l'addition, nous pouvons réutiliser ce qui a été décrit dans le Chapitre 3. En effet, pour construire un additionneur de N bits sans tenir en compte de circuit de propagation rapide de retenue, il est possible de se contenter de connecter entre eux N *full-adder*. Néanmoins, compte tenu que nous ne voulons pas, dans ce cas, réaliser de soustraction par exemple et qu'il est important de n'utiliser que le strict minimum de ressources, nous allons modifier le premier de ces N additionneurs pour qu'il ne prenne pas en compte de retenue d'entrée. De plus, cela est rendu nécessaire car dans le cas contraire il aurait fallu générer une retenue d'entrée fictive de type DATA0. Les équations de cet additionneur 1-bit sans retenue d'entrée (*half-adder*) sont données par :

$$S = A \oplus B$$

$$C_s = A.B$$

La première équation se traduit directement en utilisant les résultats obtenus pour la porte XOR (deux portes $TH24_{comp}$). Pour la génération de la retenue de sortie, plutôt que d'utiliser les résultats vus pour la porte ET, on peut se contenter d'utiliser les équations incomplètes suivantes :

$$C_{s1} = A_1.B_1$$

$$C_{s0} = A_0 + B_0$$

Ce qui se traduit par l'utilisation simple d'une porte $TH22$ et d'une porte $TH12$. La raison pour laquelle il est possible ici d'utiliser une expression incomplète pour C_{s0} réside dans le fait que la règle de complétude nous oblige à avoir un ensemble de sorties qui respecte la règle et non chaque sortie individuellement. Raisonnons alors par l'absurde et supposons que l'ensemble de sortie passe du type NULL à DATA alors que l'ensemble d'entrées n'est pas de type DATA. Cela implique tout d'abord que C_s est de type DATA. Compte tenu que l'ensemble des entrées n'est pas de type DATA cela implique que C_{s1} ne peut pas être à 1. Par conséquent, C_s est de type DATA0 et c'est C_{s0} qui est actif. Le problème étant symétrique par rapport aux entrées, supposons que cela est généré par l'entrée A qui est donc de type DATA0. L'ensemble d'entrées est donc de type {A, B} = {DATA0, NULL}. En utilisant cet ensemble d'entrées ainsi que les équations pour la fonction OU-Exclusif qui régissent le comportement de S, on constate que la

seule possibilité pour S, sachant qu'elle était initialement à NULL, est de rester dans l'état NULL. Autrement dit, il y a une contradiction avec le fait que l'ensemble des sorties passe de NULL à DATA pour un ensemble d'entrées qui n'est pas de type DATA, donc la règle de complétude par rapport aux entrées est respectée. L'additionneur sur deux bits est alors représenté sur la Figure A4-5 où l'on peut distinguer les additionneurs avec et sans retenue d'entrée.

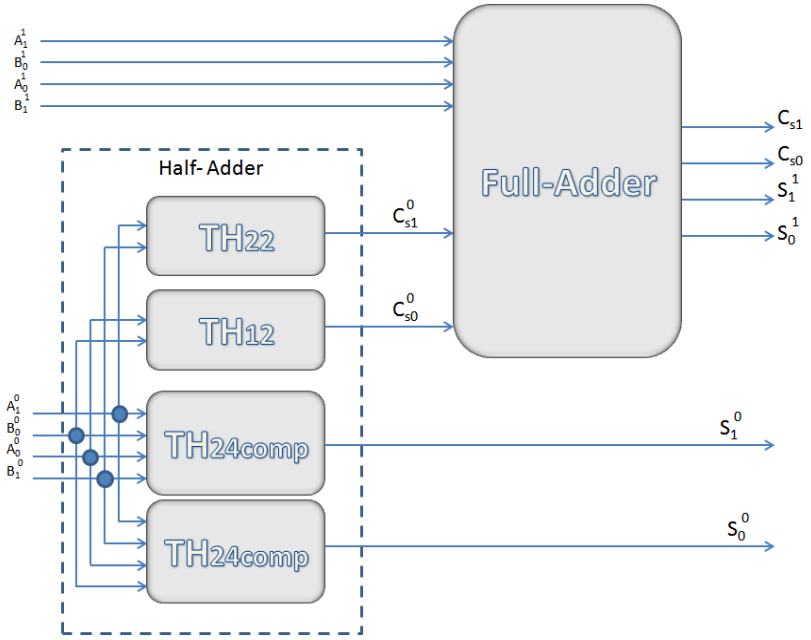


Figure A4-5. Additionneur NCL 2-bits

Calcul des sorties :

Finalement, le calcul des sorties se résume seulement à une collecte des sorties des différents modules présentés ci-dessus. Ceci est réalisé en récupérant directement des sorties comme pour le cas de la retenue de sortie, ou bien en combinant les sorties au travers d'une porte TH1N. Rappelons tout de même que dans ce cas particulier, afin de respecter la règle d'observabilité, une seule des entrées de ces portes de type TH1N est active pour un front donné.

Schéma général :

La Figure A4-6 propose un schéma général de l'unité arithmétique et logique discutée précédemment où, pour des raisons de lisibilité, les registres d'entrée et de sortie ont été omis. On y voit clairement chacun des modules présentés plus haut. Cette figure est inspirée de celle que l'on pourrait trouver dans [37].

Circuit d'incrémentation

Comme mentionné précédemment, la fonction principale de ce circuit est d'incrémenter un certain nombre d'une unité à chaque fois. Dans les circuits synchrones comme les microcontrôleurs usuels ceci peut être utile par exemple pour les *timers* ou les compteurs de boucle. Dans notre cas, ce circuit va nous permettre, comme le précédent, d'avoir un banc d'essai sur lequel faire des mesures. Notamment, comme on le verra dans le Chapitre 4, ce circuit nous permettra d'estimer le débit d'un tel module et l'impact sur celui-ci d'une éventuelle modification du NCL. De plus, cela nous permet également d'introduire les circuits séquentiels en logique asynchrone. Le circuit d'incrémentation peut se diviser en deux parties : tout d'abord une partie arithmétique chargée d'ajouter une unité au nombre précédemment calculé et un ensemble de registres permettant de mémoriser l'état du système.

Partie Arithmétique :

La partie arithmétique profite de ce qui a déjà été présenté plus haut. En effet, nous avons déjà détaillé la conception d'un additionneur N bits. On pourrait reprendre tel quel cet additionneur en fixant une des entrées à la valeur 1. Néanmoins, réécrire les équations pour un circuit d'incrémentation permet d'obtenir un circuit plus compact. En reprenant les équations du circuit d'incrémentation et en fixant par exemple l'entrée B à 0, on obtient :

$$S^l = A^l \oplus C_e^l$$
$$C_s{}^l = A^l . C_e^l$$

Dans ces équations *l* représente le bit considéré où *l = 0* est le bit de poids le plus faible. Ensuite il ne reste qu'à fixer la retenue d'entrée à 1. On en déduit qu'il est alors possible de réaliser l'élément de base du circuit d'incrémentation en utilisant des portes TH_{XOR0}, TH_{12} et TH_{22} comme représenté sur la Figure A4-7. Pour obtenir le circuit d'incrémentation complet il suffit de connecter entre eux N éléments de base à l'image de ce qui a déjà été fait pour l'additionneur.

Étage de mémorisation :

L'étage de mémorisation a deux fonctionnalités principales. La première est de permettre le bon fonctionnement des opérations en accord avec le protocole imposé par le NCL notamment l'alternance des fronts de types DATA et NULL. La seconde est d'imposer la valeur initiale du registre d'état. Pour ce faire, un minimum de trois registres est nécessaire. Ceux-ci sont agencés comme sur la Figure A4-8.

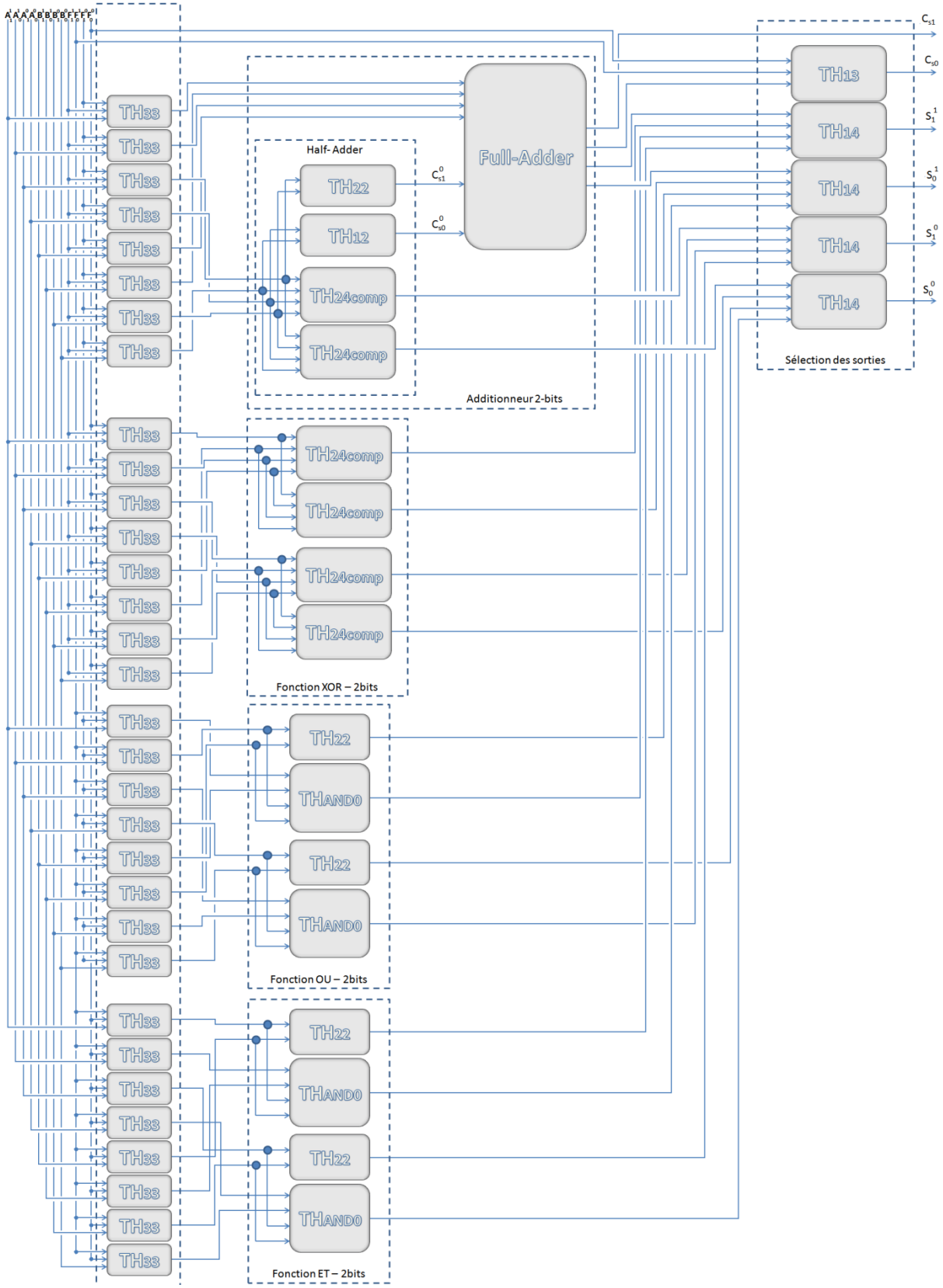


Figure A4-6. Exemple d'Unité Arithmétique et Logique en logique NCL

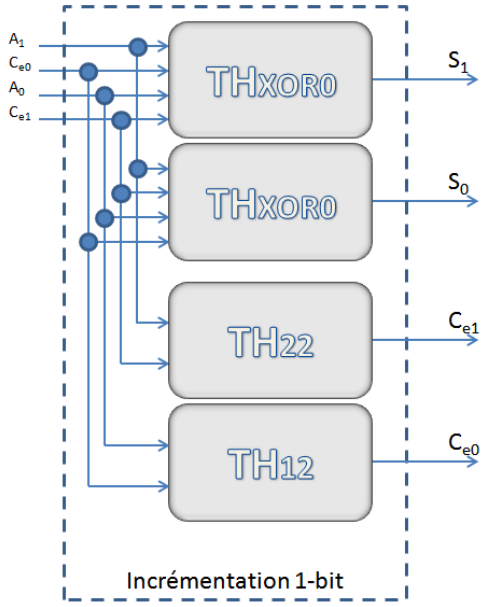


Figure A4-7. Circuit de base de l'incrémentation en logique NCL

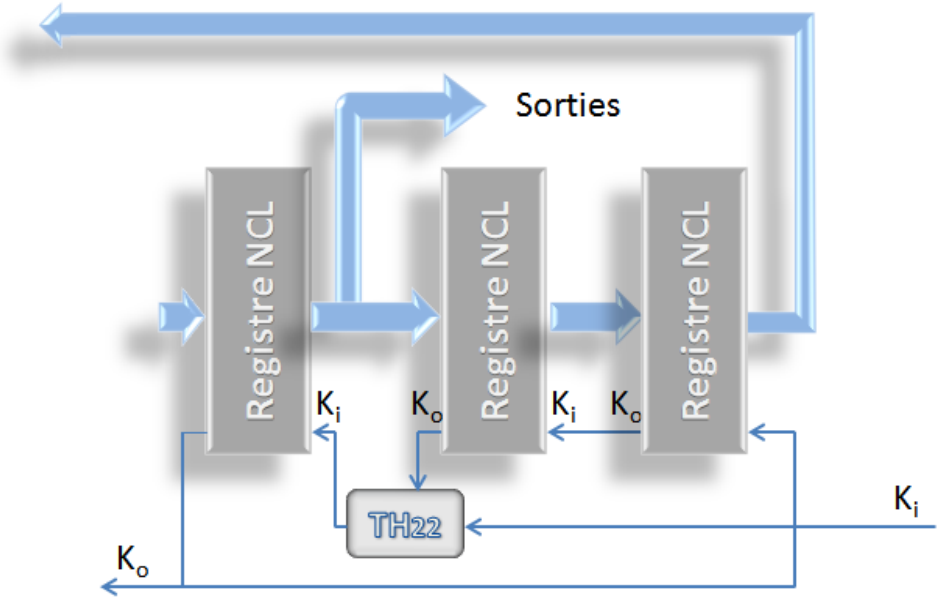


Figure A4-8. Étage de synchronisation des circuits séquentiels NCL

Les registres R1, R2 et R3 sont des registres de N bits identiques à ceux présentés dans le Chapitre 3. En revanche, le registre R2 a la particularité de prendre une valeur de type DATA lorsque le signal de remise à zéro du circuit (*reset*) est actif. Ceci nous permet de fixer la valeur de départ de notre circuit d'incrémentation, par exemple à zéro. L'entrée provient de la partie arithmétique et la sortie est dirigée vers un éventuel prochain étage du circuit. De plus, le signal K_o est le signal destiné à un éventuel étage précédent et K_i provient d'un éventuel étage suivant.

Ces deux signaux étant au cœur du protocole de communication proposé par le NCL qui a déjà été détaillé dans la sous-partie 3.4.3. En analysant chaque étape du protocole d'un tel circuit on constate que les données sont bien traitées dans l'ordre attendu, que le circuit attend que l'étage suivant soit prêt pour pouvoir calculer une nouvelle valeur et qu'il informe l'étage précédent de sa disponibilité à recevoir des calculs. Dans cet exemple, le circuit n'a pas réellement besoin d'avoir un étage précédent. En revanche pour un circuit dédié au traitement du signal par exemple, il est souvent utile de pouvoir réaliser des sommes dont le résultat est stocké dans un accumulateur. À ce moment là l'étage de registre stockerait la dernière valeur calculée, et la partie combinatoire aurait besoin, en plus de ce résultat, d'une autre valeur à ajouter par un étage précédent.

Schéma général :

Finalement pour réaliser le circuit d'incrémentation, il suffit de connecter les deux parties précédentes comme représenté sur la Figure A4-9 qui schématise l'ensemble des circuits séquentiels réalisé en respectant le protocole NCL.

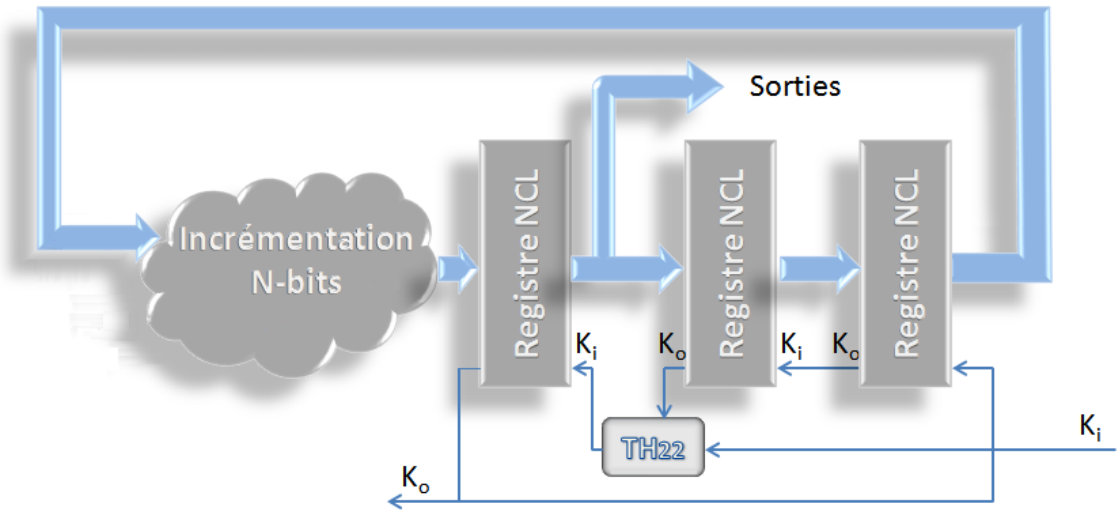


Figure A4-9. Circuit d'incrémentation N-bits en logique NCL

ANNEXE 5 - Délais dans les éléments de base du FPGA

Ici, nous recopions des extraits de la documentation concernant le FPGA Actel Igloo AGLN250V2 utilisé pour implémenter le module de réveil asynchrone proposé dans le Chapitre 5. Ces données sont particulièrement utiles pour l'analyse de la validité d'implémentation de la logique asynchrone.

Timing Characteristics

1.5 V DC Core Voltage

Table 2-168 **• Combinatorial Cell Propagation Delays**
Commercial-Case Conditions: T_J = 70°C, Worst-Case VCC = 1.425 V

Combinatorial Cell	Equation	Parameter	Std.	Units
INV	Y =!A	t_{PD}	0.80	ns
AND2	Y = A · B	t_{PD}	0.84	ns
NAND2	Y =!(A · B)	t_{PD}	0.90	ns
OR2	Y = A + B	t_{PD}	1.19	ns
NOR2	Y = !(A + B)	t_{PD}	1.10	ns
XOR2	Y = A ⊕ B	t_{PD}	1.37	ns
MAJ3	Y = MAJ(A, B, C)	t_{PD}	1.33	ns
XOR3	Y = A ⊕ B ⊕ C	t_{PD}	1.79	ns
MUX2	Y = A !S + B S	t_{PD}	1.48	ns
AND3	Y = A · B · C	t_{PD}	1.21	ns

Note: For specific junction temperature and voltage supply levels, refer to Table 2-6 on page 2-7 for derating values.

1.2 V DC Core Voltage

Table 2-169 **• Combinatorial Cell Propagation Delays**
Commercial-Case Conditions: T_J = 70°C, Worst-Case VCC = 1.14 V

Combinatorial Cell	Equation	Parameter	Std.	Units
INV	Y = !A	t_{PD}	1.34	ns
AND2	Y = A · B	t_{PD}	1.43	ns
NAND2	Y = !(A · B)	t_{PD}	1.59	ns
OR2	Y = A + B	t_{PD}	2.30	ns
NOR2	Y = !(A + B)	t_{PD}	2.07	ns
XOR2	Y = A ⊕ B	t_{PD}	2.46	ns
MAJ3	Y = MAJ(A, B, C)	t_{PD}	2.46	ns
XOR3	Y = A ⊕ B ⊕ C	t_{PD}	3.12	ns
MUX2	Y = A !S + B S	t_{PD}	2.83	ns
AND3	Y = A · B · C	t_{PD}	2.28	ns

Note: For specific junction temperature and voltage supply levels, refer to Table 2-7 on page 2-7 for derating values.

Image 1: Délais dans les éléments combinatoires

Timing Characteristics

1.5 V DC Core Voltage

Table 2-170 • Register Delays
Commercial-Case Conditions: T_J = 70°C, Worst-Case VCC = 1.425 V

Parameter	Description	Std.	Units
t_{CLKQ}	Clock-to-Q of the Core Register	0.89	ns
t_{SUD}	Data Setup Time for the Core Register	0.81	ns
t_{HD}	Data Hold Time for the Core Register	0.00	ns
t_{SUE}	Enable Setup Time for the Core Register	0.73	ns
t_{HE}	Enable Hold Time for the Core Register	0.00	ns
t_{CLR2Q}	Asynchronous Clear-to-Q of the Core Register	0.60	ns
t_{PRE2Q}	Asynchronous Preset-to-Q of the Core Register	0.62	ns
t_{REMCLR}	Asynchronous Clear Removal Time for the Core Register	0.00	ns
t_{RECCLR}	Asynchronous Clear Recovery Time for the Core Register	0.24	ns
t_{REMPRE}	Asynchronous Preset Removal Time for the Core Register	0.00	ns
t_{RECPRE}	Asynchronous Preset Recovery Time for the Core Register	0.23	ns
t_{WCLR}	Asynchronous Clear Minimum Pulse Width for the Core Register	0.30	ns
t_{WPRE}	Asynchronous Preset Minimum Pulse Width for the Core Register	0.30	ns
t_{CKMPWH}	Clock Minimum Pulse Width High for the Core Register	0.56	ns
t_{CKMPWL}	Clock Minimum Pulse Width Low for the Core Register	0.56	ns

Note: For specific junction temperature and voltage supply levels, refer to Table 2-6 on page 2-7 for derating values.

1.2 V DC Core Voltage

Table 2-171 • Register Delays
Commercial-Case Conditions: T_J = 70°C, Worst-Case VCC = 1.14 V

Parameter	Description	Std.	Units
t_{CLKQ}	Clock-to-Q of the Core Register	1.61	ns
t_{SUD}	Data Setup Time for the Core Register	1.17	ns
t_{HD}	Data Hold Time for the Core Register	0.00	ns
t_{SUE}	Enable Setup Time for the Core Register	1.29	ns
t_{HE}	Enable Hold Time for the Core Register	0.00	ns
t_{CLR2Q}	Asynchronous Clear-to-Q of the Core Register	0.87	ns
t_{PRE2Q}	Asynchronous Preset-to-Q of the Core Register	0.89	ns
t_{REMCLR}	Asynchronous Clear Removal Time for the Core Register	0.00	ns
t_{RECCLR}	Asynchronous Clear Recovery Time for the Core Register	0.24	ns
t_{REMPRE}	Asynchronous Preset Removal Time for the Core Register	0.00	ns
t_{RECPRE}	Asynchronous Preset Recovery Time for the Core Register	0.24	ns
t_{WCLR}	Asynchronous Clear Minimum Pulse Width for the Core Register	0.46	ns
t_{WPRE}	Asynchronous Preset Minimum Pulse Width for the Core Register	0.46	ns
t_{CKMPWH}	Clock Minimum Pulse Width High for the Core Register	0.95	ns
t_{CKMPWL}	Clock Minimum Pulse Width Low for the Core Register	0.95	ns

Note: For specific junction temperature and voltage supply levels, refer to Table 2-7 on page 2-7 for derating values.

Image 2. Délais dans les éléments séquentiels

ANNEXE 6 - Dessins des masques des PCB

Les images suivantes représentent le dessin des masques réalisés avec le logiciel Design Spark pour les différentes parties de la plateforme : le module de réception USB, la carte de l'étage du bas de l'*animat* et celle du haut.

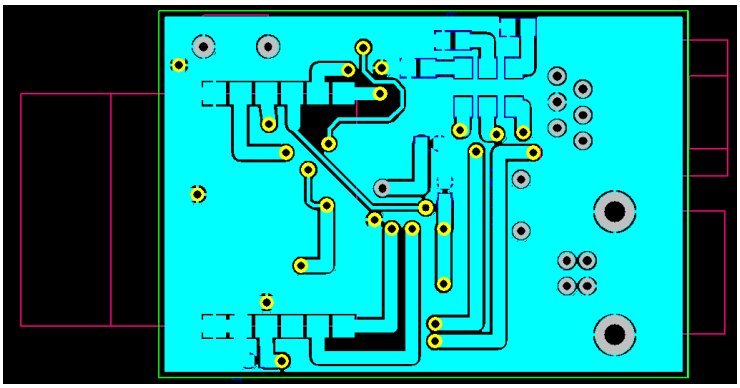


Image 3. PCB « bottom » du coordinateur

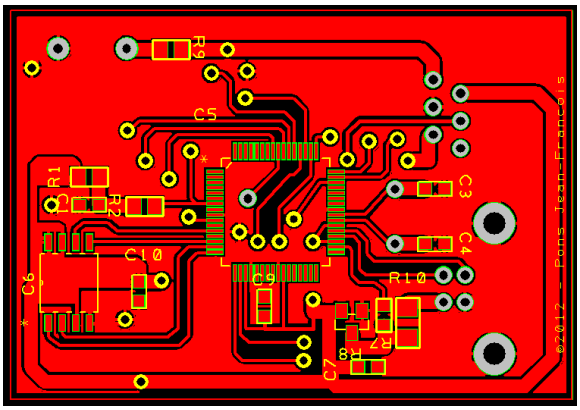


Image 4. PCB « top » du coordinateur

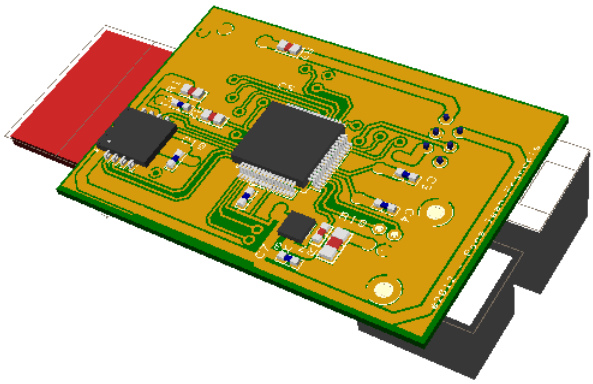


Image 5. Vue 3D du coordinateur

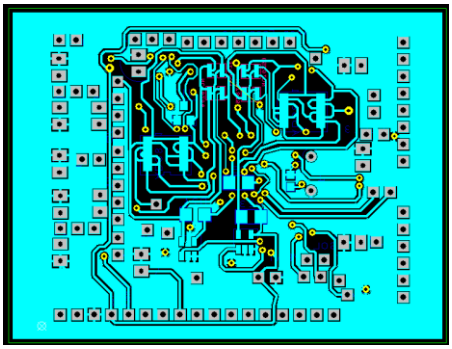


Image 6. PCB « bottom » de l'étage du bas de l'*animat*

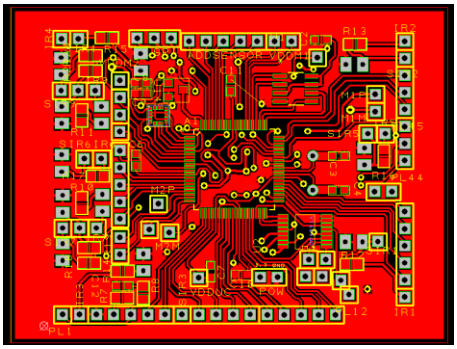


Image 7. PCB « top » de l'étage du bas de l'*animat*

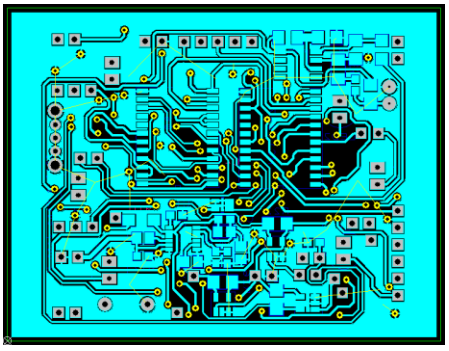


Image 8. PCB « bottom » de l'étage du haut de l'*animat*

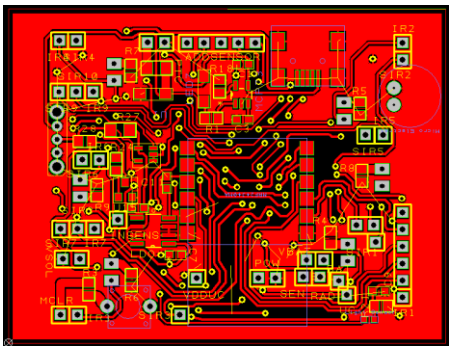


Image 9. PCB « top » de l'étage du haut de l'*animat*

ANNEXE 7 - Photos des PCB réalisés

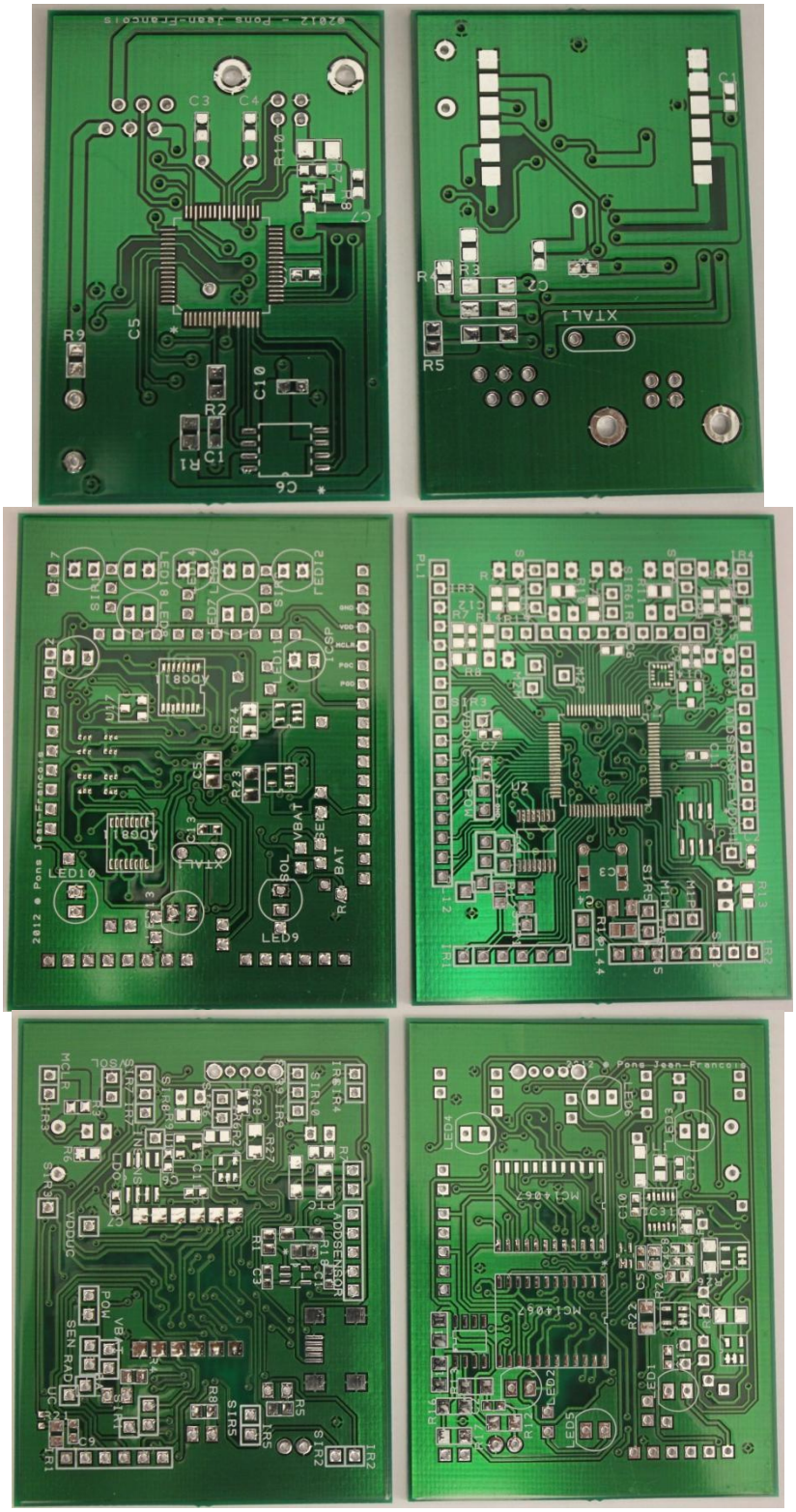


Image 10. Photos des PCB: Coordinateur, *Animat* bas, *Animat* haut

ANNEXE 8 - Conception de la coque de l'*Animat*

Dans l'optique d'augmenter la robustesse de l'*animat*, nous proposons une coque enveloppant les parties électroniques. Celle-ci a été conçue à l'aide du logiciel de CAO SolidWorks puis a été réalisée grâce à une imprimante 3D disponible dans les laboratoires de recherche de l'Ecole Polytechnique de Montréal. Ci-dessous, nous représentons la vue 3D de cette coque ainsi que quelques photos du produit réalisé. On notera que le matériau utilisé par l'imprimante 3D a une masse volumique élevée et que la masse de la coque pourrait être diminué en utilisant un plastique plus léger.

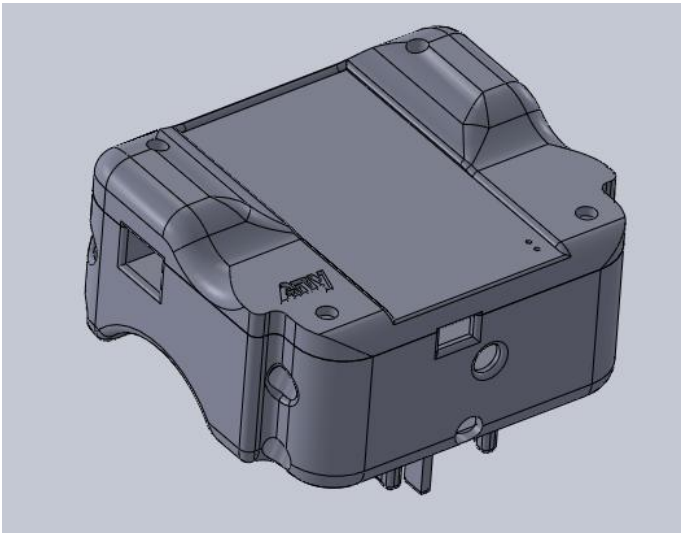


Image 11. Représentation 3D de la coque de l'*animat*

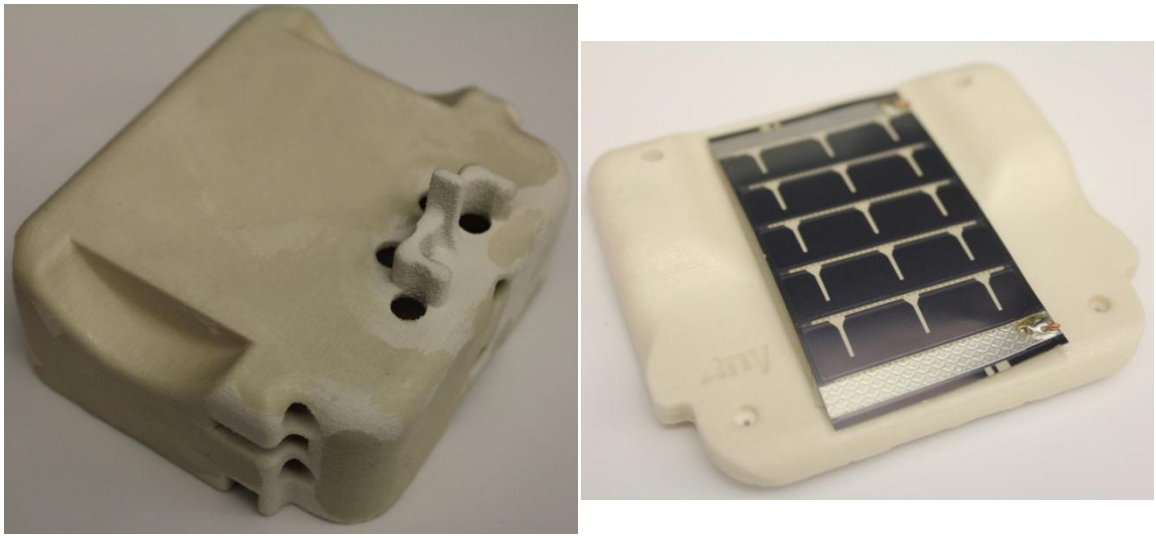


Image 12. Réalisation des deux parties de la coque et ajout du panneau solaire

ANNEXE 9 - Photos de l'*Animat*

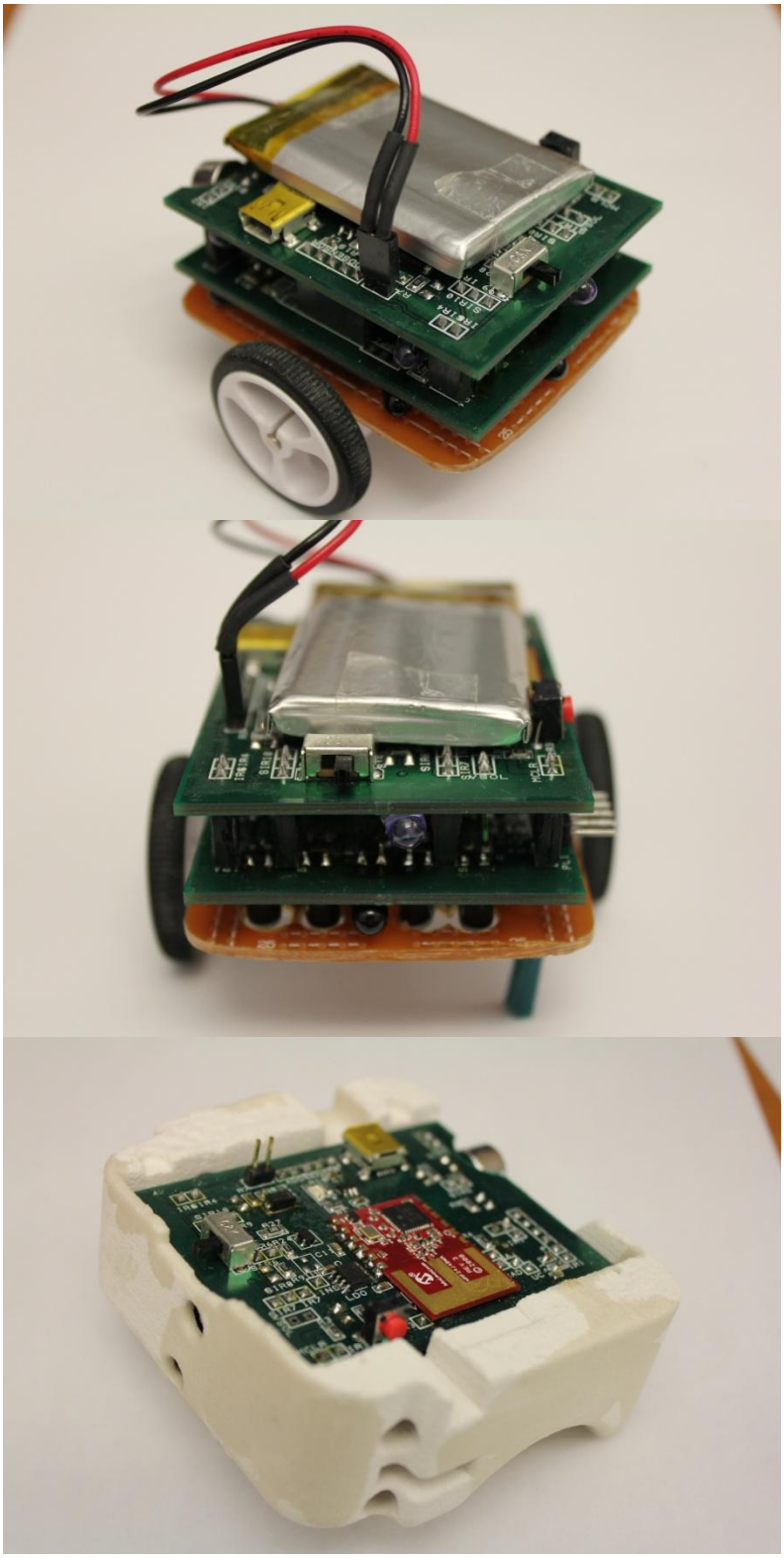


Image 13. Différentes photos de l'*Animat*

ANNEXE 10 - Interface graphique

L'interface se décompose en deux fenêtres principales : celle pour connecter le périphérique USB et celle pour l'interaction avec les *animats*. La première apparaît à l'initialisation puis à chaque fois que le périphérique USB et déconnecté. Elle permet trois modes de connexion distincts et est représentée sur l'Image 14.

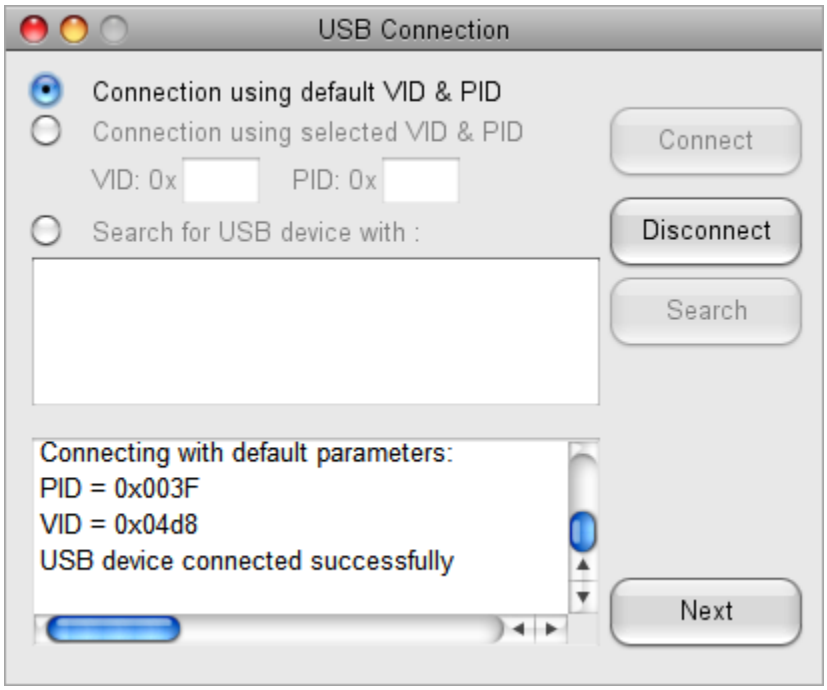


Image 14. Fenêtre de connexion USB

La seconde est destinée, comme susmentionné, à l'interaction générale avec les *animats*. Celle-ci se décompose en quatre parties : une console pour informer l'utilisateur, un arbre permettant la visualisation du réseau, une partie de description de l'*animat* sélectionné et un panneau de contrôle. Ce dernier contient trois onglets permettant respectivement des communications manuelles, de prendre le contrôle des moteurs de l'*animat* et d'interagir avec le logiciel Matlab. La décomposition de l'interface ainsi que ses fonctionnalités n'est qu'une suggestion permettant de profiter de manière intuitive de certaines des fonctionnalités de la plateforme. Sur l'Image 15, le lecteur pourra trouver l'aspect générale de cette fenêtre ainsi que des zooms sur les deux onglets de commande et d'interaction avec Matlab.

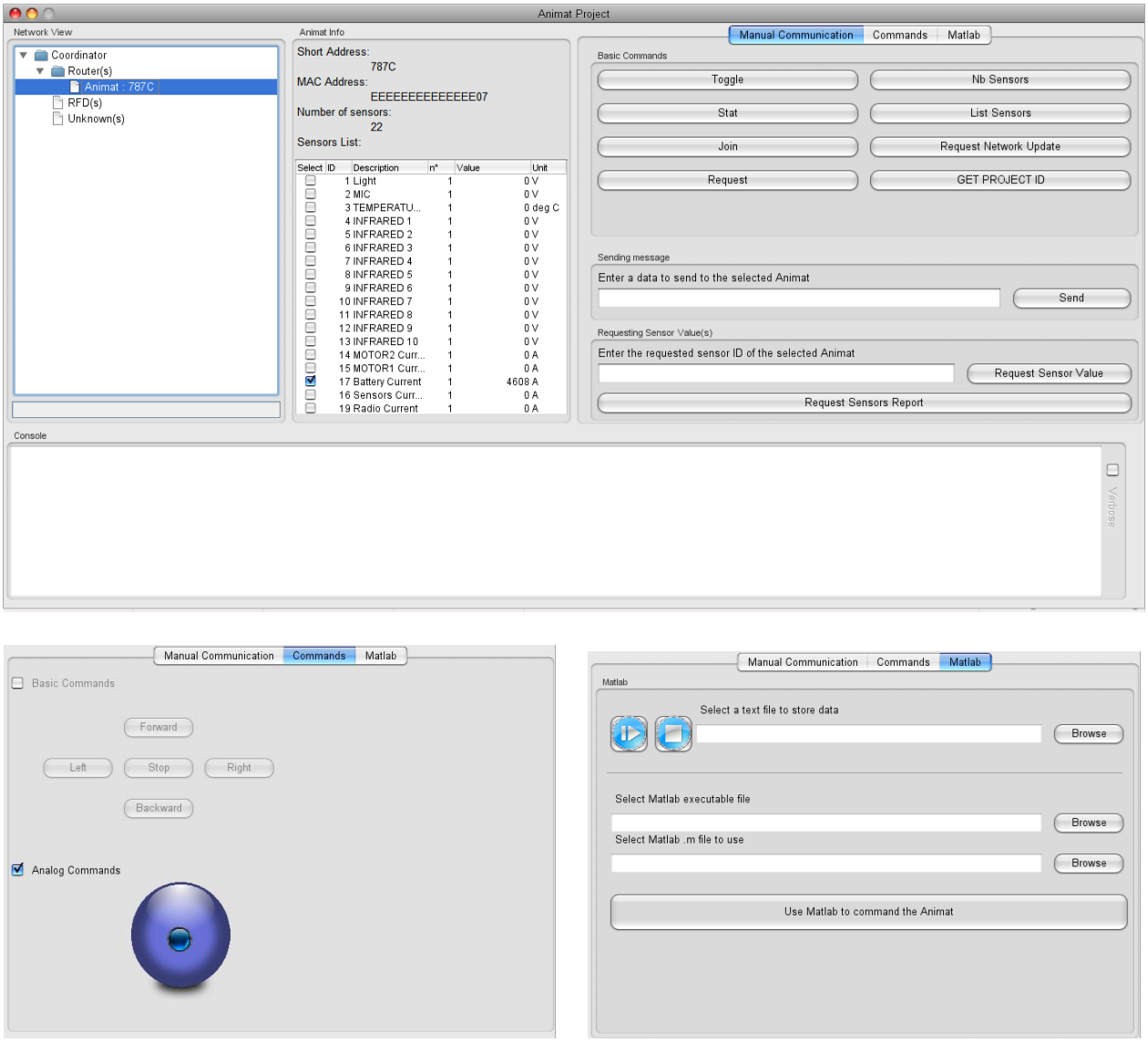


Image 15. Interface d'interaction avec les *animats* et zoom sur les onglets

ANNEXE 11 - Article NEWCAS 2012

An FPGA Compatible Asynchronous Wake-Up Receiver for Wireless Sensor Networks

Jean-François Pons, Jean-Jules Brault and Yvon Savaria
Department of Electrical Engineering
Ecole Polytechnique de Montreal
Montreal, Qc, Canada
{jean-francois.pons, jean-jules.brault, yvon.savaria}@polymtl.ca

***Abstract*— This paper explores design methods applicable to Wireless Sensors Networks, where low power consumption and energy efficiency are a must. A key component that modulates the power consumption is the main radio. Controlling its use through suitable sleep modes and wake up mechanisms is a significant issue and can be done with a wake-up receiver. But many applications are associated with low fabrication volume where custom integrated circuits are not economical and where FPGAs are the best available solution. In this paper, we explore an asynchronous solution, which permits to decrease the internal activity, thus reducing the power consumption, including that required for clock distribution. We also propose an FPGA implementation of such a wake-up receiver using the NULL Convention Logic™. The overall power consumption of the reported implementation is as low as 5µW at 250 kbps.**

I. INTRODUCTION

During the past decade, Wireless Sensor Networks (WSN) have been an important research area, especially with the development of low power devices using protocols such as ZigBee® [1]. A major concern with WSN is how long they can survive with limited power or energy. In many possible applications, the device needs to be active with a low to very low duty cycle. When the duty cycle needed by the application is very low, the overall power consumption is easily dominated by wireless communications. To address the wireless consumption problem, a common solution is to use periodic activation of the transceiver. Choosing an appropriate duty-cycle is difficult, because of its application-dependent nature and of the tradeoffs between various parameters linked to the waking-up frequency, such as communication latency and energy consumption.

One possible way to address this problem is to use a wake-up receiver (WUR), which can be used to trigger activation of the main radio by listening the wireless communication channel and generating an interrupt when the module needs to be woken-up with very low power consumption. We are assuming that the target application is such that the occurrence probability of a wake-up signal is low. This can be the case in WSN security applications or when the communication is event-driven (e.g. detection of an intrusion, etc.). In this context, the use of an always active wake-up circuit, with its associated energy overhead, can be legitimate as explained in [2]. Several WUR designs can be found in the literature: using low voltage down conversion mixer [3] or envelope detector coupled with pattern filter [4], Pulse-Width Modulation demodulator and comparator [5], matched filter and address correlation lines [6] or correlation receiver [7]. For fully-integrated, low duty-cycle and low volume application, these solutions are not well-suited, especially due to large NRE costs of Application-Specific Integrated Circuits (ASICs) and always-on clock of synchronous systems. Therefore, we propose in this paper an asynchronous wake-up receiver design implemented using NULL Convention Logic™ on an Actel AGLN250V2 FPGA. This solution contrasts with the high NRE costs ASIC reported in [3-7]. To our knowledge, the concept of asynchronous wake up receiver was never reported. This module finds its use in the architecture of WSN nodes as depicted in figure 1.

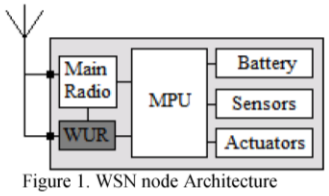


Figure 1. WSN node Architecture

The rest of the paper is organized as follows: in Section II, general considerations are discussed. Then in Section III, a brief description of the NULL Convention Logic™ (NCL) [8,9] is given and the overall architecture of the proposed WUR is described. Simulated results are presented and discussed respectively in Sections IV and V. Finally, a conclusion is given in Section VI.

II. BACKGROUND INFORMATION

In this section, an energy model is discussed and the NULL Convention Logic™ is reviewed.

A. Energy Model

Neglecting the energy consumed by actuators, which are often absent in WSNs, (1) can be used for energy model.

$$E_{tot} = E_{sens} + E_{MPU} + E_{sleep} + E_{TX} + E_{RX} + E_{nocom} + E_{WUR} \quad (1)$$

Where E_{sens}, E_{MPU}, E_{sleep}, E_{TX}, E_{RX}, E_{nocom}, and E_{WUR} respectively refer to the energy used by the sensors, the microcontroller (MPU) in normal mode, or in sleep-mode, the main radio while transmitting, receiving or not communicating, and the WUR. Note that some of the energy

components are consumed in mutually exclusive modes, while some of the modes are active simultaneously. In this paper we will consider that (1) governs the overall energy although a more detailed study of several wake-up radio schemes can be found in [2]. Thereafter, when average power consumption is reported, it refers to the energy consumed by the module considered over some significant time interval.

B. NULL Convention Logic™ (NCL)

Two main methodologies are possible to design asynchronous circuits: bounded-delay or delay insensitive design. The first one requires hard timing analysis to determine delays in all the combinational parts between two stages of some micro-pipeline. In that case, delays need to be added to the acknowledgment feedback. Delay insensitive techniques use a specific logic and some particular data representation, which, under certain conditions, give valid results by construction. We choose to use delay insensitive design for two reasons: it decreases the design time and it avoids the necessity of generating precise or bounded delays on FPGAs. Among the delay-insensitive techniques, NCL is a paradigm that establishes a well-defined structure for designers. NCL uses either dual-rail or quad-rail logic, and we use the former that has the lowest complexity. This paradigm is based on two main rules: input completion and observability. Those rules, as well as the whole supporting theory, can be found in [8, 9]. NCL defines 27 basic gates from which all designs can be derived. Even if those gates were initially designed for ASIC implementations, simulations in worst case conditions and experiments on the AGLN250V2 FPGA allowed us to use this methodology for FPGA implementation while ensuring a glitch free operation.

III. PROPOSED ARCHITECTURE

In this section, advantages of an asynchronous wake up receiver are exposed, a wake-up protocol is described, and the overall architecture of the asynchronous wake-up receiver is presented, as well as the general behavior of its components. The main objectives of the design were obtaining low power consumption together with user configurability and high reliability with respect to false wake-up alarms.

A. Asynchronous Design

For synchronous designs, an independence assumption between E_{WUR} and the activity of the wake-up signal can be made. Here we propose an asynchronous design leading to two main benefits. Obviously, substantial gain is obtained as it is no longer necessary to distribute a clock signal over all the circuit. Moreover, for such designs, dynamic consumption is only dependent on the activity of the input. In other words, when there is no transition on the incoming wake-up signal, the power consumption is only due to the static component as shown in Section IV. This is the main reason why we choose this approach. As explained in Section II, this feature can easily be obtained by following the NCL designing rules.

B. Wake-Up Signal Protocol and Inputs Assumptions

In this work, it is assumed that the wake-up signal is modulated using *On-Off Keying* (OOK). This avoids the use of complex demodulation since it only requires an envelope detector configured for a 2.4GHz carrier frequency. This contributes to reduce the WUR power consumption. Consequently, to generate wake-up messages, we choose to use an existing ZigBee® transceiver exploited in an OOK mode. In that mode, '0' and '1' symbols have different durations. For a good discrimination between symbols, we chose to represent '0' as a pulse of duration T and '1' as a pulse of duration 2T, with each pulse separated by an inactive period of duration T. Messages are organized as frames that are separated by inter-frame gaps of duration 5T. Then, in order to reduce the false alarm wake-up rate, we choose to use directly the address of the ZigBee® protocol preceded by an 8-bit preamble and a selector field. This selector indicates the size of the following address: 16-bit long for the short address of the node and the "all in the network address", and 64-bit long for the extended address of the node. Short and extended addresses are defined by ZigBee®. Figure 2 summarizes the wake-up frame format. As explained next, these addresses are user configurable. Thus, it is possible to change the "all in the network" one to select a group of nodes. In Section IV, several values of the RF signal baseband toggle rate are reported as the design can operate for diverse toggle rates or data rates. Considering the frame format and the timing considerations of the OOK modulation, Table I shows some specific worst case values imposed for simulations.

TABLE I. VALUES USED FOR SIMULATIONS

Fixed Values	Implied Worst Case Values	Condition
Data rate min = 250 kbps	Latency max = 293 µs	Only '1' 64-bit address
	Toggle rate max = 724 kHz	Only '0' 64-bit address
Latency max = 10 ms	Latency max = 10 ms	Only '1' 64-bit address
	Toggle rate max = 22 kHz	Only '0' 64-bit address

C. Architecture

The overall architecture of the WUR design is presented in figure 3 where internal components are partitioned into two parts: the front end and the comparator. Inside the first one, the "Front End To NCL" module converts Boolean data into dual rail representation. It also manages the I/O used for demodulation. The timing events necessary to decode the wake-up message are derived from RC circuits or delay lines. The values of those discrete components have been carefully selected in order to reduce power consumption and to allow full integration in case an ASIC implementation would be considered. The decoder is a Finite State Machine that manages this charging/discharging process and deduces the effective value received, '0' or '1', if any. The comparator part is then activated, only if a symbol has been found, otherwise it stays in a static state. Data provided by the front end is then compared with reference addresses. The technique used here is to compare in parallel new received data with the three reference addresses defined earlier (long, short and "all in the network"), one bit of the message each time. A counter is used to know whether or not the selector bit is treated, the end of the frame is detected or if the frame meets the standard predefined structure. A third part is used only for configuration of the reference addresses. To reduce the

resources used by this part, we only implement a simplified slave SPI controller module.

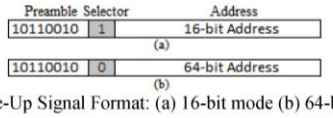


Figure 2. Wake-Up Signal Format: (a) 16-bit mode (b) 64-bit mode

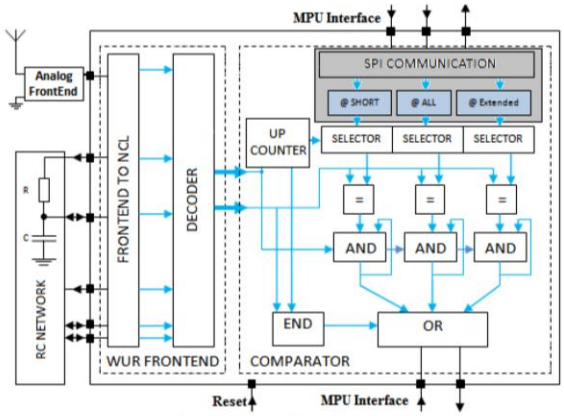


Figure 3. Overall Architecture

Broadly speaking, it only serves to receive data when an external Chip Select (CS) pin is driven low and an external clock is provided. This is the only synchronous part of the design, that it is merely used for configuration and usually never after (with clock gated off). When the configuration is processed, the rest of the circuit is in its static reset mode.

IV. RESULTS

In this section, results from simulations are reported. These results characterize resource utilization and power consumption for the target FPGA technology.

A. Resources Utilization

The targeted FPGA is the Actel AGLN250V2. This FPGA is a low power device and it provides enough logic resources to implement our design. Table II summarizes the resource usage evaluated by the synthesis tool.

B. Power Consumption of the WUR

Considering the assumptions made in Section III, and knowing the placement and routing (P&R) of the resources on the FPGA, power simulations can be done with the SmartPower tool provided by Actel. Several simulation results are summarized in Table III for the configuration mode and for the normal mode, where reception of a continuous stream of valid wake-up messages is assumed. In configuration mode, a 32 kHz clock frequency is assumed. A main benefit of the proposed WUR is its very low dynamic power consumption. Our results made it clear that even with this best in class low power FPGA, static power becomes the main source of energy consumption with the reported implementation. We thus explored how it could be reduced by reducing the power supply voltage. In experiments conducted at 23.7°C, it was observed that when the supply voltage is decreased from 1.2V to 850 mV, the static consumption is considerably reduced from 30 µW to 5µW. To be complete, the results should take into account the power consumption of the RC network estimated below 2.5nW. Also, our implementation does not include the analog front end. The solution proposed in [4] would add 2µW power consumption.

C. Variations of Dynamic Power

Table III reports power consumption of the WUR if wake-up messages are sent continuously. This is rather pessimistic. In the envisioned applications, those messages would be emitted whenever significant data have been collected. Therefore the frequency of wake-up occurrence, noted f_{WU}, is assumed to be low, figure 4 shows the variation of the dynamic power consumption for several values of f_{WU} at a 250 kbps data rate for 1.2V supply voltage.

D. System Consumption

The average power consumption of the complete system is then evaluated considering that:

- The main radio controller is the MRF24J40MA;
- The MPU is an eXtreme Low Power PIC®, which consumes only tens of nanoamps in sleep mode.

Using reported results and (1), figure 5 shows estimates of the overall consumption for several values of f_{WU} when the supply voltage is assumed to be 1.2V. These estimates take into account the use of deep sleep modes for components when available. The battery assumed in these calculations is a Li-Ion battery with a 400 mAh capacity.

V. DISCUSSION

A. About The Results

According to figure 4, we can infer that using asynchronous design leads to a solution where power consumption is mainly limited by the static power when low f_{WU} is assumed, which is in contrast with synchronous designs. We chose to use an FPGA in this work as it avoids the high NREs and fabrication delays associated with ASICs. However, using this technology introduces significant static power consumption. Therefore, even if the benefit of an asynchronous design is clear, to take full advantage of it, the leakage could be lowered by designing an ASIC in an older technology, especially because of the possibility to suppress unused gates (which is not possible with FPGAs). Nevertheless the reported design could be functional for more than 7 years when operating from a 400 mAh battery smaller than usual AA or AAA batteries from 850mV supply voltage.

B. About Other WUR

Several WUR designs can be found in the literature. However, to our knowledge, none of them are purely asynchronous or entirely designed on a FPGA. Therefore the comparison is done with similar solutions in terms of functionality. Table IV summarizes these other solutions and their main features. Technology as well as design type (asynchronous (A) or synchronous (S)) are reported. The power dissipation reported for competing designs range from2 to 50 µW. Depending on the supply voltage, our solution could consume as little as 5 µW.

TABLE II. RESOURCES REPORT

	DESCRIPTION
FPGA	IGLOO AGL250V2 1.2V Package: 100 VQFP
CORE	4275 (69.58%): Combinational = 2728, Sequential = 1547
I/O	15 (22.06%): Inputs = 6, Output = 4, Bidirectional = 5

TABLE III. CONSUMPTION IN THE TWO MODES

Mode	Frequency or Toggle Rate	SmartPower Values (µW) [at 25°C]		
		Static Power (at 1.2V)	*Dynamic Power*	*Total*
Configuration	32 kHz	29.8	3.5	**33.3**
	1 MHz		90.0	**119.7**
	2 MHz		179.2	**208.9**
Normal	22 kHz		22.5	**52.3**
	724 kHz		776.9	**806.7**

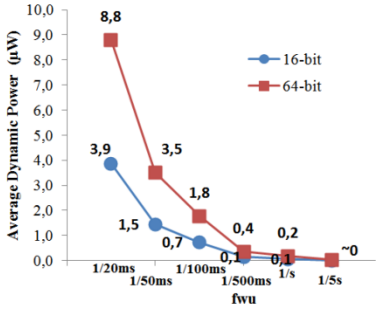


Figure 4. Variations of the Average Dynamic Power at 1.2V

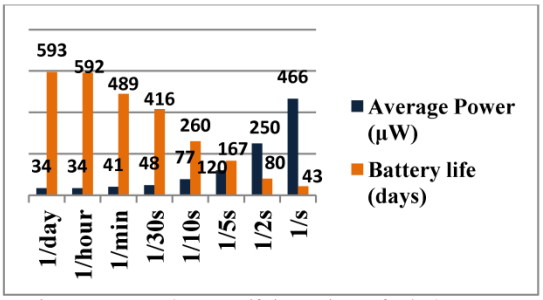


Figure 5. Power and Battery Lifetime Estimates for the System

Again, the main advantage of using an FPGA is the reduction of the NRE costs for small production and the flexibility for future modifications.

Moreover, functionality of asynchronous designs is not affected much by small to medium variations in supply voltage. Therefore, voltage regulator consumption can be decreased. Also, asynchronous designs do not switch when inputs are stable, unlike synchronous systems.

CONCLUSION

In this paper, we first presented the architecture of an asynchronous Wake-Up Receiver. We showed the benefits of using asynchronous design for such "always-on" event-driven application. For doing so, we proposed and implemented on FPGA a solution that exploits NULL Convention Logic™. Then we estimated the power consumed in several situations where wake-up signals happen not too often. When wake up frequency is very low, the power consumption is around 30 µW if the circuit operates from a standard 1.2 V supply, but that power could be lowered to 5µW if the supply is reduced to 850 mV. This work demonstrates that FPGAs can be used for ultra low power applications leading to autonomy of several years when operating from a standard AA battery.

REFERENCES

[1] 802.15.4-2003 IEEE Standard for Information Technology-Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) specifications for Low Rate WI:reless Personal Area Networks (LR-WPANS), 2003

[2] Z. Yan, *et al.*, "An analytical model for energy efficiency analysis of different wakeup radio schemes," in *Personal, Indoor and Mobile Radio Communications, 2009 IEEE 20th International Symposium on*, 2009, pp. 1148-1152.

[3] N. M. Fletcher, *et al.*, "A 2GHz 52µW wake-up receiver with -72dBm sensitivity using uncertain-IF architecture," in *2008 IEEE International Solid State Circuits Conference, ISSCC, Febrary 3, 2008 - Febrary 7, 2008*, San Francisco, CA, United states, 2008, pp. 524-525+633+521.

[4] C. Hambeck, *et al.*, "A 2.4µW Wake-up Receiver for wireless sensor nodes with -71dBm sensitivity," in *Circuits and Systems (ISCAS), 2011 IEEE International Symposium on*, 2011, pp. 534-537.

[5] P. Le-Huy and S. Roy, "Low-Power 2.4 GHz Wake-Up Radio for Wireless Sensor Networks," in *Networking and Communications, 2008. WIMOB '08. IEEE International Conference on Wireless and Mobile Computing*, 2008, pp. 13-18.

[6] Z. Yan, *et al.*, "A 3.72µW ultra-low power digital baseband for wake-up radios," in *VLSI Design, Automation and Test (VLSI-DAT), 2011 International Symposium on*, 2011, pp. 1-4.

[7] M. S. Durante and S. Mahlknecht, "An Ultra Low Power Wakeup Receiver for Wireless Sensor Nodes," in *Sensor Technologies and Applications, 2009. SENSORCOMM '09. Third International Conference on*, 2009, pp. 167-170.

[8] K. M. Fant and S. A. Brandt, "NULL Convention Logic™: a complete and consistent logic for asynchronous digital circuit synthesis," in *Application Specific Systems, Architectures and Processors, 1996. ASAP 96. Proceedings of International Conference on*, 1996, pp. 261-273.

[9] S. C. Smith and J. Di, *Designing asynchronous circuits using NULL convention logic (NCL)*: Morgan & Claypool Publishers, 2009.

TABLE IV. PERFORMANCE COMPARISON

Reference	Type of Work	Technology	Type	Power (µW)	Data rate (kbps)	Sensitivity (dBm)	Carrier Frequency	Supply Voltage (V)	Address	Configurable
This Work	Simulation Digital Baseband	FPGA *AGLN250V2* CMOS 130 nm	A	5 - 30	250	-	2.4 GHz	0.85 - 1.2	16 bits 64 bits	Yes
[3]	Realization RF Front End	ASIC CMOS 90 nm	S	52	50-200	- 72	2 GHz	0.5	-	-
[4]	Realization Complete	ASIC CMOS 130 nm	S	2.4	100	- 71	868 MHz	1	64 bits	Yes
[5]	Simulation Complete	ASIC CMOS 130 nm	S	~ 19	50	~ -50	2.4 GHz	1	8 bits	-
[6]	Partial Realization Digital Baseband	ASIC CMOS 90 nm	S	< 3.72	> 200	-18	-	1.2	8 bits	-
[7]	Realization Complete	Mixed ASIC and FPGA CMOS 120 nm	S	12.5	100	~ -55	2.4 GHz	1.5	-	-

ANNEXE 12 - Article MWSCAS 2012

State-Holding Free NULL Convention Logic™

Jean-François Pons, Jean-Jules Brault and Yvon Savaria
Department of Electrical Engineering
Ecole Polytechnique de Montreal
Montreal, Qc, Canada
{jean-francois.pons, jean-jules.brault, yvon.savaria}@polymtl.ca

***Abstract*—For several applications, such as those with low duty-cycle or event-driven activity, for which power consumption is a concern, asynchronous circuit design is very appealing. The main advantages are decreased power consumption and reduced electromagnetic interference (EMI). The NULL Convention Logic™ (NCL) paradigm provides an interesting way to design such circuits. This paper demonstrates that resources usage could be diminished by removing the state-holding capability of the 27 NCL gates. In this paper, we propose and discuss an effective mean to obtain such reduced complexity circuits. The modified protocol proposed and validated in this paper leads to a substantial reduction, as high as 49%, of the resources required in a reported example. Moreover by decreasing the number of gates, we observed that energy efficiency as well as operating frequency could be improved.**

I. INTRODUCTION

With the scaling of transistor dimensions and the difficulty to propagate clock signals through an entire circuit without suffering from excessive skew problems, microelectronic designers resort more and more to Globally Asynchronous Locally Synchronous (GALS) or totally asynchronous solutions. The main benefits targeted are improved data integrity and system reliability without extra efforts and power consumption for propagating the control signals. On the other hand, some designers choose asynchronous designs especially when power consumption is a major concern. For example, this is the case in embedded systems or in Wireless Sensor Networks (WSN), for which the internal activity is environment dependent or has a low duty-cycle.

In this context, asynchronous design methodologies have emerged. They can be sorted in two main categories: bounded-delay and delay-insensitive. The second category could be divided in several sub-categories such as pure delay-insensitive (DI), quasi-delay insensitive (QDI) and speed-independent circuits (SI). In this paper, when delay-insensitive circuits are discussed, we refer to QDI circuits, given that pure DI circuits are very restricted as explained in [1]. Bounded-delay designs such as micropipelines [2] assume that delays in both gate and wire are bounded. Delays in the worst case scenario are then added in the handshake protocol to indicate when data can be considered valid. This leads to extensive timing analysis of worst-case behaviors to ensure correct circuit operation. By contrast, with QDI circuits, delays can be unbounded. Therefore, to ensure information integrity, special data representation including the validity of the result must be adopted: for example *dual rail* or *quad rail* representations are popular solutions. Among existing QDI methods [3-5], NULL Convention Logic™ (NCL) [6,7] exhibits the possibility to optimize Boolean expressions and does not require generation of all minterms as we will see in Section II.

Some research aimed at improving NCL by using *Threshold Combinational Reduction* [8] or *Cutoff transistors* [9]. The first one is a software solution allowing successive optimizations. The second one is an improvement at the transistor level to decrease the power consumption. In both cases, the solution starts from the NCL paradigm without trying to change it. In this paper, a modification of the NCL design rules is proposed and it could benefit from the improvements presented above. The main difference is that here a modification of the NCL handshake protocol is proposed in order to decrease the resources used. To the knowledge of the authors such a modification of the NCL handshake protocol was never reported.

The rest of the paper is organized as follows: in Section II, a brief description of the NULL Convention Logic™ (NCL) [6,7] is given. Then in Section III, the proposed simplification is described. Some results are presented and discussed respectively in Sections IV and V. Finally, a conclusion is given in Section VI.

II. NULL CONVENTION LOGIC™

In this section, a brief summary of the NCL is given to help understand the impacts of the modification proposed to the handshake protocol.

A. General Considerations

NCL defines a structure for designing QDI asynchronous circuits. It is based on two basic rules and an assumption sufficient to design reliable asynchronous circuits:

Rule 1: Input-completeness

"Input-Completeness requires that all outputs of a combinational circuit may not transition from NULL to DATA until all inputs have transitioned from NULL to DATA, and that all outputs of a combinational circuit may not transition from DATA to NULL until all inputs have transitioned from DATA to NULL" [7].

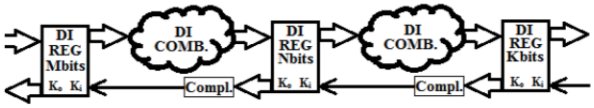


Figure 1. NCL Framework

Rule 2: Observability

"Observability requires that no *orphans* may propagate through a gate. An orphan is defined as a wire that transitions during the current DATA wavefront but that is not used in the determination of the output" (no sensitized path) [7].

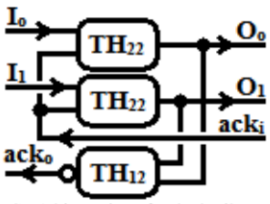


Figure 2. NCL 1-bit Register for dual rail representation

Assumption: Isochronic fork

Within basic components, if a transition happens on one end of a fork and this transition has been acknowledged, it is assumed that all the transitions on the outputs of the others branches of the fork have also happened and have been acknowledged when relevant.

B. Basic elements and NCL Framework

The NCL design method defines a set of 27 gates implementing the set of all functions of four or fewer variables [7]. It is important to notice that these gates have a state-holding capability. Therefore the input-completeness with regard to a NULL front is respected at a local level. Indeed, for the output of a NCL gate to transition from 1 to 0, all its inputs need to be 0 ensuring the overall input-completeness with regard to a NULL front of the overall combinational stage.

A conventional NCL system is composed of delay-insensitive logic sandwiched between two asynchronous registers as shown in Fig. 1. The DI logic is built from the 27 gates discussed above. Therefore the set of input signals has to follow a DATA/NULL alternation for the validity to be checked and the outputs to be deasserted. Each register consists of a set of 1-bit registers, depicted in Fig. 2, in parallel. For the output to be set, both the corresponding inputs and the ack_i signal have to be set. For the output to be cleared both the corresponding inputs and the ack_i signal have to be cleared. The TH_{22} gates used here are equivalent to a Müller C-element and the complemented TH_{12} gate is equivalent to a NOR gate. The role of such registers is to stop the data flow until the next stage is ready either to receive a DATA or NULL front. For a stage to decide if it is ready to receive such a front, the completion of the inputs received is processed and transmitted back to the previous stage through the K_o/K_i protocol.

III. Proposed State-Holding Free NCL

In this section, we present how the improved State-Holding Free NCL (SHF-NCL) emerged and explain what kind of gains could be expected. Moreover, feasibility and requirements of SHF-NCL are analyzed. In the rest of the discussion we will only use dual rail representation even if the proposed work fit with all kind of complete representation.

A. Initial Observations and Expected Gains

In our recent research, we used the NCL paradigm on an FPGA platform. We observed that using state-holding gates leads to the use of either extensive combinational resources or latches. In order to reduce complexity we experimented with the possibility of removing the state-holding capability normally embedded in each NCL gate. Obviously, the input-completeness with regard to NULL fronts would be lost, and the data integrity is not ensured anymore. We then looked for an alternate way to ensure the integrity of data without having to use local state-holding capability. That was an appealing way to reduce complexity of the combinational logical part. Additional motivations in favor of trying to decrease complexity is the potential for a reduced power consumption and an increased operating frequency. The gain would be even more significant if we could use conventional gates in the combinational parts. Although these observations were made in an FPGA environment, similar gains could be expected in an ASIC environment.

B. Feasibility

Before trying to address the problem of ensuring input completeness with regard to NULL data, it is important to investigate the impact of the state-holding capability of NCL gates in relation to the ability to prevent sources of invalid data such as logic hazards or glitches. For this purpose we enumerated the set of possible glitch conditions and how NCL may possibly prevent them to happen. Fig. 3 summarizes what could occur for a 1-bit output. The extrapolation to an N-bit output is easily obtained by developing the Cartesian product of possible behaviors on 1-bit outputs. The 4 cases on the left are some patterns that may reflect how an s_0s_1=01 data front would arrive on s_{i-0} s_{i-1}. Similarly the right side lists some patterns that could reflect how a null front would arrive at that point following an s_0s_1=01 data front.

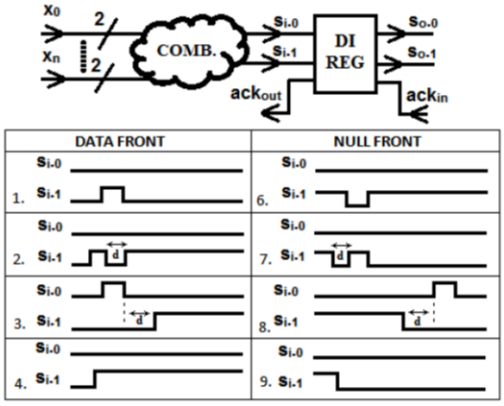


Figure 3. Different type of glitches

Cases 4 and 9 are the normal cases respectively for DATA and NULL fronts. Before going any further we note that with dual or quad rail representation, the logic is *positive.* Positive logic designates all networks of gates such that the activation of an input may ultimately lead either to the activation of one or several outputs or to no change on the outputs and the deactivation of an input may lead either to the deactivation of

one or several outputs or to no change on the outputs. Therefore, cases 1 and 6 are not possible because the output does not stabilize to a DATA value in case 1 or NULL value in case 6. The use of positive, observable and input complete logic avoids all the other cases (2, 3, 7 and 8). Indeed, as in standard Boolean circuits, we can have some reconvergeant fan-outs that are necessary observe in one of cases 2, 3, 7 and 8. However, with positive logic, it is not possible to have a path with an inverted copy of a signal. Nevertheless, if a fork has at least two branches, one may be used for the determination of an output s_0 while the other could contribute to signal s_1, with s_0 and s_1 being complementary. If that is the case, as s_0s_1=11 is not valid, one of the two branches would have to be an orphan. The delay on this branch could be high enough to make the whole system suffer from glitches. Assuming isochronic fork precludes the existence of orphans. In conclusion, all the unsafe cases are prevented by the combination of input completeness, observability, the isochronic fork assumption and the use of positive logic. Therefore the state-holding capability of individual gates is not a necessary condition to have valid outputs.

C. *Protocol Modification and Resources Reduction*

The aim of the proposed modification is to remove the combinational state-holding capability. By doing so the input-completeness with regard to NULL front as expressed in *Rule 1* is violated. Let's consider a *dual rail* non optimized function designed such as all minterms are evaluated. Using AND gates instead of TH22 gates could lead to unexpected output. For example, for a 1-bit XOR circuit which inputs are A and B, and output is S, an input set (A, B) equals to (DATA0, DATA0) leads to S equals to DATA0. When S has been acknowledged the input register is authorized to provide a NULL front. Assuming that for any reason A becomes NULL and B remains DATA0, using AND gates will lead to an unexpected NULL value of S. Therefore removing state-holding capability violates the input-completeness rule as described in *Rule 1*, with regard to NULL front. However, to ensure data integrity, what is important to guarantee is not to prevent the combinational output to become NULL, but for the next stage to believe that it is receiving a new NULL front. In other words, it is enough to avoid the output register to be cleared. This way *Rule 1* is modified such as input-completeness with regard to DATA front remains the same, but with regard to NULL front is changed for a verification at the output of the output register. We call this property *Output Register State-Holding Capability*.

To do so, we propose to modify all the registers such that the ack_{in} signal is completed with a signal called *e* indicating whether or not the previous stage output register is NULL. The behavior of the new register is as follows: if $TH_{22}(ack_{in}, e) = 1$ then the output register could be set, else the output register could be cleared. In other words, the new input acknowledge signal is defined by $ack = TH_{22}(ack_{in}, e)$. The proposed modified protocol needs to add only one Müller C-element for each register irrespectively of the number of bits.

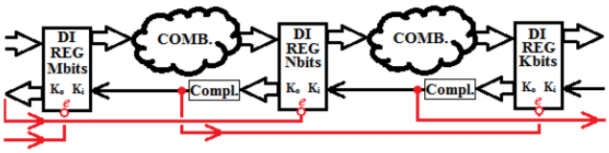


Figure 4. Proposed Framework

Let us now consider how the *e* signal could be generated. We observed that the ack_{out} signal from the output register of the previous stage could be used as the desired *e* signal. Effectively, when $ack_{out}(n-1)$ is low, it means that the $(n-1)^{th}$ stage is waiting for NULL front and as a result its outputs are DATA. When $ack_{out}(n-1)$ is high, it means that the $(n-1)^{th}$ stage is waiting for a DATA front and as a result that its outputs are NULL. Therefore the *e* signal is defined by: $e(n) = not(ack_{out}(n-1))$. The framework is thus modified as depicted in Fig. 4 where *e* is inverted inside each register.

D. *Complexity Analysis*

Complexity can be analyzed in terms of the following parameters:

- K_{comb} : number of combinational elements
- K_{reg} : number of elements for the registers
- s = K_{comb}/K with $K = K_{comb} + K_{reg}$
- A : combinational complexity reduction ratio

Let us define K_{before} and K_{after} as the total number of elements composing the original and the simplified design. We then have:

$$K_{before} = K$$
$$K_{after} = (1-s).K + s.K.(1 - A)$$

$$G = \frac{K_{before} - K_{after}}{K_{before}} = s.A \qquad (1)$$
$$G_{max} = s \text{ when } A = 1 \qquad (2)$$

G_{max} is thus a 'gain' representing the total hardware complexity reduction obtained with SHF-NCL as compared to NCL. Significant gains are obtained when *s.A* is large.

IV. RESULTS

Results concerning resources reduction, evolution of operating frequency and power consumption are reported for different benchmarks implemented in an FPGA environment.

A. *Overhead and Hardware Complexity Reduction*

First of all, adding the *e* signal require extra resources in each register. Fig. 5 shows the evolution of the overhead with the number N of bits of a register.

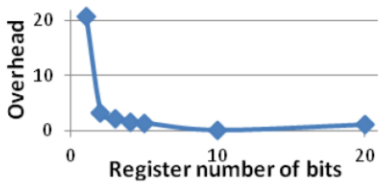


Figure 5. Evolution of the overhead with the register size

Table I shows the hardware complexity reduction for three basic elements: an XOR gate, a full-adder and an ALU described in [7]. The combinational circuit is assumed to be embedded between an input and an output register, which is a worst case because the overhead is usually shared between two stages. Two different FPGA architectures were considered. Finally, in order to validate SHF-NCL, up-counters of various sizes N were implemented as benchmarks. Observed gains are reported in Table II.

TABLE I. RESOURCE GAINS IN NUMBER OF GATES

FPGA	XOR			Full-adder			ALU		
	NCL	eNCL	Gain	NCL	eNCL	Gain	NCL	eNCL	Gain
Virtex5/4	28	22	21,4%	42	39	7,1%	197	80	59%
Igloo	49	43	12,2%	77	69	10,4%	341	174	49%
Average	**16,8%**			**8,8%**			**54%**		

B. *Operating Frequency and Power Consumption*

Impacts on operating frequency and dynamic power consumption have been evaluated by implementing the up-counter example on the *Actel Igloo nano starter kit*. An evaluation of the operating frequency has been obtained by probing the least significant bit of the output. As the speed of NCL and SHF-NCL are different, we report ratios between measured dynamic power at the measured maximum operating frequency rather than just the dynamic power. Results are summarized in Table III.

TABLE II. RESOURCES REQUIRED EXPRESSED IN NUMBER OF GATES AND GAIN ON UP-COUNTER EXAMPLE

FPGA	Type	N					
		2	3	4	5	8	16
Virtex5/4	NCL	107	131	162	186	273	517
	SHF-NCL	99	122	144	167	240	449
	Gain	7,5%	6,9%	11,1%	10,2%	12,1%	13,2%
Igloo	NCL	159	202	247	294	427	773
	SHF-NCL	149	180	215	248	351	629
	Gain	6,3%	10,9%	13,0%	15,6%	17,8%	18,6%
Average	-	**6,9%**	**8,9%**	**12,5**	**12,9%**	**15,0%**	**15,9%**

TABLE III. OPERATING FREQUENCY AND DYNAMIC POWER EVOLUTIONS

Frequency	N						
	8	12	16	24	32	48	64
NCL(MHz)	8,447	7,249	6,874	5,463	5,413	4,166	3,518
SHF-NCL (MHz)	7,575	6,975	7,036	5,988	5,458	4,701	3,760
Gain	**-10%**	**-4%**	**2%**	**10%**	**1%**	**13%**	**6%**
NCL P_{dyn}/f	198	250	273	361	377	509	605
SHF-NCL P_{dyn}/f	206	248	260	336	375	458	572
Gain	**-4%**	**1%**	**5%**	**7%**	**>1%**	**10%**	**6%**

V. DISCUSSION

A. *About Resources Reduction*

SHF-NCL reduces complexity except in extreme cases such as in straight succession of registers. Since the overhead is fixed no matter how large the register is, the relative overhead decreases with the use of multi-bits registers. Required resources were reduced by about 22% on average. Moreover, as was explained in the previous section, results would be better in a system where registers are shared between two stages.

Even though the measurements were made with FPGA technology the same trends are expected with ASIC technology. Whether static or semi-static design style is used, the state-holding capability of each gate will be removed leading to an average gain of 3.6% for semi-static implementation and 9.2% for static implementation for each gate. The two types of implementations are described in [7].

B. *About Operating Frequency and Power Consumption*

Table III reports that operating frequencies and power consumption do not change much with SHF-NCL. Further study concerning these two parameters is left for future work.

C. *About Maximal Gain*

For the examples of up-counter and ALU the maximal gains are respectively around 28% and 67%. The total gains measured are then respectively 60% and 73% of the corresponding maximal gains. As expected, the total gains depend on the ratio between the relative complexity of combinational circuits and registers.

VI. CONCLUSION

In this paper an State-Holding Free Null Convention Logic Protocol was presented. We showed that state-holding is not necessary as long as *Output Register State-Holding Capability* is maintained. By adding one Müller C-element and by modifying the handshake protocol, a substantial reduction of required resources was observed. When the combinational complexity is important compared with register complexity, resources reduction as high as 50% or more depending of the architecture have been observed.

REFERENCES

[1] A. J. Martin, "The limitations to delay-insensitivity in asynchronous circuits," presented at the Proceedings of the sixth MIT conference on Advanced research in VLSI, Boston, 1990.

[2] I. E. Sutherland, "Micropipelines," *Communications of the ACM,* vol. 32, pp. 720-738, 1989.

[3] I. David, R. Ginosar, M. Yoeli, "An efficient implementation of Boolean functions as self-timed circuits," *IEEE Transactions on Computers,* vol. 41, pp. 2-11, 1992.

[4] N.P. Singh, "A design methodology for self-timed systems", Master's Thesis, Laboratory for Computer Science, MIT, 1981.

[5] T. S. Anantharaman, *A delay insensitive regular expression recognizer*: Carnegie Mellon University, Computer Science Dept., 1989.

[6] K. M. Fant and S. A. Brandt, "NULL Convention Logic™: a complete and consistent logic for asynchronous digital circuit synthesis," in *Application Specific Systems, Architectures and Processors, 1996. Proceedings of International Conference on*, 1996, pp. 261-273.

[7] S. C. Smith and J. Di, *Designing asynchronous circuits using NULL convention logic (NCL)*: Morgan & Claypool Publishers, 2009.

[8] S.C. Smith, R.F. DeMarab, J.S. Yuanb, D. Fergusonc, D. Lamb, "Optimization of NULL convention self-timed circuits," *Integration, the VLSI Journal,* vol. 37, pp. 135-165, 2004.

[9] X. Guan, *et al.*, "Performance analysis of low power null convention logic units with power cutoff," in *2010 Asia-Pacific Conference on Wearable Computing Systems, APWCS 2010, April 17, 2010 - April 18, 2010*, Shenzhen, China, 2010, pp. 55-58.