REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

Université SAAD Dahleb de Blida

Faculté des sciences de l'ingénieur

Département d'Aéronautique

Projet de fin d'études

En vue de l'obtention du diplôme d'ingénieur d'état en Aéronautique

Option : Installation

Thème

MODELISATION ET SIMULATION SOUS SIMULINK D'UNE CHAINE DE TRANSMISSION NUMERIQUE EN BANDE DE BASE

Réalisé par :

- BELKHADOUMA Amina
- BENFATTA Atika

Encadré par :

Mr.HELAL.M

PROMOTION

2009/2010


بسم الله الرحمن الرحيم

REMERCIEMENTS

*Nos premiers remerciements vont d'abord à **DIEU** qui nous a offert le courage et la volonté nécessaires pour affronter la vie.*

*Nos remerciements ensuite en particulier à : **Mr Helal** notre encadreur de l'Institut d'Aéronautique de Blida qui a été pour nous un guide bien qualifier à chaque fois que nous avons besoin de ses orientations.*

Nous remercions vivement nos enseignants qui nous ont prodigués le savoir et la méthodologie du travail, à tous les membres du jury qui ont accepte d'examiner et évaluer ce travail.

En fin nous tenons à remercier du fond du cœur toutes les personnes qui ont contribue de près ou de loin à l'élaboration de ce modeste travail.

Dédicace

Avant tout, je tiens à remercier ***DIEU*** *qui m'a offert le courage et la volonté nécessaires pour affronter la vie.*

Je dédie ce modeste travail :

A ma chère et précieuse ***Mère*** *que je prie DIEU le tout puissant de la garder pour toujours en bonne santé.*

A mon cher oncle ***OMAR***

A mes frères et sœurs : ***Hassiba, Yacine, Adel*** *et* ***Chahinez.*** *Et particulièrement à mon neveu* ***Nedjem El Dine Rayane.***

A toute ma famille et spécialement à ma cousine ***YAMINA.***

A toutes mes amies.

A mon promoteur ***Mr. HELLAL***

et tout mes enseignants.

ATIKA

Dédicace

Je dédie ce modeste travail particulièrement à :

***Mon père**, qui n'a jamais cessé de m'encourager à poursuivre mes études en m'apportant le soutien moral, financier et matériel,*

*Lumière de ma vie **Ma mère** qui a toujours cru en moi malgré les obstacles que j'ai pu rencontrer tout au long de ce cursus universitaire,*

***Mes frères**, merci pour votre assistance et le réconfort que vous m'avez toujours apporté en dépit des distances qui nous séparent,*

*A mes très chères amies **Atika** et **Samia**,*

Toutes les autres personnes que je porte dans mon cœur et qui ont sans le savoir participé de manière considérable à ma réussite,

À ceux que ma plume a oubliés.

Amina

Résumé

Dans ce mémoire nous décrivons une série de manipulations écrites sous le logiciel Matlab/Simulink et permettant de se familiariser avec les techniques de codage de canal. L'objectif ici est d'illustrer le plus simplement possible le module de théorie de l'information dont la partie concernant le codage de canal constitue la finalité. La démarche consiste à comparer deux chaînes de transmission numérique en bande de base. La première ne contient pas d'accessoires de codage et de décodage de canal tandis que la seconde en est pourvue.

Summary

In this memory we describe a series of handling written under the Matlab/Simulink software and allowing to familiarize itself with the techniques of coding of channel. The objective here is to illustrate the most simply possible module of information theory of which the part concerning coding of channel constitutes the finality. The step consists in comparing two numerical transmission chains in baseband. The first does not contain accessories of coding and decoding of channel while the second is equipped with it.

ملخص

تحتوي هذه المذكرة على وصف لمجموعة من التطبيقات التي أنجزناها باستعمال برنامج Matlab/Simulink, والتي تسمح بالتعرف على تقنيات تشفير القناة. و الهدف من كل هذا ,هو شرح و إيضاح الوحدة النظرية للمعلومات ,أين تعتبر مرحلة تشفير القناة هي المسعى الأخير , و هذا بأبسط طريقة ممكنة . لذلك اتبعنا منهج المقارنة بين سلسلتين للإرسال الرقمي ذات المجال القاعدي , بحيث أن السلسلة الأولى لا تحتوي على أية ملحقات للتشفير أو لفك الشفرة , بعكس السلسلة الثانية و التي زودت بهذه الملحقات .

Table des matières

III - Simulation sous MATLAB/SIMULINK D'une chaîne de transmission numérique en bande de base

IV - Mode d'emploi de la simulation

Liste des figures

Liste des tableaux

Glossaire

AMI: **A**lternate **M**ark **I**nversion

BnZs: **B**ipolar with **n** **Z**ero **S**ubstitution

B3ZS: **B**ipolar with **Three-Z**ero **S**ubstitution

B8ZS: **B**ipolar with **Eight-Z**ero **S**ubstitution

DCE: **D**ata **C**ommunication **E**quipment

ETCD : **E**quipement **T**erminal de **C**ircuit de **D**onnées

FDDI: **F**iber **D**istributed **D**ata **I**nterface

GEOs: **G**éostationnaire **E**arth **O**rbiting **S**atellites

GPS: **G**lobal **P**ositioning **S**ystem

HDBn : **H**aute **D**ensité **B**inaire d'ordre **n**

IEEE: **I**nstitute of **E**lectrical and **E**lectronics **E**ngineers

LEOs: **L**ow **E**arth **O**rbiting **S**atellites

MATLAB: **M**atrix **L**aboratory

MEOs: **M**edium **E**arth **O**rbiting **S**atellites

MLT3: **M**ultiline **T**ransmission, three level

nB/mB: **n** **B**yte/**m** **B**yte

NRZ: **N**on **R**eturn to **Z**ero

NRZI: **N**on **R**eturn to **Z**ero **I**nverted

PCM : **P**ulse **C**ode **M**odulation

PE: **P**hase **E**ncode

RNIS : **R**éseau **N**umérique à **I**ntégration de **S**ervices

RZ: **R**eturn to **Z**ero

TEB: **T**aux d'**E**rreur **B**inaire

USB: **U**niversal **S**erial **B**us

Introduction générale

De nos jours, les systèmes de télécommunication numériques avancés sont de plus en plus présents dans notre quotidien et connaissent une évolution sans précédant .Les modems haut débit, les téléphones cellulaires, et les réseaux de communication par satellites ne présentent que la pointe d'un iceberg que nous découvrirons dans la décennie de ce nouveau millénaire.

La plupart de ces dispositifs électroniques nécessitent un système de transmission numérique, c'est-à-dire un émetteur, un récepteur de signaux numériques et un canal de transmission permettant au module de communiquer avec son entourage.

En plus de ces dispositifs matériels, un outil d'analyse basée sur une simulation logicielle est souvent nécessaire, car il peut réduire le temps de conception et les coûts de développement des projets complexes. Comme il permet de rectifier et d'améliorer le comportement du système face aux différents types de brouilleurs présents dans un environnement hostile.

L'objectif général de notre projet consiste à réaliser un système de transmission numérique en bande de base avec l'application du codage du canal, où Simulink et ses blocs seront employés pour démontrer des concepts du système de transmission numérique en bande de base.

Le présent mémoire est composé, en plus de l'introduction générale, de quatre chapitres et d'une conclusion.

Les chapitres de ce mémoire seront enchaînés de la façon suivante :

- Le premier chapitre aura pour but de présenter les techniques des codages les plus utilisées dans la transmission en bande de base.
- Le deuxième chapitre représente l'interface de la simulation (Matlab/Simulink) et le fonctionnement des différents schémas blocs pour modéliser un système.
- Le troisième chapitre nous permettra d'aborder plus spécifiquement le problème de construction de deux modèles d'une chaîne de transmission numérique en bande de base (Le premier modèle ne contient pas d'accessoires de codage et de décodage de canal tandis que le second en est pourvu) bloc par bloc et de choisir les paramètres de chaque bloc.

- Dans le quatrième et dernier chapitre du présent mémoire, nous avons testé le bon fonctionnement du modèle par la comparaison entre les signaux émis et reçus.

En conclusion, nous allons faire une analyse des résultats obtenus, dégager l'intérêt d'un tel logiciel et faire des propositions pour la poursuite de cette étude.

CHAPITRE I :

La transmission en bande de base

Références: [5], [6], [8], [9]

I - La transmission en bande de base

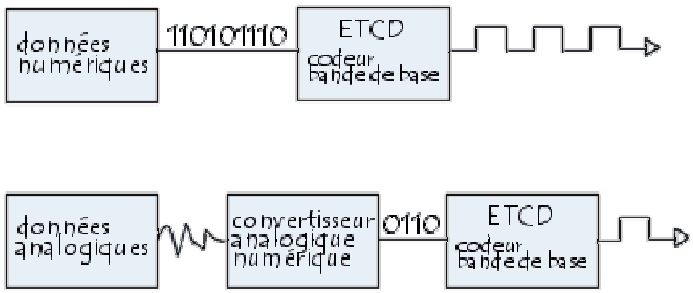
I.1 Introduction :

La transmission numérique consiste à faire transiter les informations sur le support physique de communication sous forme de signaux numériques. Ainsi, des données analogiques devront préalablement être numérisées avant d'être transmises.

Toutefois, les informations numériques ne peuvent pas circuler sous forme de 0 et de 1 directement, il s'agit donc de les coder sous forme d'un signal possédant deux états, par exemple :

- deux niveaux de tension par rapport à la masse
- la différence de tension entre deux fils
- la présence/absence de courant dans un fil
- la présence/absence de lumière
- ...

Cette transformation de l'information binaire sous forme d'un signal à deux états est réalisée par l'**ETCD** (***é**quipement **t**erminal de **c**ircuit de **d**onnées*, ou en anglais ***DCE***, ***D**ata **C**ommunication **E**quipment*), appelé aussi *codeur bande de base*, d'où l'appellation de *transmission en bande de base* pour désigner la transmission numérique...



I.2 Généralités sur la transmission en bande de base :

La transmission en bande de base consiste à transmettre directement les signaux numériques (les suites de bits) sur le support, *sur des distances limitées* (de l'ordre de 30 km). Le signal en bande de base ne subit pas de transposition de fréquence et l'**ETCD** se réduit à un simple codeur (codeur bande de base, bien que l'on dise à tort "modem bande de base"). La figure I.1 résume le principe de la transmission de données en bande de base.

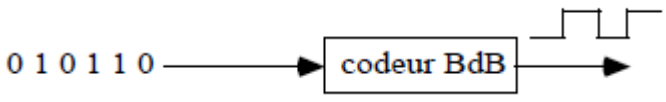


Figure I.1 : le principe de la transmission de données en bande de base

Le codeur bande de base transforme la suite de bits *{ai}* en une suite de symboles *{dk}*, pris dans un alphabet fini de q symboles :

$$dk \in \{\alpha 1,\ldots, \alpha q\}$$

Les *dk* ont tous la même durée Δ (intervalle significatif), égale ou multiple de *T* (intervalle élémentaire).

Un tel procédé, simple et peu coûteux, ne peut être employé que si le support n'introduit pas de décalage en fréquences (en particulier, on ne doit pas trouver de transformateurs d'isolement provoquant des coupures de circuits, ni d'éléments introduisant un décalage des horloges, comme les amplificateurs). On l'utilise donc en général sur des câbles à grande bande passante. Le débit maximum que l'on peut obtenir dépend de la longueur du câble et de sa section. La transmission en bande de base est désormais très largement utilisée puisqu'elle est utilisée dans les réseaux locaux.

I.3 Caractéristiques de la bande de base :

Un signal électrique, ou optique est caractérisé par sa densité spectrale de puissance *(encombrement spectral)*. On peut le transporter dans la bande d'origine avec un éventuel transcodage *(bande de base)* ou effectuer une transposition dans une autre bande de fréquence *(modulation)*.

Un signal est dit **"bande de base"** s'il n'a pas subit de transposition en fréquence.

La transmission en bande de base s'avère particulièrement simple et économique pour les signaux synchrones et rapides.

I.3.1 Codage :

La transmission directe du message n'est généralement possible que sur de très courtes distances. Un signal à transmettre subira donc un codage plus ou moins élaboré afin d'adapter son spectre au support utilisé : réduction ou suppression de la composante continue, transmission de l'horloge en synchrone ...

Les codages utilisés peuvent être classés selon le nombre de niveaux électriques :

- **Le codage à deux niveaux**: le signal peut prendre uniquement une valeur strictement négative ou strictement positive (-X ou +X, X représentant une valeur de la grandeur physique permettant de transporter le signal) : NRZ, NRZI, biphasé, biphasé différentiel…
- **Le codage à trois niveaux**: le signal peut prendre une valeur strictement négative, nulle ou strictement positive (-X, 0 ou +X) : bipolaires AMI, bipolaires haute densité, RZ…

Il existe aussi le code par bloc. Il code chaque bloc de k bits par un bloc de n symboles pris dans un alphabet de taille L. L'alphabet étant généralement binaire, ternaire, ou plus rarement quaternaire (noté resp. B, T, Q) : nB/mB, nB/mT.

I.3.2 Débits :

Débit binaire : D = 1/T en bits/s (bps) avec T=durée du bit.
Rapidité de modulation : R = 1/δ en **bauds** avec δ=durée du plus court signal transmis.
R exprime le nombre de niveaux transmissibles par secondes.
En synchrone on préfère généralement parler du débit binaire, seuls les codages multi-niveaux permettent D > R.

I.3.3 Supports :

- Voie téléphonique : impossible (bande passante, multiplex en fréquence...).
- Câble métalliques : Possible car il n'y a pas de coupure de la bande passante. La portée sera limitée par l'atténuation, la vitesse et le type de codage.
- Fibre optique : sans difficultés.

- **Le transcodage :** est le passage d'un code à un autre.

I.4 Intérêt des codages en bande de base :

Le signal binaire n'est généralement pas transmis directement sur la ligne et différents codages numériques sont utilisés pour différentes raisons :

- La récupération de l'horloge nécessaire en transmission synchrone est facilitée par des séquences qui présentent des changements d'états fréquents et évitent ainsi les longues suites de 1 ou de 0.
- Le spectre d'un signal binaire est concentré sur les fréquences basses qui sont les plus affaiblies sur la ligne.
- Les perturbations subies par un signal sont proportionnelles à la largeur de sa bande de fréquence.

La transmission est dite en bande de base si elle ne subit aucune transposition de fréquence par modulation. Les fréquences initiales du signal émis sont donc préservées. La transmission en bande de base ne peut donc par essence être utilisée que sur support cuivre.

Les signaux bande de base sont sujets à une atténuation dont l'importance dépend du support employé et doivent donc être régénérés périodiquement sur une longue distance.

I.5 Les codages :

Pour l'ensemble des différents codes décrits, nous prendrons la même suite binaire afin de permettre la comparaison : 1 0 0 0 0 1 0 1 1 1 1

I.5.1 Codage NRZ *(Non Return to Zero)*

Ce code est simple utilisé couramment entre l'ordinateur et ses périphériques. , comme la jonction V24, ou la liaison série RS232 (figure I.2).

Principe : très proche du codage binaire de base, il code un 1 par +V, un 0 par –V.

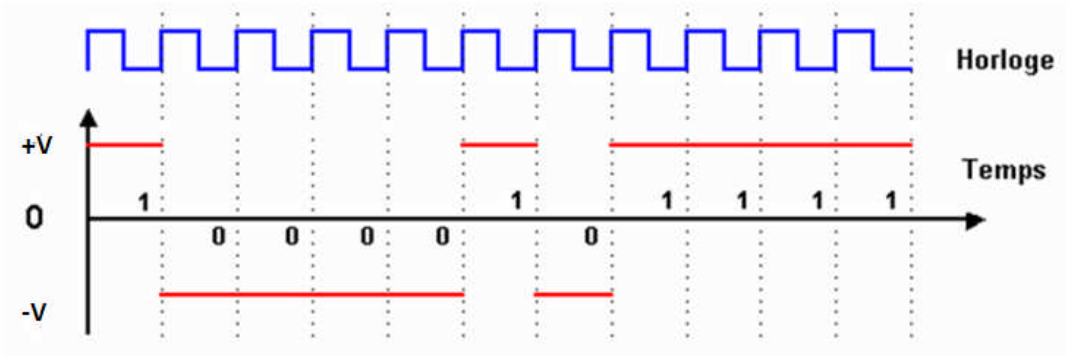


Figure I.2 : le principe de codage NRZ

Le codage NRZ améliore légèrement le codage binaire de base en augmentant la différence d'amplitude du signal entre les 0 et les 1. Toutefois les longues séries de bits identiques (0 ou 1) provoquent un signal sans transition pendant une longue période de temps, ce qui peut engendrer une perte de synchronisation.

Le débit maximum théorique est le double de la fréquence utilisée pour le signal : on transmet deux bits pour un hertz.

I.5.2 Codage NRZI *(Non Return to Zero Inverted)*

Le code binaire NRZI (figure I.3) est indépendant de la polarité et adapté à la transmission photonique.

Utilisation : Fast Ethernet (100BaseFX), FDDI, le réseau industriel BITBUS, Le bus USB.

Principe : on produit une transition du signal (changement d'état) pour chaque 1, pas de transition pour les 0.

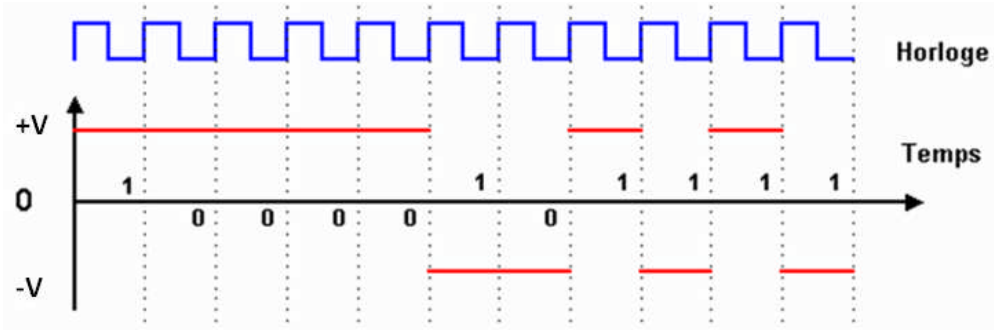


Figure I.3 : le principe de codage NRZI

Avec le codage NRZI, on voit que la transmission de longues séries de 0 provoque un signal sans transition sur une longue période. Le débit binaire est le double de la fréquence maximale du signal : on transmet deux bits pour un hertz.

Le codage NRZI possède de nombreux avantages, dont :

- La détection de la présence ou non du signal
- La nécessité d'un faible courant de transmission du signal
- Facile à mettre en œuvre, bonne utilisation de la bande passante.

Par contre, il possède un défaut: la présence d'un courant continu lors d'une suite de zéro, gênant la synchronisation entre émetteur et récepteur.

I.5.3 Codage biphasé *(Manchester ou PE pour **P**hase **E**ncode)*

C'est un code binaire, équilibré, conservation de l'horloge, mais spectre très large (le double) (figure I.4).

Utilisation : Ethernet 10Base5, 10Base2, 10BaseT, 10BaseFL.

Principe : dans le codage Manchester, l'idée de base est de provoquer une transition du signal pour chaque bit transmis. Un 1 est représenté par le passage de +V à –V, un 0 est représenté par le passage de -V à +V.

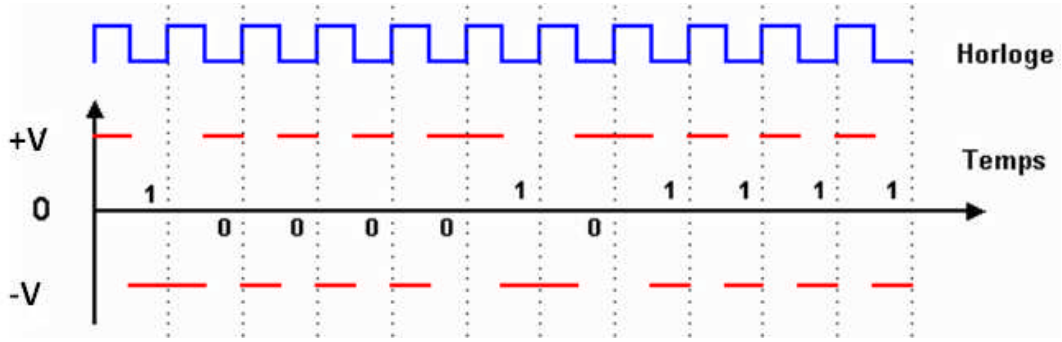


Figure I.4 : le principe de codage Manchester

La synchronisation des échanges entre émetteur et récepteur est toujours assurée, même lors de l'envoi de longues séries de 0 ou de 1. Par ailleurs, un bit 0 ou 1 étant caractérisé par une transition du signal et non par un état comme dans les autres codages, il est très peu sensible aux erreurs de transmission. La présence de parasites peut endommager le signal et le rendre incompréhensible par le récepteur, mais ne peut pas transformer accidentellement un 0 en 1 ou inversement.

Toutefois, le codage Manchester présente un inconvénient : il nécessite un débit sur le canal de transmission deux fois plus élevé que le codage binaire. Pour 10 Mbit/s transmis, on a besoin d'une fréquence à 10 Mhz.

Ceci le rend difficilement utilisable pour des débits plus élevés. L'utilisation de ce codage pour une transmission à 1 Gbit/s nécessiterait une fréquence maximale du signal de 1 Ghz, ce qui est incompatible avec les possibilités des câblages actuels ainsi qu'avec les normes sur les compatibilités électromagnétiques.

I.5.4 Codage biphasé différentiel *(Manchester différentiel)*

Il est identique au code Manchester mais présente une indépendance de la polarité.

Utilisation : Token Ring.

Principe : c'est la présence ou l'absence de transition au début de l'intervalle du signal d'horloge qui réalise le codage. Un 1 est codé par l'absence de transition, un 0 est codé par une transition au début du cycle d'horloge (figure I.5).

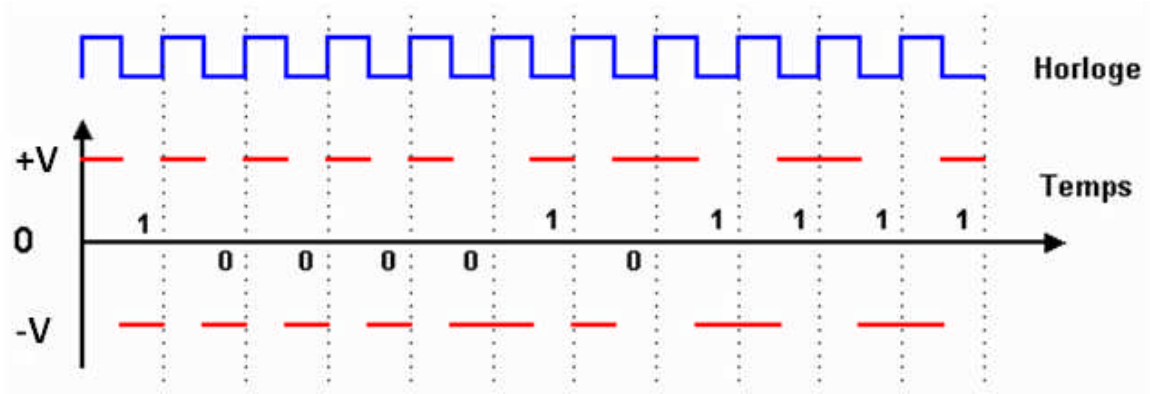


Figure I.5 : le principe de codage Manchester différentiel

A noter la présence de deux symboles particuliers : J et K. Ils sont codés par +V et –V sur toute la durée d'un cycle d'horloge. Ils ont pour but de marquer le début et la fin d'une trame.

Le codage présente le même inconvénient que le codage Manchester : nécessite une fréquence égale à celle du débit utile. Il présente par contre un avantage : ce sont les transitions du signal et non pas ses états qui représentent les bits transmis, il est donc insensible aux inversions de fils dans le câblage.

Problème s'il y a corruption d'un des symboles : la suite est mal décodée.

I.5.5 Codage bipolaire simple ou AMI *(**A**lternate **M**ark **I**nversion)*

Ce code ternaire est équilibré, indépendant de la polarité et dérive de l'horloge (suite de 0) (figure I.6).

Utilisation : Lignes transmission DS1/T1, le système de téléphonie numérique PCM, Bus S0 du RNIS *(avec un codage inversé)*.

Principe : Les 0 sont représentés par des potentiels nuls, les 1 par +V et –V en alternance.

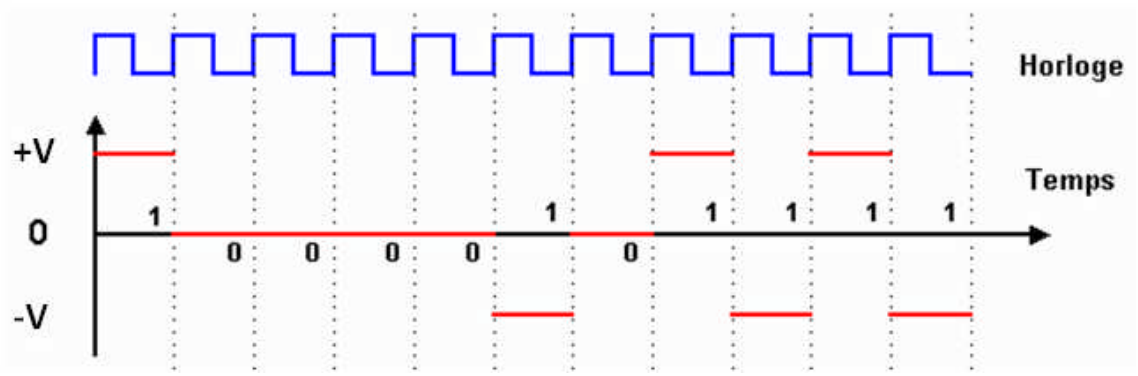


Figure I.6 : le principe de codage AMI

Ici encore, il peut y avoir de longues séquences sans potentiel et donc perte de synchronisation.

Le codage AMI possède de nombreux avantages, dont :

- Spectre étroit.
- En conséquence, le signal codé est aussi facilement modulable sur une porteuse de base, ou supportera un débit important sur un support de transmission à fréquences basses.

Par contre, il possède des défauts, dont:

- Problèmes de décodage lors de longues séquences de 0, mais ce problème est résolu par le codage régulier de bits supplémentaires de façon à maintenir la synchronisation : plus ces bits sont fréquents, plus facile sera la synchronisation.
- Le décodage bipolaire est moins stable autour du codet central (codé 0 et non -a ou +a) pour des transmissions à longue distance, car le signal tend à être transmis de façon différentielle (on ne détecte bien que les transitions, qui subissent aussi un déphasage, et l'amplitude d'une transition de +a vers 0 ou de +a vers -a est plus difficilement distinguable, ce qui rend difficile la synchronisation entre les séquences comme celle à 3 états (+a, 0, a) codant 101 et l, et celle à 2 états (+a, a) codant 11, ce qui privilégie les états -a et +a (c'est-à-dire la détection de bits 1) au détriment de l'état 0, particulièrement si les composantes continues sont importantes (longues chaînes de 0 sans transition); pour pallier le problème, il ne suffit pas d'insérer des 1 mais il faut aussi insérer quelques 0.

I.5.6 **Codage HDBn** *(**H**aute **D**ensité **B**inaire d'ordre **n**)* **ou BnZs** *(**B**ipolar with **n** **Z**ero **S**ubstitution)*

Même codage que le code bipolaire mais une transformation des suites de plus de n zéros est basée sur la violation de l'alternance : bit de viol (noté V) (figure I.7).

Utilisation : HDB3, B8ZS, B3ZS.

Principe : le principe de base est le même que pour le codage bipolaire, mais pour éviter une trop longue série de 0, on introduit un bit supplémentaire au signal pour terminer une série de n 0 consécutifs. Ce bit supplémentaire est de même phase que le dernier 1 transmis pour pouvoir l'identifier, afin qu'il ne soit pas pris en compte dans l'information transmise.

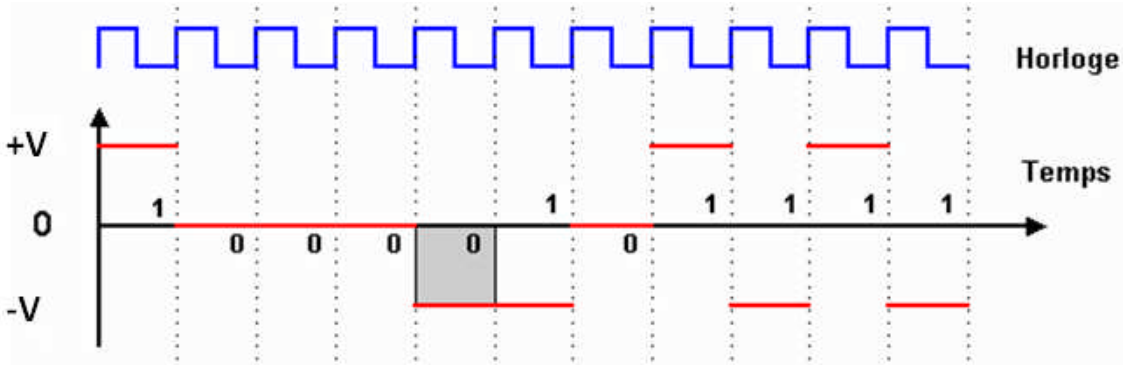


Figure I.7 : le principe de codage HDBn

• Règle de codage :

Le '**n**' de BHDn indique le nombre de 0 que l'on peut envoyer. On le choisit en fonction de la fiabilité du support et du matériel. La valeur pour le premier 1 à envoyer est fixée par convention entre l'émetteur et le récepteur.

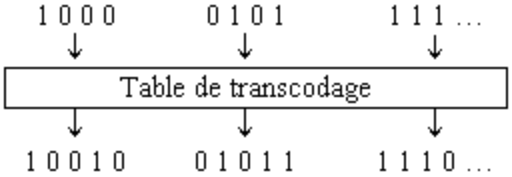
- **Si on doit envoyer un 1**, on envoie l'inverse de la valeur envoyée pour le pas précédent.
- **Si on doit envoyer un 0** :
  - Si les n+1 bits suivants ne sont pas tous à 0, on continue le codage comme en *codage bipolaire simple*.
  - Si les n+1 bits suivants sont tous à 0, les n bits suivants sont codés à 0 et le n+1 sera codé avec la même valeur que le code du 1 précédent (on viole alors l'alternance).
- **Après une violation de l'alternance** :
  - Si le bit suivant est un 1, il est codé avec la valeur inverse de celle qu'on vient de mettre pour l'alternance.
  - Si le bit suivant est un 0 :
    - Si les n+1 bits suivants ne sont pas tous à 0, on continue le codage comme en *codage bipolaire simple.*
    - Si les n+1 bits suivants sont tous à 0, le premier est codé avec la valeur inverse de celle qu'on vient de mettre pour l'alternance, les n-1 bits suivants sont codés à 0 et le $n+1^{e}$ sera codé avec la même valeur que le premier (on viole l'alternance).

I.5.7 Codage nB/mB *(n-byte/m-byte)*

Utilisation : 4B/5B, Fast Ethernet, 8B/10B, Gigabit Ethernet

Principe : Il s'agit d'un codage par bloc. On utilise une table de transcodage pour coder un groupe de n bits en m bits, avec m < n. Ce codage ne définit pas la mise en ligne des bits. On utilise généralement pour cela un codage de type NRZI ou MLT3.

La suite binaire 1 0 0 0 0 1 0 1 1 1 1 précédemment utilisée va être découpée en groupes de 4 bits. La table de transcodage ci-dessous permet de transformer chaque groupe de 4 bits en groupe de 5 bits (tableau I.1).



La suite à transmettre ne comporte pas plus de deux 0 consécutifs, ce qui la rend plus facile à transmettre une fois codée en NRZI ou MLT3.

Groupe de 4 bits	Symbole 4B5B
0000	11110
0001	01001
0010	10100
0011	10101
0100	01010
0101	01011
0110	01110
0111	01111
1000	10010
1001	10011
1010	10110
1011	10111
1100	11010
1101	11011
1110	11100
1111	11101

Tableau I.1 : transcodage 4B5B

Ce type de codage apporte la garantie de ne pas avoir à transmettre plus de deux 0 successifs. Les caractères spéciaux, hors données utiles, peuvent trouver leur place dans la table de transcodage sans nécessiter un état spécial du signal comme dans les codages Manchester.

Le codage 4B5B augmente la fréquence du signal. Par exemple 125Mhz pour 100Mbps. Associé à un codage de type NRZI, on obtient dans le cas du Fast Ethernet (100BaseFX) une fréquence de 62.5Mhz. Avec un codage MLT3, la fréquence du signal tombe à 31.25Mhz pour le Fast Ethernet 100BaseTX.

Par ailleurs ce type de codage laisse un nombre important de mots de 5 bits inutilisés. Même en éliminant les groupes pouvant posés des problèmes de transmission comme 00000 par exemple, il reste des mots pouvant être utilisés pour le contrôle de la transmission ou d'autres fonctions comme début ou fin de paquet par exemple.

I.5.8 Codge RZ *(Return to Zero)*

Ce code ternaire simple (figure I.8) limite les interférences entre symboles et le codage de l'horloge. C'est la solution du problème de synchronisation de NRZ (c.-à-d., décidant quand un bit commence et finit). La première solution consiste à faire retomber le signal à 0V après chaque valeur binaire.

Principe : Utilise 3 niveaux de tension pour coder un chiffre :- V, 0, +V.
Il y a un mi transition de bit au retour de 0, de n'importe quel niveau c'était avant.

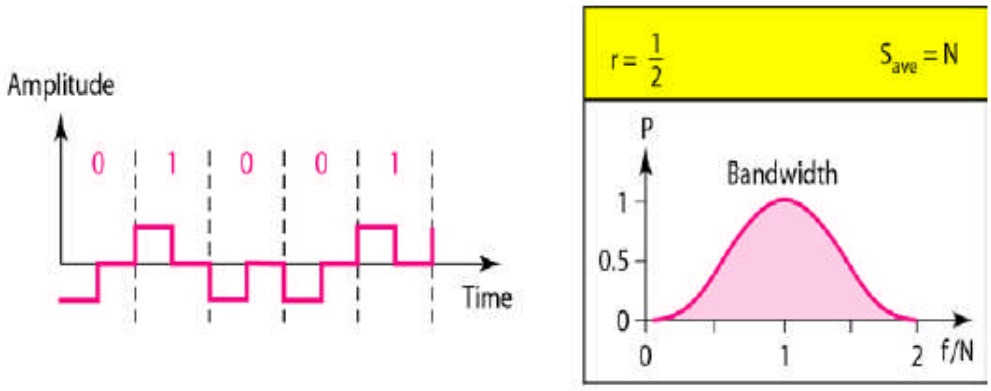


Figure I.8 : le principe de codage RZ

I.6 Les critères de choix d'un codage :

I.6.1 Les codages et leur adaptation au support :

Les supports de transmission coupent brutalement les fréquences au voisinage de la fréquence nulle. Le codage le plus mal adapté est le *NRZ*, puisque la puissance est concentrée dans les basses fréquences. Le codage biphasé a un spectre particulièrement large et ne peut être envisagé que sur des supports à large bande (câbles non chargés, coaxiaux…). On préfère en général les codes bipolaires qui, pour une même largeur de bande, permettent un débit au moins 2 fois plus élevé (sur câbles chargés).

I.6.2 Les codages et la résistance aux bruits :

La sensibilité aux bruits est directement liée au nombre de niveaux du signal (à sa valence). On remarquera que les codes bipolaires, de valence 3, sont plus sensibles aux erreurs que les codages à 2 niveaux.

I.6.3 Les codages et les problèmes d'horloge :

Nous avons vu que les longues suites de bits identiques compliquent la tâche du récepteur, lorsque l'horloge est reconstituée à partir des transitions du signal. Les codes *BHDn* apportent une réponse à ce problème mais ces techniques peuvent rendre le décodage des données impossible en cas d'erreur (un viol intempestif à cause d'un bruit introduirait une série d'erreurs).

I.7 Supports de transmission :

Les supports de transmission constituent le support physique qui permet le transport de l'information d'un point à un autre dont il existe plusieurs types et particulièrement ceux utilisés en bande de base à savoir les fibres optiques, les câbles coaxiaux et les paires torsadées…Pour de plus amples informations nous proposons à nos lecteurs de consulter la référence [9].

Pour que l'information soit transportée avec la plus grande fidélité possible, de la source vers le destinataire, le support de transmission doit présenter les caractéristiques suivantes :

- Le support ne doit pas déformer le signal transporté. Ce dernier peut être éventuellement atténué.
- L'information transportée ne doit pas être altérée par des perturbations indésirables telles que le bruit ou la diaphonie.
- La bande passante du support doit inclure le spectre fréquentiel du signal transporté.

En plus de ces caractéristiques, le support doit offrir :

- Le débit d'information le plus élevé possible.
- Le prix de l'infrastructure le plus faible possible.

Le support répondant à toutes ces exigences n'est malheureusement pas encore disponible. La recherche d'une solution économiquement viables passera souvent par l'adaptation du signal transmis au support utilisé par le biais de différentes techniques de traitement du signal telles que : le codage, la modulation, l'égalisation, etc.

I.7.1 Caractéristiques d'un support de transmission :

Un support de transmission se comporte généralement comme un filtre passe bande (figure I.9) ne laissant passer que les signaux dont les fréquences sont comprises entre une fréquence basse (f_b) et une fréquence haute (f_h). Il est clair que pour les signaux dont la densité spectrale présente une forte composante aux voisinages de la fréquence nulle, une grande partie de l'information sera perdue. La même remarque est valable pour les signaux présentant des composantes hautes fréquences ($>f_h$).

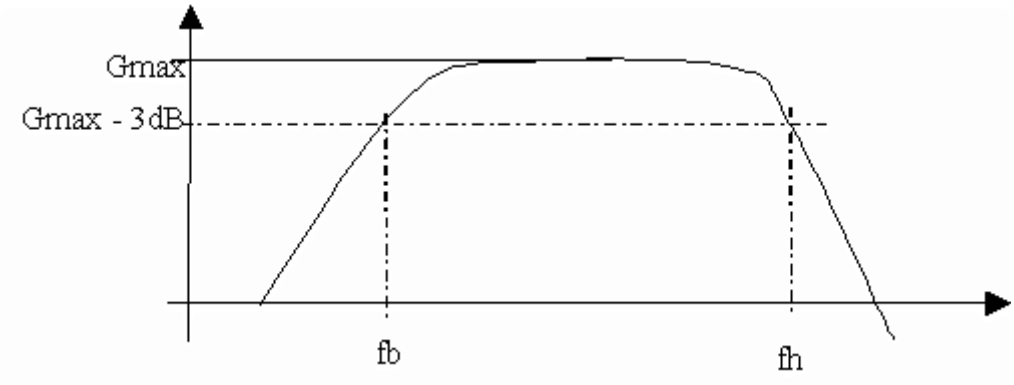


Figure I.9 : gabarit d'un filtre passe bande modélisant une ligne de transmission.
La bande passante est égale à : $BP=f_h-f_b$

• Bruit

Le bruit est un signal aléatoire s'ajoutant au signal utile pendant la transmission. Suivant sa puissance il peut être une source d'erreur non négligeable. Il existe plusieurs types de bruit :

- ***Le bruit blanc*** : Sa densité spectrale de puissance est constante en fonction de la fréquence d'où son nom en comparaison avec la lumière blanche. Il peut être d'origine thermique ou d'origine quantique (bruit de grenaille). Le bruit thermique est généré par l'agitation thermique des électrons dans les conducteurs et les éléments passifs du système de transmission. Le bruit de grenaille quant à lui, est généré par les éléments discrets constituant le système de transmission (transistors, diodes, …).
- ***Le bruit impulsionnel*** : il est d'origine électromagnétique et vient des sources électriques proches qui induisent des courants électriques sur la ligne de transmission. Ce type de bruit dépend fortement de l'environnement du support. Sa puissance est faible mais peu atteindre de fortes valeurs pendant de courts instants d'où son nom. Le blindage des lignes est le meilleur moyen pour réduire son influence.
- ***Le bruit additif*** : il s'ajoute au signal indépendamment de celui-ci. Il peut donc être facilement caractérisé en absence du signal utile.

La qualité d'une transmission sera définie par le rapport signal sur bruit R=S/B. S étant la puissance du signal utile et B la puissance du bruit. Ce rapport peut être également donné en dB, dans ce cas $R_{dB}=10\log_{10}(S/B)$.

Le taux d'erreur introduit par la ligne de transmission et le débit de la ligne seront définis en fonction du rapport R.

•Diaphonie

La diaphonie est la perturbation du signal transporté par un autre signal utile mais transmis sur une ligne voisine. La diaphonie est dite intelligible si elle peut être correctement interprétée par le récepteur auquel elle n'est pas initialement destinée. Ceci peut avoir de grave conséquences puisque le secret de la communication est violé. Si maintenant l'information est

intelligible elle sera traitée comme un bruit. L'effet de la diaphonie sera encore une fois réduit par le blindage des lignes.

•Débit

Le débit binaire théorique (D) d'un canal de transmission est donnée par l'expression: $D=BP.\log_2(1+R)$.Avec BP, la bande passante du canal et R le rapport signal sur bruit. Le débit binaire s'exprime en bit/s.

I.7.2 Les liaisons radioélectriques :

Définition

Les liaisons radioélectriques utilisent la propagation des ondes électromagnétiques dans l'air libre. Elles ont l'avantage de ne pas nécessiter de lourds travaux d'infrastructure. Cependant le support utilisé est commun à tout le monde. Les bandes de fréquences représentent donc une ressource rare et leur utilisation est réglementée par des organismes officiels nationaux et internationaux. Etant donné que les bandes de fréquences utilisées sont imposées, le signal à transmettre sera toujours transposer en fréquence par modulation. De ce fait, le type de support n'utilise jamais la transmission en bande de base. Notre étude ici se limitera à donner les principales caractéristiques des liaisons par faisceaux hertzien et par satellite.

I.7.2.1 Les faisceaux hertziens :

Un faisceau hertzien est un système de transmission de signaux, numériques ou analogiques, entre deux points fixes. Il utilise des ondes radioélectriques très fortement concentrées à l'aide d'antennes directives. La directivité du faisceau et d'autant plus grande que la longueur d'onde utilisée est petite et que la surface de l'antenne émettrice est grande.

Le faisceau est un support de type pseudo-4 fils. Les deux sens de transmission sont portés par des fréquences différentes. Pour des raisons de distance et de visibilité, le trajet hertzien entre l'émetteur et le récepteur est souvent découpé en plusieurs tronçons, appelés bonds, reliés par des stations relais qui reçoivent, amplifient et remettent le signal modulé vers la station suivante.

I.7.2.2 La transmission par satellite :

Les satellites permettent de réduire le nombre de point hertziens grâce à leur grande couverture. Ils sont utilisés en télécommunication pour les liaisons intercontinentales et national pour les applications nécessitant la couverture de grandes zones géographiques telles que le GPS, la télévision par satellite, la téléphonie mobile etc. Les fréquences utilisées se situent entre 4 et 15 GHz.

Techniquement, les liaisons par satellite peuvent se substituer complètement aux faisceaux hertzien mais l'importance des coûts de maintenance des satellites limites leur utilisation à des d'applications spécifiques.

On distingue 3 types principaux de satellites :

- Les satellites à orbite basse de défilement LEOs et MEOs (pour Low/Medium Earth Orbiting Satellites)
- Les satellites géostationnaires GEOs

Les caractéristiques de ces 3 types de satellites sont résumées dans le tableau suivant :

	GEOs	MEOs	LEOs
Altitude	36000 km	13000 km	De 640 à 1600km
Applications	Diffusion radio et TV, VSAT	Téléphonie mobile, transmission de données à faible débit	Téléphonie mobile, transmission de données
Débit binaire	Jusqu'à 155 Mbit/s	De 9.6 à 38.4kbits/s	De 2,4 à 155 Mbits/s
Durée de vie	10 à 15 ans	8 à 10 ans	8 à 10 ans
Temps de transit	0.25 à 0.5s	0.1s	50ms

Tableau I.2 : les caractéristiques des satellites (LEOs, MELs et GEOs)

I.8 Conclusion :

Aucune méthode de codage pour les transmissions en bande de base n'est à priori parfaite, donc le type de codage doit être choisi en fonction des paramètres connus du support. Avec l'avènement des réseaux locaux, seul un petit nombre de codage bande de base sont réellement employés.

Les techniques de transmission ne suffisent pas à assurer que les communications se déroulent sans aucune erreur. C'est pourquoi des techniques de protection contre les erreurs sont développées.

CHAPITRE II :

Présentation du logiciel MTLAB/SIMULINK

Références: [1], [7], [10], [11]

II - Présentation du logiciel MATLAB/SIMULINK

II.1 Introduction :

Ce chapitre a pour objectif premier de présenter un support à l'utilisation des outils MALAB/SIMULINK.

MATLAB est un produit de la firme ***The Mathworks Inc***. Le nom du logiciel MATLAB vient du groupe de mots anglais **M**atrix **L**aboratory, il est orienté en un premier lieu sur l'élaboration des massifs de données (des matrices et des vecteurs).

MATLAB est un programme interactif pour le calcul scientifique qui facilite le calcul matriciel dans différents domaines scientifiques comme le traitement de signal, l'électronique, les commandes de processus et qui permet une visualisation rapide de données. Il fonctionne dans plusieurs environnements tels que X-Windows, Windows, Macintosh.

L'ajout de ″boîtes à outils″ (Toolboxes) permet d'introduire de nouvelles fonctions mathématiques spécifiques à un domaine d'application particulier (signal Processing Toolbox, System Identification Toolbox, Control System Tool-box, u-Synthesis and Analysis Toolbox, Robust Control Toolbox, Optimization Toolbox, Neural Network Toolbox, Image processing Toolbox, etc.) afin de tester rapidement les performances de nouvelles méthodes. Dans cette optique, MATLAB contient de nombreuses fonctionnalités pour :

- L'analyse de données et la visualisation,
- Le calcul numérique et symbolique,
- La modélisation, la simulation,
- Le développement d'application et l'interface graphique.

SIMULINK est un instrument interactif pour la modélisation, l'imitation et l'analyse des systèmes dynamiques. Il donne la possibilité de construire des blocs diagrammes graphiques, d'imiter les systèmes dynamiques, d'étudier la capacité de fonctionnement des systèmes et d'améliorer les projets. Simulink est entièrement incorporé dans MATLAB, et permet un accès rapide à un large spectre d'instruments d'analyse et de conception. Il dispose d'une bibliothèque de blocs de calcul élémentaire (figure II.2) que l'utilisateur peut assembler pour créer ses systèmes sans écrire des lignes de codes.

Dans SIMULINK, la simulation se fait en deux étapes. La première consiste à créer le modèle graphique du système à simuler en utilisant l'éditeur de modèle de SIMULINK. Ensuite, en spécifiant l'étendu temporel, il est possible de simuler le comportement du système.

II.2 La structure du SIMULINK :

Simulink étant un outil de Matlab, pour le démarrer on doit d'abord démarrer Matlab en allant dans le menu démarrer. La fenêtre de commande de Matlab est montrée à la figure II.1.

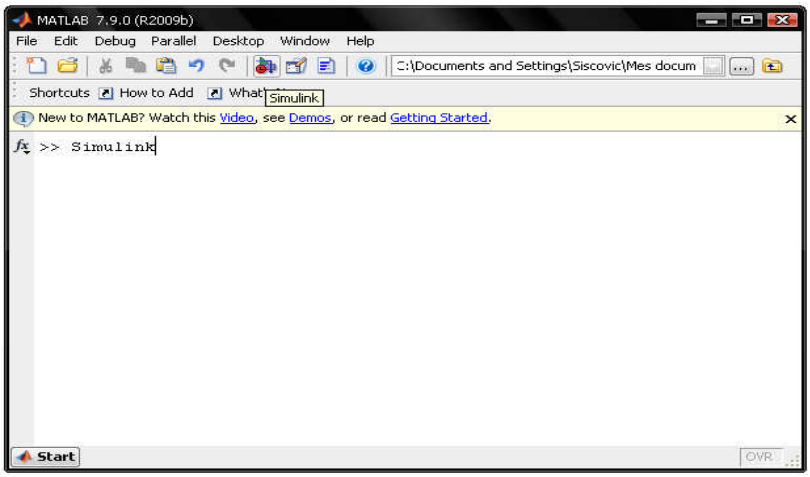


Figure II.1 : La fenêtre de commande du Matlab

Après l'ouverture de la fenêtre principale du programme MATLAB (figure II.1), il faut démarrer le programme SIMULINK. Cela peut se faire de trois manières différentes :

- Cliquez sur l'icône  (Simulink) dans la fenêtre de commande de MATLAB.
- Dans la fenêtre de commande de MATLAB tapez Simulink et appuyer sur la touche « entrer » du clavier.
- Exécuter la commande Open…dans le menu File et ouvrir le fichier modèle (*mdl-file*).

Cette action ouvre la fenêtre Simulink Library Browser qui permet l'accès à la bibliothèque Simulink (figure II.2) ainsi qu'à d'autres bibliothèques (Control system, par exemple). Simulink comme chaque bibliothèque importante, est subdivisée en compartiments : Continuous, Discret,…Sources qui contiennent les blocs.

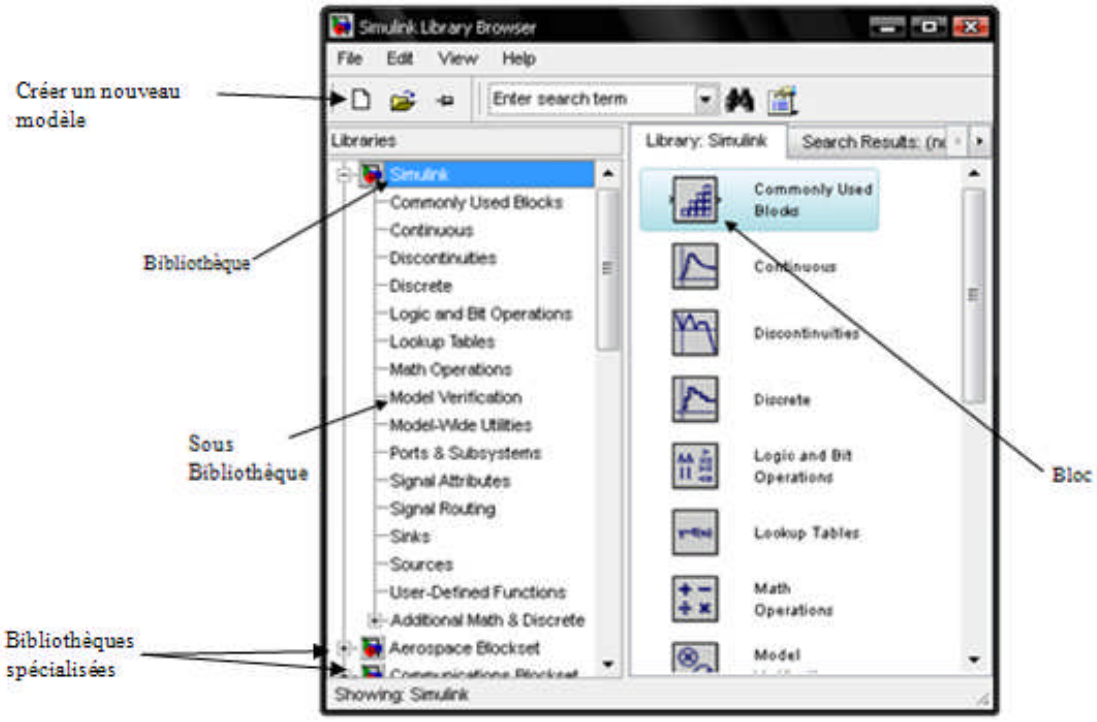


Figure II.2 : Liste des librairies de Simulink

La bibliothèque principale Simulink est mise en relief sur la figure II.2 (dans la partie gauche de la fenêtre) et ses chapitres dans la partie droite de la fenêtre.

La liste des différentes parties de la bibliothèque Simulink est représentée sous forme d'arbre, et les règles de leur utilisation sont communes à toutes les listes de telle forme. Le choix d'une partie à gauche, entraine l'affichage à droite de la fenêtre de son contenu. Pour travailler dans une fenêtre on utilise les commandes regroupées dans le menu. Le menu de l'afficheur des bibliothèques contient les points suivants :

- ✓ *File* (fichier)- utilisation des fichiers des bibliothèques,
- ✓ *Edit* (édition)- ajout des blocs et leurs recherches (par nom),
- ✓ *View* (affichage)- commande de l'affichage des éléments de l'interface,
- ✓ *Help* (aide)- affichage de la fenêtre d'aide.

Pour travailler avec l'afficheur des bibliothèques, on peut aussi utiliser les icônes qui se trouvent sur le panneau des instruments. Les boutons du panneau des instruments jouent les rôles suivants :

Créer un nouveau S-modèle (ouvrir une nouvelle fenêtre du modèle),

Ouvrir un des S-modèles existants,

Changer les propriétés de la fenêtre de l'afficheur « au dessus de toutes les fenêtres ». Un second click sur ce bouton annule ce régime,

Recherche du bloc par nom (par les premiers symboles du nom). Après avoir retrouver le bloc, dans la fenêtre de l'afficheur s'ouvre la partie correspondante de la bibliothèque, et le bloc

sera sélectionné. Si le bloc avec un tel nom est absent, alors dans la fenêtre de l'afficheur il sera mentionné l'information *Not found* « le nom de bloc » (le bloc n'est pas trouvé).

On aboutit à l'écran représenté par la fenêtre d'exploitation Simulink library contenant les différentes librairies que l'on peut ouvrir en double cliquant dessus :

- **Continuous**

Cette bibliothèque (figure II.3) propose des dérivateur, des intégrateurs, des fonctions de transfert, des blocs de retard pour construire des systèmes à temps continu.

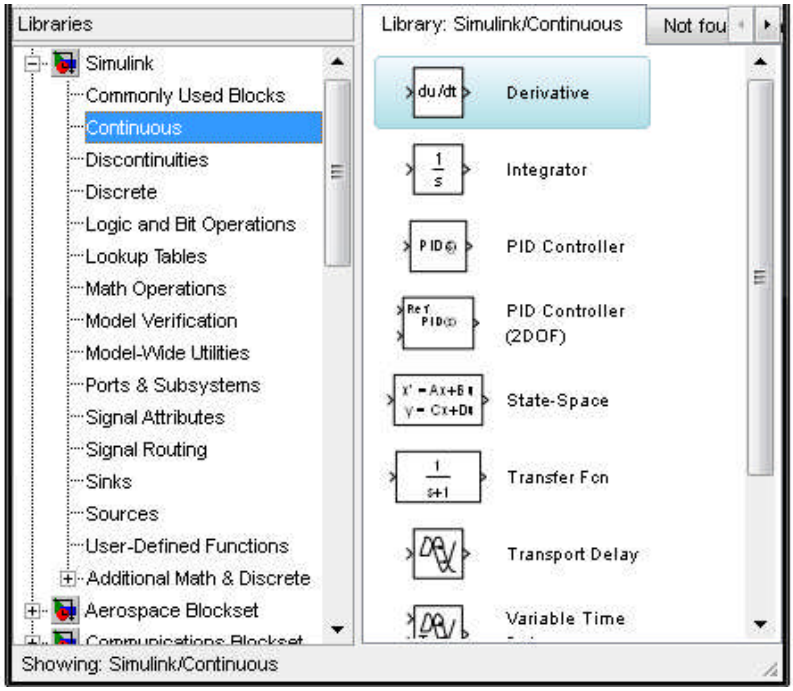


Figure II.3 : La bibliothèque Continuous

- **Math Operations**

Cette bibliothèque (figure II.4) fournit des opérations mathématiques sur les signaux telles que des gains, des multiplicateurs, des sommateurs.

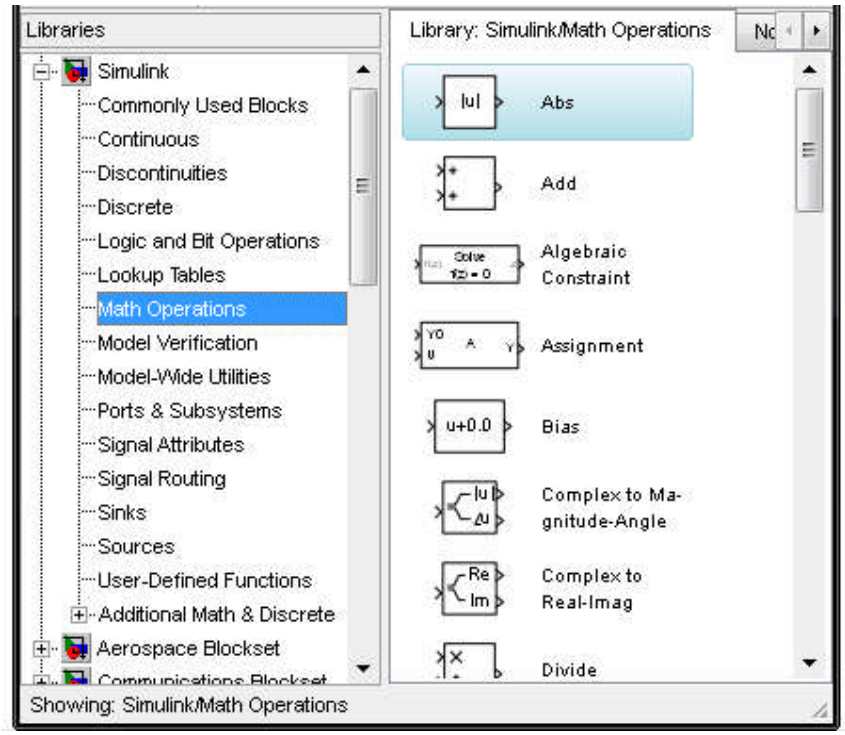


Figure II.4 : La bibliothèque Math Operations

- **Sources**

Cette bibliothèque (figure II.5) contient des éléments générateurs de signaux tels que saut unité, sinusoïde, carré, ramp, fichiers de points (*.mat), variable MATLAB, bruit, séquences, le temps courant de la simulation (horloge), des digital clock, etc., sans oublier le générateur de signal lui-même. Sans parler du fait que l'on peut soi-même créer son propre générateur de signal, on note ici qu'il y a vraiment intérêt à paramétrer ces blocs à l'aide de variables définies proprement dans MATLAB.

En sélectionnant par un double clic, le bloc sources, on ouvre la librairie des sources. Cette fenêtre contient des blocs qui vont permettre de simuler différents types de sources.

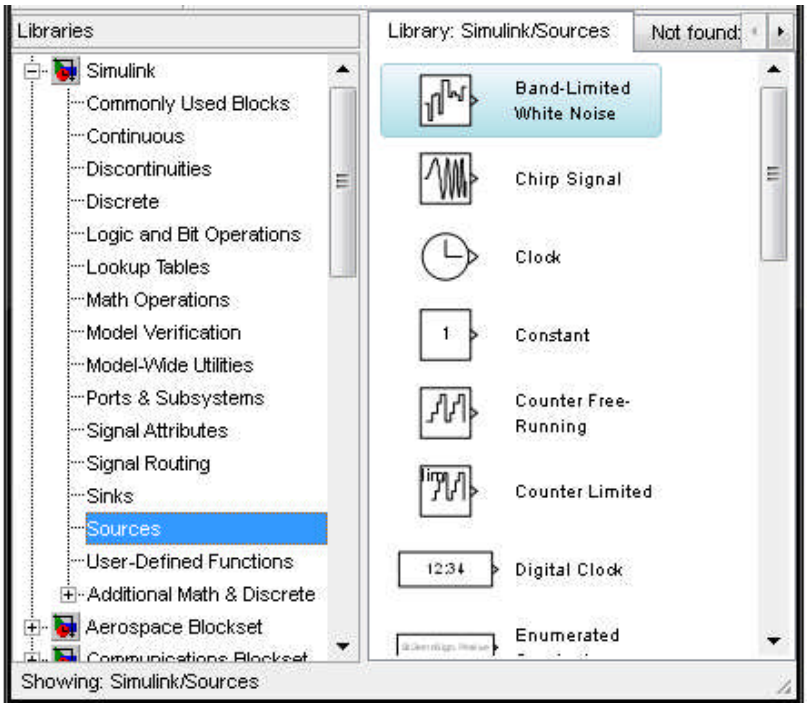


Figure II.5 : La bibliothèque Sources

- **Sinks**

Cette bibliothèque (figure II.6) contient des blocs permettant d'envoyer des données qui peuvent être traité par un fichier *.m ainsi que des éléments de visualisation et d'affichage (scope, display).

L'oscilloscope est considéré plutôt comme un gadget permettant de vérifier le bon fonctionnement de la simulation. La combinaison de l'oscilloscope avec un multiplexeur analogique (bibliothèque Connections) permet de grouper plusieurs signaux sur une seul ligne de créer ainsi un oscilloscope multi trace. La présence d'un grand nombre d'oscilloscopes dans un même schéma ralentit considérablement la simulation, comme d'ailleurs la sauvegarde systématique des signaux dans des variables MATLAB.

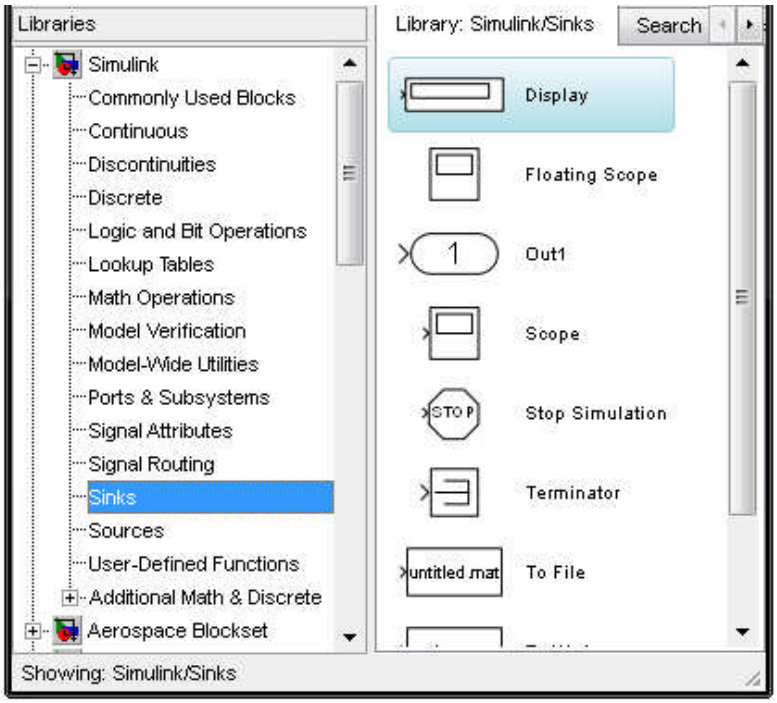


Figure II.6 : La bibliothèque Sinks

- **Logic and Bit Operations**

Cette bibliothèque (figure II.7) contient des détecteurs de changement, de diminution ainsi d'augmentation de la valeur du signal, des opérateurs logiques, des comparateur à zéro, des comparateurs par rapport à une constante, etc.

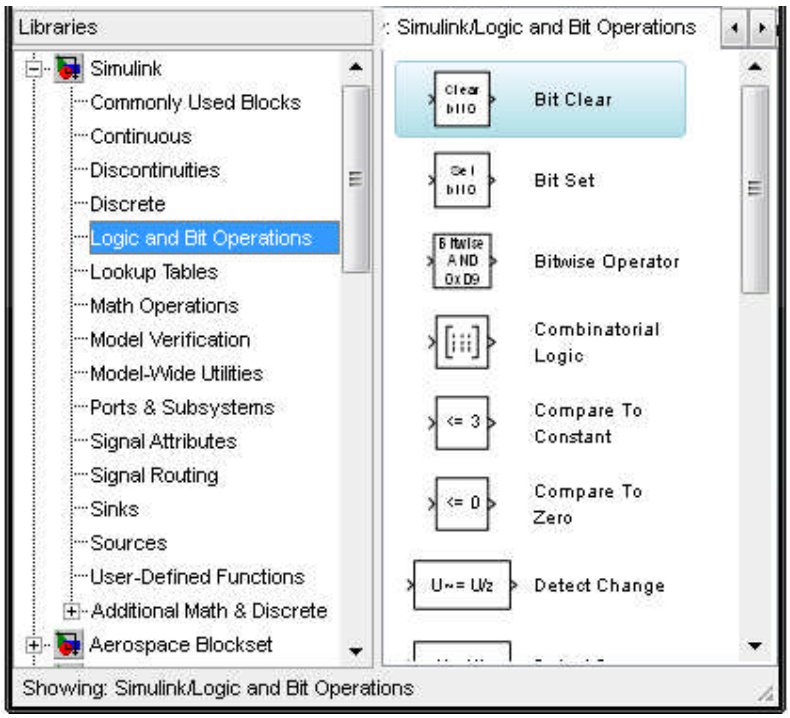


Figure II.7 : la bibliothèque Logic and Bit Operations

- **User-Defined Functions**

La bibliothèque User-Defined Functions (figure II.8) contient des fonctions MATLAB, ainsi des blocs de fonctions MATLAB pour appeler la fonction MATLAB ou une expression à une entrée, des Function-S et des constructeurs de Function-S.

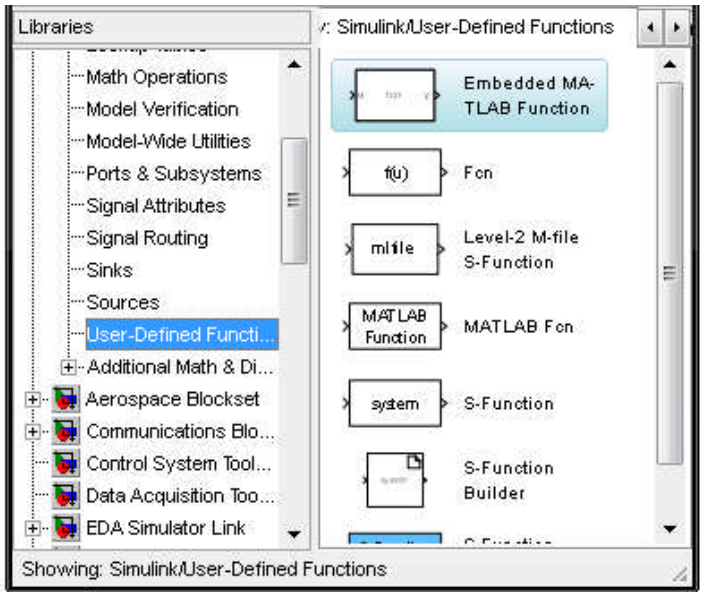


Figure II.8 : la bibliothèque User-Defined Functions

- **Signal Routing**

Cette bibliothèque (figure II.9) contient des multiplexeurs, des démultiplexeurs, Contrôleur d'environnement, Commutateur manuel, des commutateurs multiport pour choisir entre les entrées multiples de bloc, mémoire de magasin de données (Data Store Memory), etc.

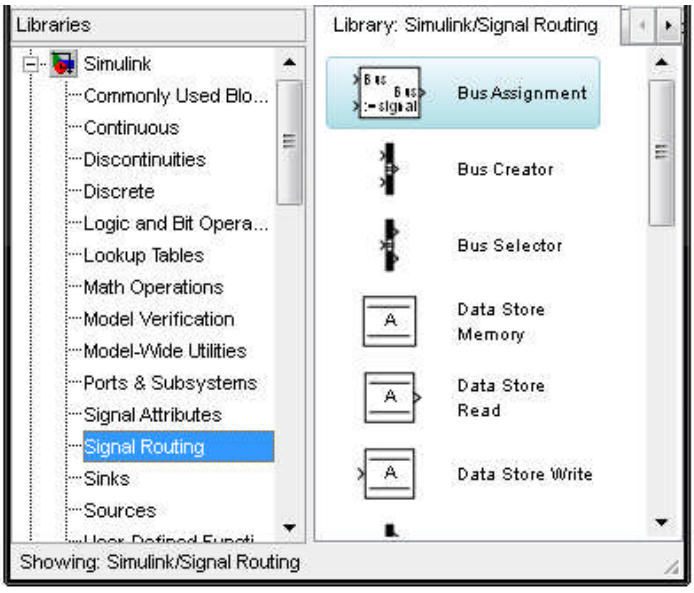


Figure II.9 : La bibliothèque Signal Routing

- **Discrete**

Dans la bibliothèque Discrète (figure II.10) on trouve des fonctions de transfert discrètes, des mémoires, des filtres discrets, des intégrateurs de temps- discret, etc.

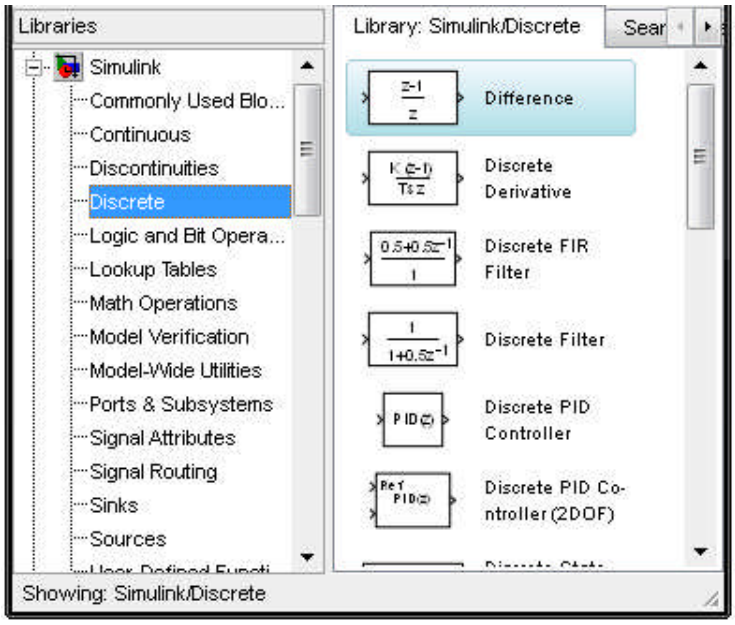


Figure II.10 : La bibliothèque Discrete

- **Ports and Subsystems**

Cette bibliothèque (figure II.11) contient des blocs d'entrées et de sorties, des modèles pour inclure le modèle en tant que bloc dans un autre modèle, les sous-systèmes, les fonctions d'appel d'un sous-système, la fonction If, For, etc.

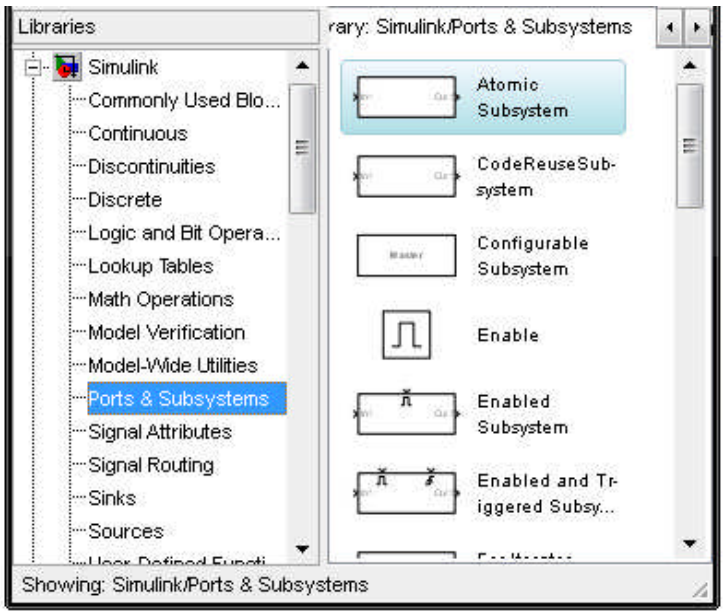


Figure II.11 : La bibliothèque Ports and Subsystems

- **Utility Blocks**

Cette bibliothèque (figure II.12) contient des blocs qui permet de trouver le retard entre deux signaux, mesurer le rapport signal/bruit (SNR), convertir les entiers en un groupe de bits, convertir un groupe de bits en un nombre entier, etc.

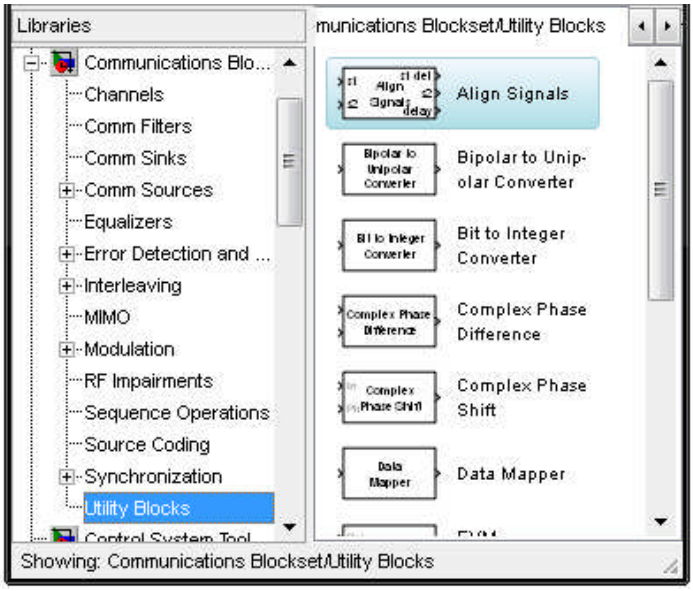


Figure II.12 : La bibliothèque Utility Blocks

II.3 Description des blocs de Simulink :

Nous attardons uniquement sur les blocs les plus utiles, en particulier sur ceux dont nous avons eu besoin pour réaliser notre projet. Des informations approfondies sont disponibles dans le manuel d'usager de Simulink disponible au site Internet de Mathworks.Inc :

(**http://www.mathworks.com/access/helpdesk/help/toolbox/simulink/simulink.shtml**)

Notre projet utilise des blocs, nous allons les citer selon les bibliothèques qui comprennent ces blocs.

II.3.1 La bibliothèque « Simulink » :

Cette bibliothèque comprend des blocs d'utilisation générale.

II.3.1.1 Bloc Constant (Constante)

Le bloc Constante de la librairie « Sources » produit un signal de la valeur constante, est montré à la figure II.13.

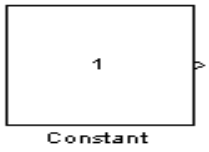


Figure II.13 : bloc « Constant »

Selon la dimension du paramètre de la valeur Constante, le bloc peut indiquer un scalaire, un vecteur, ou une matrice. Il est possible de fixer, via la fenêtre ci-dessous (figure II.14), la valeur de la constante.

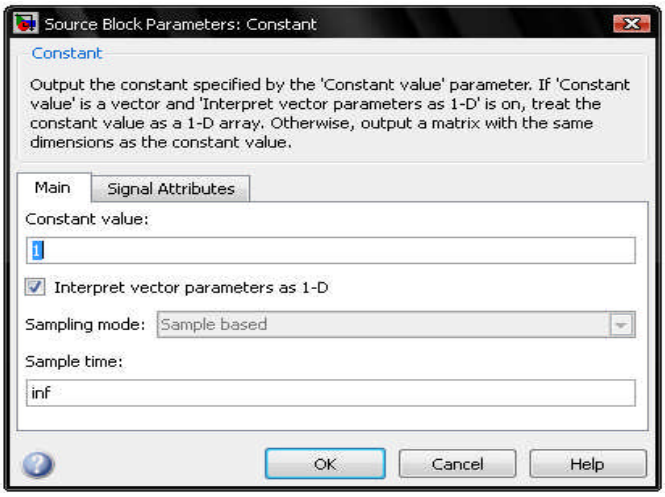


Figure II.14 : configuration du bloc « Constant»

II.3.1.2 Bloc product (Produit)

Le Bloc Produit de la librairie « Math Operations » sert à multiplier des entrées. Il est montré à la figure II.15.

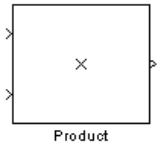


Figure II.15 : bloc « Product »

Le paramétrage du bloc « Product » est montré à la figure II.16.

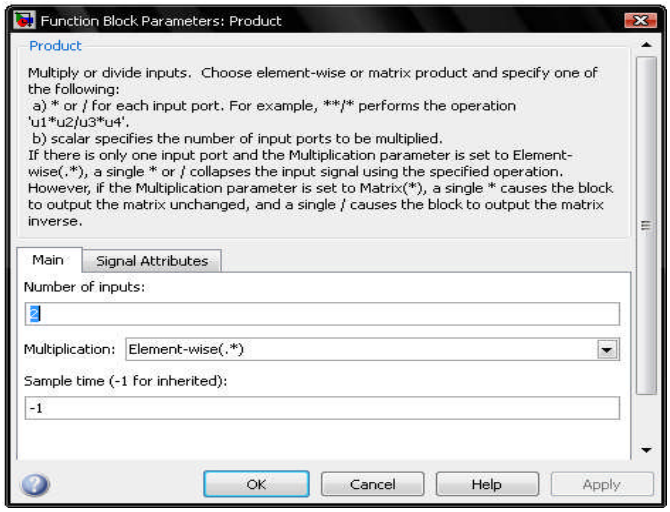


Figure II.16 : paramétrage du bloc « Product »

II.3.1.3 Bloc Sum (Somme)

Pour réaliser la somme entre le signal de consigne et le signal de sortie, on choisit le bloc Sum (figure II.17), de la librairie « Math Operations ».

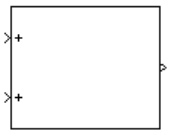


Figure II.17 : bloc « Sum »

En cliquant deux fois sur cet icône : une boîte à dialogue (figure II.18) apparaitra et nous permet de définir le nombre d'entrée et de transformer ce bloc en soustracteur par des signes « + -».

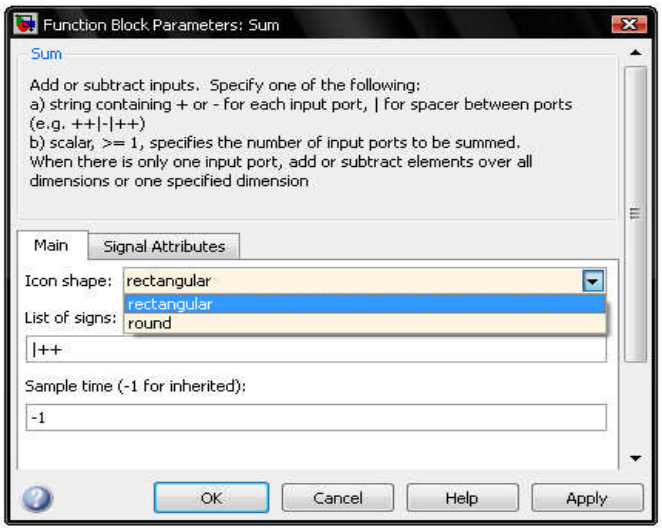


Figure II.18 : boîte de dialogue de bloc « Sum »

II.3.1.4 Bloc Mux (Multiplexeur)

Il est possible de regrouper plusieurs signaux en utilisant le bloc Mux. Ceci rend ainsi un diagramme plus lisible. Le multiplexeur Mux, de la librairie « Signal Routing », permet d'envoyer simultanément plusieurs signaux vers un fichier ou un instrument de visualisation. Le bloc est montré à la figure II.19.

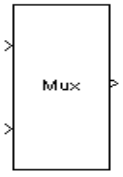


Figure II.19 : bloc « Mux »

On effectue un double clic sur le composant Mux, la fenêtre de paramétrage (figure II.20) s'ouvre. On tape les valeurs désirées : ici la valeur 2 pour indiquer 2 entrées et on indique le

modèle d'affichage de l'icône du bloc. On ferme cette fenêtre par Close, les nouvelles valeurs sont prises en compte.

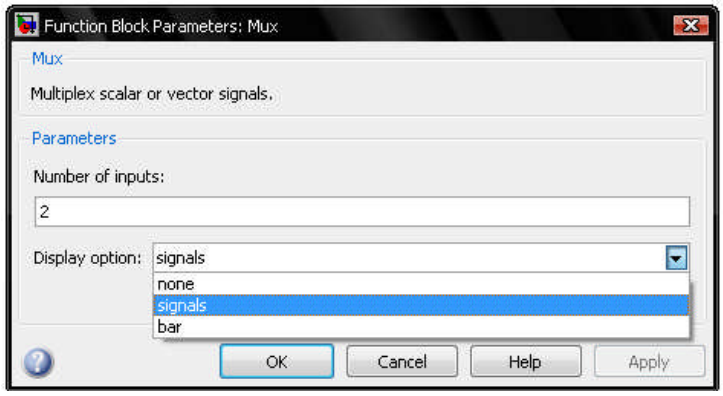


Figure II.20 : paramétrage du bloc « Mux »

II.3.1.5 Bloc Demux (Demultiplexeur)

Le démultiplexeur se trouve dans la librairie « Signal Routing » permet d'extraire les composants d'un signal d'entrée et produit les composants en tant que signaux séparés. Les signaux de sortie sont commandés de haut en bas dans la porte de sortie. Le bloc est montré à la figure II.21.

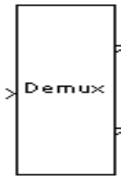


Figure II.21 : bloc « Demux »

Un double clic nous permet de configurer le nombre de voies (figure II.22)

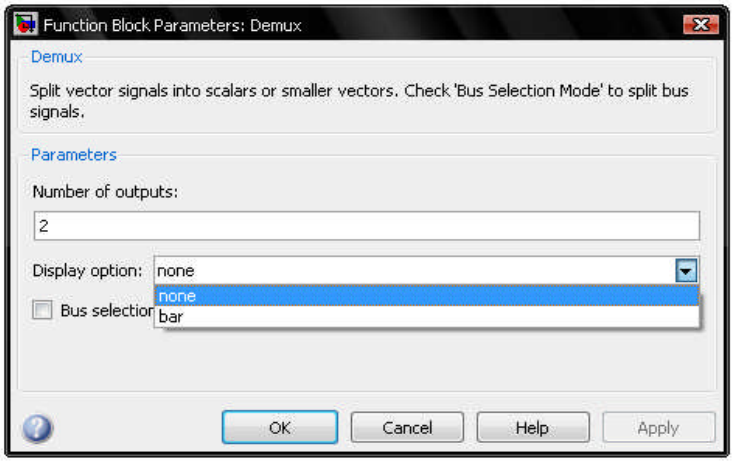


Figure II.22 : boîte de dialogue de bloc « Demux »

II.3.1.6 Bloc Scope (Oscilloscope)

Le bloc scope, outil de visualisation du signal, permet de visualiser les signaux en fonction du temps produit pendant la simulation, se trouve dans la librairie « Sinks ».

Il permet la mesure des signaux, la loupe x agrandit l'axe X, la loupe y agrandit l'axe Y, la paire de jumelle permet l'adaptation d'échelle automatique, et l'icône autoscale permet d'ajuster les échelles pour voir le signal complet. On y a accès que lors de la simulation. On peut imprimer les résultats. Il est montré à la figure II.23.

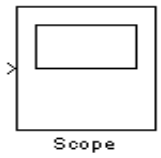


Figure II.23 : bloc « Scope »

Celui-ci permet de visualiser les courbes résultant de la simulation et aussi d'enregistrer ces données pour qu'elles soient récupérées dans Matlab.

-Double- cliquer sur le bloc « Scope ». Une fenêtre s'ouvre (voire figure II.24) : c'est là que seront affichées les courbes suite à la simulation.

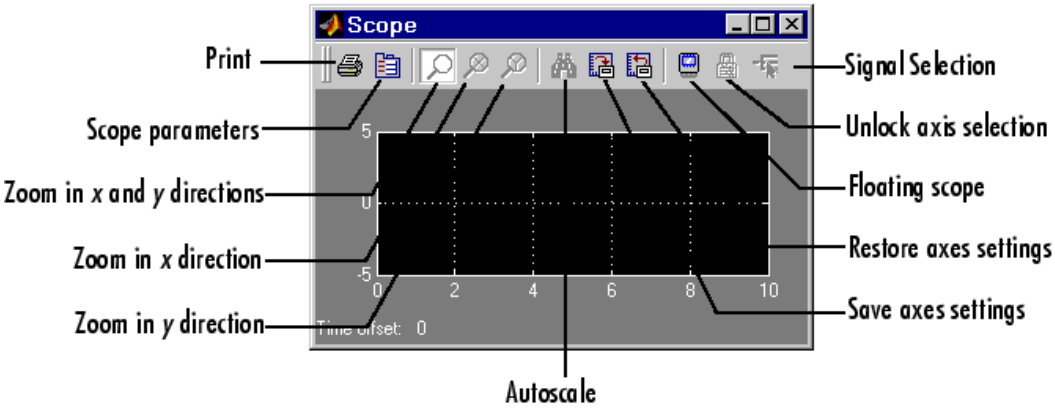


Figure II.24 : la fenêtre du « Scope »

-Cliquer droite sur la fenêtre de l'oscilloscope pour modifier les propriétés de l'axe Y (figure II.25)

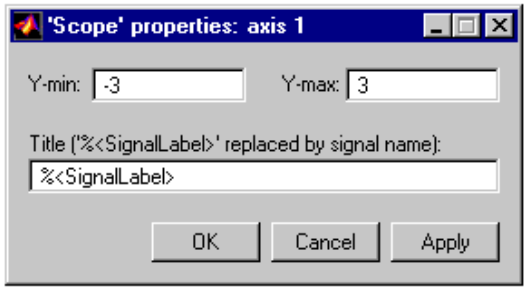
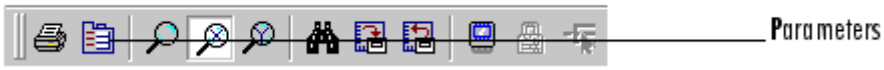


Figure II.25 : boîte de dialogue de propriétés des axes

Il est possible d'avoir plusieurs entrées dans le bloc « Scope ». Pour ce faire, il s'agit :

-Double-cliquer sur le bouton **Parametres** de la fenêtre Scope



- Aller dans les deux ongles *General Data History* (voir figure II.26 et II.27),
- Sélectionner *Save data* to workspace,
- Au besoin, modifier le nom de variable « Scope Data »pour le nom voulu,
- Choisir le format *Array*,
- Cliquer sur l'onglet **General** et configurer les paramètres du bloc Scope suivant la figure II.26,
- Cliquer sur le bouton **OK** pour confirmer.

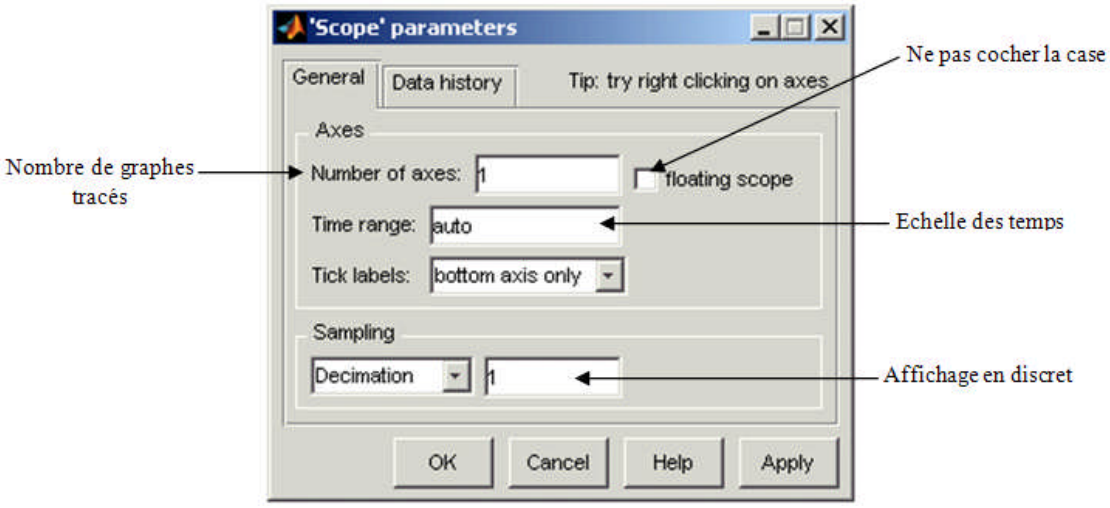


Figure II.26 : onglet permettant d'ajuster les paramètres du « Scope »

L'oscilloscope « Scope » ne conserve que les dernières données de la simulation. Il est possible de fixer le nombre de points conservés en l'indiquant à la suite de la mention **Limit data points to last** de la fenêtre **Data History** (figure II.27). Par défaut, le nombre de points est fixé à 5000, ce qui s'avère généralement adéquat.

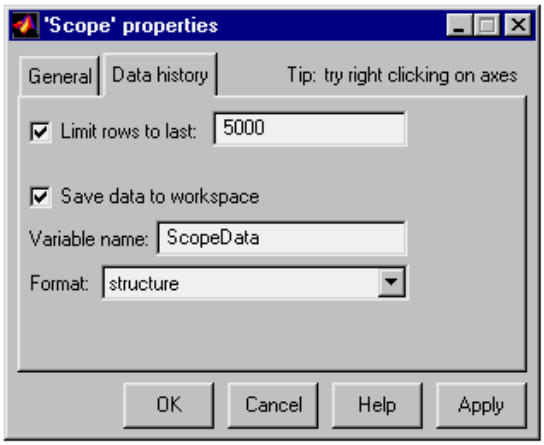


Figure II.27 : onglet permettant d'ajuster les paramètres du « Scope »

II.3.1.7 Bloc Display (Display)

Le bloc Display de la librairie « Sinks » sert à afficher la valeur de l'entrée, est montré à la figure II.28.

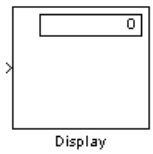


Figure II.28 : bloc « Display »

Le type de format de l'affichage est sélectionné comme suit :

- Format short : affiche une valeur de 5 chiffres avec une virgule fixe.
- Format long : affiche une valeur de 15 chiffres avec une virgule fixe.
- Format short-e : pour afficher les résultats numériques en notation scientifique.
- Format long-e : affiche une valeur de 16 chiffres avec une virgule flottante.
- Format banque : affiche une valeur en dollars fixes et **les cent formats**.

Le paramétrage de ce bloc est donné à la figure II.29.

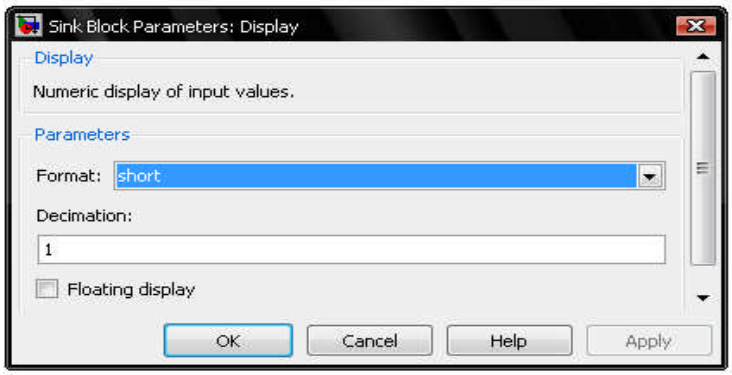


Figure II.29 : boîte de dialogue du bloc « Display »

II.3.1.8 Bloc Zero-order hold (Bloqueur d'Ordre Zéro)

Le bloc bloqueur d'ordre zéro de la librairie « Discrete » est présenté à la figure II.30.

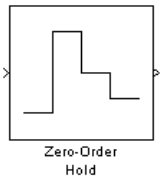


Figure II.30 : bloc « Zero-order hold»

Le bloc du bloqueur des échantillons bloque le signal d'entrée pour une période d'échantillonnage donnée. Le paramétrage de ce bloc est donné à la figure II.31.

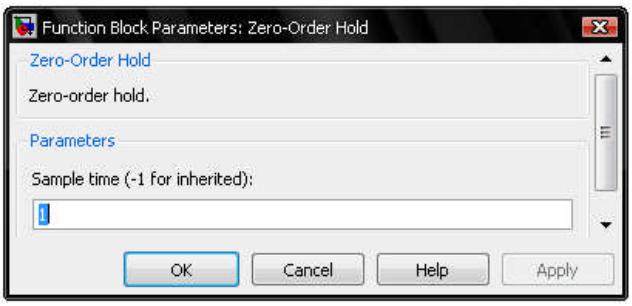


Figure II.31 : paramétrage du bloc « Zero-order hold»

II.3.1.9 Bloc Inport (Bloc d'Entrée)

Ce bloc de la librairie « Ports & Subsystem » permet de créer un bloc d'entrée pour un sous-système ou une entrée externe. Celui-ci est montré à la figure II.32.

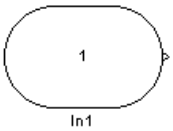


Figure II.32 : bloc « Inport »

Le bloc d'entrée sert de lien entre l'extérieur d'un système et l'intérieur d'un système. Le paramétrage du bloc Inport est montré à la figure II.33.

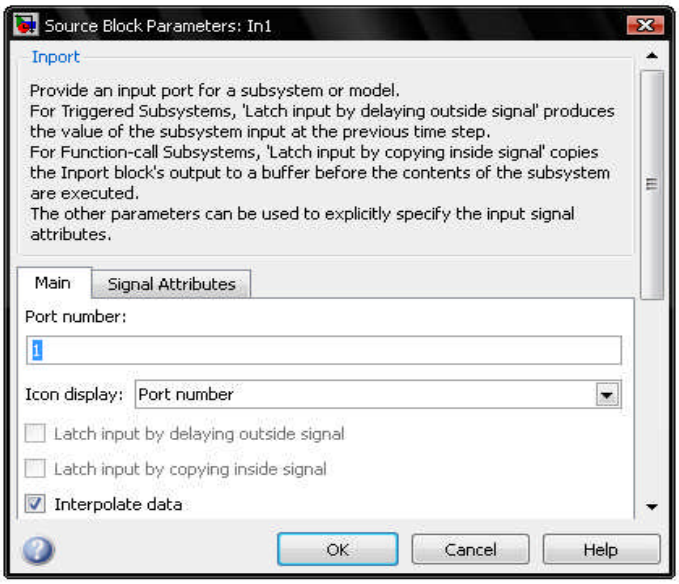


Figure II.33 : boîte de dialogue de bloc « Inport »

On spécifie le nombre, la dimension du port du bloc Inport et la période d'échantillonnage du signal d'entrée.

II.3.1.10 Bloc Outport (Bloc de Sortie)

Ce bloc de la librairie « Ports & Subsystem » permet de créer un bloc de sortie pour un sous-système ou une production externe. Le bloc est montré à la figure II.34.

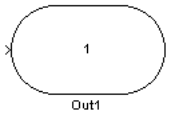


Figure II.34 : bloc « Outport »

Le bloc de sortie est le lien d'un système à une destination à l'extérieur du système. La boîte de dialogue est montrée à la figure II.35.

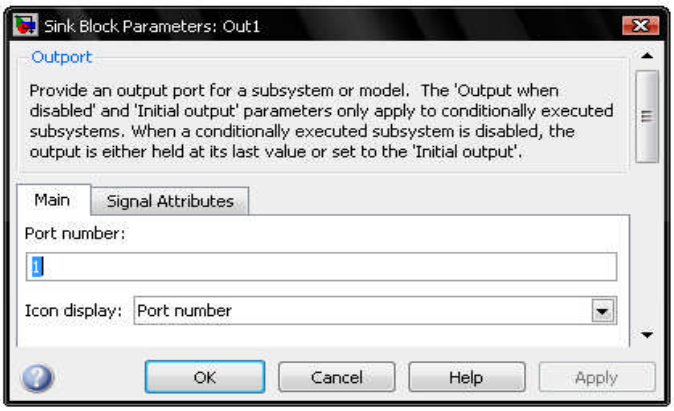


Figure II.35 : boîte de dialogue de bloc « Outport »

On spécifie le nombre de port du bloc Outport. La sortie désactivée, et la sortie initiale sont utilisées avec des systèmes spéciaux.

II.3.1.11 Bloc Digital Clock (Horloge Digitale)

Ce bloc de la librairie « Sources » produit le temps de simulation seulement à l'intervalle de prélèvement indiqué, la valeur par défaut est **1** seconde. Le bloc est montré à la figure II.36.

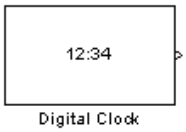


Figure II.36 : bloc « Digital Clock »

La boîte de dialogue est montrée à la figure II.37.

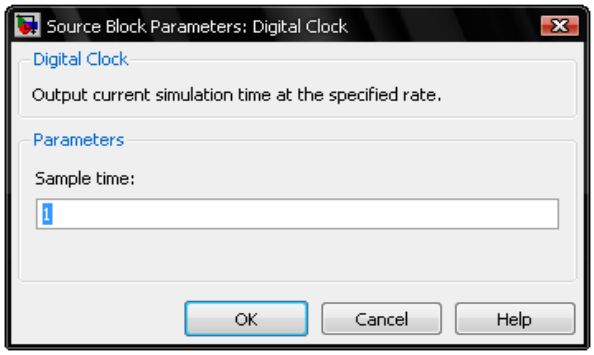


Figure II.37 : boîte de dialogue de bloc « Digital Clock »

II.3.1.12 Bloc Ground (La Masse)

Ce bloc de la librairie « Sources » peut relier les blocs dont les ports d'entrée ne se relient pas à d'autres blocs. Si on exécute une simulation avec des blocs ayant les ports non liés d'entrée, le logiciel de Simulink indique des avertissements. L'utilisation de la masse peut empêcher les avertissements. Le bloc la masse produit un signal avec la valeur nulle. Le bloc est montré à la figure II.38.

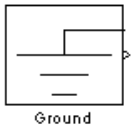


Figure II.38 : bloc « Ground »

La boîte de dialogue est montrée à la figure II.39.

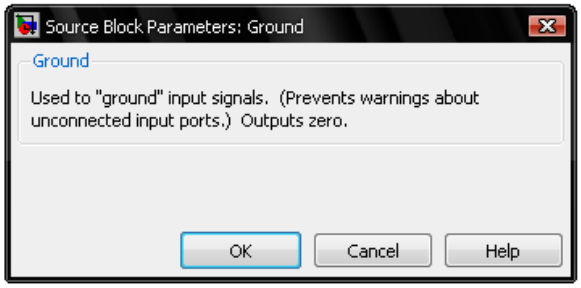


Figure II.39 : boîte de dialogue de bloc « Ground »

II.3.1.13 Bloc Gain (Gain)

Le bloc de Gain de la librairie «Math Operations » multiplie l'entrée par une valeur constante (gain). L'entrée et le gain mettent en boîte chacune soient une grandeur scalaire, un vecteur, ou une matrice. Le bloc est montré à la figure II.40.

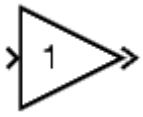


Figure II.40 : bloc « Gain »

La boîte de dialogue est montrée à la figure II.41.

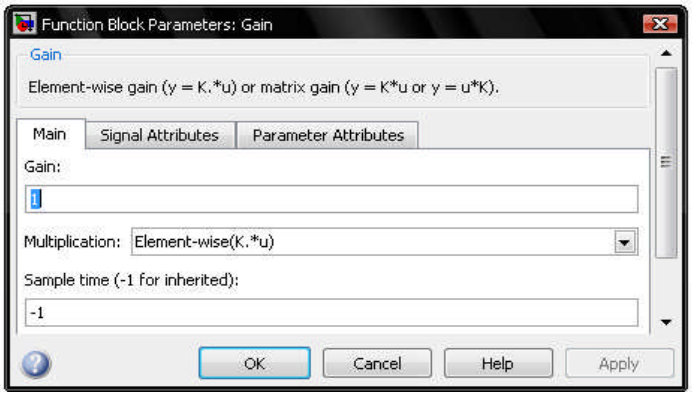


Figure II.41 : boîte de dialogue de bloc « Gain »

II.3.1.14 Bloc Band-Limited White Noise (Bruit Blanc)

Le bloc de bruit blanc band-Limited de la librairie «Sources» produit des nombres aléatoires normalement distribués pour présenter le bruit blanc dans les systèmes continus ou hybrides.
Le bloc est montré à la figure II.42.

Figure II.42 : bloc « Band-Limited White Noise»

La boîte de dialogue est montrée à la figure II.43.

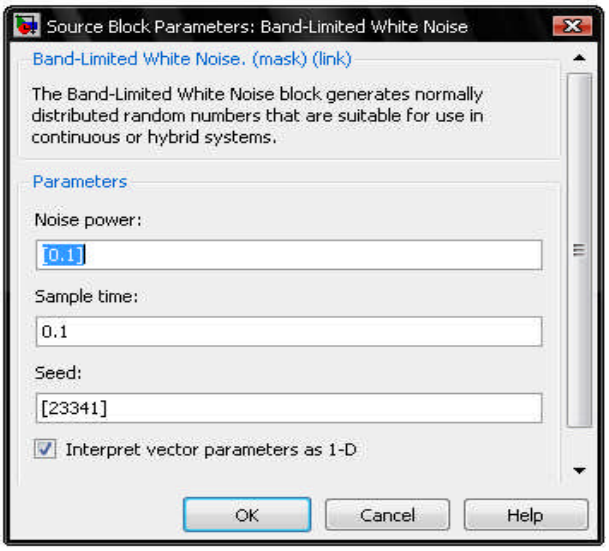


Figure II.43 : boîte de dialogue de bloc « Band-Limited White Noise»

II.3.1.15 Bloc Pulse Generator (Générateur d'impulsion)

Le bloc générateur d'impulsion de la librairie «Sources» produit des impulsions carrées à des

intervalles réguliers. Les paramètres de la forme d'onde du bloc, **amplitude**, **largeur d'impulsion**, **période**, et **la phase retardée**, déterminent la forme de la forme d'onde de sortie. Le bloc est montré à la figure II.44.

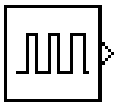


Figure II.44 : bloc « Pulse Generator»

La boîte de dialogue est montrée à la figure II.45.

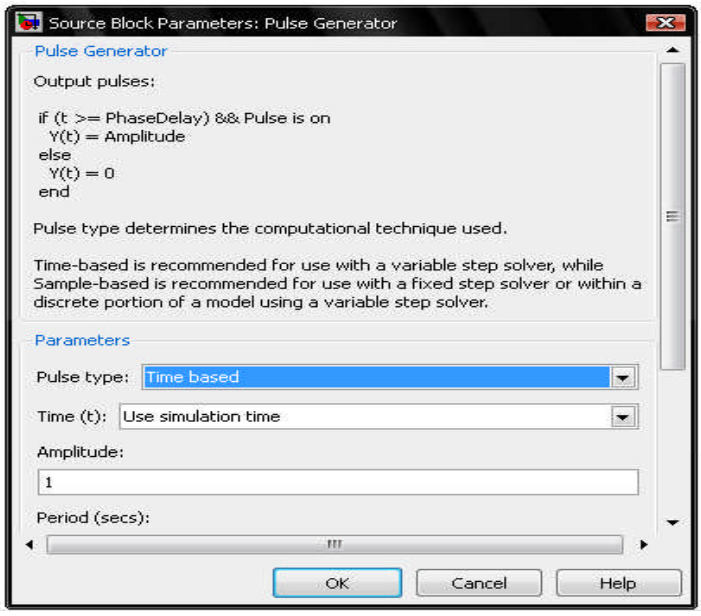


Figure II.45 : boîte de dialogue de bloc « Pulse Generator»

II.3.1.16 Bloc Logical Operator (Opérateur logique)

Le bloc opérateur logique de la librairie «Logic and Bit Operations» effectue l'opération logique indiquée sur ses entrées. Une valeur d'entrée est VRAI(1) si elle est non zéro et FAUX(0) si elle est zéro. Le bloc est montré à la figure II.46.

Figure II.46 : bloc « Logical Operator »

On choisit l'opération booléenne reliant les entrées à la liste de paramètre **Operator**. Si on choisit **rectangulaire** comme propriété **de forme d'icône**, les mises à jour de bloc pour montrer le nom de l'opérateur choisi. Les opérations soutenues sont données ci-dessous :

Opération	Description
ET(AND)	VRAI si toutes les entrées sont VRAIES
OU(OR)	VRAI si au mois une des entrées est VRAIE
NON ET(NAND)	VRAI si au mois une des entrées est FAUSSE
NON OU(NOR)	VRAI quand aucune entrée n'est VRAIE
XOR (OU EXCLUSIF)	VRAI si un nombre impair d'entrée est VRAI
NXOR (NON OU EXCLUSIF)	VRAI si un nombre pair des entrées est VRAI
NOT (NON)	VRAI si l'entrée est fausse

Tableau II.1 : les opérateurs logiques

Si on choisit **distinctif** comme **forme d'icône**, l'aspect du bloc indique sa fonction. Le logiciel de Simulink montre une forme distinctive pour l'opérateur choisi, conformément aux symboles graphiques standard d'IEEE pour des fonctions logiques :

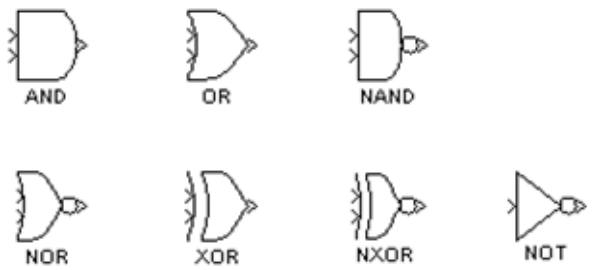


Figure II.47 : les portes logiques

La boîte de dialogue est montrée à la figure II.48.

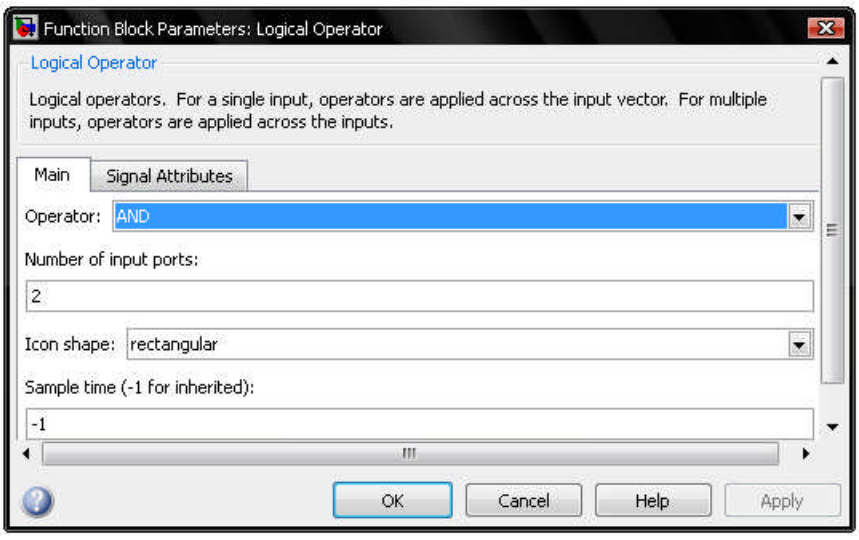


Figure II.48 : boîte de dialogue de bloc « Logical Operator »

Le bloc opérateur logique accepte des vrais signaux de n'importe quel type de données numérique soutenu par le logiciel de Simulink, y compris les types de données à point fixe.

II.3.1.17 Bloc Relational Operator (Opérateur Relationnel)

Le bloc Opérateur Relationnel de la librairie «Logic and Bit Operations» effectue l'opération indiquée sur ses entrées. Le bloc est montré à la figure II.49.

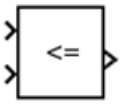


Figure II.49 : bloc « Logical Operator »

Les opérations soutenues sont données ci-dessous :

Operation	Description
==	VRAI si la première entrée est égale à la deuxième entrée
~=	VRAI si la première entrée n'est pas égale à la deuxième entrée
<	VRAI si la première entrée est moins que la deuxième entrée
<=	VRAI si la première entrée est inferieure ou égale à la deuxième entrée
>=	VRAI si la première entrée est supérieure ou égale à la deuxième entrée
>	VRAI si la première entrée est plus grande que la deuxième entrée

Tableau II.2 : les opérations relationnelles

La boîte de dialogue est montrée à la figure II.50.

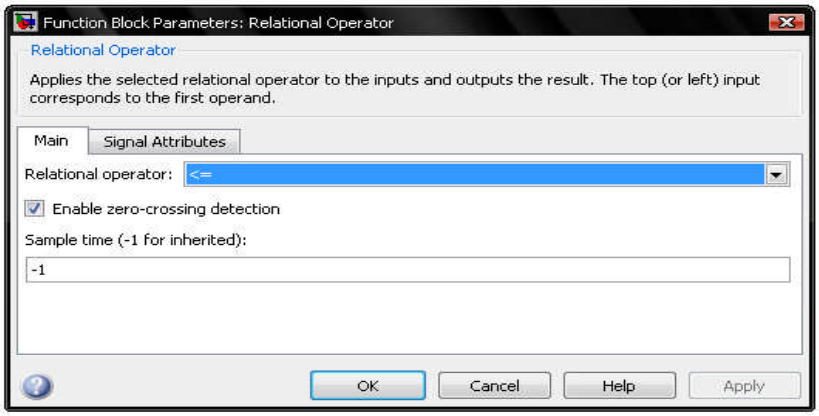


Figure II.50 : boîte de dialogue de bloc « Logical Operator »

Le bloc d'opérateur Relationnel accepte des vrais ou complexes signaux de n'importe quel type de données soutenu par le logiciel de Simulink, y compris les types de données à point fixe et énumérés.

II.3.1.18 Bloc S-Function (Fonction-S)

Le bloc Fonction-S de la librairie «User-Defined Functions» permet d'accéder aux Fonctions-S d'un schéma fonctionnel. Les fonctions sont des enchaînements de commandes **Matlab** regroupées sous un nom de fonction permettant de commander leur exécution. De manière générale, la syntaxe de définition d'une fonction externe est :

```
function [y1,...,ym]=toto(x1,....,xn)
```

Le bloc est montré à la figure II.51.

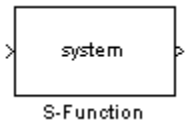


Figure II.51 : bloc « S-Function»

La boîte de dialogue est montrée à la figure II.52.

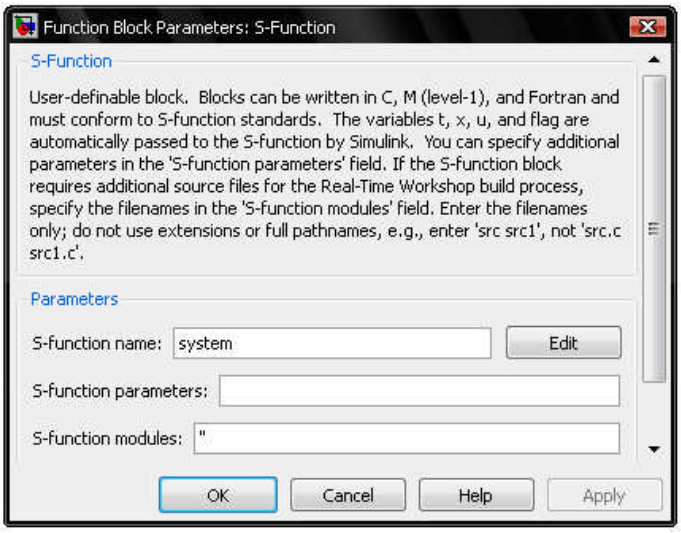


Figure II.52 : boîte de dialogue de bloc « S-Function»

Le type des données d'entrée dépend de l'exécution du bloc Fonction-S.

II.3.1.19 Bloc Subsystem (Sous-Système)

Le bloc Sous-système de la librairie « Ports & Subsystems »représente un sous-ensemble du système qui le contient. Le bloc est montré à la figure II.53.

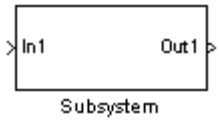


Figure II.53 : bloc « Subsystem »

On peut créer un Sous-système de la manière suivante :

- Copiez le bloc Sous-système de la bibliothèque Ports & Subsystems dans votre modèle. On peut alors ajouter des blocs en ouvrant le bloc de Sous-système (voir figure II.54) et en copiant des blocs dans sa fenêtre.

Le nombre des portes d'entrée dessinés sur l'icône du bloc de Sous-système correspond au nombre de blocs d'Inport dans le Sous-système. De même, le nombre des portes de sortie dessinés sur le bloc correspond au nombre de blocs d'Outport dans le Sous-système.

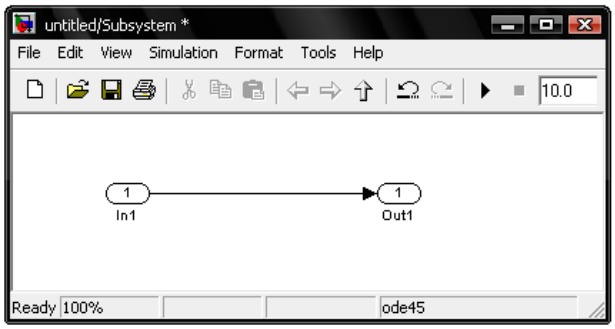


Figure II.54 : un Sous-système

II.3.1.20 Bloc Multiport Switch (Commutateur Multiport)

Le bloc Commutateur Multiport de la librairie « Signal Routing » choisit entre un certain nombre d'entrées. Le bloc est montré à la figure II.55.

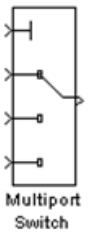


Figure II.55 : bloc « Multiport Switch »

La première entrée(en haut) est l'entrée de commande et les autres entrées sont des entrées de données. La valeur de l'entrée de commande détermine quelle source de données à passer par la porte de sortie.

Si l'entrée de commande n'est pas une valeur entière, le Commutateur Multiport tronque la valeur à l'entier le plus proche et émet un avertissement. Si l'entrée de commande (tronquée) est inférieure à un ou supérieure au nombre des portes d'entrée, le commutateur émet un hors-limite d'erreur. Sinon, le commutateur passe à l'entrée de données qui correspond à l'entrée de commande (tronquée).

La boîte de dialogue est montrée à la figure II.56.

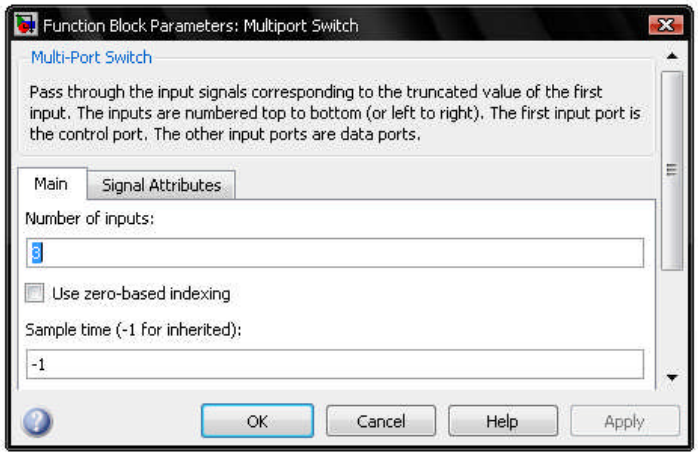


Figure II.56 : boîte de dialogue de bloc « Multiport Switch »

Les données peuvent être des entrées scalaires ou vectorielles. L'entrée de commande peut être un scalaire ou un vecteur. Le bloc de sortie est déterminé par les règles suivantes:

- Si les entrées sont scalaires, le résultat est un scalaire.
- Si le bloc a plus d'une entrée de données, au moins un qui est un vecteur, la sortie est un vecteur. Toutes entrées scalaires sont étendues à des vecteurs.
- Si le bloc a une seule entrée de données et que l'entrée est un vecteur, le bloc de sortie est l'élément du vecteur qui correspond à la valeur tronquée de l'entrée de commande.

L'entrée de commande d'un bloc Commutateur Multiport accepte un signal à valeurs réelles de tout type de données intégrées, sauf booléen. Les entrées de données acceptent des valeurs réelles ou complexes. Toutes les entrées de données doivent être les mêmes données et de type numérique. Le type de signal de la sortie du bloc est le même que celui de ses entrées de données.

II.3.1.21 Bloc Integrator (Intégrateur)

Le bloc d'intégrateur de la librairie « Continuous » produit l'intégrale de son entrée à l'étape de temps courant. Le bloc est montré à la figure II.57.

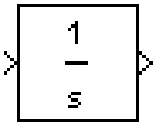
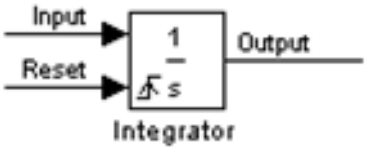


Figure II.57 : bloc « Integrator »

Le bloc peut remettre son état à zéro à l'état initial indiqué basé sur un signal externe. Pour le faire, choisir un des choix ***external reset***. Un port de déclenchement apparaît au-dessous du port de l'entrée du bloc et indique le type de déclenchement.



La boîte de dialogue est montrée à la figure II.58.

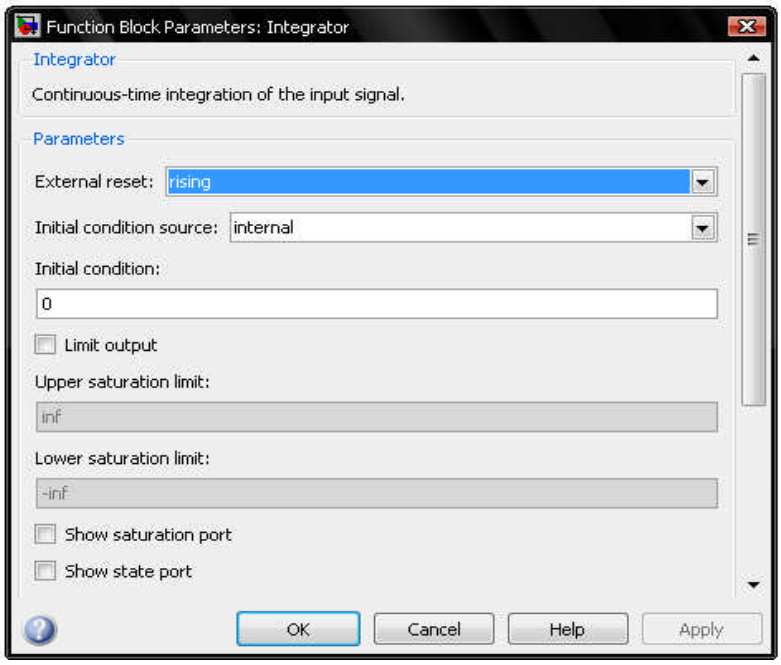


Figure II.58 : boîte de dialogue de bloc « Integrator »

Le bloc d'intégrateur accepte les signaux de sortie du type double sur ses portes de données. Le port externe de remise accepte des signaux de type double ou booléen.

II.3.2 La bibliothèque « Communication » :

Cette bibliothèque comprend des blocs qui sont utilisés en communication.

II.3.2.1 Bloc Bit to integer converter (Convertisseur Bits-Entier)

Ce bloc de la librairie « utility functions » convertit un groupe de bits présents à l'entrée en un nombre entier en sortie. Le bloc est montré à la figure II.59.

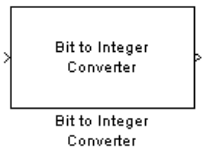


Figure II.59 : bloc « Bit to integer converter »

Si le M est le nombre de bits par entier, alors le bloc convertit chaque groupe de M bits en un nombre entier compris entre 0 et 2^M-1.

La boîte de dialogue est montrée à la figure II.60.

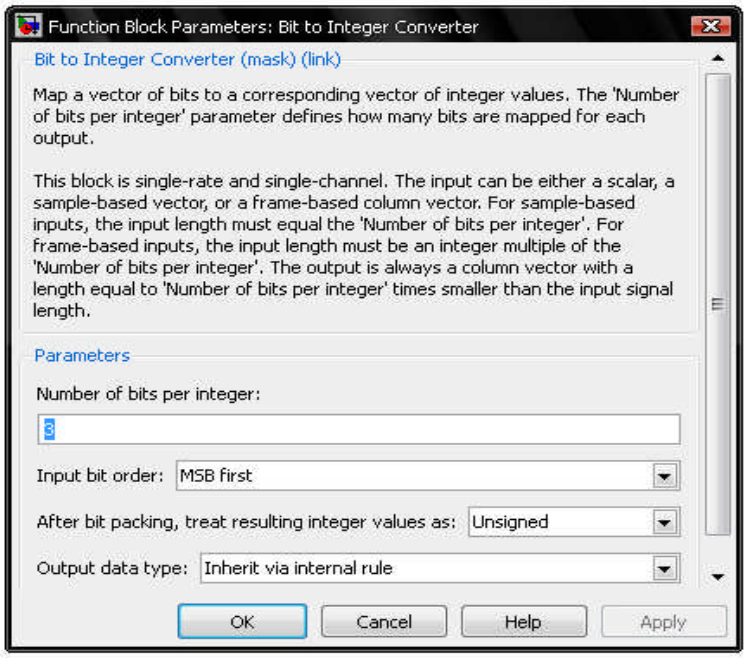


Figure II.60 : boîte de dialogue de bloc « Bit to integer converter »

Exemple :

Si l'entrée est [1 ; 0 ; 1 ; 0 ; 1 ; 1 ; 0 ; 0] et le paramètre du nombre de bits est quatre, alors la sortie est [10 ; 12].

II.3.2.2 Bloc Integer to bit converter (Convertisseur entier-bits)

Ce bloc de la librairie « utility functions » convertit chaque nombre entier ou valeur à point fixe présenté à l'entrée à un groupe de bits présenté à la sortie. Le bloc est montré à la figure II.61.

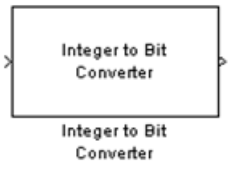


Figure II.61 : bloc « Integer to bit converter »

Si le M est le nombre de bits par entier, alors l'entrée est un nombre entier compris entre 0 et 2^M-1.

La boîte de dialogue est montrée à la figure II.62.

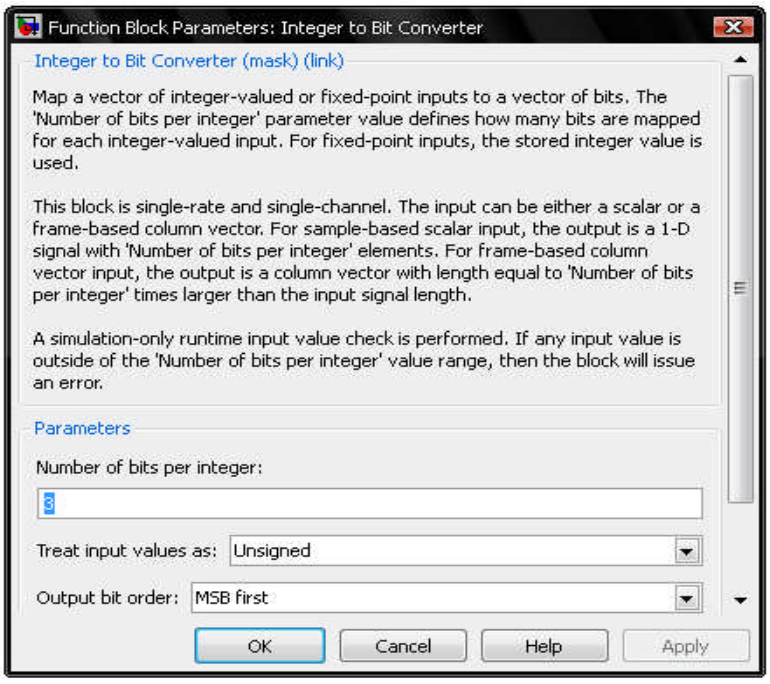


Figure II.62 : boîte de dialogue de bloc « Integer to bit converter »

Le nombre de bits par entier doit être un nombre entier entre 1 et 31.

II.4 Création des schémas blocs :

Les schémas blocs doivent être crées dans un fichier Simulink distinct, de type *.mdl. Pour créer un nouveau fichier de travail Simulink : faire File, New, Model ou cliquer sur l'icône  .

Pour ouvrir un ancien fichier : faire File, open ou cliquer sur l'icône  .

Pour créer le modèle dans le milieu SIMULINK, il est nécessaire de procéder de la manière suivante :

- Créer un nouveau fichier du modèle à l'aide de la commande *File/New/Model*, ou en utilisant le bouton sur le panneau des instruments. Une nouvelle fenêtre de commande Matlab S'ouvrira (voir figure II.1).
- Placer les blocs dans la fenêtre du modèle. Pour cela il est nécessaire d'ouvrir la partie correspondante de la bibliothèque (par exemple, Sources – les sources). Ensuite, en indiquant avec le pointeur de la souris le bloc demandé puis le faire glisser jusqu'à l'endroit approprié dans le fichier de travail (fenêtre du modèle), en maintenant le bouton gauche de la souris enfoncé. Relâcher le bouton de la souris lorsque le bloc est placé à l'endroit voulu (figure II.63). Cet outil utilise la technique de ***drag and drop*** (sélectionner et faire glisser).

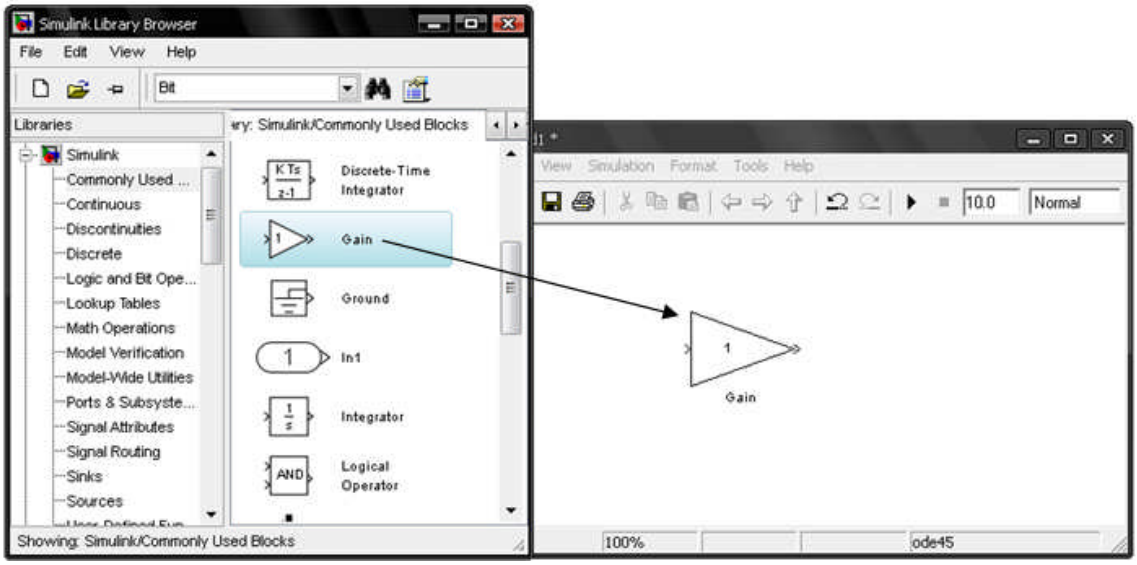


Figure II.63 : insérer un bloc dans un modèle

Pour éliminer le bloc, il est nécessaire de le sélectionner (l'indiquer avec le pointeur de la souris et appuyer le bouton gauche de la « souris »), et ensuite appuyer le bouton *Delete* sur le clavier.

On peut rechercher un bloc de la liste en tapant une partie de son nom dans la case blanche apparaissant en haut de la librairie, puis en appuyant sur Retour.

Figure II.64 : la case blanche

- Après l'installation sur le schéma de tous les blocs des bibliothèques demandées, il est nécessaire de procéder à la liaison des éléments du schéma. Pour relier les blocs, il est nécessaire d'indiquer avec le pointeur de la souris la sortie du bloc : nous pouvons l'identifier par le sens de la flèche. Le pointeur change de forme et devient une croix, après appuyer et, sans relâcher le bouton gauche de la souris, mener une ligne jusqu'à l'entrée de l'autre bloc (le pointeur change encore vers une croix à double trait). Après cela relâcher le bouton.

Pour créer le point de bifurcation sur la ligne de liaison, il suffit d'emmener le pointeur au point de nœud proposé, en appuyant sur le bouton droit de la souris, tirer la ligne. Le schéma du modèle, sur lequel sont exécutées les liaisons entre les blocs, est représenté sur la figure II.65.

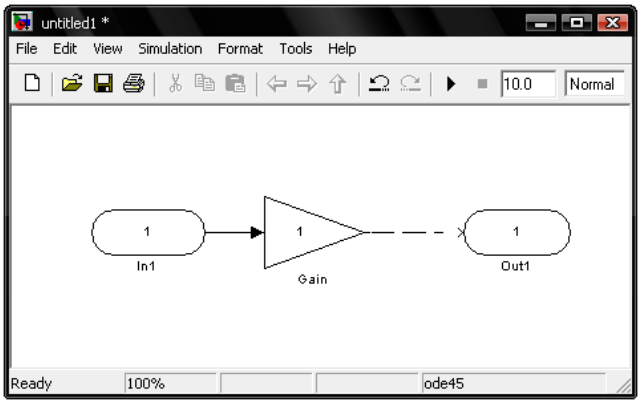


Figure II.65 : les connexions entre les blocs

Pour supprimer la ligne, il faut la sélectionner (comme pour le bloc), et ensuite appuyer sur le bouton Delete sur le clavier. Il est aussi possible de déplacer un segment du trait et même d'ajouter un nouveau trait à partir de celui en cliquant avec le bouton droit.

Ensuite, si nécessaire, il faut changer les paramètres du bloc, installés par le programme. Pour cela il est nécessaire de cliquer deux fois le bouton gauche de la « souris », sur l'image du bloc. La fenêtre de rédaction des paramètres du bloc donné est ainsi ouverte (on a déjà vu) dans laquelle on insère les valeurs appropriées. Après les changements on doit fermer la fenêtre par le bouton OK.

Une fois les différents blocs assemblés, on sélectionne l'ensemble à la souris et on les groupe à l'aide de la commande ***create subsystem*** pour obtenir un seul bloc.

Un double clic sur l'icône sous-système permet l'ouverture de la fenêtre dans laquelle on retrouve les icônes assemblées auxquelles Simulink a ajouté automatiquement les blocs d'entrée Inport et de sortie Outport (figure II.66)

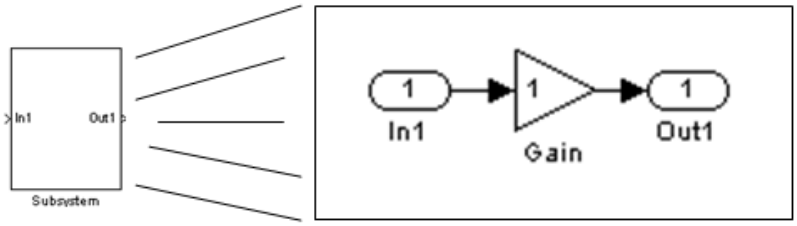


Figure II.66 : le contenu du sous-système

Une fois le diagramme terminé, on peut l'enregistrer dans un fichier: dans le menu ***File***, choisir ***Save As*** et donner un nom (*.mdl) au fichier.

II.5 Ajustement des paramètres de la simulation :

Avant de lancer une simulation, on doit choisir les paramètres appropriés au modèle du système.

Cliquons dans le menu Simulation de la barre de menu dans le haut de la fenêtre où se trouve le schéma, puis sélectionnons ***Normal***. Ensuite, toujours dans Simulation, il faut :

- Cliquer sur Simulation Parameters,
- Aller dans l'onglet Solver,
- Définir les temps de début et de fin de simulation (par défaut, ces temps sont 0s et 10s respectivement),
- Choisir le mode de simulation,

La boîte de dialogue de paramètres de la simulation est montrée à la figure II.67.

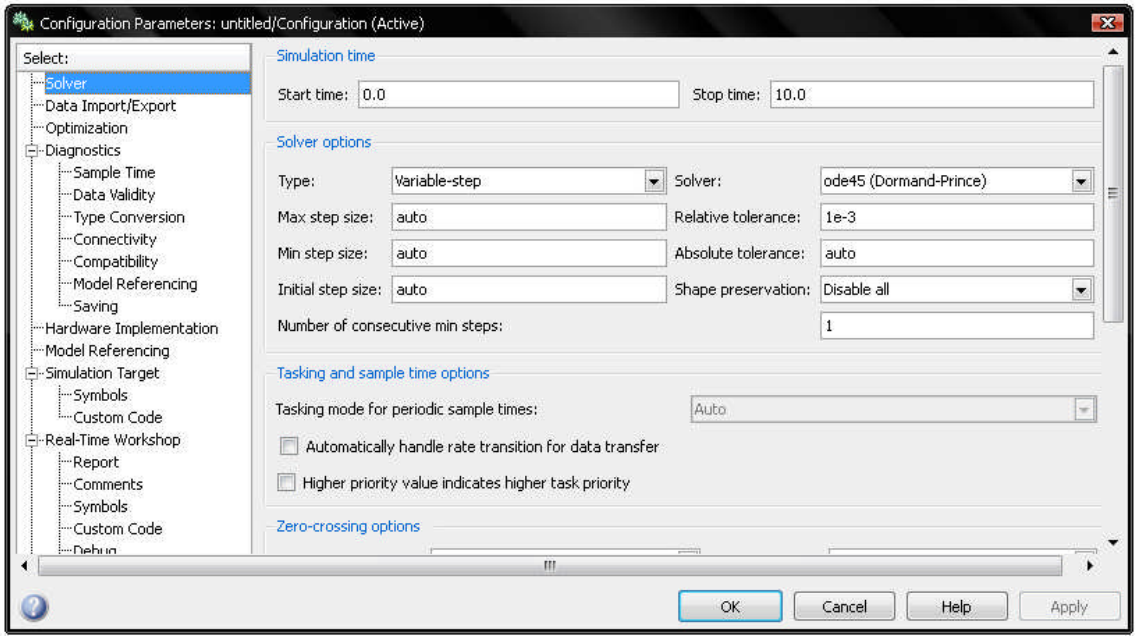


Figure II.67 : la boîte de dialogue de paramètre de la simulation

Enfin, lorsque les paramètres sont entrés, la simulation peut être amorcée de la façon suivante :

- ✓ Cliquer sur l'icône de la barre d'outils (flèche noire),
- ✓ Ou aller dans l'item Simulation du menu et faire Star.

II.6 Conclusion :

Ce deuxième chapitre se veut seulement un « tour d'horizon » qui nous a permis de décrire les outils de la bibliothèque de Simulink et la fonctionnalité de ces blocs qui sont utilisés dans notre projet. Nous avons trouvé que Simulink est un outil de modélisation et de simulation facile avec une grande efficacité, où un bloc peut remplacer l'écriture d'un programme qui peut être compliqué, avec la possibilité de visualisation à n'importe quelle point du modèle.

CHAPITRE III :

Simulation sous MATLAB/SIMULINK d'une chaîne de transmission numérique en bande de base

Références: [1], [2], [3], [4]

III - Simulation sous MATLAB/SIMULINK D'une chaîne de transmission numérique en bande de base

III.1 Introduction :

La simulation de systèmes de communications est un moyen efficace et rapide pour mettre en lumière les performances et les principales difficultés de conception de ces derniers. En se servant de bibliothèques, un outil de simulation comme Matlab/Simulink permet de modéliser des chaînes de transmissions, pour en analyser par exemple les performances en termes de taux d'erreur binaire (TEB). C'est ainsi qu'on vous propose de modéliser et simuler, sous Matlab/Simulink, une chaîne, simplifié, de transmission numérique en bande de base.

L'objectif essentiel de ce chapitre est d'illustrer les notions abordées en théorie de l'information, en particulier le codage de canal. La première étape consiste à modéliser une chaîne simple (source-canal-destinataire), sans aucun accessoire (codage, décodage, etc.). Puis les blocs de codage/décodage. La réalisation de la chaîne de transmission va permettre, d'une part, de visualiser relativement les effets du bruit sur signal utile et d'autre part d'observer l'effet du seuil de décision, des codeurs et décodeurs sur le taux d'erreurs binaire. La sécurisation de l'information n'est pas le seul point à être observé, car il faut étudier aussi l'effet du codage .La manipulation doit être paramétrable, dans le sens où elle doit posséder un certain nombre de paramètres réglables, tels que l'amplitude du signal utile et la puissance du bruit, la durée des symboles, etc. De plus, il faut découper la chaîne de transmission en blocs différents, en se servant du caractère modulaire de Matlab/Simulink, tout en dissociant les problèmes. C'est-à-dire que la chaîne de transmission doit faire apparaître clairement les modules de base que sont entre autres, la source pour générer les données, l'émetteur pour la mise en forme des signaux, le détecteur optimal qui effectue un prétraitement sur le bruit, le canal lui-même, etc.

III.2 Description de la chaîne de transmission (sans codage de canal)

III.2.1 Présentation générale :

La chaîne de transmission permet le transport, en bande de base, d'une information sous forme de données numériques codées en binaire. Elle est composée de :

- Un émetteur (source binaire) ;
- Un canal ;
- Un récepteur.

Les perturbations sont modélisées par un bruit blanc (de puissance réglable), gaussien et centré qui est ajouté au signal utile en sortie du canal. Le synoptique général de la manipulation est présenté à la figure III.1.

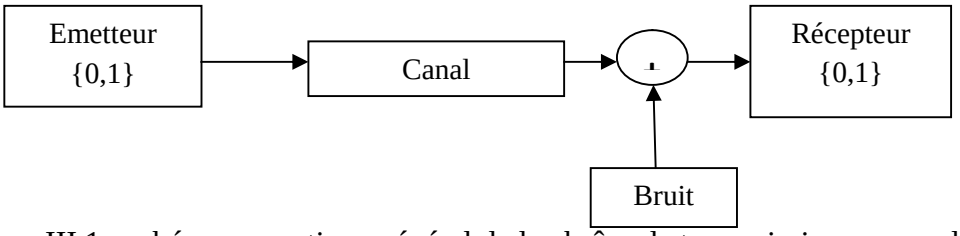


Figure III.1 : schéma synoptique général de la chaîne de transmission sans codage de canal

La chaîne de transmission permet de transmettre une information sous forme de données numériques codées en binaire.

Les différents possibilités sont : 0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1001, 1010, 1011, 1100, 1101, 1110 , 1111 (de 0 jusqu'à 15).

A la réception il faut récupérer les symboles émis sans erreurs.

III.2.1.1 L'émetteur : Source binaire

L'émetteur est une source binaire simple (sans mémoire et stationnaire), qui émet des symboles indépendants. La figure III.2 représente le schéma bloc de la partie émettrice de la chaîne de transmission, il est composé des éléments suivants :

- Un paquet de 6 données codées sur 4 bits génère un train de nombres entiers de 1 à 6;
- Un convertisseur Entier-Bits convertit chaque nombre entier ou valeur à point fixe présenté à l'entrée à un groupe de bits présenté à la sortie.
- Une Fonctions-S que l'on doit programmer (serie0) ;
- Sept blocs constants;
- Un émetteur bipolaire pour l'émission ;
- Un oscilloscope (Scope2) pour l'affichage.

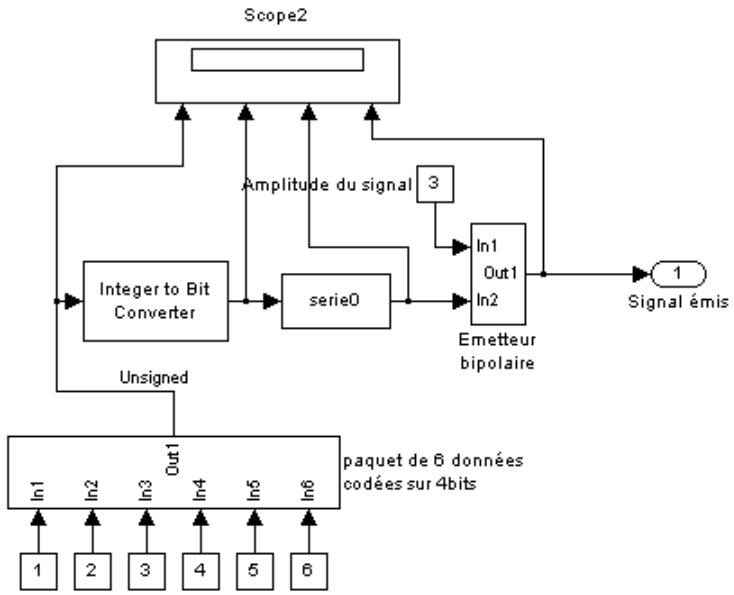


Figure III.2 : schéma bloc de la partie émettrice de la chaîne de transmission numérique

La transmission s'effectue en bande de base c'est-à-dire sans porteuse.

III.2.1.2 Le récepteur :

La figure III.3 représente le schéma bloc du récepteur de la chaîne de transmission, il est composé des éléments suivants :

- Un générateur d'impulsion (synchro) permet de produire des impulsions carrées à des intervalles réguliers ;
- Un détecteur optimal ;
- Un multiplexeur à deux entrées ;
- Trois oscilloscopes pour l'affichage ;
- Trois Fonctions-S que l'on doit les programmer (receptionparallele0, blocagedonnees0, donneesparalleles);
- Un convertisseur Bits-Entier convertit un groupe de bits présents à l'entrée en un nombre entier en sortie.
- Un bloc Display pour afficher les valeurs de sortie.

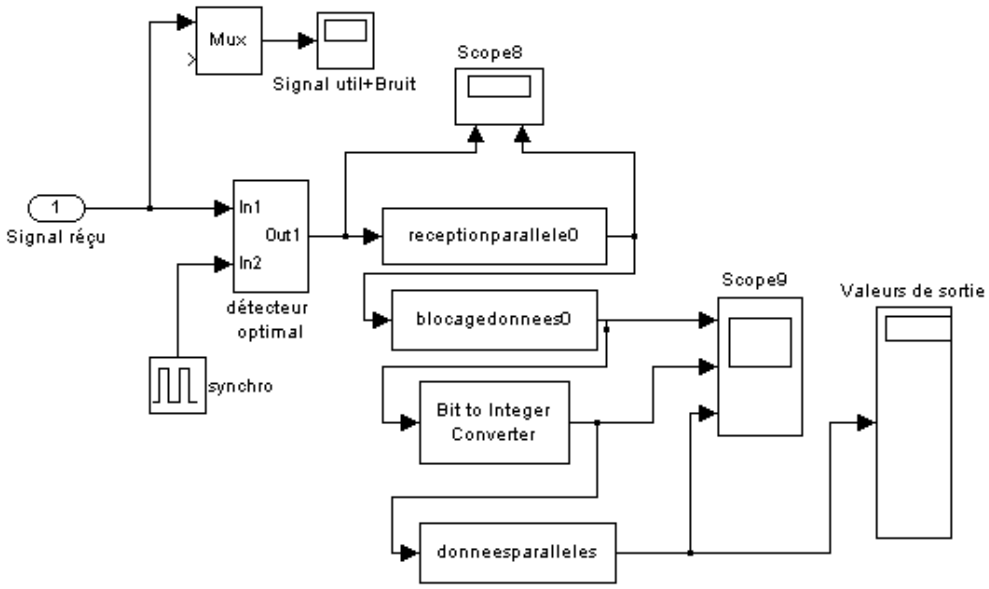


Figure III.3 : schéma bloc du récepteur de la chaîne de transmission numérique

III.2.2 Mise en œuvre sous Matlab/Simulink :

Dans cette partie on va assembler les différents blocs qui forment la chaîne de transmission de façon similaire à la réalisation pratique.

On va simuler chaque partie de la chaîne (émetteur, canal, récepteur) à part, et ensuite on fait l'assemblage des trois parties.

III.2.2.1 L'émetteur :

III.2.2.1.1 Source d'information :

La figure III.4 représente le paquet de 6 données codées sur 4 bits. Il est composé des éléments suivants :

- Une horloge digitale produit le temps de simulation seulement à l'intervalle de prélèvement indiqué (un signal en escalier) (figure III.6 (a)).
- Deux blocs constants ;
- Un multiplieur des signaux à deux entrées ;
- Un sommateur des signaux à deux entrées ;
- La masse (Ground) ;
- Un oscilloscope (Scope1) pour l'affichage ;
- Un commutateur multiport permet de choisir entre un certain nombre d'entrées ; La première (en haut) d'entrée est l'entrée de commande et les autres entrées sont des entrées de données. La valeur de l'entrée de commande détermine quelle source de données à passer par le port de sortie.
- 6 blocs In1 pour l'entrer des données ;
- Un bloc Out1 pour la sortie.

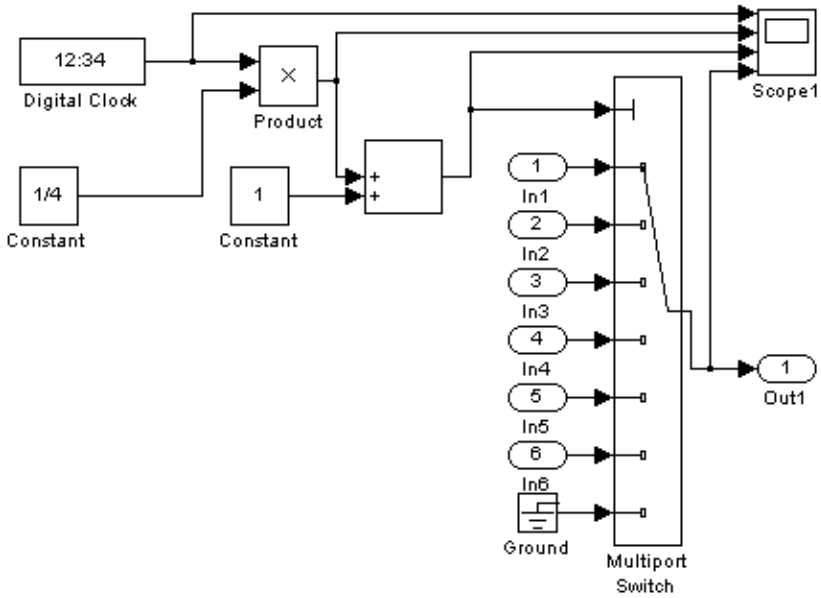


Figure III.4 : la source d'information

Dans notre cas l'horloge digitale produit un temps de simulation (temps d'échantillonnage) toutes les **4** secondes. Les paramètres et la boîte de dialogue de cette horloge digitale sont donnés à la figure (III.5) :

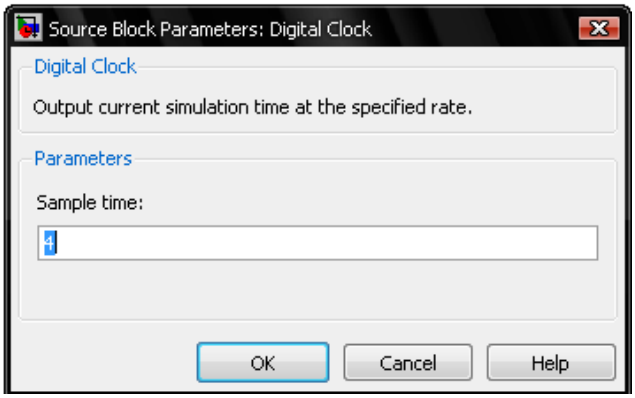


Figure III.5 : boîte de dialogue « horloge digital »

Les données sont codées sur 4 bits, le signal d'horloge est multiplié par **¼** (figure III.6 (b)). Afin d'avoir des signaux de moyenne nulle, nous avons ajouté « 1 » au signal d'horloge multiplié par « ¼ » (figure III.6 (c)). En sortie de la source d'information, le signal obtenu a des valeurs respectives : 1, 2, 3, 4, 5, 6 (figure III.6 (d)).

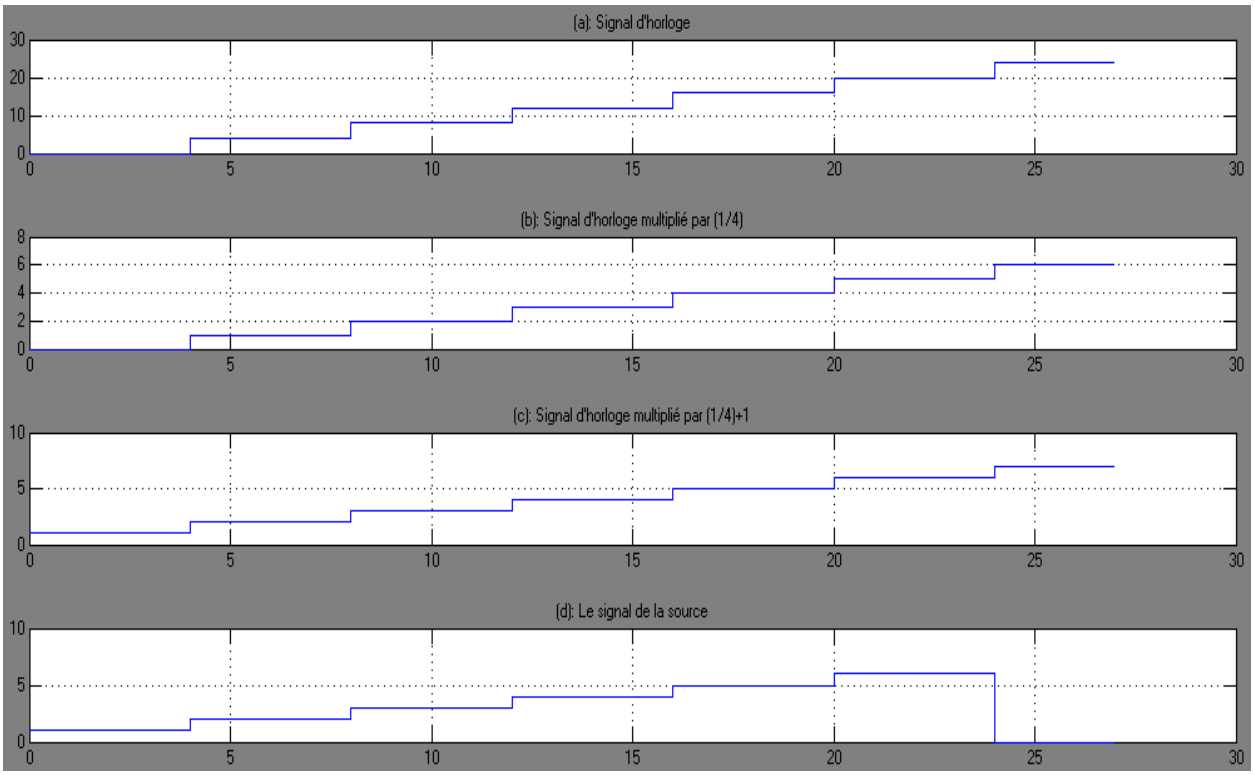


Figure III.6 : les signaux au niveau du Scope1

A la sortie de la source d'information, on utilise un convertisseur entier-binaire qui permet de convertir les données numériques qui représentent l'information sur **4** bits. Elles sont présentées à l'entrée de la chaîne de transmission les unes après les autres. Les paramètres de ce convertisseur sont montrés à la figure III.7.

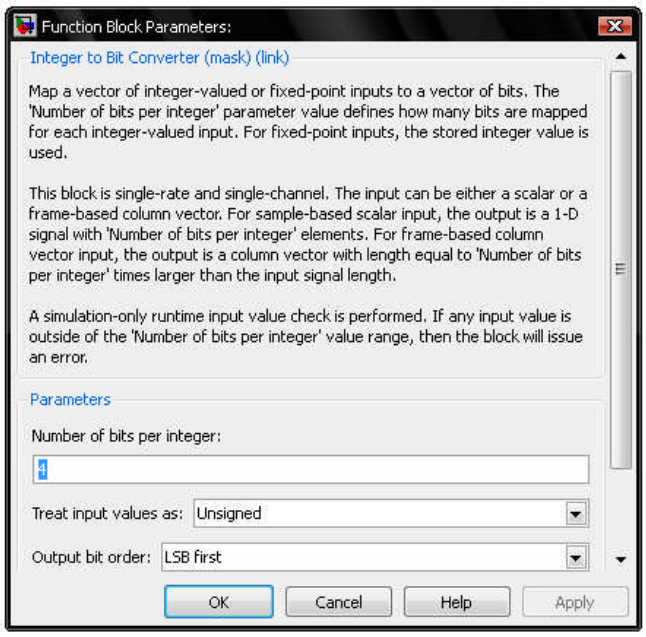


Figure III.7 : boîte de dialogue « convertisseur entier-bits »

Exemples :

Les valeurs en décimal	Les valeurs en binaire (sur 4 bits)
1	0001
4	0100
5	0101

Le signal à la sortie de ce convertisseur « entier-bits » est illustré à la figure III.8, chaque valeur contient 4 bits.

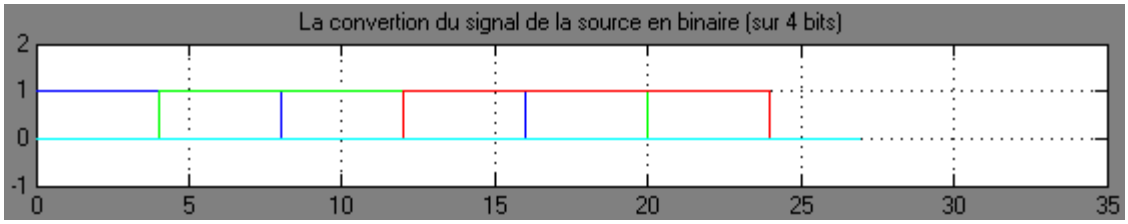


Figure III.8 : Le signal à la sortie du convertisseur « entier-bits »

III.2.2.1.2 La Fonction-S « serie0 » :

En cours d'une simulation, SIMULINK appelle à chaque itération les blocs Fonctions-S, demande un calcul des sorties, une mise à jour des états discrets, encadrés au lancement de la simulation par une étape d'initialisation et à la fin par une étape d'arrêt.

Dans le cas d'une Fonction-S écrite en langage Matlab (M-file), les appels sont gérés par le contenu de l'indicateur flag dans la fonction appelante. Le paramètre flag permet de savoir à tout moment dans quelle étape de la simulation se trouve le système.

Les Fonctions-S que nous avons utilisés dans notre simulation sont écrites à l'aide de fichier M.

Ce fichier enregistré comme un M-file, contient le protocole dans lequel Simulink peut accéder aux informations à partir de MATLAB.

Les significations de l'indicateur flag sont données ci –dessous :

Etape de la simulation	M-File Flag
Initialisation	Flag = 0
Calcul du prochain pas d'échantillonnage (optionnel)	Flag = 4
Calcul des sorties	Flag = 3
Mise à jour de l'état discret	Flag = 2
Mise à jour de l'état continu	Flag = 1
Fin de simulation	Flag = 9

La syntaxe de la Fonction-S spécifiée par un fichier M est la suivante :
Function [sys, x0, str,ts] = nom_fonction (t,x,u,flag,tech,lgmotcode)
t, **x**, **u**, **flag** : variables passées par défaut à la Fonction-S par le système
tech, **lgmotcode** : variables que l'on désire passer à la Fonction-S.
Les variables retournées au système en fonction du temps dépendent de l'état des vecteurs x, u, et de l'indicateur flag.
sys est le principal vecteur de résultats demandé par Simulink. Selon le drapeau envoyé par Simulink, ce vecteur tiendra des informations différentes.
Flag = 0, la fonction d'initialisation doit retourner dans sys les informations de taille système, dans x0 l'état initial du système, dans **str** une chaîne vide (pour les fonctions M) et dans **ts** les instants d'échantillonnage. Pour cela, on peut utiliser la fonction **mdlInitializesizes**, afin d'initialiser une structure de type **simsizes** contenant les champs suivants que l'on affectera à la variable sys :

```
NumContStates      % nombre d'états continus
NumDiscStates      % nombre d'états discrets
NumOutputs         % nombre de sorties
NumInputs          % nombre d'entrées
DirFeedthrough     % matrice D vide
NumSampleTimes     % nombre de lignes matrices ts
```

Les cas flag =2 et flag= 1 sont utilisés, respectivement dans les cas de processus discrets et continus.

Une Fonction-S serie0.m est générée pour simuler le processus discret.

Chacune des données numériques doit rester présente pendant **4** unités de temps **1sec**, une interface série, la Fonction-S « serie0 », permettant ensuite d'émettre ces **4** bits de façon série à travers le canal. L'unité de temps **1 sec** choisie est bien sûr arbitraire et elle cadencera de façon synchrone tous les blocs de la chaîne de transmission. **1/1 sec** est donc le rythme digital.
Le listage des programmes Matlab correspondants est proposé en annexe.

La Fonction-S est programmée suivant l'organigramme de la figure III.9.

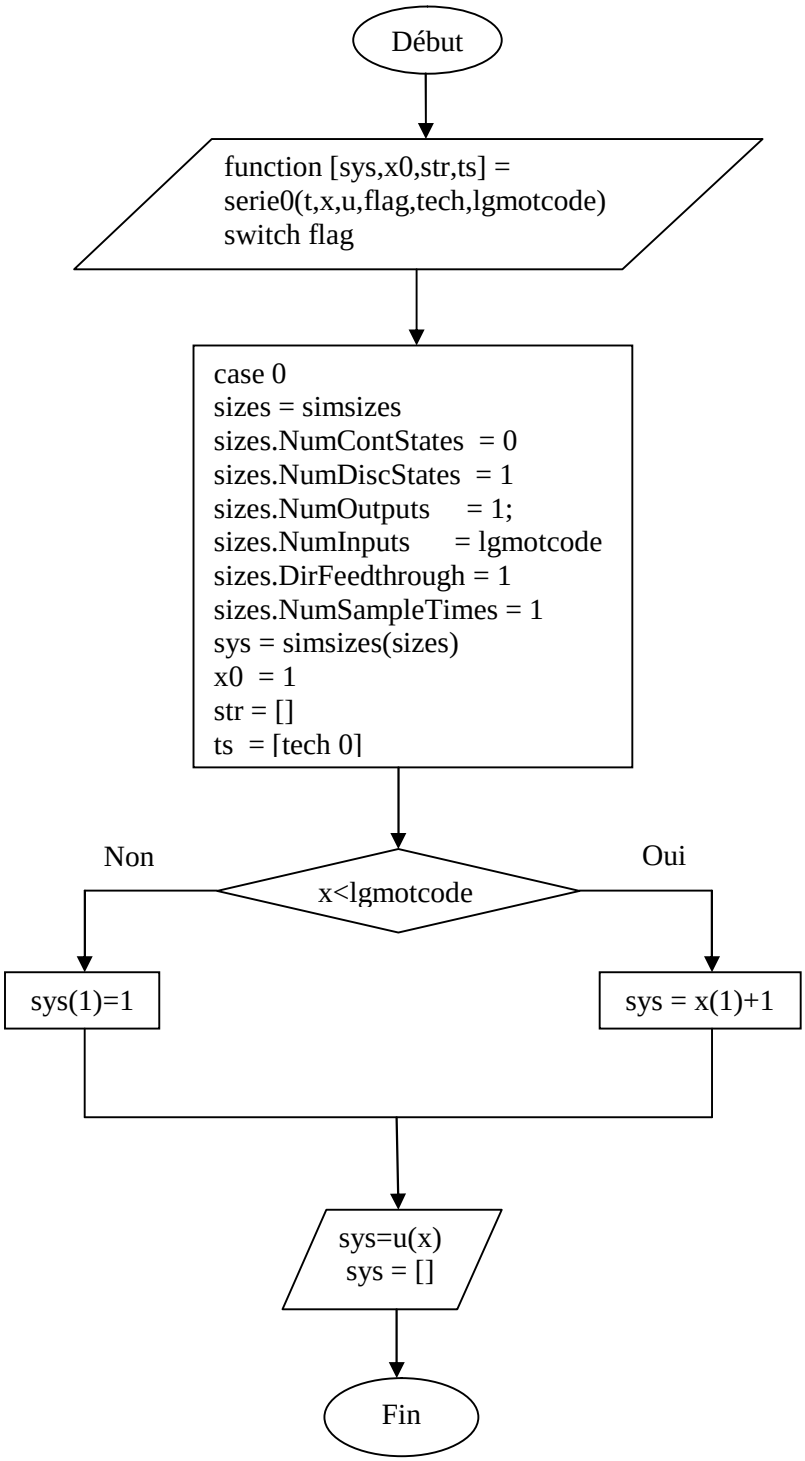


Figure III.9 : l'organigramme de la Fonction-S « serie0 »

Les paramètres de la Fonction-S « serie0 » sont montrés à la figure III.10.

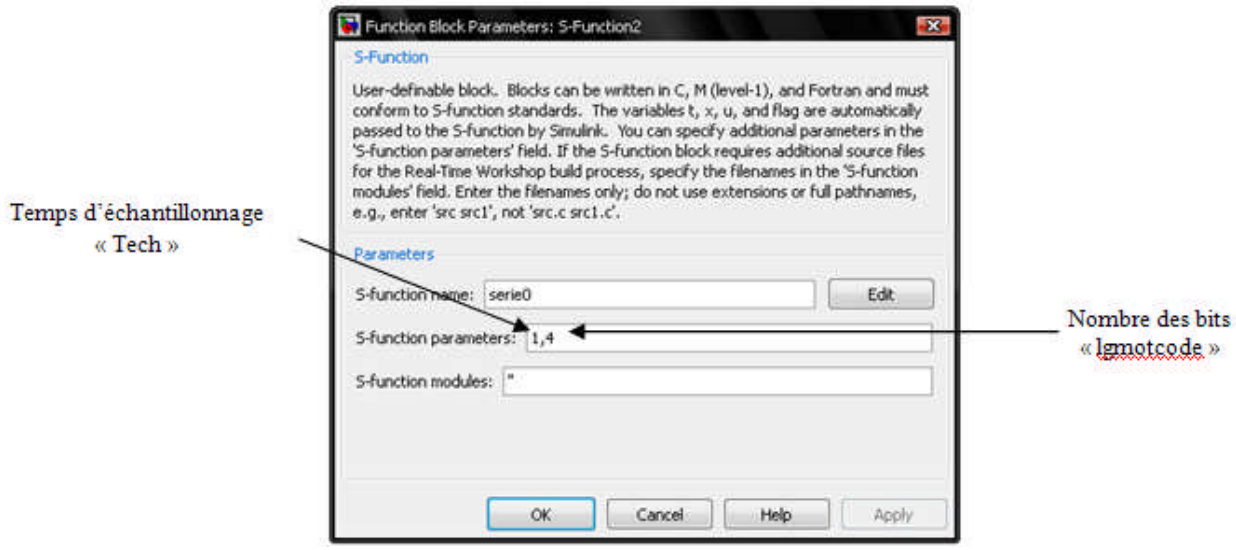


Figure III.10 : boîte de dialogue de la Fonction-S « serie0 »

Le signal à la sortie de cette Fonction-S est donné à la figure III.11.

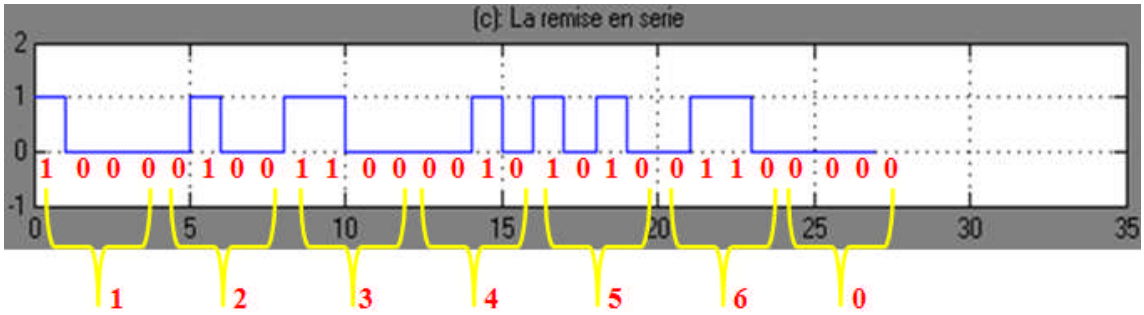


Figure III.11 : le signal à la sortie de la Fonction-S « serie0 »

III.2.2.1.3 Emetteur bipolaire :

L'émetteur bipolaire (émetteur physique) permet d'assurer la mise en forme des signaux en associant aux symboles binaires des tensions bipolaires d'amplitude réglable. Il s'agit donc d'un signal NRZ. Il est constitué des éléments suivants :

- Deux blocs constants ;
- Deux multiplieurs des signaux l'un à deux entrées et l'autre à trois ;
- Un sommateur à deux entrées ;
- Un oscilloscope pour l'affichage.

Le schéma de cet émetteur bipolaire est montré à la figure III.12.

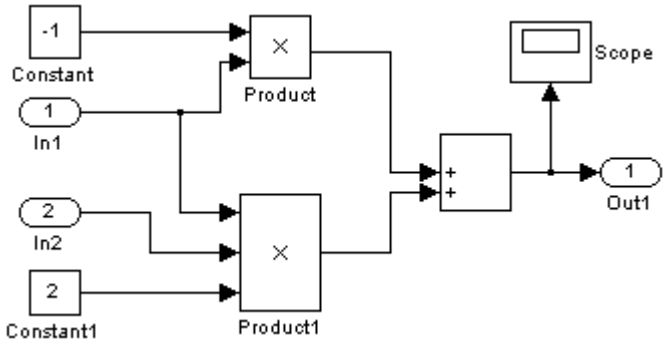


Figure III.12 : l'émetteur bipolaire

Les deux signaux entrants (amplitude du signal et le signal de la source) sont amplifiés par multiplication par des constants biens définis. Après ils sont additionnés afin d'être transmis (transmission cohérente). Le signal émis est montré à la figure III.13.

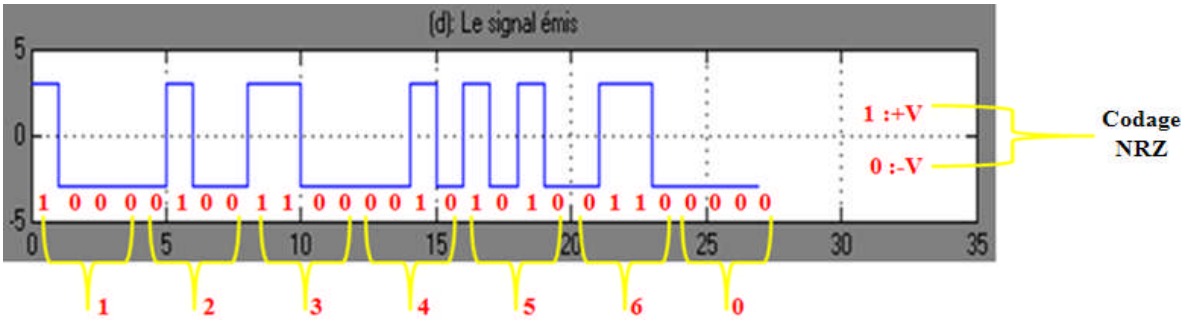


Figure III.13 : le signal émis

III.2.2.2 Le canal de transmission :

Un canal représente la liaison entre l'émetteur et le récepteur et peut être de différentes natures (espace libre ou un câble) selon le type de grandeur qu'il permet de véhiculer. Dans un souci de simplification, le canal est supposé idéal dans le sens où il n'est constitué que d'un fil de transmission. L'effet de canal est limité, donc, aux perturbations modélisées par un bruit additif. Cependant, il est tout à fait possible de choisir un modèle plus réel qui prendrait en compte une atténuation, une réponse en fréquence non uniforme et aussi les phénomènes d'échos et de trajets multiples. Il est montré à la figure III.14.

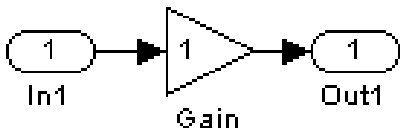


Figure III.14 : le canal de transmission

Dans ce projet nous avons utilisé un canal classique (idéal) avec un **bruit blanc**. Nous avons vu les caractéristiques du bloc bruit blanc dans le chapitre II. Les paramètres du bloc bruit blanc sont indiqués à la figure III.15.

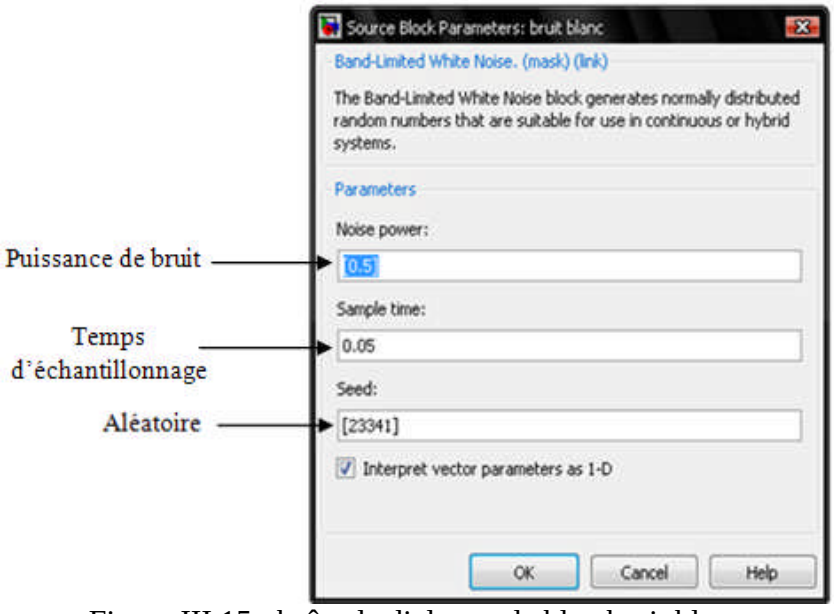


Figure III.15 : boîte de dialogue de bloc bruit blanc

Le bruit blanc est une perturbation uniforme du signal, c'est-à-dire qu'il rajoute au signal une petite amplitude dont la moyenne sur le signal est nulle. Le bruit blanc est généralement caractérisé par un ratio appelé ***rapport signal/bruit***, qui traduit le pourcentage d'amplitude du signal par rapport au bruit (son unité est le décibel). Celui-ci doit être le plus élevé possible.

L'augmentation de la puissance du signal émis améliore la qualité du signal reçu.
Le signal utile + bruit est montré à la figure III.16.

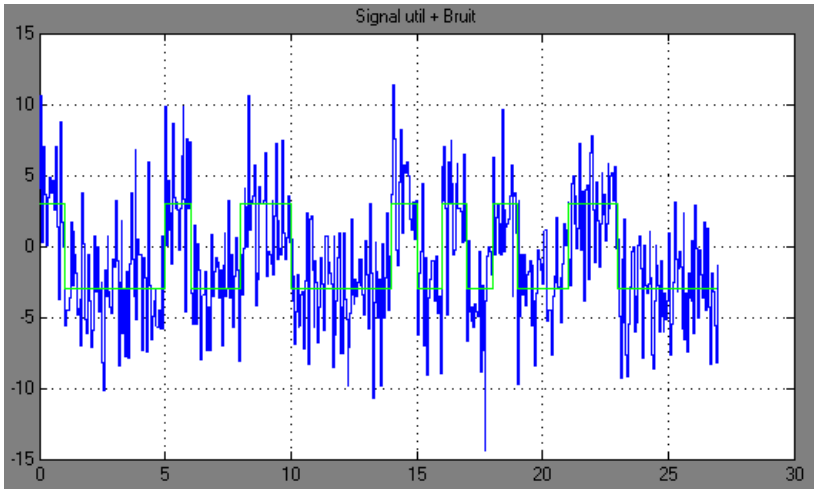


Figure III.16 : le signal utile + le bruit

III.2.2.3 Le récepteur :

Le récepteur qui a pour fonction de reconstituer le message émis par la source à partir du signal reçu, comprend une partie de détection optimale et une partie de remise en forme des données numériques reçues.

III.2.2.3.1 **Détecteur optimal :**

Le détecteur optimal, comme le montre la figure III.17, peut être réalisé à l'aide de :

- Un bloc constant (0) pour définir la valeur du seuil ;
- Un intégrateur ;
- Un échantillonneur, synchronisé par le rythme de la source ;
- Un comparateur à seuil dont la valeur du seuil dépend bien sûr de l'entropie de la source (elle est supposée maximum ; on considère donc que l'opération de codage de source a été effectuée) et de l'émetteur bipolaire. Dans notre cas, la valeur du seuil est nulle

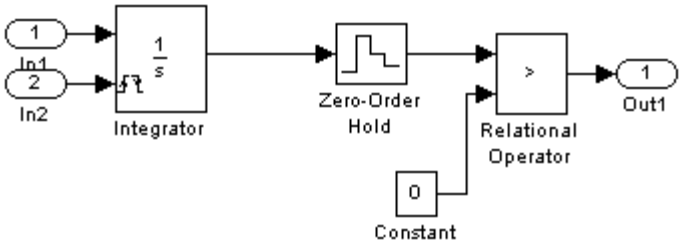


Figure III.17 : le détecteur optimal

Le détecteur optimal permet de supprimer le bruit. En premier temps, Le signal reçu sera intégré (figure III.18 (c)) puis échantillonné à des instants caractéristiques (figure III.18 (d)). Finalement un comparateur à seuil identifie la valeur des éléments binaires transmis à partir des échantillons reçus (figure III.18 (e)).

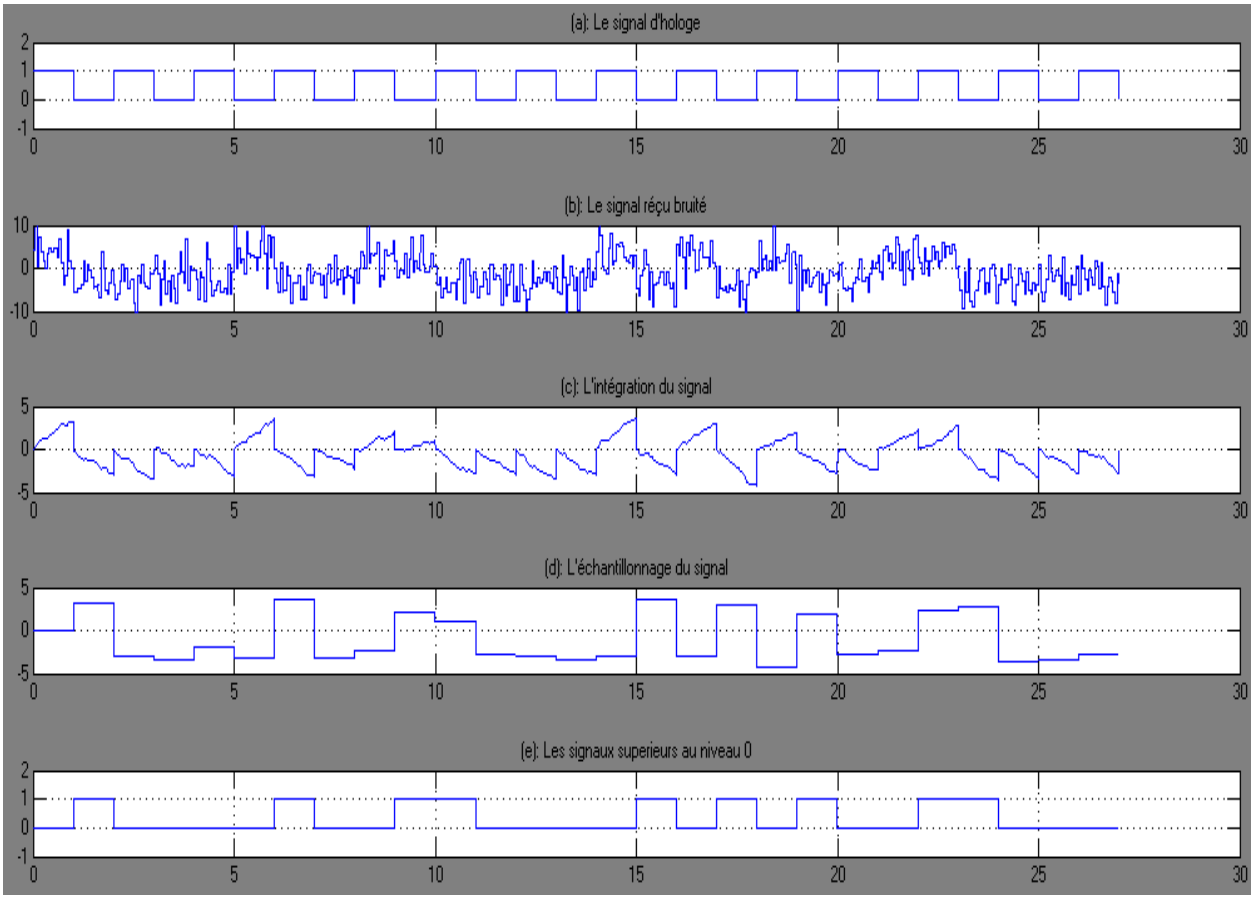


Figure III.18 : les opérations effectuées au niveau du détecteur optimal

La partie remise en forme des données numériques permet de la remise en forme les symboles binaires de façon à reconstituer l'information reçue. Elle est constituée de trois Fonction-S qui sont receptionparallele0, blocagedonnees0, donneesparalleles.

- Les paramètres de la Fonction-S « receptionparallele0 » sont montés à la figure III.19.

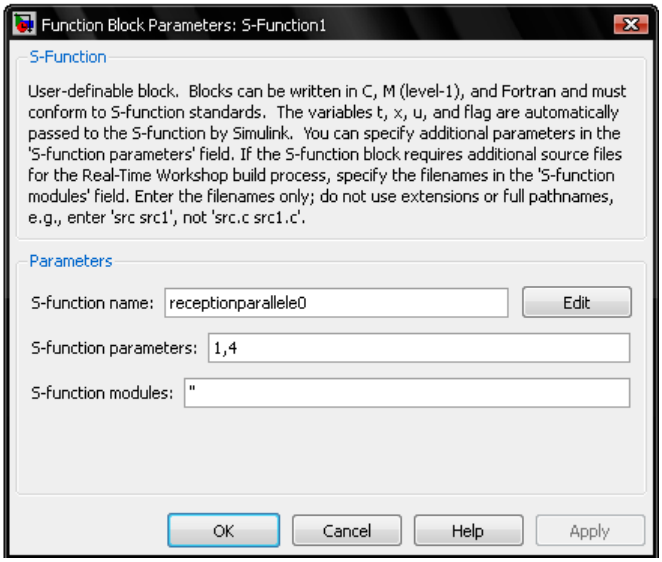


Figure III.19 : boîte de dialogue de la Fonction-S « receptionparallele0 »

A la sortie de la Fonction-S « receptionparallele0 », on obtient le signal de la figure III.20

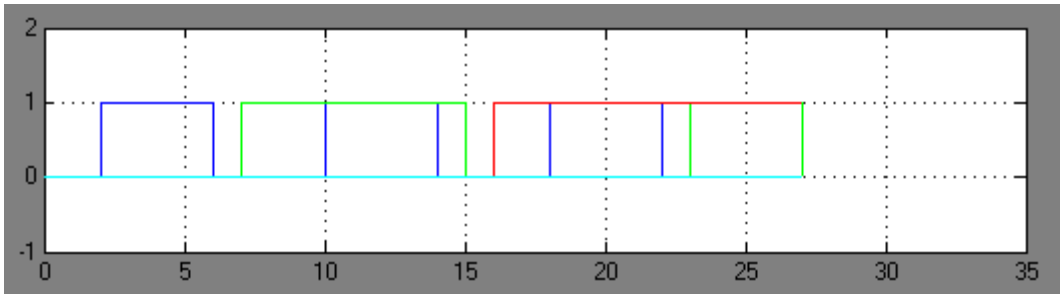


Figure III.20 : le signal à la sortie de la Fonction-S « receptionparallele0 »

La Fonction-S « receptionparalle0 » est programmée selon l'organigramme suivant (figure III.21) :

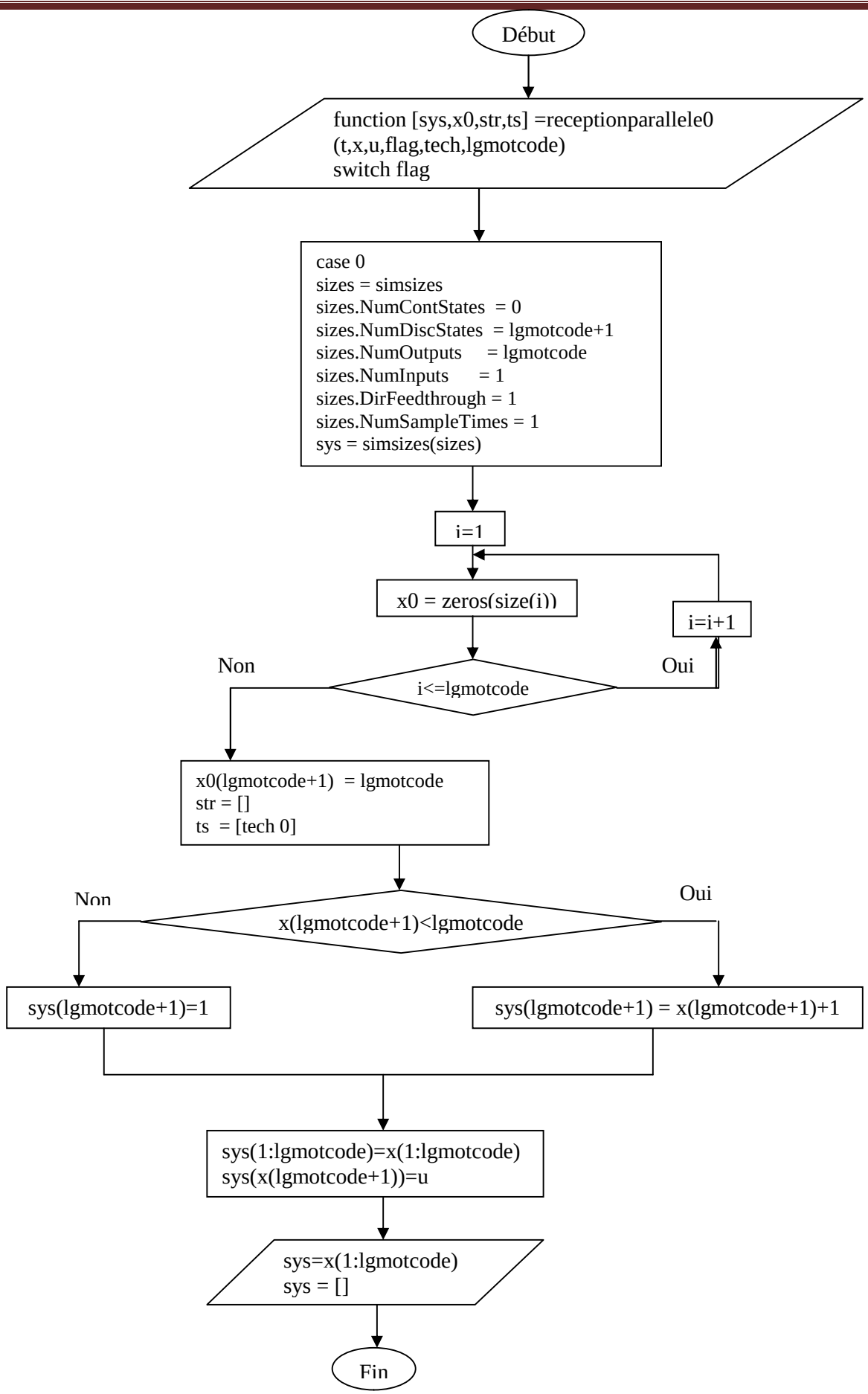


Figure III.21 : l'organigramme de la Fonction-S « receptionparallele0 »

- Les paramètres de la Fonction-S « blocagedonnees0 » sont montés à la figure III.22.

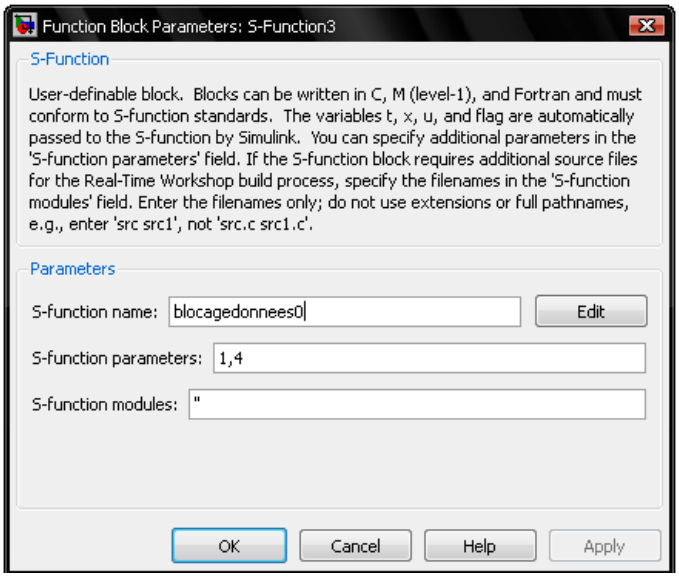


Figure III.22 : boîte de dialogue de la Fonction-S « blocagedonnees0 »

La Fonction-S « blocagedonnees0 » est programmée de la façon suivante :

La figure III.23 montre le signal à la sortie de la Fonction-S « blocagedonnees0 ».

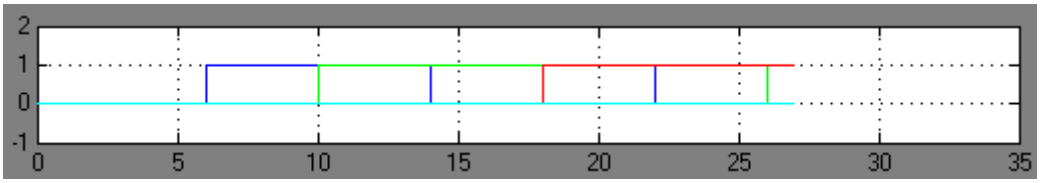


Figure III.23 : le spectre du signal à la sortie de la Fonction-S « blocagedonnees0 »

La Fonction-S « blocagedonnees0 » est programmée selon l'organigramme suivant (figure III.24) :

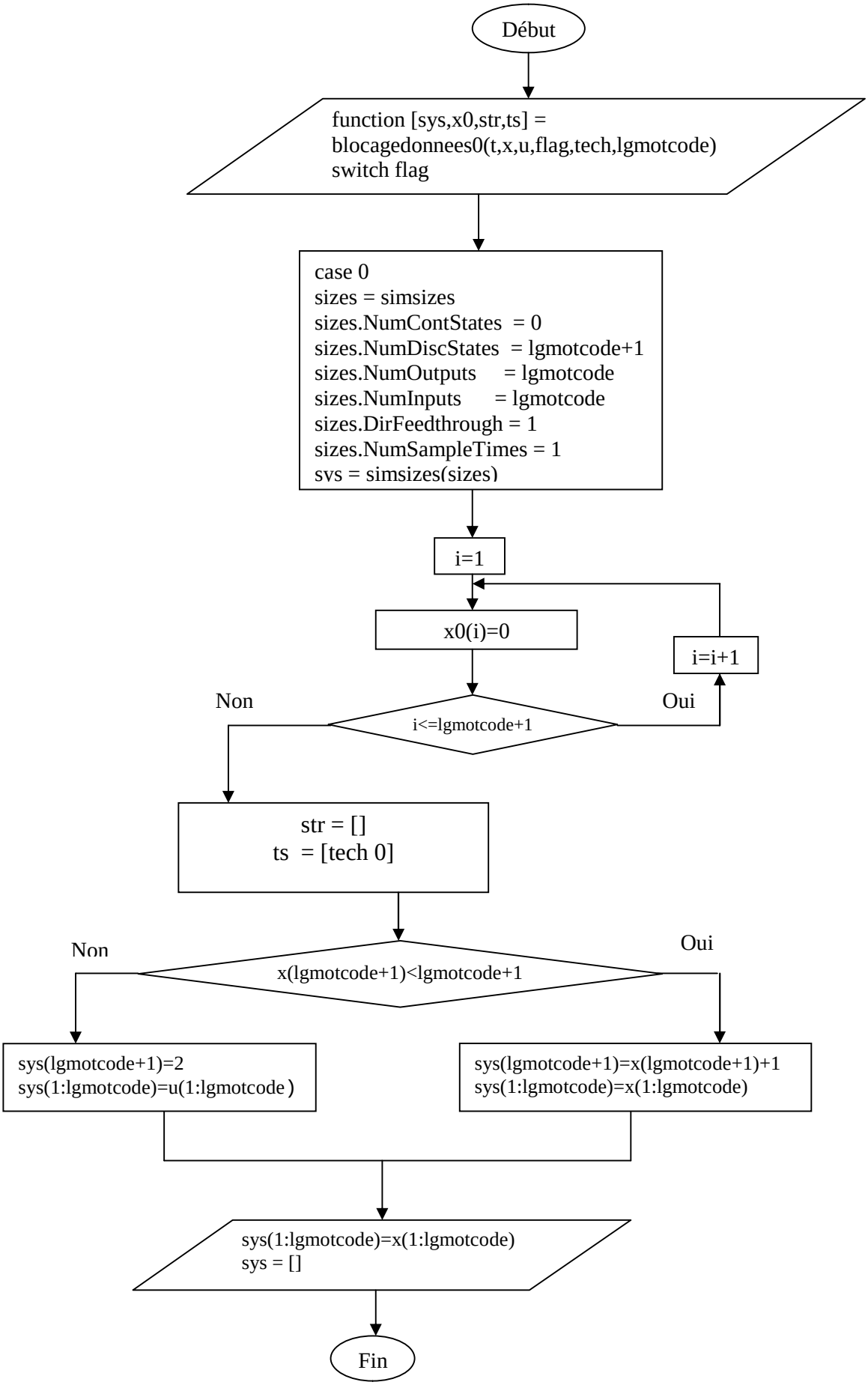


Figure III.24 : l'organigramme de la Fonction-S « blocagedoonnees0 »

A la sortie de la Fonction-S « blocagedonnees0 », on utilise un convertisseur binaire-entier qui permet de convertir un groupe de bits présents à l'entrée en un nombre entier en sortie. Les paramètres de ce convertisseur sont montrés à la figure III.25.

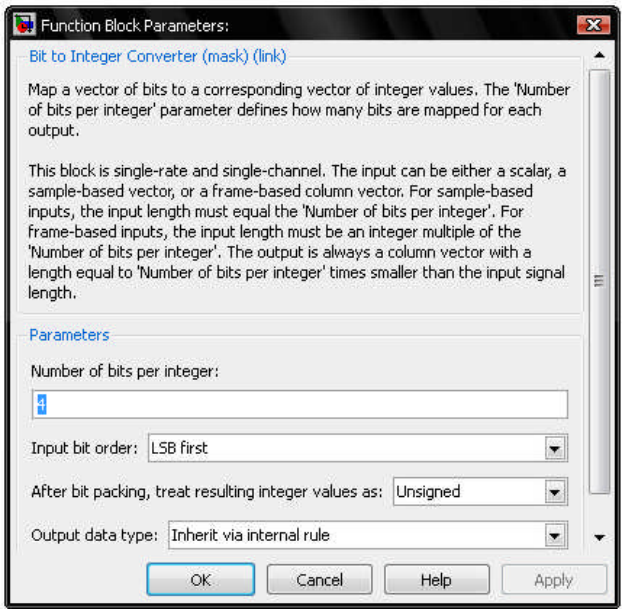


Figure III.25 : boîte de dialogue « convertisseur bits-entier »

Le signal à la sortie de convertisseur bits-entier est donné à la figure III.26.

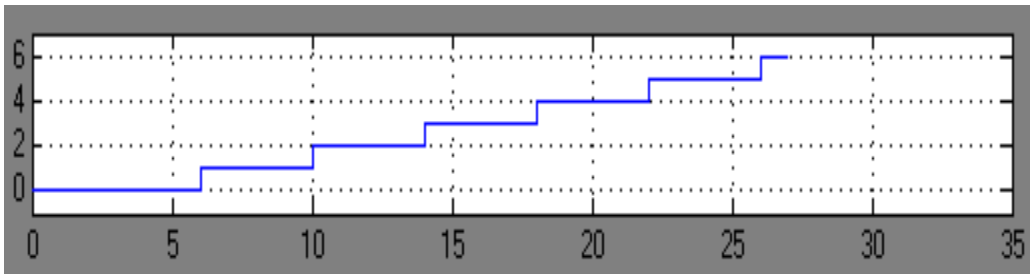


Figure III.26 : le signal à la sortie de convertisseur bits-entier

- Les paramètres de la Fonction-S « donneesparalleles » sont montés à la figure III.27.

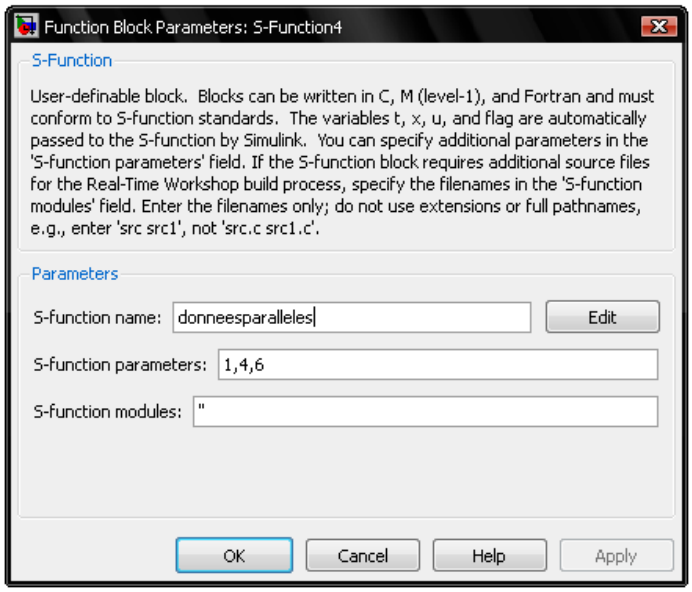


Figure III.27 : boite de dialogue de la Fonction-S « donneesparalleles»

Le nombre **6** indique le nombre des données.

La figure III.28 montre le signal à la sortie de la Fonction-S « donneesparalleles ».

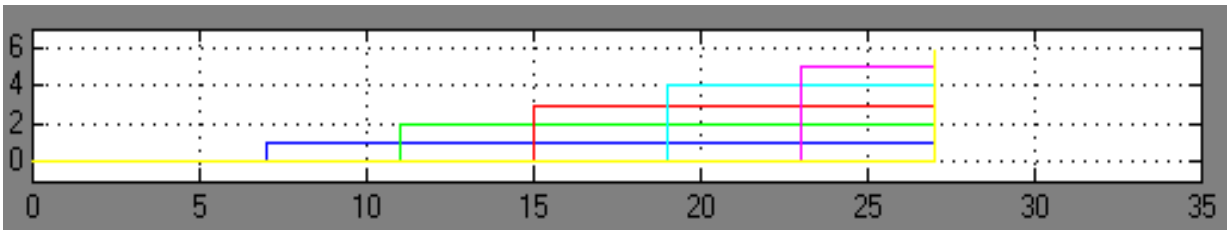


Figure III.28 : le signal à la sortie de la Fonction-S « donneesparalleles »

La Fonction-S « donneesparalleles » est programmée selon l'organigramme suivant (figure III.29) :

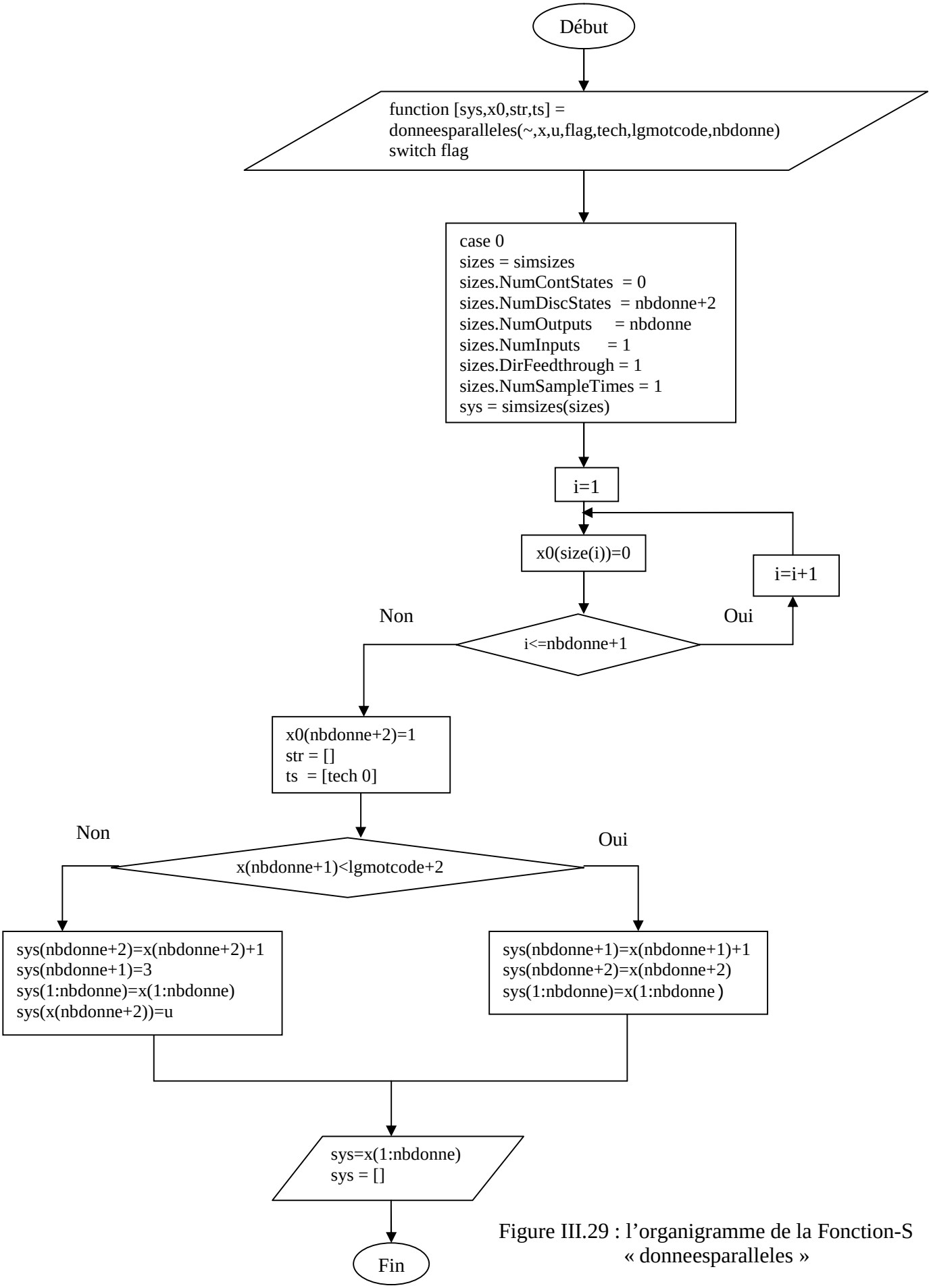


Figure III.29 : l'organigramme de la Fonction-S « donneesparalleles »

III.2.3 Modèle complet de la chaîne de transmission : (sans codage de canal)

Nous avons complété notre modèle de la chaîne de transmission numérique et le schéma global est montré à la figure III.30. Il faut noter que tous les éléments de la chaîne, les Fonctions-S ainsi que les blocs, sont paramétrés selon un certain nombre de variables qui sont ici l'unité de temps **Tech=1sec** et le nombre de bits **lgmotcode=4 bits** pour la représentation binaire des données d'information.

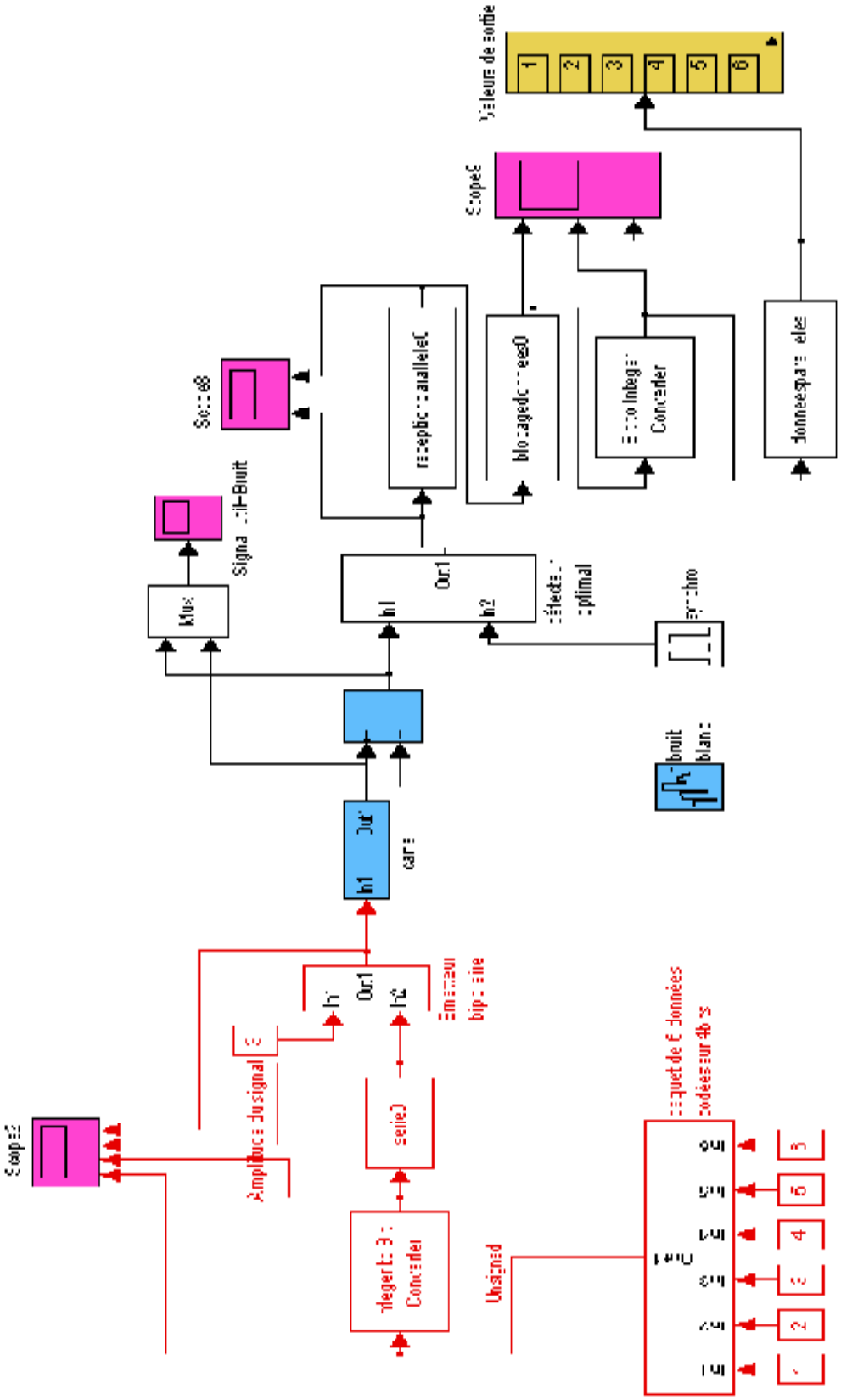


Figure III.30 : la chaîne de transmission sans codage du canal

III.3 Insertion des techniques de codage de canal :

III.3.1 Présentation générale :

On conserve la chaîne réalisée précédemment, avec un bruit blanc gaussien et additif. L'objectif, ici, est d'insérer des blocs de codage et de décodage de canal pour détecter et corriger les erreurs. La figure III.31 illustre cette insertion. Tout le problème se résume à prendre en compte le fait que le codage de canal introduit des bits supplémentaires au niveau de la représentation des données d'information.

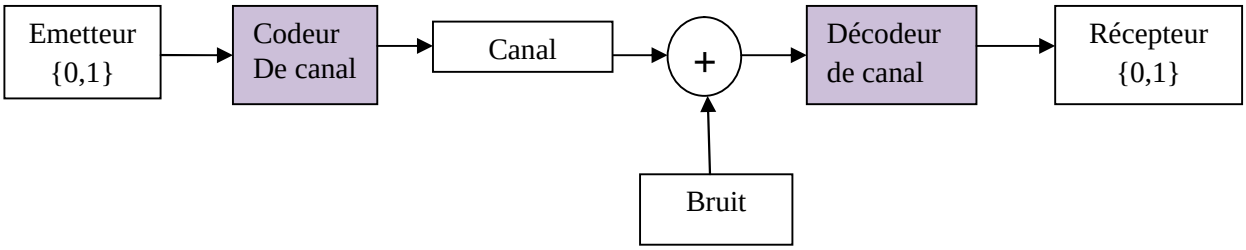


Figure III.31 : schéma synoptique général de la chaîne de transmission avec codage de canal

On va faire quelques modifications au niveau de la source d'information :

- ✓ Changement de temps d'échantillonnage de l'horloge digitale (de 4 secondes à 7 secondes) ;
- ✓ La multiplication du signal d'horloge par **1/7.**

La figure III.32 montre le schéma bloc de la source d'information (paquet de 6 données codées sur 7 bits) :

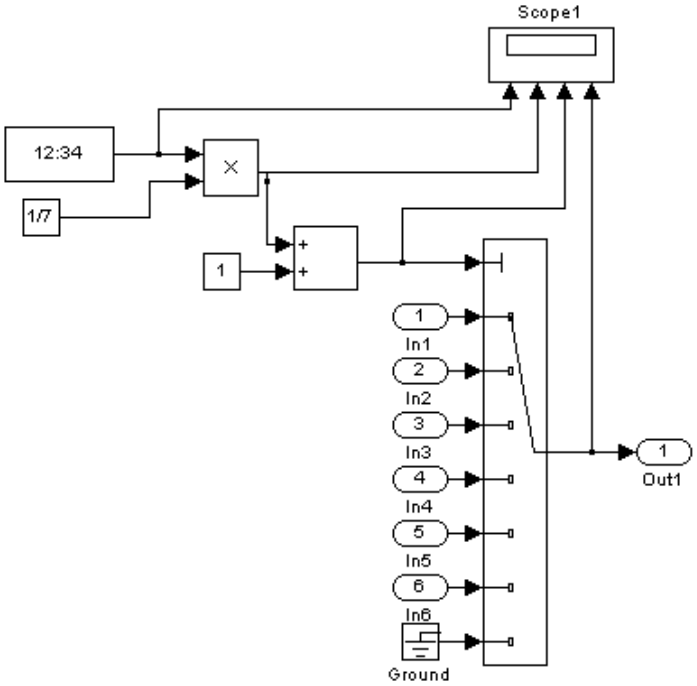


Figure III.32 : la source d'information

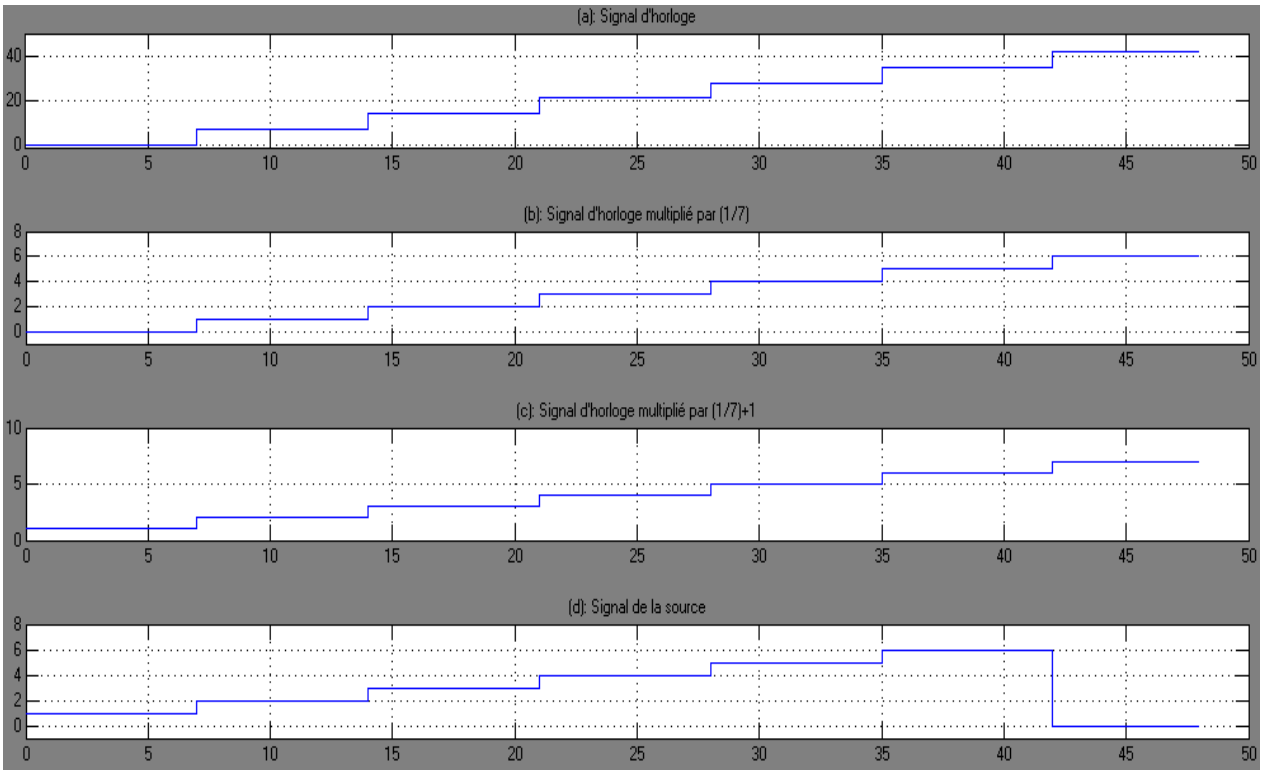


Figure III.33 : les signaux du Scope1

III.3.1.1 Code correcteur d'erreur (codeur de Hamming) :

Cette fois on peut donc continuer à progresser en mettant en œuvre des codes correcteurs d'erreur, l'exemple le plus connu étant sans doute le code de Hamming. Il s'agit d'un code en bloc linéaire, basé sur le respect d'une distance minimale de **3** entre tous les mots du code et qui permet ainsi la localisation de toutes les erreurs simples. L'avantage d'un tel code est qu'il est facilement implémentable à l'aide de portes logiques et qu'il peut être décliné sous différentes formes suivant la matrice génératrice retenue (systématique ou non).

Lors de la transmission de données, des erreurs de transmission peuvent modifier un ou plusieurs bits dans un mot binaire. Afin de détecter, et si possible, de corriger ces erreurs, on ajoute au mot transmis un ou plusieurs bits supplémentaires.

Pour le code de Hamming, si l'on veut corriger une seule erreur sur **m** bits, il suffit du plus petit **r** bits de contrôle qui vérifie $\geq (\quad + \quad + \quad)$
Si l'on veut corriger une erreur pour 4 bits transmis, il faut 3 bits de contrôle
r = 3 car $2 \geq (4 + 3 + 1)$ et l'on transmit les 7 bits.

Le tableau suivant indique les nombres de bits de contrôle, de données pour différentes valeurs de **r**.

r=3	m=4	n=7
r=4	m=11	n=15
r=5	m=26	n=31

Dans la suite de l'étude, on retient **r=3**.

Par exemple, pour un code de Hamming **H(3,7)** non systématique dont une matrice génératrice est donnée ci-dessous, le schéma d'un codeur possible est présenté sur la figure III.35.

$$G = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

Figure III.34 : matrice génératrice G du code de Hamming **H(3,7)**

Le schéma (figure III.35) du codeur comprend simplement **3** portes logiques *ou-exclusif*. À l'entrée de ce codeur, sont connectés les **4** bits d'information et à la sortie de celui-ci nous obtenons le mot-code constitué des **4** mêmes bits d'information et de **3** bits de contrôle. La chaîne de transmission sera donc synchronisée cette fois à partir d'une longueur de mot code égale à **7**.

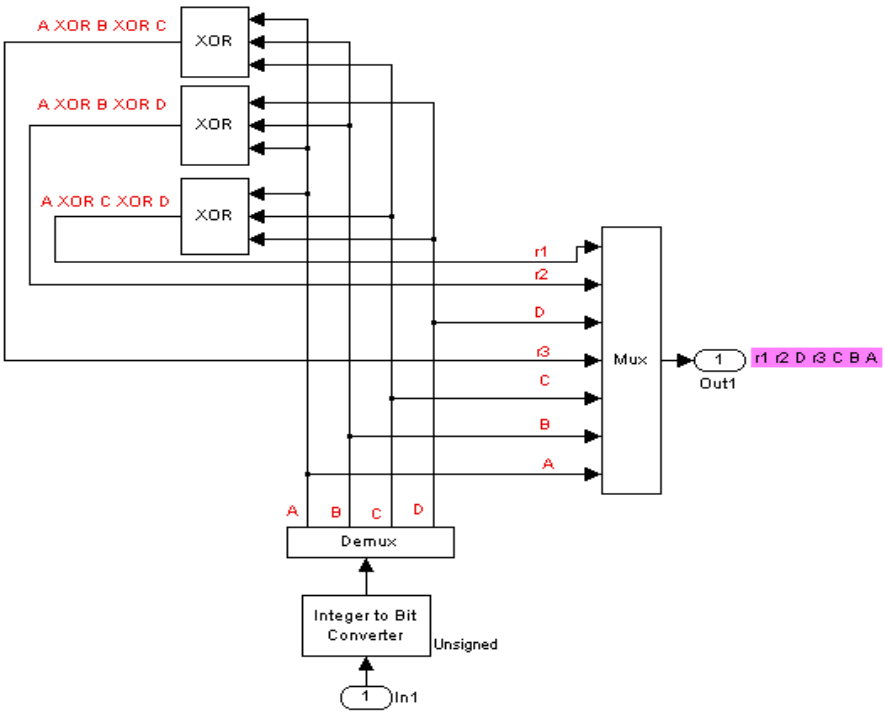


Figure III.35 : codeur de Hamming non systématique **H(3,7)**

➢ Position des r bits de contrôle

Les **r** bits de contrôle sont placés dans le bloc envoyé aux positions d'indice une puissance de **2** en comptant a partir de la gauche. Ainsi, en notant **r1 r2 r3** les bits de contrôle et **A B C D** bits de données, le bloc envoyé est :

M = r1 r2 D r3 C B A.

- **Calcul des r bits de contrôle**

Les **r** bits de contrôle sont calculés en utilisant une matrice de contrôle **H**, représentée ci-dessous pour **r=3**.

$$H = \begin{array}{c} \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{matrix} \\ \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix} \end{array}$$

Figure III.36 : matrice de contrôle H du code de Hamming **H(3,7)**

La colonne i de la matrice représente en binaire la valeur de i.

L'équation **G. H^T = 0** définit la relation entre les deux matrices d'un code. (**G** la matrice génératrice)

Les **r** bits de contrôle sont tels qu'en considérant le vecteur $M=\begin{pmatrix} r1 \\ r2 \\ D \\ r3 \\ C \\ B \\ A \end{pmatrix}$, on ait $H.M=\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$.

On obtient ainsi **3** équations scalaires que doivent vérifier les **r** bits de correction :

$$\begin{cases} r1+D+C+A=0_{modulo2} \\ r2+D+B+A=0_{modulo2} \\ r3+C+B+A=0_{modulo2} \end{cases}$$

Les opérations d'addition sont faites à modulo 2. Par exemple : 1+0=1 ; 1+1=0 ; 1+1+1=1 ; etc.

Le bloc **M** parfaitement déterminé est alors envoyé.

Le tableau ci-après indique les différentes possibilités de M :

A	B	C	D	r1=XOR (A,C,D)	r2=XOR (A,B,D)	r3=XOR (A,B,C)	Sorties multiplexeur M=r1.r2.D.r3.C.B.A
0	0	0	0	0	0	0	0000000
0	0	0	1	1	1	0	1110000
0	0	1	0	1	0	1	1001100
0	0	1	1	0	1	1	0111100
0	1	0	0	0	1	1	0101010
0	1	0	1	1	0	1	1011010
0	1	1	0	1	1	0	1100110
0	1	1	1	0	0	0	0010110
1	0	0	0	1	1	1	1101001
1	0	0	1	0	0	1	0011001
1	0	1	0	0	1	0	0100101
1	0	1	1	1	0	0	1010101
1	1	0	0	1	0	0	1000011
1	1	0	1	0	1	0	0110011
1	1	1	0	0	0	1	0001111
1	1	1	1	1	1	1	1111111

Tableau III.1 : la table de vérité du codeur de Hamming non systématique **H(3,7)**

Le signal à la sortie du codeur de Hamming est donné à la figure III.37.

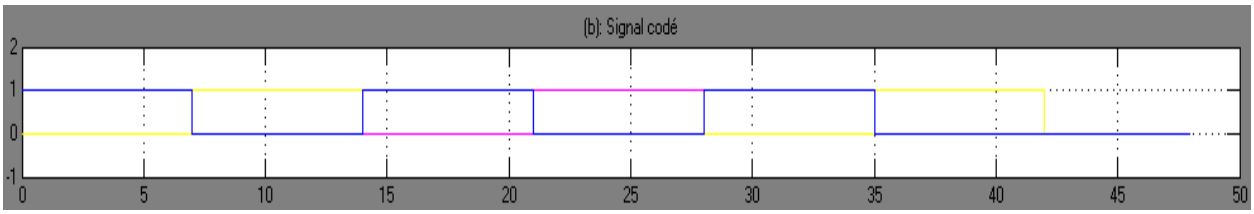


Figure III.37: le signal à la sortie du codeur de Hamming

III.3.1.2 Code Détecteur d'erreurs de Hamming H(3,7) (décodeur de Hamming) :

La correction des erreurs dans un tel code passe obligatoirement par la construction d'un syndrome d'erreur. Celui-ci sera construit très simplement à l'aide de **3** portes *ou-exclusif* comme dans le cas du codeur. Le détecteur est présenté figure III.38. Le syndrome d'erreur est codé sur **3** bits et permet de localiser les bits erronés dans le mot code reçu. Dans le cas d'un syndrome d'erreur nul, on considère que la réception de l'information est correcte.

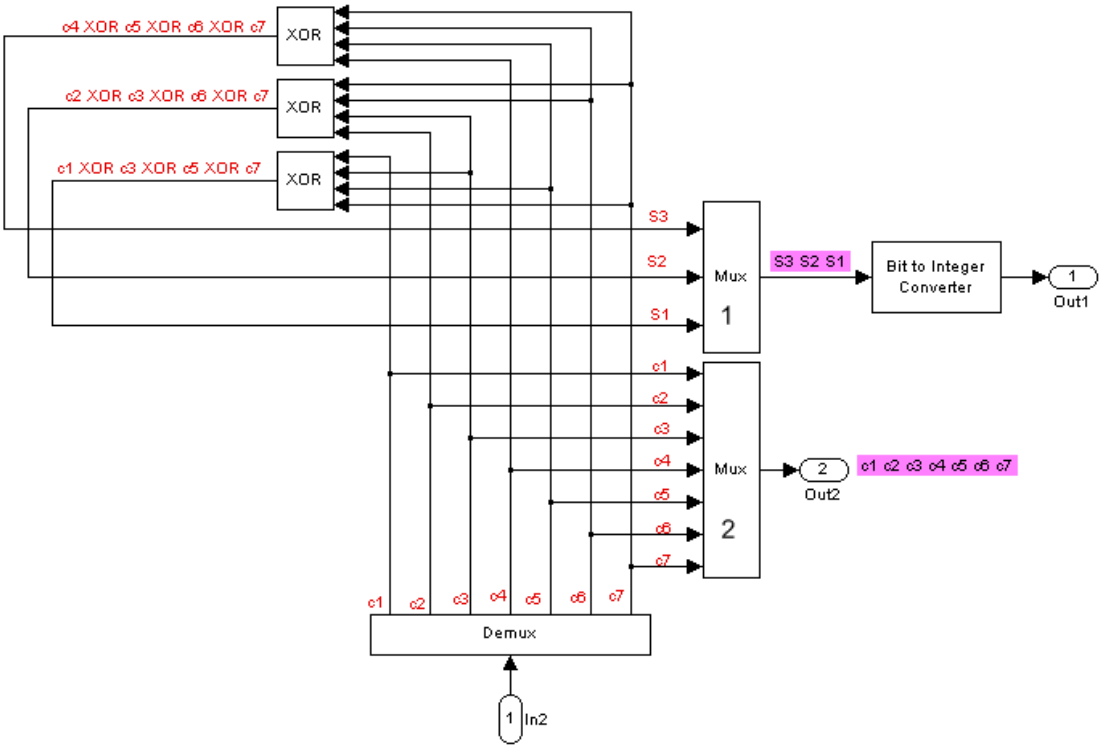


Figure III.38 : décodeur de Hamming non systématique **H(3,7)**

- **Réception des données et vérification**

On reçoit le bloc **C = c1 c2 c3 c4 c5 c6 c7** qui peut être différent du bloc **M** si il y a eut des perturbations sur la ligne.

Si on considère qu'il n'y a eut qu'une seule erreur de transmission, alors on peut écrire : C = M + E ou E est un bloc contenant 6 bits à 0 et 1 bit à 1.

Les positions des 0 et du 1 sont inconnues dans le bloc.

On calcule le vecteur S tel que : $S=\begin{pmatrix} S1 \\ S2 \\ S3 \end{pmatrix} = H.C= H.(M.E) =H.M+H.E= H.E.$

Finalement, S est une des colonnes de la matrice de contrôle dont l'indice nous donne la position de l'erreur dans le bloc C. L'erreur est corrigée en changeant le bit considéré d'état. **S3 S2 S1** est le code binaire de position de l'erreur dans le bloc C que nous obtenons à partir des équations suivantes :

$$\begin{cases} S1= (c1+c3+c5+c7)_{\text{modulo2}}. \\ S2= (c2+c3+c6+c7)_{\text{modulo2}}. \\ S3= (c4+c5+c6+c7)_{\text{modulo2}}. \end{cases}$$

Si S3 = S2 = S1 = 0, alors il n'y a pas eut d'erreur.

Le tableau ci-dessous indique le principe de détection :

c1	c2	c3	c4	c5	c6	c7	S3=XOR (c4,c5,c6,c7)	S2=XOR (c2,c3,c6,c7)	S1=XOR (c1,c3,c5,c7)	Sorties multiplexeur1 S3 S2 S1
0	0	0	0	0	0	0	0	0	0	000
1	1	1	0	0	0	0	0	0	0	000
1	0	0	1	1	0	0	0	0	0	000
0	1	1	1	1	0	0	0	0	0	000
0	1	0	1	0	1	0	0	0	0	000
1	0	1	1	0	1	0	0	0	0	000
1	1	0	0	1	1	0	0	0	0	000
0	0	1	0	1	1	0	0	0	0	000
1	1	0	1	0	0	1	0	0	0	000
0	0	1	1	0	0	1	0	0	0	000
0	1	0	0	1	0	1	0	0	0	000
1	0	1	0	1	0	1	0	0	0	000
1	0	0	0	0	1	1	0	0	0	000
0	1	1	0	0	1	1	0	0	0	000
0	0	0	1	1	1	1	0	0	0	000
1	1	1	1	1	1	1	0	0	0	000
1	0	0	1	1	0	1	1	1	1	111 (erreur)
0	0	1	1	0	1	0	0	0	1	001 (erreur)

Tableau III.2 : la table de vérité du décodeur de Hamming non systématique **H(3,7)**

III.3.1.3 Code correcteur d'erreur (correcteur d'erreur de Hamming) :

Le dernier maillon de l'opération de décodage est le correcteur d'erreur (figure III.39). Il utilise le syndrome d'erreur calculé précédemment pour corriger le mot-code reçu. Il s'agit ici d'additionner ce dernier, avec un mot de même longueur dont le seul bit à **1** indique la position de l'erreur. Ce vecteur d'erreur est construit à partir de la valeur du syndrome d'erreur. La réalisation du correcteur nécessite quelques portes logiques, des portes *and* et *non* pour la construction du vecteur d'erreur et des portes *ou-exclusif* pour l'addition.

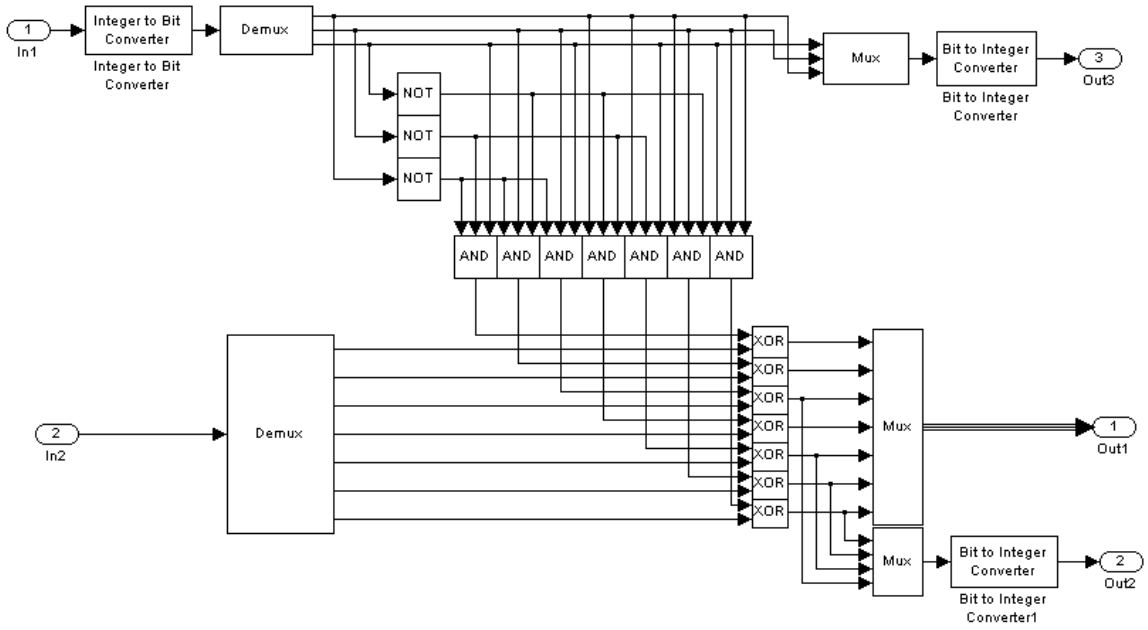


Figure III.39 : Correcteur d'erreur de Hamming non systématique **H(3,7)**

Le signal a la sortie du correcteur d'erreur de Hamming est donné à la figure III.40.

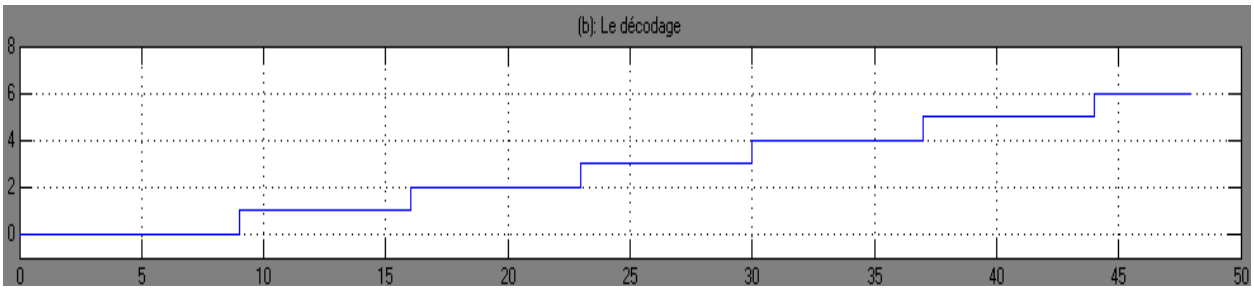


Figure III.40 : le signal à la sortie du correcteur d'erreur de Hamming

III.3.2 Modèle complet de la chaîne de transmission avec codage de canal :

La chaîne complète intégrant le codeur, le décodeur et le correcteur est présentée à la figure III.41. Ici on constate tout de suite l'intérêt de ce type de codage car bien que le détecteur ait détecté des erreurs (erreurs simples), celles-ci ont toutes été corrigées. Nous avons complété notre modèle de la chaîne de transmission numérique avec codage du canal.

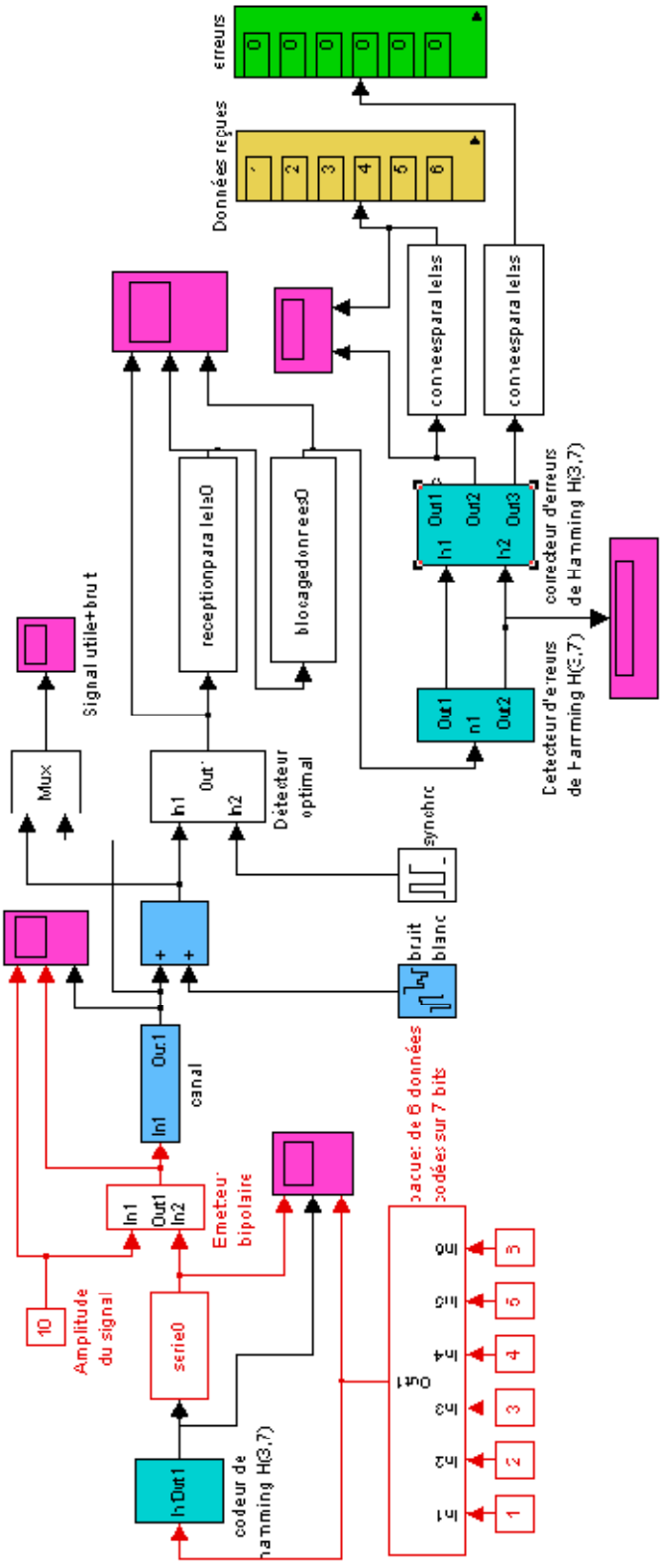


Figure III.41 : chaîne de transmission avec codage du canal

III.4 Conclusion :

Une telle manipulation est intéressante dans le sens où elle nous permet de bien visualiser les principes de base du codage de canal. Les possibilités offertes par une telle manipulation sont multiples dans le sens où il est possible de mettre en œuvre et de vérifier l'efficacité de nombreux types de codage : codages de Hamming + parité, codage cyclique... Il est aussi possible de tester pour un même code des rendements différents.

Finalement, nous avons présenté ici une ébauche de chaîne de transmission que l'on peut faire évoluer à souhait, en intégrant des composants plus proches de la réalité : canal, filtres, amplis, convertisseurs.

Si les valeurs émises sont les mêmes que les valeurs reçus, donc les deux chaînes fonctionnent très bien en simulation.

CHAPITRE IV :

Mode d’emploi de la simulation

Références: [1], [3], [4]

IV - Mode d'emploi de la simulation

IV.1 Introduction :

Nous allons simuler et observer durant ce chapitre le comportement des deux chaînes de transmission une fois les modèles sont complets.

IV.2 L'ouverture du modèle :

Pour ouvrir le modèle la première étape est de cliquer deux fois sur l'icône de Matlab se trouvant sur le bureau. Une fenêtre de commande de Matlab apparait comme il est indiqué dans la figure IV.1.

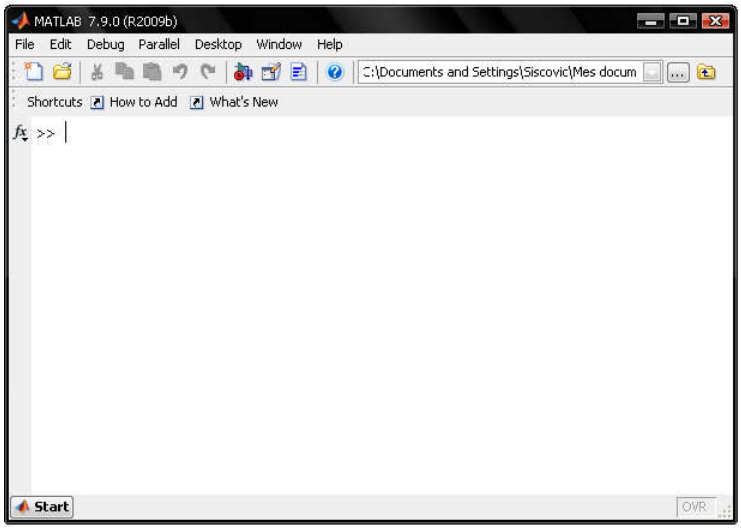


Figure IV.1 : la fenêtre principale du Matlab

La deuxième étape consiste à ouvrir l'outil Simulink à l'aide de l'icône de Simulink se trouvant sur la barre d'outils en utilisant la commande de Matlab « Simulink » comme le montre la figure IV.2.

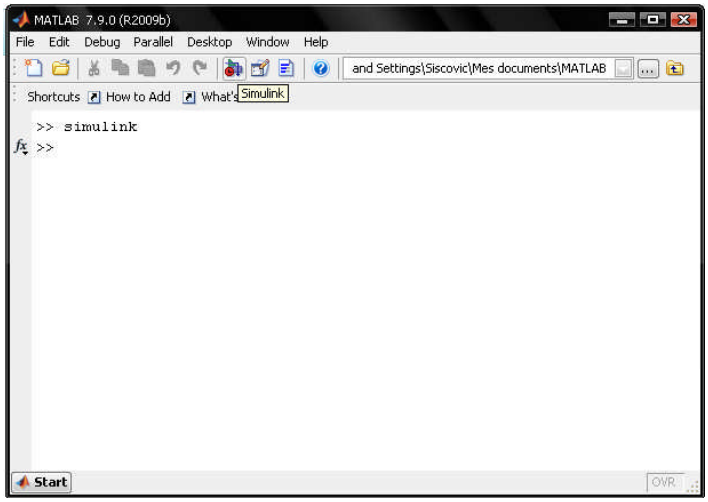


Figure IV.2 : l'icône du Simulink et la commande Simulink

Après cette étape, nous avons la fenêtre de Simulink montré à la figure IV.3.

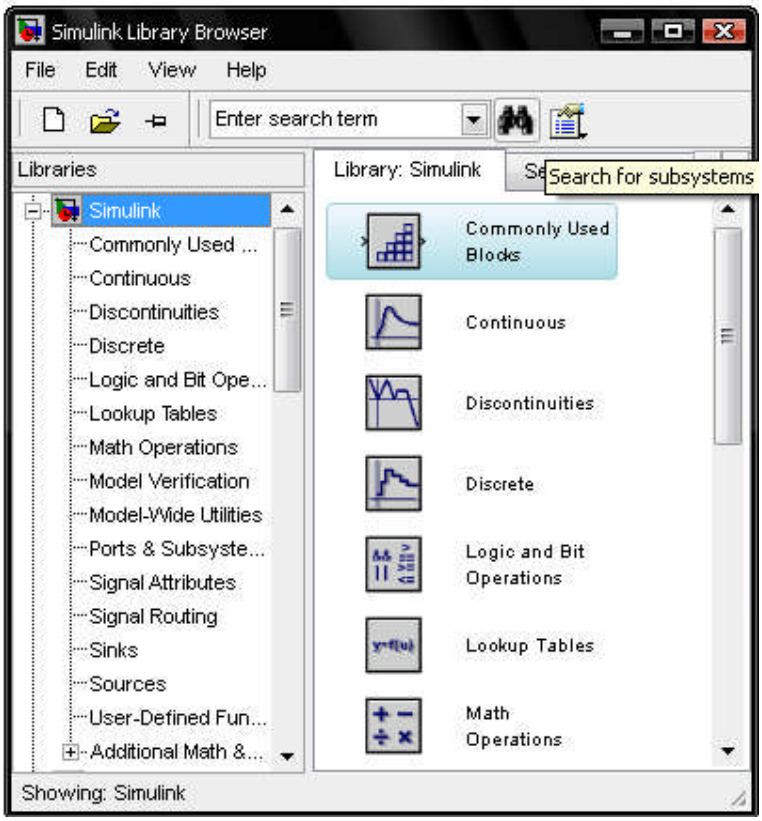


Figure IV.3 : la fenêtre du Simulink

Maintenant dans la fenêtre de Matlab nous cliquons sur le bouton « ouvrir » se trouvant sur la barre d'outils pour ouvrir le fichier de Matlab nommé «chainesanscodage » dans le répertoire «MATLAB ». La boîte de dialogue est montrée à la figure IV.4.

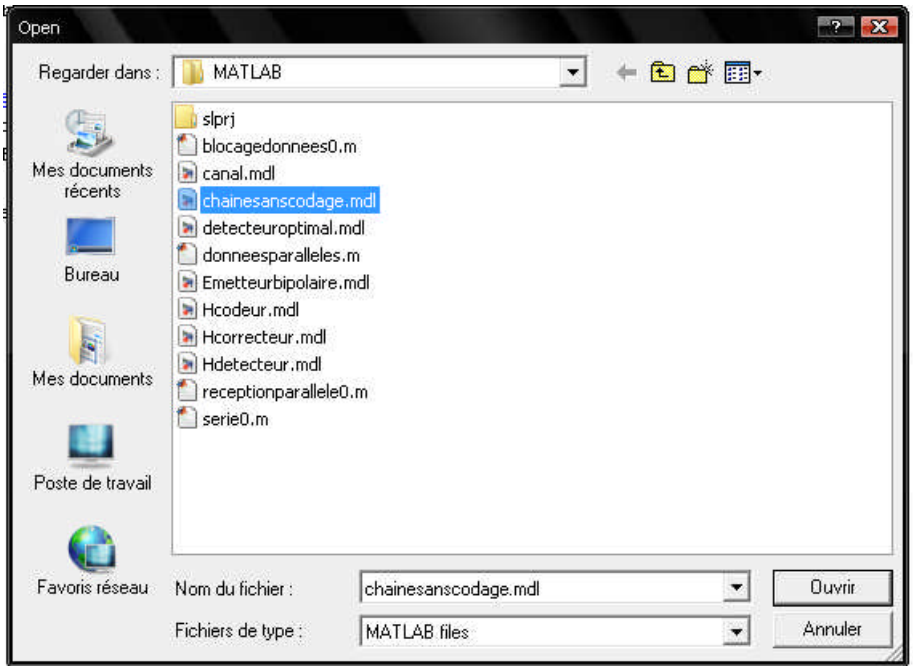


Figure IV.4 : la boîte de dialogue permettant d'ouvrir un fichier MATLAB

Nous pouvons ouvrir le modèle nommé « chainesanscodage » trouvant sur la fenêtre de Simulink. Le modèle va apparaître comme il est indiqué dans la figure IV.5.

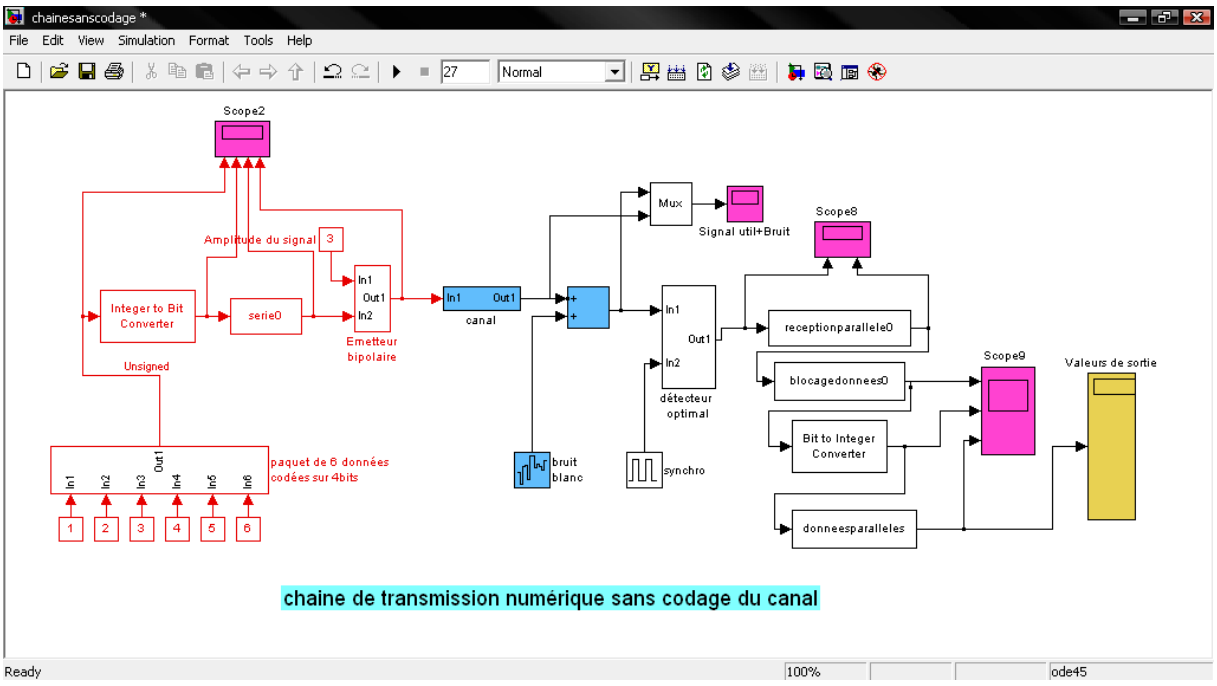


Figure IV.5 : la fenêtre du modèle de la chaîne de transmission numérique

IV.3 Le démarrage de la simulation :

Dans la fenêtre précédente, il faut cliquer sur le bouton « start simulation »comme indiqué à la figure IV.6.

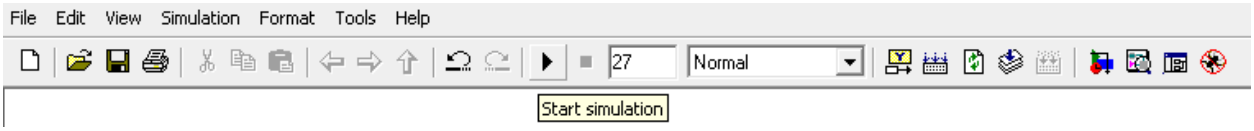


Figure IV.6 : démarrage de la simulation

Pendant la simulation, nous avons dans le coin gauche en bas le mot anglais « running » (la simulation est en cours). Quant la simulation est finie, nous avant le mot « Ready » (prête) donc la simulation est terminée.

IV.4 Chaîne sans codage du canal :

Chainesanscodage.mdl présente une chaîne de transmission constituée d'un **émetteur en rouge** d'un **récepteur en noir** et d'un **canal en présence de bruit en bleu.**
6 données codées sur 4 bits sont envoyées à travers le canal.

Paramétrage du fichier sous la fenêtre MATLAB command window :

- ***tech*** *(temps d'échantillonnage)=1s*
- ***lgmotcode*** *(nombre des bits)=4*
- ***nbdonne*** *(nombre des données)=6*

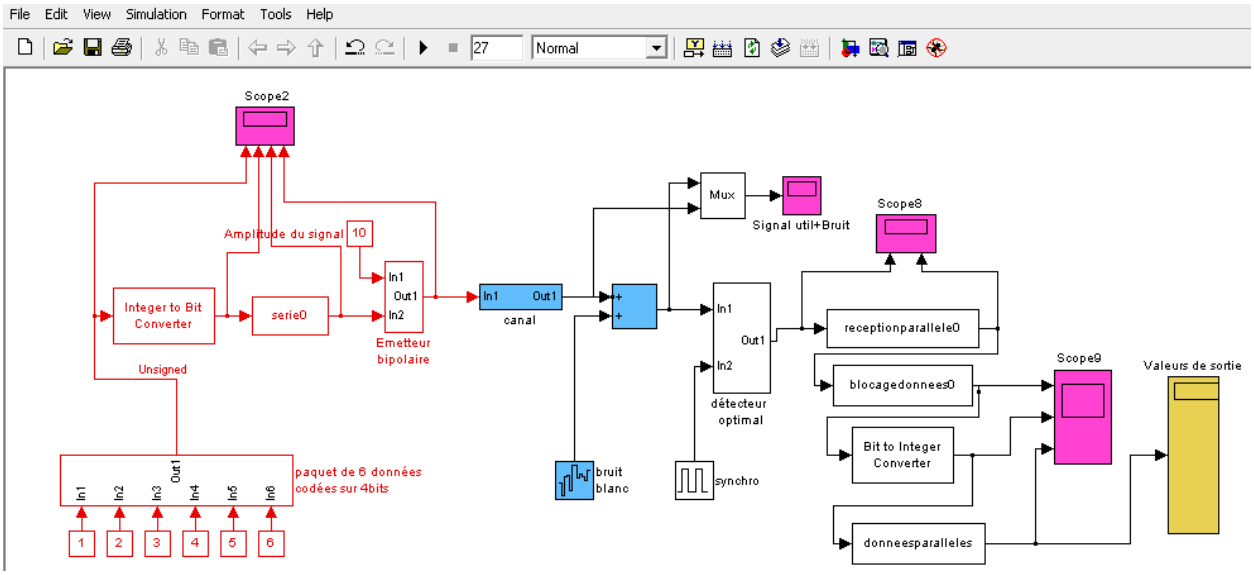
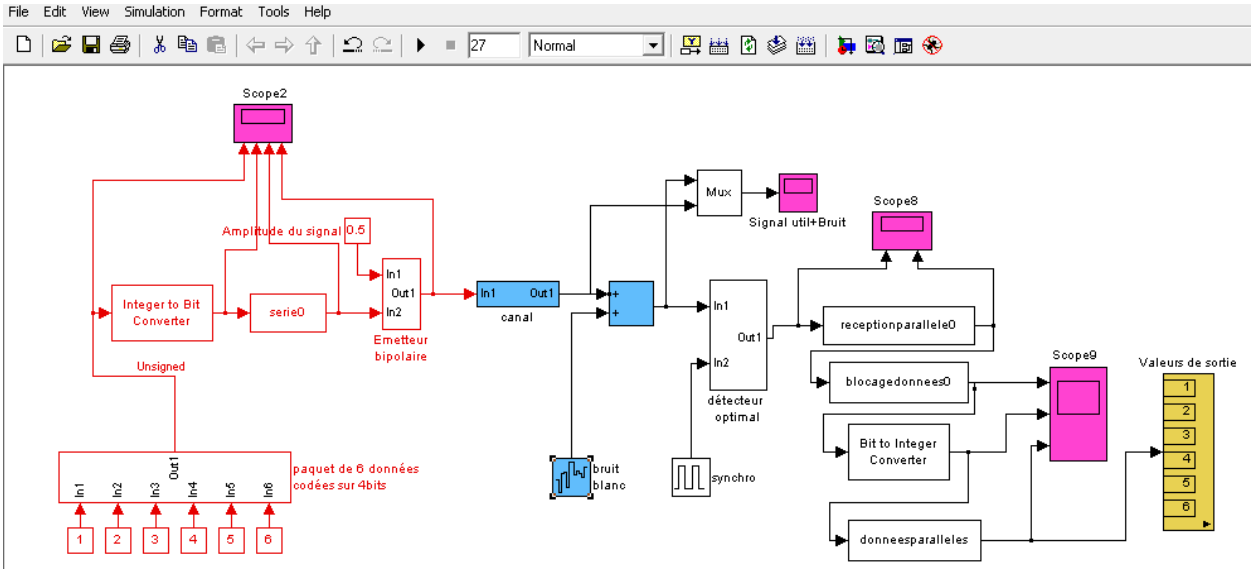


Figure IV .7 : chaîne de transmission numérique sans codage du canal

Absence de bruit :

Amplitude du signal utile = **0,5 V**
Puissance du bruit = **0,0 dB**
Temps d'échantillonnage =**0,05 U**

Toutes les données reçues sont transmises sans erreurs.



La figure IV.8 montre le signal utile sans bruit.

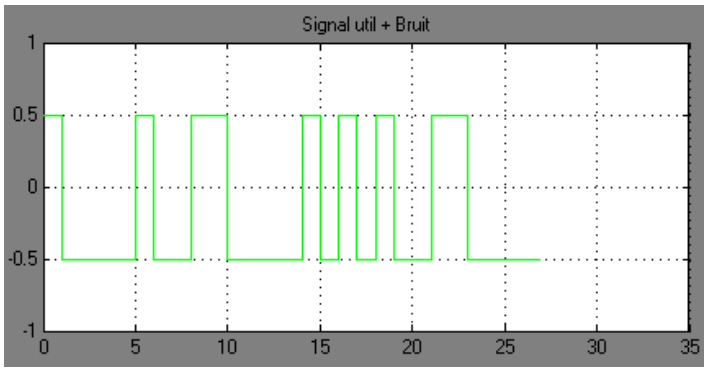
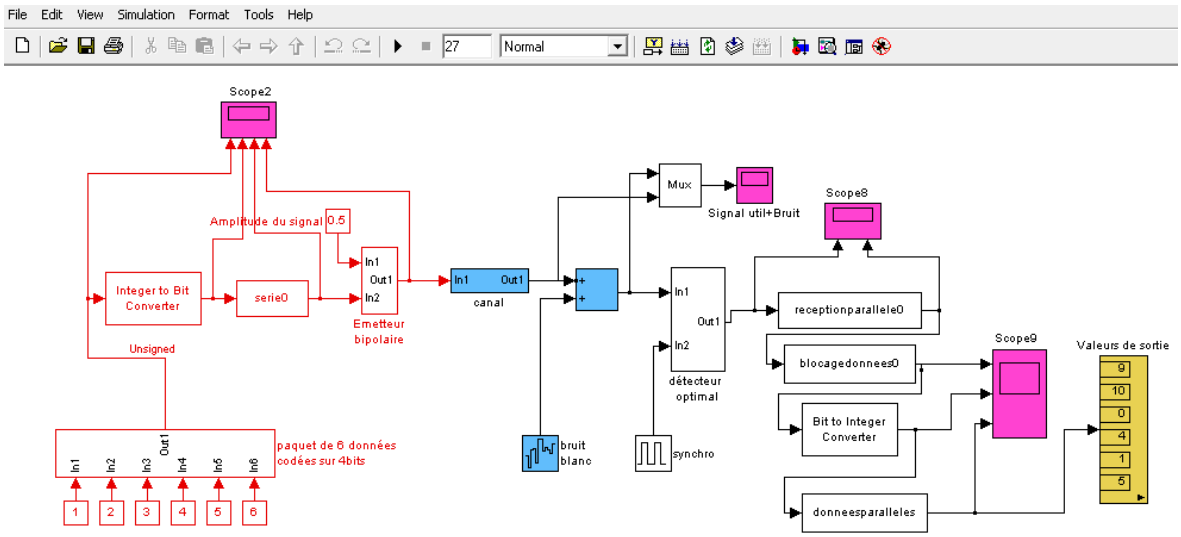


Figure IV.8 : signal utile sans bruit

Présence de bruit :

1°/ Amplitude du signal utile = **0,5 V**
Puissance du bruit = **0,5 dB**
Temps d'échantillonnage= **0,05 U**

Toutes les données reçues sont fausses excepté la valeur de sortie 4



La figure IV.9 montre le signal utile + Bruit.

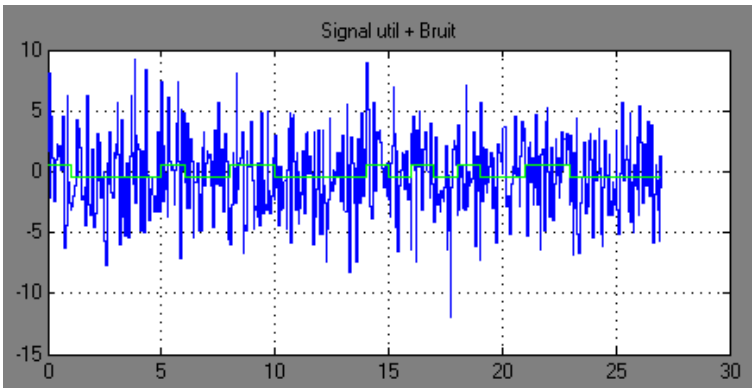
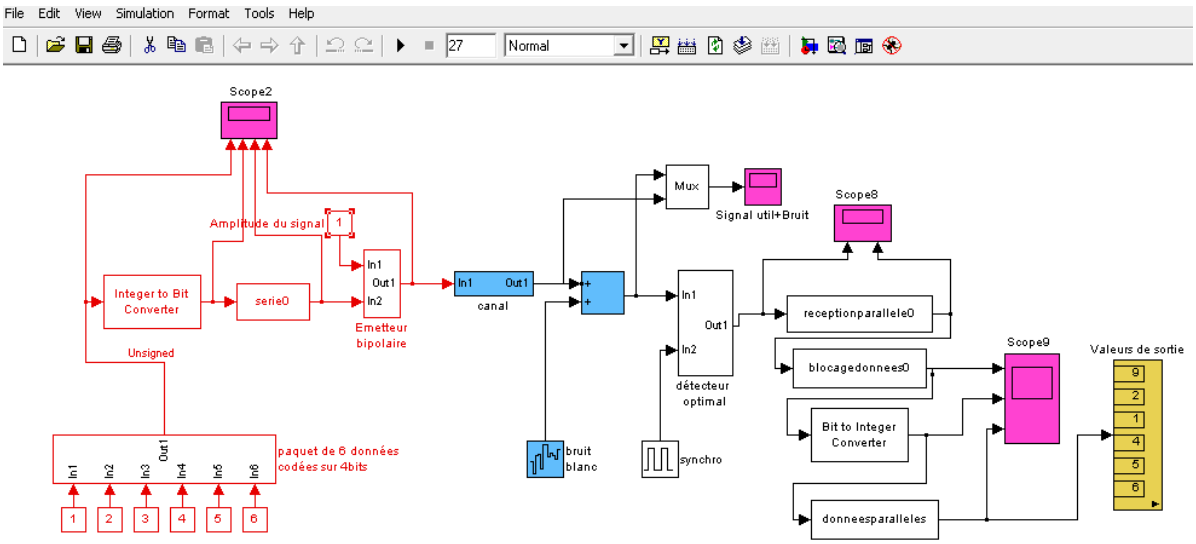


Figure IV.9 : signal utile + Bruit

2°/ On augmente l'amplitude du signal utile = **1 V**.

2 données reçues sur 6 sont fausses.



La figure IV.10 montre le signal utile + Bruit.

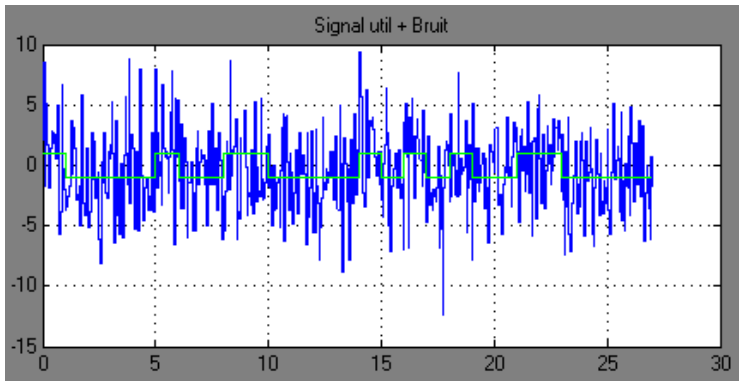
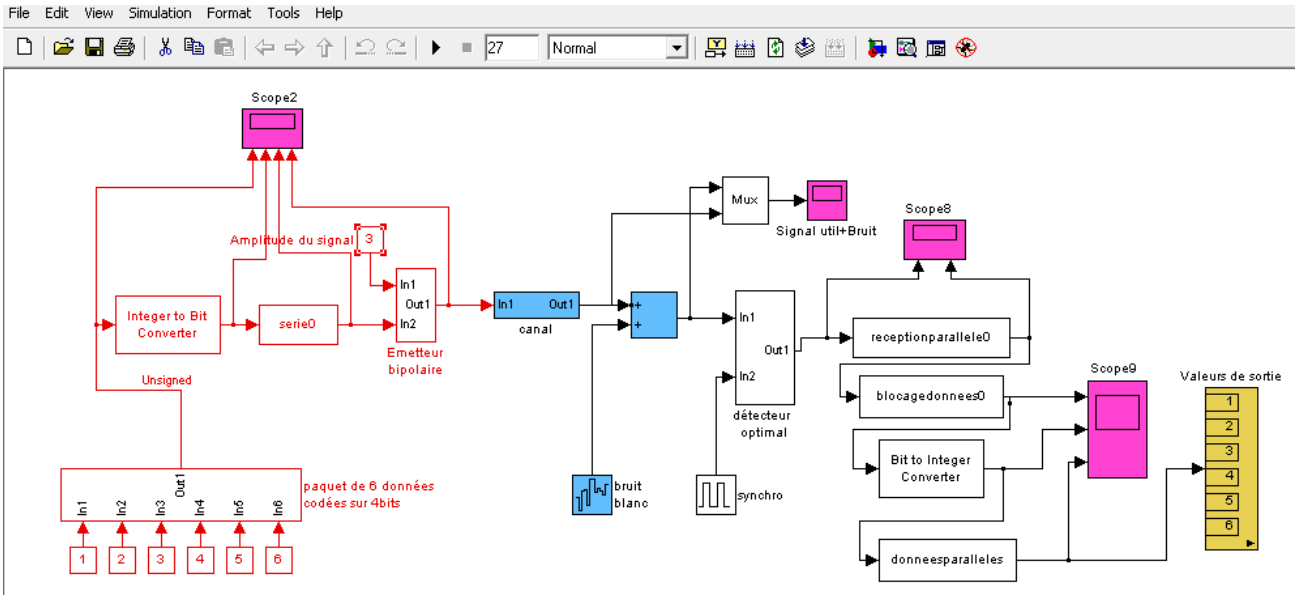


Figure IV.10 : signal utile + Bruit

3°/ On augmente encore l'amplitude du signal utile = **3 V**, **10 V,** etc.

Toutes les données reçues sont transmises sans erreur car dans ce cas la puissance du signal utile est tellement importante que l'on voit bien se dégager les amplitudes –1 et 1 correspondant aux symboles 0 et 1 (cf. Oscilloscope).



La figure IV.11 montre le signal utile + Bruit.

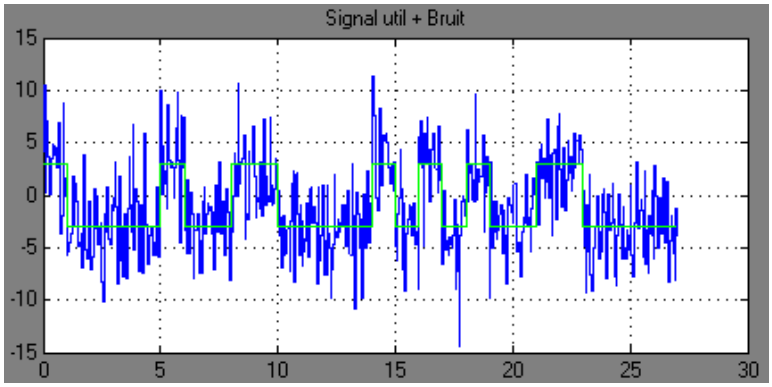


Figure IV.11 : signal utile + Bruit

Amplitude du signal utile = **10 V**

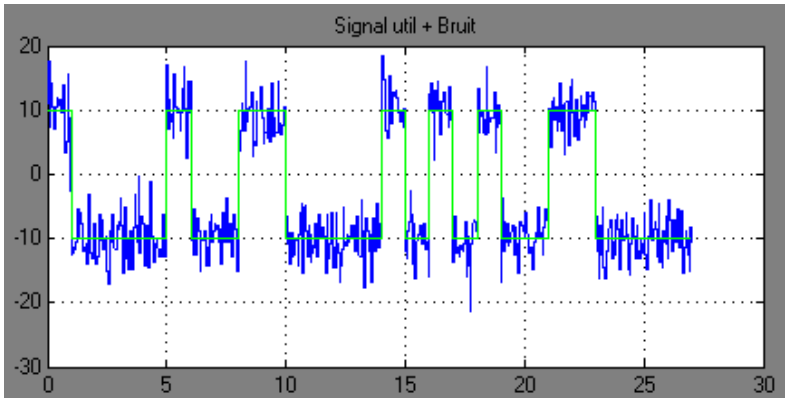
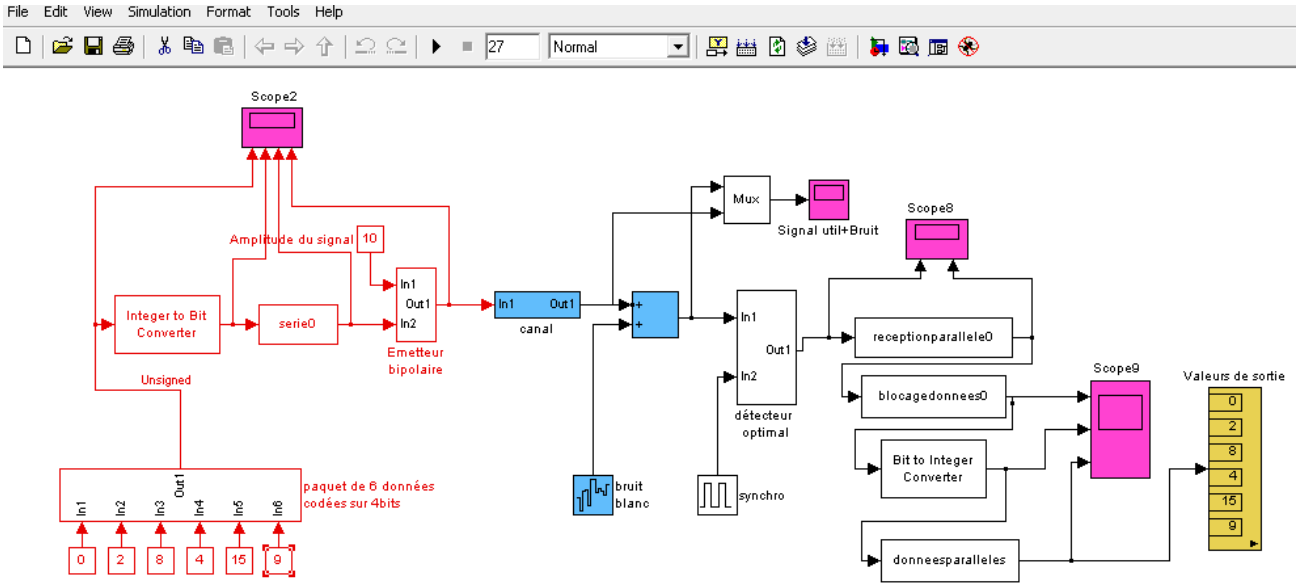
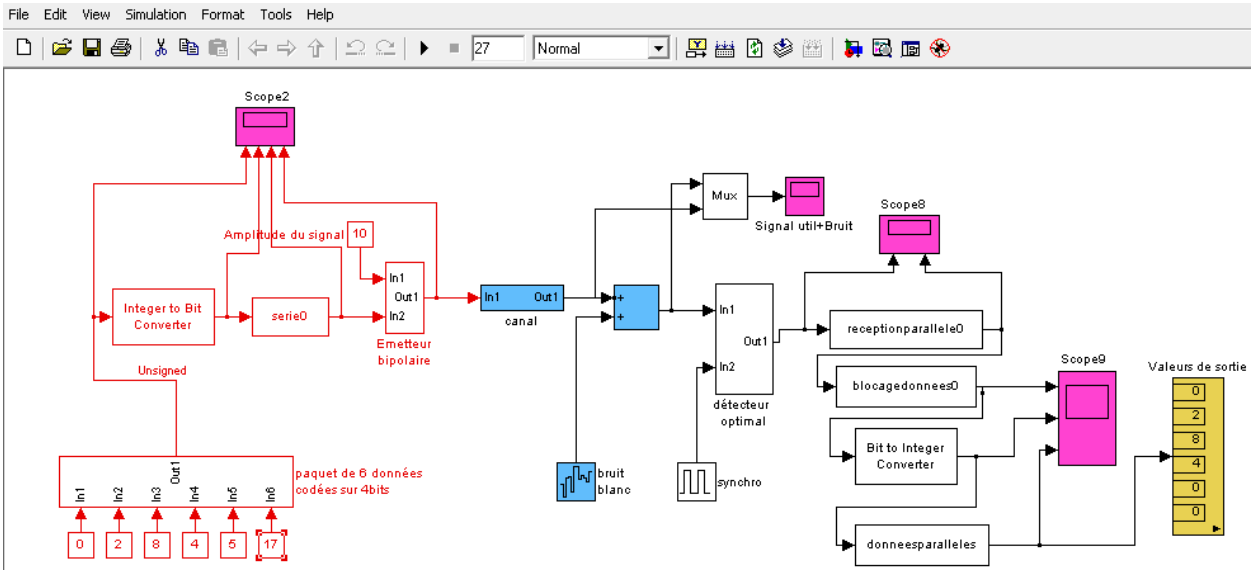


Figure IV.12 : signal utile + Bruit

On fait un changement pour les données transmises, toutes les données reçues sont transmises sans erreurs (de 0 à 15 on les écrit sur 4 bits).



Si on met des valeurs supérieures à 15, sauf les données à 4 bits sont reçues sans erreurs et les autres sont fausses.



IV.5 Chaîne avec codage du canal :

chaineaveccodage.mdl présente cette fois une chaîne de transmission constituée d'un **émetteur en rouge,** d'un **récepteur en noir** et d'un **canal en présence de bruit en bleu.** Un **codeur en cyan** (de type Hamming) a été introduit dans l'émetteur et un **décodeur en cyan** (constitué d'un détecteur d'erreurs et d'un correcteur d'erreurs) a été introduit dans le récepteur.
6 données codées sur 7 bits sont envoyées à travers le canal.
7 bits = 4 bits d'information + 3 bits de contrôle

Paramétrage du fichier sous la fenêtre MATLAB command window :
tech=1s, **lgmotcode**=7 et **nbdonne** =6

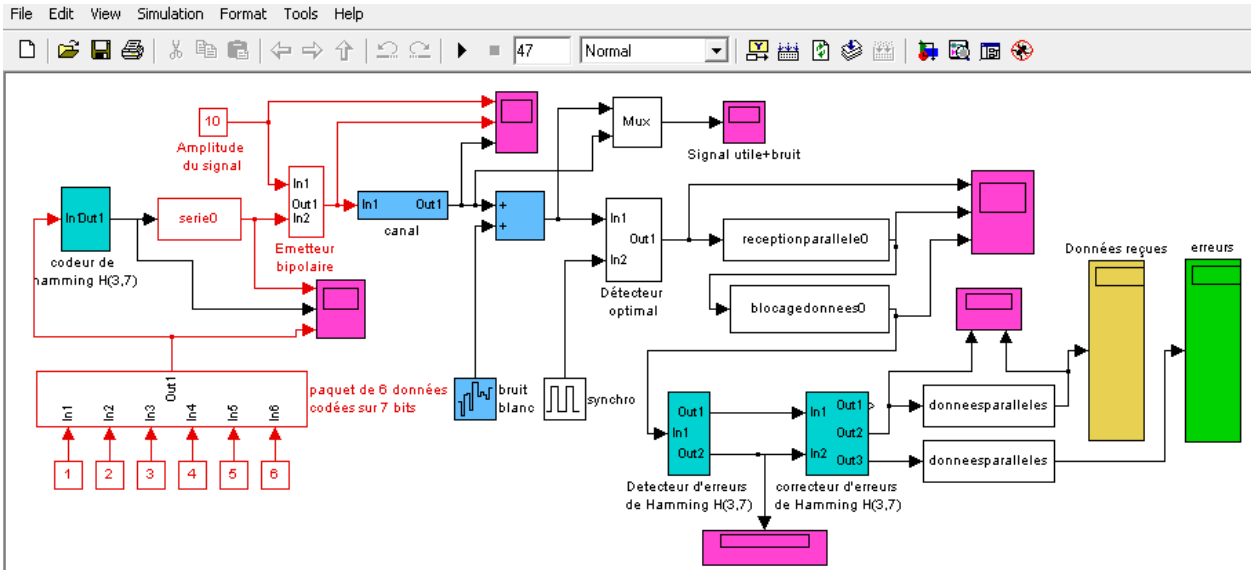
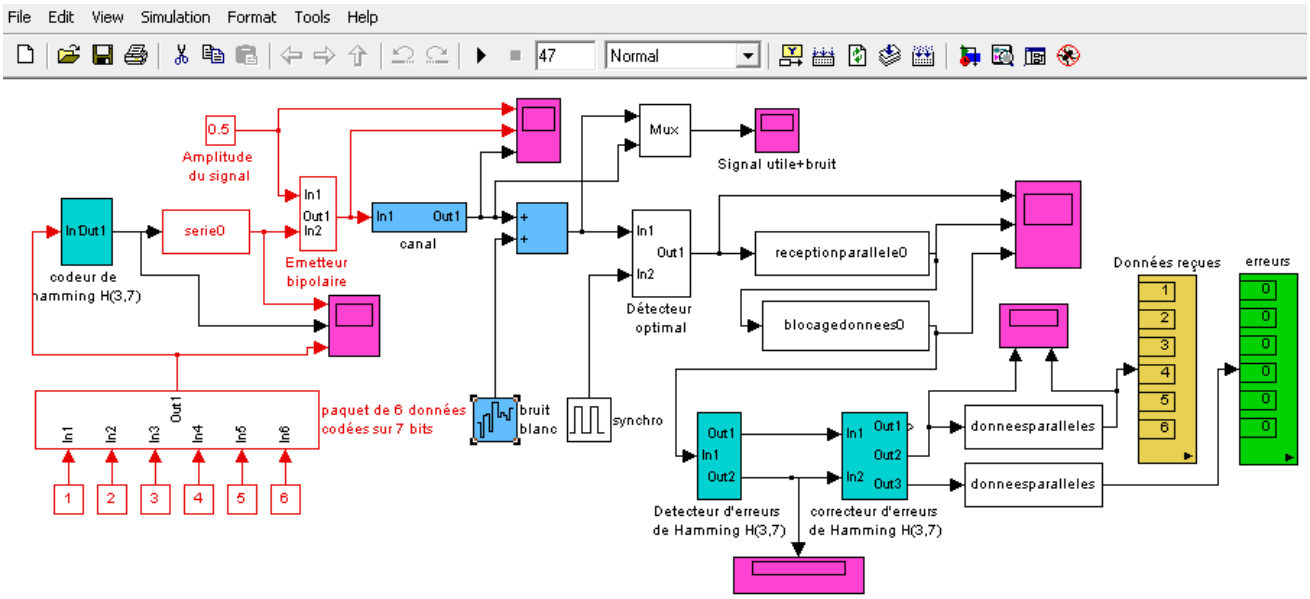


Figure IV.13 : la chaîne de transmission numérique avec codage du canal

Absence de bruit :

Amplitude du signal utile = **0.5 V**
Puissance du bruit = **0,0 dB**
Temps d'échantillonnage= **0,05 U**

Toutes les données reçues sont transmises sans erreurs.



La figure IV.14 montre le signal utile sans bruit.

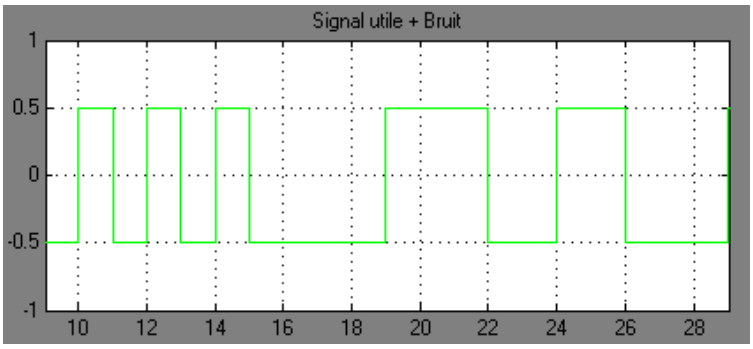
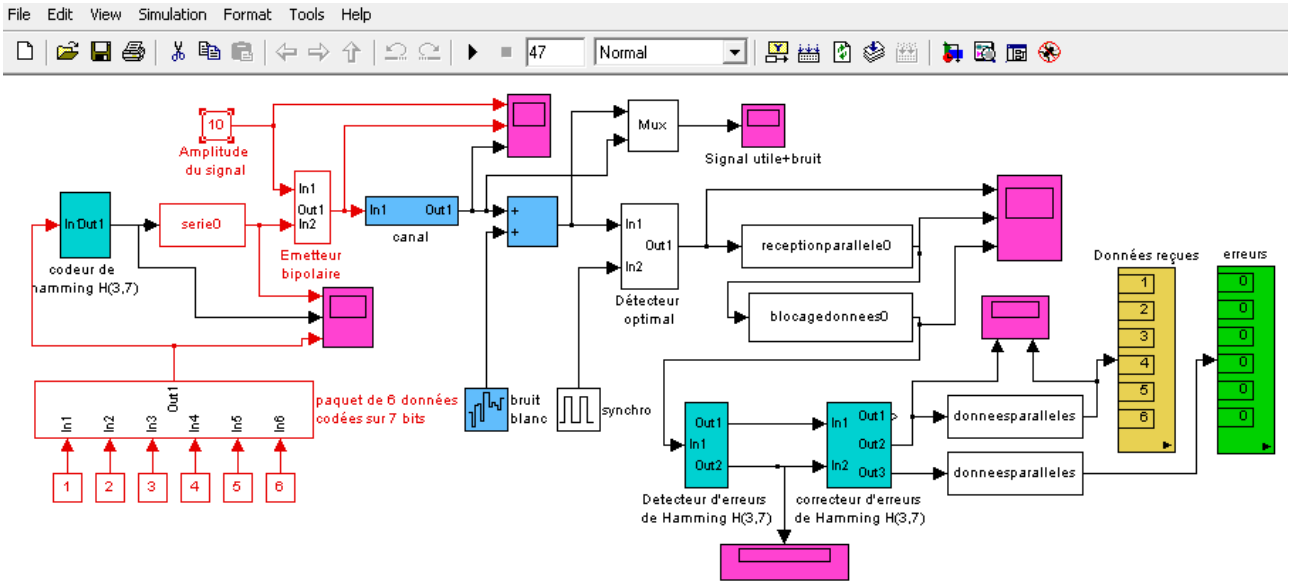


Figure IV.14 : signal utile sans bruit

Présence de bruit :

1°/ Amplitude du signal utile = **10 V**
Puissance du bruit = **0,5 dB**
Temps d'échantillonnage **0,05 U**

Toutes les données reçues sont transmises sans erreurs.
C'est normale car comme dans le cas précédent, la puissance du signal est importante et permet de bien séparer le signal du bruit.



La figure IV.15 montre le signal utile + Bruit.

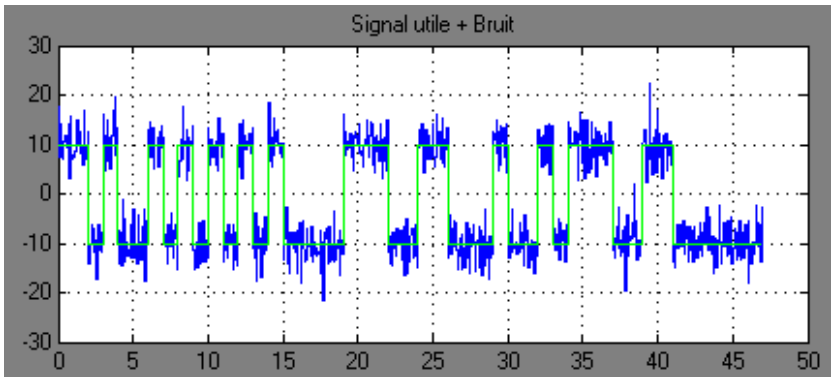
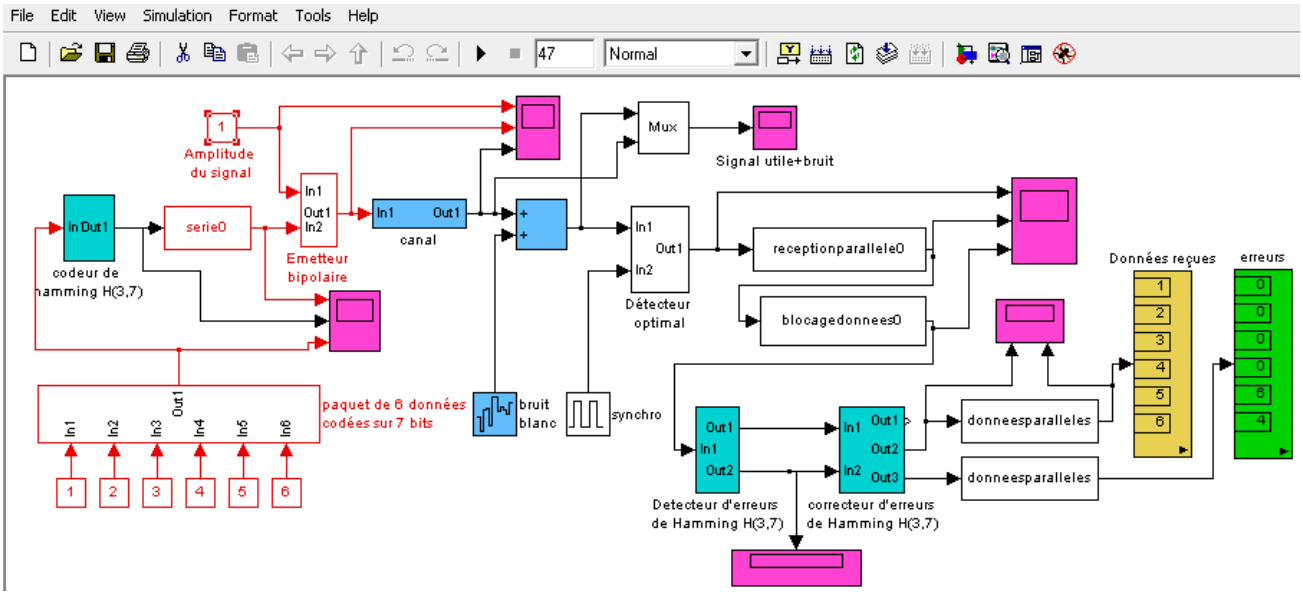


Figure IV.15 : signal utile + Bruit

2°/ On diminue l'amplitude du signal utile = **1 V**.

Le décodeur détecte 2 erreurs sur les 2 dernières données reçues.
Ces erreurs sont ensuite corrigées et finalement toutes les données émises sont restituées au récepteur (dans cet exemple il s'agit d'erreurs simples).



La figure IV.16 montre le signal utile + Bruit.

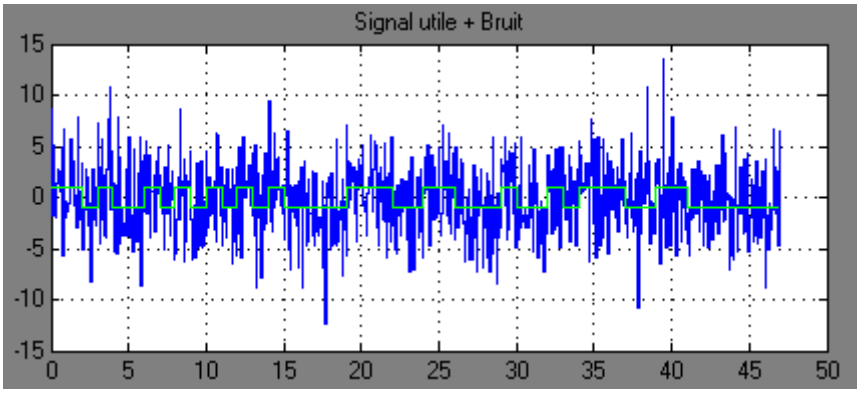
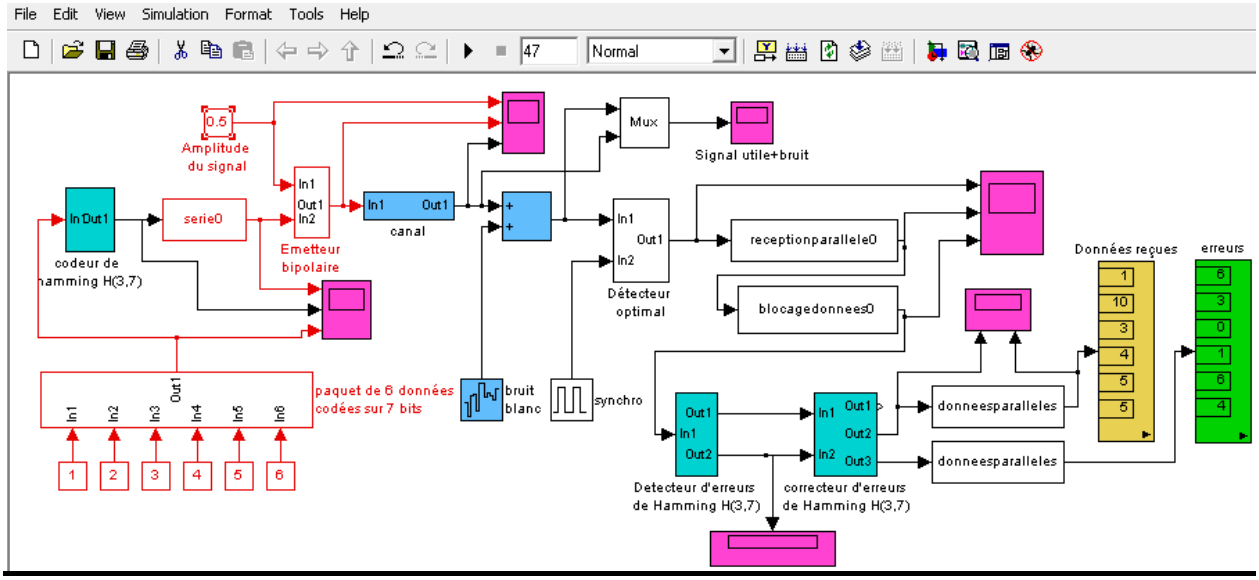


Figure IV.16 : signal utile + Bruit

3°/ On diminue encore l'amplitude du signal utile = **0.5 V**.

Le décodeur détecte 6 erreurs sur les 6 données reçues et est capable d'en corriger 4 (dans cet exemple nous sommes en présence de 4 erreurs simples et de 2 erreurs multiples).



La figure IV.17 montre le signal utile + Bruit.

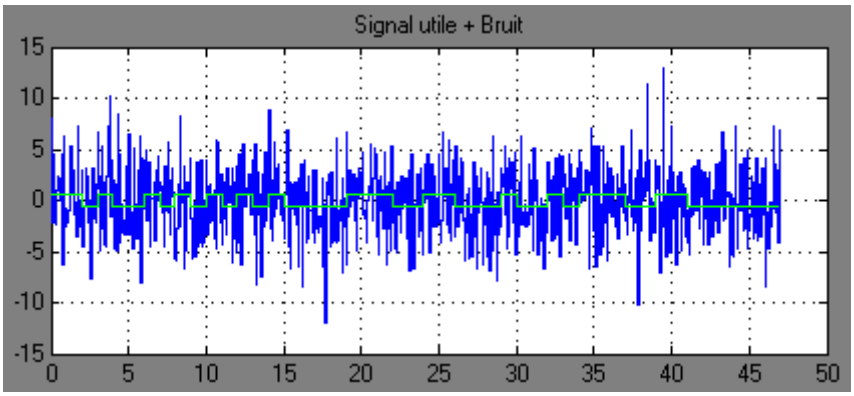


Figure IV.17 : signal utile + Bruit

IV.6 Avantages :

Si l'on regarde la visualisation oscilloscope sur l'exemple 2, nous voyons que le signal utile est complètement noyé dans le bruit et que le codage de canal permet quand même de l'extraire. Finalement cette technique permet d'assurer une relativement bonne fiabilité à l'information transmise sans nécessiter de puissance trop élevée pour le signal utile. C'est une technique très employée pour tous les systèmes de télécommunications et notamment pour les systèmes mobiles et les systèmes embarqués pour lesquels il y a des contraintes sévères au niveau des réserves en énergie.

IV. 7 Conclusion :

Nous avons testé la simulation des deux chaînes de transmission numérique, et nous avons visualisé les résultats en utilisant différents valeurs. Nous avons vu que les signaux sont identiques pour des valeurs bien définies, ce que signifie que les deux modèles fonctionnent très bien avec les paramètres choisis.

Conclusion générale

Dan ce travail, nous avons simulé sur la base de l'outil *Matlab-Simulink* une chaîne de transmission numérique. La transmission se fait en bande de base, sur un canal idéal perturbé par du bruit. Une telle manipulation permet aux étudiants de visualiser aisément l'efficacité des techniques de codage sur la sécurisation de l'information transmise, mais aussi en contrepartie la dégradation du débit. Notre chaîne de transmission peut être évolué à souhait en intégrant des composants plus proches de la réalité à savoir : canal, filtres, amplis, convertisseurs.

Ce projet nous a permis de voir une application immédiate et utile des notions abordées dans les cours de traitement du signal et de télécommunication. En étudiant le modèle de transmission, nous avons appris à utiliser Simulink avec une toolbox appropriée pour simuler le système. Nous pensons qu'il pourrait être intéressant de réaliser des Travaux pratiques simples en utilisant ces deux modèles.

Nous sommes plutôt satisfaits des capacités de l'outil pédagogique développé ; maintenant il est possible de réaliser des Travaux Pratiques très didactiques, permettant aux étudiants de réaliser eux-mêmes une modulation/démodulation en visualisant les signaux à chaque étape.

Il est aussi possible d'utiliser les blocs de la toolbox Communication et de réaliser une chaîne de transmission numérique, avec un canal à bande limitée bruité. Les performances en réception peuvent être évaluées en mesurant le taux d'erreur binaire.

La réalisation de travaux pratiques à partir des logiciels MatLab et Simulink est facilitée par l'information que l'on peut trouver sur le site Matworks.

A ce titre, nous proposons pour améliorer le modèle de développer une manipulation similaire sur la base d'une modulation FSK radiofréquence.

Bibliographie

[1] El Aissa Hamed (promotion : 2005/2006)

Thèse d'ingéniorat : « Simulation sous MATLAB/SIMULINK d'une chaîne de transmission numérique radio fréquence ».

Université Saad Dahleb : Département d'Aéronautique.

[2] Hadj-Said Naima, Ali-Pacha Adda, Belgoraf A., Belmekki B « Codage canal : codes correcteurs d'erreurs », ***RIST*** Vol, 14 n°01 Année 2004

[3] http://www.j3ea.org

[4] http://dx.doi.org/10.1051/bib-j3ea:2003014

[5] http://deptinfo.cnam.fr/Enseignement/Memoires/LUSTEAU.Franck/Pages/Les_codages.htm

[6] http://www.pdfqueen.com/html/aHR0cDovL3lsZXNjb3AuZnJlZS5mci9tcmltL2NvdXJzL2JhbmRlX2RlX2Jhc2UucGRm

[7] http://www.mathworks.com/access/helpdesk/help/toolbox/simulink/simulink.shtml

[8] http://fr.wikipedia.org/wiki/Non_Return_to_Zero

[9] http://www.cyber.uhp-nancy.fr/demos/GTRT-002/cha_1/cha_1_1.html

[10] MATHWORKS, Matlab 7.9.0 (R2009b) help

[11] Le site web: WWW.bensoire.free.fr/expe/simul_matlab.htm

Annexe

Programme de Fonction-S « serie0 » **:**

```
%La syntaxe de la Fonction-S%
function [sys,x0,str,ts] = serie0(t,x,u,flag,tech,lgmotcode)
switch flag,

% Initialization %
case 0,
sizes = simsizes;
sizes.NumContStates  = 0;%Nombre d'états continus%
sizes.NumDiscStates  = 1;%Nombre d'états discrets%
sizes.NumOutputs     = 1;%Nombre de sorties%
sizes.NumInputs      = lgmotcode;%Nombre d'entrées%
sizes.DirFeedthrough = 1;%Matrice D vide%
sizes.NumSampleTimes = 1;%Nombre de lignes matrices ts%
sys = simsizes(sizes);
x0  = 1;
str = [];
ts  = [tech 0];

% Update %
case 2,%Mise à jour de l'état discret%
   if x<lgmotcode
      sys = x(1)+1;
    else
      sys(1)=1;
    end

% Outputs %
case 3,%Calcul des sorties%
   sys=u(x);

% Terminate %
case 9,%Fin de simulation%
    sys = [];

% Unexpected flags %
otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end
```

Programme de Fonction-S « receptionparallele0 » **:**

```
%La syntaxe de la Fonction-S%
function [sys,x0,str,ts] =
receptionparallele0(t,x,u,flag,tech,lgmotcode)
switch flag,

% Initialization %
case 0,
sizes = simsizes;
sizes.NumContStates  = 0;%Nombre d'états continus%
sizes.NumDiscStates  = lgmotcode+1;%Nombre d'états discrets%
sizes.NumOutputs     = lgmotcode;%Nombre de sorties%
sizes.NumInputs      = 1;%Nombre d'entrées%
sizes.DirFeedthrough = 1;%Matrice D vide%
sizes.NumSampleTimes = 1;%Nombre de lignes matrices ts%
sys = simsizes(sizes);
for i=1:lgmotcode
   %x0(i)=0;
   x0 = zeros(size(i));
end
x0(lgmotcode+1)  = lgmotcode;
str = [];
ts  = [tech 0];

% Update %
case 2, %Mise à jour de l'état discret%
   if x(lgmotcode+1)<lgmotcode
      sys(lgmotcode+1) = x(lgmotcode+1)+1;
   else
      sys(lgmotcode+1)=1;
   end
   sys(1:lgmotcode)=x(1:lgmotcode);
   sys(x(lgmotcode+1))=u;

% Outputs %
case 3,%Calcul des sorties%
   sys=x(1:lgmotcode);

% Terminate %
case 9,%Fin de simulation%
    sys = [];

% Unexpected flags %
  otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end
```

Programme de Fonction-S « blocagedonnees0 » :

```
%La syntaxe de la Fonction-S%
function [sys,x0,str,ts] =
blocagedonnees0(t,x,u,flag,tech,lgmotcode)
switch flag,

% Initialization %
case 0,
sizes = simsizes;
sizes.NumContStates  = 0;%Nombre d'états continus%
sizes.NumDiscStates  = lgmotcode+1;%Nombre d'états discrets%
sizes.NumOutputs     = lgmotcode;%Nombre de sorties%
sizes.NumInputs      = lgmotcode;%Nombre d'entrées%
sizes.DirFeedthrough = 1;%Matrice D vide%
sizes.NumSampleTimes = 1;%Nombre de lignes matrices ts%
sys = simsizes(sizes);
for i=1:lgmotcode+1
   x0(i)=0;
   end;
str = [];
ts  = [tech 0];

% Update %
case 2,%Mise à jour de l'état discret%
   if x(lgmotcode+1)<lgmotcode+1
      sys(lgmotcode+1)=x(lgmotcode+1)+1;
      sys(1:lgmotcode)=x(1:lgmotcode);
   else
      sys(lgmotcode+1)=2;
      sys(1:lgmotcode)=u(1:lgmotcode);
   end;

% Outputs %
case 3,%Calcul des sorties%
       sys(1:lgmotcode)=x(1:lgmotcode);
  %%%%%%%%%%%%
  % Terminate %
  %%%%%%%%%%%%
case 9,%Fin de simulation%
    sys = [];

  %%%%%%%%%%%%%%%%%%%%
  % Unexpected flags %
  %%%%%%%%%%%%%%%%%%%%
  otherwise
    error(['Unhandled flag = ',num2str(flag)]);

end
```

Programme de fonction-S « donneesparalleles » :

```
%La syntaxe de la Fonction-S%
 function [sys,x0,str,ts] =
donneesparalleles(t,x,u,flag,tech,lgmotcode,nbdonne)
switch flag,

% Initialization %
case 0,
sizes = simsizes;
sizes.NumContStates  = 0;%Nombre d'états continus%
sizes.NumDiscStates  = nbdonne+2;%Nombre d'états discrets%
sizes.NumOutputs     = nbdonne;%Nombre de sorties%
sizes.NumInputs      = 1;%Nombre d'entrées%
sizes.DirFeedthrough = 1;%Matrice D vide%
sizes.NumSampleTimes = 1;%Nombre de lignes matrices ts%
sys = simsizes(sizes);
for i=1:nbdonne+1
   x0(size(i))=0;
end
x0(nbdonne+2)=1;
str = [];
ts  = [tech 0];

% Update %
case 2,%Mise à jour de l'état discret%
   if x(nbdonne+1)<lgmotcode+2
      sys(nbdonne+1)=x(nbdonne+1)+1;
      sys(nbdonne+2)=x(nbdonne+2);
      sys(1:nbdonne)=x(1:nbdonne);
   else
      sys(nbdonne+2)=x(nbdonne+2)+1;
      sys(nbdonne+1)=3;
      sys(1:nbdonne)=x(1:nbdonne);
    sys(x(nbdonne+2))=u;
   end
    % Outputs %
 case 3,%Calcul des sorties%
       sys=x(1:nbdonne);

 % Terminate %
case 9, %Fin de simulation%
    sys = [];

  %%%%%%%%%%%%%%%%%%%
  % Unexpected flags %
  %%%%%%%%%%%%%%%%%%%
  otherwise
    error(['Unhandled flag = ',num2str(flag)]);

end
```

L'opérateur binaire XOR combine l'état de 2 bits selon le tableau suivant :

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Table de vérité XOR (OU EXCLUSIF)

L'opérateur unaire NOT inverse l'état d'un bit selon le tableau suivant :

A	NOT A
0	1
1	0

Table de vérité NOT (NON)

L'opérateur binaire AND combine l'état de 2 bits selon le tableau suivant :

A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1

Table de vérité AND (ET)