Exercices dirigés

UTC505/USRS4D

-Eléments Physiques-

E. Gressier-Soudan

2021-2022

Ce polycopié a été élaboré par l'équipe enseignante "Réseaux et protocoles" à partir d'exercices rédigés par M. Florin, et M. Gressier-Soudan, qu'ils en soient ici remerciés.

ED• La détection d'erreurs

Les erreurs de transmission sur un canal de transmission sont très pertubatrices pour un échange de données. Il faut savoir les détecter, c'est l'objet du CRC (Cyclic Redundancy Code). Puis il faut en tenir compte quand on dimensionne des architectures. C'est l'objet de cette séance d'exercices.

Quelques rappels et définitions :

Unité	Symbole	Valeur (bits)	Conversions
kilo-bit	Kb	10^3=1 000	
méga-bit	Mb	10^6=1 000 000	1Mb = 1000 Kb
giga-bit	Gb	10^9 =1 000 000 000	1Gb = 1000 Mb = 10^6 Kb
téra-bit	Tb	10^{12}=1 000 000 000 000	1Tb = 1000 Gb = 10^6 Mb

Unités multiples des bits

Unité	Symbole	Valeur (octets)	Conversions
kilo-octet	Ko	2^{10}=1024	
méga-octet	Mo	2^{20}=1 048 576	1Mo = 2^{10} Ko
giga-octet	Go	2^{30}=1 073 741 824	1Go = 2^{10} Mo = 2^{20} Ko
téra-octet	To	2^{40}=1 099 511 627 776	1To = 2^{10} Go = 2^{20} Mo

Unités multiples des octets (1 octet = 8 bits)

On trouve une table plus complète dans https://fr.wikipedia.org/wiki/Octet (13/01/2021). Cette source donne de nombreuses conversions, mises en garde, règles d'écriture.

Codification et Etats/Symboles/Valeurs :

- 1 bit permet de coder/représenter 2 (2^1) états/symboles/valeurs : 0 et 1 ;
- 2 bits permettent de coder/représenter 4 (2^2) états/symboles/valeurs : 00, 01, 10 et 11 ;
- 3 bits permettent de coder/représenter 8 (2^3) états/symboles/valeurs : 000, 001, 010, 011, 100, 101, 110 et 111 ;
- . . .
- n bits permettent de coder/représenter 2^n états/symboles/valeurs

Même source :
"Un octet peut représenter 2^8 soit 256 valeurs différentes. La valeur de tout octet peut s'écrire avec un entier naturel entre 0_{10} et 255_{10} compris. Elle peut aussi s'écrire avec huit chiffres binaires, entre 00000000_2 et 11111111_2 compris, ou avec deux chiffres hexadécimaux, entre 00_{16} et FF_{16} compris."
Pour la notation hexadécimale on peut aussi écrire les chiffres entre 0x00 et 0xFF compris.

Exercice 1 : Codes polynômiaux et détection d'erreurs de transmission

Opérations binaires, le XOR (ouexclusif logique) correspond à l'addition et le AND (et logique) à la multiplication :

Here is the bitwise equivalent operations of two bits P and Q:

p	q	F^0	NOR^1	Xq^2	$\neg p^3$	$\nrightarrow^4$	$\neg q^5$	XOR^6	$NAND^7$	AND^8	$XNOR^9$	q^{10}	$If/then^{11}$	p^{12}	$Then/if^{13}$	OR^{14}	T^{15}
1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
1	0	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
0	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
Bitwise equivalents		0	NOT (p OR q)	(NOT p) AND q	NOT p	p AND (NOT q)	NOT q	p XOR q	NOT (p AND q)	p AND q	NOT (p XOR q)	q	(NOT p) OR q	p	p OR (NOT q)	p OR q	1

Pour la division polynomiale, on utilise aussi la soustraction de bits un à un :

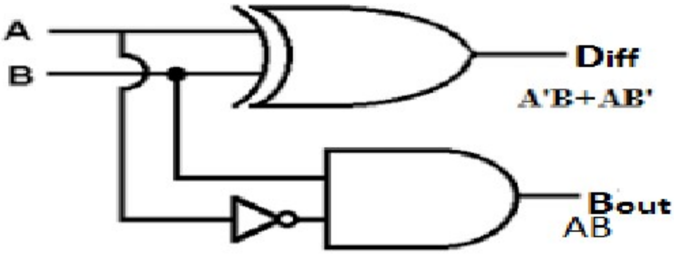


Fig1: Gate Level 1 bit Conventional Half Subtractor.

retenue

A	B	Bout	Diff	Comment
0	0	0	0	0-0=0,No borrow
0	1	1	1	0-1=-1,borrow 2, so:2-1=1
1	0	1	0	1-0=1, No borrow
1	1	0	0	1-1=0, No borrow

source : "Design and Implementation of Full Subtractor using CMOS 180 nm Technology", S. MonikashreeT., Dr. J. S. Baligar, 1421 ISSN: 2278 – 7798 All Rights Reserved © 2014 IJSETR

Entourés en rouge ce qu'on utilise dans la division polynomiale.

On considère le code polynomial sur 14 bits généré par le polynôme $G(x)=x^4 + x + 1$

On rappelle le principe des codes polynômiaux :

Les codes polynomiaux utilisent les mots de données pour les transformer en messages en utilisant la division polynomiale avec un polynome générateur G(x).

- A l'émission, via un circuit spécialisé qui implante la division polynomiale par G(x) un ensemble de bits de redondance est calculé. Ces bits sont le résultat de la division du polynome correspondant aux données multiplié par xle degré de G(x). Le reste de cette division est un polynome de degré inférieur à celui de G(x). On extrait de ce polynome reste la suite de bits correspondante qu'on colle à la fin des données pour fabriquer le message à transmettre.

- A la réception, quand on reçoit le message, on effectue à nouveau la division du polynome correspondant au message reçu par G(x) via le même circuit spécialisé.
  - Si le reste est nul, il n'y a pas eu d'erreur de transmission, les données sont traitées par la couche de liaison
  - Si le reste est non nul, alors il y a eu une erreur de transmission, les données sont mises à la poubelle.

Le calcul du CRC est fait par un un circuit électronique spécialisé (portes logiques, registres à décalage, horloge…) au fil de l'eau. C'est-à-dire au fur et à mesure que les bits de la trame sont transmis, le reste de la division polynomiale, qui est appelé CRC (Cyclic Redundancy Code) ou FCS (Frame Check Sequence) suivant la technologie, est accolé immédiatement derrière la partie données dès que celle-ci a été complètement envoyée. Ce circuit électronique est très rapide, implanté dans la couche physique même si c'est une fonction de la couche liaison. Le CRC ne fait pas l'objet d'un calcul complet en mémoire, sauf exception.

Quel est le nombre de bits de redondance qui seront ajoutés par G ?

Correction :

Le polynome G(x) est de degré 4, il sera ajouté 4 bits de redondance aux données. Le reste de la division polynomiale sera un polynome de degré 3 (x^3).

On veut calculer les bits de redondance du mot M=1101011011

Question 1

Donner D(x), le polynôme correspondant à M.

Correction :

M=1101011011, on va donc transformer M en D(x) polynome associé.

Pour accomplir cette transformation, on associe à chaque bit une puissance de x fonction de la position du bit dans M :

1 1 0 1 0 1 1 0 1 1

x^9 x^8 x^7 x^6 x^5 x^4 x^3 x^2 x^1 x^0

Ce qui donne le polynome suivant :

$1*x^9+1*x^8+0*x^7+1*x^6+0*x^5+1*x^4+1*x^3+0*x^2+1*x^1+1*x^0$

on peut l'écrire de façon plus lisible : $D(x) = x^9+x^8+x^6+x^4+x^3+x+1$

Pour des raisons pédagogiques, on gardera les deux formes dans la suite de l'exercice en passant de l'une à l'autre suivant l'intérêt pour l'explication.

Question 2

Calculer R(x), le reste de la division euclidienne de $D(x)*x^4$ par G(x). En déduire la valeur des bits de redondance ?

Correction :

Il faut fabriquer D(x)*x^4. Cette étape est indispensable pour augmenter le degré du polynome afin de créer un "espace" à la fin de la partie donnée pour y insérer le reste de la division polynomiale. On a besoin de 4 positions, d'où la multiplication par x^4 ce qui permet d'ajouter un polynome de degré 3.

On obtient donc :

$D(x)*x^4=1*x^{13}+1*x^{12}+0^*x^{11}+1*x^{10}+0*x^9+1*x^8+1*x^7+0*x^6+1*x^5+1*x^4+0*x^3+0*x^2+0*x^1+0*x^0$

Sous une forme plus habituelle en mathématique :

$D(x)*x^4=x^{13}+x^{12}+x^{10}+x^8+x^7+x^5+x^4$

Effectuons maintenant la division polynomiale par G(x), attention le "-" en numération binaire est équivalent à "+", donc ci-dessous aucun signe "-" n'apparaît.

$$
\begin{array}{l|l}
x^{13}+x^{12}+x^{10}+x^8+x^7+x^5+x^4 & \underline{x^4+x+1} \\
\underline{-(x^{13}+x^{10}+x^9)} & x^9+x^8+x^3+x \\
\quad x^{12}+x^9+x^8+x^7+x^5+x^4 & \\
\quad \underline{-(x^{12}+x^9+x^8)} & \\
\qquad x^7+x^5+x^4 & \\
\qquad \underline{-(x^7+x^4+x^3)} & \\
\qquad\quad x^5+x^3 & \\
\qquad\quad \underline{-(x^5+x^2+x)} & \\
\qquad\qquad x^3+x^2+x &
\end{array}
$$

On ne peut plus diviser car c'est un polynome de degré inférieur au degré de G(x), donc x^3+x^2+x est le reste de la division polynomiale de $x^{13}+x^{12}+x^{10}+x^8+x^7+x^5+x^4$ par x^4+x+1.

On peut le réécrire de la façon suivante :

$1*x^3+1*x^2+1*x+0*x^0$

La suite de bits, qui correspond au CRC et qu'on va ajouter après les données, est la suivante : 1110

Le message transmis sur le réseau est alors :

Premier bit à transmettre ← $1101011011\boxed{1110}$

On retrouve dans ce qui est encadré les bits de redondance que nous venons de calculer avec le reste R(x), ou encore le CRC.

Pour se faire une idée de ce qu'il se passe à la réception, on peut faire l'hypothèse qu'il n'y a aucune erreur de transmission, on reçoit donc :

$1101011011\boxed{1110}$

Le circuit CRC du récepteur va donc divisé avec G(x) le polynôme qui correspond au train de bits reçu :

$$
\begin{array}{l|l}
x^{13}+x^{12}+x^{10}+x^{8}+x^{7}+x^{5}+x^{4}+x^{3}+x^{2}+x & \underline{x^{4}+x+1} \\
\underline{-(x^{13}+x^{10}+x^{9})} & x^{9}+x^{8}+x^{3}+x \\
\quad x^{12}+x^{9}+x^{8}+x^{7}+x^{5}+x^{4}+x^{3}+x^{2}+x & \\
\quad \underline{-(x^{12}+x^{9}+x^{8})} & \\
\qquad x^{7}+x^{5}+x^{4}+x^{3}+x^{2}+x & \\
\qquad \underline{-(x^{7}+x^{4}+x^{3})} & \\
\qquad\quad x^{5}+x^{2}+x & \\
\qquad\quad \underline{-(x^{5}+x^{2}+x)} & \\
\qquad\qquad 0 &
\end{array}
$$

Le reste de la division polynomiale est bien 0 comme on l'attendait vu qu'il n'y a pas eu d'erreur de transmission.

Supposons qu'il y ait eu une erreur de transmission, et qu'on reçoive : 1101011011$\boxed{1100}$

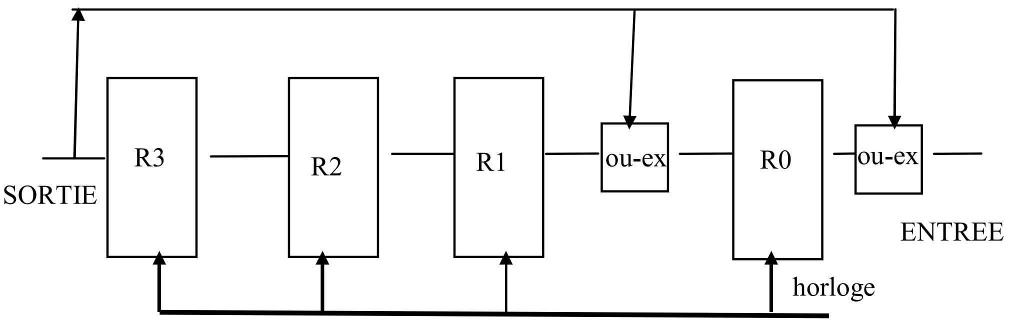
Le cicuit CRC fait le calcul suivant :

$$
\begin{array}{l|l}
x^{13}+x^{12}+x^{10}+x^{8}+x^{7}+x^{5}+x^{4}+x^{3}+x^{2} & \underline{x^{4}+x+1} \\
\underline{-(x^{13}+x^{10}+x^{9})} & x^{9}+x^{8}+x^{3}+x \\
\quad x^{12}+x^{9}+x^{8}+x^{7}+x^{5}+x^{4}+x^{3}+x^{2} & \\
\quad \underline{-(x^{12}+x^{9}+x^{8})} & \\
\qquad x^{7}+x^{5}+x^{4}+x^{3}+x^{2} & \\
\qquad \underline{-(x^{7}+x^{4}+x^{3})} & \\
\qquad\quad x^{5}+x^{2} & \\
\qquad\quad \underline{-(x^{5}+x^{2}+x)} & \\
\qquad\qquad x &
\end{array}
$$

Le reste de la division polynomiale est non nul, la trame est mise à la poubelle.

Pour les curieux : Mise en circuit logique d'une division polynomiale

Les bits de redondances sont en général fabriqués par un dispositif matériel assez simple, utilisant des portes logiques et des registres à décalage. Pour x^4+x+1, le circuit a le schéma ci-après :

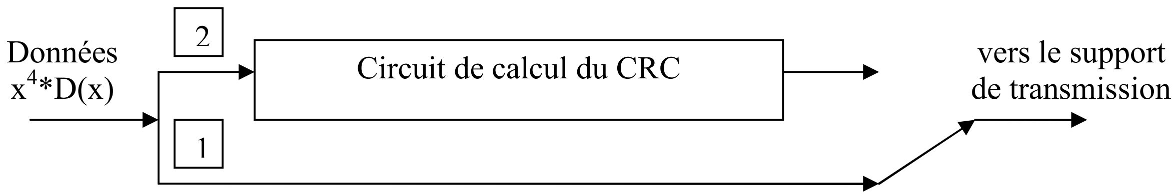


Le circuit recoit en entrée les bits du mot M (le bit de poid fort d'abord). Les registres R0 … R3 sont initialisés à zéro. Au 4ème top d'horloge, ils ont été amorcés en contenant les 4 premiers bits du message. Au 14ème coup, tous les bits de la partie donnée ont traversé le circuit. C'est pendant les 4 tops suivants que se fabriquent les bits de redondance qui seront ajoutés aux données.

Question

Donner le contenu des registres pour chaque coup d'horloge.

Le circuit CRC est utilisé de la façon suivante :



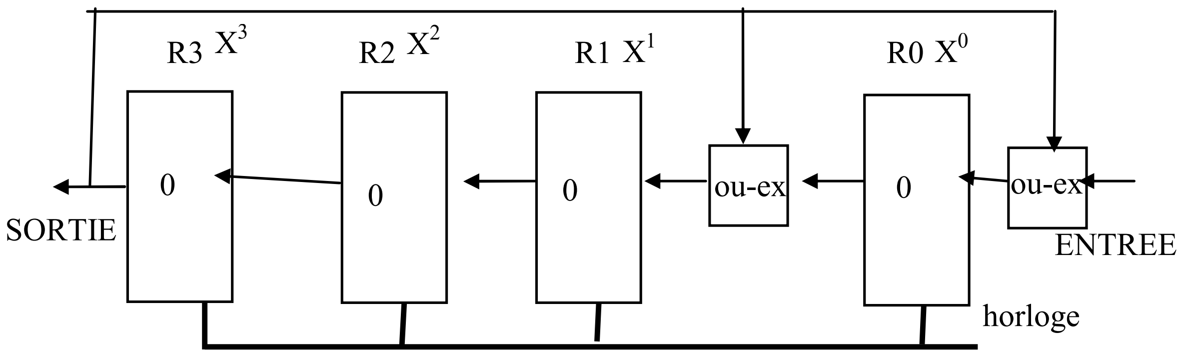
$x^4 * D(x)$ va vers les deux branches du circuit, en phase 1, pendant "d" tops d'horloge (nombre de bits de données, ici 10) les bits sortent vers le support de transmission, en phase 2 les bits du CRC sont poussés vers le réseau pendant "g" tops (longueur du CRC, ici 4).

Rappel : table de vérité du ou-exclusif :

Ou-ex	1	0
1	0	1
0	1	0

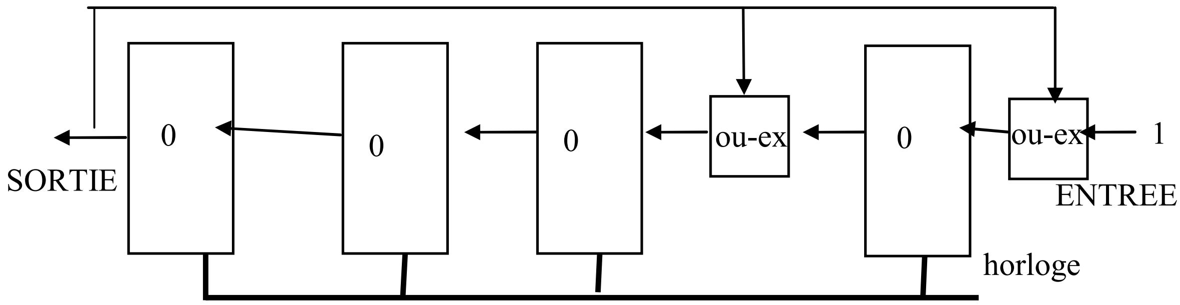
le ou exclusif fait fonction d'additionneur de deux bits[1] (modulo 2)

Situation avant que le circuit soit mis en marche pour la génération du CRC du message 1101011011, les registres contiennent des 0 :

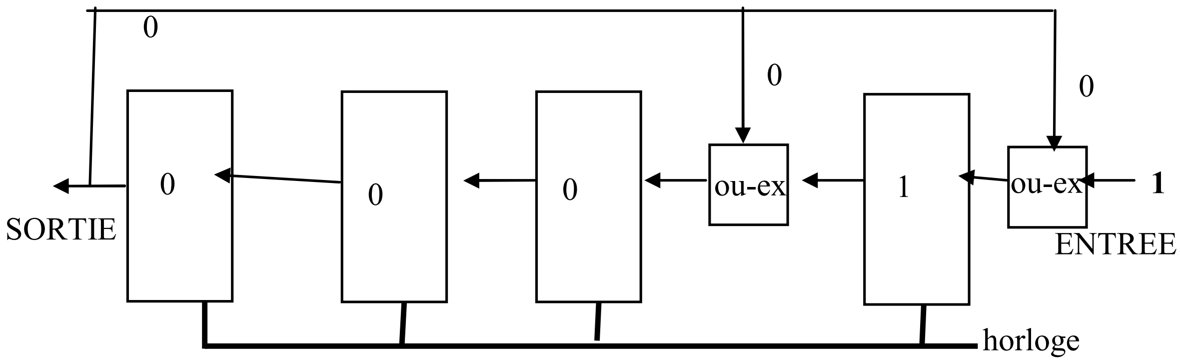


Amorce du circuit avec les 4 premiers bits du message, présentation du 1 :

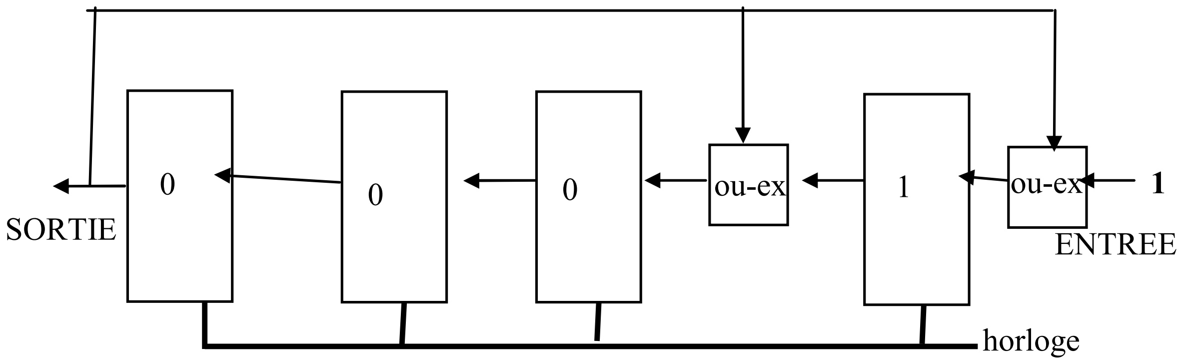
[1] Attention 1 + 1 = 0, on est en arithmétique binaire, c'est-à-dire modulo 2. Et dans notre cas, il n'y a pas de retenue.



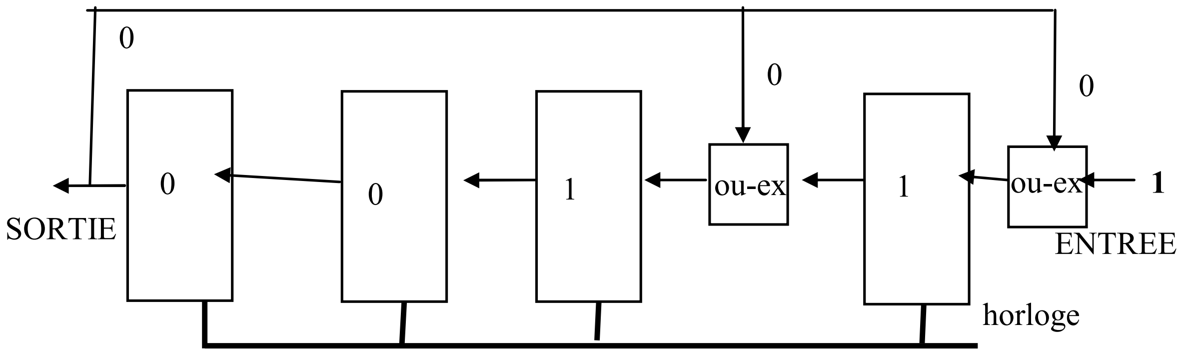
Lors de l'entrée du 1 au top d'horloge qui fait produire toutes les opérations en même temps dans le circuit logique et qui fait que ça se décale d'un registre à l'autre :



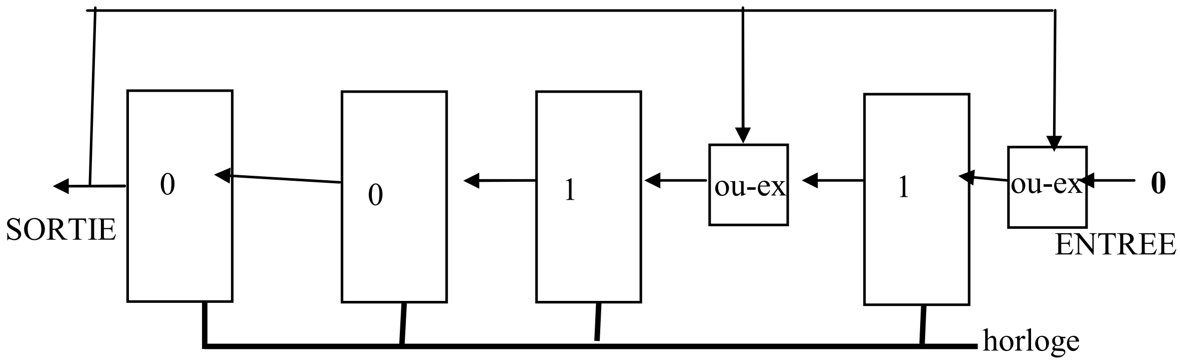
Présentation du 1 suivant :



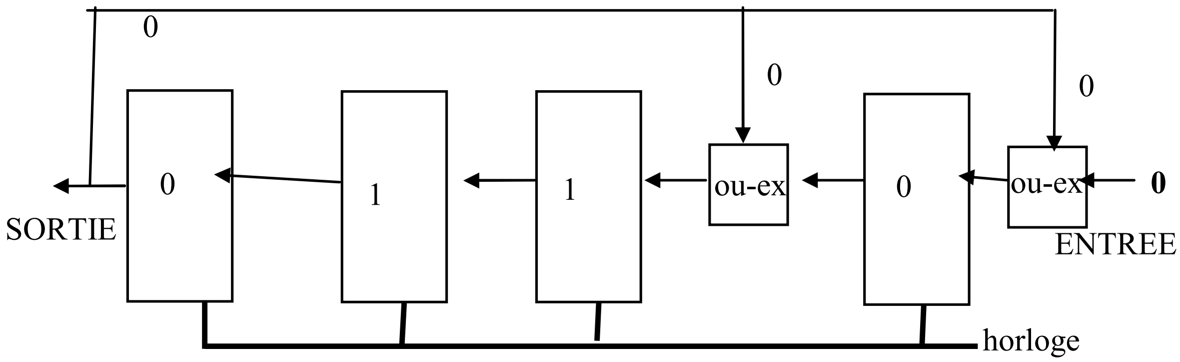
Lors de l'entrée du 1 suivant au top d'horloge :



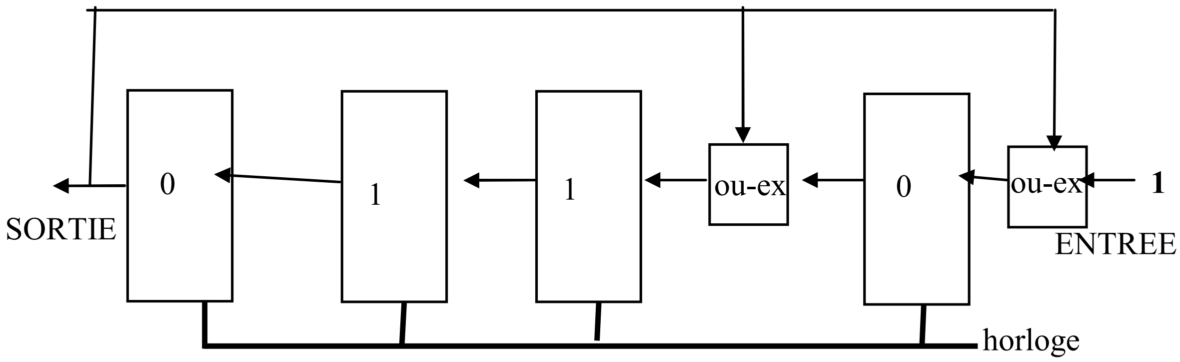
Présentation du 0 suivant :



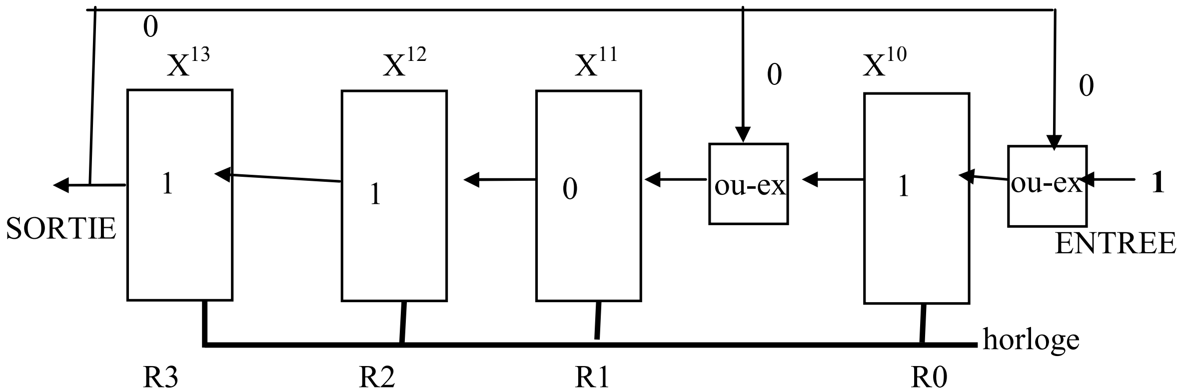
Lors de l'entrée du 0 suivant au top d'horloge:



Présentation du 1 suivant :



Lors de l'entrée du 1 suivant au top d'horloge :

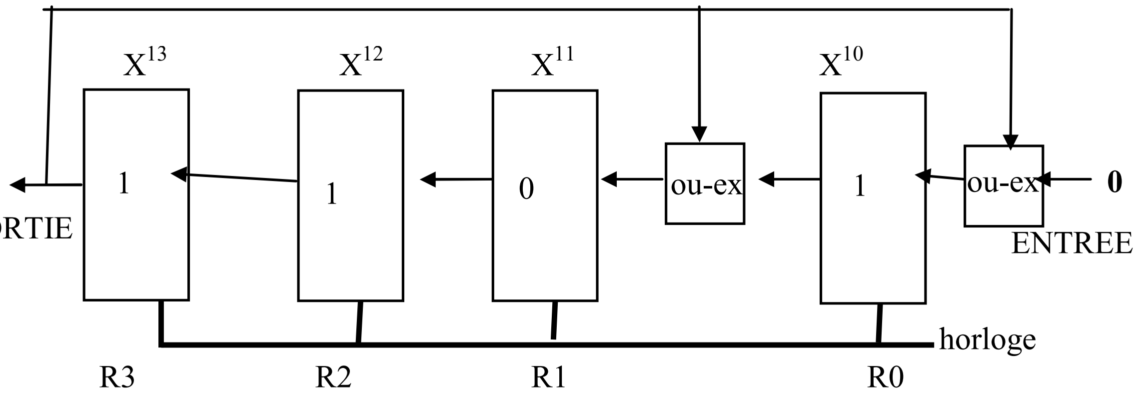


Le circuit est amorcé : on a ingnoré les 4 premiers 0 qui sont sortis, ils n'ont pas été poussés vers le réseau. On constate que dans les registres il y a le début du polynome à diviser. On a remis le numéro des registres, et les puissances de x qui correspondent à l'étape de division polynomiale. Ce qui est dans les registres touche les coefficients encadrés de $x^4*D(x)$:

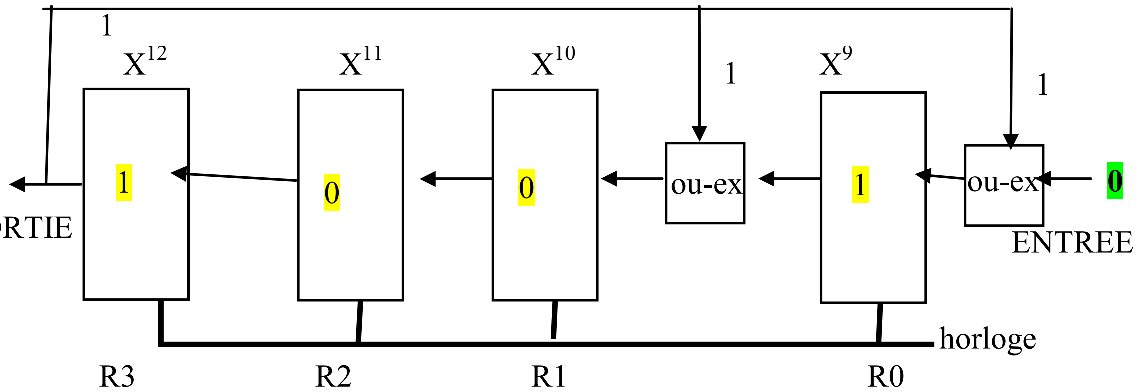
$\boxed{1*x^{13}+1*x^{12}+0*x^{11}+1*x^{10}}+0*x^9+1*x^8+1*x^7+0*x^6+1*x^5+1*x^4+0*x^3+0*x^2+0*x^1+0*x^0$

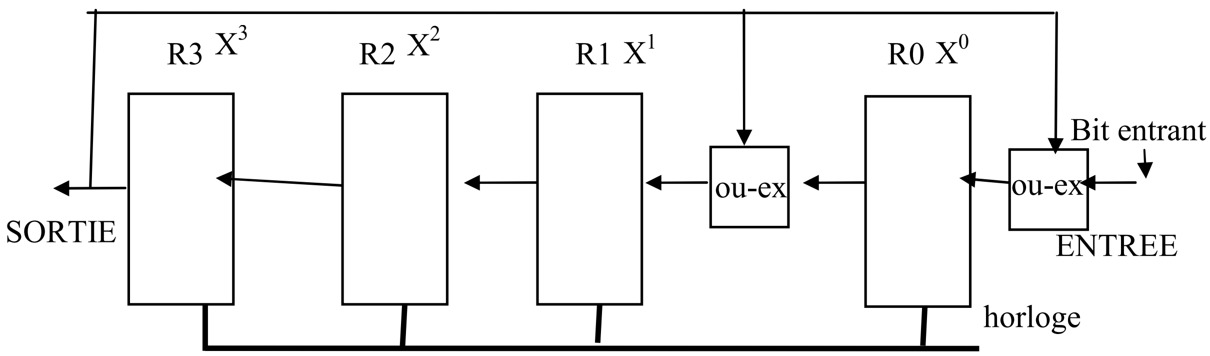
A chaque top d'horloge on va faire avancé le processus de division du polynome $x^4*D(x)$ pour aboutir au reste prévu. Dans la suite, on ne met plus les dessins des registres, mais le résultat qu'ils contiennent.

Présentation du 0 suivant :



Lors de l'entrée du 0 suivant qui fait sortir un **1,** au top d'horloge, le premier bit qui va effectivement sortir pour être transmis sur le réseau :





R3	R2	R1	R0	Bit entrant du message (coef de $x^4*D(x)$
1	1	0	1 +	0[2]+
$\boxed{1}\ x^{12}$	$\boxed{0}\ x^{11}$	$\boxed{0}\ x^{10}$	$\boxed{1}\ x^{9}$	1
$\boxed{0}\ x^{11}$	$\boxed{0}\ x^{10}$	$\boxed{0}\ x^{9}$	$\boxed{0}\ x^{8}$	1
$\boxed{0}\ x^{10}$	$\boxed{0}\ x^{9}$	$\boxed{0}\ x^{8}$	$\boxed{1}\ x^{7}$	0
$\boxed{0}\ x^{9}$	$\boxed{0}\ x^{8}$	$\boxed{1}\ x^{7}$	$\boxed{0}\ x^{6}$	1
$\boxed{0}\ x^{8}$	$\boxed{1}\ x^{7}$	$\boxed{0}\ x^{6}$	$\boxed{1}\ x^{5}$	1
$\boxed{1}\ x^{7}$	$\boxed{0}\ x^{6}$	$\boxed{1}\ x^{5}$	$\boxed{1}\ x^{4}$	
Les données sont toutes passées dans le circuit. Les 4 bits suivants sont poussés dans le circuit, des 0, servent à faire sortir le résultat de la division polynomiale hors des registres R3-R0 et donc le CRC associé au message. Ce sont les bits qui ont servi à "créer de l'espace pour générer le CRC" quand on à fait $x^4*D(x)$				
				0
$\boxed{0}\ x^{6}$	$\boxed{1}\ x^{5}$	$\boxed{0}\ x^{4}$	$\boxed{1}\ x^{3}$	0
$\boxed{1}\ x^{5}$	$\boxed{0}\ x^{4}$	$\boxed{1}\ x^{3}$	$\boxed{0}\ x^{1}$	0
$\boxed{0}\ x^{4}$	$\boxed{1}\ x^{3}$	$\boxed{1}\ x^{2}$	$\boxed{1}\ x^{1}$	0
$\boxed{1}\ x^{3}$	$\boxed{1}\ x^{2}$	$\boxed{1}\ x^{1}$	$\boxed{0}\ x^{0}$	

On retrouve au fur et à mesure les coefficients de la division polynomiale.

La division est reprise dans son ensemble avec des $\boxed{}$ pour figurer le contenu des registres à décalage au fur et à mesure de l'avancement du calcul pour obtenir le CRC.

2 En vert et Jaune, ce sont les valeurs dans la ligne qui est dans le dessin

$$\boxed{1}*x^{13}+\boxed{1}*x^{12}+\boxed{0}*x^{11}+\boxed{1}*x^{10}+0*x^{9}+1*x^{8}+1*x^{7}+0*x^{6}+1*x^{5}+1*x^{4}+0*x^{3}+0*x^{2}+0*x^{1}+0*x^{0} \;\Big|\; \underline{1*x^{4}+0*x^{3}+0*x^{2}+1*x^{1}+1*x^{0}}$$

$$-(1*x^{13}+0*x^{12}+0*x^{11}+1*x^{10}+1*x^{9}) \qquad \Big|\; 1*x^{9}+1*x^{8}+0*x^{7}+0*x^{6}+0*x^{5}$$

$$\boxed{1}*x^{12}+\boxed{0}*x^{11}+\boxed{0}*x^{10}+\boxed{1}*x^{9}+1*x^{8}+1*x^{7}+0*x^{6}+1*x^{5}+1*x^{4}+0*x^{3}+0*x^{2}+0*x^{1}+0*x^{0} \;\Big|\; +0*x^{4}+1*x^{3}+0*x^{2}+1*x^{1}+0*x^{0}$$

$$-(1*x^{12}+0*x^{11}+0*x^{10}+1*x^{9}+1*x^{8})$$

$$\boxed{0}*x^{11}+\boxed{0}*x^{10}+\boxed{0}*x^{9}+\boxed{0}*x^{8}+1*x^{7}+0*x^{6}+1*x^{5}+1*x^{4}+0*x^{3}+0*x^{2}+0*x^{1}+0*x^{0}$$

$$-(0*x^{11}+0*x^{10}+0*x^{9}+0*x^{8}+0*x^{7})$$

$$\boxed{0}*x^{10}+\boxed{0}*x^{9}+\boxed{0}*x^{8}+\boxed{1}*x^{7}+0*x^{6}+1*x^{5}+1*x^{4}+0*x^{3}+0*x^{2}+0*x^{1}+0*x^{0}$$

$$-(0*x^{10}+0*x^{9}+0*x^{8}+0*x^{7}+0*x^{6})$$

$$\boxed{0}*x^{9}+\boxed{0}*x^{8}+\boxed{1}*x^{7}+\boxed{0}*x^{6}+1*x^{5}+1*x^{4}+0*x^{3}+0*x^{2}+0*x^{1}+0*x^{0}$$

$$-(0*x^{9}+0*x^{8}+0*x^{7}+0*x^{6}+0*x^{5})$$

$$\boxed{0}*x^{8}+\boxed{1}*x^{7}+\boxed{0}*x^{6}+\boxed{1}*x^{5}+1*x^{4}+0*x^{3}+0*x^{2}+0*x^{1}+0*x^{0}$$

$$-(0*x^{8}+0*x^{7}+0*x^{6}+0*x^{5}+0*x^{4})$$

$$\boxed{1}*x^{7}+\boxed{0}*x^{6}+\boxed{1}*x^{5}+\boxed{1}*x^{4}+0*x^{3}+0*x^{2}+0*x^{1}+0*x^{0}$$

$$-(1*x^{7}+0*x^{6}+0*x^{5}+1*x^{4}+1*x^{3})$$

$$\boxed{0}*x^{6}+\boxed{1}*x^{5}+\boxed{0}*x^{4}+\boxed{1}*x^{3}+0*x^{2}+0*x^{1}+0*x^{0}$$

$$-(0*x^{6}+0*x^{5}+0*x^{4}+0*x^{3}+0*x^{2})$$

$$\boxed{1}*x^{5}+\boxed{0}*x^{4}+\boxed{1}*x^{3}+\boxed{0}*x^{2}+0*x^{1}+0*x^{0}$$

$$-(1*x^{5}+0*x^{4}+0*x^{3}+1*x^{2}+1*x^{1})$$

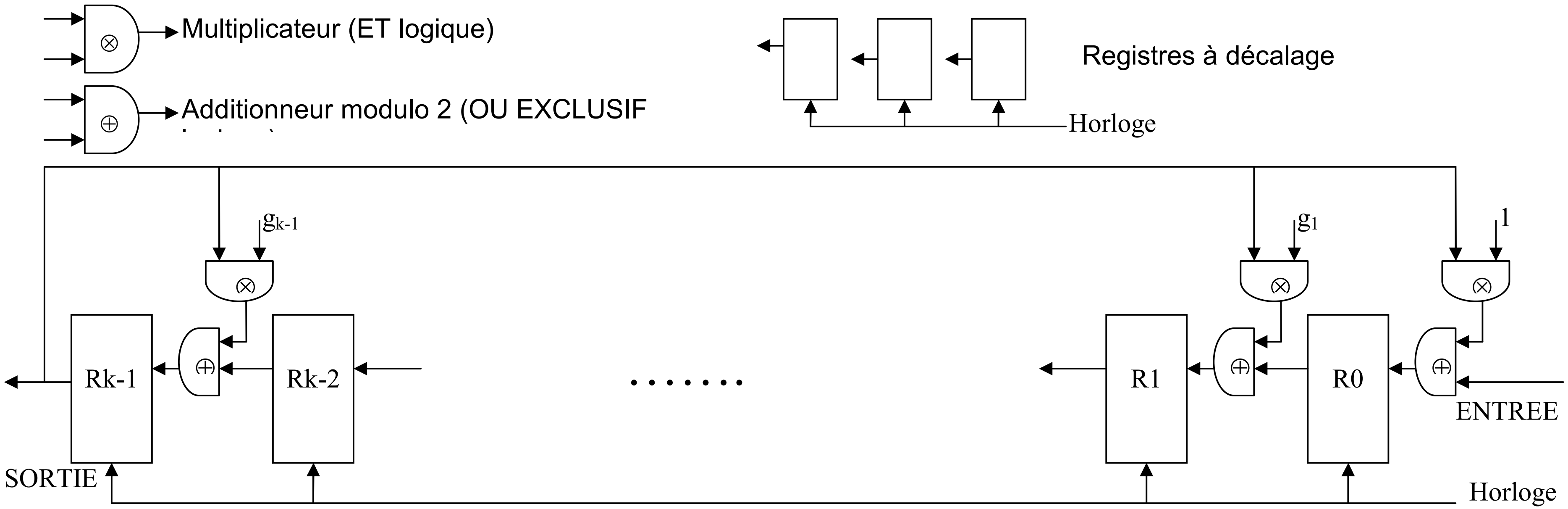
$$\boxed{0}*x^{4}+\boxed{1}*x^{3}+\boxed{1}*x^{2}+\boxed{1}*x^{1}+0*x^{0}$$

$$-(0*x^{4}+0*x^{3}+0*x^{2}+0*x^{1}+0*x^{0})$$

C'est le reste dans le circuit électronique $R(x) \rightarrow \boxed{1}*x^{3}+\boxed{1}*x^{2}+\boxed{1}*x^{1}+\boxed{0}*x^{0}$

et en particulier dans les registres dans l'ordre suivant : R3 - R2 - R1 - R0

Généralisation du circuit :



Le polynome générateur est : $G(x) = 1* x^k + g_{k-1} x^{k-1} + g_{k-2} * x^{k-2} +…. + g_1 * x^1 + 1* x^0$

Le circuit au circuit donné dans 2 livres en fin de poly d'exercices.

On soumet dans le circuit CRC toujours le bit le plus significatif en premier (bit de poids fort) qui est aussi associé à la puissance de x la plus élevée.

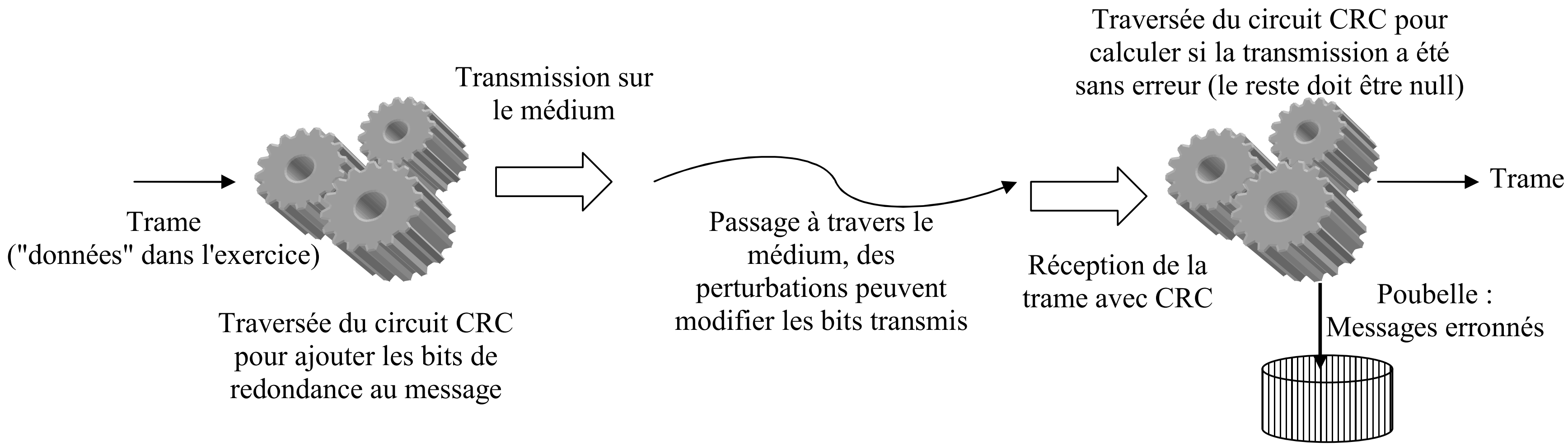
En entrée :

- à l'émission on soumet $x^k * D(x)$, initialement le registre à décalage ne contient que des 0, on obtient le reste R(x) de la division de $x^k * D(x)$ par G(x) :

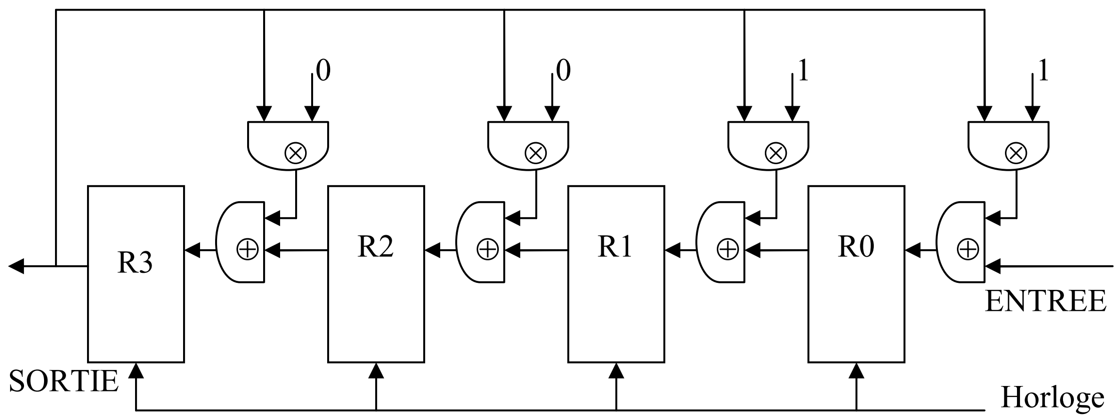
  $R(x) = r_{k-1}\ x^{k-1} + r_{k-2} * x^{k-2} + \ldots + r_1 * x^1 + r_0 * x^0$ qui forme les k bits de redondance dans le message émis sur le médium

- à la réception on fait entrer le message reçu, initialement le registre à décalage ne contient que des 0, soit A'(x) le polynome associé au message reçu. Si la division de A'(x) par G(x) donne :
  - un reste null, alors il n'y a pas eu d'erreur de transmission,
  - un reste non null, alors il y a eu erreur de transmission, et la trame reçue est écartée, donc non traitée par la liaison

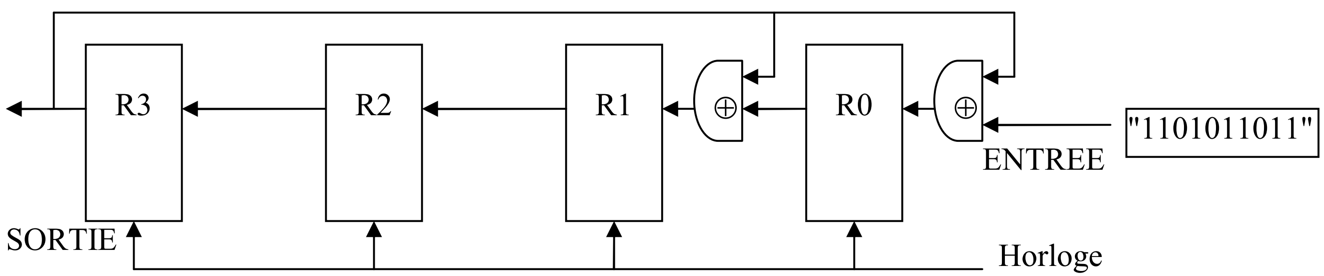
L'ensemble du fonctionnement de l'Emission à la Réception de la trame, de la soumission à la livraison de la trame :



Circuit pour l'exercice : $G(x) = 1*x^4+0*x^3+0*x^2+1*x^1+1*x^0$ d'où $g_3 = g_2 = 0$ et $g_1 = 1$



Par simplification on obtient :



Pour plus d'informations sur les circuits CRC, vous pouvez vous reporter aux deux livres suivants :

- Jacques Clavier, Marcel Niquil, Gérard Coffinet, Francis Behr. "Théorie et technique de la transmission des données". 6 février 1997.
- César Macchi, Jean-François Guilbert et 13 co-auteurs. Téléinformatique: Transport et traitement de l'information dans les réseaux et systèmes téléinformatiques et télématiques Broché – 6 novembre 1990. Dunod.

Le premier est certainement le plus intéressant des deux pour un point de vue théorique. Le deuxième j'y ai puisé le circuit général.

Très intéressants d'ailleurs pour tout ce qui concerne les bases de la théorie du signal : théorie de l'information, entropie, codes polynomiaux, décomposition en séries de Fourrier… Mais probablement qu'ils sont épuisés ou d'occasion mais devraient être à la bibliothèque du Cnam.

Exercice 2 : Prise en compte de la détection d'erreurs dans le calcul de dimensionnement de flux

Mesures de performance :
Débit : Le débit d'un réseau mesure la quantité d'information que le réseau peut transmettre par unité de temps :

$$\text{Débit} = \frac{\text{quantité d'information}}{\text{délai nécessaire pour transmettre cette quantité d'information}}$$

L'unité est le bit par seconde, noté b/s, attention à bien effectuer les conversions qui s'imposent pour trouver le bon résultat dans l'unité b/s.

Débits nominal : Le débit nominal c'est le débit théorique maximal sur support de transmission. C'est souvent une donnée connue du problème à résoudre.

Débit utile : Le débit utile correspond à la quantité d'informations utiles (souvent les données applicatives débarassées de toute information protocolaire d'accompagnement) effectivement transmises par unité de temps. C'est souvent un paramètre à calculer ou à déduire d'autres informations fournies dans le problème à résoudre.

Taux d'utilisation : C'est le rapport débit utile sur débit nominal :

taux d'utilisation = débit utile /débit nominal

En général le taux d'utilisation est inférieur à 100% à cause des données protocolaires, des pertes/erreurs de transmission qui nécessitent une retransmission, la fréquence d'envoi des messages.
Une fréquence d'envoi faible laisse plus de temps entre l'envoi de deux messages qu'une fréquence élevée.

Délai de transmission : C'est le temps mis pour transmettre une quantité d'information correspondant à un message :

délai_transmission = quantité information / débit

Attention, les unités doivent être homogènes : octet, bit… millisecondes, secondes…

Délai de propagation : C'est le temps mis par le signal pour traverser la distance entre la source et la destination sur le support de transmission considéré (fil métallique, fibre optique, air…). Si des équipements physiques/matériels sont traversés, ils peuvent introduire des retards. C'est par exemple le cas dans le calcul du délai de propagation sur Ethernet 10Mb/s où on comptabilise la traversée de tous les circuits électroniques.

Délai_propagation = distance parcourue / vitesse du signal

Délai d'acheminement d'un message : C'est la somme du délai pour transmettre un message plus le délai de propagation de la source au destinataire considéré.

délai_acheminement = délai_transmission + délai_propagation

Pour un réseau local, en général pour la transmission d'une trame on ne comptabilise pas le délai de propagation alors qu'on va complètement le prendre en compte dans une transmission satellite parce qu'il correspond à la part la plus grande du délai d'acheminement.

Sur une liaison hertzienne urbaine à 250000 bit/s, des messages de 64 bits sont envoyés. La fréquence d'émission est de 2500 messages/seconde. On peut replacer cette question dans un cadre applicatif de type réseau de capteurs destiné à une application de télémesure d’instruments (pollution, relevé de compteurs…) distribués sur un site industriel (Industry 4.0), une ville (Smart City)… On peut imaginer qu’on déploie 2500 capteurs qui envoient un message par seconde, le message contient une mesure qui ne prend pas tellement d’octets.

Les valeurs numériques proposées correspondent à une technologie réseau de capteurs actuelle, comme IEEE 802.15.4, qui offre un débit nomminal de 250Kb/s. Elles sont aussi choisies pour simplifier les calculs.

On aurait pu choisir un autre exemple dans le même domaine comme la technologie UNB (Ultra Narrow Band) qui sert à de la télé-relève de compteurs par exemple. C'est utilisé dans le domaine de l'Internet des Objets, ou encore M2M (Machine to Machine) et c'est promu en France en particulier par l'entreprise SIGFOX. C'est un réseau très bas débit à 100bits par seconde, mais à très faible consommation d'énergie. Son concurrent s'appelle LoRa. Donc la taille des messages est faible, et les émissions peu fréquentes. La portée est semble-t-il plus longue que celle d'un réseau de capteurs IEEE 802.15.4. Je ne peux en dire plus car je n'ai pas d'expérience pratique de ce type de réseau/technologie.

La question est formulée dans un contexte d’étude de dimensionnement pour de la maîtrise d’ouvrage. Les calculs que vous avez à faire serviront à valider une technologie qui permet de supporter 2500 capteurs par réseau. Cette validation consolidant un devis qui sera envoyé à votre client.

Question 1

Calculer le taux d'utilisation de la liaison.

Correction :

La ligne supporte 2500 messages/s de 64 bits soit une charge utile de 2500*64b/s soit 160 000 b/s, ou 160Kb/s

Le débit nominal de la ligne est 250000b/s

Le taux d'utilisation est : charge utile/débit nominal soit 2500*64/250000

Soit 64/100, soit 0,64.

Le taux d'utilisation est 64%.

On vous donne maintenant plus de précision sur le déploiement de la technologie. La liaison étant de mauvaise qualité, le taux d'erreur par bit (noté p) est compris entre 0,01 et 0,001. p représente la probabilité qu'un bit soit mal reçu.

Question 2

Calculer en fonction de p, la probabilité qu'un message soit reçu en erreur. On suppose que les altérations des bits sont indépendantes entre elles.

Correction :

Proba(message mal reçu[3]) = 1 – Proba (message bien reçu)

Proba(message bien reçu) =

Proba(1er bit reçu correct et 2ème bit reçu correct et …et 64ème bit reçu correct)

[3] mal reçu doit se comprendre comme message contenant une erreur de transmission

Les erreurs sont indiquées comme indépendantes les unes des autres[4], donc les "et" se transforment en "produit" quand on passe aux probabilités des différents événements :

Proba(message bien reçu) =

proba(1^er^bit reçu correct)*proba(2^ème^bit reçu correct)*…*proba(64^ème^bit reçu correct)

Proba qu'un bit soit bien reçu = 1 – proba qu'un bit soit érronné = 1 –p d'où :

Proba(message bien reçu) = $(1\text{-}p)^{64}$

Proba qu'un message soit mal reçu = $1 - (1\text{-}p)^{64}$

- Si p = 0,01, 10^{-2} ou encore 1/100, et s'interprète comme une erreur sur 100 bits

Proba(message bien reçu) = 0,525

Proba(message soit reçu erroné) = 0,475

- Si p = 0,001, 10^{-3} ou encore 1/1000, s'interprète comme une erreur sur 1000 bits

Proba(message bien reçu) = 0,938

Proba(message soit reçu erroné) = 0,062

Remarque : Dans la réalité, les erreurs ne sont pas indépendantes, elles modifient une suite de bits. Une perturbation électro-magnétique perturbe un bit et ses voisins. Ce n'est pas une perturbation ponctuelle.

Question 3

Par quel mécanisme le récepteur peut-il détecter qu'un message est reçu avec une erreur de transmission ?

Correction :

Le mécanisme pour détecter qu'un message est reçu avec une erreur de transmission correspond à la mise en place d'un code détecteur d'erreur type CRC.

On suppose maintenant que l'émetteur sait quand un message est mal reçu, dans ce cas il le retransmet. Rien n'est dit sur comment il le sait, cela sera décrit dans d'autres cours.

Question 4

Calculer en fonction de p le nombre moyen de transmissions. Est-il possible de respecter (en négligeant le temps écoulé entre 2 retransmissions) la fréquence d'émission de 2500 messages/seconde ?

Correction :

[4] Dans la réalité, les erreurs sur les bits ne sont généralement pas indépendantes, cf remarque.

En fait la question demande de calculer une moyenne mathématique, appelée espérance mathématique en probabilité si je ne me trompe pas. On la note N.

N = Σ n*Proba(envoyer n fois un message)

Pour qu'il n'y ait que n envois, il faut n-1 échecs suivi d'un succès, celui-ci correspond nécessairement au dernier envoi.

Proba (envoyer n fois un message) = Proba (échouer n-1 fois et réussir le dernier envoi)

Succès et réussite des tentatives de transmission sont indépendants. D'où

Proba (envoyer n fois un message) =

Proba (échouer n-1 fois) * Proba(réussir le dernier envoi)

Proba (échouer 1 fois)$^{(n-1)}$ * Proba(réussir le dernier envoi)

On appelle q = Proba(message bien reçu) = $(1-p)^{64}$, on a alors

Proba (envoyer n fois un message) = $(1-q)^{(n-1)}$ * q

On en déduit que N = $\Sigma\, n*(1-q)^{(n-1)} * q$

On peut sortir q de la sommation soit $N = q * \Sigma\, n* (1-q)^{(n-1)}$

$N = q * 1 / (1-(1-q))^2 = q/q^2$ donc $N = 1/q$ (Nombre moyen de transmission pour une donnée)

- Si p = 0,01,

  q = 0,525, N = 1/0,525, N = 1,9 cela indique qu'on est très proche de 2 transmissions en moyenne par message

- Si p = 0,001,

  q = 0,938, N = 1/0,938, N = 1,06 ce résultat indique qu'on est très proche d'une tranmission en moyenne par message

Avec un peu de bon sens, on voit déjà se dessiner la conclusion. Quand p = 0.01 (taux d'erreur de 10^{-2}) un message est quasiment envoyé 2 fois avant d'arriver à l'application. On va donc utiliser un débit applicatif d'environ 2*160Kb/s (débit trouvé à la question 1), ou 320Kb/s… qui ne passera pas sur un lien à 250Kb/s.

Quand p = 0.001 (taux d'erreur de 10^{-3}) on transmet le message environ une fois. Donc l'application va pouvoir faire passer son trafic sur le réseau puisque les erreurs ne l'impactent pas vraiment.

Rappel : Pour calculer l'espérance mathématique ci-dessus, on applique la formule :

$$X=\Sigma\ n^*\ x^{(n-1)} = 1/(1-x)^2 \text{ si } x \text{ est } < 1 \text{ (c'est un résultat connu au lycée)}$$

Elle est transposée à N en posant x = 1-q.

Cette formule se démontre en utilisant les séries. Soit S_n une série telle que :

$S_0 = 1$

$S_1 = 1 + x$

…

$S_n = 1 + x + x^2 + … + x^n$

On pourrait aussi noter le terme S_n différemment en le faisant dépendre de x, par exemple $S_n(x)$. Mais ça aloudirait l'écriture et donc la démonstration et sa compréhension sans apporter plus de sens.

Calculons S_n pour obtenir sa valeur quand n tend vers l'infini. On utilise une autre façon de l'écrire, soit

$x^*S_n = x^* (1 + x + x^2 + … + x^n)$

si on fait la différente $S_n – x^*S_n$

$S_n = 1 + x + x^2 + … + x^n$

$x^*S_n = x + x^2 + … + x^n + x^{n+1}$

on obtient une équation très simple pour trouver S_n :

$S_n – x^*S_n = 1- x^{n+1}$

$(1-x) * S_n = 1- x^{n+1}$

d'où $S_n = 1- x^{n+1} / (1-x)$

Mais ce qu'on cherche c'est $\Sigma\ x^n$, la somme porte sur n valant de 0 à $+\infty$

Pour avoir $\Sigma\ x^n$ on fait un calcul à la limite en faisant tendre n vers $+\infty$, c'est classique :

$\lim_{n\to +\infty} x^{n+1} = 0$ car x<1 (c'est une probabilité)

donc $\lim_{n\to +\infty} (1-x^{n+1}) = 1$

On en déduit que $\lim_{n\to +\infty} S_n = \lim_{n\to +\infty} [(1-x^{n+1})/(1-x)] = 1/(1-x)$

Par conséquent : $\Sigma\ x^n = \lim_{n\to +\infty} S_n = 1/(1-x)$ c'est presque ce dont on a besoin…

Il suffit de calculer la dérivée de cette série pour avoir la formule que nous recherchons. Rappelons aussi que la dérivée d'une somme est égale à la somme des dérivées.

$d(\Sigma\ x^n)/dx = \Sigma\ dx^n/dx = \Sigma\ n\ x^{n-1} = d(1/(1-x))/dx = 1/(1-x)^2$ si x<1

Du coup, on a bien le résultat qui a été rappelé plus haut :

$$\Sigma\ n\ x^{n-1} = 1/(1-x)^2 \text{ si } x<1$$

Question 5

Quand vous comparez les résultats de la question 1 et de la question 4 que constatez vous ? Quelle est votre conclusion si vous aviez proposé un devis à l'issue de la question 1.

Correction :

- Le contexte de la question 1 est plutôt naif car il ne considère par le taux d'erreur sur le support. Donc le débit applicatif est supporté sans problème.
- Dans la question 4 on adopte un point de vue plus réaliste, on considère la possibilité de subir des erreurs.

  - Quand p=0,01, en fait en moyenne 1,9 transmissions par message à acheminer. En conséquence, le support est chargé par 1,9*64*2500/250000 =121,1 % donc le débit nomminal de la voie de transmission est insuffisant.
  - Quand p=0,001, en fait en moyenne 1,06 transmissions par message à acheminer. En conséquence, le support est chargé par 1,06*64*2500/250000 = 67,84 % donc le débit nomminal de la ligne est suffisant.