

Introduction à l'informatique et la programmation

IF1 2010-2011

Matthieu Picantin



LIAFA UMR 7089
CNRS & Université Paris 7 Denis Diderot

28/10/2010

Connecteurs, opérateurs, fonctions

Une fonction booléenne est une fonction $f : \mathbb{B}^k \rightarrow \mathbb{B}$ où k est l'arité de f

Les quatre opérateurs booléens unaires

	$f(0)$	$f(1)$	
	0	0	contradiction
	0	1	affirmation
	1	0	négation NOT
	1	1	tautologie

Il existe exactement 2^{2^k} fonctions booléennes d'arité k

Algèbre de Boole

- ◆ ensemble $\mathbb{B} = \{0, 1\}$ muni d'un opérateur unaire \neg (négation) et de deux opérateurs binaires \vee (disjonction) et \wedge (conjonction)
- ◆ vérifiant, pour toutes variables booléennes x, y, z de \mathbb{B} , les axiomes suivants :

associativité	$x \vee (y \vee z) = (x \vee y) \vee z$	$x \wedge (y \wedge z) = (x \wedge y) \wedge z$
commutativité	$x \vee y = y \vee x$	$x \wedge y = y \wedge x$
absorption	$x \vee (x \wedge y) = x$	$x \wedge (x \vee y) = x$
distributivité	$x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$	$x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$
complémentation	$x \vee \neg x = 1$	$x \wedge \neg x = 0$

- ◆ ayant, pour toutes variables booléennes x, y de \mathbb{B} , les propriétés suivantes :

idempotence	$x \vee x = x$	$x \wedge x = x$
neutres	$x \vee 0 = x$	$x \wedge 1 = x$
involution	$\neg \neg x = x$	
De Morgan	$\neg(x \vee y) = \neg x \wedge \neg y$	$\neg(x \wedge y) = \neg x \vee \neg y$

- ◆ interprétation des valeurs :

$0 \rightsquigarrow$ false et $1 \rightsquigarrow$ true

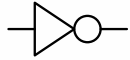
Connecteurs, opérateurs, fonctions

Une fonction booléenne est une fonction $f : \mathbb{B}^k \rightarrow \mathbb{B}$ où k est l'arité de f

	$f(0, 0)$	$f(0, 1)$	$f(1, 0)$	$f(1, 1)$	
	0	0	0	0	contradiction
	0	0	0	1	conjonction AND
	0	0	1	0	négation de l'implication $\not\supset$
	0	0	1	1	affirmation de x
	0	1	0	0	négation de l'implication inverse $\not\supseteq$
	0	1	0	1	affirmation de y
	0	1	1	0	ou exclusif XOR
	0	1	1	1	disjonction OR
	1	0	0	0	négation connexe de Peirce NOR
	1	0	0	1	équivalence NXOR
	1	0	1	0	négation de y
	1	0	1	1	implication inverse \supseteq
	1	1	0	0	négation de x
	1	1	0	1	implication \supset
	1	1	1	0	incompatibilité de Shaffer NAND
	1	1	1	1	tautologie

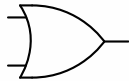
Négation

x	$\neg x$
0	1
1	0



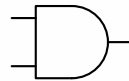
Disjonction

x	y	$x \vee y$
0	0	0
0	1	1
1	0	1
1	1	1



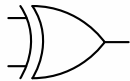
Conjonction

x	y	$x \wedge y$
0	0	0
0	1	0
1	0	0
1	1	1



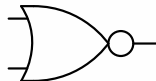
Ou exclusif

x	y	$x \oplus y$
0	0	0
0	1	1
1	0	1
1	1	0



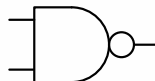
Connecteur de Peirce

x	y	$x \downarrow y$
0	0	1
0	1	0
1	0	0
1	1	0



Incompatibilité de Sheffer

x	y	$x \uparrow y$
0	0	1
0	1	1
1	0	1
1	1	0



Choix des connecteurs

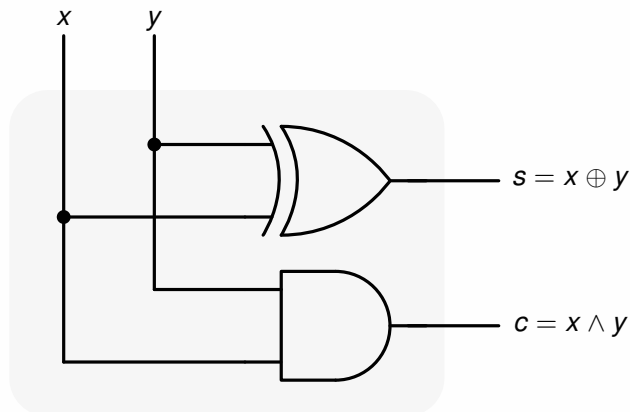
- ♦ tous les connecteurs peuvent se déduire d'un nombre limité des autres
- ♦ différents choix sont possibles pour ces *connecteurs primitifs* et la simplicité des expressions dépend beaucoup de ce choix
 - ▶ dans les langages de programmation, on dispose généralement de
 - ★ la *négation* \neg (en Java : !),
 - ★ la *conjonction* \wedge (en Java : &&),
 - ★ la *disjonction* \vee (en Java : ||),
 - ★ et parfois le *ou exclusif* \oplus (comme en C ou Java : ^)
 - ▶ chacun des connecteurs NAND et NOR est universel

Circuit combinatoire

- ♦ un *circuit combinatoire* est **une** mise en oeuvre matérielle d'une fonction combinatoire (*une parmi une infinité de possibilités*)
- ♦ un système logique est dit *combinatoire* si l'état de sa sortie ne dépend que de l'état de son entrée (pas de l'histoire du système)

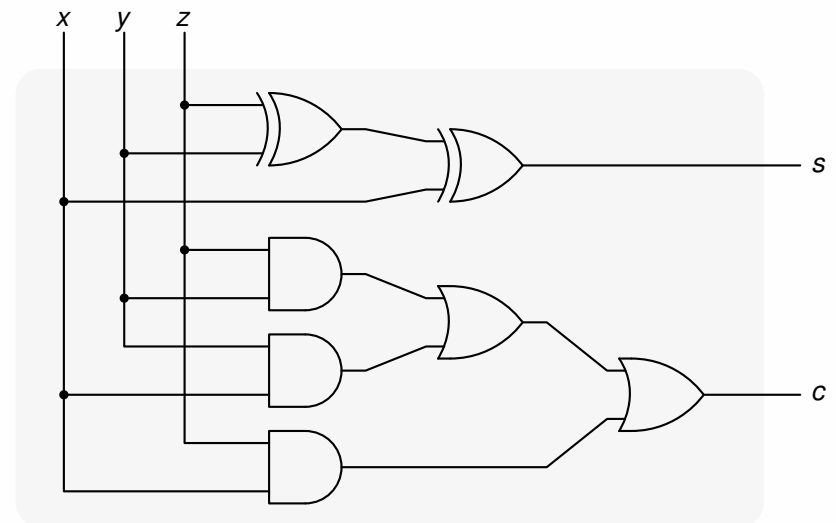
<http://www.neuroproductions.be/logic-lab/>

Demi-additionneur



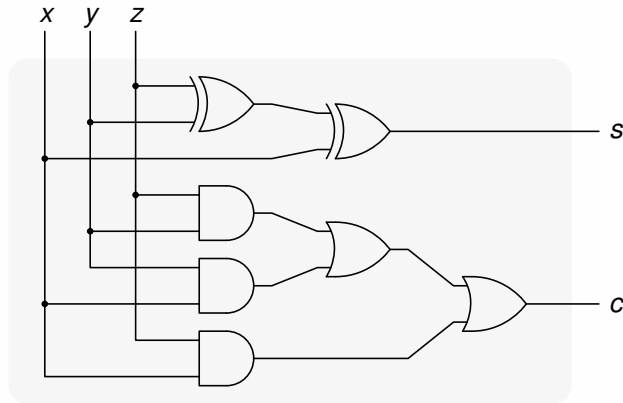
x	y	$x \wedge y$	$x \oplus y$
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Additionneur complet



Additionneur complet

x	y	z	c	s
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

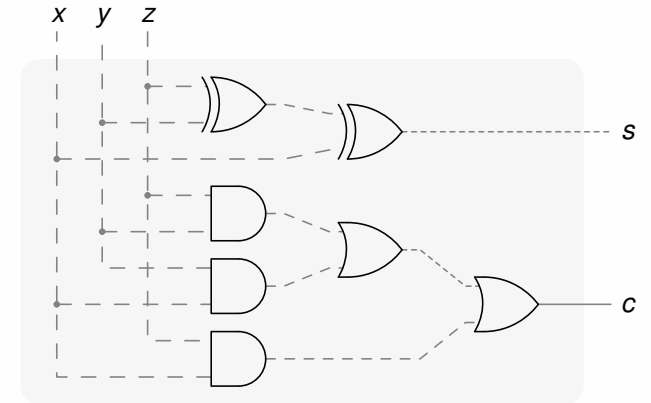


$$c = \text{majorité}(x, y, z) = (x \wedge y) \vee (y \wedge z) \vee (x \wedge z)$$

$$s = \text{imparité}(x, y, z) = x \oplus y \oplus z$$

Additionneur complet

x	y	z	c	s
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



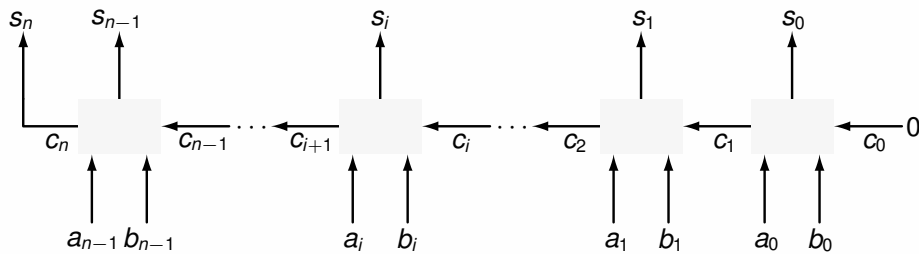
Profondeur d'un circuit

- ◆ correspond à son temps d'exécution dans le pire des cas
- ◆ est définie comme la profondeur maximale des câbles de sortie
 - ▶ un câble d'entrée est de profondeur 0
 - ▶ si une porte possède des entrées de profondeur d_1, \dots, d_n ses sorties seront de profondeur $\max(d_1, \dots, d_n) + 1$

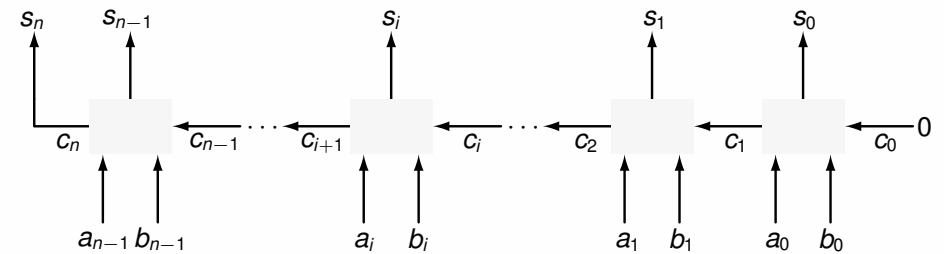
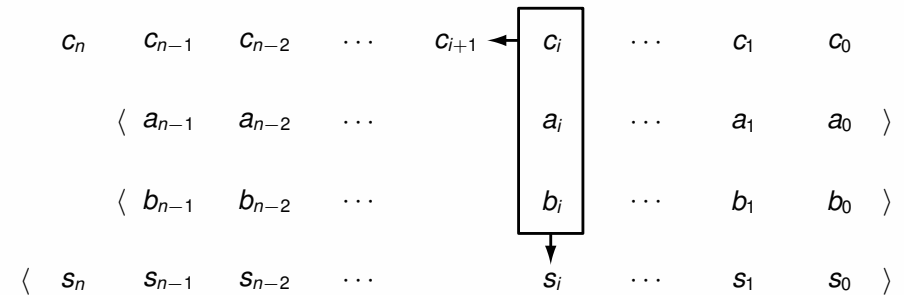
Additionneur en cascade

$$\sum_{i=1}^n a_i 2^i = a = \langle a_{n-1} \ a_{n-2} \ \dots \ a_i \ \dots \ a_1 \ a_0 \rangle$$

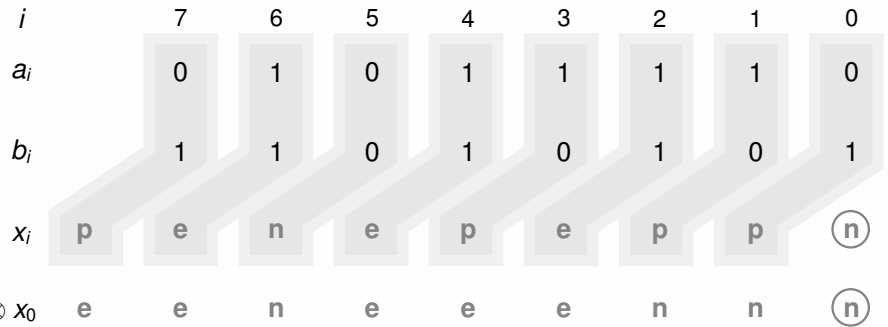
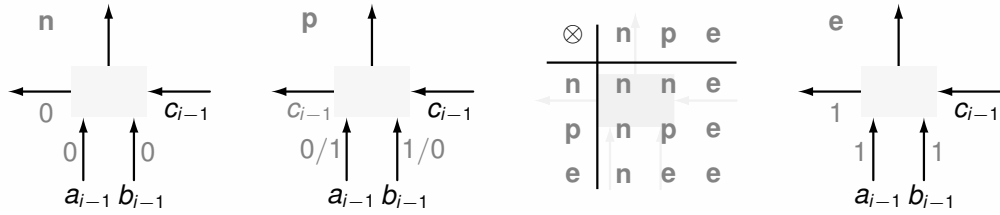
$$\sum_{i=1}^n b_i 2^i = b = \langle b_{n-1} \ b_{n-2} \ \dots \ b_i \ \dots \ b_1 \ b_0 \rangle$$



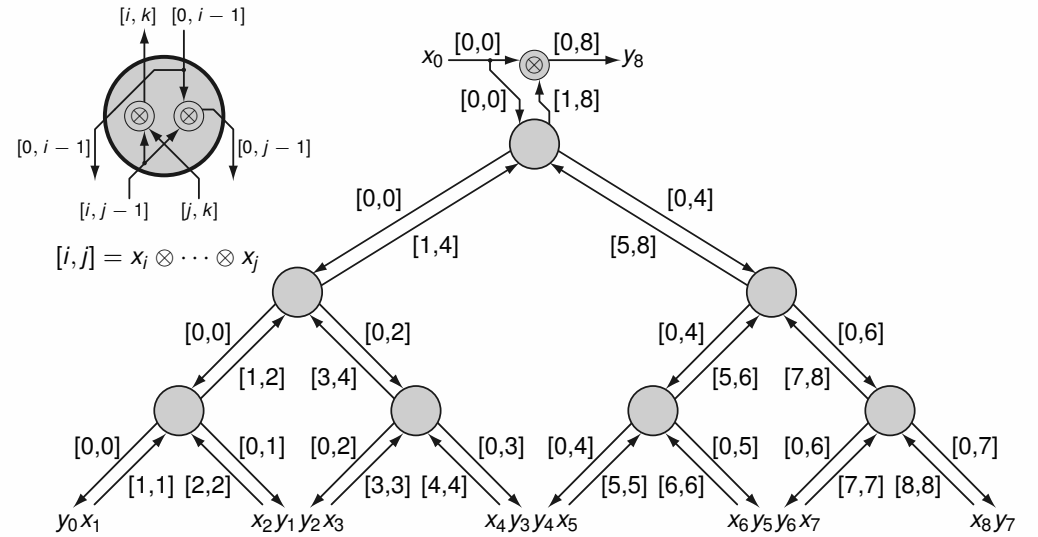
Additionneur en cascade



Additionneur avec prévision de retenue



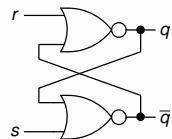
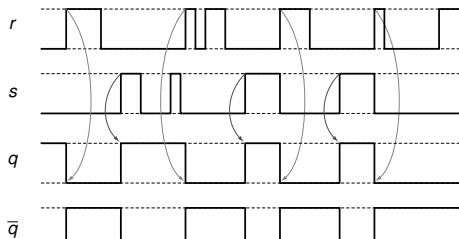
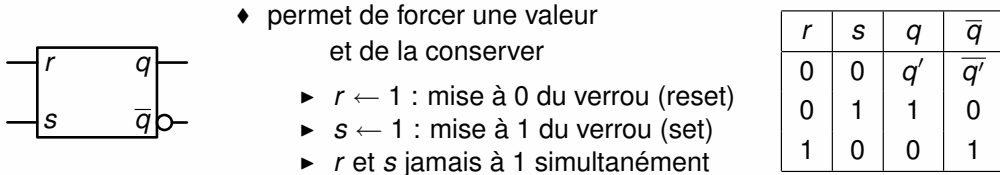
Prévision de retenue par circuit préfixe parallèle



Circuits séquentiels

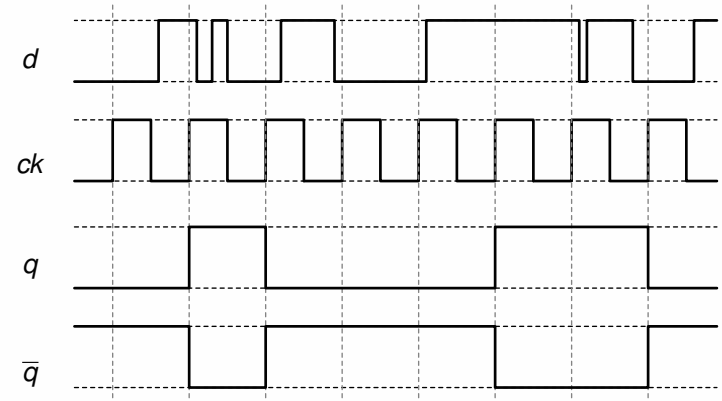
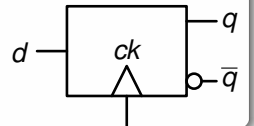
- ◆ les sorties sont fonctions des entrées mais aussi de l'état interne du système
 - ▶ la sortie maintient son état même après disparition du signal de commande
- ◆ deux types se distinguent
 - ▶ les verrous : fonctionnement asynchrone
 - ▶ les bascules : fonctionnement synchrone (basé sur une entrée d'horloge)

Verrou RS



Basculer D

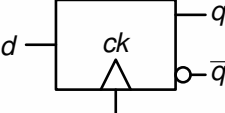
- ◆ échantillonne une entrée de donnée sur un événement de commande
 - ▶ d entrée de donnée
 - ▶ q sortie de donnée
 - ▶ ck entrée d'horloge
- ◆ si active sur **front montant** : $q = \begin{cases} d & \text{si } ck \text{ subit } 0 \rightarrow 1 \\ q' & \text{sinon} \end{cases}$

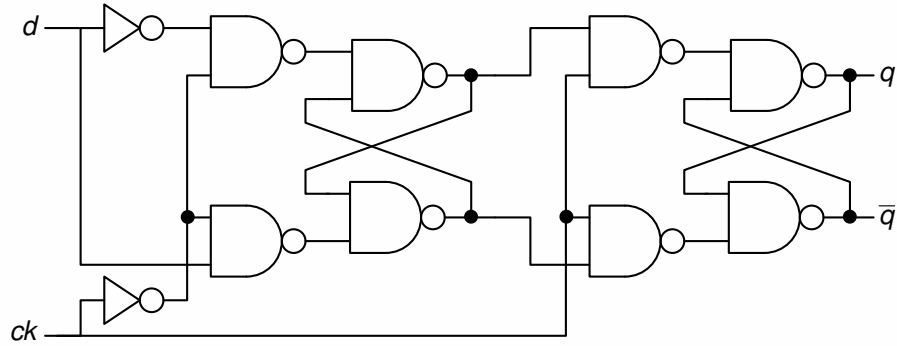


Bascule D

- ◆ échantillonne une entrée de donnée sur un évènement de commande
 - ▶ d entrée de donnée
 - ▶ q sortie de donnée
 - ▶ ck entrée d'horloge

◆ si active sur **front montant** : $q = \begin{cases} d & \text{si } ck \text{ subit } 0 \rightarrow 1 \\ \bar{d} & \text{sinon} \end{cases}$





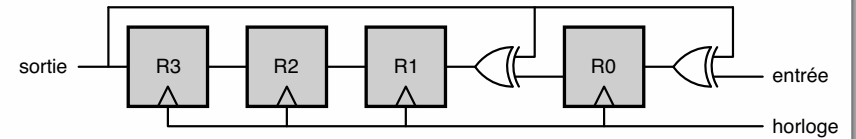
Codes polynômiaux détecteurs d'erreurs

- ◆ correspondance entre rang des bits et degré des monômes
 - ▶ par exemple, le mot **10011** code le polynôme $x^4 + x + 1$
- ◆ arithmétique polynomiale (soustraction modulo 2 et division euclidienne)

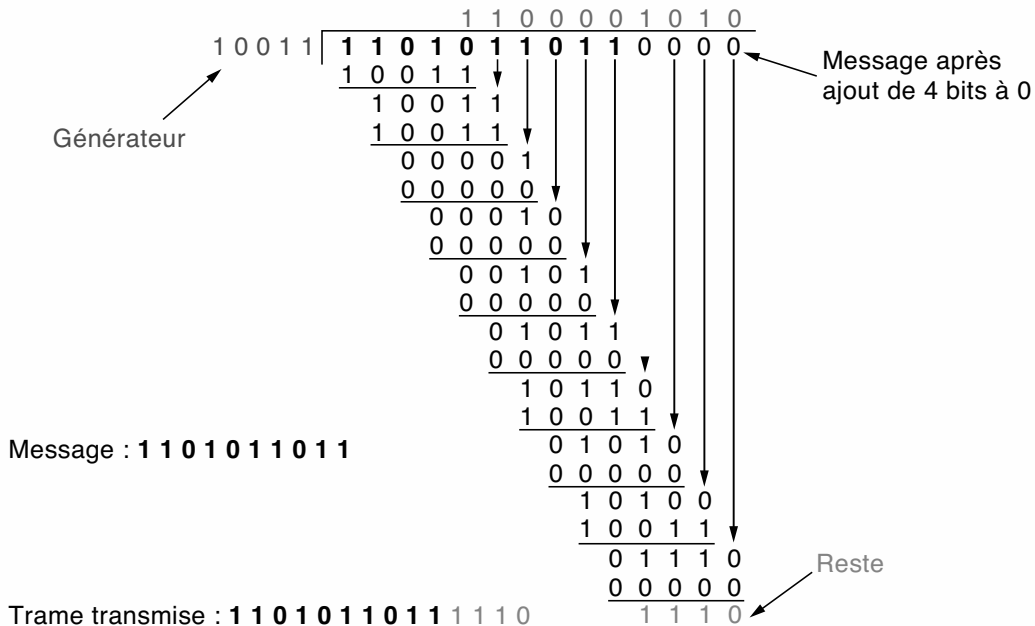
Utilisation d'un polynôme générateur $G(x)$

- ◆ $G(x)$ de degré r et message $M(x)$
- ◆ ajout de r bits à 0 après le bit de poids faible de $M(x)$
- ◆ division de $x^r M(x)$ par $G(x)$: reste $R(x)$
- ◆ envoi de $T(x) = x^r M(x) - R(x)$
- ◆ $T(x)$ est divisible par $G(x)$ (à vérifier par le récepteur !)

Circuit associé au polynôme $x^4 + x + 1$



Calcul à la main du reste dans la division par $x^4 + x + 1$



Trace du calcul par le circuit du reste dans la division par $x^4 + x + 1$

