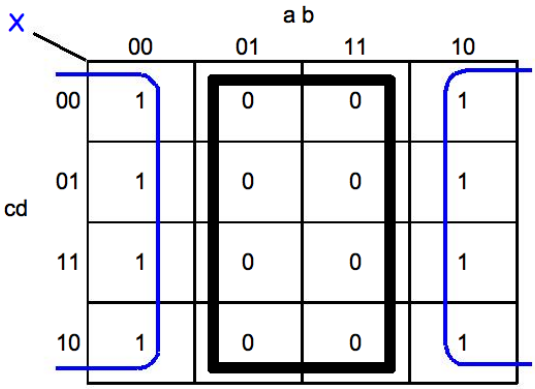
Table de Karnaugh

Exemple : On reprend la table de Karnaugh de

$$X = \bar{a}\bar{b}\bar{c}\bar{d} + \bar{a}\bar{b}\bar{c}d + \bar{a}\bar{b}cd + \bar{a}\bar{b}c\bar{d} + a\bar{b}\bar{c}\bar{d} + a\bar{b}\bar{c}d + a\bar{b}cd + a\bar{b}c\bar{d}$$



$$X = \bar{b}$$

1. Soit on regroupe les “0” et on fait le **produit des maxtermes** de variables qui ne changent pas et qui rendent faux X.
2. Soit on regroupe les “1”et on fait la somme des mintermes de variables qui ne changent pas et qui rendent vrai X.

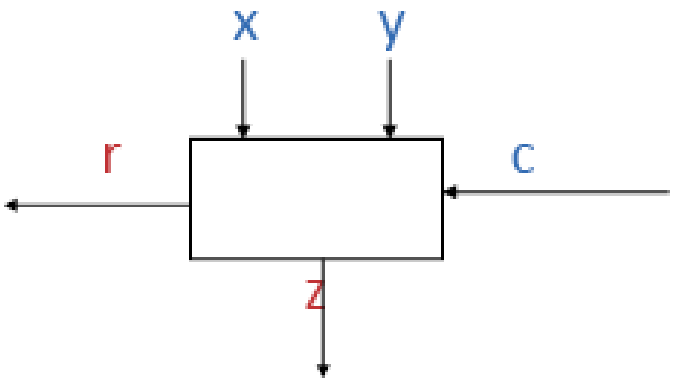
Circuits logiques et booléens

- Les *circuits logiques*, composants de base des ordinateurs, sont conçus à partir de *circuits élémentaires* correspondants aux opérations booléennes “.”, “+” et “¯”.
- Un circuit logique peut être vu comme une boîte noire ayant
  $n \geq 1$ ports d’entrée $e_1, e_2, \ldots, e_n$
  $m \geq 1$ ports de sortie $s_1, s_2, \ldots, s_m$
- Il traite des informations codées sur n bits et donne des informations codées sur m bits.
- Le codage de l’information, en entrée ou sortie, est représenté par l’absence (0) ou la présence (1) d’une tension électrique.
- On appelle ces circuits *logiques* car un bit d’information 0 est assimilé à la valeur de vérité *faux* et 1 à *vrai*
- On représente ainsi une application de $\{0,1\}^n$ dans $\{0,1\}^m$

Exemple : additionneur 1 bit

Soit un circuit électronique destiné à l'addition bit à bit.

Le schéma est le suivant

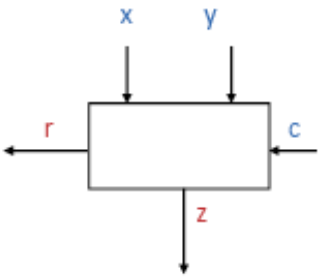


- Les entrées sont les bits X et Y et le report C .
- Les sorties sont le bit de sortie Z et la retenue r.
- Le résultat de $(X+Y+C)_2$ est donné par Z et r.

Table de vérité de l'additionneur bit à bit

On obtient ainsi la table de vérité

entrées			sorties	
x	y	c	z	r
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



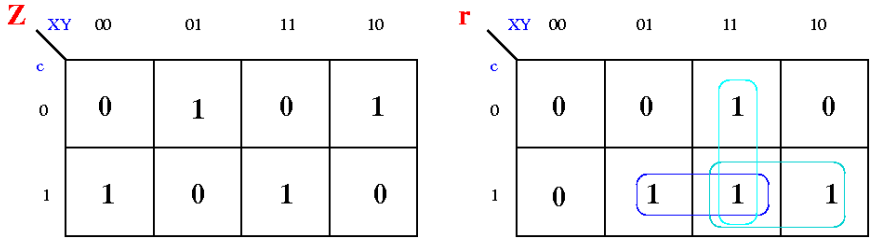
D'où :

$$Z = (\bar{x}.\bar{y}.c) + (.\bar{x}.y.\bar{c}) + (x.\bar{y}.\bar{c}) + (x.y.c)$$

$$r = (\bar{x}.y.c) + (x.\bar{y}.c) + (x.y.\bar{c}) + (x.y.c)$$

Additionneur : table de Karnaugh

- On peut dresser les tables de Karnaugh pour Z et r :



- On déduit une forme normale disjonctive simplifiée pour r

$$r = (x.y) + (x.c) + (y.c)$$

Il suffit de trois . et deux $+$ pour implémenter la retenue.

- La sortie Z s'écrit plus facilement avec l'opérateur $\oplus$, «ou exclusif»

$$Z = x \oplus y \oplus c\,, \qquad \text{avec}$$

$\oplus$	0	1
0	0	1
1	1	0

Spécification d'un circuit logique

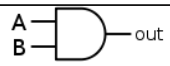
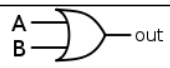
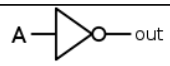
- C'est la description du fonctionnement interne du circuit
  - Par une table de vérité : on donne les valeurs de sortie pour chacune des entrées possibles : il y en a 2^n

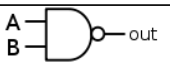
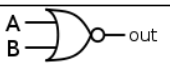
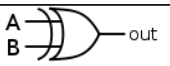
e_1	...	e_n	s_1	...	s_m
⋮		⋮	⋮		⋮
i_1	...	i_n	o_1	...	o_m
⋮		⋮	⋮		⋮

  - Par m fonctions booléennes de n variables, chacune donnant une sortie s_i, $1 \leq i \leq m$.
- On parle aussi de circuit logique «combinatoire» :
  le résultat ne dépend que des entrées,
  il n'y pas besoin de synchronisation ou d'«horloge».

Représentation des portes logiques élémentaires

- Un circuit est représenté grâce aux opérateurs de bases.
- La représentation des portes logiques de base est définie par des normes ANSI/IEEE.

Nom	Symbole	Opération
ET		$A.B$
OU		$A + B$
NON		$\overline{A}$

Nom	Symbole	Opération
NON-ET		$\overline{A.B}$
NON-OU		$\overline{A + B}$
OU exclusif		$A \oplus B$

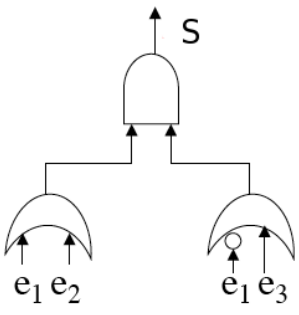
Voir `http://fr.wikipedia.org/wiki/Fonction_logique`

- La porte NON est souvent appelé **inverseur**.
  Utilisée en entrée ou en sortie, elle est représentée comme une «bulle».

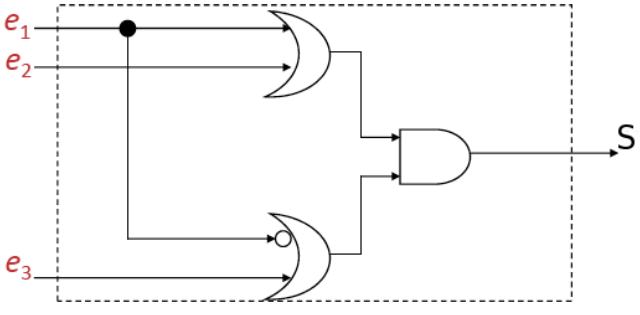
Des formules aux circuits

Exemple : $S = (e_1 + e_2).(\bar{e_1} + e_3)$

- Le circuit doit avoir autant d'entrées que de variables booléennes.
- On introduit des bifurcations par des • pour répéter des variables.



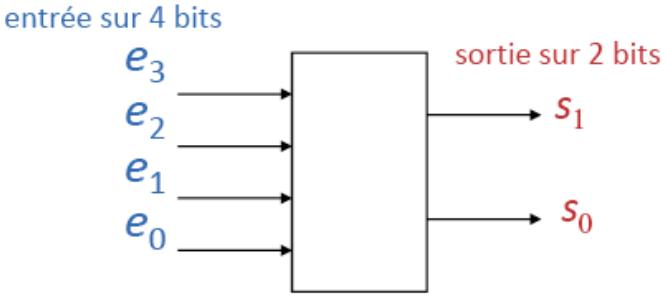
devient



Exemple : reste de la division par 3

Spécifier un circuit logique tel que

- l'entrée est un entier n compris entre 0 et 15
- la sortie est le reste de la division entière de n par 3
- alors
  1. l'entrée peut être codée en binaire naturel sur 4 bits ;
  2. la sortie peut être codée sur 2 bits ;
  3. Le schéma est



Exemple : reste de la division par 3

La table de vérité est

entier	e_3	e_2	e_1	e_0	s_1	s_0	Valeur du reste
0	0	0	0	0	0	0	**0**
1	0	0	0	1	0	1	**1**
2	0	0	1	0	1	0	**2**
3	0	0	1	1	0	0	**0**
4	0	1	0	0	0	1	**1**
5	0	1	0	1	1	0	**2**
6	0	1	1	0	0	0	**0**
7	0	1	1	1	0	1	**1**
8	1	0	0	0	1	0	**2**
9	1	0	0	1	0	0	**0**
10	1	0	1	0	0	1	**1**
11	1	0	1	1	1	0	**2**
12	1	1	0	0	0	0	**0**
13	1	1	0	1	0	1	**1**
14	1	1	1	0	1	0	**2**
15	1	1	1	1	0	0	**0**

Exemple : reste de la division par 3

L'on obtient les formes normales :

$$s_0 = (\bar{e}_3.\bar{e}_2.\bar{e}_1.e_0) + (\bar{e}_3.e_2.\bar{e}_1.\bar{e}_0) + (\bar{e}_3.e_2.e_1.e_0) + (e_3.\bar{e}_2.e_1.\bar{e}_0) + (e_3.e_2.\bar{e}_1.e_0)$$

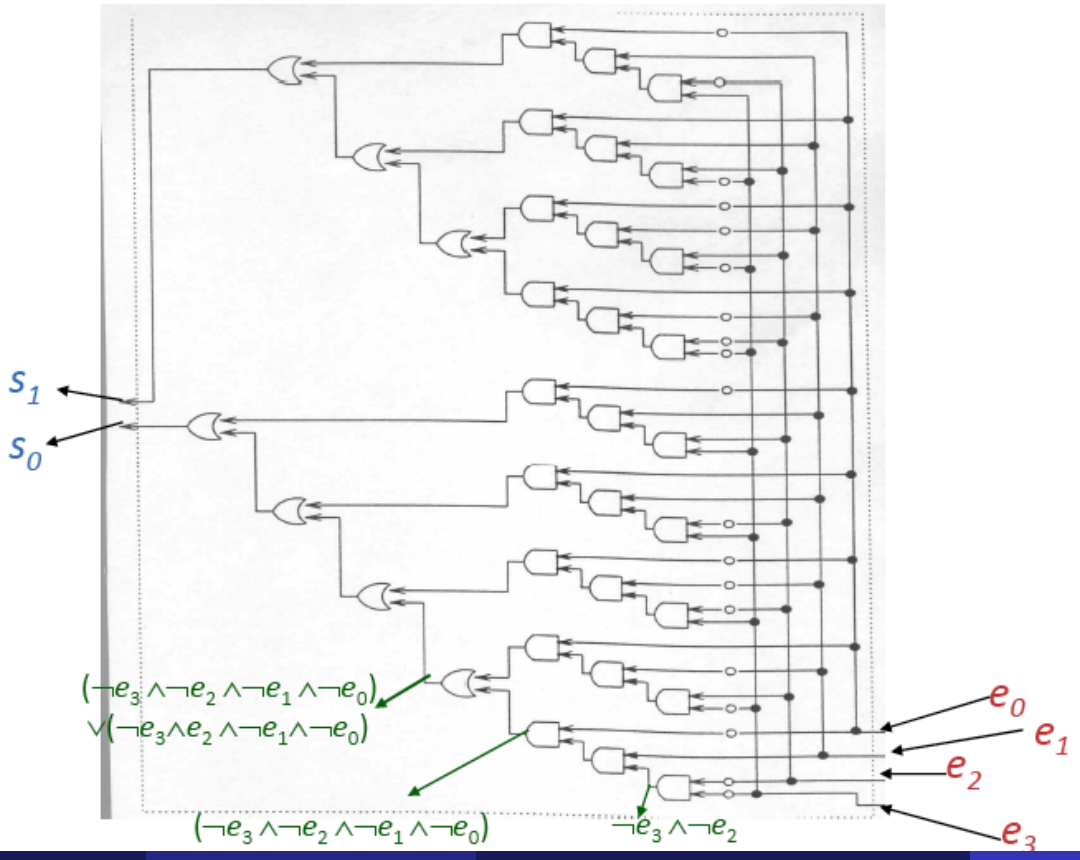
et

$$s_1 = (\bar{e}_3.\bar{e}_2.e_1.\bar{e_0}) + (\bar{e}_3.e_2.\bar{e}_1.e_0) + (e_3.\bar{e}_2.\bar{e}_1.\bar{e}_0) + (e_3.\bar{e}_2.e_1.e_0) + (e_3.e_2.e_1.\bar{e}_0)$$

d'où l'on peut tirer le schéma du circuit. . .

Exemple : reste de la division par 3

Le circuit. . .



Exemple : circuit additionneur 16 bits

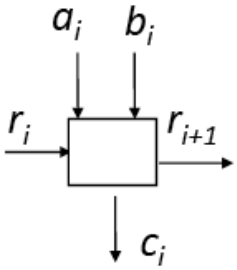
Un circuit additionneur 16 bits a 32 entrées et 17 sorties :

- 16 entrées reçoivent un premier entier naturel codé sur 16 bits ;
- 16 autres entrées reçoivent un deuxième entier naturel codé sur 16 bits ;
- 16 sorties constituent la représentation en base 2 de la somme deux entiers en entrée ;
- Le 17ème bit est un éventuel dépassement de capacité.

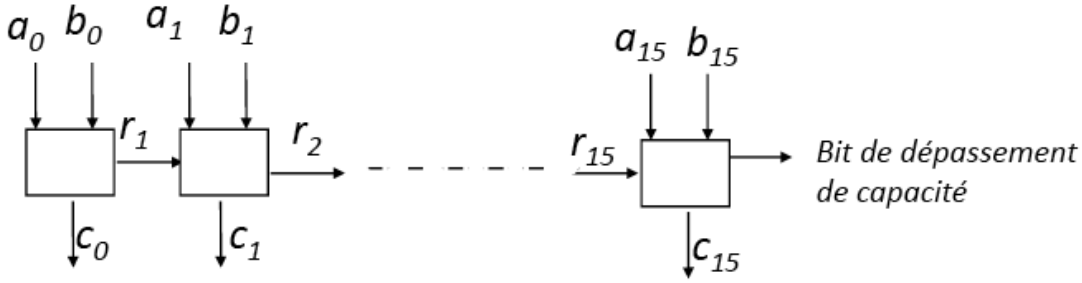
- Une spécification par table de vérité n'est pas envisageable.
- De même, écrire les 17 formules spécifiant les 17 sorties sous forme normale est laborieux !
- On construit/développe un circuit additionneur 1 bit.
  On composera autant de «circuit 1 bit » que nécessaire.
- Le circuit sera un circuit séquentiel encore appelé bascule :
  une partie du circuit doit attendre le résultat d'une autre partie du circuit, d'où besoin d'une horloge.

Exemple : circuit additionneur 16 bits

- L'additionneur 1 bit avec retenue



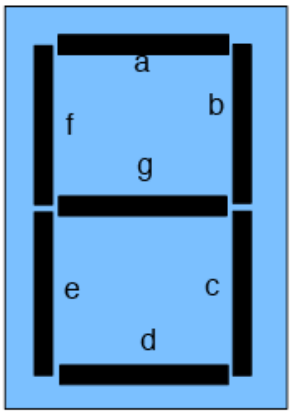
- L'additionneur 16 bits consiste en la succession de 16 additionneurs 1 bit

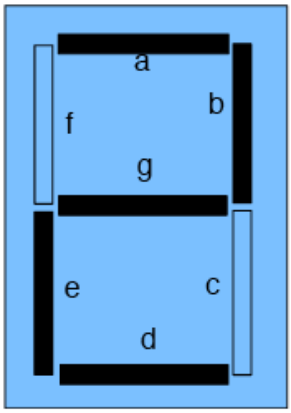


- Le $i$ème circuit 1 bit doit attendre le report du $(i-1)$ème circuit 1 bit, $1 \leq i \leq 16$, avant de commencer.

Affichage à cristaux liquides

- Un afficheur numérique est composé de sept barres : a, b, c, d, e, f et g qui peuvent être allumées ou éteintes ;
- On veut afficher les chiffres de 0 à 9 : l'entrée du circuit est le code binaire du chiffre à afficher, la sortie est le signal qui allume ou éteint chaque barre.
- Exemple pour afficher 2 : il faut que a=1, b=1 , c=0 , d=1 , e=1 , f=0 et g=1 .





Affichage à cristaux liquides

Circuit de la barre e : soient p, q, r et s les quatre bits d'entrée pour coder les 10 chiffres.
La table de vérité pour la barre e s'écrit alors

p	q	r	s	e
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0

Affichage à cristaux liquides

On écrit la formule E

$$E = (\bar{p}.\bar{q}.\bar{r}.\bar{s}) + (\bar{p}.\bar{q}.r.\bar{s}) + (\bar{p}.q.r.\bar{s}) + (p.\bar{q}.\bar{r}.\bar{s})$$

On simplifie

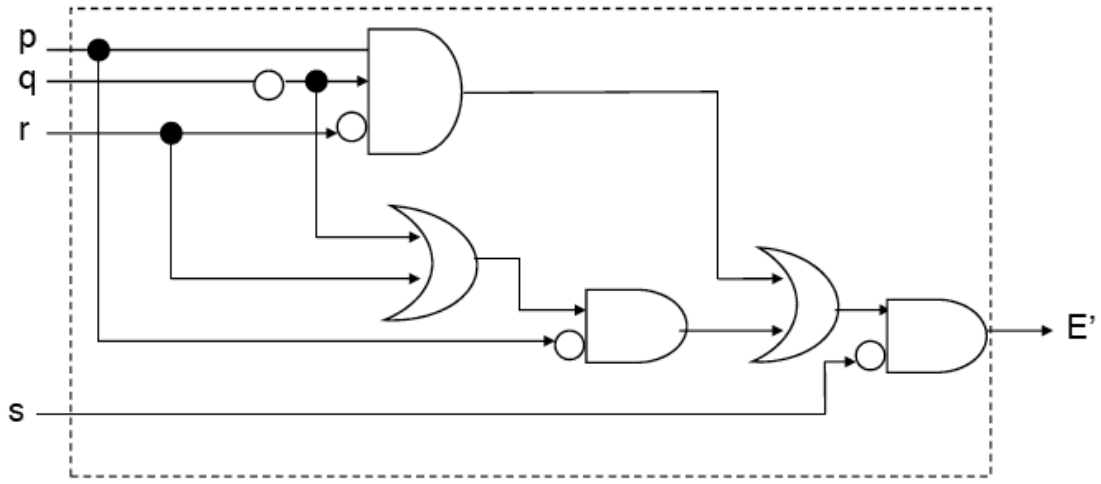
$$E = \bar{s}.([\bar{p}.(\bar{q} + r)] + (p.\bar{q}.\bar{r}))$$

L'affichage à cristaux liquides

À partir de

$$E = \bar{s}.([\bar{p}.(\bar{q} + r)] + (p.\bar{q}.\bar{r}))$$

on obtient le circuit de la barre e :



Notation polonaise

- Pour écrire les expressions algébriques impliquant des *opérateurs binaires*, on utilise en général la notion infixée :
  l'opérateur est écrit **entre** les opérandes.

  Exemples : "$2 + 2$" ; "$2 * (4 - 5)$" ; "$p \wedge (q \vee r)$" ; "$(a.b) + c$" ...

- En mathématiques, on utilise le plus souvent une notation préfixée pour les *opérateurs unaires* :
  $\sin\theta$ «sinus theta» ; $\log x$ «logarithme de x» .
  Une exception est $n!$ qui se lit «factorielle n».

- Le mathématicien polonais Jan Łukasiewicz a proposé en 1924 une notation préfixée pour les opérateurs binaires :
  l'opérateur est écrit **avant** les opérandes.

  Exemples : "$2 + 2$" s'écrit "+ 2 2" et "$2 * (4 - 5)$" s'écrit "* 2 - 4 5"

- En honneur de Łukasiewicz on parle de notation polonaise.

Notation polonaise

- Cette notation ne nécessite pas de parenthèses
  et est sans ambiguïté si les opérateurs sont binaires :

  Exemples : "* - 3 2 4" signifie "(3-2)*4" ;
  "/ * 4 2 2" signifie "(4*2)/2" et "* / 4 2 2" signifie "(4/2) * 2".

- Des notations préfixe sont utilisées dans des langages de programmations tels que `Lisp`, `Tcl`, `Apl`.

- Dans la notation polonaise inverse ou notation postfixée
  l'opérateur est écrit **après** les opérandes.

  Exemples : "2 2 +" ou "3 4 * 5 1 - *"

  Dans les années 1970/80, les calculatrices HP (Hewlett-Packard) ont utilisé ce principe :
  il y a une touche [ENTER] qui permet de remplir une **pile**,
  une touche [CHS], mais pas les touches [=], [(] ou [)].
  On utilise moins de touches qu'avec la notation infixe.

Notation polonaise

- Afin d'éviter des ambiguïtés un opérateur est soit unaire, soit binaire,. . .
- Łukasiewicz a introduit les notations suivantes pour le calcul des propositions

Not. standard	$\neg p$	$p \wedge q$	$p \vee q$	$p \rightarrow q$	$p \leftrightarrow q$
Not. polonaise	Np	Kpq	Apq	Cpq	Epq

- *Exemple :* Soit la formule $F = ((\neg a \vee b) \wedge c) \rightarrow (\neg\neg a \wedge \neg b)$

  Elle s'écrit $CKANabcKNNaNb$
- Cette notation
  1. n'utilise aucune parenthèse ;
  2. définit la formule F sans ambiguïté.

Notation polonaise préfixée

- Pas n'importe quelle suite de symboles ne représente une formule.
- Mais la notation polonaise préfixée permet un *test de bonne formation* très simple :

  On affecte
  1. le poids -1 aux variables
  2. le poids +1 aux connecteurs binaires K, A, C et E
  3. le poids 0 au connecteur unaire N
- Alors
  1. La somme des poids d'une formule est -1.
  2. Toute somme partielle à partir de la gauche est positive.
- *Exemples :*
  $ACpqr$ est bien formée et signifie "$(p \rightarrow q) \vee r$"
  $ACpEqr$ est mal formée : la somme vaut 0.