Algèbre de Boole et circuits logiques

Architecture des ordinateurs

Guillaume Blin

IGM-LabInfo UMR 8049,
Bureau 48066
Université de Marne La Vallée
gblin@univ-mlv.fr
http://igm.univ-mlv.fr/~gblin

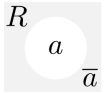
Références

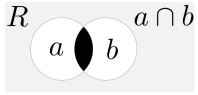
- Cours de Frédéric Goualard de l'Université de Nantes
  http ://supports.goualard.free.fr/
- Cours de Eric Garcia de l'IUT GTR, Montbéliard
- Emmanuel Viennet de l'IUT de Viltaneuse
- Cours de Danger et al de l'Électronique Numérique Intégrée de Telecom Paris

Plan

Plan

Notion ensembliste





Logique des propositions

- Proposition = énoncé vrai ou faux
  - a : les élèves sont présents
  - b : le professeur est présent
  - cours si a et b
- Combinaison par des connecteurs
  - $\neg$ (non logique) $\wedge$ (et logique) $\vee$ (ou logique non exclusif)

a	b	$\neg a$	$a \vee b$	$a \wedge b$
V	V	F	V	V
V	F	F	V	F
F	V	V	V	F
F	F	V	F	F

Passage à l'algèbre de Boole

- Il y a d'autres connecteurs et quelques lois dans la logique des propositions : $\neg a \wedge \neg b = \neg(a \vee b) \ldots$

a	b	$\neg a$	$\neg b$	$\neg a \wedge \neg b$	$a \vee b$	$\neg(a \vee b)$
V	V	F	F	F	V	F
V	F	F	V	F	V	F
F	V	V	F	F	V	F
F	F	V	V	V	F	V

- Georges Boole a exprimé la logique des propositions en termes algébriques
  - $a \vee b \Rightarrow a + b$
  - $a \wedge b \Rightarrow a.b$
  - $\neg a \Rightarrow \overline{a} = 1 - a$
- Claude Shannon a proposé : $V = 1$ et $F = 0$

Algèbre de Boole

- Il existes 16 fonctions à deux variables : toutes ne sont pas intéressantes

a	b	$a+b$	$\overline{a+b}$	ab	$\overline{ab}$	$a \oplus b$	$\overline{a \oplus b}$
0	0	0	1	0	1	0	1
0	1	1	0	0	1	1	0
1	0	1	0	0	1	1	0
1	1	1	0	1	0	0	1

- Ou exclusif (xor) $a \oplus b = a\overline{b} + \overline{a}b$
- Non ou (nor) $\overline{a+b}$
- Non et (nand) $\overline{ab}$

Opérations de base

- Toutes les fonctions peuvent s'exprimer à l'aide des trois opérations logiques et, ou, non
  - Avec n variables on peut construire 2^n fonctions
  - Une fonction à trois variables peut se décomposée en deux
    - $f(a,b,c) = f(0,b,c)$ si $a = 0$ et $f(a,b,c) = f(1,b,c)$ si $a = 1$
    - $f(a,b,c) = a.f(1,b,c) + \overline{a}.f(0,b,c)$
  - groupe logique complet = ensemble de fonction à partir desquels il est possible de réaliser toutes les fonctions {et,ou,non}...
  - Fonctions à n variables décomposables et fonctions à 2 variables exprimable avec {et,ou,non} $\Rightarrow$ récurrence ...
- L'algèbre de Boole se construit sur les booléens à partir des trois opérations internes : $+$, $*$ et

Théorèmes et axiomes

- Théorème des constantes

$$a + 0 = a \quad a.0 = 0 \quad a \oplus 0 = a$$
$$a + 1 = 1 \quad a.1 = a \quad a \oplus 1 = \overline{a}$$

- Idempotence

$$a + a = a \quad a.a = a \quad a \oplus a = 0$$

- Complémentation

$$a + \overline{a} = 1 \quad a.\overline{a} = 0 \quad a \oplus \overline{a} = 1$$

- Commutativité

$$a + b = b + a \quad a.b = b.a \quad a \oplus b = b \oplus a$$

Théorèmes et axiomes

- Distributivité

$$a + (bc) = (a + b)(a + c)$$
$$a(b + c) = ab + ac$$

- Associativité

$$a + (b + c) = (a + b) + c = a + b + c$$
$$a(bc) = (ab)c = abc$$

- Théorème de De Morgan

$$\overline{ab} = \overline{a} + \overline{b}$$
$$\overline{a + b} = \overline{a}.\overline{b}$$

Table de vérité et équation

- Pour concevoir un circuit, il faut le modéliser.
  - on utilise une table de vérité
  - on obtient une équation qu'on simplifie (algébrique ou Karnaugh)
  - on trace le schéma électrique du circuit
  - éléments constitutifs d'un ordinateur : exclusivement portes nand ou bien nor (système complet)
- Table de vérité : présente toutes les combinaisons possibles des n entrés (2^n lignes) et les états de sortie correspondant
- Une fois la table de vérité écrite, il faut la transformer en équation logique
  - somme (fonction ou) de produit (fonction et)

Équation : exemple

a	b	F(a,b)
0	0	1
0	1	0
1	0	1
1	1	1

$F(a,b) = 1$ si
$a = 0$ et $b = 0$ soit $\overline{a} = 1$ et
$\overline{b} = 1$ donc si $\overline{a}.\overline{b} = 1$
$a = 1$ et $b = 0$ soit $a = 1$ et
$\overline{b} = 1$ donc si $a.\overline{b} = 1$
$a = 1$ et $b = 1$ donc si $a.b = 1$
$\Rightarrow F(a,b) = \overline{a}.\overline{b} + a.\overline{b} + a.b$

Équation : exemple

a	b	F(a,b)
0	0	1
0	1	0
1	0	1
1	1	1

$F(a,b) = \overline{a}.\overline{b} + a.(\overline{b} + b)$

$F(a,b) = 1$ si
$a = 0$ et $b = 0$ soit $\overline{a} = 1$ et
$\overline{b} = 1$ donc si $\overline{a}.\overline{b} = 1$
$a = 1$ et $b = 0$ soit $a = 1$ et
$\overline{b} = 1$ donc si $a.\overline{b} = 1$
$a = 1$ et $b = 1$ donc si $a.b = 1$
$\Rightarrow F(a,b) = \overline{a}.\overline{b} + a.\overline{b} + a.b$

Équation : exemple

a	b	F(a,b)
0	0	1
0	1	0
1	0	1
1	1	1

$F(a,b) = \overline{a}.\overline{b} + a.(\overline{b} + b)$
$F(a,b) = \overline{\overline{a}.\overline{b} + a}$

$F(a,b) = 1$ si
$a = 0$ et $b = 0$ soit $\overline{a} = 1$ et
$\overline{b} = 1$ donc si $\overline{a}.\overline{b} = 1$
$a = 1$ et $b = 0$ soit $a = 1$ et
$\overline{b} = 1$ donc si $a.\overline{b} = 1$
$a = 1$ et $b = 1$ donc si $a.b = 1$
$\Rightarrow F(a,b) = \overline{a}.\overline{b} + a.\overline{b} + a.b$

Équation : exemple

a	b	F(a,b)
0	0	1
0	1	0
1	0	1
1	1	1

$F(a,b) = \overline{a}.\overline{b} + a.(\overline{b} + b)$
$F(a,b) = \overline{\overline{a}.\overline{b} + a}$
$F(a,b) = ((\overline{\overline{a}.\overline{b}}.\overline{a}))$

$F(a,b) = 1$ si
$a = 0$ et $b = 0$ soit $\overline{a} = 1$ et
$\overline{b} = 1$ donc si $\overline{a}.\overline{b} = 1$
$a = 1$ et $b = 0$ soit $a = 1$ et
$\overline{b} = 1$ donc si $a.\overline{b} = 1$
$a = 1$ et $b = 1$ donc si $a.b = 1$
$\Rightarrow F(a,b) = \overline{a}.\overline{b} + a.\overline{b} + a.b$

Équation : exemple

a	b	F(a,b)
0	0	1
0	1	0
1	0	1
1	1	1

$F(a,b) = 1$ si
$a = 0$ et $b = 0$ soit $\overline{a} = 1$ et
$\overline{b} = 1$ donc si $\overline{a}.\overline{b} = 1$
$a = 1$ et $b = 0$ soit $a = 1$ et
$\overline{b} = 1$ donc si $a.\overline{b} = 1$
$a = 1$ et $b = 1$ donc si $a.b = 1$
$\Rightarrow F(a,b) = \overline{a}.\overline{b} + a.\overline{b} + a.b$

$F(a,b) = \overline{a}.\overline{b} + a.(\overline{b} + b)$
$F(a,b) = \overline{\overline{\overline{a}.\overline{b} + a}}$
$F(a,b) = \overline{(\overline{(\overline{a}.\overline{b}.\overline{a})})}$
$F(a,b) = \overline{(a+b).\overline{a}}$

Algèbre de Boole Electronique Circuits combinatoires UAL Circuits séquentiels : bascules Circuits séquentiels : Machines à états ou comment faire une machine à laver

Équation : exemple

a	b	F(a,b)
0	0	1
0	1	0
1	0	1
1	1	1

$F(a,b) = 1$ si
$a = 0$ et $b = 0$ soit $\overline{a} = 1$ et
$\overline{b} = 1$ donc si $\overline{a}.\overline{b} = 1$
$a = 1$ et $b = 0$ soit $a = 1$ et
$\overline{b} = 1$ donc si $a.\overline{b} = 1$
$a = 1$ et $b = 1$ donc si $a.b = 1$
$\Rightarrow F(a,b) = \overline{a}.\overline{b} + a.\overline{b} + a.b$

$F(a,b) = \overline{a}.\overline{b} + a.(\overline{b} + b)$
$F(a,b) = \overline{\overline{\overline{a}.\overline{b} + a}}$
$F(a,b) = \overline{(\overline{(\overline{a}.\overline{b}.\overline{a})})}$
$F(a,b) = \overline{(a+b).\overline{a}}$
$F(a,b) = \overline{a.\overline{a} + b.\overline{a}}$

Algèbre de Boole Electronique Circuits combinatoires UAL Circuits séquentiels : bascules Circuits séquentiels : Machines à états ou comment faire une machine à laver

Équation : exemple

a	b	F(a,b)
0	0	1
0	1	0
1	0	1
1	1	1

$F(a,b) = 1$ si
$a = 0$ et $b = 0$ soit $\overline{a} = 1$ et
$\overline{b} = 1$ donc si $\overline{a}.\overline{b} = 1$
$a = 1$ et $b = 0$ soit $a = 1$ et
$\overline{b} = 1$ donc si $a.\overline{b} = 1$
$a = 1$ et $b = 1$ donc si $a.b = 1$
$\Rightarrow F(a,b) = \overline{a}.\overline{b} + a.\overline{b} + a.b$

$F(a,b) = \overline{a}.\overline{b} + a.(\overline{b} + b)$
$F(a,b) = \overline{\overline{\overline{a}.\overline{b} + a}}$
$F(a,b) = \overline{(\overline{(\overline{a}.\overline{b}.\overline{a})})}$
$F(a,b) = \overline{(a+b).\overline{a}}$
$F(a,b) = \overline{a.\overline{a} + b.\overline{a}}$
$F(a,b) = \overline{b.\overline{a}}$
$F(a,b) = \overline{b} + \overline{\overline{a}}$
$F(a,b) = \overline{b} + a$

Algèbre de Boole Electronique Circuits combinatoires UAL Circuits séquentiels : bascules Circuits séquentiels : Machines à états ou comment faire une machine à laver

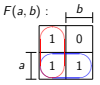
Diagrammes de Karnaugh

- Table de Karnaugh : principe
  - départ : somme de produits où chaque produit doit contenir toutes les variables (pour simplifier) : $(a + \overline{a})b = b$
  - chaque colonne diffère de sa voisine d'un seul littéral
  - les tables de Karnaugh sont à deux dimensions : on regroupe des variables
  - on regroupe les 1 en morceaux rectangulaires
    - plus grands morceaux possibles
    - moins de morceaux possibles
    - nouveau morceau que s'il permet de regrouper des 1 non encore regroupés
    - la ligne du haut et du bas ainsi que colonne de droite et de gauche sont adjacentes
  - $\Rightarrow$ Morceau = produit de variable : variable et son inverse dans le même morceau = élimination de la variable

Algèbre de Boole Electronique Circuits combinatoires UAL Circuits séquentiels : bascules Circuits séquentiels : Machines à états ou comment faire une machine à laver

Karnaugh : Exemple 1

- $F(a,b) = \overline{a}.\overline{b} + a.\overline{b} + a.b$

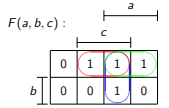


- $a + \overline{b}$

Algèbre de Boole Electronique Circuits combinatoires UAL Circuits séquentiels : bascules Circuits séquentiels : Machines à états ou comment faire une machine à laver

Karnaugh : Exemple 2

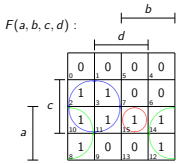
- $F(a,b,c) = \overline{a}.\overline{b}.c + a.\overline{b} + a.c$
  $F(a,b,c) = \overline{a}.\overline{b}.c + a.\overline{b}.(c + \overline{c}) + a.c.(b + \overline{b})$
  $F(a,b,c) = \overline{a}.\overline{b}.c + a.\overline{b}.c + a.\overline{b}.\overline{c} + a.b.c + a.\overline{b}.c$
  $F(a,b,c) = \overline{a}.\overline{b}.c + a.\overline{b}.c + a.\overline{b}.\overline{c} + a.b.c$



- $a\overline{b} + \overline{b}.c + ac$

Karnaugh : Exemple 3

- $F(a,b,c,d) = \overline{a}.\overline{b}.c + a.\overline{d} + a.c$

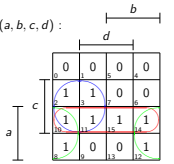


- $\overline{b}.c + a.b.c.d + a.\overline{d}$!!!!!!!
- PAS BON !

Algèbre de Boole Electronique Circuits combinatoires UAL Circuits séquentiels : bascules Circuits séquentiels : Machines à états ou comment faire une machine à laver

Karnaugh : Exemple 3

- $F(a,b,c,d) = \overline{a}.\overline{b}.c + a.\overline{d} + a.c$



- $\overline{b}.c + a.b.c.d + a.\overline{d}$!!!!!!!
- PAS BON !

G. Blin - Architecture des ordinateurs Algèbre de Boole et circuits logiques

Algèbre de Boole Electronique Circuits combinatoires UAL Circuits séquentiels : bascules Circuits séquentiels : Machines à états ou comment faire une machine à laver

Portes et circuits logiques

- Un ordinateur travaille en base 2
  - Électroniquement 0 correspondait à une tension de 0 à 0,8V et 1 à une tension de 2,8 à 2,5V (tensions données par les constructeurs de composants)
  - Toute fonction binaire peut être représentée par une expression booléenne
  - Tout circuit électrique ou électronique à deux valeurs de tension peut être représenté par une expression booléenne
- Préférable de représenter les circuits par des symboles logiques et non par des expressions booléennes
  - Correspondance entre les différentes fonctions logiques $(+, *\ldots)$ et des symboles appelés portes logiques

G. Blin - Architecture des ordinateurs Algèbre de Boole et circuits logiques

Algèbre de Boole Electronique Circuits combinatoires UAL Circuits séquentiels : bascules Circuits séquentiels : Machines à états ou comment faire une machine à laver

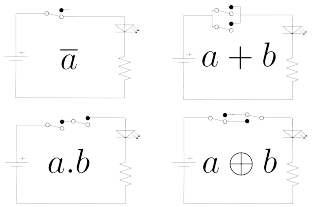
Portes logiques

NON(a)		$\overline{a}$
AND(a,b)		$a+b$
OR(a,b)		$a.b$
XOR(a,b)		$a \oplus b$

G. Blin - Architecture des ordinateurs Algèbre de Boole et circuits logiques

Algèbre de Boole Electronique Circuits combinatoires UAL Circuits séquentiels : bascules Circuits séquentiels : Machines à états ou comment faire une machine à laver

Portes logiques et circuits



G. Blin - Architecture des ordinateurs Algèbre de Boole et circuits logiques

Algèbre de Boole Electronique Circuits combinatoires UAL Circuits séquentiels : bascules Circuits séquentiels : Machines à états ou comment faire une machine à laver

Portes logiques : mise en série

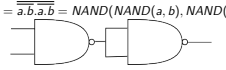
NOR(a,b)		$\overline{a+b}$
NAND(a,b)		$\overline{a.b}$
ID(a,b)		$\overline{a \oplus b}$

G. Blin - Architecture des ordinateurs Algèbre de Boole et circuits logiques

Algèbre de Boole Electronique Circuits combinatoires UAL Circuits séquentiels : bascules Circuits séquentiels : Machines à états ou comment faire une machine à laver

Système complet

- Porte NAND = la plus simple à réaliser du point de vue technologique.
- Possible de réaliser toutes les fonctions logiques en utilisant uniquement NAND
- AND : $a.b = \overline{\overline{a.b}.\overline{a.b}} = NAND(NAND(a,b), NAND(a,b))$

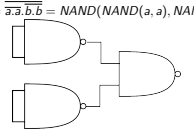


G. Blin - Architecture des ordinateurs Algèbre de Boole et circuits logiques

Algèbre de Boole Electronique Circuits combinatoires UAL Circuits séquentiels : bascules Circuits séquentiels : Machines à états ou comment faire une machine à laver

Système complet

- Porte NAND = la plus simple à réaliser du point de vue technologique.
- Possible de réaliser toutes les fonctions logiques en utilisant uniquement NAND
- OR : $a + b = \overline{\overline{a.a}.\overline{b.b}} = NAND(NAND(a,a), NAND(b,b))$



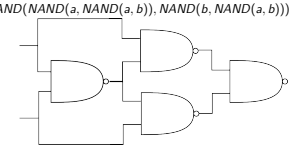
G. Blin - Architecture des ordinateurs Algèbre de Boole et circuits logiques

Système complet

- Porte NAND = la plus simple à réaliser du point de vue technologique.
- Possible de réaliser toutes les fonctions logiques en utilisant uniquement NAND
- NOT : $\overline{a} = \overline{a.a} = NAND(a, a)$

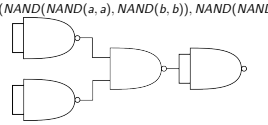
Système complet

- Porte NAND = la plus simple à réaliser du point de vue technologique.
- Possible de réaliser toutes les fonctions logiques en utilisant uniquement NAND
- XOR : $a \oplus b = \overline{\overline{a.a.b}.\overline{b.a.b}} = NAND(NAND(a, NAND(a, b)), NAND(b, NAND(a, b)))$



Système complet

- Porte NAND = la plus simple à réaliser du point de vue technologique.
- Possible de réaliser toutes les fonctions logiques en utilisant uniquement NAND
- NOR : $\overline{a + b} = \overline{\overline{a.a.b.b}.\overline{a.a.b.b}} = NAND(NAND(NAND(a, a), NAND(b, b)), NAND(NAND(a, a), NAND(b, b)))$



Plan

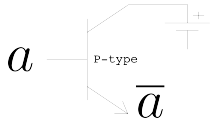
Transistor

- Un transistor est un interrupteur qui permet de réunir ou de séparer deux fils.
- L'état d'un transistor (passant ou non), est commandé par un troisième fil.
- Il existe deux types de transistors :
  - les transistors de type N ("non-passant par défaut"), pour lesquels le transistor est non passant quand le fil de commande est dans l'état 0,

  - et les transistors de type P ("passant par défaut"), pour lesquels le transistor est passant quand le fil de commande est dans l'état 0.

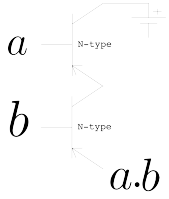
Circuits intégrés

- A partir de ce dispositif on peut réaliser des portes de base (NON, ET, OU ...)



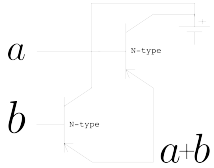
Circuits intégrés

- A partir de ce dispositif on peut réaliser des portes de base (NON, ET, OU ...)



Circuits intégrés

- A partir de ce dispositif on peut réaliser des portes de base (NON, ET, OU ...)



Circuits intégrés

- Les circuits intégrés (chip) peuvent être classifiés
  - SSI (Small Scale Integration) contenant moins de 100 portes
  - MSI (Medium) : entre 100 et 1000 portes
  - LSI (Large) : entre 1000 et 10^5 portes
  - VLSI (Very Large) : entre 10^5 et 10^7 portes
  - ULSI (Ultra Large) : plus de 10^7 portes
- Portes logiques : réalisées électroniquement par un ou deux transistors
- Plusieurs portes logiques forment un circuit logique = circuit intégré
  - circuits combinatoires : ne font que combiner les variables d'entrée selon une table de vérité
  - circuits séquentiels : construits à partir de circuits combinatoires + capacité de mémorisation
  - La réalisation d'un circuit passe par la recherche des expressions booléennes, puis par leur simplification (règles ou tableau de Karnaugh)

Plan

Demi-additionneur

- Addition de deux bits x et y

x	y	S	R
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

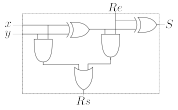
$R = xy$ $S = x \oplus y = (x + y).xy$

Additionneur complet

- Addition de deux nombres
  - addition bit à bit
  - considération de la retenue précédente
  - somme de 3 bits = additionneur complet
- Somme S
  - vaut 1 si entre x, y et re (retenue d'entrée) : le nombre de bits à 1 est impaire
- Retenue de sortie rs
  - vaut 1 si x et y valent 1, ou si l'un des deux vaut 1 alors que revaut 1

Additionneur complet

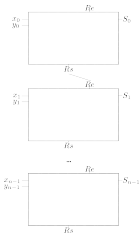
x	y	re	S	rs
1	1	0	0	1
1	1	1	1	1
1	0	0	1	0
1	0	1	0	1
0	1	0	1	0
0	1	1	0	1
0	0	0	0	0
0	0	1	1	0



- $S = re \oplus x \oplus y$
- $rs = xy + x.re + y.re$
- $rs = xy + re.(x \oplus y)$

Additionneur complet : propagation

- Forme la plus simple : propagation de retenue
  - le calcul sur chaque bit se fait de façon différée
  - rang 0 en premier puis rang 1 avec la retenue du rang 0
- Temps de calcul
  - si un additionneur 1 bit met 1T pour calculer la retenue et 1,5T pour le résultat
  - rs_0 à T et S_0 à 1,5T
  - rs_1 à 2T et S_1 à 2,5T
  - rs_{n-1} à nT et S_{n-1} à (n+0,5)T
  - Temps : (n+0,5)*T = linéaire



Additionneur complet : améliorations

- La lenteur de l'additionneur par propagation de retenue impose d'utiliser d'autres techniques pour des additionneurs ayant un nombre important de bits. Comme cette lenteur est due au temps nécessaire à la propagation de la retenue, toutes les techniques ont pour but d'accélérer le calcul des retenues. La première technique appelée anticipation de retenue consiste à faire calculer les retenues par un circuit extérieur.

Additionneur complet : améliorations

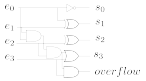
- Additionneur par anticipation de retenue
  - Afin de faciliter le calcul des retenues, on introduit deux quantités appelées G (pour Generate en anglais) et P (pour Propagate en anglais).
  - Soient $A = A_{n-1} \dots A_0$ et $B = B_{n-1} \dots B_0$ deux entrées de n bits. On note C_i la retenue de l'addition des i bits de poids faible de A et B. Pour accélérer le calcul des C_i, on introduit les deux quantités G_i et P_i associées aux entrées A_i et B_i par les formules suivantes.
    $G_i = A_i B_i$ et $P_i = A_i + B_i$
  - $C_{i+1} = G_i + P_i C_i$
  - $C_0 = G_0 + Re$
  - Par récursivité on peut calculer toutes les C_i en traversant seulement 3 portes logiques
  - Pb : il faut des portes logiques avec de nombreuses entrées

Additionneur complet : améliorations

- Additionneur par sélection de retenue
  - L'idée générale de l'additionneur par sélection de retenue est d'effectuer en parallèle le calcul avec une retenue de 0 et le calcul avec une retenue de 1 puis de sélectionner le bon résultat lorsque la retenue est enfin connue.
  - Le problème est que ca double le nombre de portes

Incrémenteur

- Ajouter ou retrancher 1 : opération fréquente d'un processeur
  - utiliser l'addition : utilisation non optimale
  - Exemple : incrémenteur 4 bits



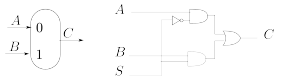
$s_0 = \overline{e_0}$
$s_i = \overline{e_i}$ *si* $e_{i-1} \dots e_0 = 1$; e_i *sinon*
donc $s_i = e_i \oplus (e_{i-1}x \dots xe_0)$

Multiplexeur

- Permet d'envoyer sur la sortie (C) l'état d'une entrée (A) ou de l'autre (B) en fonction d'un signal de sélection (S)

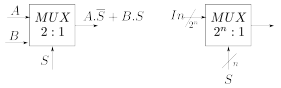
A	B	S	C
0	X	0	0
1	X	0	1
X	0	1	0
X	1	1	1

$C = A.(B + \overline{B}).S + (A + \overline{A}).B.S$
$C = A.\overline{S} + B.S$

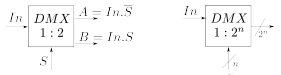


Multiplexeur/Demultiplexeur

- Multiplexeur à deux et 2^n entrées (n bits pour coder 2^n val $\neq$)



- Demultiplexeur : aiguille l'entrée sur la sortie num S

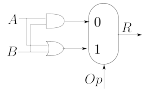


Plan

UAL : principe

- UAL : Unité Arithmétique et logique
  - effectue les opérations de bases (arithmétiques et logiques)
  - un code d'entrée détermine la partie du circuit qui va effectuer les opérations
- UAL 1 bit : opération ET / OU
  - en fonction d'un signal Op le circuit calcul a ET b (Op=0) ou bien a OU b (Op=1)

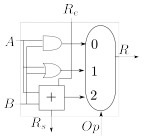
$S = a.b.\overline{Op} + (a + b).Op$
$\Rightarrow$ Multiplexeur



UAL : ET, OU, +

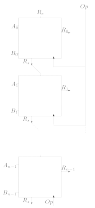
- UAL 2 bits : opération ET / OU / +
  - on ajoute un additionneur

Op1	Op0	S
0	0	a et b
0	1	a ou b
1	0	a+b
1	1	libre



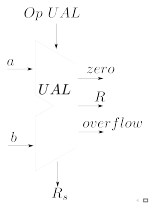
UAL n bits

- Traitement de données codées sur n bits



UAL : représentation

- UAL chargée des opérations
  - logiques : AND, OR, XOR, NOT, CMP, LSL, LSR, ASR (décalages)
  - arithmétiques : ADD, SUB, MUL, DIV, INC (+1), DEC (-1)



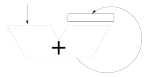
UAL : autre représentation

- UAL 8 bits
  - c_0, c_1, c_2, c_3 = sélection de l'opération
  - R = résultat
  - OF, CF, ZF, SF, PF = Drapeau (booléen) décrivant l'état du résultat

Plan

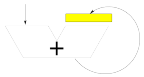
La logique séquentielle : rôle

- Elle permet d'organiser les calculs booléens dans le temps
- Ex : L'addition de 1000 nombres
  - Sol. 1 : effectuer 999 additions avec 999 additionneurs
  - Sol. 2 : additionner successivement les 1000 opérandes avec 1 seul additionneur.
    ⇒ Le circuit séquentiel
    - présenter successivement les 1000 opérandes
    - accumuler le résultat de l'addition et le présenter comme 2nde opérande
    - arrêter le calcul sur le 1000ème opérande



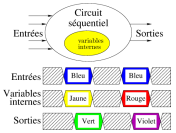
La logique séquentielle : rôle

- La logique séquentielle procure également la possibilité d'ordonnancement temporel et conditionnel
- Le séquencement nécessite une fonction propre à la logique séquentielle : la mémorisation.
- Celle-ci permet de geler les données et les commandes de façon à les réutiliser dans un ordre défini.



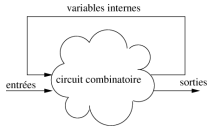
Comment reconnaître la logique séquentielle ?

- Combinatoire : à tout instant, mêmes valeurs d'entrée ⇒ mêmes résultats
- Séquentiel : pas cette propriété ; la connaissance des entrées appliquées à un instant donné ne suffit pas à déterminer les valeurs des sorties
- Due aux variables supplémentaires internes au circuit dont la valeur évolue au cours du temps
- Prévoir les sorties impose la connaissance de la valeur de ces variables internes



Comment construire la logique séquentielle ?

- Les valeurs des var. internes ≡ l'état du système (qui dépend des entrées et de l'état précédent)
- Les var. internes ≡ une partie de l'histoire du circuit
- Circuit séquentiel ≡ circuit combinatoire calculant les var. internes suivant les entrées et la valeur actuelle des var. internes ⇒ rebouclage.
- Ce rebouclage = mémorisation de la logique séquentielle

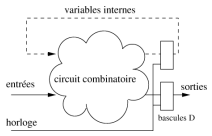


Problème de ce dispositif

- Problème
  - Temps de propagation dans un circuit combinatoire variables et dispersifs
  - Fiabilité du dispositif ?
    courses entre signaux ↦ dysfonctionnement du système
- Solution
  - ⇒ Synchronisation des calculs via des mémoires
  - On fige les valeurs des var. internes et sorties jusqu'à leurs m-à-j à la fin du calcul
  - Les sorties sont mémorisées car potentiellement utilisées comme entrées d'autres circuits séquentiels.
  - La m-à-j des mémoires est synchrone via une horloge

Problème de ce dispositif

- Problème
  - Temps de propagation dans un circuit combinatoire variables et dispersifs
  - Fiabilité du dispositif ?
    courses entre signaux $\mapsto$ dysfonctionnement du système
- Solution



Condition de fiabiilté de ce dispositif

- Synchroniser sur une unique horloge impose
  - de connaître le temps maximum de calcul du circuit combinatoire
  - fixer une période d'horloge supérieure à ce temps
- Chemin le plus lent d'un circuit combinatoire est appelé chemin critique
- On calcul ce chemin dans les conditions d'utilisation les pires

Mémorisation via des bascules

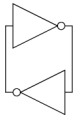
- La brique de base = point mémoire
- Différentes technologies pour créer ce dernier
- La bascule D est un composant de mémorisation pour un seul point mémoire
- Cette dernière rentre dansla composition de la Random Access Memory qui n'est autre qu'un ensemble de points mémoires regroupés dans une matrice

Le point mémoire élémentaire

- Basé sur le principe de stabilité des systèmes en boucle fermée
- Ex un amplificateur (une porte identité) rebouclé sur elle même = un système stable (qui ne change pas d'état)
- Avec un moyen de l'initialiser avec une valeur donnée, celle ci est gelée $\mapsto$ point mémoire

Le point mémoire bi-stable

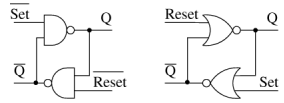
- En pratique, l'amplificateur est réalisé avec 2 inverseurs en tête bêche



- Pour initialiser le point mémoire, il faut forcer un niveau sur l'entrée d'un inverseur et son opposé sur l'autre inverseur.

Le point mémoire bi-stable : Bascule RS

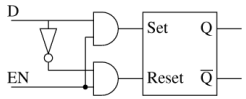
- La bascule RS (Reset Set) utilise une initialisation logique à l'aide de portes NAND ou NOR
- Ex RS à base de NAND et NOR :



- Cas avec NOR
  - RESET actif et SET inactif $\Rightarrow$ Q=0
  - SET actif et RESET inactif $\Rightarrow$ Q=1
  - RESET et SET inactifs $\Rightarrow$ mode mémoire – stable
  - le cas restant est instable
- Ici var. interne $\equiv$ Q (réentrant)

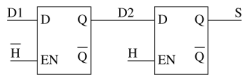
De la bascule RS à la bascule D sur état : le latch

- Bascule D = RS où Reset et Set sont remplacées par une unique entrée $D = S = \overline{R}$
- Mémorisation si $R = S = 0 \Rightarrow$ entrée EN (la clock) pour forcer les entrées à 0
- Quand EN=1, il y a recopie de l'entrée sur la sortie, le latch est transparent
- Quand EN=0, il est en état de mémorisation



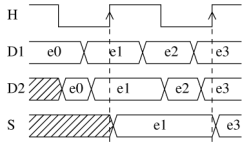
La bascule D sur front ou Flip-Flop

- Le latch ne convient pas car il perd sa fonction de mémorisation durant une phase de l'horloge, quand il est transparent (les variations en entrées sont recopiées en sortie).
- La bascule D = 2 latches en cascade disposant d'entrées EN complémentaires
- Quand la première est transparente, la seconde mémorise ; quand la seconde mémorise



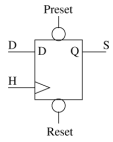
La bascule D sur front ou Flip-Flop

- S ne change que sur front montant de l'horloge H, et est mémorisée pendant une période d'horloge.
- Toutefois
  - S ne change pas immédiatement ↔ temps de propagation
  - Si D1 change entre 2 fronts ↗ (e.g. e0 et e2) ↦ pas pris en compte
  - Seules comptent les valeurs de D1 au moment du front ↗



La bascule D sur front ou Flip-Flop

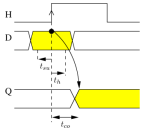
- Entrées optionnelles de mise à 1 (Preset) ou mise à 0 (Reset/Clear)
- Ces entrées sont asynchrones (i.e. actives immédiatement) et actives à 0 (indiqué par un cercle)
- La bascule D peut être sensibile au front ↘ plutôt que ↗
- Dans ce cas le symbole de la bascule dispose d'un cercle sur l'entrée de l'horloge signifiant l'inversion de polarité.



D	H	P	R	Q	Etat
X	↗	1	1	X	échantillonage
X	X	1	1	Q	memorisation
X	X	0	1	1	Preset
X	X	1	0	0	Reset

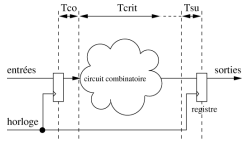
Conditions d'utilisation de la bascule

- Les constructeurs spécifient une fenêtre temporelle autour de l'instant d'échantillonnage, dans laquelle la variable d'entrée ne doit pas changer de valeur : t_{co}
- Cette fenêtre est définie à l'aide de :
  - t_{su} : set up time ; temps durant lequel les données doivent rester constantes avant le front montant d'horloge.
  - t_h : hold time ; temps durant lequel les données doivent rester constantes après le front montant d'horloge.



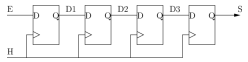
Influence sur le chemin critique

- Dans le calcul du chemin critique, les temps T_{co} et T_{su} doivent être pris en compte
- Si T_{cl} est la période d'horloge alors il faut respecter : $T_{cl} > T_{co} + T_{crit} + T_{su}$

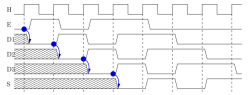


Utilisation des bascules D : registre à décalage

- Un registre est par définition un ensemble de bascules.
- Un registre à décalage est constitué de N bascules en cascade

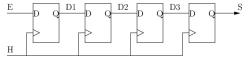


- A chaque front d'horloge, le contenu de chaque bascule amont est décalé dans la bascule aval (propagation de la 1ere valeur en N fronts montants)



Utilisation des bascules D : registre à décalage

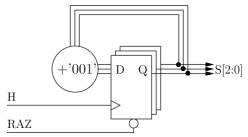
- Un registre est par définition un ensemble de bascules.
- Un registre à décalage est constitué de N bascules en cascade



- A chaque front d'horloge, le contenu de chaque bascule amont est décalé dans la bascule aval (propagation de la 1ere valeur en N fronts montants)
- Utile, entre autre, pour les passages série/parrallèle
- Série à parallèle : les bits rentrent en série et les N bits du registre sont les sorties.
- Parallèle à série : les bascules sont initialisées par un mot d'entrée et la sortie s'effectue sur la dernière bascule.

Utilisation des bascules D : les compteurs

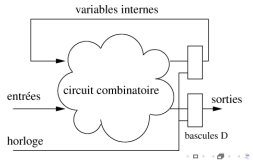
- Version simple : incrémenteur synchronisé sur horloge
- Version plus complexe : compteur cyclique e.g. 0, 1, 2, 3, 4, 5, 6, 7, 0, 1, ...
- Compteur = une horloge et un signal de RAZ asynchrone + un incrémenteur combinatoire de 3 bits (+'001') + un registre 3 bits



Plan

Machine à états - Définition

- Les machines à états = circuits séquentiels servant exclusivement à générer des signaux de commande
- La machine à état représente les parties contrôle et opérative
- Les états de la machine à états représentent toutes les valeurs que peuvent prendre les variables internes du circuit de logique séquentielle

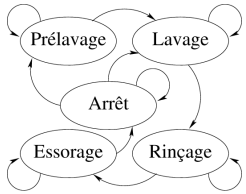


Le graphe d'états - représentation du comportement

- Dans une machine à états, l'évolution du système et les sorties suivent des lois établies par le concepteur
- Le graphe d'états est l'un des outils les plus utilisés pour la spécification de la machine à états (entrées, sorties, fonctionnement souhaité).
- Le graphe d'états représente graphiquement les états d'une machine à états.

Le graphe d'états - par l'exemple

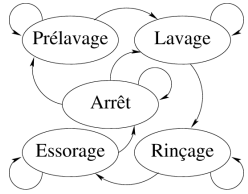
- Ex. d'une machine à laver



- Le graphe = l'évolution possible au cours du temps

Le graphe d'états - par l'exemple

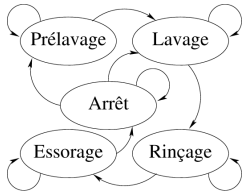
- Ex. d'une machine à laver



- A tout instant la machine est dans l'un des états (i.e. l'état courant)

Le graphe d'états - par l'exemple

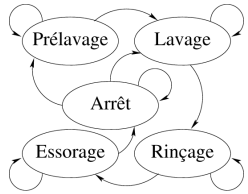
- Ex. d'une machine à laver



- A chaque front montant, la machine emprunte l'une des transitions possibles à partir de l'état courant

Le graphe d'états - par l'exemple

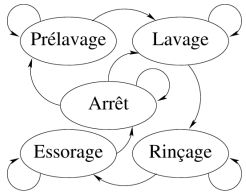
- Ex. d'une machine à laver



- Elle change alors d'état

Le graphe d'états - par l'exemple

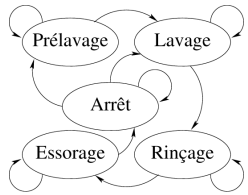
- Ex. d'une machine à laver



- Machine synchrone sur front montant : elle reste dans un état donné pendant une période

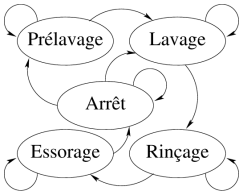
Le graphe d'états - par l'exemple

- Ex. d'une machine à laver



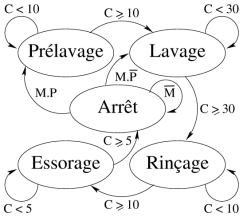
- Les transitions sont quasi-instantanées et ≡ aux fronts montants

Machine à laver



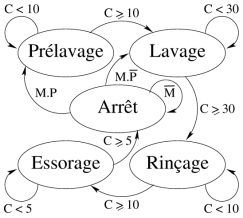
- Supposons que les entrées sont : M(arche) ou arrêt, P(rélavage) ou pas, C(hronomètre) en minutes raz à chaque étape

Machine à laver



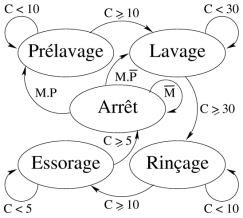
- Cela donne lieu aux conditions logiques associées à chaque transition

Machine à laver



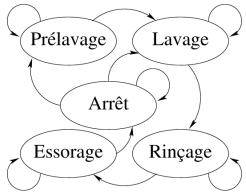
- Quand M est vrai au moment d'un front montant la machine change d'état

Machine à laver



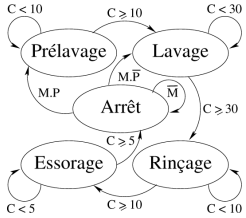
- Attention, la valeur des entrées n'a d'importance qu'au moment précis des fronts montants de l'horloge

Machine à laver



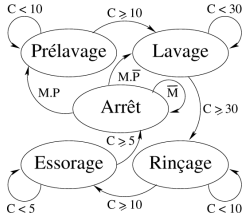
- Les durées des différentes étapes sont prélavage : 10 mn, lavage : 30 mn, rinçage : 10 mn et essorage : 5 mn

Machine à laver



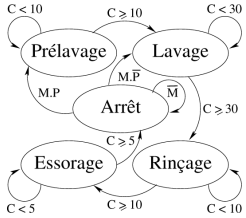
- e.g. une machine en Arrêt y reste tant que M est faux au moment d'un front montant de l'horloge

Machine à laver



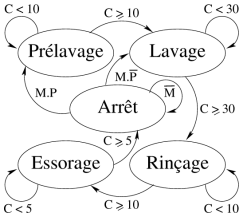
- Elle passe dans l'état Prélavage si P est vrai et dans l'état Lavage sinon

Machine à laver



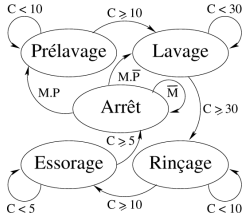
- C'est une conséquence du fait que notre machine est synchrone sur front montant

Machine à laver



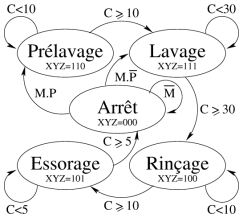
- 2 sortes de machines à états : a) sorties ne dépendent que de l'état courant (Moore) b) et des entrées (Mealy).

Machine à laver



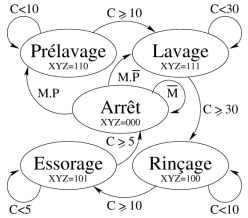
- Le programmateur de notre lave-linge est donc une machine de Moore dont les sorties ne dépendent que de l'état courant

Machine à laver



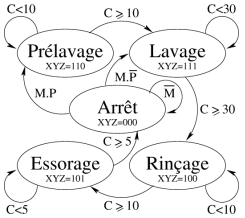
- Sorties : X, Y et Z destinés à piloter les différents moteurs du lave-linge

Machine à laver



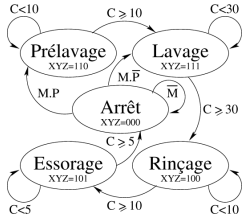
- Les spécifications précisent leur valeur pour chaque état que peut prendre la machine

Machine à laver



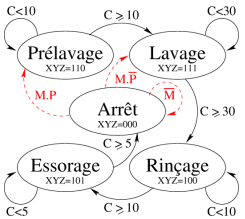
- La traduction des spécifications en graphe d'état est source d'erreur

Machine à laver



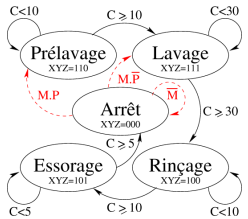
- Il faut donc vérifier le graphe avant de poursuivre la réalisation de la machine

Machine à laver



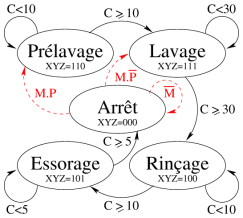
- Deux propriétés fondamentales

Machine à laver



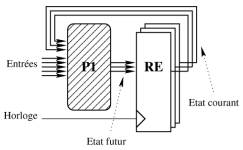
- 1) complet ou non ambigu : une seule transition possible compte tenu des entrées

Machine à laver



- 2) non contradictoire : toujours une transition valide

Du graphe à la machine à états

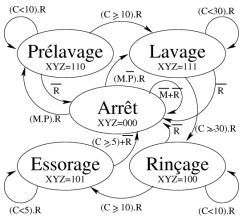


- Il existe donc, entre autres, un circuit combinatoire chargé de calculer le prochain état (i.e. état futur) à partir de l'état courant et des entrées de la machine.
- Ses entrées sont : 1) L'état courant (mémorisé dans le registre RE) et 2) les entrées
- Sa sortie est l'état futur.

Du graphe à la machine à états

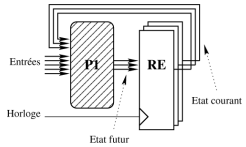
- Une machine à état est un dispositif avec rétroaction : l'état courant conditionne les états futurs.
- Dans un tel dispositif la question des conditions initiales se pose.
- En d'autres termes, pour que le fonctionnement soit celui souhaité dès la mise sous tension, il faut introduire un moyen de forcer un état de départ.
- Il en va de même pour le microprocesseur qui constitue l'unité de calcul de votre ordinateur.

Du graphe à la machine à états



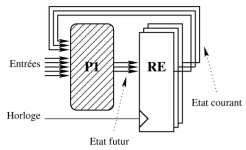
- Le reset synchrone ;

Du graphe à la machine à états



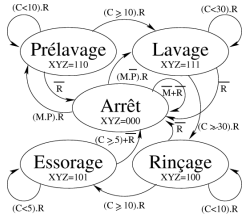
- En logique séquentielle synchrone, l'état courant est modifié à chaque front montant
- Entre deux fronts montants de l'horloge il reste stable, ce qui donne le temps aux circuits combinatoires qui composent la machine de calculer le prochain état et les sorties.

Du graphe à la machine à états



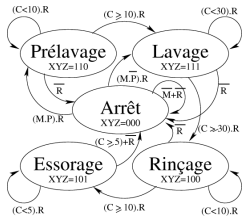
-
- Le registre d'état est composé de plusieurs bascules D synchronisées sur l'horloge
- Son entrée est l'état futur.
- Sa sortie, l'état courant, (qui sert d'entrée à P1 mais aussi au circuit destiné à calculer les sorties)

Du graphe à la machine à états



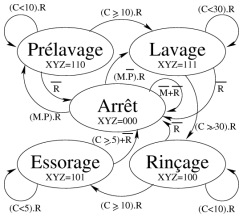
- Il existe deux méthodes pour forcer l'état initial avec le Reset :

Du graphe à la machine à états



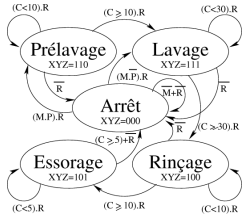
- Le reset synchrone ; pris en compte uniquement sur le front montant

Du graphe à la machine à états



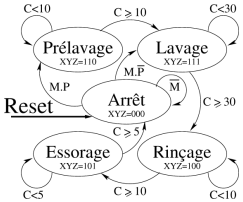
- Le reset synchrone ; Une nouvelle entrée de la machine mais prioritaire sur les autres

Du graphe à la machine à états



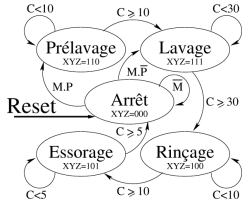
- Le reset synchrone ; Lorsque cette entrée est active l'état futur que calcule P1 est l'état initial

Du graphe à la machine à états



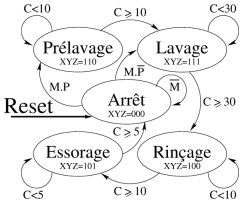
- Le reset asynchrone ;

Du graphe à la machine à états



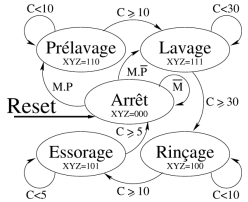
- Le reset asynchrone ; utilise les entrées Set et Reset des bascules D pour forcer l'état initial

Du graphe à la machine à états



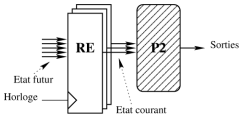
- Le reset asynchrone ; Les entrées de P1 et le graphe d'état ne sont pas modifiés

Du graphe à la machine à états



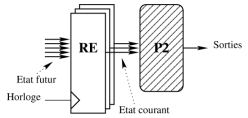
- Le reset asynchrone ; Solution plus simple à préférer

Du graphe à la machine à états



- Le circuit combinatoire de calcul des sorties prend en entrées l'état courant et fournit les sorties de la machine.
- Au changement d'état, sur front montant, ce circuit commence à calculer les sorties caractéristiques du nouvel état

Du graphe à la machine à états



- Pour pouvoir mémoriser n'importe quel état dans les bascules D du circuit le nombre de bascules doit être au moins égal à la taille du nom le plus long
- Le choix du codage n'est pas trivial
- On considèrera par facilité le nombre minimal de bascules D

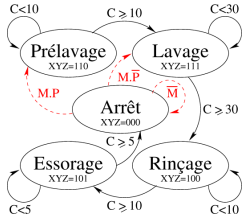
La machine à états de la machien à laver

- Etape 1 : lister les entrées

Nom	Mode	Description
H	Entrée	Horloge
R	Entrée	Reset actif à 0 , initialise à l'état Arrêt
M	Entrée	Position du bouton Marche/Arrêt
P	Entrée	Existence d'une phase de prélavage
C5	Entrée	Chronomètre supérieur ou égal à 5 minutes
C10	Entrée	Chronomètre supérieur ou égal à 10 minutes
C30	Entrée	Chronomètre supérieur ou égal à 30 minutes
X	Sortie	Vaut 0 dans l'état Arrêt, 1 dans les autres
Y	Sortie	Vaut 1 dans les états Prélavage et Lavage, 0 dans les autres
Z	Sortie	Vaut 1 dans les états Lavage et Essorage, 0 dans les autres

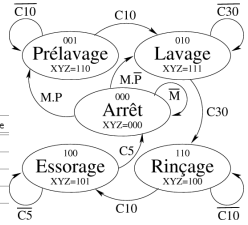
La machine à états de la machien à laver

- Etape 2 : graphe d'état (non ambigu ni contradictoire)



La machine à états de la machien à laver

- Etape 3 : coder les états



Etat	Codage
Arrêt	000
Prélavage	001
Lavage	010
Rinçage	110
Essorage	100

La machine à états de la machien à laver

- Etape 4 : calculer les circuits combinatoires

Etat courant			Entrées					Etat futur		
EC2	EC1	EC0	M	P	C5	C10	C30	EF2	EF1	EF0
0	0	0	0	X	X	X	X	0	0	0
0	0	0	1	1	X	X	X	0	0	1
0	0	0	1	0	X	X	X	0	1	0
0	0	1	X	X	X	0	X	0	0	1
0	0	1	X	X	X	1	X	0	1	0
0	1	0	X	X	X	X	0	0	1	0
0	1	0	X	X	X	X	1	1	1	0
1	1	0	X	X	X	0	X	1	1	0
1	1	0	X	X	X	1	X	1	0	0
1	0	0	X	X	0	X	X	1	0	0
1	0	0	X	X	1	X	X	0	0	0

- On détermine et simplifie les équations de EF2, EF1 et EF0

La machine à états de la machien à laver

- Etape 5 : On calcul l'éventuel circuits combinatoires des sorties