Nom : Kany Daphné Sujet : 23 Jury : Page : 1/3

Ne rien écrire hors du cadre.

Legon 23 : Fonctions et circuits booléens en architecture des ordinateurs

Niveau : MPII/MPI ; 1ère

I - Cadres théorique (prépa)

I.1 - Expression booléenne

Def 1 : On définit l'ensemble EB des expressions booléennes par induction avec la signature :
- car de base : $\bot, \top, V$ un ensemble de variable
- constructeurs : $\cdot \neg$ un constructeur unaire
  $\cdot \wedge, \vee$: constructeurs binaires

Informellement : $\bot, \top$, et V sont des EB, et si e_1 et e_2 le sont, alors $\neg e_1$, $e_1 \wedge e_2$, $e_1 \vee e_2$ le sont.

Exercice 2 : Définir inductivement les variables apparaissant dans une formule et en déduire une définition rigoureuse du nombre de variables d'une formule.

Def 3 : Une valuation est une fonction $\sigma : V \to \{0,1\}$. On définit $[\,]_\sigma$: EB $\to \{0,1\}$ par induction sur EB par :
- $[\bot]_\sigma = 0$, $[\top]_\sigma = 1$, $[x]_\sigma = \sigma(x)$
- $[e_1 \wedge e_2]_\sigma = \max([e_1]_\sigma, [e_2]_\sigma)$
- $[e_1 \vee e_2]_\sigma = \min([e_1]_\sigma, [e_2]_\sigma)$
- $[\neg e_1]_\sigma = 1 - [e_1]_\sigma$

Def 4 : Soit $a, b \in$ EB, on dit que a et b sont équivalentes, noté $a \equiv b$, si $\forall \sigma : V \to \{0,1\}$, $[a]_\sigma = [b]_\sigma$

Def 5 : On peut éventuellement définir le XOR, avec $a \oplus b = (a \vee b) \wedge \neg(a \wedge b)$

I.2 - Fonctions booléennes

Def 6 : Une fonction booléenne est une fonction $f : \{0,1\}^n \to \{0,1\}^m$.

Exemple 7 : $f : \{0,1\}^{64} \to \{0,1\}^{64}$ qui prend en entrée le codage binaire (double) d'un flottant x, et renvoie le codage binaire des flottant e^x approximé.

Remarque 8 : On peut se limiter à $f : \{0,1\}^n \to \{0,1\}$ (en décomposant $f : \{0,1\}^n \to \{0,1\}^m$ en m fonctions sur chaque dimension).

Def 9 : (table de vérité) : La table de vérité d'une fonction booléenne est la donnée de sa valeur sur toutes ses entrées possibles. Pour $f : \{0,1\}^n \to \{0,1\}^m$, on la représente par un tableau à $n+m$ colonnes, où chaque ligne représente la valeur d'un n-uplet d'entrée et du m-uplet de sortie associé (rq : la table de vérité aura 2^n lignes).

Remarque 10 : On peut définir une fonction booléenne :
- par intention (en donnant ses propriétés : ex : 1 ssi le nombre de variable à 1 est premier)
- par extension, en donnant sa table de vérité

Théorème M : Toute fonction $f : \{0,1\}^n \to \{0,1\}$ est équivalente à une formule propositionnelle. Plus rigoureusement, il existe une formule e dont les variables sont $x_1, \dots, x_n$ et tq $\forall \sigma : V \to \{0,1\}$, $[e]_\sigma = f(\sigma(x_1), \dots, \sigma(x_n))$.

Développement 1 : Preuve du théorème M et discussion sur la complexité.

I.3 - Représentation des expressions booléennes par des graphes orientés acycliques

Def 12 : (Circuit booléen) Un circuit est un DAG étiqueté par $V \cup \{\vee, \wedge, \neg, \top, \bot\}$ où un sommet d'étiquette x a :
- un degré entrant 0 si $x \in V \cup \{\top, \bot\}$
- un degré entrant 1 si $x = \neg$
- un degré entrant 2 si $x = \vee$ ou $x = \wedge$

Remarque 13 : On autorise souvent des étiquettes supplémentaires (XOR, NAND, NOR)

Def 14 : On définit l'évaluation d'un graphe par $\sigma : V \to \{0,1\}$ comme un étiquetage des nœuds où un nœud a vaudra :

* $[a]_\sigma$ si $a \in V \cup \{\top, \bot\}$
* $1-y$ si $a = \neg$ et le sommet entrant de a est valué en y
* $\min(y,z)$ (resp $\max(y,z)$) si $a = \wedge$ (resp $\vee$) et les sommets entrant de a sont valués en y et z.

Propriété 15 : Cette définition est correcte (et ne boucle pas à l'infini)

Remarque 16 :

* Les degrés sortants ne sont pas limités.
* Si les degrés sortants valent tous 1 sauf pour un sommet où c'est 0, on obtient exactement les expressions booléennes (c'est leur représentation sous forme d'arbre)
* Les circuits booléens correspondraient à un ensemble d'EB où l'on aurait le droit de définir des alias.

Conclusion 17 : Les circuits booléens peuvent représenter tout ce qu'on veut calculer de fini et qu'on peut coder en binaire.

II - Dans un ordinateur

II.1 - Circuits booléens

Def 18 : Un bit est la plus petite unité d'information d'un ordinateur. Il peut prendre deux valeurs (0 ou 1, 0V ou 5V, ...)

Def 19 : (porte logique) Une porte logique est un circuit électronique réalisant des opérations logiques sur une séquence de bits.

Exemple 20 : On est capable de faire les portes ET, OU, NON

Def 21 : Un circuit combinatoire est alors une succession de portes logiques dont la sortie de certaines est branchée sur l'entrée d'autres, le tout sans cycle.

Remarque 22 : C'est l'implémentation physique des circuits booléens.

Conclusion 23 : On est capable électroniquement de calculer tout ce qui est représentable de manière finie par des bits.

II.2 - Mesure d'un circuit

Motivation 24 : Quand on construit un circuit électronique, on a plusieurs impératifs à respecter :

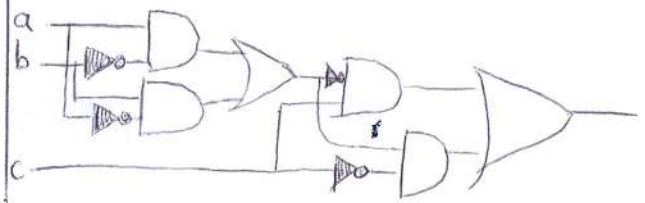
* On veut, pour des raisons économiques, minimiser le nombre de transistors (et donc de portes)
* On veut tirer des câbles courts, pour avoir un circuit compact

Il y a un délai de propagation entre le moment où l'entrée d'une porte change et le moment où sa sortie change.

Def 25 : (chemin critique)
Le chemin critique d'un circuit booléen (et donc d'un circuit combinatoire) est le plus long chemin entre une entrée (degré entrant 0) et une sortie (degré sortant 0). Puisque le graphe est acyclique, c'est un plus long chemin.

Objectif 26 : Minimiser la longueur d'un chemin critique, qui est proportionnelle au délai que met un circuit combinatoire à effectuer un calcul.

Exemple : $a \barwedge b \barwedge c$



II.3 - Des circuits particuliers

II.3.1 - Additionneur n bits

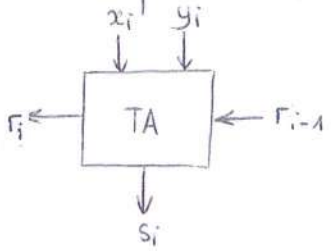
Algorithme 27 : Addition (x, y) : # x et y sont deux nombres de # n chiffres en binaire

$r_{-1} = 0$
for $i = 0$ to n :
  $r_i, s_i = TA\ (r_{i-1}, x_i, y_i)$ # r_i : retenu ; s_i : $i$ème bit de la somme

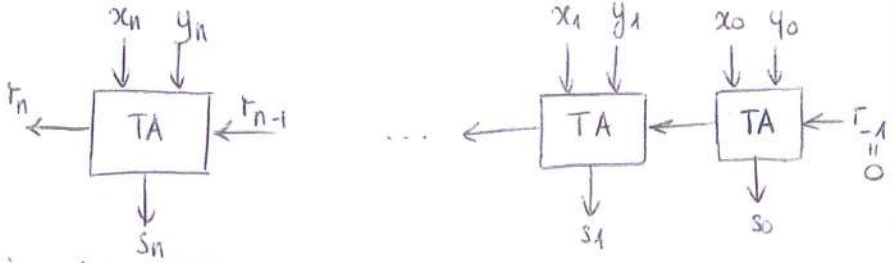
avec TA (table d'addition) définie par la table de vérité suivante :

r_{i-1}	x_i	y_i	r_i	s_i
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

Exercice 28 : Proposer un circuit booléen pour cette table.



En combinant les tables d'addition, on peut créer un additionneur n bits :



Inconvénient 28 : Chemin critique de taille n. (propagation de la retenue)

Développement 2 : Construction d'un additionneur n bits à retenues anticipées

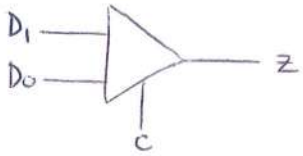
II.3.2 - Multiplexeur

Def 29 : Un multiplexeur à deux entrées sert à sélectionner une entrée parmi deux, grâce à une entrée de contrôle.

Table de vérité d'un multiplexeur :

D_1	D_0	c	z
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

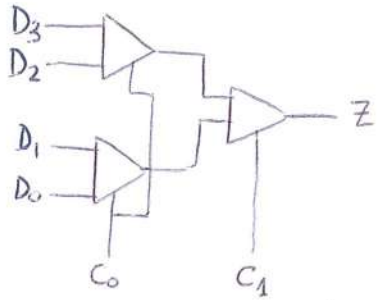
$z = (c \wedge D_1) \vee (\neg c \wedge D_0)$



Exercice 30 : Construire un circuit booléen pour le multiplexeur à deux entrées.

Remarque 31 : En combinant les multiplexeurs à deux entrées, on peut générer des multiplexeurs à 2^k entrées.

Exemple 32 : Multiplexeur à 4 entrées :



Remarque 33 : Si on interprète C_1C_0 comme un nombre binaire, sa valeur correspond à l'entrée sélectionnée.