

Module CAO

Cours du Professeur A.Greiner

Mai 2008

2 heures sans documents

Réduction des expressions Booléennes Multi-niveaux

On considère des expressions Booléennes multi-niveaux comportant un nombre quelconque de niveaux de parenthésage, n'utilisant que les trois opérateurs Booléens OR, AND et NOT. L'arité (nombre d'opérandes) des opérateurs Booléens AND et OR est quelconque.

Chaque expression Booléenne est représentée par un arbre binaire de type ABL (structure de données présentée en cours, et utilisée en TP).

On dit qu'une expression Booléenne est réduite lorsque les seuls opérateurs NOT apparaissant dans l'expression agissent sur une variable booléenne, mais pas sur une sous-expression. On obtient une expression réduite par application successive des lois de Morgan.

On cherche dans un premier temps à écrire en langage C les deux fonctions récursives qui effectuent la réduction d'une expression Booléenne E :

```
bip_t *red(bip_t *p)
bip_t *red_not(bip_t *p)
```

La fonction red() prend pour argument un pointeur p sur un ABL représentant une expression Booléenne quelconque E, et renvoie un pointeur sur un ABL représentant une autre expression Booléenne red(E), équivalente à E, mais réduite. La fonction red_not() prend pour argument un pointeur p sur un ABL représentant une expression Booléenne quelconque E, complémente cette expression, et renvoie un pointeur sur un ABL représentant une autre expression Booléenne red_not(E), équivalente à NOT(E). Autrement dit, red_not() effectue une complémentation avant d'effectuer la réduction.

Q1) Représenter graphiquement les deux arbres ABL représentant l'expression Booléenne $E = \text{OR}(a \text{ NOT}(\text{OR}(b \text{ NOT}(c) \text{ AND}(d, e))))$ et l'expression Booléenne réduite red(E). On veillera à soigner la mise en page, pour rendre ces graphes lisibles...

Q2) Ecrire – en Français - l'algorithme de la fonction récursive red(E). Lorsque l'expression E n'est pas une simple variable, on précisera la relation qui existe entre l'expression Booléenne réduite red(E), et les expressions Booléennes réduites associées à chacun des opérandes de E. On étudiera successivement les trois cas correspondant aux trois opérateurs OR, AND, NOT. On traitera également le cas terminal où E est une simple variable. Même question pour la fonction récursive red_not(E).

Q3) Ecrire en langage C les deux fonctions récursives red() et red_not().

On utilisera la fonction bip_t *cons_bip(bip_t *next, void *data) qui construit une bi-pointeur, et initialise les deux champs NEXT et DATA. L'arbre ABL représentant l'expression E ne doit pas être modifié pour construire l'ABL de l'expression réduite, et tous les bi-pointeurs du nouvel ABL doivent donc être créés avec la fonction cons_bip().

Représentation des réseaux Booléens

On sait qu'une description comportementale VHDL utilisant des assignations concurrentes (style d'écriture « data-flow ») représente un réseau Booléen. On ne considère dans cet exercice que des circuits purement combinatoires, où tous les signaux sont des bits (pas de vecteurs de bits). On se propose de définir une structure de donnée générale permettant de représenter n'importe quel réseau Booléen respectant les restrictions ci-dessus.

Q4) Soit la description comportementale suivante, où les mots-clés du langage VHDL sont en majuscules, et les noms de variables en minuscules.

```
ENTITY encodeur IS
PORT (
    e0 : in BIT;
    e1 : in BIT;
    e2 : in BIT;
    e3 : in BIT;
    s0 : out BIT;
    s1 : out BIT;
    s2 : out BIT;
    s3 : out BIT;
);
END encodeur;
ARCHITECTURE vbe OF encodeur IS
    SIGNAL ne1 : BIT;
    SIGNAL ne2 : BIT;
    SIGNAL ne3 : BIT;
BEGIN
    ne3 <= NOT( e3 );
    ne2 <= NOT( e2 );
    ne1 <= NOT( e1 );
    s3 <= e3;
    s2 <= e2 AND ne3 ;
    s1 <= e1 AND ne2 AND ne3 ;
    s0 <= e0 AND ne1 AND ne2 AND ne3 ;
END;
```

Dessiner le graphe biparti représentant le réseau Booléen associé au circuit encodeur ci-dessus, comportant 4 signaux d'entrée, 4 signaux de sortie, 3 signaux intermédiaires, et 7 assignations concurrentes (correspondant à 7 processus s'exécutant en parallèle). On représentera les signaux par des carrés, et les processus par des ronds.

Q5) Proposez une structure de données générale permettant de représenter n'importe quel réseau Booléen. On définira les structures de données décrivant les trois objets « boolnet_t », « signal_t » et « processus_t », ainsi que les relations entre ces objets. Cette structure de données doit permettre d'identifier facilement le type des signaux (entrée, sortie, interne). Il faut pouvoir parcourir facilement le graphe dans les deux sens (des entrées vers les sorties, et des sorties vers les entrées). L'expression Booléenne associée à une assignation concurrente sera représentée par un ABL, dans lequel les variables Booléennes sont définies par un pointeur sur un objet « signal_t ». Attention : contrairement aux structures de données utilisées dans le TME7, on utilisera des bipointeurs (objet bip_t) pour toutes les listes chaînées représentant des ensembles d'objets de même type. Représenter graphiquement la structure de données en mémoire dans le cas particulier du réseau Booléen correspondant au circuit « tiny » ci-après :

```

ENTITY tiny IS
PORT (
    e0 : in BIT;
    e1 : in BIT;
    s  : out BIT
);
END tiny;
ARCHITECTURE vbe OF tiny IS
BEGIN
    s <= e1 AND e0 ;
END;

```

Q6) Rappeler pourquoi le réseau Booléen décrivant un circuit combinatoire doit être acyclique.

Puisque ce graphe est acyclique, il est possible d'attacher à chaque signal S du réseau (S peut être une entrée, une sortie, ou un signal interne) un attribut $\text{Prof}(S)$ représentant la distance maximale entre ce signal S et une entrée E_i du réseau. Si $\text{Distance}(X,Y)$ mesure le nombre de processus rencontrés sur le chemin le plus long entre les signaux X et Y , on a donc $\text{Prof}(S) = \text{Max} [\text{Distance}(E_i, S)]$. Le Max porte sur toutes les entrées E_i .

On a évidemment $\text{Prof}(E_i) = 0$ pour tous les signaux d'entrée E_i , et les valeurs $\text{Prof}(S)$ augmentent au fur et à mesure qu'on s'éloigne des entrées pour se rapprocher des sorties.

Proposez – en Français - un algorithme permettant de calculer les valeurs $\text{Prof}(S)$ pour tous les signaux du réseau Booléen.

Attention : le réseau Booléen est acyclique, mais ce n'est pas un arbre, et il peut exister plusieurs chemins entre deux signaux X et Y .

Q7) Question subsidiaire : On souhaite écrire un simulateur logique « zero delay » pour ce type particulier de réseau Booléen combinatoire. Sans entrer dans les détails expliquez comment on peut exploiter le caractère acyclique du graphe et l'attribut $\text{Prof}(S)$ pour mettre en œuvre une technique d'ordonnancement « statique ». Dans un simulateur à ordonnancement statique, on supprime l'échéancier global utilisé par l'algorithme de simulation « event-driven », et on remplace l'ordre d'évaluation des processus, calculé dynamiquement par l'échéancier au moment de l'exécution par un ordre d'évaluation statique, calculé une fois pour toutes avant le démarrage de la simulation.