BRUCE ROSE 3700694
ELIES BENMANSOUR 3504377

Projet : Problème de Via Minimization

Rapport de Bruce Rose et Elies Benmansour

Table des matières

Introduction

Dans le cadre du cours d'Algorithmique appliquée et structures de données (2I006) de la faculté Pierre et Marie Curie, nous avons été amené à concevoir un programme de résolution de problème dit de Via Minimization. Ce rapport présente nos travaux.

1 Description générale du code

1.1 Organisation des dossiers

Le dossier du projet est constitué :

- d'un dossier PartieA/
- d'un dossier doc/ contenant ce rapport ;
- d'un dossier src/ contenant les fichiers .c en rapport avec la Partie B ;
- un dossier bin/ contenant les fichiers .o ;
- un Makefile
- un fichier CHANGELOG.txt décrivant les évolutions majeures du projet (à lire avec gedit, dans un environnement Linux) ;
- ...

Le dossier PartieA/ contient tous les travaux en rapport avec la Partie A du projet, c-à-d :

- un dossier src/ contenant tous les fichiers .c en rapport avec la Partie A;
- un dossier bin/ contenant les fichiers .o ;
- un dossier nets/ contenant des instances de Netlist ;
- un dossier doc/ contenant le rapport intermédiaire de la partie A ;
- un Makefile ;
- …

1.2 Partie A

Un ensemble de méthodes permettant la manipulation de Netlists se trouve dans le fichier PartieA/src/Netlist.c. Elles permettent (entre autres) de :

- Manipuler une liste de Segments ;
- Manipuler une liste de Points ;
- Créer une Netlist ;
- Créer un Reseau ;
- Créer un Point ;
- Créer un Segment ;
- Déterminer la direction d'un segment étant donnés ses deux points ;
- Compter le nombre de Segments et de Points dans un Reseau ;
- Sauvegarder une instance de Netlist sur un fichier .net ;
- Charger une instance de Netlist depuis un fichier .net.

Dans le fichier PartieA/src/intersec.c se trouvent diverses méthodes permettant de :

- Déterminer si deux segments donnés s'intersectent ;
- Construire un tableau des segments d'une instance de Netlist ;
- Construire un tableau des points d'une instance de Netlist ;
- Créer par une approche naïve les listes d'intersection des segments d'une instance de Netlist donnée ;
- Afficher les segments d'un tableau de segments ;
- Sauvegarder les intersections d'une instance de Netlist sur un fichier .int.

Dans le fichier PartieA/src/balayage.c se trouvent toutes les méthodes permettant de déterminer les intersections des segments d'une instance de Netlist en utilisant la technique du balayage.

Pour rappel, voici ce que l'énoncé indique pour la méthode par balayage :

« *L'idée est la suivante : on considère une droite de balayage verticale imaginaire qui se déplace à travers l'ensemble de segments, de la gauche vers la droite. À une abscisse x donnée de son balayage, notons* k_x *le nombre de segments horizontaux coupant la droite. On considère aussi l'ensemble T de taille* k_x *de tous ces segments horizontaux, triées par leurs ordonnées. Un segment vertical d'abscisse x ne pourra alors intersecter que les segments de T. Pour mettre en œuvre ce balayage, on considère l'échéancier des points d'évènement E qui est définie comme une séquence d'abscisses, triées de la gauche vers la droite, qui vont définir les positions d'arrêt de la droite de balayage. Dans notre cas, les points d'évènements seront les extrémités gauche et droite des segments horizontaux et les abscisses des segments verticaux. Attention : dans ce tri, pour deux points de même abscisse, un point gauche de segment [horizontal] est avant un segment vertical et un segment vertical est avant un point droit de segment horizontal : en effet, l'évènement menant à la recherche d'intersection est le fait de balayer un segment vertical et il ne faut pas qu'à ce moment, les segments horizontaux pouvant le croiser soit hors de la structure T.* »

L'énoncé nous demande de proposer une structure de données pour manipuler des pointeurs sur Extremite, qui correspondra à un échéancier E. Pour ce faire, nous avons choisi d'implémenter une table de hachage sur listes. La table de hachage a pour taille le nombre d'abscisses, multiplié par 3. Ceci permet de stocker dans les trois « tiers » d'abscisse les extrémités gauches, les segments verticaux et enfin les extrémités droites, dans cet ordre. De cette façon, un parcours du tableau permet d'obtenir directement les extrémités dans le bon ordre. Quand un nouvel élément est inséré dans l'échéancier, il est positionné dans la case dont l'indice est l'abscisse x de l'élément additionnée à l'attribut GouVouD[1] qui vaut 0 si l'élément est une extrémité gauche, 1 si c'est un segment vertical et 2 si c'est une extrémité droite. Il est alors rangé dans la liste à cette case en fonction de son ordonnée y. Remarquons qu'il n'y a pas de gâchis de place, puisque chaque case de la table de hachage est une liste. Quand la liste est vide, elle vaut tout simplement NULL, et donc ne prend pas de place mémoire. Dans un fichier PartieA/src/hachage.c sont définies quelques méthodes de manipulation de tables de hachage. La structure de la table est dans PartieA/src/hachage.h.

1 Anciennement VouGouD, il nous a semblé plus pratique au vu de notre structure de changer cet attribut en GouVouD, pour garder l'ordre correct.

1.3 Partie B

Le fichier src/graphe.c contient les fonctions qui permettent de créer un graphe à partir d'un fichier d'instance de Netlist .net et d'un fichier d'intersection .int. La structure du graphe est définie dans src/graphe.h.

Nous avons décidé d'utiliser le format Postscript pour générer une visualisation du graphe. Le fichier src/generatePostScript.c permet ceci. On notera qu'il est nécessaire de faire une homothétie de la figure pour pouvoir l'afficher dans une page de format A4. Dans ce fichier apparaît également une fonction d'affichage de la solution au problème de Via Minimization. En bleu sont représentés les segments présents sur la face 1, en rouge ceux présents sur la face 2. Seuls les points qui sont des vias sont représentés par des points noirs.

Dans le fichier src/viaMinimization.c se trouvent les méthodes de résolution du problème de Via Minimization. Notamment, s'y trouvent un méthode de résolution horizontale/verticale ainsi qu'une autre qui s'appuie sur les cycles impairs formés lors de conflits dans le graphe.

1.4 Autres

Le fichier PartieA/src/tools.c contient des fonctions pratiques. Celles-ci permettent de :

- Extraire le nom d'un fichier sans son extension ;
- Dire si une chaîne de caractère donnée se trouve dans un fichier en tant que ligne ;
- Donner le nombre de valeurs égales à 0 dans un tableau d'entier.

Le fichier PartieA/src/debug.h propose des définitions de macros utiles pour déboguer un programme. Elles permettent d'afficher une ligne, le nom de la fonction courante, le nom du fichier courant. Elles permettent également d'afficher les valeurs d'entiers. Vous remarquerez probablement apparaître de multiples fois dans nos fichiers des mots comme DEBUG ou LINE. Il s'agit des macros définies dans debug.h. Pour voir les affichages de ces macros, il faut préalablement remplacer DEBUG_MODE_NO par DEBUG_MODE_YES au début du fichier.

2 Jeux d'essais

3 Analyse de performances