Réalisation du simulateur de netlists

Baptiste Lefebvre, Li-yao Xia, Antonin Delpeuch

22 octobre 2012

Viendez écrire ! C'est fun ! Surtout avec LaTeX.

1 Algorithme

Nous avons implémenté l'algorithme suggéré au cours du TP1. La netlist représentée sous forme d'arbre de syntaxe abstraite définit un graphe des dépendances : un arc joint a à b si connaître la valeur de a est nécessaire pour calculer celle de b. On réalise un tri topologique pour déterminer dans quel ordre il faut calculer les valeurs des variables de sorte qu'on n'ait besoin que de valeurs déjà calculées à chaque étape.

Dans les exemples suivants, on a représenté les variables d'entrée dans le graphe, en pratique elles n'ont pas besoin d'y figurer.

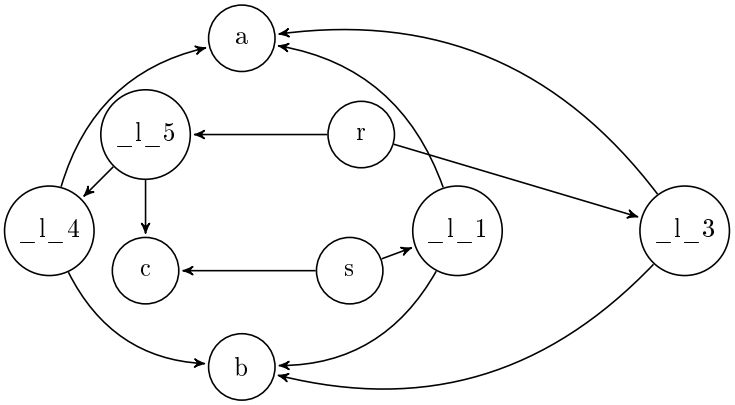

FIGURE 1 – Le graphe obtenu à partir de la netlist `fulladder.net`

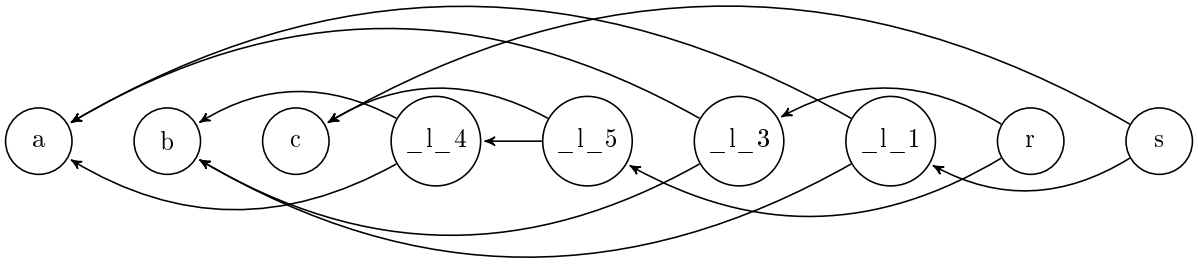

FIGURE 2 – Le même graphe après tri topologique

2 Implémentation

Nous avons réutilisé le parser et le lexer fournis dans le TP1, ainsi que les types définis dans `netlist_ast.ml`. Nous avons écrit le reste, en séparant le code (dans la mesure du possible) en modules indépendants :

- `Graph` : représentation des graphes et tri topologique ;
- `Netlist_proxy` : traduction des noms de variables en entiers, extraction des dépendances ;
- `Tape` : gestion des opérations `RAM` et `ROM` ;
- `Scheduler` : jointure entre `Graph` et `Netlist_proxy`.

3 Problèmes rencontrés

- Ceci
- Cela

4 Améliorations possibles

- Ajouter le support des opérations RAM et ROM ;
- Ajouter un mode de traitement par lot (pour calculer d'un seul coup les sorties après n cycles sur un jeu d'entrées) ;
- Mieux séparer les fonctions dans des modules différents ;
- ...

5 Autre chose ?

ouais, et toi ?