Optimisation de l'utilisation de la mémoire pour un simulateur de circuits

Timothée Bossart, Alix Munier et Francis Sourd

LIP6 Université Pierre et Marie Curie
{timothee.bossart, alix.munier, francis.sourd}@lip6.fr

Mots-clefs : optimisation combinatoire, graphes, simulation

La simulation est un enjeu crucial pour l'élaboration de circuits intégrés. Il est en effet dans ce cadre nécessaire d'itérer un processus de développement et de simulation avant de passer à la réalisation matérielle pour des tests en conditions réelles. Un circuit intégré (figure 1) est composé d'une part d'un ensemble de portes combinatoires reliées entre elles selon un graphe orienté G sans circuit (figure 2), et d'autre part d'un ensemble de registres qui permettent de stocker les résultats des calculs entre deux itérations de la simulation.

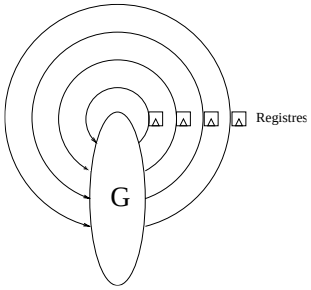
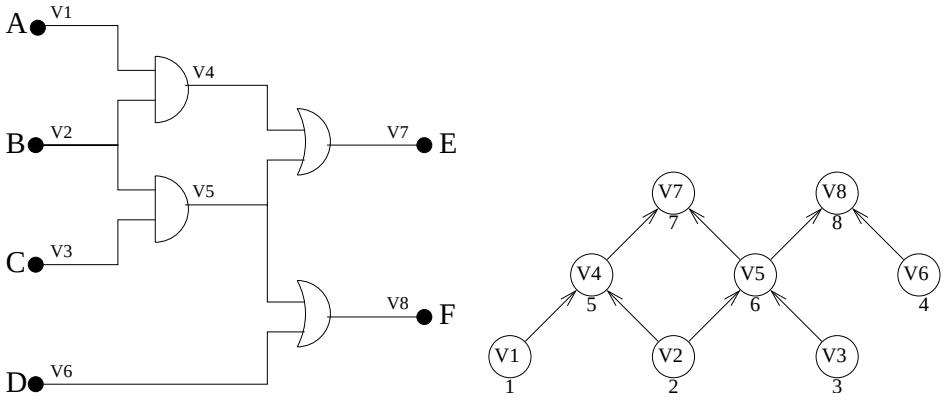


FIG. 1 – Vision abstraite d'un circuit

Un simulateur est un programme qui consiste à activer toutes les portes combinatoires d'un circuit en respectant l'ordre induit par G. Un exemple est donné en figure 2.



(a) Exemple de circuit

(b) Exemple de numérotation du graphe de précédence G du circuit

FIG. 2 – Exemple

Expérimentalement, il a été observé que la compilation du simulateur forme un important goulot d'étranglement en terme de gestion de la mémoire. En effet, celui-ci possède une structure

$$
\begin{array}{rcl}
V_1 & \longleftarrow & A \\
V_2 & \longleftarrow & B \\
V_3 & \longleftarrow & C \\
V_6 & \longleftarrow & D \\
V_4 & \longleftarrow & \mathrm{AND}(V_1, V_2) \\
V_5 & \longleftarrow & \mathrm{AND}(V_2, V_3) \\
V_7 & \longleftarrow & \mathrm{OR}(V_4, V_5) \\
V_8 & \longleftarrow & \mathrm{OR}(V_5, V_6) \\
E & \longleftarrow & V_7 \\
F & \longleftarrow & V_8
\end{array}
$$

TAB. 1 – Simulateur pour le circuit de la figure 2

et une taille qui ne se prêtent pas aux heuristiques classiquement utilisées par les compilateurs standards.

Un des objectifs de ce projet consiste donc à renuméroter les sommets du graphe G de sorte à minimiser une fonction de coût qui prend en compte les accès mémoire réalisés par chaque instruction du simulateur.

Nous nous proposons dans un premier temps de présenter en détails un modèle simple de mémoire pour un programme donné. Les hypothèses de celui-ci sont les suivantes :

- La mémoire est organisée en pile,
- Au départ, tous les nœuds sans prédécesseur correspondent à des valeurs stockées en mémoire,
- On dispose de 2 types d'instructions :
  - *LD* V_i déplace la valeur V_i en haut de la pile,
  - *OP* (V_i, V_j, θ) applique l'opérateur θ aux valeurs V_i et V_j et place le résultat en haut de la pile. Remarque : on suppose que V_i et V_j ont été préalablement déplacées en haut de la pile, et qu'à la fin de l'opération, le résultat est au sommet.

Nous présenterons dans un second temps les résultats obtenus pour deux fonctions de coût :

- On considère que les n premières valeurs de la pile sont dans des registres, et on ne compte que les autres accès (qu'on peut considérer comme des accès au cache de niveau 1 par exemple)[1]. Le cas $n = 1$ est *NP-complet* [BS76],
- On peut aussi pénaliser chaque accès mémoire en lui donnant un coût égal à la hauteur de la valeur souhaitée dans la pile.

Références

[BS76] J. Bruno and R. Sethi. Code generation for a one-register machine. *J. Assoc. Comp. Mach.*, 23(3) :502–510, July 1976. ctr127.

[1] Exemple : Si $n = 2$, la numérotation de la figure 2 donne 12 accès au cache, alors qu'en retardant au maximum le chargement de V_6, on peut réduire ce nombre à 9.