Compilation de graphes représentant des circuits électroniques en formules de logiques pour la vérification de propriété

Mots-clefs : Sémantiques de Circuits Intégrés; Formules Logiques; Solveurs SMT; Solveurs SAT; Analyse de Performance; `OCaml`

Contexte

La fabrication d'un circuit électronique est composé de nombreuses étapes. Vous avez probablement vu en cours d'architecture la partie numérique du flot de conception, où la fonctionnalité du circuit est décrite essentiellement avec des 1 et 0. La description logique (sous forme de portes ou de programme dans des langages comme VHDL) est ensuite transformée en une implémentation utilisant des transistors, en ajoutant des notions qui ne sont pas visibles au niveau logique comme le fait que différentes parties du circuit peuvent être alimentées par des tensions différentes pour économiser l'énergie. Le passage du niveau logique au niveau transistor est essentiellement automatisé, mais certaines parties peuvent être réalisées par l'utilisateur, et donc sujettes à erreurs. Il est ainsi important d'être capable de vérifier l'absence d'erreur sur une description de circuit au niveau transistor. L'équipe CASH du laboratoire LIP travaille avec la start-up Aniah sur le sujet.

L'approche globale de vérification de circuit est une combinaison de méthodes rapides, mais qui peuvent lever un grand nombre de fausses alarmes (autrement dit, lever un avertissement là où il n'y a pas d'erreur), et de méthodes plus lentes et plus précises pour éliminer les fausses alarmes. L'équipe CASH s'intéresse surtout à cette seconde catégorie de méthodes, en modélisant le fonctionnement du circuit avec des formules de logique, et en vérifiant si ces formules de logiques sont satisfaisables avec des solveurs existants (notamment le solveur Z3, développé par Microsoft). La vérification de satisfaisabilité de formule est un problème NP-complet, mais les solveurs se comportent bien en pratique sur des formules de taille raisonnables. À l'heure actuelle, nous disposons d'un compilateur permettant de générer une formule de logique à partir d'une description de circuit, et nous pouvons l'utiliser pour prouver des propriétés. Pour l'instant, les formules de logique que nous utilisons mélangent des valeurs numériques et des booléens, et nous utilisons le solveur Z3 pour en vérifier la satisfaisabilité.

1 Objectifs du projet

L'objectif de ce projet est d'expérimenter des variantes de l'encodage existant dans notre outil. Des formules équivalentes sur le plan logique peuvent donner des résultats très différents en termes de performance au niveau du solveur. Notre compilateur effectue déjà certaines optimisations, mais il nous reste du travail :

- Certaines variantes n'ont pas encore été implémentées. Par exemple, lorsqu'un fil est connecté à plusieurs alimentations en même temps (situation de court-circuit), la valeur de tension associée à ce fil peut être n'importe quelle valeur comprise entre la plus petite et la plus grande des alimentations auxquelles il est connecté. Ceci peut être exprimé sous forme logique en calculant la plus petite valeur possible avec

une cascade de if/elseif/else, ou bien avec un ensemble de clauses exprimant des contraintes sur cette tension, et nous n'avons implémenté que la seconde.

- Dans certains cas, nous n'avons expérimenté que la version *a priori* optimisée, sans avoir essayé la version naive non-optimisée. Un travail expérimental plus approfondi est nécessaire pour vérifier que la version optimisée est effectivement meilleure que la version naive en pratique, et mesurer les gains offerts par cette optimisation.

Bien sûr, ce travail sera accompagné d'une bibliographie sur le sujet de la vérification utilisant les solveurs de formule logique (SAT et SMT en particulier).

Profil attendu

Le projet peut être réalisé seul(e) ou en binôme.

L'étudiant(e) devra avoir de bonnes notions de logique, et être intéressé(e) par l'algorithmique. Notre compilateur est implémenté dans le langage OCaml, donc une connaissance de ce langage serait un plus, mais il est aussi possible de réaliser les expérimentations en dehors de l'outil en utilisant n'importe quel autre langage.

Le projet peut prendre plusieurs orientations selon les goûts de l'étudiant : on peut l'axer sur la théorie, comprendre et éventuellement modifier la sémantique sur laquelle nous nous basons pour la compilation, prouver formellement qu'une optimisation est correcte ; ou bien au contraire l'axer sur la pratique, l'implémentation et l'analyse des résultats expérimentaux.

Une poursuite du travail en stage de master, voire en thèse, est possible si l'étudiant(e) est motivé(e) pour le faire.

Encadrement

- Matthieu Moy, maître de conférences UCBL/LIP, `https://matthieu-moy.fr/`,
- Bruno Ferres, post-doctorant au LIP, `https://perso.ens-lyon.fr/bruno.ferres`.