Projet de système digital

Rapport intermédiaire sur le simulateur

Florian Bourse, Joris Giovannangelli, Théo Zimmermann

Spécifications

Le langage de description de circuit que nous avons créé permet de définir des portes qui sont des combinaisons des opérateurs logiques élémentaires and, or, xor, not, nand et mux, des constantes True, False, et des registres reg. La définition des portes utilise des variables locales correspondant à des nœuds internes. Celle-ci a lieu au début du fichier entre les mots-clés def et end.

Les portes peuvent avoir un nombre d'entrées quelconque ; ces entrées sont utilisées de la même manière que les variables locales.

Les portes peuvent aussi avoir un nombre de sorties quelconque.

Les variables locales sont définies une unique fois et associées à une expression logique qui peut faire appel aux autres variables locales.

Les portes s'utilisent ensuite en les instanciant sous forme de blocs. Les entrées des blocs sont reliées aux sorties d'autres blocs.

Un bloc est défini en fournissant son type (le nom de la porte qui est instanciée), son nom et ses entrées qui correspondent aux sorties d'autres blocs (sauf pour le premier bloc dont les entrées sont sans importance).

Les portes, blocs et variables logiques à l'intérieur des définition des portes peuvent être déclarées dans un ordre quelconque à l'exception notable des premier et dernier blocs.

Le premier bloc reçoit nécessairement les entrées du circuit (il détermine donc leur nombre).

Le dernier bloc a nécessairement pour sorties celles du circuit.

La grammaire du langage est donc la suivante :

```
circuit = def
        porte
        porte
        …
        porte
        end
        bloc
        bloc
        …
        bloc
```

(nom) *(entrées)*
porte = <ident> ← (<ident>,<ident>,...,<ident>)
<ident> = expression ;
<ident> = expression ;
…
<ident> = expression ;
→ (<ident>,<ident>,...,<ident>)
(sorties)

bloc = <ident> <ident> ← (<ident>[int],<ident>[int],...,<ident>[int])
(type) *(nom)* *(bloc)* *(numéro de la sortie)*

Description du fonctionnement

Les circuits sont compilés dans un langage intermédiaire, LICS, qui fait des calculs de manière linéaire. Pendant la compilation, les circuits combinatoires déclenchent une erreur et les registres sont éclatés en deux morceaux correspondant à la sortie du registre pour le cycle courant et l'entrée qu'il reçoit, ce qui permet de trier topologiquement le graphe correspondant au circuit.

Les entrées sont assignées dans l'ordre à des variables, les sorties sont définies dans l'ordre.

La syntaxe de LICS est la suivante :

programme = instruction
instruction
…
instruction

instruction = <int> = expression
| <int> = intput
| <int> = output
| <int> = intputreg
| <int> = outputreg

expression = constante <bool>
| unaire <int>
| binaire <int> <int>
| ternaire <int> <int> <int>

unaire = Not

binaire = Or | And | Nand | Xor

ternaire = Mux

Tests

Les tests sont joints (format .scd), lisibles avec un éditeur de texte quelconque.