# TP2 : Synthèse logique et optimisation structurelle



# 1. Introduction

## 1.1 Synthèse logique

La synthèse logique permet d'obtenir une netlist de portes à partir d'un réseau booléen (format .vbe). Plusieurs outils sont disponibles :

- L'outil **BOOM** permet l'optimisation de réseau booléen avant synthèse.
- L'outil **BOOG** offre la possibilité de synthétiser une netlist en utilisant une bibliothèque de cellules précaractérisées telle que **SXLIB**.

  La netlist peut être soit au format **.vst** soit au format **.al**. Vérifier la variable d'environnement **MBK_OUT_LO=vst**.

Pour plus de renseignements sur ces outils, reportez vous au man.

## 1.2 Résolution des problèmes de fanout (sortance)

Les netlists générées contiennent parfois des signaux internes attaquant un nombre

important de portes (grand fanout). Ceci se traduit par une détérioration des fronts (rise time et fall time). Il y a alors une perte en performance temporelle. Afin de ré- soudre ces problèmes, l'outil **LOON** remplace les cellules ayant un fanout (i.e sortance) trop grand par des cellules plus puissantes ou bien insère des buffers.

## 1.3 Visualisation de la chaîne longue

A tout moment, les netlists peuvent être éditées graphiquement. L'outil **XSCH** permet de visualiser le chemin le plus long grâce aux fichiers **.xsc** et **.vst** générés à la fois par **BOOG** et par **LOON**.

La résistance équivalente R de la figure est calculée sur la totalité des transistors du AND appartenant au chemin actif. De même, la capacité C est calculée sur les transistors passants du NOR correspondant au chemin entre i0 et la sortie de la cellule.

## 1.4 Vérification de la netlist

La netlist doit être certifiée. Pour cela, on dispose du simulateur **ASIMUT**

# 2 Travail à effectuer

## 2.1 Optimisation du réseau booléen

Pour analyser l'effet de l'optimisation booléenne :

On utilise le travail fait le TP précédent sur les descriptions des automates

digicode.vbe

- Lancer l'optimisation booléenne avec l'outil **BOOM** en demandant une optimisation

en surface puis en délai ;

```
>boom -V <vbe_source> <vbe_destination> -
```

- Essayer **BOOM** avec les différents algorithmes **-s**, **-j**, **-b**, **-g**, **-p**... Le mode et le

niveau d'optimisation sont aussi à changer.

- Comparer le nombre de littéraux après factorisation.

## 2.2 Mapping sur cellules précaractérisées

Pour chacun des réseaux booléens obtenus précédemment :

- positionner les variables d'environnement ;
- synthétiser la vue structurelle :

```
>boog <vbe_source> -
```

- lancer **BOOG** sur les différentes netlists pour observer l'influence des options de

**SYF** et de **BOOM**.

- valider le travail de **BOOG** en resimulant avec **ASIMUT** les netlists obtenues

avec les vecteurs de test qui ont servi à valider le réseau booléen initial.

## 2.3 Visualisation de la netlist

La chaîne longue est décrite dans le fichier **.xsc** produit par **BOOG**. L'outil **XSCH**

l'utilisera pour colorer son chemin. Pour lancer l'éditeur graphique :

```
>xsch -I vst -l <vst_source> -
```

La couleur rouge désigne le chemin critique. Si vous utilisez l'option '-slide' qui permet d'afficher un ensemble de netlists, n'oubliez pas d'appuyer sur les touches '+' ou '-' pour éditer vos fichiers !

## 2.4 Optimisation de la netlist

Pour toutes les vues structurelles obtenues précédemment :

Lancer **LOON** avec la commande :

```
>loon <vst_source> <vst_destination> <lax_param> -
```

Effectuer une optimisation de fanout en modifiant le facteur de fanout dans le fichier d'option **.lax**. Imposer des valeurs de capacités sur les sorties.

## 2.5 Vérification de la netlist

À effectuer sur la meilleure (justifiez votre choix) de vos netlists : Valider le travail de **LOON** en resimulant sous **ASIMUT** les netlists obtenues avec les vecteurs de test qui ont servi à valider la vue comportementale initiale.

Mettre les différents résultats (surface/temps/optimisation) dans votre rapport. Quelle est, selon vous, la meilleure des netlists ? Pourquoi ?