TP N°2
Commande d'un ascenseur par FPGA

I. But de la manipulation

Il s'agit d'effectuer la commande d'une maquette d'ascenseur par un circuit logique programmable de type FPGA de la famille Xilinx (voir Annexe 1 du cahier de TP).

L'objectif de la commande est d'assurer le déplacement d'une cabine d'ascenseur en fonction des appels utilisateurs reçus.

Cette manipulation a pour but d'une part d'illustrer le cours de « système de commande logique » sur la commande d'un système à événement discret, d'autre part d'établir une passerelle avec les enseignements de VHDL dispensés en 4ème année AE.

II. Présentation de la maquette à commander

Il s'agit d'une maquette d'ascenseur didactique de la société LANGLOIS. Les maquettes d'ascenseur ASC89 et ASC-19 comporte quatre paliers : un rez-de-chaussée et 3 étages.

Figure 1. La maquette d'ascenseur

A chaque étage se trouve un bouton d'appel pour la montée (sauf au 3ème étage) placé à l'extérieur de la cabine avec un voyant d'enregistrement de la demande et un bouton d'appel de descente (sauf au rez-de-chaussée) avec un voyant d'enregistrement de l'appel. Sur chaque palier une barrière opto-électrique permet de détecter la présence de la cabine au niveau du palier qui est alors indiquée sur un voyant lumineux.

L'ouverture et la fermeture des portes d'étages sont gérées par des motoréducteurs électriques derrière une plaque en plexiglas transparente sans accès manuel possible. Deux opto-détecteurs permettent à chaque étage, de repérer respectivement les positions portes ouvertes et portes fermées. Par mesure de sécurité, des fins de course sont en place pour les mouvements des portes sans accès de programmation possible.

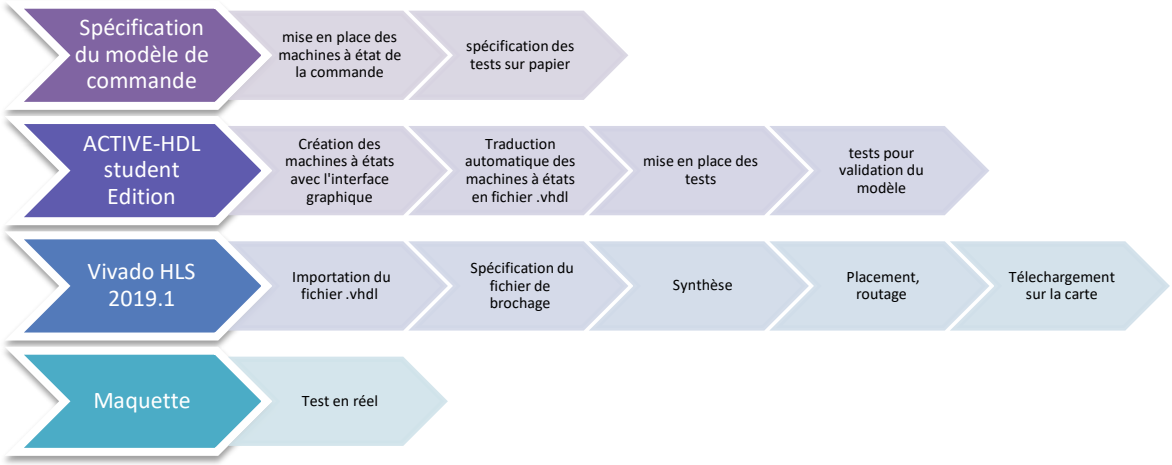
Sur la face avant de la maquette, une zone de couleur rouge symbolise l'intérieur de la cabine. A l'intérieur de la cabine un panneau de commande avec quatre boutons d'étages et un bouton stop est disponible. Quatre voyants d'étages indiquent la position de la

cabine qui est également équipée d'un voyant d'éclairage. Enfin un « switch » permet de simuler un obstacle à la fermeture de la porte.

L'annexe 2 du cahier de TP donne les entrées et les sorties disponibles sur la maquette et dont on se servira pour le processus de commande.

III. Manipulation

La méthodologie pour mettre en place une commande via la carte FPGA est illustrée ci-dessous.

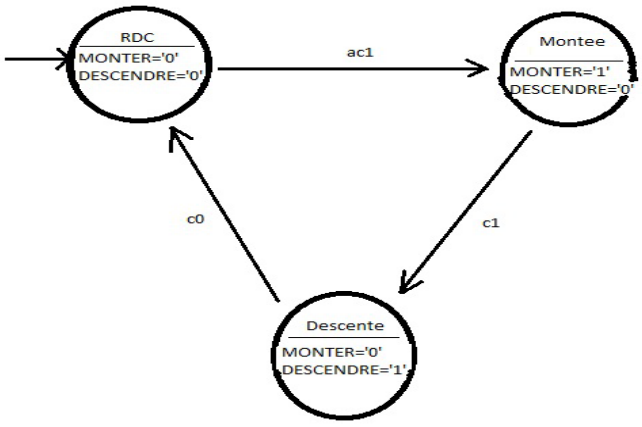


Partie 1 : Prise en main des logiciels à partir d'une modélisation de déplacement simple

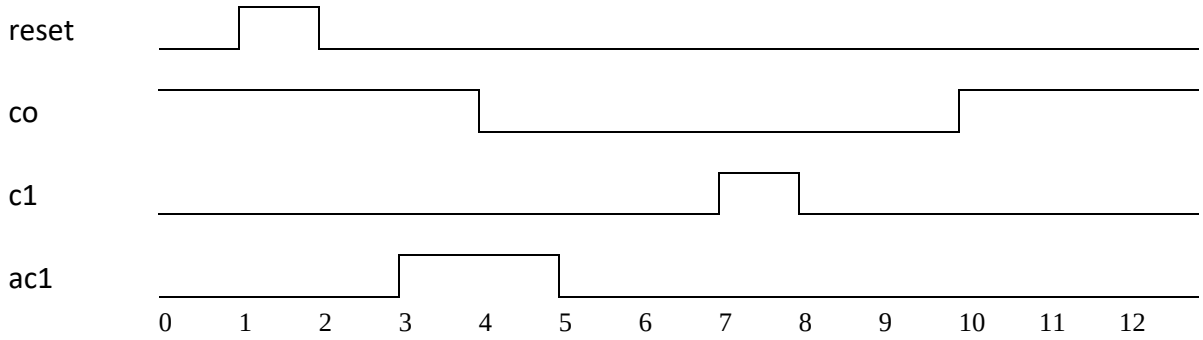
a) Spécification du modèle de commande

A l'état initial, la cabine est supposée être au RDC. Suite à un appel cabine pour le premier étage, la cabine se déplace jusqu'à l'étage 1, puis redescend directement au RDC.

La machine à états qui modélise cette commande est la suivante :



Les tests de validation à coder ont été choisis de manière à être les plus exhaustifs possibles.



Les chronogrammes des entrées sont tous spécifiés. Les sorties seront vérifiées lors des tests.

b) ACTIVE-HDL Student Edition

A l'aide du tutoriel ACTIVE-HDL disponible sur moodle, mettre en place la machine à état du déplacement simple, générer le fichier.vhdl, mettre en place la séquence de test donnée précédemment et valider le fonctionnement de la commande en vérifiant que les sorties obtenues sont celles attendues.

c) Vivado HLS 2019.1

A l'aide du tutoriel Vivado disponible sur moodle, importer le fichier .vhdl créé avec ACTIVE-HDL, spécifier le fichier de brochage, effectuer la synthèse.

A cette étape, il est possible d'avoir une vue du logigramme synthétisé. Pour cela, cliquer sur Open Synthestized Design dans l'arborescence de SYNTHESIS. Cliquer sur Schematic.

Remarquer les entrées et les sorties du système de commande.

Effectuer ensuite le placement, le routage, puis télécharger le programme sur la carte.

d) Maquette

Tester le programme sur la maquette et valider cette première commande.
Attention : N'oubliez avant de tester votre programme de placer la cabine au niveau du RDC. Pour cela utilisez le boitier de commande externe et vérifiez le positionnement de la cabine par rapport au capteur.

N'oubliez pas non plus d'initialiser votre machine à états, en appuyant sur le bouton RESET à côté de la carte.

Partie 2 : Mémorisation des appels

a) Spécification du modèle de commande

Préparation : Pour chacun des étages, modéliser par une machine à états la mémorisation de l'appel. Préparer une séquence de test permettant de tester une des machines à états de manière exhaustive.

Chaque mémorisation d'étage sera modélisée par une machine distincte, on aura donc 4 machines à états, une pour chaque étage.

Ces machines auront deux états :

- Appel mémorisé à l'étage i

- Pas d'appel pour l'étage i.

Lorsque l'appel à un étage i est mémorisé (quel que soit le bouton d'appel utilisé), un des voyants associés à cet étage doit être allumé. Le voyant devra être éteint, dès que l'appel ne sera plus mémorisé i.e. quand l'étage aura été atteint.

On utilisera les noms des variables d'entrées et de sorties présentées en Annexe 2 du cahier de TP.

b) ACTIVE-HDL Student Edition

En suivant la méthodologie présentée, créer un nouveau projet et coder sur l'interface graphique de ACTIVE-HDL les 4 machines à états de mémorisation.

Tester le fonctionnement d'une des 4 machines en simulation uniquement.

Préparation : préparer les chronogrammes de la séquence de test

Partie 3 : Gestion des mouvements de la cabine

a) Spécification du modèle de commande

Préparation : Modéliser par machine à états la commande de la cabine

Il s'agit d'effectuer la commande de l'ascenseur c'est à dire la commande des mouvements de la cabine et de la porte selon le cahier des charges suivant :

- Les appels (cabine et paliers) sont mémorisés par les machines à états précédentes
- La cabine doit monter et descendre selon les appels effectués
- En cas de conflit entre un appel vers un étage supérieur et inférieur, on privilégiera l'appel qui est dans le sens du mouvement de l'ascenseur.
- Il n'y a pas de priorité entre les appels de l'extérieur et ceux actionnés depuis l'intérieur de la cabine.
- Les portes ne sont pas gérées.

En considérant les deux types de déplacement (MONTER et DESCENDRE). Le graphe d'états peut être facilement obtenu en considérant que la cabine peut être dans un des quatre états suivants :

- cabine en train de monter,
- cabine en train de descendre,
- cabine arrêtée en montée,
- cabine arrêtée en descente.

b) ACTIVE-HDL Student Edition

A l'aide du tutoriel ACTIVE-HDL disponible sur moodle, mettre en place la machine à état du déplacement dans le même projet que celles des mémorisations. Générer le fichier.vhdl.

c) Vivado HLS 2019.1

A l'aide du tutoriel Vivado disponible sur moodle, importer le fichier .vhdl créé avec ACTIVE-HDL, spécifier le fichier de brochage, effectuer la synthèse, le placement, le routage, puis télécharger le programme sur la carte.

d) Maquette

Tester le programme final sur la maquette en fonctionnement réel.

Partie 4 : Extension possible : Gestion des portes

Lorsque vous avez terminé la réalisation de la commande de la cabine sans la gestion des portes, rajoutez la prise en compte des portes.

Il vous faut reprendre toutes les étapes depuis la mise en place de votre design.

IV. Références

- Manuel d'utilisation du logiciel Vivado

  https://www.xilinx.com/support/documentation/sw_manuals/xilinx2019_2/ug904-vivado-implementation.pdf

- Manuel d'utilisation ACTIVE-HDL

  https://courses.cs.washington.edu/courses/cse352/10au/Tutorials/tutorials.html

V. Préparation attendue

- Machines à états de mémorisation des appels (4 machines à états)
- Séquence de test
- Commande complète de la cabine

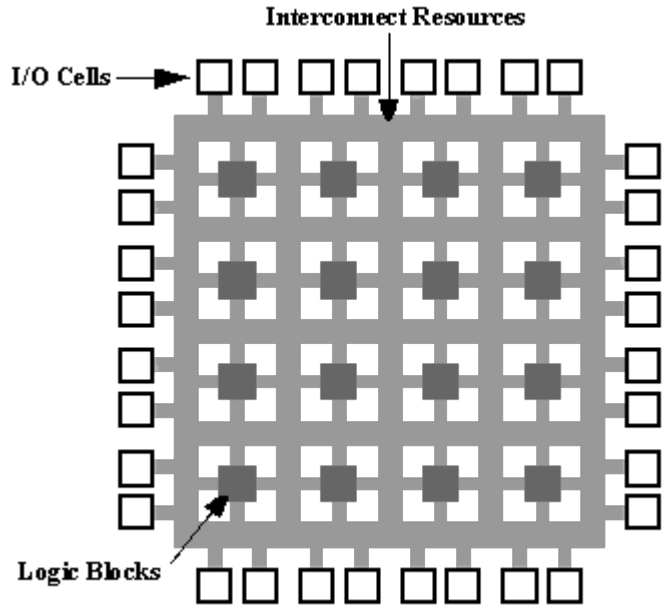
Annexe 1

Quelques mots sur les FPGA…..

Un FPGA (Field Programmable Gate Arrays) est un composant entièrement reconfigurable. L'avantage de ce genre de circuit est sa grande souplesse qui permet de les réutiliser à volonté dans des algorithmes différents en un temps très court.

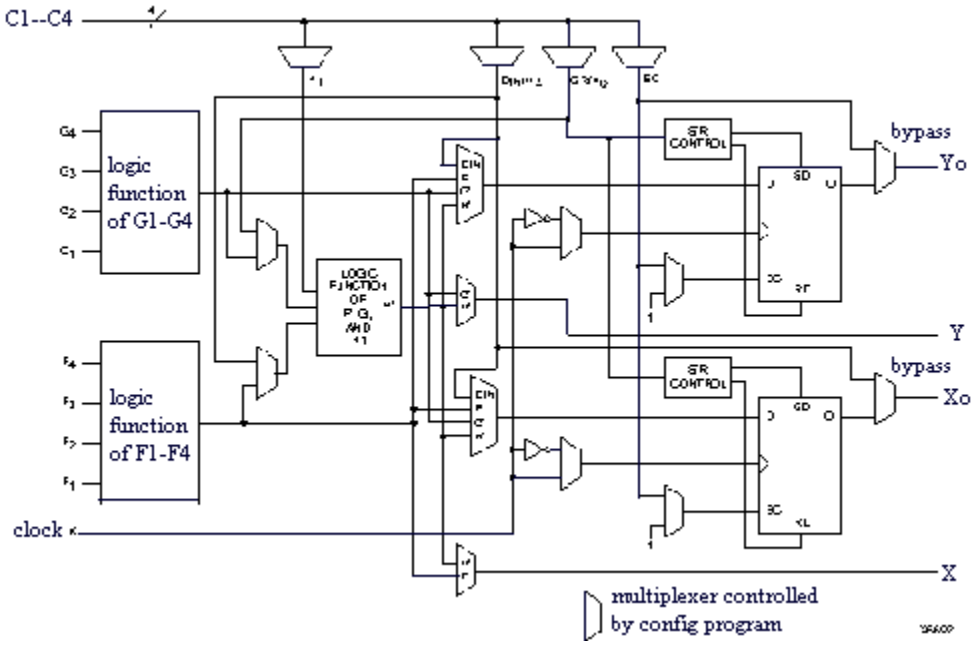
Les circuits FPGA sont constitués d'une matrice de blocs logiques (Logic Block) programmables entourés de blocs d'entrée sortie programmables. L'ensemble est relié par un réseau d'interconnexions programmable. Plusieurs classes de FPGA existent selon l'agencement des blocs et des interconnexions sur le circuit.

Les FPGA utilisés dans ce TP de chez Xilinx ont une structure de type matrice symétrique selon le principe de la figure ci-dessous.



Structure interne d'un FPGA de chez Xilinx

L'utilisateur peut programmer la fonction réalisée par chaque cellule logique (CLB) (slice chez Xilinx) et programmer les interconnexions entre cellules. Les CLB permettent de réaliser des fonctions combinatoires et séquentielles (bascules).

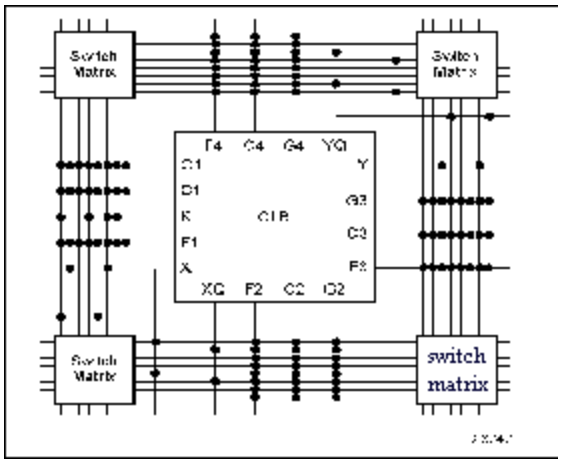


Structure interne d'un CLB

Les blocs IOB (input output bloc) permettent l'interface entre les broches du composant FPGA et la logique interne développée à l'intérieur du composant. Ils sont présents sur toute la périphérie du circuit FPGA. Chaque bloc IOB contrôle une broche du composant et il peut être défini en entrée, en sortie, en signaux bidirectionnels ou être inutilisé.

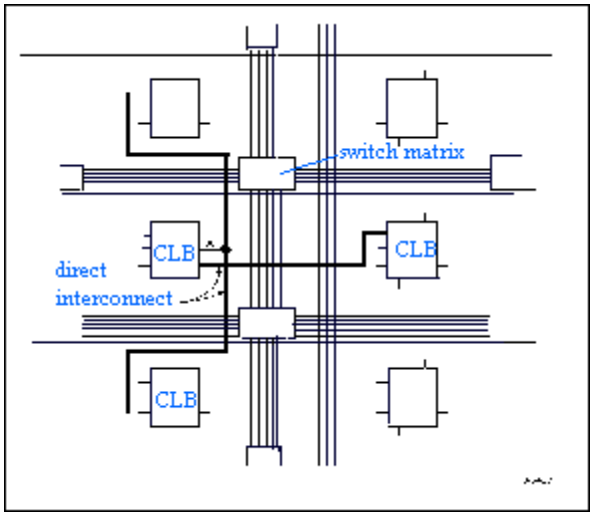
Les connexions internes dans les circuits FPGA sont composées de segments métallisés. Le rôle de ces interconnexions est de relier avec un maximum d'efficacité les blocs logiques et les entrées/sorties afin que le taux d'utilisation dans un circuit donné soit le plus élevé possible. Xilinx propose trois sortes d'interconnexions selon la longueur et la destination des liaisons.

Des interconnexions à usage général, mises en place selon une grille de cinq segments métalliques verticaux et quatre segments horizontaux positionnés entre les rangées et les colonnes de CLB et de l'IOB. A chaque intersection des matrices de commutation ont pour rôle de raccorder les segments entre eux. Ces interconnexions sont utilisées pour relier un CLB à n'importe quel autre.



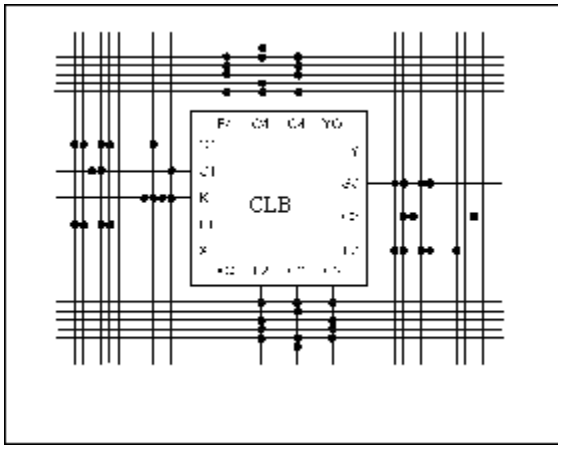
Connexions à usage général et matrices de commutation.

Des interconnexions directes pour l'établissement de liaisons entre les CLB et les IOB avec un maximum d'efficacité en termes de vitesse et d'occupation du circuit. Ces lignes peuvent aussi connecter directement certaines entrées d'un CLB aux sorties d'un autre.



Les interconnexions directes.

Enfin, les longues lignes sont de longs segments métallisés parcourant toute la longueur et la largeur du composant. L'utilisation de ces lignes permet de minimiser les points d'interconnexion.



Les longues lignes.

Les performances des interconnexions dépendent du type de connexions utilisées. Pour les interconnexions à usage général, les délais générés dépendent du nombre de segments et de la quantité d'aiguilleurs employés. Le délai de propagation de signaux utilisant les connexions directes est minimum pour une connectique de bloc à bloc. Quant aux segments utilisés pour les longues lignes, ils possèdent une faible résistance mais une capacité importante. De plus, si on utilise un aiguilleur, sa résistance s'ajoute à celle existante.

En plus des CLB et des interconnexions, un oscillateur à quartz placé dans un angle du composant peut être activé lors de la phase de programmation pour réaliser un oscillateur.

Les circuits FPGA ne possèdent pas de programme résident. A chaque mise sous tension, il est nécessaire de les configurer. Leur configuration permet d'établir des

interconnexions entre les CLB et IOB. Pour cela, ils disposent d'une RAM interne dans laquelle sera écrit le fichier de configuration.

La programmation d'un FPGA de déroule en plusieurs étapes résumées sur le schéma en fin de l'annexe.

Le circuit à réaliser (design) est tout d'abord décrit soit à partir d'un langage de description matériel (e.g. VHDL, Verilog, ABEL, SystemC...) soit en rentrant directement le schématique. Le synthétiseur qui est un compilateur particulier, se charge ensuite à partir du langage de description VHDL ou Verilog, de générer une description structurelle du circuit. A travers la description synthétisable le synthétiseur va reconnaître un certain nombre de primitives qui lui sont propres et qu'il va implanter et connecter entre elles. Ce sera au minimum des portes NAND, NOR, NOT, des bascules D, des buffers d'entrées-sorties, mais cela peut être aussi un compteur, un multiplexeur, une RAM, un registre, un additionneur, un multiplieur ou tout autre bloc particulier.

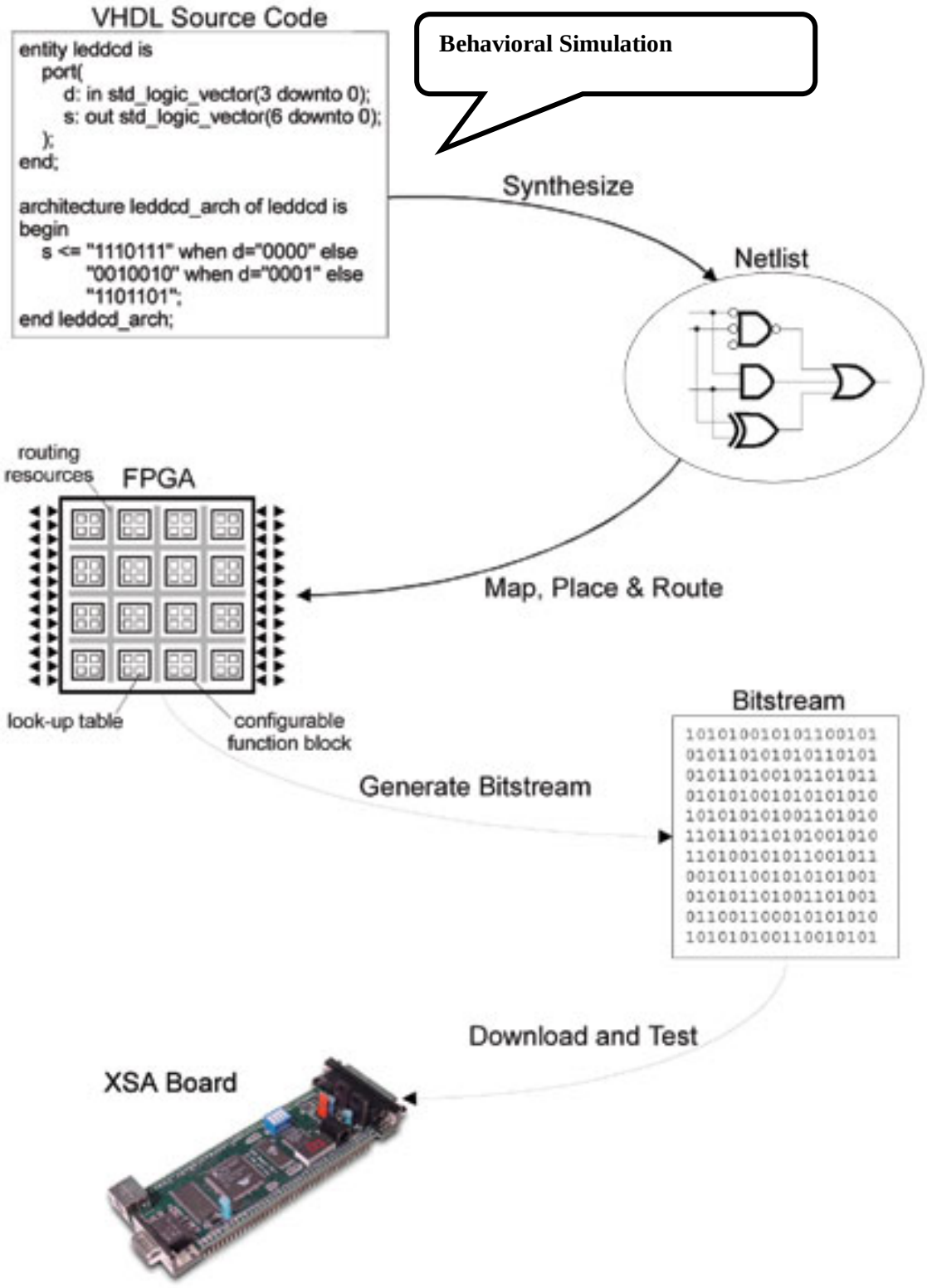
Le synthétiseur va ainsi générer ce qu'on appelle la netlist qui définie les types de blocs et le nombre de blocs utilisés pour le design. Il faut ensuite procéder au « placement » de tous les composants utilisés et effectuer le routage entre les différentes cellules logiques. La netlist contient toutes les informations nécessaires pour le routage. Le placement-routage est un problème d'optimisation difficile qui nécessite l'emploi de techniques métaheuristiques (algorithmes itératifs, qui progressent vers un optimum global) évoluées. Il est toutefois nécessaire de laisser une marge de manœuvre à l'algorithme pour obtenir plus rapidement une solution acceptable mais non-optimale.

A la fin de ces étapes, le synthétiseur va générer le bitstream, sorte d'exécutable qui sera envoyé vers le FPGA permettant ainsi de réaliser la programmation proprement dite du FPGA. Le format détaillé du bitstream est un format propriétaire.

Pour en savoir plus :

http://www.xilinx.com/support/documentation/spartan-3.htm

http://www.fpga4fun.com/index.html



Développement d'une application sur FPGA

Annexe 2

Tableau des connexions pour la carte ARTIX 7

PIN	Nom	Fonction
L17	clk	horloge
G19	reset	reset
A16	MONTER	mise en marche du moteur de montée
L3	DESCENDRE	mise en marche du moteur de descente
K2	VC0	Voyant cabine pour l'étage 0
L1	VC1	Voyant cabine pour l'étage 1
L2	VC2	Voyant cabine pour l'étage 2
M1	VC3	Voyant cabine pour l'étage 3
N3	EC	Eclairage cabine
J1	FP0	Moteur qui ferme les portes de l'étage 0
A14	FP1	Moteur qui ferme les portes de l'étage 1
A15	FP2	Moteur qui ferme les portes de l'étage 2
C15	FP3	Moteur qui ferme les portes de l'étage 3
J3	OP0	Moteur qui ouvre les portes de l'étage 0
B15	OP1	Moteur qui ouvre les portes de l'étage 1
H1	OP2	Moteur qui ouvre les portes de l'étage 2
K3	OP3	Moteur qui ouvre les portes de l'étage 3
G17	VPD1	Voyant palier à l'étage 1 pour descente
N1	VPD2	Voyant palier à l'étage 2 pour descente
P3	VPD3	Voyant palier à l'étage 3 pour descente
M3	VPM0	Voyant palier à l'étage 0 pour montée
N2	VPM1	Voyant palier à l'étage 1 pour montée
M2	VPM2	Voyant palier à l'étage 2 pour montée
V4	obs	Choix du mode avec ou sans obstacle
U2	stop	Bouton stop
W2	pf0	capteur portes fermées étage 0
U1	pf1	capteur portes fermées étage 1
T2	pf2	capteur portes fermées étage 2
T1	pf3	capteur portes fermées étage 3
W5	po0	capteur portes ouvertes étage 0
V3	po1	capteur portes ouvertes étage 1
W3	po2	capteur portes ouvertes étage 2
V2	po3	capteur portes ouvertes étage 3
U4	ac0	appel cabine pour l'étage 0
V5	ac1	appel cabine pour l'étage 1
W4	ac2	appel cabine pour l'étage 2
U5	ac3	appel cabine pour l'étage 3

U3	ad1	appel descente depuis le palier de l'étage 1
W7	ad2	appel descente depuis le palier de l'étage 2
V8	ad3	appel descente depuis le palier de l'étage 3
W6	am0	appel montée depuis le palier de l'étage 0
U7	am1	appel montée depuis le palier de l'étage 1
U8	am2	appel montée depuis le palier de l'étage 2
R2	c0	capteur cabine à l'étage 0
T3	c1	capteur cabine à l'étage 1
R3	c2	capteur cabine à l'étage 2
P1	c3	capteur cabine à l'étage 3