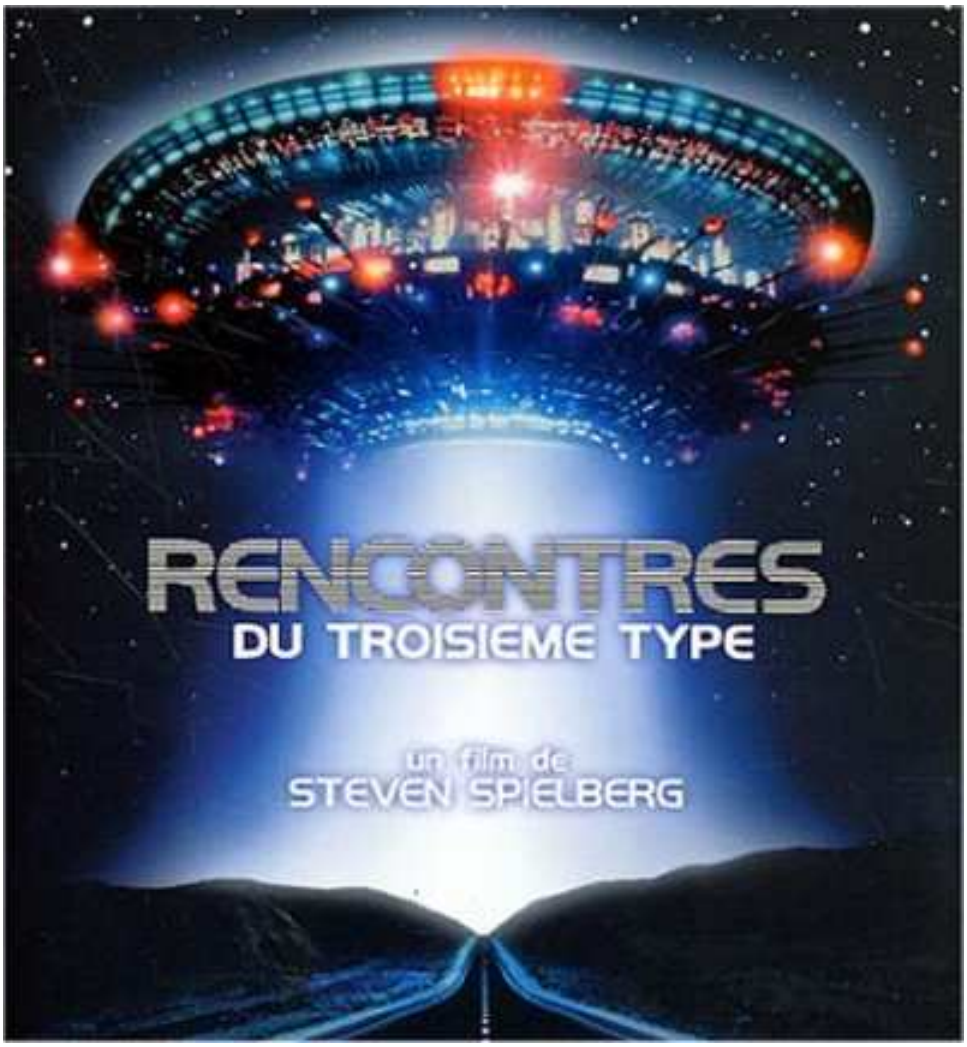
Mini-Projet d'Electronique Numérique
LE201

Licence d'Electronique

Hiver 2007



Introduction

Vous avez dans les mains le texte du mini-projet d'électronique numérique, LE201.

Dans ce texte il y a 3 parties qu'il vous faut soigneusement lire :

1. Une partie de description du fonctionnement des outils.
2. Une partie qui énonce le mini-projet.
3. Une partie de description de la carte sur laquelle vous allez travailler.

Ce mini-projet se prépare, il ne faut pas arriver en scéance de TP sans avoir une solide idée de ce que vous voulez mettre en oeuvre. Il faut notamment avoir réalisé des schémas des différentes composants que vous voulez décrire en VHDL.

Toutes les scéances de TP sont obligatoires, une absence doit être justifiée au secrétariat. Une absence injustifiée aura des conséquences sur la note de TP finale.

L'évaluation de votre projet aura lieu à l'aide de trois critères

- Un critère de participation et de maîtrise des outils lors des scéances de TP (15 % de la note).
- Une évaluation d'un rapport écrit que vous remettrez à votre enseignant de TP (50 % de la note).
- Une soutenance finale où vous ferez la démonstration de votre projet et répondrez à des questions (35 % de la note).

Bon mini-projet,

L'équipe enseignante.

Documentation des outils

1 Configuration du compte sous Linux

Cette étape est à réaliser une fois pour toute.

Lancer un terminal et taper la commande `/opt/fpgadv63/Fpgadv/bin/conftuto`

2 Lancement du logiciel

Une fois votre compte configuré, pour lancer le logiciel il faut que vous lanciez un terminal et que dans ce terminal vous tapiez la commande `fa_with_ps`.

3 Ouverture d'un projet

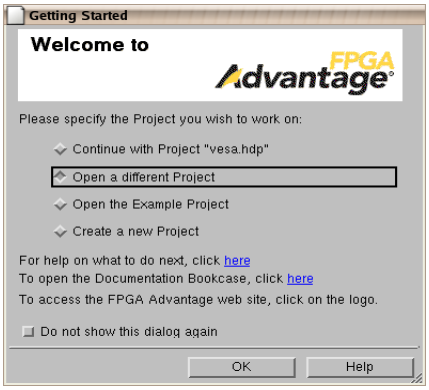


Figure 1: Fenêtre "Bienvenue dans le monde Merveilleux de Fpga-Advantage"

Quand vous lancez le logiciel une première fenêtre, figure 1, s'ouvre permettant d'ouvrir un projet, cochez le bouton `Open a Different Project` et cliquez sur le bouton `OK`.

Ceci va vous ouvrir une seconde fenêtre, figure 2, dans laquelle vous pourrez choisir un des projets qui existe déjà.

4 Création d'un projet

Quand le logiciel est démarré, une première fenêtre s'ouvre permettant de choisir un projet, figure 1. La première fois que vous lancer le logiciel, il faut créer un nouveau projet, pour cela cliquez sur le bouton `Create a new Project` et ensuite cliquez sur le bouton `ok`.

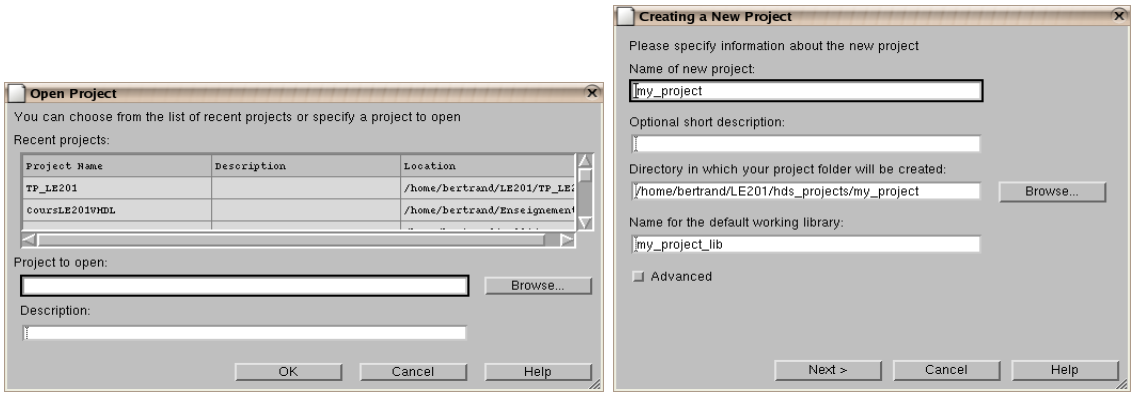


Figure 2: Fenêtres de choix d'un projet existant et de création d'un joli projet

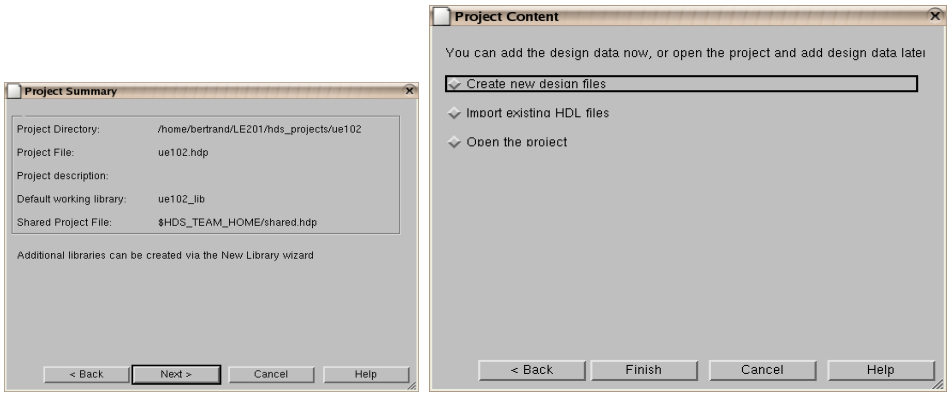


Figure 3: Fenêtre de résumé des choix du projet et de fin de création d'un projet

Une nouvelle fenêtre, figure 2, s'ouvre pour configurer votre projet. Dans cette fenêtre il y a des champs à renseigner qui sont :

- Le champ `Name of new project` où vous donnez un nom à votre projet, par exemple un nom comme `UE201`.
- Le champ `Directory in which your project folder will be created`, ici vous indiquez un chemin valide à partir de votre racine, par exemple `/home/utilisateur/UE201`, dans cet exemple il faut remplacer `/home/utilisateur` par le chemin menant à la racine de votre compte.
- Le champ `Name for the default working library`, ici vous indiquez la librairie VHDL de travail (work en anglais) par défaut, pour la première scéance de TP mettez par exemple `DCB_lib`

Une fois ces champs renseignés, cliquez sur le bouton `next`. Une nouvelle fenêtre s'ouvre qui résume vos choix, figure 3, ici cliquez encore sur le bouton `next`. Enfin, une dernière fenêtre s'ouvre, figure 3, dans laquelle vous vérifierez que toutes les cases sont décochées, si oui cliquez sur le bouton `Finish`.

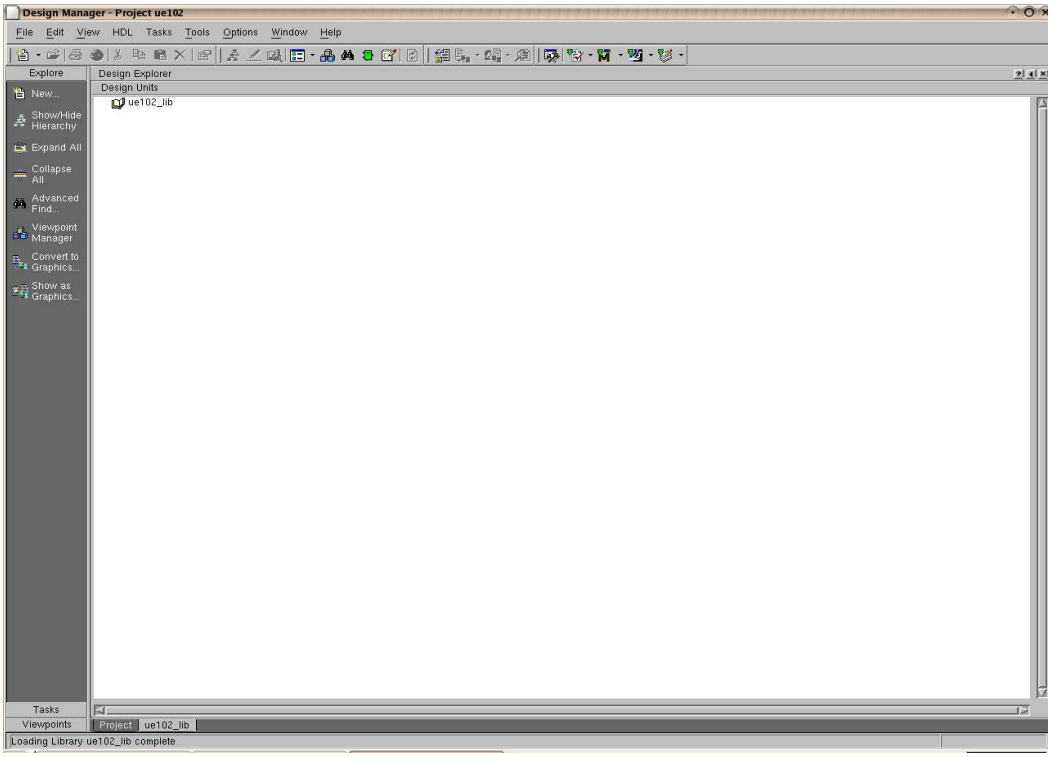


Figure 4: Fenêtre du projet et de la librairie par défaut

Vous avez créé un nouveau projet avec une librairie par défaut.

Vous devez voir apparaître, le nom de votre librairie par défaut avec un symbole qui est un livre tel qu'en figure 4.

5 Création d'un fichier VHDL

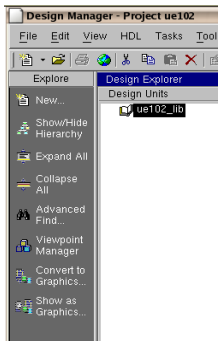


Figure 5: Bouton de création d'un nouveau fichier VHDL

Pour créer un nouveau fichier VHDL, sélectionnez votre librairie et ensuite cliquez sur l'icône `new` sur votre gauche, figure 5.

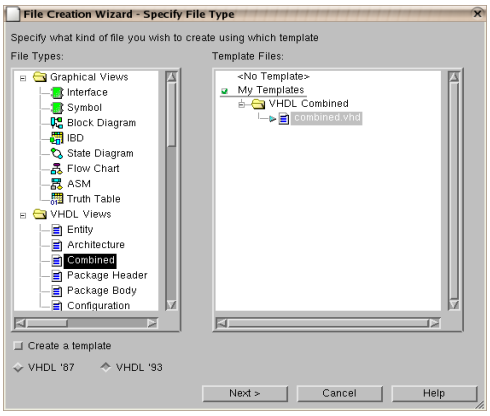


Figure 6: Fenêtre d'aide à la création de composants VHDL

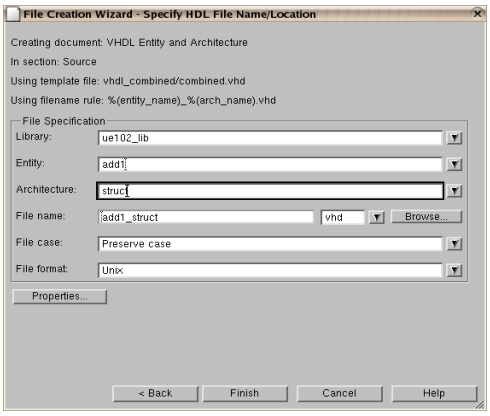


Figure 7: Fenêtre de création d'un composant VHDL

Une fenêtre, figure 6, d'aide à la création de composants VHDL s'ouvre, dans cette fenêtre vous sélectionnez dans `VHDL views` la vue de type `combined`, vous vérifiez que la case `VHDL'93` est bien cochée, si non vous la cochez. Ensuite vous cliquez sur le bouton `next`.

Une fenêtre s'ouvre, figure 7, avec des champs à renseigner, qui sont :

- Le champ `Entity`, dans lequel vous indiquez le nom de votre entité, par exemple `add1`.
- Le champ `Architecture`, dans lequel vous indiquez le nom de votre architecture, par exemple `struct`.

Une fois les champs renseignés, vous cliquez sur le bouton `finish`, là un éditeur de texte VHDL doit s'ouvrir, figure 8, avec de déjà pré-écrit les librairies par défaut à inclure et les patrons de votre entité et de votre architecture.

Vous saisissez votre code VHDL dans cet éditeur, vous le sauvegardez et une fois fini vous le fermez.

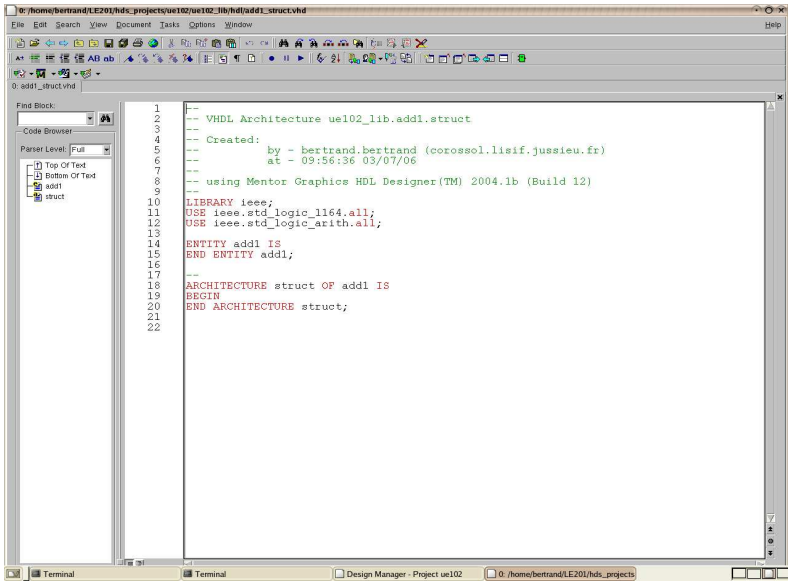


Figure 8: Fenêtre de l'éditeur VHDL

6 Compilation

Une fois que votre composant est écrit. Vous vous positionnez dans votre librairie et vous le sélectionnez, ensuite vous appuyez sur l'icône représentant un grand M bleu  . Ceci à pour effet de compiler votre composant et de lancer la simulation. Lors de la compilation, une fenêtre appelée `Log Window` s'ouvre, c'est là que sont répertoriées vos erreurs, si il y en a des lignes rouges vous indiquant l'erreur en question appaîtront.

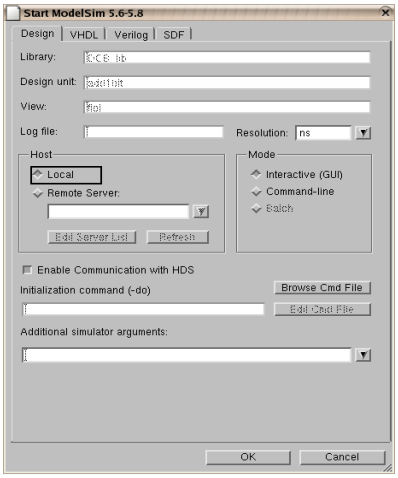


Figure 9: Fenêtre de démarrage de ModelSim

Si la compilation est bonne, une autre fenêtre s'ouvre, c'est la fenêtre `Start ModelSim 5.6-5.8`, figure 9, dans cette fenêtre vous cliquez sur le bouton `ok` et cela démarre le simulateur.

7 Simulation

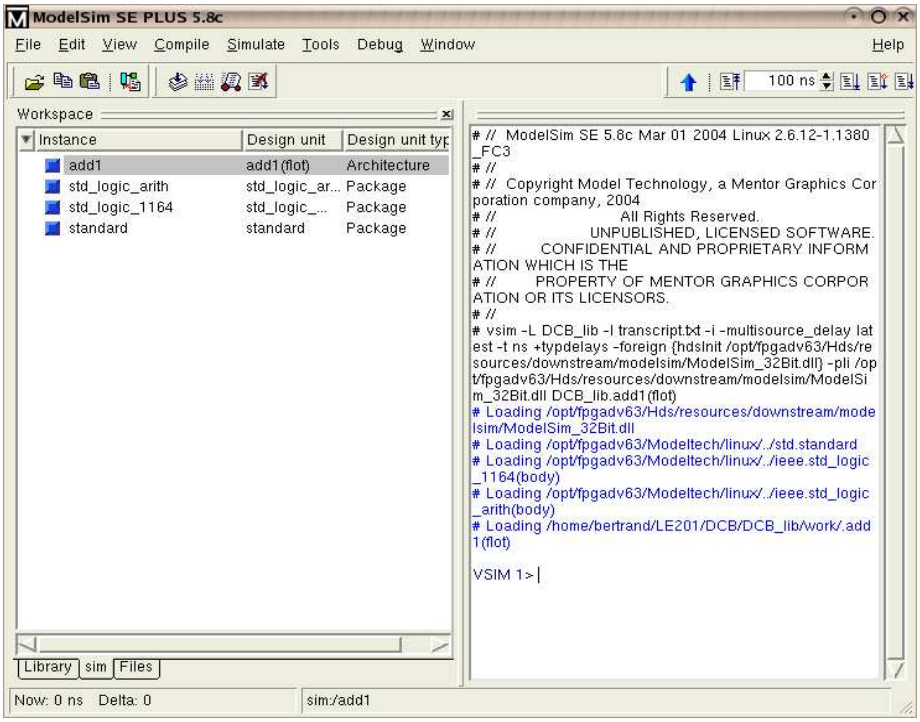


Figure 10: Fenêtre principale de ModelSim

Une fois le simulateur démarré, figure 10, vérifiez que votre composant, dans la partie gauche de la fenêtre, est bien sélectionné. Ensuite dans le menu `View` du simulateur, choisissez `Wave` et ensuite choisissez `Signals`.

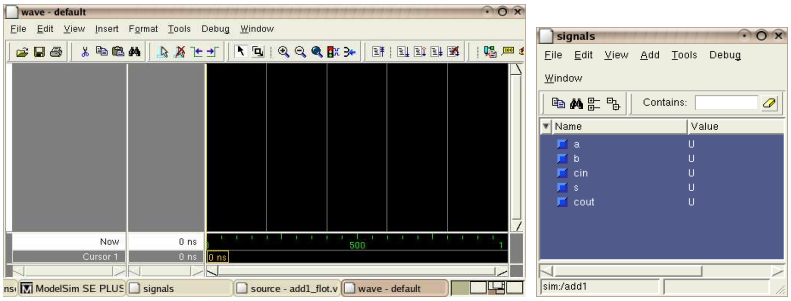


Figure 11: Fenêtre de visualisation des signaux et des signaux de Modelsim

Cela ouvre deux fenêtres, une fenêtre `Wave`, figure 11, et une fenêtre `Signals`, figure 11. Dans la fenêtre `Signals`, dans le menu `Edit` faites `Select All`, cela sélectionne tous les signaux de cette fenêtre, figure 12.

Ensuite positionnez le curseur de la souris sur l'un de ces signaux, enfin en appuyant sur le bouton gauche et en le maintenant appuyé faites glisser les signaux de la fenêtre `Signals` à la fenêtre `Wave`,

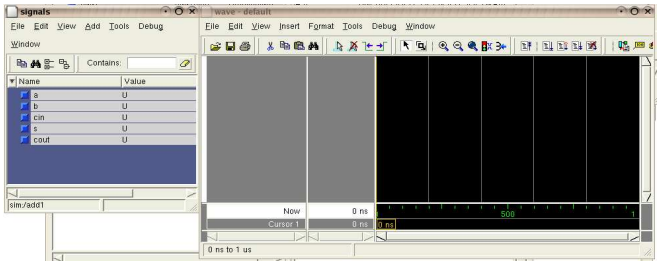


Figure 12: Sélection des signaux

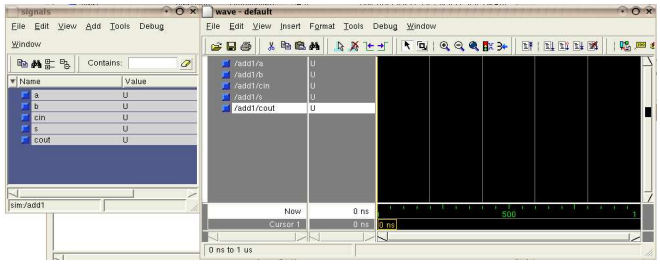


Figure 13: Mise à jour de la fenêtre de visualisation des signaux

figure 13.

Vous pouvez fermer la fenêtre `Signals`.

Allez maintenant dans la fenêtre `Wave`, cliquez sur l'un des signaux et appuyez sur le bouton de droite de la souris en le maintenant appuyé. Un menu déroulant apparaît, dans ce menu vous pouvez mettre une valeur constante à votre signal en choisissant `Force`, ou le faire varier comme un signal périodique carré en choisissant `Clock`.

Affectez une valeur valide à tous vos signaux d'entrée. En aucun cas vous n'affectez de valeur aux signaux de sortie !. Ensuite, appuyez sur l'icône représentant une feuille de papier avec une flèche bleue pointée vers le bas,  . A chaque appui sur cet icône vous simulez un pas de temps, ce pas étant par défaut de 100 ns.

Voilà c'est fini. Vous pouvez dorénavant décrire de nouveaux composants.

8 Mise en Oeuvre matérielle ou synthèse

Afin de pouvoir programmer la carte Spartan3, il vous faut réaliser ce que l'on nomme la synthèse de votre description VHDL, qui permettra ensuite à l'aide des outils propriétaires de Xilinx de réaliser un fichier permettant de programmer le composant spartan3.

Cette synthèse est réalisée à l'aide de l'outil précision, la démarche pour utiliser cet outil est décrite ci-après.

8.1 Mise en correspondance entre des entrées-sorties des composants et des broches du circuit physique

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity et_ou_xor is
port(a,b : in std_logic;
s : out std_logic_vector(2 downto 0));

-- Début de la mise en correspondance des ports d'entrées-sorties
-- avec des broches physique de la carte spartan3

-- Déclaration d'un type mentor_s et des 2 attributs

type mentor_s is array (natural range <>, natural range <>) of character;
attribute pin_number : string;
attribute array_pin_number : mentor_s;

-- Mise en correspondance des signaux s(2), s(1) et s(0)
-- avec les leds LD2, LD1 et LD0

attribute array_pin_number of s : signal is ("L12","P14","K12");

-- Mise en correspondance des signaux a et b avec les interrupteurs
-- SW0 et SW1

attribute pin_number of a : signal is "F12";
attribute pin_number of b : signal is "G12";

-- Fin de la mise en correspondance

end entity et_ou_xor;

architecture flot of et_ou_xor is
begin
s(0) <= a and b;
s(1) <= a or b;
s(2) <= a xor b;
end architecture flot;
```

Figure 14: Exemple d'affectation des broches du Spartan3 en VHDL

Une fois écrit et testé votre composant VHDL, il est nécessaire avant de passer à la synthèse, de mettre en correspondance certaines entrées-sorties de votre composant et des broches du circuit spartan3, pour réaliser ceci il suffit d'utiliser une structure VHDL qui se nomme `attribut` et dont l'utilisation est décrite à travers l'exemple de la figure 14.

Dans cet exemple je met en correspondance les entrées a et b de mon composant avec les interrupteurs SW0 et SW1 de la carte spartan3 et les sorties s(2), s(1) et s(0) avec respectivement les led LD2, LD1 et LD0.

Pour effectuer cette mise en correspondance, vous devez obligatoirement écrire après la déclaration des ports de votre entité et la fin de votre entité les lignes qui permettent de définir les attributs `pin_number` et `array_pin_number`, ces attributs sont ensuites reconnus par l'outil de synthèse Précision pour connecter la bonne broche au bon signal.

Ces lignes sont :

```
type mentor_s is array (natural range <>, natural range <>) of character;
attribute pin_number : string;
attribute array_pin_number : mentor_s;
```

Ensuite pour chaque signal que vous voulez connecter à un port physique, il vous suffit d'utiliser soit l'attribut `pin_number` pour un signal simple soit l'attribut `array_pin_number` pour un vecteur de signaux.

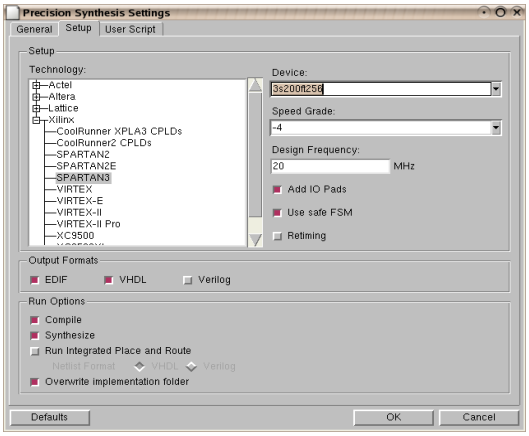


Figure 15: Démarrage de Précision

8.2 Synthèse avec Précision

Une fois la mise en correspondance effectuée, il est possible de réaliser la synthèse, pour cela il suffit de sélectionner votre composant dans votre librairie et cliquez sur l'icone où il y a les lettres PS sur fond bleu et noir  .

Cela va lancer la synthèse en ouvrant une première fenêtre, figure 15, dans laquelles vous allez indiquer quel est le composant que vous désirez programmer.

Dans cette fenêtre, vous avez un espace à gauche dans lequel sont indiqués des constructeurs de circuits programmables (Altera, Xilinx, ...), choisissez `Xilinx` et cliquez sur le symbole `+` encadré à gauche, cela ouvre une liste de composants Xilinx disponibles. Parmi ces composants, sélectionnez le composant `SPARTAN3`. Quand vous sélectionnez ce composant, dans le champ `Device` à droite dans la fenêtre apparaît une référence qui débute par `3s`, allez dans ce champs et cliquez sur la petite flêche en bout à droite pour afficher la liste des Spartan3 disponibles, choisissez la référence `3s200ft256`.

Une fois le composant choisi, cliquez sur le bouton `OK` en bas à droite, cela lance maintenant l'outil de synthèse avec le bon composant.

Une fenêtre s'ouvre et la synthèse est initiée, figure 16, la première phase de synthèse est finie une fois que l'on peut accéder aux différentes possibilités de l'outil. Quand cette phase est terminée, allez dans le menu `Tools`, puis dans le sous menu `Set options`. Cela ouvre une fenêtre dans laquelle il est possible de définir les options de synthèse. Dans cette fenêtre sélectionnez `ISE6.3` et cliquez sur

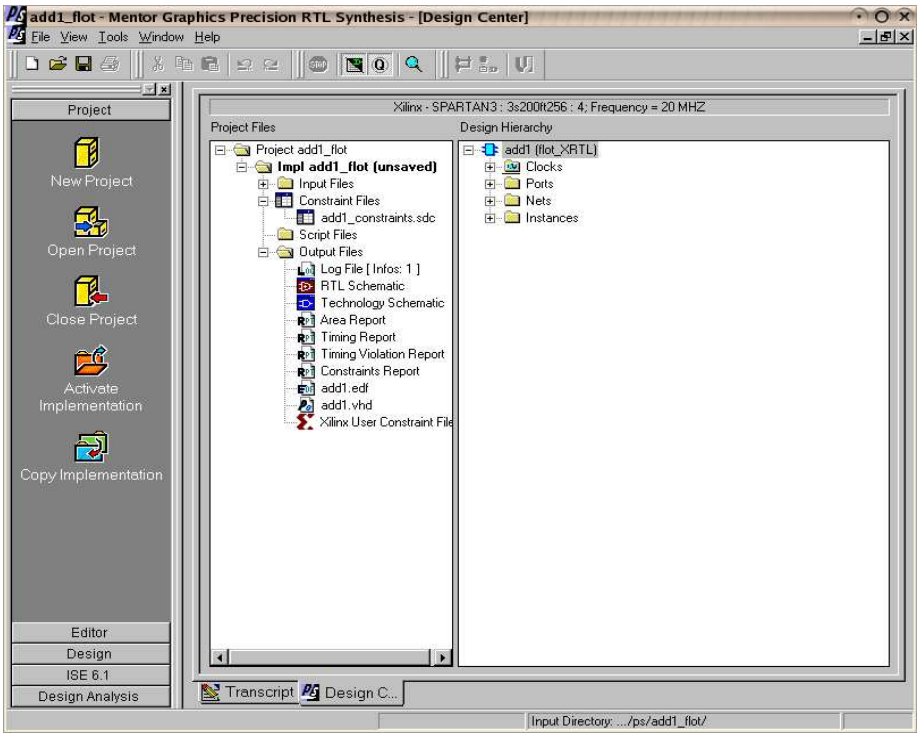


Figure 16: Fenêtre Principale de Précision

le + encadré à gauche. Une liste s'affiche sélectionnez `Integrated Place and Route` et vérifiez que le choix `Generate BitGen File` est bien coché. Une fois coché, cliquez sur le bouton `Apply` puis sur le bouton `OK` en bas à gauche.

Dans la fenêtre Précision cliquez sur l'onglet `ISE6.1` à gauche et ensuite sur le bouton `Place and Route`. Cela est la seconde phase de la synthèse et va générer un fichier avec une extension `.bit` permettant de programmer le spartan3.

Une fois cette seconde phase terminée, allez dans le menu `File` et choisir `Save Project` ensuite choisir dans le menu `File` l'item `Exit`.

La synthèse est terminée, il suffit maintenant de programmer le spartan3.

8.3 Programmation du spartan3

La programmation du spartan3 se fait à l'aide du programme `progXilinx`. Pour programmer la carte, il suffit d'appuyer sur le bouton  dans la barre des boutons.

Cela lance une fenêtre, figure 17, dans cette fenêtre il y a bouton `Fichier a Telecharger`, cliquez sur ce bouton cela vous ouvre une fenêtre de choix de fichier à télécharger, figure 17 et 18.

Choisir votre fichier d'extension `.bit` à télécharger et cliquez sur le bouton `open`. Vous avez choisi votre fichier et son nom doit apparaître dans la bande reservée à cet usage, comme on le voit sur la figure 18.

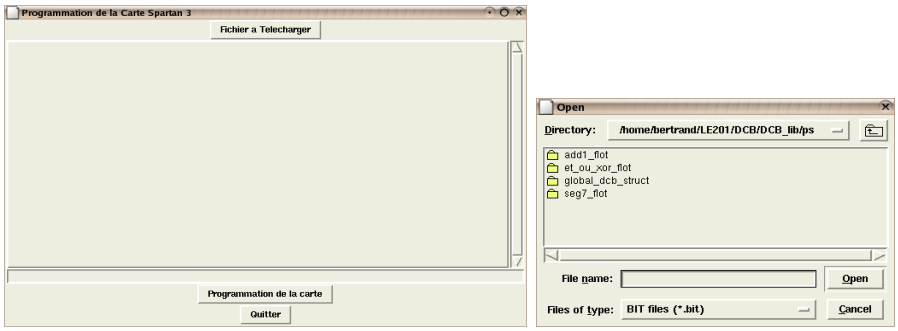


Figure 17: Fenêtre de programmation de la carte Spartan3 et de choix de fichier à télécharger

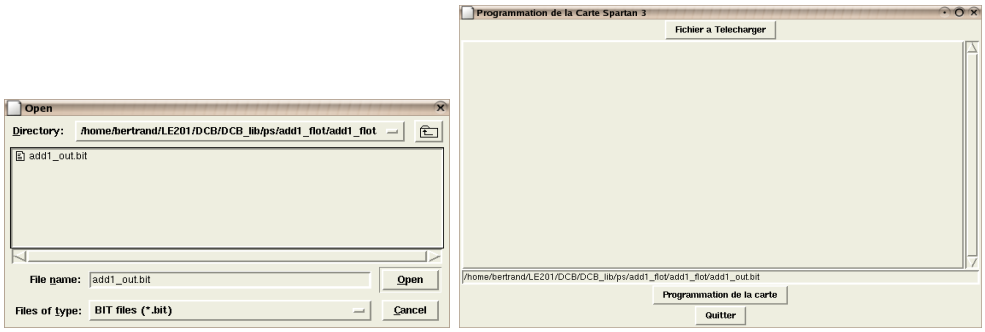


Figure 18: Fenêtre de choix du fichier à télécharger

Il vous suffit maintenant d'appuyer sur le bouton `Programmation de la Carte` pour programmer la carte, une fois finie appuyer sur le bouton `Quitter`.

Enoncé

Prise en Main de l'environnement de travail : Additionneur DCB

1. Ouvrez le projet `UE201_tuto`
2. Dans ce projet il existe une librairie nommée `DCB_lib`
3. Vous créérez ensuite tous les composants de cet énoncé dans cette librairie.
4. Lors de la commande `/opt/fpgadv63/Fpgadv/bin/conftuto`, un certain nombre de fichiers ont été copié. Ces fichiers décrivent certains composants nécessaires à la réalisation de l'additionneur DCB.

1. **Additionneur 1 bit**
   (a) Compilez le composant additionneur 1 bit qui vous ait fourni.
   (b) Testez, en le simulant, ce composant.
2. **Mise en Oeuvre additionneur 1 bit**
   Vous allez mettre en oeuvre l'additionneur 1 bit sur la carte Spartan3.
   (a) Ouvrez le fichier VHDL de votre composant, et indentifiez les lignes qui permettent de mettre en correspondance les broches du circuit Spartan3 avec les entrées-sorties de l'additionneur 1 bit.
   (b) A l'aide de l'outil Précision faites la synthèse de votre additionneur 1bit.
   (c) Programmez la carte.
   (d) Vérifiez le bon fonctionnement de votre additionneur sur la carte.
3. **Additionneur 4 bits**
   (a) Vérifier le code VHDL du composant additionneur 4 bits fourni.
   (b) Simulez le composant additionneur 4 bits.
4. **Multiplexeur 1 bit**
   (a) Vérifier le code VHDL du composant multiplexeur 1 bit.
   (b) Simulez le composant multiplexeur 1 bit.
5. **Multiplexeur 4 bits**

(a) Décrivez en VHDL un multiplexeur 2 vers 1 de deux nombres de 4 bits. L'entité de ce composant se nommera `mux2v1_4`, les entrées se nommeront `a4`, `b4` et `sel` et la sortie `s4`.

(b) Testez, en le simulant, votre composant.

6. **Additionneur DCB 4 bits**
On désire réaliser un additionneur de nombres codés en DCB, pour cela on utilise l'additionneur 4 bits fourni.

(a) **Détecteur de nombre supérieur à 10**
Soit un nombre binaire codé sur 5 bits, le cinquième bit correspondant à une retenue précédemment générée lors d'une addition de 2 nombres de 4 bits.

i. Déterminez une fonction booléenne permettant en fonction des bits du nombre de 5 bits de savoir si il est supérieur ou égal à 10.

ii. Décrivez en VHDL un composant qui renvoie 1 si le nombre de 5 bits en entrée est supérieur ou égal à 10, qui renvoie 0 sinon. L'entité de ce composant se nommera `detect10`, l'entrée se nommera `a5` et la sortie `s`.

iii. Testez, en le simulant, votre composant.

(b) **Ajustement Décimal**
Afin de n'avoir que des nombres codés en DCB, il est nécessaire de réaliser un ajustement décimal dès lors qu'un nombre est supérieur ou égal à 10.

i. L'ajustement décimal peut se résumer à une addition entre le nombre à ajuster et une valeur constante, déterminez la valeur de cette constante.

ii. Décrivez en VHDL un composant qui réalise l'ajustement décimal. L'entité de ce composant se nommera `ad`, les entrées de ce composant se nommeront `a4` et `cin`, les sorties se nommeront `s4` et `cout`.

iii. Testez, en le simulant, votre composant.

(c) **Assemblage**
Afin de réaliser l'additionneur 4 bits DCB, il faut maintenant assembler tous les éléments, pour obtenir un système correspondant à la figure 19.

i. A l'aide de tous les composants dont vous disposez, décrivez un composant correspondant à l'assemblage de l'ensemble du système. L'entité de ce composant se nommera `add_dcb`, les entrées se nommeront `a4`, `b4` et `cin` et les sorties se nommeront `s` et `cout`.

ii. Testez, en le simulant, votre composant.

7. **Décodeur 7 segments**
Il y a sur la carte spartan3 4 afficheurs 7 segments, ce sont des afficheurs composés de 7 segments de type led, chaque segment pouvant être allumé ou éteint indépendamment des autres.

Tous les segments d'un même afficheur sont reliés à la même anode.

Pour pouvoir allumer un segment d'un des 4 afficheurs, il faut mettre l'anode correspondant à cet afficheur à la valeur logique '0' et la cathode du segment que l'on désire allumer à la valeur logique '0'.

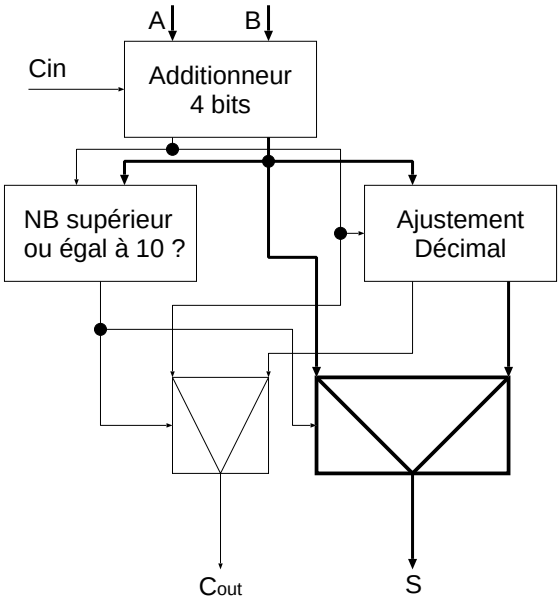


Figure 19: Additionneur DCB

Nous désirons pouvoir afficher sur 1 de ces 4 afficheurs le résultat de l'additionneur DCB, pour cela il faut établir les équations booléennes des 7 segments en fonction des 4 bits qui forme un quartets, la figure 20 donne la correspondance entre un chiffre décimal et un caractère affiché sur l'afficheur.

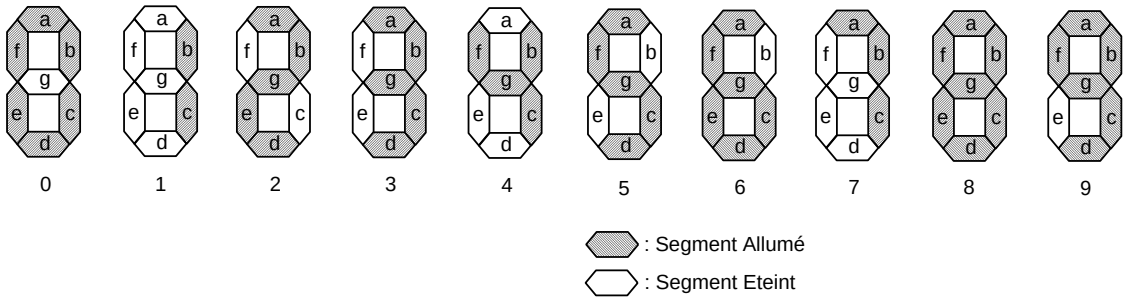


Figure 20: Chiffres décimaux sur un afficheur 7 segments

(a) Vérifiez le code VHDL du composant 7seg fournit.

(b) Compilez et simulez le composant

8. Montage Global

Vous allez maintenant interconnecter l'additionneur DCB avec l'afficheur 7 segments.

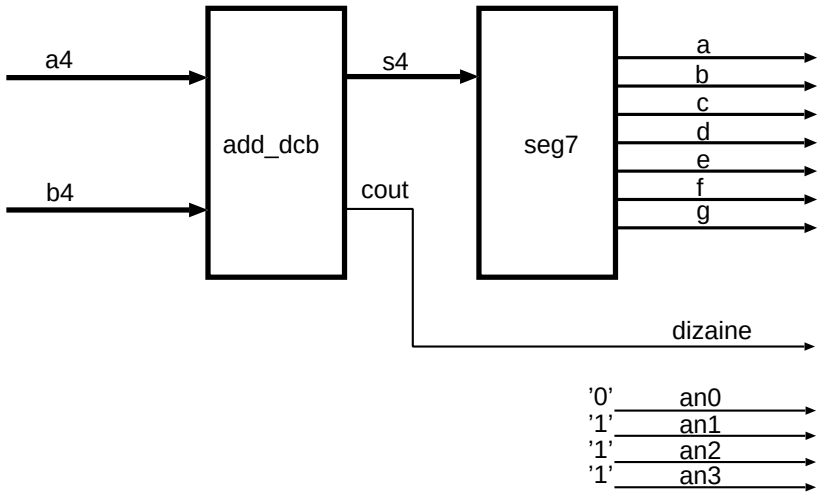


Figure 21: Montage Global

(a) Editez le composant global_DCB et vérifiez le code VHDL.

9. **Mise en oeuvre de l'additionneur DCB sur la carte Spartan3**

On désire mettre en oeuvre sur la carte spartan3 le montage global, pour ce faire vous connecterez les 4 interrupteurs SW0, SW1, SW2 et SW3 sur l'entrée `a4`, les 4 interrupteurs SW4, SW5, SW6 et SW7 sur l'entrée `b4`. Vous connecterez l'afficheur sept segments relié à l'anode AN0 sur les unités résultantes de votre additionneur. Afin de visualiser si l'addition a généré une dizaine ou non, vous connecterez la led LD0 sur le bit de retenue de l'additionneur.

(a) A l'aide de l'outil Précision faites la synthèse de votre montage.

(b) Programmez la carte.

(c) Vérifiez le bon fonctionnement de votre additionneur sur la carte.

Mini-Projet : Rencontre du troisième type

Dans le film de Steven Spielberg "rencontre du troisième type" qui date de 1977, le contact est établi entre des extra-terrestres et des hommes à l'aide de séquences lumineuses et musicales créé à partir de 4 couleurs et 4 sons. Ces séquences apparaissent aléatoires et sont générées par les extra-terrestres, le hommes eux doivent être capables de comprendre le mécanisme de génération des séquences pour les rejouer.

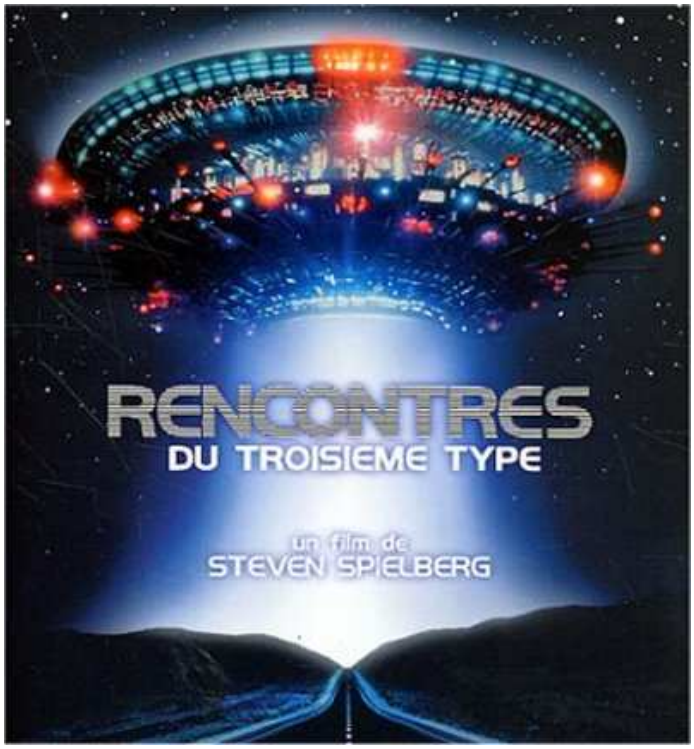


Figure 22: Affiche du Film Recontre du Troisième Type

Nous allons dans ce projet créer des séquences lumineuses pseudo-aléatoires à l'aide de 4 couleurs rouge, bleu, vert et jaune.

Pour réaliser ce projet nous allons procéder en plusieurs étapes, il vous ait fourni ici un cahier des charges détaillé avec des pistes pour vous permettre d'arriver à la réalisation du générateur de séquences lumineuses pseudo-aléatoires.

Afin de réaliser ce projet, le mieux est de le découper en sous-tâches qui sont :

1. Réalisation d'un contrôleur d'affichage VGA
2. Réalisation d'un générateur pseudo-aléatoire
3. Réalisation d'un modulateur temporel de l'affichage

Sous FPGA-Advantage, vous créérez une librairie `Sequence` dans un projet nommé `ET` que vous aurez préalablement créez, et vous mettrez tous vos composants dans cette librairie.

9 Réalisation d'un contrôleur d'affichage VGA

La séquence lumineuse va être affichée sur un écran VGA. Cet écran va être divisé en quatre parties égales qui sont

- HG : Pour la partie Haut à Gauche
- HD : Pour la partie Haut à Droite
- BG : Pour la partie Bas à Gauche
- BD : Pour la partie Bas à Droite

La partie HG sera rouge, la partie HD sera verte, la partie BG sera bleue et la partie BD sera jaune.

Le but final est d'obtenir à l'écran une alternance à l'affichage entre toutes les couleurs affichées en même temps et la couleur choisie par le générateur pseudo-aléatoire, comme représentée par la figure 23. Sur cette figure durant les alternances i et $i+2$ toutes les couleurs sont affichées, et durant les alternances $i+1$ et $i+3$ seule la couleur choisie par le générateur pseudo-aléatoire est affichée. Toutes les alternances ont la même durée.

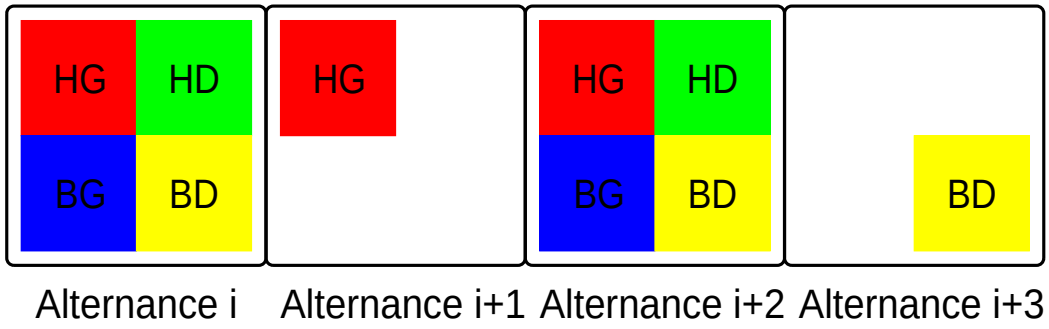


Figure 23: Affichage de la séquence

Afin d'arriver à afficher, vous allez décomposer le problème en trois étapes, la première est d'arriver à afficher une couleur sur tout l'écran, la seconde est d'arriver à afficher 4 couleurs sur l'écran dans les 4 zones HG, HD, BD et BG et la troisième est d'arriver à afficher alternativement les 4 couleurs puis une seule parmi 4 pour cela vous devrez préalablement avoir décrit un générateur de nombres pseudo-aléatoires.

9.1 Contrôleur VGA pour une couleur

La première des choses à faire pour pouvoir afficher est de générer les signaux de synchronisation de l'écran VGA. L'affichage sur un écran VGA est réalisé à l'aide d'un spot lumineux qui traverse un écran, comme on peut le voir sur la figure 24. Il est nécessaire ici d'avoir deux signaux de synchronisation, un signal HS de synchronisation horizontale qui va permettre au spot de passer d'une ligne à une autre, en fait de faire un retour à gauche de l'écran. Un signal VS de synchronisation

verticale qui va permettre un retour du coin en bas à droite de l'écran au coin en haut à gauche de l'écran.

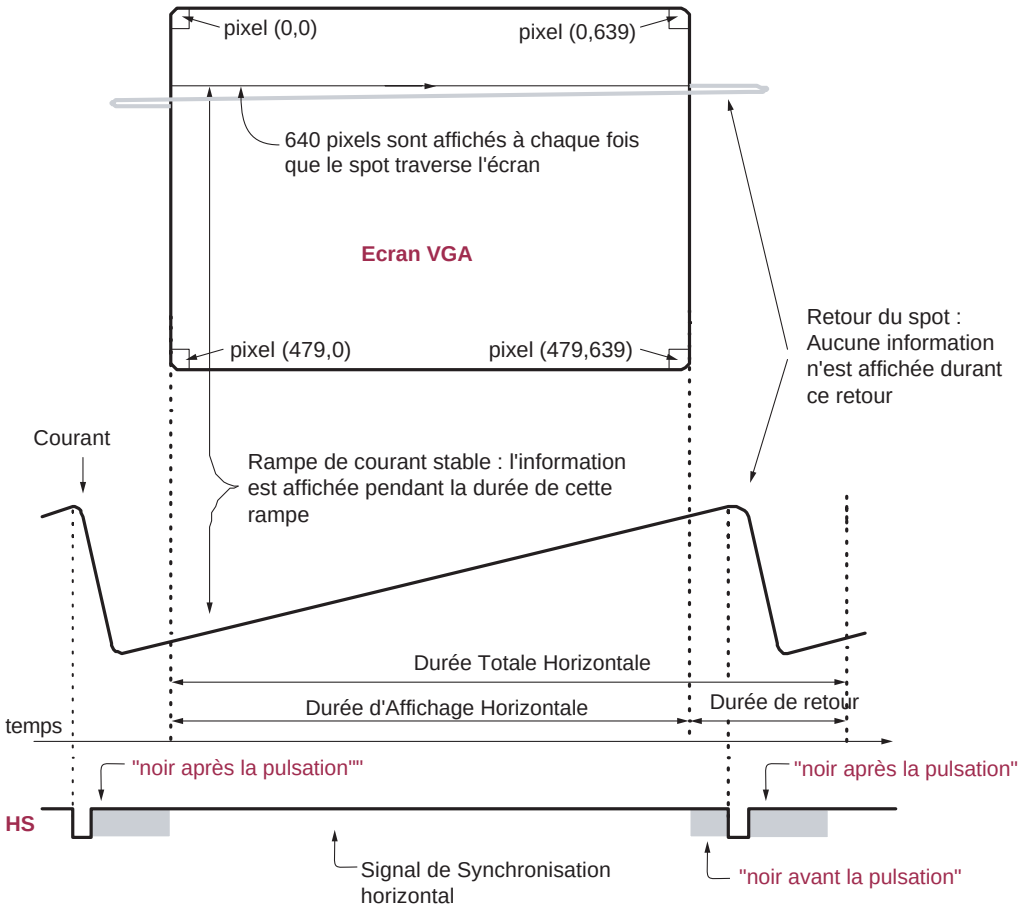


Figure 24: Principe de l'affichage VGA

Les valeurs des temps de synchronisation et d'affichage sont données par la table 1.

Ces valeurs correspondent à un signal de synchronisation, qui a la même forme pour la synchronisation horizontale et la synchronisation verticale, et qui est représenté sur la figure 25.

1. Quel est le composant qui va permettre de mesurer le temps ?
2. En utilisant ce composant, déterminer un système capable de générer HS et VS, vous pourrez pour cela vous aider de comparateurs et de bascules RS asynchrones.
3. Sachant que les temps indiqués dans le tableau 1 sont donnés pour une horloge de $25MHz$ et que l'horloge globale de la carte, récupérable sur la broche $T9$, est une horloge de $50MHz$, décrivez en VHDL un système capable de générer une horloge de $25MHz$ à partir d'une horloge de $50MHz$.

Symbole	Paramètre	Synchro Verticale			Synchro Horizontale	
		Temps	Horloge	Lignes	Temps	Horloge
T_S	Durée synchro	$16,7ms$	416800 cycles	521	$32\mu s$	800 cycles
T_{DISP}	Durée affichage	$15,36ms$	384000 cycles	480	$25,6\mu s$	640 cycles
T_{PW}	Largeur pulsation	$64\mu s$	1600 cycles	2	$3,84\mu s$	96 cycles
T_{FP}	Noir avant pulsation	$320\mu s$	8000 cycles	10	$640ns$	16 cycles
T_{BP}	Noir après pulsation	$928\mu s$	23200 cycles	29	$1,92\mu s$	48 cycles

Table 1: Table des durées de synchronisation

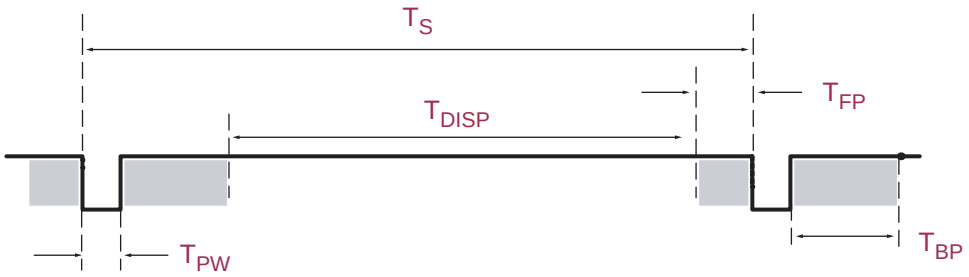


Figure 25: Durée des temps de synchro pour l'affichage VGA

4. Décrivez en VHDL le système capable de générer les signaux HS et VS. Vous ferez attention aux problèmes d'aléas (Glitchs) et vous les corrigerez si il y en a.

5. Compilez et Simulez ce système.

Pour afficher des couleurs sur l'écran, il est nécessaire d'avoir des signaux qui nous indiquent quelle couleur pour quel pixel, c'est le rôle des signaux R, G et B. Pour la carte Spartan3, ces signaux fonctionnent en tout ou rien et les couleurs que l'on peut générer en fonction de la valeur de ces signaux sont résumées dans le tableau 2.

R	G	B	Couleur
0	0	0	Noir
0	0	1	Bleu
0	1	0	Vert
0	1	1	Cyan
1	0	0	Rouge
1	0	1	Magenta
1	1	0	Jaune
1	1	1	Blanc

Table 2: 8 couleurs affichables sur l'écran

Afin que l'affichage soit bien synchronisé, il faut absolument que les signaux R, G et B soit à 0 quand le spot lumineux est en dehors de l'écran (période de retour du spot).

1. Modifiez le système de synchronisation afin d'afficher une couleur uniforme sur l'écran.
2. Décrivez ce système en VHDL.
3. Compilez-le et simulez le.
4. A l'aide de la documentation de la carte, modifiez ce système pour que les signaux soient affectés aux bonnes broches.
5. Faites la synthèse de ce système.
6. Programmez la carte Spartan3 et vérifier le bon fonctionnement de votre système.

9.2 Contrôleur VGA pour 4 couleurs

Une fois que vous savez synchroniser votre écran et afficher une couleur, il faut maintenant découper cet écran en 4 zone telles que celles visibles sur la figure 23.

Pour cela il suffit de repérer le milieu horizontal et le milieu vertical de l'écran. Il faut modifier le système de synchronisation pour qu'il repère ces deux milieux.

1. Calculez la valeur du milieu vertical et du milieu horizontal
2. Modifiez votre système de synchronisation afin qu'il vous indique ces deux milieux
3. Décrivez en VHDL un système capable d'afficher 4 couleurs sur l'écran comme visible sur la figure 23 à l'aide de ces informations de milieu.
4. Compilez et simulez votre système.
5. A l'aide de la documentation de la carte, affectez à ce système les bonnes broches pour les bons signaux.
6. Faites la synthèse de votre système.
7. Programmez la carte Spartan3 et vérifier le bon fonctionnement de votre système.

10 Réalisation d'un générateur pseudo-aléatoire

10.1 Définition d'aprés Wikipédia (http://fr.wikipedia.org/)

Un générateur de nombres pseudo-aléatoires est un système qui génère une séquence de nombres que l'on pense aléatoires. Ceux-ci sont approximativement indépendants les uns des autres, et il est difficile de repérer des comportements de groupe à l'intérieur de ces nombres. Par comportements de groupe, il faut entendre qu'il est difficile de repérer des groupes de nombres qui suivent une certaine règle.

Cependant, les sorties d'un tel générateur ne sont pas entièrement aléatoires. Elles s'approchent toutefois des propriétés idéales des sources complètement aléatoires. John von Neumann insista sur

ce fait avec la remarque suivante : " Quiconque considère des méthodes arithmétiques pour produire des nombres aléatoires est, bien sûr, en train de commettre un péché ". Les nombres aléatoires peuvent être produits avec du matériel qui tire parti de certaines propriétés physiques (bruit d'une résistance par exemple).

La raison pour laquelle on se contente d'un rendu pseudo-aléatoire est : d'une part qu'il est difficile d'obtenir de " vrais " nombres aléatoires et que, dans certaines situations, il est possible d'utiliser des nombres pseudo aléatoires, en lieu et place de vrais nombres aléatoires. D'autre part que ce sont des générateurs particulièrement adaptés à une implémentation électronique, donc plus facilement et plus efficacement utilisables.

Les méthodes pseudo-aléatoires sont souvent employées sur des ordinateurs dans diverses tâches comme la méthode de Monte-Carlo, la simulation ou les applications cryptographiques. Une analyse mathématique rigoureuse est nécessaire pour déterminer le degré d'aléa d'un générateur pseudo-aléatoire. Robert R. Coveyou du Oak Ridge National Laboratory écrivit dans un article que " la génération de nombres aléatoires est trop importante pour être confiée au hasard ".

La plupart des systèmes pseudo-aléatoires essaient de produire des sorties qui sont uniformément distribuées. Une classe très répandue de générateurs utilise une congruence linéaire. D'autres s'inspirent de la suite de Fibonacci en additionnant deux valeurs précédentes ou font appel à des registres à décalage dans lesquels sont injectés le résultat précédent après une transformation intermédiaire. Certains générateurs pseudo-aléatoires sont dits cryptographiques quand certaines contraintes sont satisfaites.

Un des systèmes pour générer des nombres pseudo-aléatoires s'appuie sur des automates cellulaires, une classification de ce type d'objet a été décrit par S. Wolfram en 1983 [1].

10.2 Générateur de nombres pseudo-aléatoire à l'aide d'automate Cellulaire

Dans le cadre de ce projet vous allez utiliser la génération de nombres pseudo-aléatoires à l'aide d'automate cellulaire simple.

10.2.1 Automate Cellulaire simple

L'automate cellulaire simple consiste en une grille unidimensionnelle de cellules ne pouvant prendre que deux états (0 ou 1), avec un voisinage constitué, pour chaque cellule, d'elle-même et des deux cellules qui lui sont adjacentes, une cellule voisine de gauche et une cellule voisine de droite.

Chacune des cellules pouvant prendre deux états, il existe $2^3 = 8$ configurations possibles d'un tel voisinage.

Pour que l'automate cellulaire fonctionne, il faut définir quel doit être l'état, à la génération suivante, d'une cellule pour chacun de ces motifs. Il y a $2^8 = 256$ façons différentes de s'y prendre, soit donc 256 automates cellulaires différents pouvant être simulés sur une telle grille.

Ces automates sont bouclés, c'est à dire que si l'automate a une taille n, la cellule 0 à comme voisin de gauche la cellule $n-1$, et la cellule $n-1$ a comme voisin de droite la cellule 0.

[1] Si vous voulez en savoir plus sur les automates cellulaires, lire le livre de S. Wolfram "A New Kind of Science" paru aux éditions Wolfram Media en 2002.

Considérons l'automate cellulaire défini par la table suivante qui définit la valeur de la cellule centrale à l'instant $t+1$ en fonction de la valeur de la cellule et de ses cellules adjacentes à l'instant t :

Valeur de la cellule et des cellules adjacentes (instant t)	111	110	101	100	011	010	001	000
Valeur suivante de la cellule centrale (instant $t+1$)	0	0	0	1	1	1	1	0

Cela signifie que si par exemple, à un temps t donné, une cellule est à l'état 1 , sa voisine de gauche à l'état 1 et sa voisine de droite à l'état 0, au temps $t+1$ elle sera à l'état 0.

10.2.2 Automate Générateur Pseudo-Aléatoire

Suivant le principe de l'automate cellulaire décrit à la section 10.2.1, il existe des équations qui permette d'avoir un comportement pseudo-aléatoire tel que définit à la section 10.1.

Une de ces règles est donné par l'équation booléenne suivante :

$$c(i) = (c(i-1) + c(i)) \oplus c(i+1)$$

ici $c(i)$ représente une des cellules de l'automate. Pour que cet automate fonctionne il faut qu'il soit tout d'abord initialisé à la valeur 1.

1. A l'aide de cette règle définir le composant de base nécessaire pour réaliser un automate cellulaire. Ce composant devra pouvoir être initialisé de façon asynchrone soit à 0 soit à 1.
2. Décrivez en VHDL un automate cellulaire de taille 16 (ce qui permet d'avoir une séquence pseudo-aléatoire de l'ordre de $2^{16} = 65536$) basé sur la règle définie précédemment.
3. A l'aide de votre automate cellulaire de taille 16 décrivez un composant qui permet de n'avoir que 2 sorties pseudo-aléatoires.
4. Compilez et testez votre composant.
5. Faites la synthèse et programmez votre composant sur la carte Spartan3 en mettant les 2 sorties sur 2 leds.

Ce générateur une fois réalisé va nous permettre de contrôler quelle couleur afficher, 1 parmi 4 (rouge, vert, bleu et jaune) sur un écran VGA.

11 Réalisation d'un modulateur temporel de l'affichage

Dans "Rencontre du troisième Type" la séquence semble aléatoire et accélère au fur et à mesure du temps, il vous faut donc maintenant pouvoir alterner aléatoirement les couleurs et accélérer leur affichage, c'est ce que nous vous proposons de réaliser.

11.1 Contrôleur VGA pour alternance 4 couleurs - 1 couleur

Il faut interconnecter le générateur de nombre pseudo-aléatoires et le contrôleur VGA afin de réaliser l'alternance.

Pour éviter tout phénomène d'affichage hasardeux, il est nécessaire de synchroniser le générateur pseudo-aléatoire à l'aide d'une horloge qui ait une fréquence multiple de celle de l'affichage d'une image 640x480.

Il faut ensuite déterminer un signal périodique, dont la fréquence est aussi une fréquence multiple de celle de l'affichage d'une image 640x480. Ce signal commandera l'alternance, quant il sera à 1, alors les 4 couleurs seront affichées simultanément sur l'écran, quant il sera à 0 ne sera affiché à l'écran que la couleur déterminée par le générateur pseudo-aléatoire.

1. Décrivez en VHDL un système qui permet cet affichage alterné.
2. Compilez et simulez votre système.
3. Affectez les bonnes broches à votre système.
4. Faite la synthèse de votre système.
5. Programmez votre système sur la carte et vérifiez son bon fonctionnement.

11.2 Accélération de l'affichage

1. Modifiez votre système pour que périodiquement l'alternance d'affichage s'accélère, on supposera qu'une fois arrivé à l'alternance la plus rapide, l'alternance suivante sera la plus longue pour former ainsi un cycle. On pourra dans un premier temps utiliser un des interrupteurs de la carte pour passer d'une fréquence d'alternance à une autre.
2. Décrivez en VHDL un tel système.
3. Compilez et Simulez votre système.
4. Affectez les bonnes broches.
5. Faite la synthèse de votre système.
6. Programmez votre système sur la carte et vérifiez son bon fonctionnement.

LICENCE SCIENCES & TECHNOLOGIES
MENTION ELECTRONIQUE

UNIVERSITE PIERRE & MARIE CURIE
LA SCIENCE A PARIS

Documentation de la carte