TD1

PRISE EN MAIN DU FLOT DE CONCEPTION

Quartus II

Objectif :

L'objectif de ce TD est de prendre en main les outils de conception Quartus. Vous allez apprendre dans ce TD un flow de conception de type top-down, c'est-à-dire de la spécification à la synthèse de composants décrits en VHDL vers la simulation.

Le TD est divisé en deux parties, la première porte sur les outils et la carte FPGA, tandis que la deuxième partie, porte sur des exercices d'application.

Partie I. Prise en main de Quartus II

I. Création d'un Projet Quartus

Un système est composé de plusieurs composants exécutant chacun une fonction spécifique. Chaque composant est conçu, compilé et simulé seul avant d'être relié aux autres pour exécuter des fonctions plus complexes. Pour créer un composant, vous devez créer un projet.

1. Lancer Quartus II.

2. Menu *File* et *New Project Wizard*. Cet assistant vous guide pas à pas pour créer votre projet.

Remarque : Sous Quartus II, un projet est lié à une cibles matérielles (FPGA ou CPLD) parce que les simulations sont *post-synthèse* par défaut, c'est-à-dire qu'elles prennent en compte les temps de propagation à l'intérieur du circuit (qui sont variables selon le FPGA considéré). La référence est écrite sur la puce !

3. Créez un répertoire *IUT_FPGA* sur la racine de votre compte et nommez votre premier projet *TD1*.

4. Vous pouvez observer la composition de votre projet (entité de haut niveau et circuit utilisé).

II. Edition et compilation

Une fois le projet créé, vous pouvez insérer plusieurs fichiers de description de composants : des descriptions structurelles à l'aide de fichier BDF (Block Diagram File) ou des descriptions dataflow ou comportementales (fichiers VHDL, AHDL ou Verilog).

1. Description Dataflow

1. Menu *File*, *New*.

2. L'éditeur du fichier *Vhd1.vhd* apparaît. Avant d'écrire votre code, enregistrez le fichier.

Attention : le nom du fichier correspond au nom que vous donnerez à votre entité.

3. Ecrivez le code VHDL d'une porte NAND. Le nom de l'entity sera *Porte_NAND* et le fichier doit porter le même nom que l'entity (donc *Porte_NAND.vhd*). Une fois enregistré, le fichier s'ouvre dans le Project Navigator.

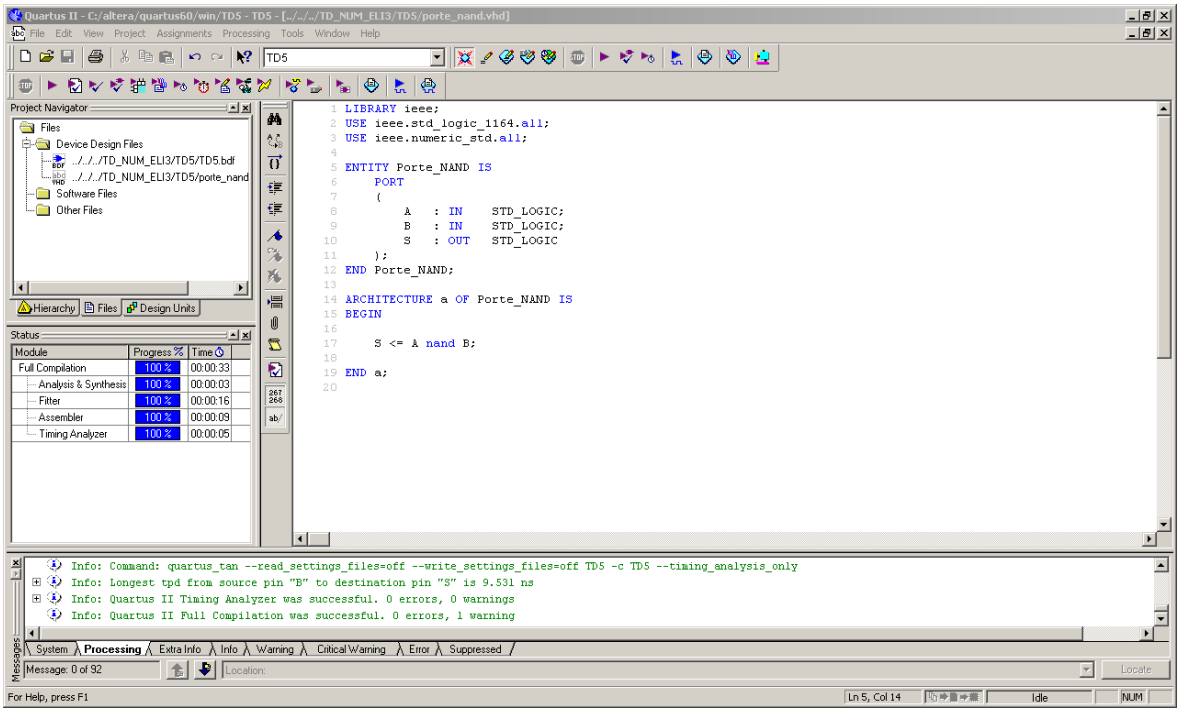


Figure 1 : Description dataflow.

2. Description structurelle

1. Menu *File*, *New*, sélectionnez *Block Diagram/Schematic File*. Une fenêtre s'ouvre. Enregistrez votre fichier sous le nom *Porte_NAND_SCHEME.**bdf***

2. Quartus II vous propose une librairie de composants décrits en VHDL. Pour les utiliser, cliquez sur la commande ***Symbol Tool*** (symbole d'une porte ET).

3. Une fenêtre s'ouvre. Prenez le composant *porte NAND* (nand2) et placez-le dans votre première fenêtre.

Remarque : Afin de pouvoir être simulé, il faut adjoindre à la porte des entrées/sorties physiques.

4. Sélectionnez dans la fenêtre ***symbole***, les composants **input** et **output**. Reliez les entrées/sorties du composant avec des fils (les signaux 1 bit correspondent à la commande WIRE). Enregistrez votre fichier.

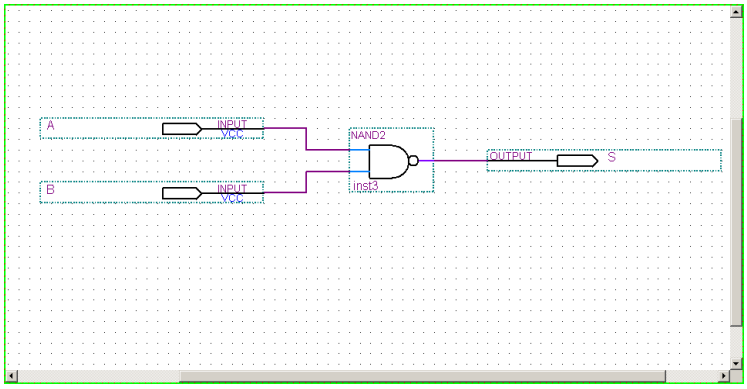


Figure 2 : Description structurelle

3. Compilation

Avant de simuler un circuit, vous devez vérifier s'il ne comporte pas d'erreur de syntaxe. Le compilateur est là pour ça !
Remarque : Si votre projet comporte plusieurs descriptions, le compilateur synthétise l'entité de plus haut niveau. En effet, elle décrit tout le système.

1. Onglet *Files* du Project Navigator, sélectionnez le fichier à compiler. Clic droit et *Set as Top-Level Entity*. Ainsi, le compilateur synthétise uniquement ce composant.

2. Menu *Processing* et *Start Compliation*. A la fin du processus, le rapport de synthèse et de placement routage s'ouvre et vous résume les ressources nécessaires pour l'implémentation matérielle.

III. Simulation

Cette étape consiste à vérifier le comportement du composant créé. Le simulateur permet de vérifier le comportement temporel et fonctionnel de descriptions dataflow, structurelles et comportementales.
Attention : cette étape nécessite la compilation du circuit.

1. Menu *File, New*, Onglet *Other files* et *University Program VWF*. Ce fichier décrit visuellement le test bench que vous allez utiliser pour tester votre circuit. Ce fichier permet de lancer le simulateur Modelsim en tâche de fond.

2. Enregistrez le fichier **Bench_*nom*** du composant. Par exemple pour la bascule D, le nom du test bench sera **Bench_Porte_NAND.vwf**.

3. Le fichier de simulation est divisé en deux parties. Une colonne pour les broches d'entrées/sorties du composant, et une zone graphique munie d'une échelle temporelle.

Dans la colonne *Name*, clic droit, *Insert a node or bus*, puis l'outil *Node Finder*. Cet outil permet de récupérer les noms des entrées/sorties du composant.

4. Sélectionner les signaux A, B et S. et terminez l'opération.

5. A et B sont des entrées. Vous pouvez leur affecter des valeurs manuellement ou utiliser des stimulis prédéfinis. La barre d'outils Waveform Editor prédéfinit plusieurs types de signaux (High, Low, Overwrite Clock, etc.). Afin de couvrir l'ensemble des combinaisons de A et B, pré-positionnez les valeurs des entrées comme sur la figure ci-dessous.

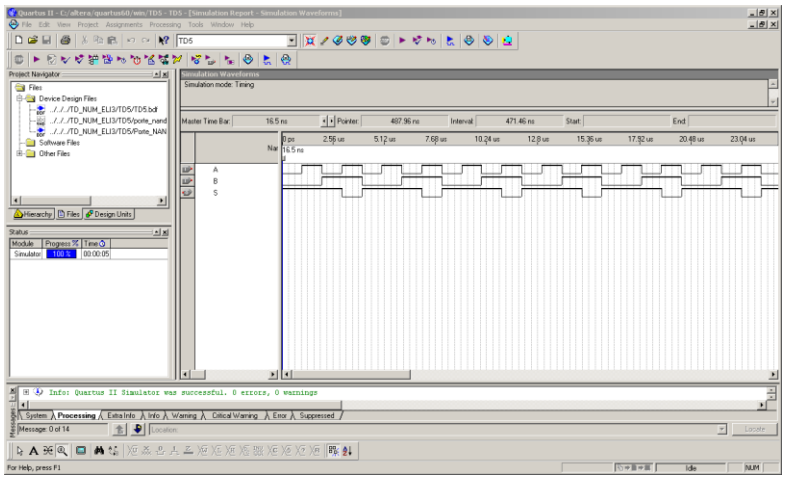


Figure 9 : Résultat de simulation

6. Vous allez simuler le comportement du circuit avec le test bench que vous venez de réaliser. Pour cela, réglez les paramètres de simulation et lancez la simulation. Vérifiez le résultat obtenu.

IV. Implémentation du code sur la puce

Vous avez conçu, codé et simulé votre système. Tout va bien, le cahier des charges semble respecté. Dernière étape : l'implémentation du code sur la carte FPGA. Quartus II fournit les outils pour programmer la carte et pour vérifier le système en fonctionnement sur la carte.

1. Reliez les entrées/sorties de votre description VHDL aux entrées/sorties du FPGA. A l'aide du manuel de la carte Cyclone III, déterminez quelles sont les éléments que vous voulez utiliser (boutons, switch, LED, etc…). Pour l'exemple de la porte NAND, utilisez pour entrées les switches SW0 et SW1. Le résultat (sortie S) sera connecté directement à la ledG7.

Remarque : Pour affecter une entrée/sortie d'une description VHDL à une entrée/sortie physique, menu *Assignments* puis *Pin planner.*

2. Recompilez votre circuit pour prises en compte des assignations. Après les opérations de synthèse et de placement/routage, un fichier .sof (SRAM Object Files) est généré pour programmer le circuit.

3. Menu *Tools*, sélectionner *Programmer* pour télécharger le fichier .sof sur le FPGA.

4. Avec les switch SW0 et SW1, vérifiez le comportement de la porte NAND décrite en VHDL au moyen de la LED G7.

Partie II. Applications

Maintenant que vous maîtrisez l'environnement de développement, c'est à vous de jouer !

Exercice 1 – Contrôle de LEDs

On désire connecter les 10 interrupteurs aux 10 LEDs de la carte. Nous allons utiliser les instructions VHDL d'assignation inconditionnelle.

LED(0)<=SW(0) ;
LED(1)<=SW(1) ;
…
LED(9)<=SW(9) ;

Une partie du code VHDL est donné ci-dessous :

```
LIBRARY ieee;
USE ieee.std logic 1164.all;


ENTITY switch_led IS

PORT (
      SW : IN STD_LOGIC_VECTOR(9 DOWNTO 0);
      LED : OUT STD_LOGIC_VECTOR(9 DOWNTO 0) pas de « ; »
        );
END switch_led;

ARCHITECTURE Behavior OF switch_led IS
BEGIN
     LED <= SW;
END Behavior ;
```

1. Créez un nouveau projet LAB1

2. Créez un nouveau fichier VHD que vous nommez *switch_led.vhd* , recopier le code fourni, compilez et créez une représentation schématique (symbole).

3. Créez un fichier LAB1.bdf et incluez le symbole *switchled*.

4. A partir de la datasheet de la carte (voir p. 24 et 25 du fichier *DE0_reference_manual.pdf*), assignez les broches d'entrée/sortie.

5. Testez sur la carte.

Exercice 2 – Multiplexeur 8 bits 2->1

On veut réaliser le multiplexeur 8bits 2->1 en VHDL. Utilisez le switch SW9 pour la commande, les switch 0 à 7 pour l'entrée X et la valeur constante 10101010 pour Y.

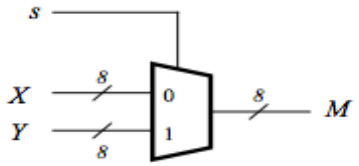


Figure 3 : Multiplexeur 8 bits 2->1

Une partie du code VHDL est donné ci-dessous :

```
LIBRARY ieee;
USE ieee.std logic 1164.all;


ENTITY mux21 IS

PORT (
        X : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
        Y : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
        s : IN STD_LOGIC;
        LED : OUT STD LOGIC_VECTOR(7 DOWNTO 0)
        );
END mux21;

ARCHITECTURE Behavior OF mux21 IS
BEGIN
        LED<= X when s='0' else Y ;
END Behavior
```

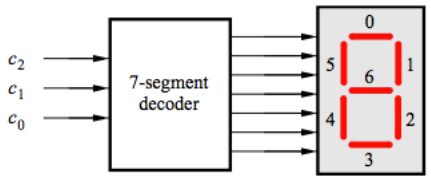
1. Créez un fichier VHD que vous nommerez *mux21.vhd*, recopiez le code, compilez et créez un symbole.

2. A partir de la datasheet de la carte (voir p. 24 et 25 du fichier *DE0_reference_manual.pdf*), assignez les broches d'entrée/sortie.

3. Testez sur la carte.

Exercice 3 – Décodeur binaire + Afficheur 7 segments

On veut réaliser un décodeur binaire et un affichage 7 segments qui affichent les caractères H, E, L et O en fonction des commandes c2, c1 et c0.



$c_2c_1c_0$	Character
000	H
001	E
010	L
011	O
100	
101	
110	
111	

1. Déterminez les équations des segments 0, 1, 2, 3, 4, 5 et 6 en fonction des commandes c2, c1 et c0.

2. Créez un fichier VHD *helo.vhd* et complétez le code donné ci-dessous.

```
LIBRARY ieee;
USE ieee.std logic 1164.all;

ENTITY helo IS
PORT (
        c2 : IN STD_LOGIC;
        c1 : IN STD_LOGIC;
        c0 : IN STD_LOGIC;

        a : OUT STD_LOGIC ;
        b : OUT STD_LOGIC ;
        c : OUT STD_LOGIC ;
        d : OUT STD_LOGIC ;
        e : OUT STD_LOGIC ;
        f : OUT STD_LOGIC ;
        g : OUT STD_LOGIC
        );
END;

ARCHITECTURE Behavior OF helo IS
BEGIN
        a<= --à compléter;
        b<= --à compléter;
        c<= --à compléter;
        d<= --à compléter;
        e<= --à compléter;
        f<= --à compléter;
        g<= --à compléter;

END Behavior
```

3. Créez un symbole helo.

4. Les commandes c2, c1 et c0 sont respectivement les switches SW2, SW1 et SW0.

5. Testez sur la carte.

Exercice 4 : Message sur 4 afficheurs 7 segments

On veut afficher le message HELO sur 4 afficheurs (HEX0, HEX1, HEX2 et HEX3 – cf. *DE0_ref_manual*). Pour cela, nous allons concevoir un contrôleur spécifique ayant une commande c sur 2 bits et dont les sorties (a2, a1, a0, b2, b1, b0, c2, c1, c0, d2, d1, d0) seront connectées aux entrées du décodeur définit dans l'exercice 3. La table de vérité du contrôleur est donnée ci-dessous.

c1	c0
0	0
0	1
1	0
1	1

a2	a1	a0	b2	b1	b0	c2	c1	c0	d2	d1	d0
0	0	0	0	0	1	0	1	0	0	1	1
0	1	1	0	0	0	0	0	1	0	1	0
0	1	0	0	1	1	0	0	0	0	0	1
0	0	1	0	1	0	0	1	1	0	0	0

1. Déterminez les équations du contrôleur
2. Créez un fichier VHD *controleur.vhd* et écrire le code VHDL.
3. Créez un symbole et connectez les sorties aux entrées de quatre blocs HELO.
4. Les commandes c1 et c0 sont respectivement les switches SW1 et SW0.
5. Testez sur la carte.