ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

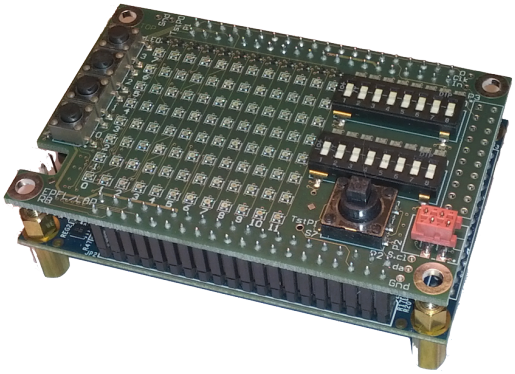
Introduction à la carte FPGA4U

1 Introduction

Durant le premier semestre vous avez appris à modéliser des systèmes simples à l'aide de Quartus en y créant des schémas blocs. Il n'est toutefois pas aisé de concevoir un système complexe avec des schémas. C'est ici que le VHDL intervient, il va permettre de décrire ce que le matériel doit faire et Quartus s'occupera d'en faire la synthèse en schéma logique.

Avant toute chose, nous allons nous assurer que votre disque réseau *MyFiles* a correctement été connecté, pour ce faire veuillez suivre le tutoriel sur `http://wiki.epfl.ch/myfiles-windows` .

2 FPGA4U DE0-nano Extension



Le laboratoire LAP [1] de l'EPFL a mis au point une carte d'extension pour la carte de développement Terasic DE0-nano qui vous permet de concevoir un système logique et d'en faire la synthèse à l'aide de Quartus II et d'y télécharger sur une FPGA.

2.1 Composants

Nous n'aurons malheureusement pas la chance de pouvoir utiliser tous les composants que la carte fournie, voici la liste de ceux que nous utiliserons :

- Une matrice de 96 LEDs RGB (12x8), soit 12 lignes de 8 LEDs.
- 4 Bouttons poussoirs
- 2 Switch 8 bits

Vous pouvez trouvez la liste d'affectation des pins à l'adresse ci-dessous, celle-ci vous sera utile pour faire le lien entre les entrées/sorties de votre programme et la carte.

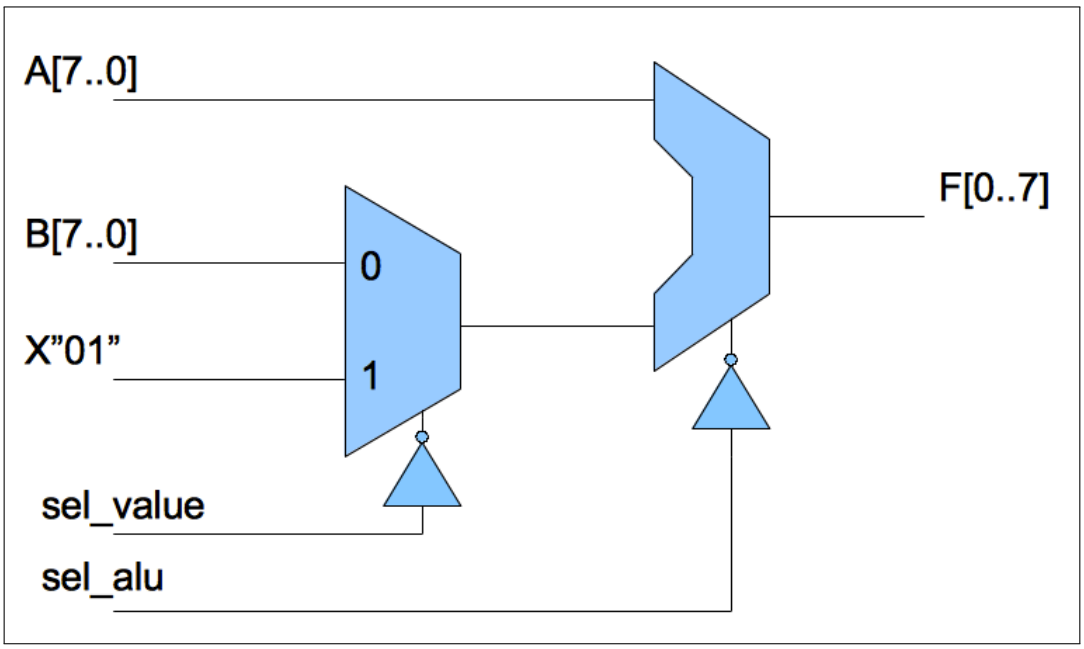
`http://fpga4u.epfl.ch/`

1. `http://lap.epfl.ch/`

3 Laboratoire

Le laboratoire de cette semaine a pour but de vous initier à la carte FPGA4U. Pour vous montrer la force de Quartus et la possiblité de lier *VHDL* et *Schéma bloc*, nous allons créer les composants complexes en VHDL et les inter-connecter à l'aide d'un schéma bloc.

Vous allez avoir la tâche de concevoir une ALU 8 bits, la conception qui a été choisie ici aurait pu être simplifiée, cependant pour bien vous faire comprendre qu'il est facile de lier VHDL et Schéma bloc, nous allons la rendre un peu moins "logique". Voici le schéma logique du système que vous allez devoir concevoir, trois des composants seront implémentés en VHDL (la constante *X"01"*, le multiplexeur et l'ALU).



Aucun document ne vous sera fourni pour ce laboratoire, cependant vous devez télécharger le module monochrome *FPGA4U RGB LEDs 96* permettant de connecter les LEDs, sur le site de la *FPGA4U*.

3.1 Créer un projet

Voici les informations dont vous avez besoin pour créer un projet, les étapes sont les mêmes qu'au semestre d'automne.

1. Directory, Name, Top-Level Entity
   - **Working Directory :** Z :/lab01/
   - **Project Name :** lab01
   - **Top-Level Entity :** lab01
2. Add Files
   - rgb_led96.vhd (*FPGA4U RGB LEDs 96*) : A mettre à la base du répertoire du projet.
3. Family & Device Settings
   - **Family :** Cyclone IV E
   - **Device :** EP4CE22F17C6
4. EDA Tool Settings
   - **Design Entry/Synthesis :** Precision Synthesis
   - **Simulation :** Altera-ModelSim → VHDL

3.2 Création des composants en VHDL

3.2.1 Création d'un multiplexeur 2 bits

Créer un nouveau document VHDL (*File → New... → Design Files → VHDL File*) et enregistrez-le dans le document mux2.vhd

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  entity mux2 is
5     port(
6        A, B    : in std_logic_vector(7 downto 0);
7        sel     : in std_logic;
8        output  : out std_logic_vector(7 downto 0)
9    );
10 end;
11
12 architecture behaviour of mux2 is
13 begin
14    with sel select
15       output <= A when '0',
16                 B when '1',
17                 X"00" when others;
18
19 end behaviour;
```

Listing 1 – mux2.vhd

3.2.2 Création de l'ALU 8 bits

Voici le code pour l'ALU 8 bits, créer le fichier ainsi que le composant comme pour le multiplexeur 2 bits.

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.std_logic_arith.all;
4  use ieee.std_logic_unsigned.all;
5
6  entity alu8 is
7     port(
8        A, B    : in std_logic_vector(7 downto 0);
9        sel     : in std_logic;
10       output    : out std_logic_vector(7 downto 0)
11    );
12 end;
13
14 architecture behaviour of alu8 is
15 begin
16    with sel select
17       output <= A + B when '0',
18                 A - B when '1',
19                 X"00" when others;
```

```
20
21 end behaviour;
```

Listing 2 – alu8bits.vhd

3.2.3 Création d'une constante

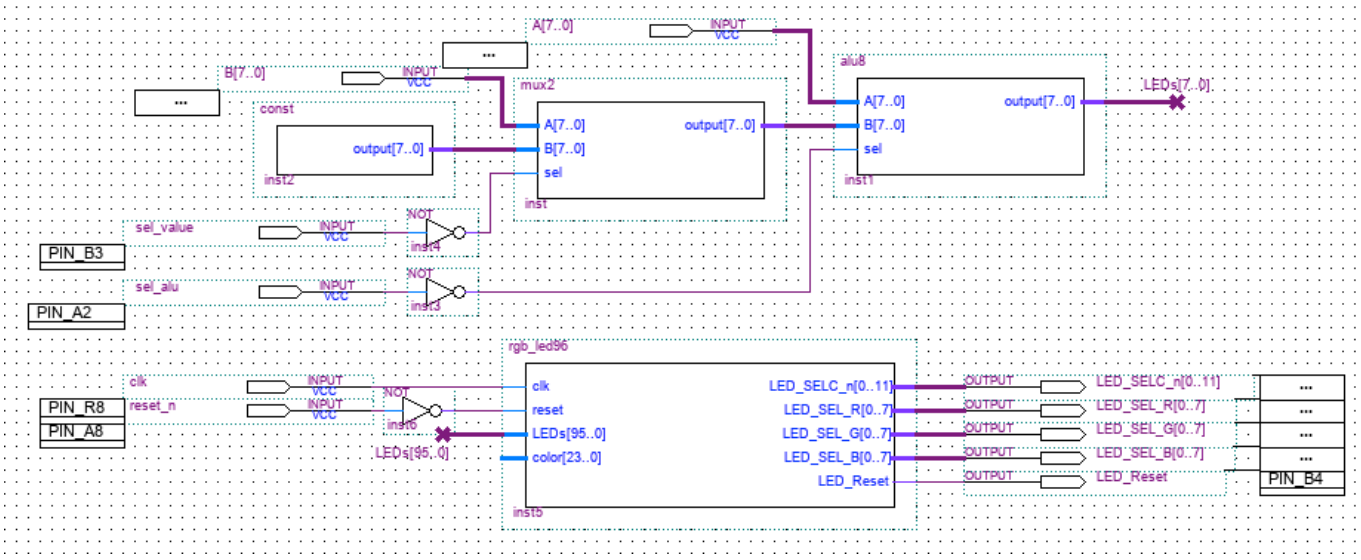
Pour insérer la constante *X"01"* de notre schéma, nous allons créer un composant pour le faire. Comme précédemment, créez un document VHDL *const.vhd* avec le code suivant :

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  entity const is
5      port(
6          output: out std_logic_vector(7 downto 0)
7      );
8  end;
9
10 architecture behaviour of const is
11 begin
12     output <= X"01";
13 end behaviour;
```

Listing 3 – const.vhd

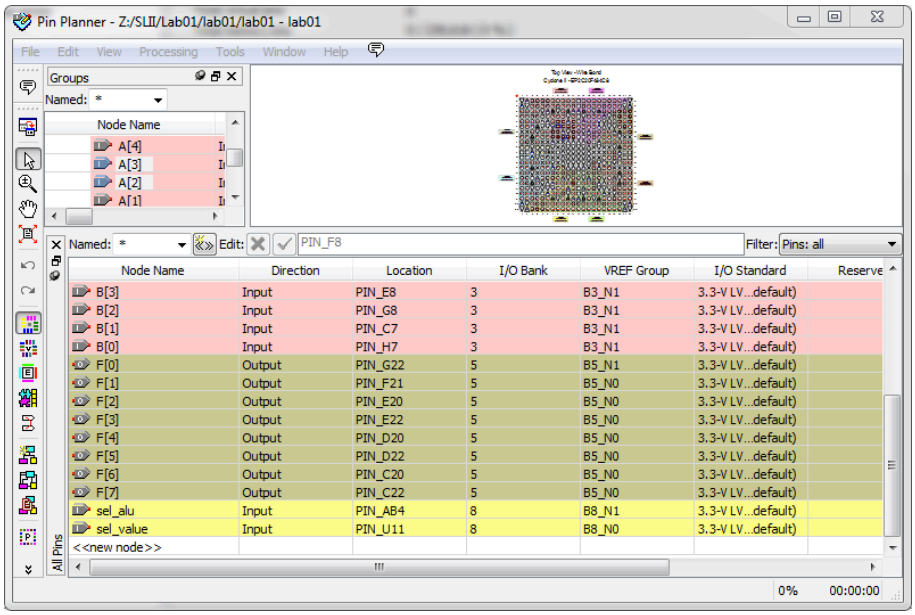
3.2.4 Lier les composants

Générez chacun des composants avec *File → Create / Update → Create Symbol Files for Current File*. Créez un schéma bloc du nom de *lab01* où vous y relierez tous les composants comme sur le schéma ci-dessous, faites attention aux noms des composants, insérez-y également le composant *rgb_led96* qui permettra de connecter les LEDs.
Finalement, créez les pins d'entrées/sorties (*input* / *output*) comme sur le schéma ci-dessous.



3.3 Affectation des pins

Il va maintenant falloir relier nos pins avec nos entrées/sorties.
Avant tout, commencez par compiler votre projet et allez dans *Assignments → Device → Device and Pin Options...* et mettez *Reserve all unused pins* à *As input tri-stated*



Pour ce faire aller dans *Assignments → Pin Planner*, vous devriez alors avoir une fenêtre comme sur l'image ci-dessous, il ne vous reste plus qu'à remplir les *Location* avec les valeurs que vous pouvez trouver dans le document Excel d'affectation des pins sur le site de la carte FPGA4U.

- *clk* → Clock (PIN_R8)
- *reset_N* → Button_n[0] (PIN_A8)
- *sel_alu* → Button_n[2] (PIN_A2)
- *sel_value* → Button_n[3] (PIN_B3)
- *A[0..7]* → Sw_LedA[0..7]
- *B[0..7]* → Sw_LedB[0..7]
- *LED_Reset* → LED_Reset (PIN_B4)
- *LED_SEL_R[0..7]* → LED_Sel_R[0..7]
- *LED_SEL_B[0..7]* → LED_Sel_B[0..7]
- *LED_SEL_G[0..7]* → LED_Sel_G[0..7]
- *LED_SELC_n[0..11]* → LED_SelC_n[0..11]

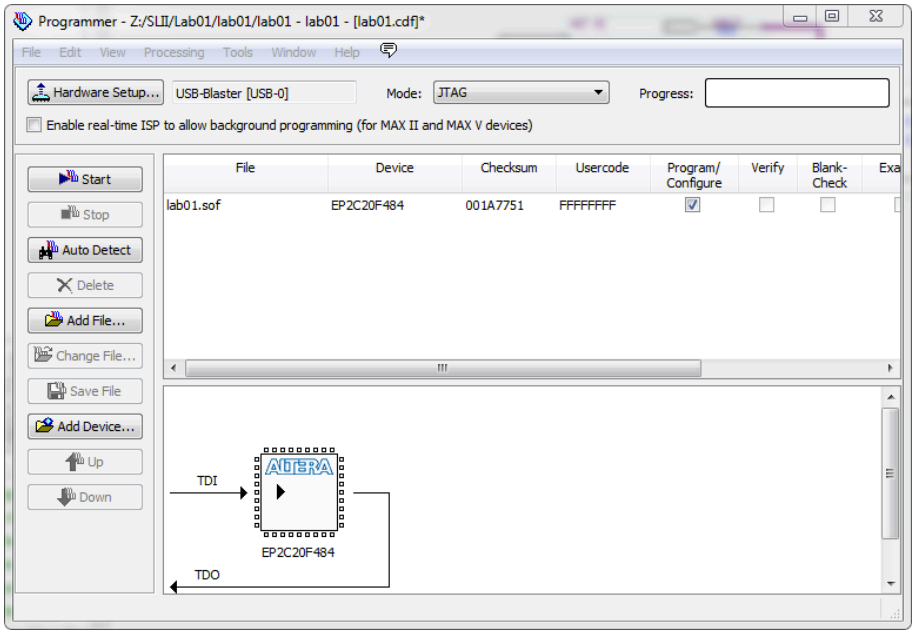
Une fois l'affectation finie, vous devez à nouveau recompiler le projet pour prendre en compte les modifications.

3.4 Programmer la carte FPGA4U

Branchez votre carte au port USB, celle-ci devrait afficher "FPGA4U" sur la matrice de LEDs.
Afin de charger votre programme sur la carte, ouvrez le programmeur avec *Tools → Programmer* puis vérifiez que la carte a été sélectionnée, il doit y être écrit : *USB-Blaster*, sinon sélectionnez-là dans *Hardware Setup... → Currently selected hardware*. Pour lancer le chargement appuyez sur *Start*, une fois celui-ci à 100%, le programme est automatiquement lancé sur votre carte.

1. Entrez un nombre en binaire sur le *switch 0* (celui du haut)
2. Entrez un nombre en binaire sur le *switch 1* (celui du bas)

3. On peut voir l'addition $A + B$ sur la matrice de LEDs (ligne du haut)
4. Appuyez sur le *bouton 0* pour remplacer B par 1 qui affiche $A + 1$
5. Appuyer sur le *bouton 3* pour remplacer l'addition par la soustraction qui affiche $A - B$



4 Simulation

Il est toujours possible d'effectuer des simulations à l'aide de *ModelSim* comme décrit dans les laboratoires du semestre précédent.