TP VHDL 1 : Découverte de l'environnement VIVADO 2025.1

Introduction :

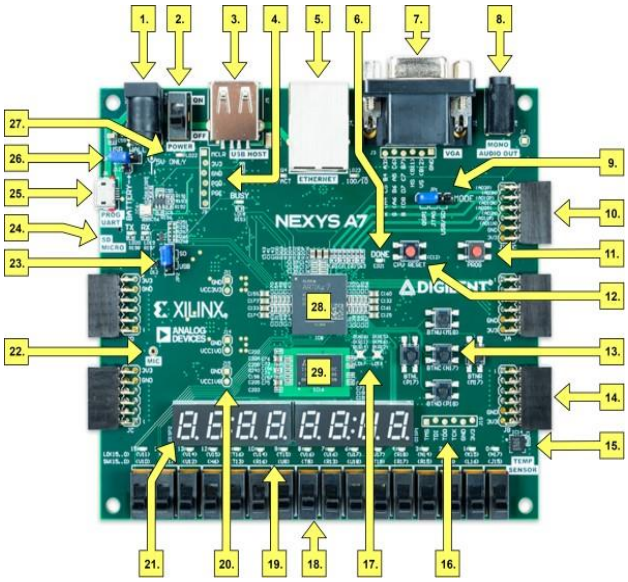
VIVADO 2025.1 est un IDE (*Integrated Development Environment*) ou EDI (Environnement de Développement Intégré en français), c'est-à-dire une interface qui permet de développer, compiler et exécuter un programme dans un langage donné. La carte de développement que nous utiliserons est la carte de développement NEXYS-A7 construite autour d'un FPGA Artix 7.

Présentation de la Carte :

La documentation de la carte NEXYS-A7 Artix-7 est sur Moodle.

Caractéristiques :

- FPGA Artix A7™ (XC7A100T-1CSG324C)
- 128 MB de mémoire RAM de type DDR2
- Ethernet 10/100 PHY via RJ45
- Alimentation sur USB ou toute alimentation externe 4,5-5,5 V
- Oscillateur 100 MHz
- 4 connecteurs d'extensions pour utilisation de modules optionnels "PMOD"
- 1 connecteur d'extension pour signal XADC
- Sortie port VGA 12 bit
- Connecteur port USB (par exemple pour clavier, souris ou clef USB) et micro SD
- 8 afficheurs 7 segments à Leds
- 16 Leds
- 16 interrupteurs et 5 boutons-poussoir
- 1 capteur de température et 1 accéléromètre



Programmation du FPGA :

La programmation du FPGA peut être effectuée soit à partir d'un PC via le port USB, soit à partir d'une carte micro SD. Le cavalier JP2 (SD, USB) permet de choisir la configuration (USB ou micro SD). Le FPGA peut automatiquement charger une configuration à partir de la Plateforme Flash à la mise sous tension si le cavalier de mode de configuration JP1 est réglé sur "Master série ". Si le cavalier Mode est réglé sur "JTAG", le FPGA attendra la programmation à partir du PC (via le câble USB).

- Connecter la carte NEXYS-A7 au PC via le câble USB et la mettre sous tension. La carte est alors en mode de démonstration.
- Positionner le cavalier JP3 (SELECT POWER) sur USB
- Positionner le cavalier JP1 (MODE) en position QSPI/JTAG

1 – Tutorial : Inverter

Cette première partie a pour objectif de se familiariser avec la programmation de la carte NEXYS A7. Ouvrir le document **Tutorial VIVADO 2025.1** sur Moodle et exécuter l'ensemble des 7 étapes.

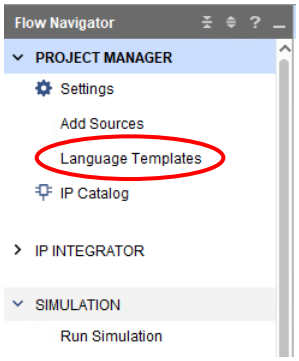
2 - Compteur 4 bits

Cette deuxième partie a pour objectif de décrire un compteur 4 bits, chaque bit étant relié à une LED de la carte. Un interrupteur permettra d'incrémenter ou décrémenter le compteur.
Suivre les étapes comme indiqué dans le tutorial de l'inverseur en tenant compte des éléments suivants :

✓ Nommer le fichier **counter_4bits** et indiquer les E/S suivantes :

- CLOCK : in
- DIRECTION : in
- COUNT_OUT[3 :0] : out

✓ Dans la fenêtre **Flow Navigator**, cliquer sur « **Language Templates** » du premier bloc **« Project Manager ».** Sélectionner **VHDL,** puis **Synthesis Constructs / Coding Examples /Counters / Binary / Up-Down Counters / Simple Counter**



✓ Copier le code de la fenêtre à droite et l'insérer dans le fichier **counter_4bits.vhd**.

✓ Modifier ensuite le code comme ci-dessous :

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3  use IEEE.NUMERIC_STD.ALL;       -- à rajouter pour permettre le comptage
4
5  entity counter_4bits is
6  Port (  CLOCK : in  STD_LOGIC;
7          DIRECTION : in  STD_LOGIC;
8          COUNT_OUT : out  STD_LOGIC_VECTOR (3 downto 0));
9  end counter_4bits;
10
11 architecture Behavioral of counter_4bits is
12 signal count_int : unsigned (3 downto 0) :="0000"; --ou (others=>'0')
13 begin
14 process (CLOCK)
15 begin
16     if rising_edge(CLOCK) then
17         if DIRECTION='1' then
18             count_int <= count_int + 1;
19         else
20             count_int <= count_int - 1;
21         end if;
22     end if;
23 end process;
24
25 COUNT_OUT <= std_logic_vector(count_int);
26
27 end Behavioral;
```

✓ Créer le fichier de simulation **tb_counter_4bits** et effectuer la simulation fonctionnelle

✓ Définition des entrées-sorties (I/O Floorplaner) :
Les E/S sont de type IOS STANDARD LVCMOS33.
- CLK : E3
- DIRECTION : J15
- COUNT_OUT(0) : H17
- COUNT_OUT(1) : K15
- COUNT_OUT(2) : J13
- COUNT_OUT(3) : N14

✓ Dérouler le flot de conception et programmer le FPGA

✓ Observer le fonctionnement du compteur 4 bits sur la carte et proposer une solution pour l'améliorer.

2 – Etude d'un chronomètre

L'entité principale a pour objectifs :

• l'affichage des dixièmes de secondes sur une barette de **leds** (chenillard)

• l'affichage des secondes et dizaines de secondes sur **deux afficheurs 7 segments**

Sur la carte, on dispose :

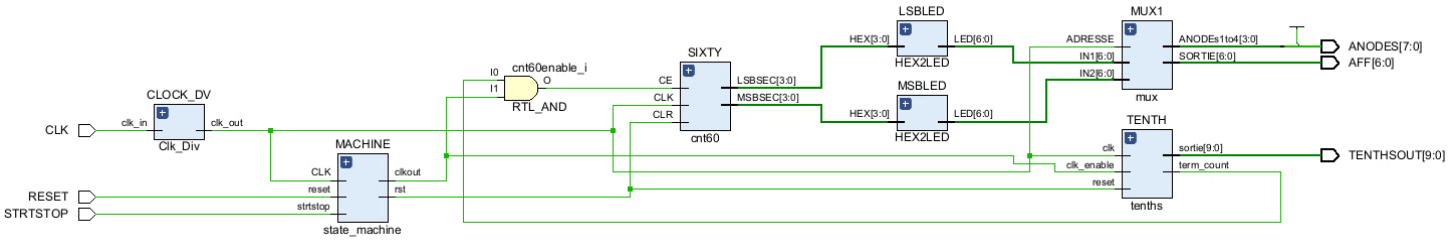
• **d'un bouton poussoir** de remise à zéro

• **d'un bouton poussoir** de démarrage du comptage, arrêt comptage et reprise comptage

Les entrées sorties de l'entité **stopwatch.VHD** sont les suivantes :

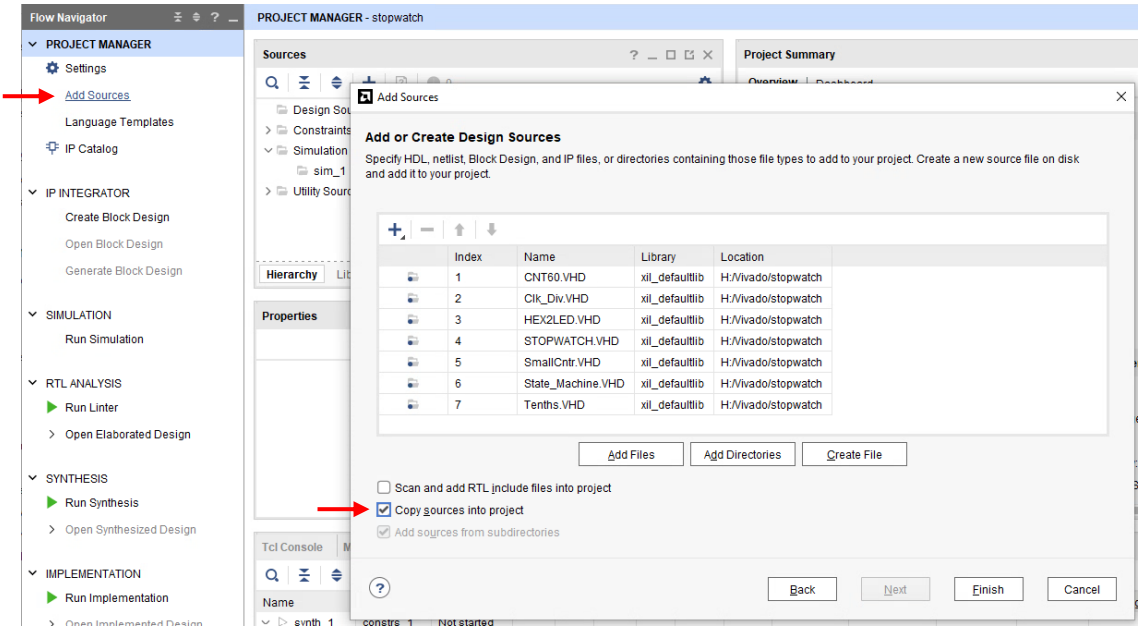
- **<u>Entrées</u>**
  - Clk : horloge externe
  - Reset : remise à zéro principale
  - Startstop : comptage, arrêt comptage, reprise comptage
- **<u>Sorties</u>**
  - Tenthsout [9:0] : 10ème de seconde
- **<u>Cas sans multiplexage des afficheurs</u>**
  - Onesout [6:0], tensout [6:0] : secondes **et** dizaines de seconde

L'objectif sera de modifier le projet proposé pour le rendre portable sur la carte NEXYS-A7 en gérant le multiplexage des afficheurs. Le schéma final correspondant est le suivant :

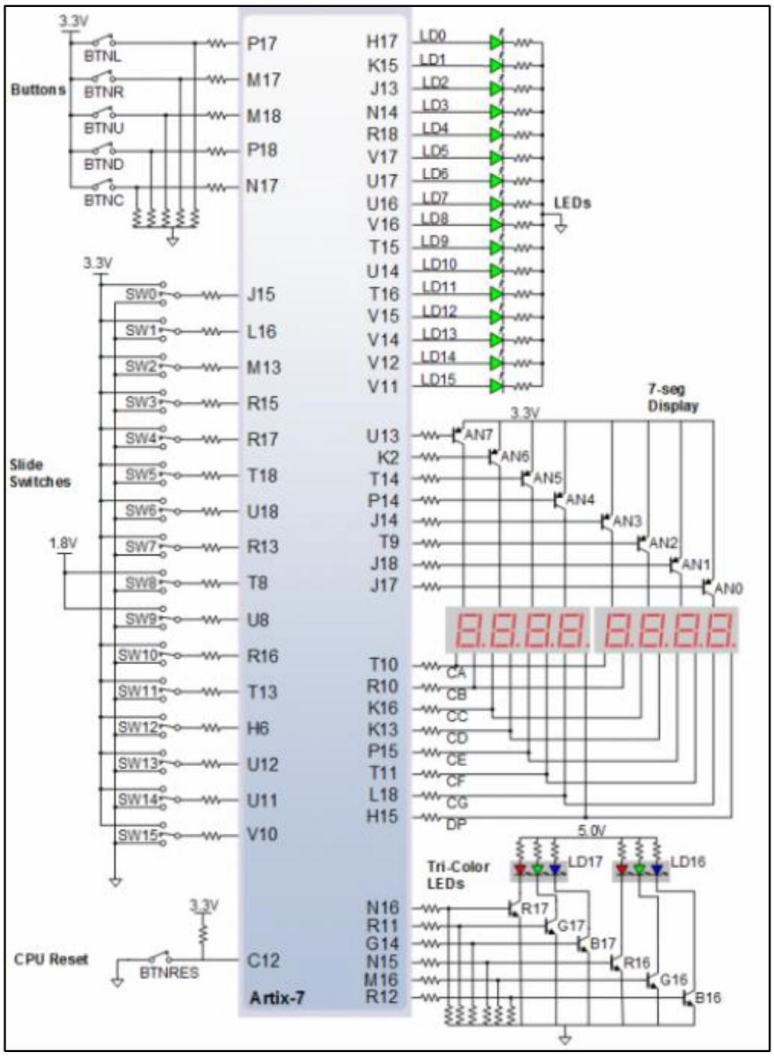


ETAPES :

- ✓ Décompresser l'archive **TP_E1.zip** (disponible sur Moodle/EC3/Sources du TP Chronomètre) dans votre répertoire de travail.
- ✓ Créer le projet : **stopwatch.xpr** dans ce même répertoire
- ✓ Créer le fichier de contrainte **stopwatch.xdc** à l'invite de la fenêtre **Add Constraints**
- ✓ Ajouter les sources vhd à votre projet en sélectionnant **Add Sources** dans le menu **Project Manager** de la fenêtre **Flow Navigator.** Cliquer sur **Add Files** et sélectionner l'ensemble des fichiers vhd en prenant soin de cocher la case **Copy sources into project,** comme indiqué ci-dessous.



- ✓ Etudier le code de la machine d'états. Dessiner la machine d'états sous forme graphique et identifier son type (Moore ou Mealy).
- ✓ Ecrire la description VHDL de la fonction multiplexage nécessaire à l'affichage des secondes et des dizaines de secondes.
- ✓ Modifier la page principale (**stopwatch.VHD**) pour intégrer le multiplexeur. La sortie du multiplexeur remplace les 2 sorties des blocs LSBLED (secondes) et MSBLED (dizaines de secondes). On utilisera les afficheurs 1 et 2 pour les secondes et dizaines de secondes, les afficheurs 3 et 8 seront désactivés (A3 à A8 reliés au niveau logique haut).
- ✓ Assigner les pins d'entrées/sorties du FPGA. Sélectionner **I/O Pin planning** comme indiqué dans le tutoriel de l'inverseur. Le câblage est donné sur la figure ci-dessous (aussi directement lisible sur la carte). Enregistrer le fichier de contrainte.
- ✓ Générer le bistream et programmer la carte.


3.3V
BTNL
BTNR
BTNU
BTND
BTNC
Buttons
P17
M17
M18
P18
N17
H17
K15
J13
N14
R18
V17
U17
U16
V16
T15
U14
T16
V15
V14
V12
V11
LD0
LD1
LD2
LD3
LD4
LD5
LD6
LD7
LD8
LD9
LD10
LD11
LD12
LD13
LD14
LD15
LEDs
3.3V
SW0
SW1
SW2
SW3
SW4
SW5
SW6
SW7
SW8
SW9
SW10
SW11
SW12
SW13
SW14
SW15
Slide Switches
1.8V
J15
L16
M13
R15
R17
T18
U18
R13
T8
U8
R16
T13
H6
U12
U11
V10
7-seg Display
3.3V
AN7
AN6
AN5
AN4
AN3
AN2
AN1
AN0
U13
K2
T14
P14
J14
T9
J18
J17
CA
CB
CC
CD
CE
CF
CG
DP
T10
R10
K16
K13
P15
T11
L18
H15
5.0V
Tri-Color LEDs
LD17
LD16
R17
G17
B17
R16
G16
B16
N16
R11
G14
N15
M16
R12
3.3V
CPU Reset
BTNRES
C12
Artix-7