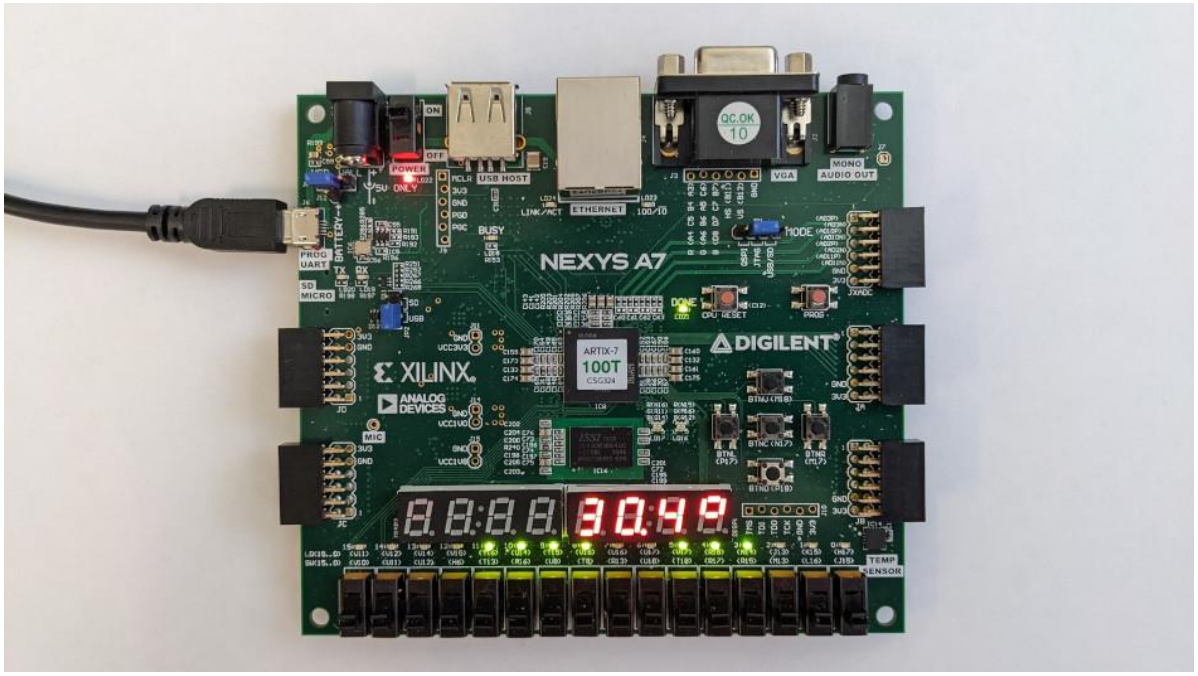
TP VHDL 3 & 4 : Interfaçage d'un capteur de température

Objectif :

L'objectif de ce TP est de réaliser une lecture périodique de la valeur de température mesurée par le capteur Analog Device ADT7420 déjà implémenté sur la carte Nexys A7. Ce capteur de température possède par défaut une résolution de 13 bits (0.0625 °C/LSB) et réalise une conversion tous les 240 ms. Cette conversion est stockée sur un ensemble de 2 registres 8 bits situés aux adresses 0 et 1 du capteur, qui sont accessibles en lecture à l'aide d'un bus de communication série I2C, et dont les 7 bits d'adresse du capteurs sont « 1001011 » (0x4B). Les 13 bits de la valeur mesurée constituent les 13 bits de poids le plus fort comme présentés dans le tableau ci-dessous. Le bit D12 est un bit de signe égal à zéro lorsque la température est positive.

Registre 0								Registre 1							
D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	X	X	X

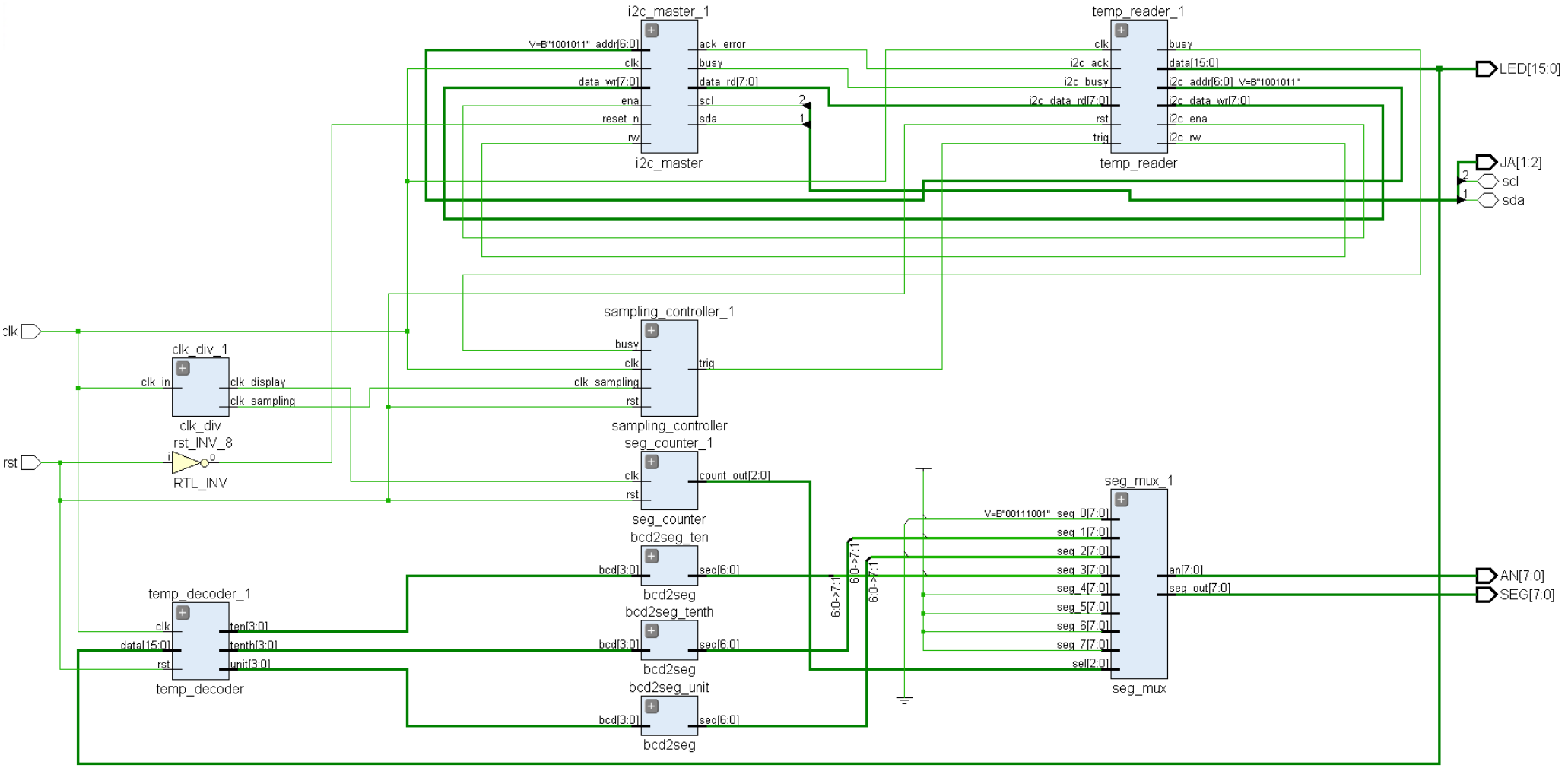
Dans un deuxième temps, il s'agira d'afficher cette valeur au format décimal en utilisant les afficheurs 7 segments disponibles sur la carte, comme représenté sur la figure suivante :



Le schéma RTL de l'application complète est donné sur la page suivante.

Lors de la première séance, seules les entités *main*, *temp_reader*, *clk_div* et *sampling_controller* sont à réaliser. L'entité *i2c_master* est fournie.

A la deuxième séance, les entités d'affichage *temp_decoder*, *seg_counter* et *seg_mux* seront à concevoir et à ajouter au fichier principal.


i2c_master_1
V=B"1001011" addr[6:0]
clk
data_wr[7:0]
ena
reset_n
rw
ack_error
busy
data_rd[7:0]
scl
sda
i2c_master
temp_reader_1
clk
i2c_ack
i2c_busy
i2c_data_rd[7:0]
rst
trig
busy
data[15:0]
i2c_addr[6:0] V=B"1001011"
i2c_data_wr[7:0]
i2c_ena
i2c_rw
temp_reader
LED[15:0]
JA[1:2]
scl
sda
clk
clk_div_1
clk_in
clk_display
clk_sampling
clk_div
rst
rst_INV_8
RTL_INV
sampling_controller_1
busy
clk
clk_sampling
rst
trig
sampling_controller
seg_counter_1
clk
rst
count_out[2:0]
seg_counter
bcd2seg_ten
bcd2seg_tenth
bcd2seg_unit
bcd[3:0]
seg[6:0]
bcd2seg
temp_decoder_1
clk
data[15:0]
rst
ten[3:0]
tenth[3:0]
unit[3:0]
temp_decoder
6:0->7:1
seg_mux_1
V=B"00111001" seg_0[7:0]
seg_1[7:0]
seg_2[7:0]
seg_3[7:0]
seg_4[7:0]
seg_5[7:0]
seg_6[7:0]
seg_7[7:0]
sel[2:0]
an[7:0]
seg_out[7:0]
seg_mux
AN[7:0]
SEG[7:0]

Partie 1 : lecture de la valeur binaire de température (4h)

En vous aidant de la documentation du capteur (pp. 17 à 19) ainsi que de celle de l'entité *i2c_master* qui vous est fournie, vous allez réaliser une entité *temp_reader* permettant d'envoyer les commandes I2C nécessaires à la lecture de la valeur de température ainsi que de restituer la valeur lue dans le capteur.

- Déterminer les commandes I2C d'écriture et de lecture à envoyer au capteur pour lire la valeur de température.
- Quelle sortie de l'entité *i2c_master* va permettre la synchronisation du déclenchement successif de ces commandes ?
- Comment doit être contrôlé le signal *ena* de l'entité *i2c_master* pour s'assurer de la continuité de l'envoi des commandes sans que l'entité n'émette une condition STOP sur le bus I2C ?
- Dessiner la machine d'état dont les sorties contrôleront les entrées *addr, ena*, *rw* et *data_rw* de l'entité *i2c_master*. Un signal externe *trig* permettra de déclencher une nouvelle lecture, et une sortie *busy* permettra de signaler à une entité externe qu'une lecture est en cours.
- Ajouter dans cette machine d'état la lecture du signal *data_rw* au moment opportun pour stocker successivement les valeurs des registres 0 et 1 du capteur dans un signal STD_LOGIC_VECTOR de 16 bits.
- Implémenter et simuler cette machine d'état dans l'entité *temp_reader*
- Réaliser un fichier principal *main* qui reçoit en entrée une horloge *clock* de la carte ainsi qu'un signal *reset*. L'entité aura comme sorties :
    - LED : un vecteur 16 bits de la valeur binaire de température
    - JA : un vecteur 2 bits représentant 2 broches du port JA.

  Pour le bus I2C, le signal *scl* sera déclaré comme sortie (out) et le signal *sda* comme entrée-sortie (inout).
- Réaliser un diviseur d'horloge *clk_div* pour générer un signal d'horloge *clk_sampling* d'environ 3 Hz ainsi qu'une entité *sampling_controller* dont le rôle sera de créer une impulsion adaptée à l'entrée *trig* de *temp_*reader à partir du signal busy.
- Instancier ces deux entités dans le fichier *main* ainsi que les entités *i2c_master* et *temp_reader*. Les signaux *sda* et *scl* de l'entité *i2c_master* seront connectées à la fois au bus I2C de l'entité *main* ainsi qu'aux 2 broches du port JA pour une possibilité de debug.
- Affecter les entrées/sorties à l'aide du logiciel PlanAhead en associant
    - L'entrée *clock* au cristal 100 Mhz de la carte
    - L'entrée *reset* au bouton poussoir du milieu
    - Le vecteur 16 bits sur chacune des 16 LEDs de la carte
    - Les signaux *sda* et *scl* du bus I2C au broches du capteur de température.
    - Les 2 bits du signal JA aux broches du port JA correspondantes.

Partie 2 : décodage de la valeur binaire (2h)

Dans cette partie, il s'agit de convertir le signal STD_LOGIC_VECTOR 16 bits contenant la valeur de température en binaire en 3 signaux BCD 4 bits distincts :

- La valeur des dizaines de degrés ;
- La valeur des unités ;
- La valeur des dixièmes.

Ces trois signaux permettront par la suite l'affichage de la valeur de température sur les afficheurs 7 segments.

- Déterminez les entrées / sorties de l'entité *temp_decoder* à concevoir et qui sera chargée de la réalisation de ce décodage.

L'entité *binary_bcd* sert à décoder des valeurs entières, et retourne en sortie des signaux 4 bits contenant la valeur des unités, dizaines, centaines, etc (jusqu'à 5 chiffres possibles au total).

- Sur quels bits du signal 16 bits contenant la valeur de température cette entité doit-elle être appliquée pour effectuer la conversion ?
- Instancier l'entité *binary_bcd* pour qu'elle réalise la conversion de la partie entière de la valeur de température.

Pour les dixièmes de degré, il est nécessaire de réaliser une approximation de la fraction binaire en une fraction décimale.

- A l'aide d'une instruction concurrente appropriée, décrire la table de conversion de la fraction binaire restante en une valeur BCD codée sur 4 bits qui formera les dixièmes de degrés.
- Simuler l'entité *temp_decoder* pour vérifier son bon fonctionnement à l'aide de la valeur 16 bits « 0000 1100 1000 1000 » qui correspond à 25.1 °C.

Partie 3 : Affichage de la valeur décimale (2h)

Les quatre afficheurs 7 segments de droite de la carte seront utilisés pour afficher respectivement :

- La valeur des dizaines de degrés ;
- La valeur des unités avec le point décimal ;
- La valeur des dixièmes de degrés ;
- Le symbole de l'unité des degrés « ° »

- Instancier l'entité *bcd2seg* autant de fois que nécessaire pour convertir les valeur BCD des unités, dizaines et dixièmes de degrés au fort 7 segments.
- Créer les signaux nécessaires à l'entrée du multiplexeur 8:1 *seg_mux* qui réalise l'affichage successif des 8 entrées sur chacun des 8 afficheurs 7 segments, en incluant le point décimal.
- Ajouter une horloge supplémentaire *clk_display* en sortie de l'entité *clk_div* qui devra avoir une fréquence d'environ 6 kHz.
- Créer un compteur *seg_counter* qui servira de sélecteur pour le multiplexeur à partir de l'horloge d'affichage *clk_display*.
- Modifier l'entité principale *main* pour intégrer les sorties nécessaires à l'affichage de la valeur décimale sur les afficheurs 7 segments et les ajouter au fichier de contraintes.