



---

# RAPPORT PROJET 4A

---



Redigé par :

- MAMBO MOTSOU JUNIOR DEOGRACIAS
- KIMBELY OLIVIA NSENG MBAZOA

## Table of Contents

I.	Introduction .....	4
II.	Description du Matériel et des Outils .....	5
1.	Carte Nexys 4 DDR FPGA .....	5
2.	Capteur de Température ADT7420 : .....	5
3.	Protocole I2C : .....	5
III.	Fonctionnement du Système.....	6
1.	Acquisition des données .....	6
1.1	Protocole I2C.....	6
A.	Fonctionnement du protocole I2C .....	7
2.	Traitement des données .....	9
2.1	Acquisition des Données Brutes.....	9
2.2	Conversion du Signal Brut (13 bits) en Degrés Celsius.....	11
3.	Affichage de la Température sur un Afficheur 7 Segments.....	11
4.	Transmission UART .....	14
5.	Défis Techniques .....	17
6.	Applications Pratiques .....	19
7.	Conclusion et Perspectives .....	20
	Figure 1:I2C Protocol .....	6

Figure 2:Temperature sensor Component.....	8
Figure 3:I2C communication enable.....	8
Figure 4:Acquisition de la temperature.....	9
Figure 5:Temperature conversion.....	12
Figure 6:7 Segment Display .....	14

# I. Introduction

Dans un monde de plus en plus connecté et orienté vers des solutions technologiques avancées, le domaine de l'électronique automobile joue un rôle fondamental dans l'évolution des systèmes embarqués. Mon projet, réalisé dans le cadre de mes études en génie électrique, porte sur la mesure de la température à l'aide d'un capteur **ADT7420** intégré à une carte **Nexys 4 DDR FPGA**.

Ce projet s'inscrit dans une démarche pédagogique visant à renforcer mes compétences en conception de systèmes électroniques complexes, notamment à travers l'utilisation des protocoles de communication comme I2C et UART, ainsi que l'affichage sur des écrans 7 segments.

L'objectif principal est de mesurer avec précision la température, de la convertir en degrés Celsius et de l'afficher tout en permettant la transmission des données par une interface série. Ce projet prépare à des applications pratiques dans des systèmes embarqués automobiles tels que la surveillance thermique des composants, un domaine crucial dans l'industrie automobile moderne.

En outre, je souhaite poursuivre mes études dans le domaine de l'électronique automobile afin d'approfondir mes connaissances et de contribuer au développement de solutions innovantes pour des véhicules plus sûrs, intelligents et respectueux de l'environnement.

## II. Description du Matériel et des Outils

Le projet repose sur l'utilisation de matériels et d'outils technologiques avancés pour garantir une mise en œuvre efficace et précise :

1. **Carte Nexys 4 DDR FPGA** : Cette carte est équipée d'un FPGA Artix-7 qui permet une grande flexibilité dans la conception de systèmes logiques complexes. Elle intègre des connecteurs GPIO, un afficheur à 7 segments et des interfaces pour des protocoles de communication comme UART et I2C.
2. **Capteur de Température ADT7420** : Ce capteur haute précision, communiquant via le protocole I2C, mesure la température avec une résolution de 0,0625 °C. Il est idéal pour les applications sensibles à la température comme celles trouvées dans les systèmes automobiles.
3. **Protocole I2C** : Utilisé pour assurer une communication efficace entre le capteur ADT7420 et la carte FPGA. Ce protocole s'avère indispensable pour réaliser une transmission fiable des données.

L'intégration de ces différents éléments permet de garantir la cohérence et la fiabilité des mesures, tout en offrant une interface utilisateur intuitive et accessible.

### III. Fonctionnement du Système

Le système est conçu pour fonctionner de manière autonome et efficace. Voici les principales étapes du fonctionnement :

#### 1. Acquisition des données

Le capteur de température ADT7420 mesure la température ambiante et envoie les données sous forme numérique via une interface I2C.

##### 1.1 Protocole I2C

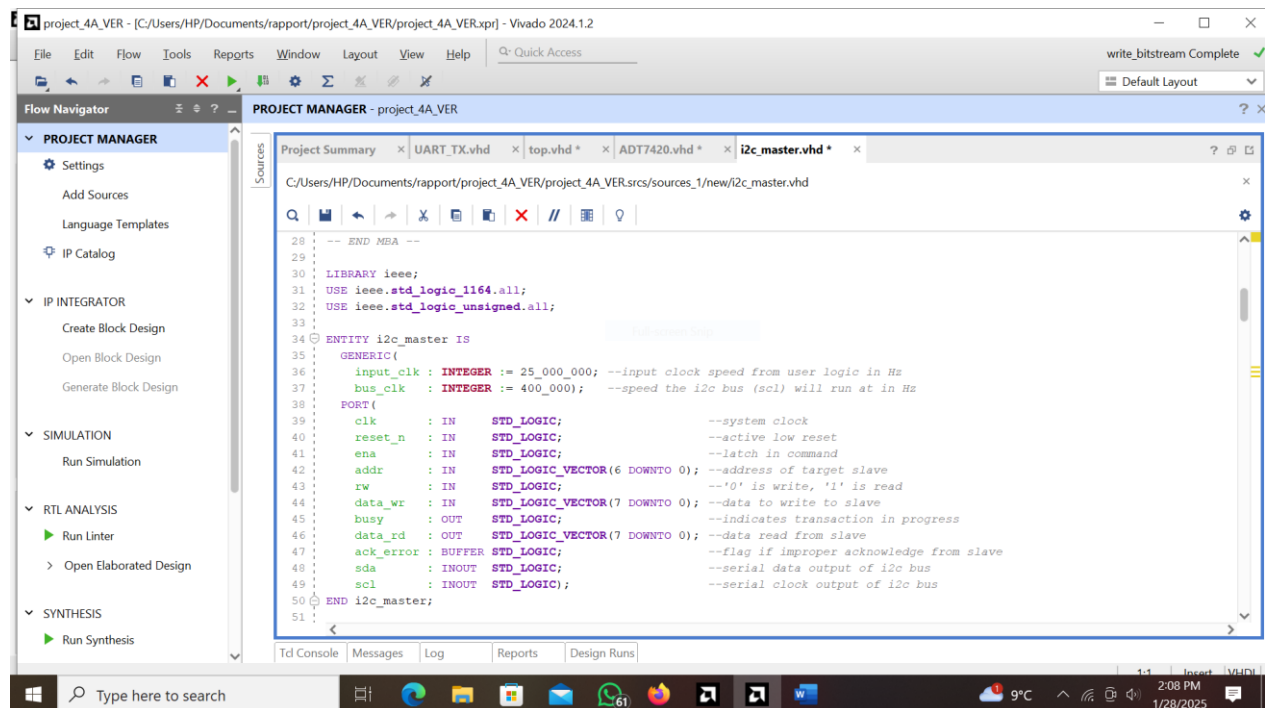


Figure 1:I2C Protocol

Le protocole **I2C** (Inter-Integrated Circuit) est un protocole de communication série synchrone utilisé pour la transmission de données entre plusieurs périphériques sur une même ligne. Ce protocole est largement utilisé dans les systèmes embarqués en raison de sa simplicité et de sa faible consommation de broches.

## A. Fonctionnement du protocole I2C

- **Architecture maître-esclave** : L'I2C fonctionne avec un dispositif maître qui initie les communications, et un ou plusieurs dispositifs esclaves qui répondent aux requêtes du maître.
- **Lignes de communication** : Deux lignes principales sont utilisées :
  - **SDA (Serial Data Line)** : Pour la transmission des données.
  - **SCL (Serial Clock Line)** : Pour la synchronisation des données.
- **Adresses uniques** : Chaque esclave connecté au bus I2C dispose d'une adresse unique, permettant au maître d'envoyer des requêtes ciblées.
- **Transmission des données** :
  - Les données sont envoyées sous forme de bits avec des cycles de démarrage et d'arrêt spécifiques.
  - Les messages comprennent une adresse d'esclave, un bit de lecture/écriture, et les données.

Dans ce projet, le protocole I2C est utilisé pour établir une communication entre la carte Nexys 4 DDR et le capteur de température **ADT7420** :

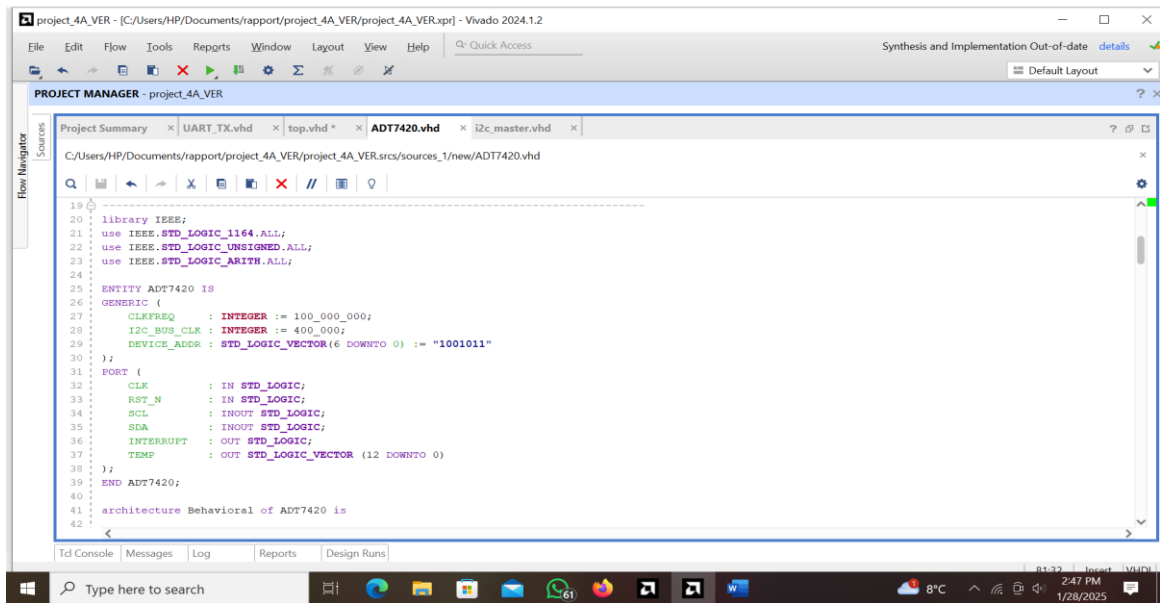


Figure 2: Temperature sensor Component

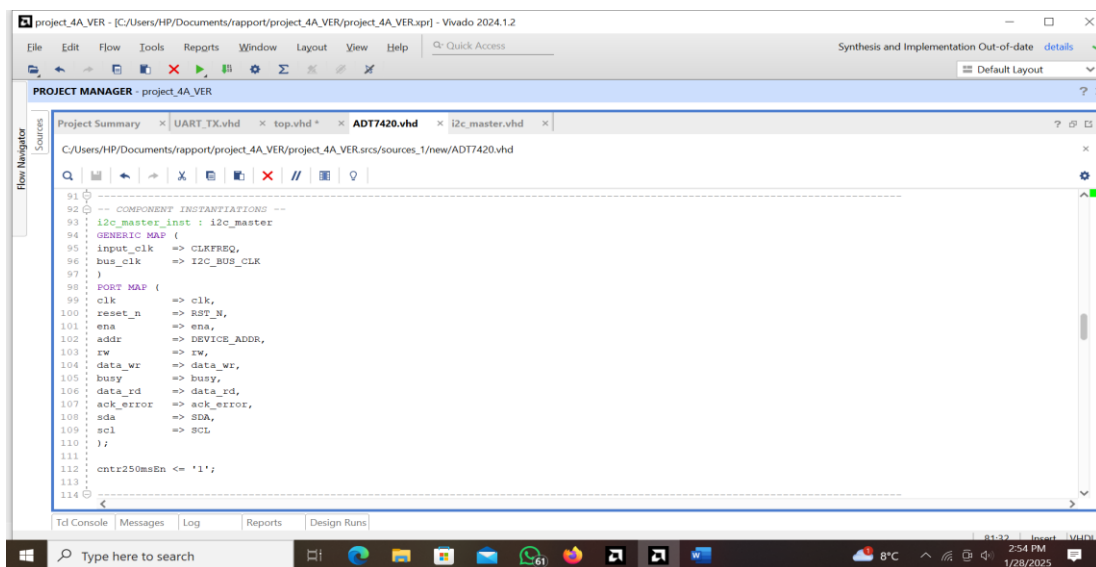


Figure 3: I2C communication enable

1. **Condition de Départ (Start)** : Le FPGA initie la communication en tirant SDA à bas pendant que SCL est haut.
2. **Adresse et Commande** : Le FPGA envoie l'adresse de l'ADT7420 (par défaut 0x48) avec un bit de lecture/écriture (R/W).



3. **Acknowledge (ACK)** : L'ADT7420 confirme la réception de l'adresse en répondant par un ACK.
4. **Lecture des Données** :
  - L'ADT7420 envoie les données de température sur 16 bits (MSB suivi de LSB).
  - Le FPGA lit ces deux octets et effectue une conversion en température.
5. **Condition d'Arrêt (Stop)** : La communication se termine en relâchant SDA.

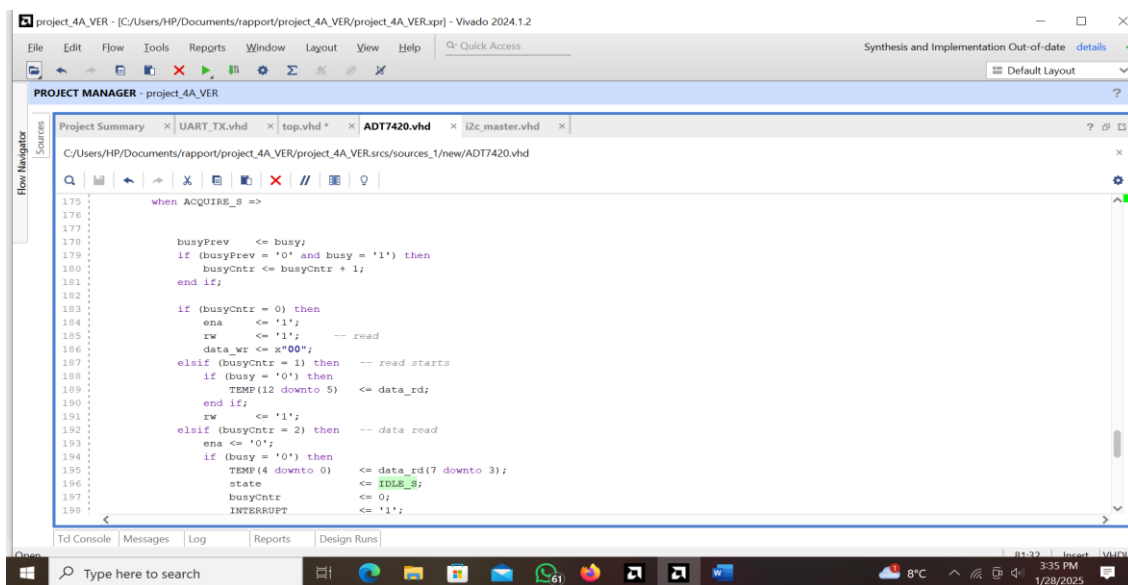
## 2. Traitement des données

La carte FPGA Nexys 4 DDR reçoit les données du capteur et effectue les conversions nécessaires pour traduire les valeurs brutes en degrés Celsius.

Le traitement des données obtenues via le capteur **ADT7420** se déroule en plusieurs étapes essentielles pour exploiter les informations de température et les afficher ou les utiliser dans des applications spécifiques. Voici un résumé des étapes de traitement :

### 2.1 Acquisition des Données Brutes

- Après l'établissement de la communication I<sup>2</sup>C entre le FPGA et l'ADT7420, le capteur envoie deux octets (MSB et LSB) représentant la valeur de la température.



```
175 when ACQUIRE_S =>
176
177
178     busyPrev <= busy;
179     if (busyPrev = '0' and busy = '1') then
180         busyCnt <= busyCnt + 1;
181     end if;
182
183     if (busyCnt = 0) then
184         ena <= '1';
185         rw <= '1'; -- read
186         data_wr <= x"00";
187     elsif (busyCnt = 1) then -- read starts
188         if (busy = '0') then
189             TEMP(12 downto 5) <= data_rd;
190         end if;
191         rw <= '1';
192     elsif (busyCnt = 2) then -- data read
193         ena <= '0';
194         if (busy = '0') then
195             TEMP(4 downto 0) <= data_rd(7 downto 3);
196             state <= IDLE_S;
197             busyCnt <= 0;
198             INTERRUPT <= '1';
199         end if;
200     end if;
```

Figure 4: Acquisition de la température

Le code gère la collecte des données de température à partir d'un capteur, en lisant les données en deux étapes pour obtenir une valeur de 13 bits. Le processus est contrôlé par une machine d'état et repose sur un compteur pour gérer les différentes phases de lecture des données.

**busyPrev et busy** : Ces signaux vérifient si le capteur est occupé. Si le signal busy passe de '0' à '1', cela indique que le capteur est occupé et le compteur busyCntr s'incrémente.

- **Étape 1 (busyCntr = 0) :**

- Le système active la lecture en mettant ena à '1' et rw à '1' (pour lire les données).
- data\_wr est mis à x"00" car il n'y a pas d'écriture à effectuer. Cela prépare simplement le capteur pour la lecture.

- **Étape 2 (busyCntr = 1) :**

- Le système commence à lire les données.
- Si le capteur n'est plus occupé (busy = '0'), les 8 premiers bits de la température (de TEMP(12 downto 5)) sont récupérés à partir de data\_rd.

- **Étape 3 (busyCntr = 2) :**

- La lecture des 5 derniers bits de la température se fait lorsque le capteur est de nouveau prêt (busy = '0'). Ces bits sont récupérés à partir de data\_rd(7 downto 3) et stockés dans TEMP(4 downto 0).
- Le système passe ensuite à l'état IDLE\_S et réinitialise le compteur busyCntr à 0. Un signal INTERRUPT est déclenché pour indiquer que la collecte de la température est terminée.

**Résultat final :**

La température complète en 13 bits est stockée dans le registre TEMP :

- Les 8 premiers bits sont dans TEMP(12 downto 5).
- Les 5 derniers bits sont dans TEMP(4 downto 0).

## 2.2 Conversion du Signal Brut (13 bits) en Degrés Celsius

Le capteur de température **ADT7420** utilisé sur la carte **Nexys 4 DDR** communique les données de température au FPGA sous forme d'un signal numérique codé sur **16 bits**. Cependant, en mode haute résolution (13 bits), seuls les **13 bits significatifs** sont utilisés pour représenter la température.

### Structure du Signal Brut (13 bits) :

- Les 13 bits représentent la température selon le format suivant :
  - **Bit 12** : Bit de signe (0 pour une température positive, 1 pour une température négative).
  - **Bits 11 à 0** : Partie entière et fractionnaire de la température.

La valeur brute est convertie en degrés Celsius grâce à la formule suivante :

$$\text{temp\_celsius} \leftarrow \text{temp\_decimal} \times \frac{625}{10000}$$

Cela traduit la précision de **0,0625 °C** par unité (résolution du capteur).

```
process (TEMP)
    variable temp_decimal : integer;
begin

    if TEMP(12) = '1' then
        temp_decimal := to_integer(signed(TEMP));
    else
        temp_decimal := to_integer(unsigned(TEMP));
    end if;

    temp_celsius <= temp_decimal * 625 / 10000;
end process;
```

## 3. Affichage de la Température sur un Afficheur 7 Segments

Ce code permet d'afficher la température, lue depuis le capteur **ADT7420**, sur un afficheur 7 segments connecté à la carte **Nexys 4 DDR**. La conversion et l'affichage utilisent plusieurs processus pour gérer les étapes nécessaires.

#### A. Division de l'Horloge et Contrôle du Multiplexage

- Un diviseur d'horloge est implémenté pour générer un signal à **1 kHz**, permettant un rafraîchissement rapide de l'affichage.
- Un compteur (**digit\_number**) est incrémenté pour activer les anodes des 4 digits de l'afficheur 7 segments de manière séquentielle (multiplexage). Cela permet de réduire le nombre de connexions tout en maintenant une image stable sur l'afficheur.

#### Extraction des Chiffres à Afficher :

- Les trois chiffres représentant la température sont extraits à partir de la valeur absolue de **temp\_celsius** :
  - **digit2** : Centaines.
  - **digit1** : Dizaines.
  - **digit0** : Unités.
- Si la température est négative, un drapeau (**is\_negative**) est activé pour afficher le signe - sur le 4ème digit.

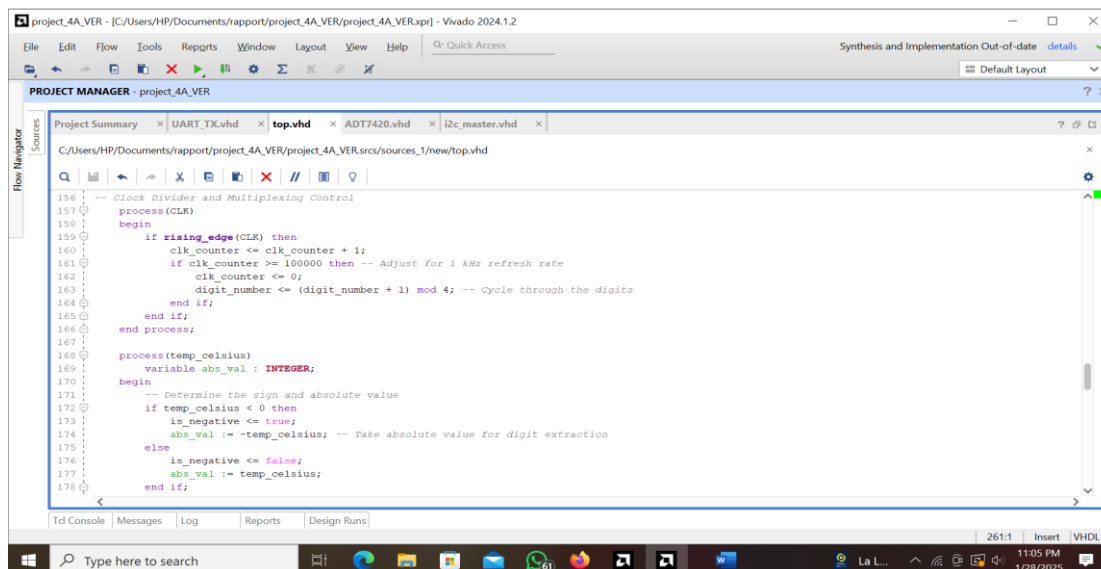
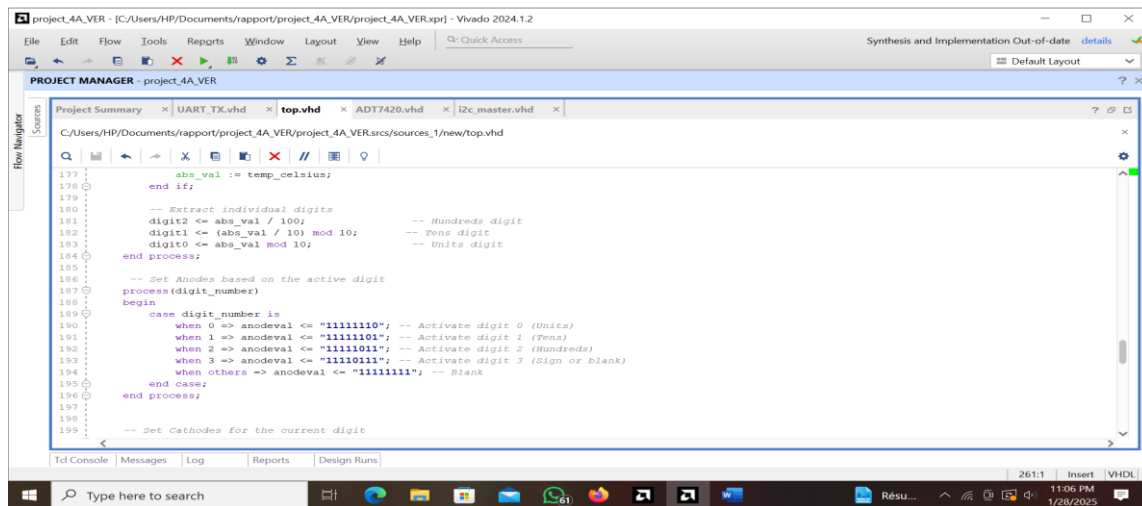


Figure 5: Temperature conversion



## Contrôle des Anodes :

- Les anodes des afficheurs 7 segments sont activées en utilisant un signal actif-bas. Par exemple :
  - "11111110" : Active le 1er digit (unité).
  - "11111011" : Active le 3ème digit (centaines).

## Contrôle des Cathodes :

- Les cathodes sont contrôlées pour afficher les chiffres (ou le signe négatif). Les combinaisons binaires des segments sont préconfigurées pour chaque chiffre (0 à 9) et pour le signe - :
  - "11000000" : Affiche le chiffre 0.
  - "10111111" : Affiche le signe -.
  - "11111111" : Éteint les segments pour laisser l'afficheur vide.

Ce code met en œuvre une solution efficace pour afficher une température sur un afficheur 7 segments en utilisant les ressources du FPGA, tout en respectant les contraintes de temps réel et de multiplexage.

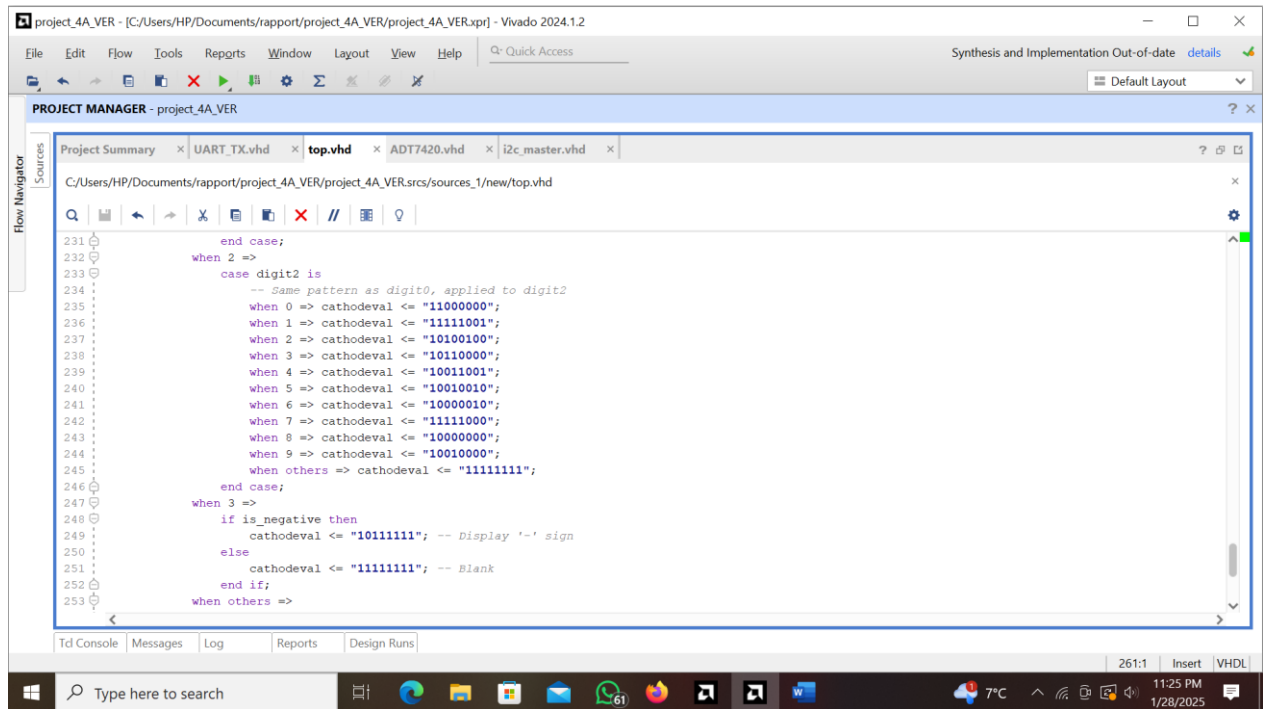


Figure 6:7 Segment Display

## 4. Transmission UART

### Transmission UART et Intégration du Composant UART\_TX

#### 1. Introduction à la Transmission UART :

La transmission UART (**Universal Asynchronous Receiver Transmitter**) est utilisée pour transmettre des données série de manière asynchrone. Le module **UART\_TX** présenté ici est configuré pour :

- Une horloge de **100 MHz** (fréquence de la carte Nexys 4 DDR),
- Un débit en bauds de **115200 bauds** (vitesse de transmission standard),
- Une longueur de donnée de **8 bits**,
- Un bit de stop configuré avec **2 périodes de l'horloge (SB\_TICK)**.

L'objectif principal de ce module est d'envoyer les données de température au format binaire à un périphérique externe via une ligne série.

## 2. Description du Composant UART\_TX :

### Entité UART\_TX :

- **Ports d'entrée :**
  - **CLK** : Signal d'horloge, ici à **100 MHz**.
  - **TX\_START** : Signal de contrôle pour démarrer la transmission.
  - **DIN** : Les données à transmettre (8 bits).
- **Ports de sortie :**
  - **TX\_DONE\_TICK** : Indicateur signalant la fin de la transmission.
  - **TX** : Signal série transmis à la ligne UART.

## 3. Fonctionnement du Module :

Le fonctionnement est basé sur une **machine d'états** avec les étapes suivantes :

### a) Idle (Repos) :

- Par défaut, le système reste dans cet état.
- Le signal **TX** est maintenu à 1 (ligne inactive).
- Lorsque le signal **TX\_START** passe à 1, la transmission démarre et passe à l'état suivant : **Start**.

### b) Start (Bit de Démarrage) :

- L'état envoie un bit de démarrage (0) pour signaler le début de la transmission.
- La durée du bit de démarrage est déterminée par la constante **s\_reg\_lim**, qui représente le nombre de cycles nécessaires pour une période à **115200 bauds**.

### c) Data (Bits de Données) :

- Les **8 bits** des données à transmettre (**DIN**) sont envoyés séquentiellement, bit par bit.

- Une fois chaque bit envoyé, le compteur **n\_reg** est incrémenté pour préparer l'envoi du bit suivant.
- Le registre temporaire **b\_reg** est utilisé pour décaler les bits à transmettre.

#### d) Stop (Bit de Stop) :

- Un ou plusieurs bits de stop (1) sont envoyés selon la configuration (**SB\_TICK** = 2 ici).
- Après avoir envoyé les bits de stop, l'état retourne à **Idle**, et le signal **TX\_DONE\_TICK** passe à 1 pour indiquer la fin de la transmission.

#### 4. Génération du Signal Série :

- La génération du signal **TX** est synchronisée avec l'horloge principale.
- La fréquence de la ligne UART est obtenue grâce à un diviseur d'horloge basé sur la constante **CLK\_FREQ / BAUD**.
- Les données binaires sont transmises bit à bit, commençant par le bit de démarrage (0), suivi des **8 bits de données**, et se terminant par les bits de stop (1).

#### 5. Avantages du Système :

- Transmission fiable des données de température.
- Standard de communication simple et compatible avec de nombreux périphériques.
- Configurable pour différents débits en bauds, tailles de données, et bits de stop.

#### Conclusion :

Le module **UART\_TX** permet d'envoyer efficacement des données de température à des périphériques externes via une interface UART. Ce composant est essentiel pour établir une communication série dans le système.



## 5. Défis Techniques

### 1. Conversion de la Température Brute en Degrés Celsius

La conversion de la température brute mesurée par le capteur ADT7420 en degrés Celsius a posé plusieurs défis, notamment :

- **Gestion des valeurs signées** : Le capteur fournit une mesure brute sur 13 bits, où le bit le plus significatif indique le signe de la température (0 pour positif, 1 pour négatif). Une logique spécifique a été implémentée pour convertir ces données en une valeur entière correcte tout en tenant compte des températures négatives.
  - Cela implique l'utilisation d'opérations arithmétiques pour ajuster correctement les valeurs binaires selon leur signe (positif ou négatif).
  - Une étape supplémentaire a été nécessaire pour extraire la valeur absolue afin d'afficher chaque chiffre individuellement sur les segments.

### 2. Synchronisation des Processus pour I2C, UART et l’Affichage

Le système comporte plusieurs modules travaillant simultanément :

- **I2C** : Pour lire la température brute depuis le capteur ADT7420.
- **UART** : Pour transmettre les données converties au format binaire à un périphérique externe.
- **Affichage sur 7 segments** : Pour présenter les températures sur l’écran intégré.

#### Défis rencontrés :

- **Synchronisation des horloges** : Les horloges des différents sous-systèmes (I2C, UART, affichage) doivent fonctionner de manière cohérente pour éviter des conflits ou des erreurs. Cela nécessite une gestion rigoureuse des signaux d'activation et de temporisation.
- **Traitement en temps réel** : La lecture, la conversion, l’envoi UART, et l’affichage doivent se faire dans des intervalles précis pour garantir la fluidité du système.

### 3. Multiplexage Efficace pour Afficher les Chiffres sur l'Écran 7 Segments

Pour afficher la température sur les **4 digits** du 7 segments, un multiplexage a été nécessaire :

- Chaque chiffre est activé successivement via ses anodes, et les segments correspondants sont contrôlés via les cathodes.
- Le multiplexage doit être rapide pour donner l'illusion que tous les chiffres sont affichés simultanément.

#### A. Problèmes rencontrés :

- **Contrôle de la fréquence de multiplexage** : Une fréquence trop basse entraîne un scintillement, tandis qu'une fréquence trop élevée peut perturber le fonctionnement des autres sous-systèmes.
- **Gestion des signes** : Le multiplexage inclut un chiffre supplémentaire pour afficher le signe (positif ou négatif) de la température, ce qui a ajouté une complexité au système.

#### B. Solutions et Optimisations

##### 1. Conversion de température :

- Utilisation de la représentation signée et non signée des valeurs binaires en VHDL pour traiter les températures négatives efficacement.
- Utilisation d'un processus dédié pour convertir la valeur brute en degrés Celsius avec une précision de 0,0625 °C.

##### 2. Synchronisation :

- Implémentation de diviseurs d'horloge spécifiques pour chaque sous-système afin de garantir une fréquence d'opération appropriée.
- Utilisation de signaux de contrôle pour coordonner les lectures I2C, l'envoi UART et l'affichage.

##### 3. Multiplexage :

- Ajustement de la fréquence de multiplexage à **1 kHz**, suffisamment rapide pour éviter tout scintillement perceptible.
- Gestion des chiffres et du signe par un processus unique pour minimiser la consommation de ressources sur l'FPGA.

## 6. Applications Pratiques

- **Automobiles** : Suivi en temps réel des températures des composants critiques (moteur, batterie, etc.).
- **IoT** : Détection et transmission à distance des températures via des systèmes embarqués.
- **Industrie** : Intégration dans des chaînes de production nécessitant un contrôle précis des températures.

## 7. Conclusion et Perspectives

- **Résumé :**

- Ce projet a permis de développer une solution complète combinant capteur, FPGA, I2C, UART et affichage.
- L'apprentissage porte sur la maîtrise des protocoles I2C et UART, le multiplexage d'un écran 7 segments et la gestion des horloges sur un FPGA.

- **Perspectives :**

- Ajout d'une fonctionnalité de **PWM audio** pour alerter en cas de température critique.
- Optimisation pour réduire l'utilisation des ressources sur le FPGA.
- Intégration d'autres capteurs pour un système de surveillance multi-paramètres.