TD1 VHDL
SPI et Capteur de Température

Objectifs

- ➢ Construire un convertisseur Binaire → BCD simple.
- ➢ Comprendre les périphériques et protocoles SPI / MicroWire.
- ➢ Savoir définir et implémenter un séquenceur (Machine à Etats)
- ➢ Implémenter un contrôleur SPI,
- ➢ Comprendre et utiliser un modèle comportemental (capteur de température).
- ➢ Mettre au point par la simulation RTL.

Récupérer les sources du TD :

- Dans le répertoire Src, vous avez les sources qu'il faudra compléter.
- Dans le répertoire Simu, vous avez le banc de test et le script de simulation.
- Dans le répertoire **Doc** vous avez la documentation technique du LM70.

Ce qu'il faut faire

1/ Comprendre le protocole SPI du LM70

- ➢ Analyser la documentation de ce circuit, fournie dans le sous répertoire **Doc**.
  En particulier, on s'intéressera à la figure 7.
- ➢ Il est **indispensable** de bien comprendre le fonctionnement du LM70 et du protocole d'échange. En particulier, les points suivants :
  - La broche unique **SI/O** sert aux transferts dans les *deux* directions.
  - Les *deux* fronts du signal **SC** (horloge SPI) ont un rôle précis. Il faudra en particulier veiller à échantillonner et présenter des données sur SI/O au moment adéquat. Se reporter aux contraintes timing de la documentation.
  - L'information de température, **binaire sur 11 bits**, est **cadrée** de façon précise dans les seize bits de la trame...
  - La conversion (acquisition) de température par le LM70 demande un « certain temps ». Il ne faut donc **pas** l'interroger **plus** de **4 fois par seconde**.
  - La **période minimale** pour SC est de 0.16 µs (6 MHz), mais elle peut être aussi grande que voulue (DC). On pourrait par exemple retenir une période de 20 µs, facile à établir grâce au signal **Tick10us**.

2.0 Serial Bus Timing Diagrams

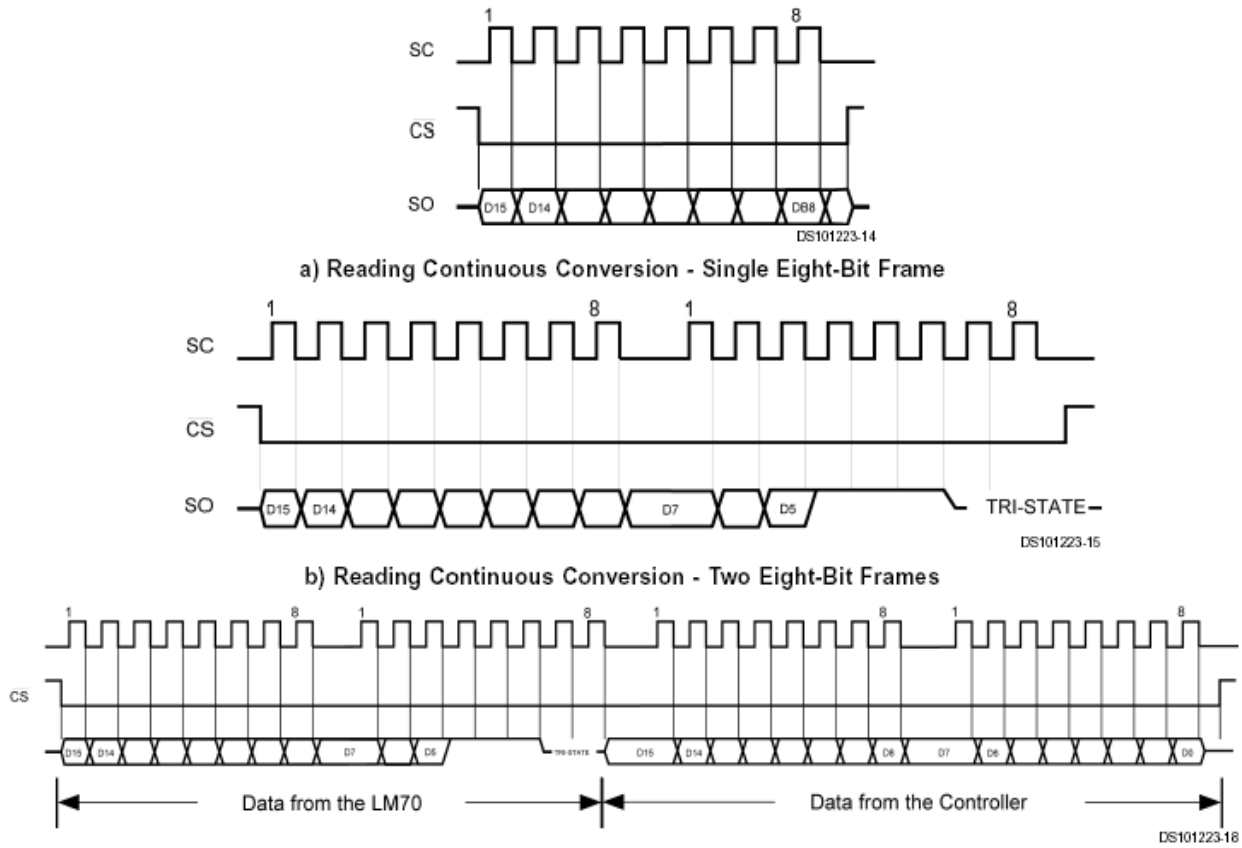

FIGURE 7. Timing Diagrams

Examiner, **comprendre** et **vérifier** le **modèle comportemental** fourni (situé dans le banc de test **Simu/SPI_Temp_TB.vhd**) par rapport à la documentation.

2/ Coder le contrôleur SPI / MicroWire : SPI_Temp.vhd

- ➢ Établir un cahier des charges du contrôleur SPI que l'on construira sous la forme d'une **Machine à Etats** (FSM). On pourra par exemple représenter les différents états du séquenceur, les transitions et les actions sous forme d'un diagramme à bulles. Penser à utiliser les signaux en provenance du diviseur **FDIV** : Tick10us et Tick10ms en particulier...
- ➢ Ouvrir le fichier **Src/SPI_Temp.vhd** et coder le séquenceur.
- ➢ Simuler et Vérifier à l'aide du Banc de test fourni (**spi_temp_tb.vhd**) et du script de simulation (**sim_spi_temp.do**). Pour cela, utiliser **Modelsim** et changer de répertoire (**Change Directory**) pour se placer dans le répertoire **Simu**. Lancer le script de simulation : **do sim_spi_temp.do**

3 / Construire le Convertisseur Binaire → BCD : BCD_Combin.vhd

Pourquoi cette conversion ?

Le code retourné par le LM70 est binaire signée exprimée en quarts de degrés Celsius, mais nous avons besoin d'afficher des nombres représentés en base 10.

Vu la précision absolue du capteur, on peut se contenter de la valeur en degré. De plus, pour simplifier encore le convertisseur, nous nous contentons dans cet exercice d'une information non signée sur six bits, car nous admettons que la température sera comprise entre 0 et 63 °C.

Ainsi, correctement re-cadrée et dans le cas ou la température est positive ou nulle, la valeur recueillie sera pour 25°C = *000* **0 1100 1** *001 1111* (cf documentation du LM70) dont nous isolons **01 1001** qui est bien 25 en binaire (ou 19 en hexadécimal).

Pour afficher « 2 » et « 5 », il faut donc faire une conversion du système binaire au système décimal : $2^4 + 2^3 + 2^0$ → **2** x 10^1 + **5** x 10^0

Plusieurs méthodes sont possibles pour effectuer cette conversion :

A) méthode séquentielle (itérative) par divisions successives par deux,

B) méthode combinatoire « parallèle ».

La méthode (B) est la plus simple à coder, car il suffit de définir la table de transcodage (*Look Up Table*), mais elle n'est viable que pour les codes de taille limitée (ce qui est le cas ici). Dans cette méthode, on peut soit coder toute la table manuellement (64 cas), soit utiliser un algorithme de conversion : le code sera plus compact et plus facile à vérifier, mais le résultat en terme d'efficacité sur le composant sera très vraisemblablement le même.

Ouvrir le fichier **Src/bcd_combin.vhd**

A. Coder la conversion binaire → BCD avec la table de transcodage dans l'architecture COMB1.

B. Écrire un Banc de test (**bcd_combin_tb.vhd**) pour ce module. Le banc de test complet doit générer toutes les valeurs d'entrées et faire une auto vérification en indiquant les erreurs sur un signal booléen OK ou par des assertions.

C. Simuler et vérifier le convertisseur décrit à l'aide de la table de transcodage.