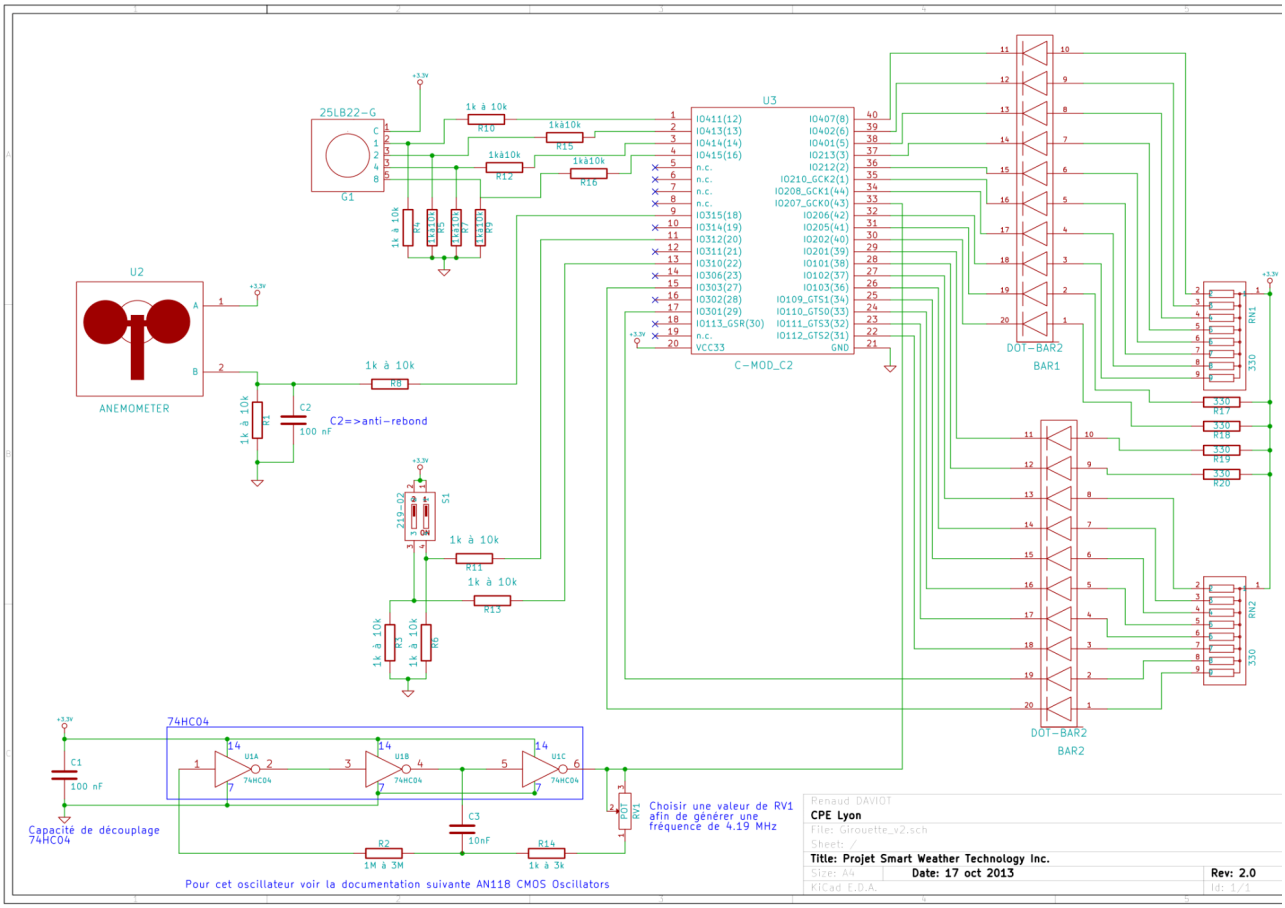
ELN 1 - TP 3/4

RÉALISATION D'UN PROTOTYPE DE SYSTÈME METEO A L'AIDE D'UN CPLD

Kleitz Antoine, Hayotte Morgane, Michalopoulos Zaphirios, Cornaton Maxime - Groupe C - Equipe 1



1. INTRODUCTION

Le but de ce TP est de réaliser un système météo. Le système électronique fait intervenir un module CPLD XC2C64A de Xilinx, et c'est sur la réalisation des fonctions de ce CPLD et des circuits du système électronique que sera dirigée notre étude. Le composant et ses fonctions seront développés en VHDL grâce au logiciel ISE.

Fonction réalisée par le module:

Le module doit être en capacité d'afficher les données d'un anémomètre ou d'une girouette sur un bargraph LED.

Entrées et sorties:

Les grandeurs d'entrée sont :

- Les positions de la girouettes, codé en GRAY grâce à un encodeur mécanique sur 4 bit soit un total de 16 directions possibles.
- La vitesse du vent mesurée par l'anémomètre, codé sur 1 bit et qui fonctionne sur un principe « tout ou rien ».
- Un interrupteur 1 permettant de choisir l'affichage des informations de la girouette ou de l'anémomètre.
- Un interrupteur 2 permettant de choisir l'affichage de la direction du vent (girouette) soit en code GRAY soit en binaire naturel.
- Une horloge d'une fréquence comprise entre 10 et 15 kHz.

Les grandeurs de sorties sont :

- Les 4 premières LED du bargraph, qui indiquent soit la direction vent en binaire ou en gray selon la position de l'interrupteur 2, soit la 4 derniers bits de la force du vent, selon la position de l'interrupteur 1.
- Les 16 dernières LED du bargraph, qui indiquent soit la direction du vent parmi les 16 directions, soit la force du vent selon l'interrupteur 1.

Contraintes:

Les composants imposées sont :

- Un CPLD XC2C64A de XILINX
- Un encodeur mécanique en code GRAY de type 25LB22-G de GrayHill
- Un anémomètre à coupelle
- Deux interrupteurs micro-switch
- Deux barres de LEDs (bar-graph) de type DC-10YWA de Kingbright
- Résistances, condensateurs et autres composants du laboratoire.

La tension d'alimentation imposée est de 3.3V. Le projet se déroule en équipe de 4 sur 2 séances de 4 heures chacune. La 1ère séance s'axe sur le développement des fonctions de base et sur la création des fonctions combinatoires en VHDL. La 2nd s'axe sur la finalisation du programme VHDL et la réalisation du circuit.

2. DESCRIPTION FONCTIONNELLE

Synoptique général du système (niveau hiérarchique 0):

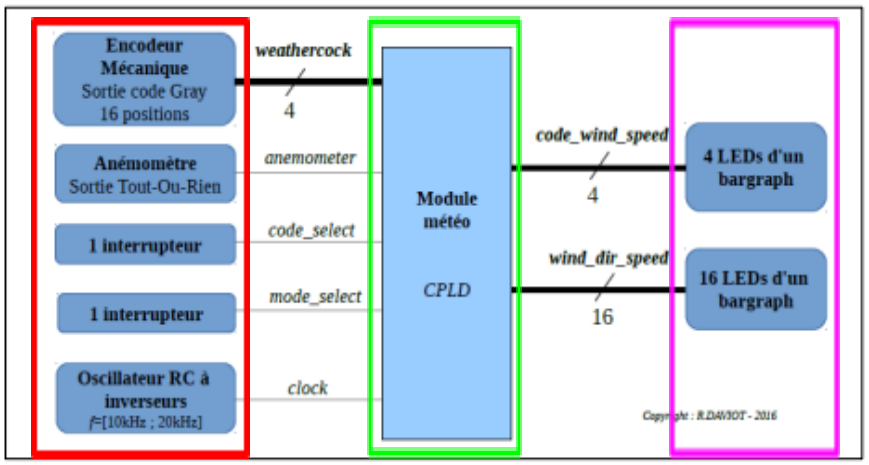


Figure 1 : Synoptique général du système météo

Le système météo à réaliser est constitué de 3 sous-systèmes :

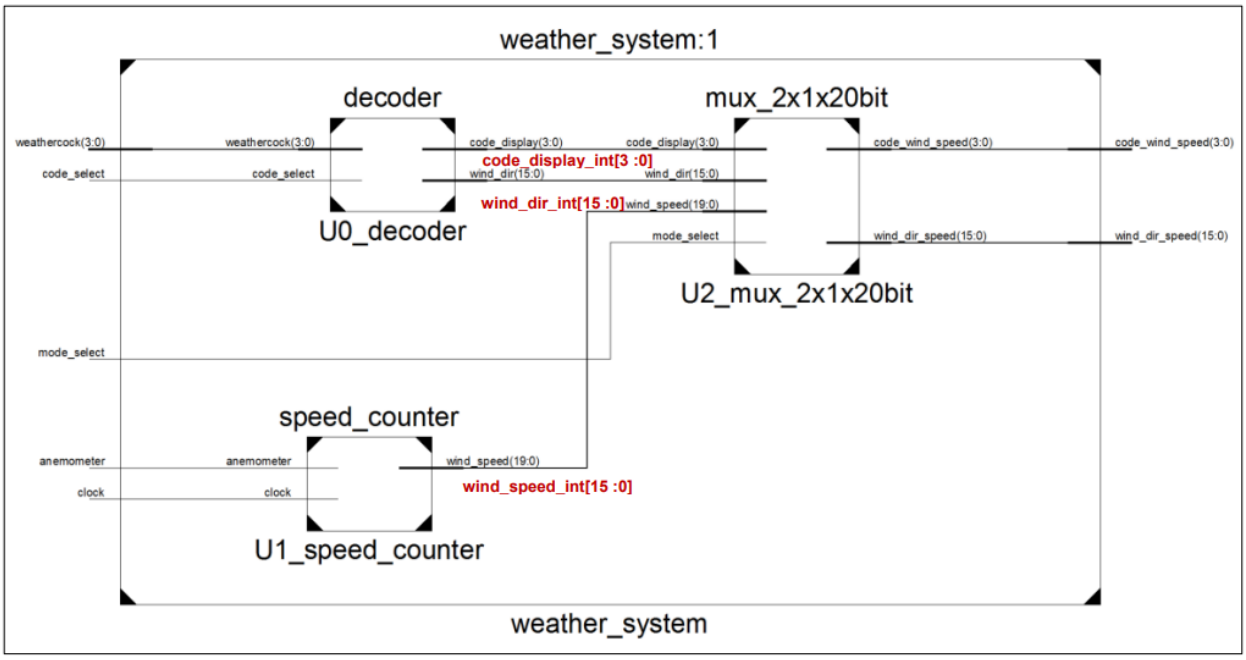
- un sous-système assurant le conditionnement des signaux issus des éléments analogiques d'entrée,
- un sous-système assurant le traitement des informations,
- un sous-système permettant la visualisation des informations de sortie.

La rotation de la flèche de la girouette entraîne la rotation de l'arbre de l'encodeur mécanique délivrant un code GRAY sur 4 bits. Ainsi, chaque position de la girouette correspond à un code parmi 16. L'analyse de la vitesse du vent, mesurée par l'anémomètre à coupelles, sera gérée au travers d'éléments de logique séquentielle. Ces éléments seront cadencés par un signal d'horloge de forme carrée, de rapport cyclique de 50 % et de fréquence comprise entre 10kHz et 15kHz. Ce signal sera obtenu à l'aide d'un oscillateur à INVERSEURS. Un premier interrupteur micro-switch permettra de choisir entre le mode GIROUETTE et le mode ANÉMOMÈTRE. Un second interrupteur micro-switch sera opérationnel uniquement en mode GIROUETTE et permettra de choisir entre l'affichage du code GRAY et l'affichage du code BINAIRE NATUREL.

Le sous-système de traitement aura pour nom weather_system et sera intégré dans le CPLD XC2C64A de XILINX. Le composant dispose de 44 entrées-sorties et son nombre de macrocellules est de 64. Le code GRAY obtenu en sortie de l'encodeur mécanique sera transcodé en BINAIRE NATUREL et chacune des 16 combinaisons possibles sera décodée afin d'indiquer le sens du vent à partir de l'éclairement d'une LED parmi 16. Le code GRAY ou le code BINAIRE NATUREL sera affiché sur 4 autres LEDs. Le signal issu de l'anémomètre à coupelles sera traité de manière à afficher l'intensité du vent sur 20 LEDs.

La visualisation des signaux est assurée par deux afficheurs à LEDS ou bar-graph (2*10 LEDs). Une LED sera éclairée lorsque le signal à visualiser sera à l'état BAS.

Synoptiques du sous-système de traitement (niveau hiérarchique 1):

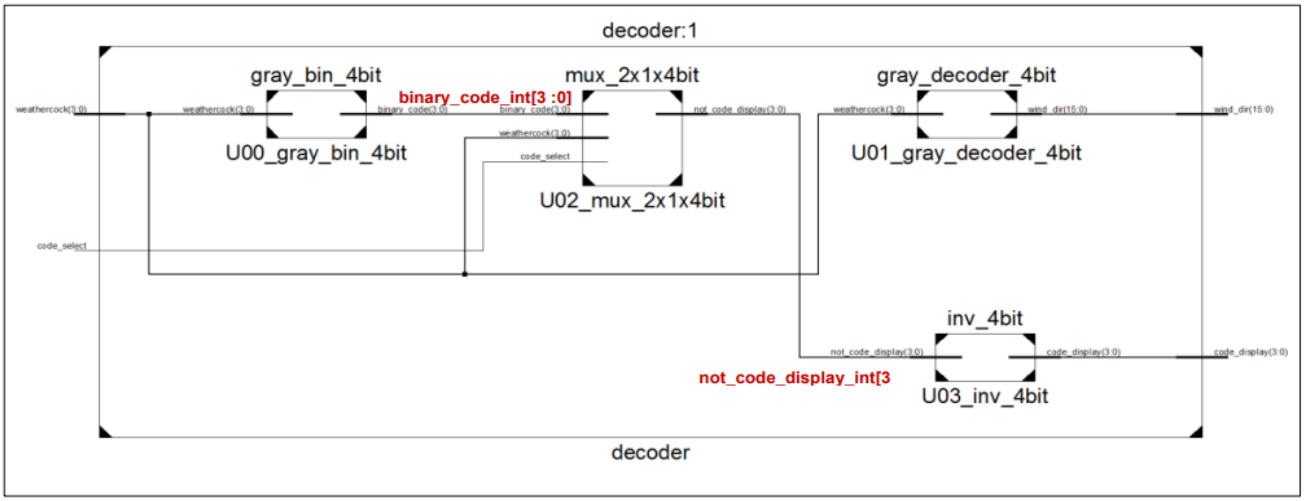


Le bloc fonctionnel decoder fournit en sortie le signal wind_dir défini sur 16 bits. Ce signal indique la position du vent à partir du signal weathercock codé en GRAY sur 4 bits et appliqué sur les entrées du bloc. Il fournit également en sortie le signal code_display défini sur 4 bits correspondant au code à afficher selon la valeur du signal code_select.

Le bloc fonctionnel speed_counter fournit en sortie le signal wind_speed défini sur 20 bits. Ce signal indique la vitesse du vent à partir des signaux anemometer et clock appliqués sur les entrées du bloc.

Synoptique détaillé de chacun des blocs du sous-système de traitement :

Decoder :



La fonction gray_bin_4bit est un TRANSCODEUR qui fournit sur ses sorties le signal binary_code codé en BINAIRE NATUREL sur 4 bits à partir du signal weathercock appliqué sur ses entrées.

La fonction gray_decoder_4bit est un DECODEUR qui active en sortie un bit parmi 16 du signal wind_dir défini sur 16 bits partir du signal weathercock appliqué sur ses entrées. Les sorties du DECODEUR sont actives à l'état BAS. Le NORD est associé au code weathercock = "0000" et à la position code wind_dir (0).

Table de vérité gray_bin_4bit :

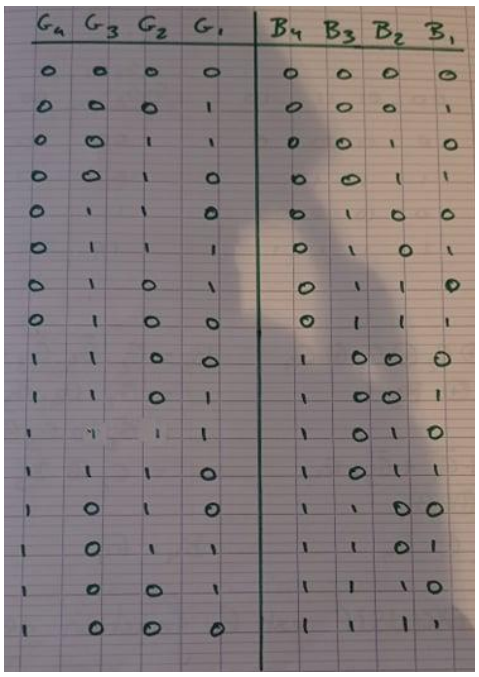
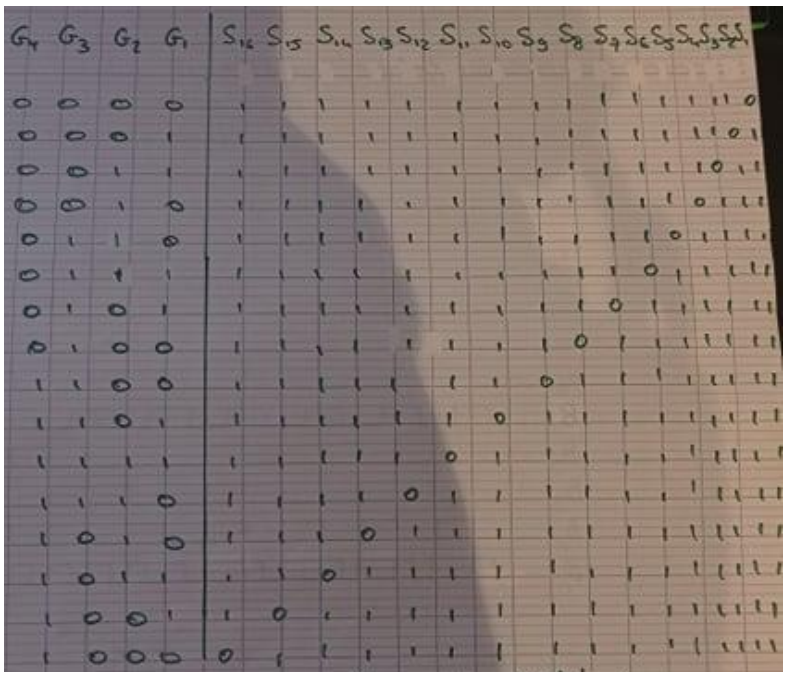
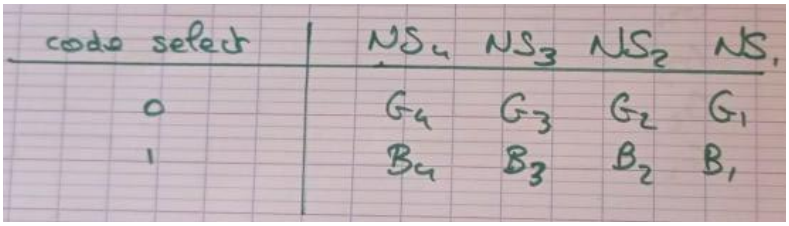


Table de vérité gray_decoder_4bit :



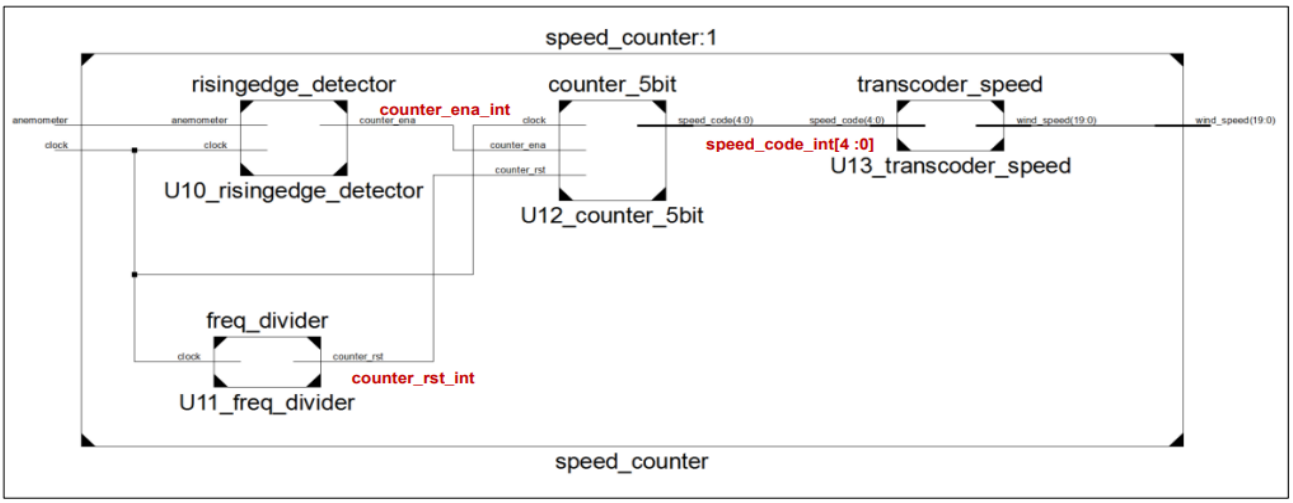
La fonction mux_2x1x4bit est un MULTIPLEXEUR qui transmet sur ses sorties not_code_display défini sur 4 bits, soit le signal weathercock codé en GRAY sur 4 bits, soit le signal binary_code codé en BINAIRE NATUREL sur 4 bits en fonction de la valeur du signal de sélection code_select : code_select = 0 code GRAY, code_select = 1 code BINAIRE NATUREL.

Table de vérité mux_2x1x4bit:



La fonction inv_4bit correspond à 4 INVERSEURS et fournit sur ses sorties le signal code_display défini sur 4 bits. Ce signal inverse du signal not_code_display défini sur 4 bits obtenu en sortie de mux_2x1x4bit.
Ainsi pour chacune des 4 entrées dans la table de vérité, il en sort son inverse.

Speed_counter:



La fonction risingedge_detector compare les signaux anemometer et clock appliqués sur ses entrées et fournit sur sa sortie le signal counter_ena passant à 1 pour chaque front du signal anemometer. On obtient ainsi un train d'impulsion qui évolue au rythme de la rotation de l'anémomètre.

La fonction freq_divider est un diviseur de fréquence du signal clock appliqué sur son entrée. Elle génère régulièrement sur sa sortie, un signal counter_rst de remise à zéro synchrone.

La fonction counter_5bit est un compteur binaire synchrone ayant en entrée les signaux clock, counter_ena et counter_rst et en sortie le signal speed_code codé sur 5 bits. Le compteur compte le nombre d'impulsions du signal counter_ena présentes sur le laps de temps compris entre deux remises à zéro par counter_rst. Plus l'anémomètre tourne vite, plus le nombre d'impulsions est grand.

La fonction transcoder_speedest un TRANSCODEUR qui fournit en sortie le signal wind_speed défini sur 20 bits en fonction de la valeur du signal speed_code codé sur 5 bits appliqué sur ses entrées. Les sorties du TRANSCODEUR sont actives à l'état BAS.

Table de vérité transcoder_speed :

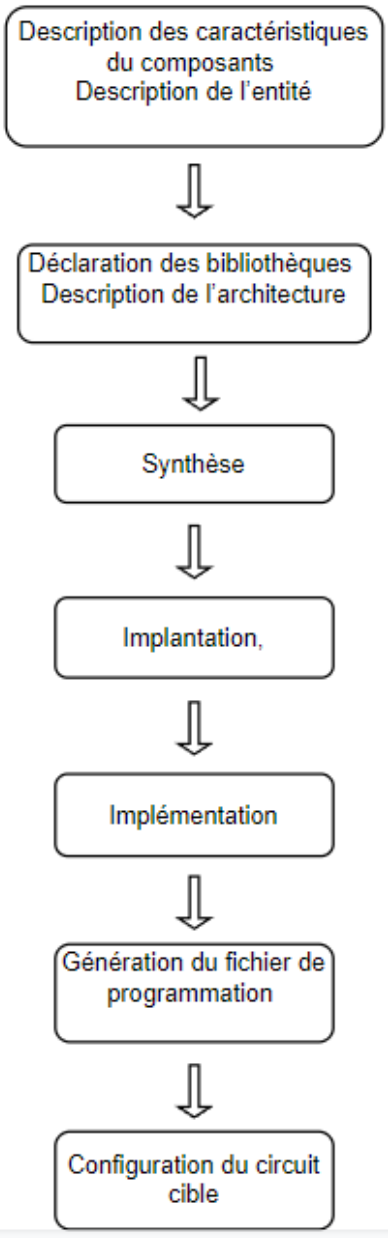
speed_code					wind_speed																
4	3	2	1	0	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	1	1	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	1	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	1	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
0	1	0	1	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
0	1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0
0	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0
0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0
0	0	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
0	0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0
0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

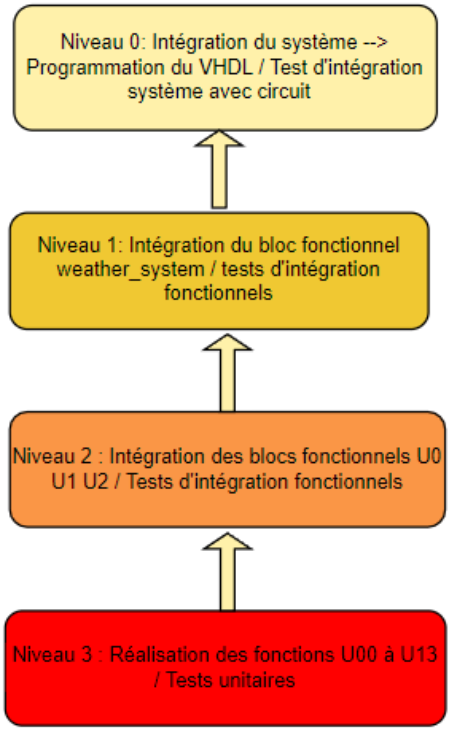
3. DÉVELOPPEMENT DU SOUS-SYSTÈME DE TRAITEMENT À L'AIDE DE L'OUTIL ISE

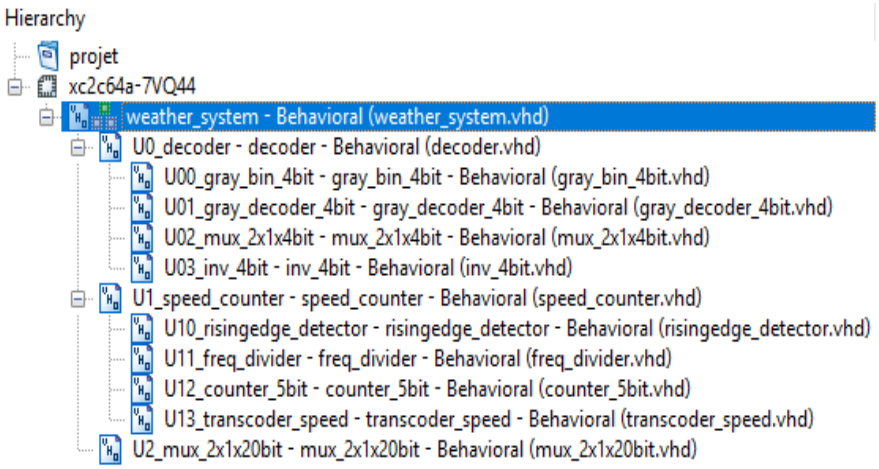
Flot de développement et de "programmation" du CPDL:

Décomposition du composant en sous-systèmes afin de déterminer ses variables d'entrées et de sorties ainsi que les signaux qui connectent les différentes fonctions élémentaires
A partir du TestBench effectué pour chaque sous-système, on décrit leur comportement
Simulation behavioral : permet de nous informer des éventuelles erreurs commises lors de la description du système
Simulation Post-Fit : test du comportement du système, de la cohérence des entrées/sorties (simulation comportementale)
Translate+Fit (idem mais simulation temporelle)
Une fois les vérifications effectuées : on génère le fichier de programmation
Connexion du CPLD à l'ordinateur pour le programmer puis mis en circuit du CPLD pour tester son bon fonctionnement à l'aide de voyants lumineux et d'un anémomètre



Arborescence des fichiers sources :





4. ETUDE DU SOUS-SYSTÈME DE TRAITEMENT

Etude spécifique de la fonction gray_decoder_4bit:

En regardant le schéma on observe que la fonction est construite à partir de 2 portes ET, 26 portes OU, 20 portes OUI et 18 portes NON, avec comme entrée weathercock en 4 bit, et comme sortie wind_dir en 15 bit.
Les résultats obtenus dans la simulation comportementale sont précisément les résultats attendus à travers la table de vérité, ce qui signifie que le comportement de notre fonction est correct.
Enfin avec la simulation Post-Fit en remarque un retard de 6,7 ns, avec 2,4 ns pour le buffer d'entrée et 2,8 ns pour le buffer de sortie.

Etude spécifique de la fonction mux_2x1x4bit:

En regardant le schéma on observe que la fonction est construite à partir de 8 portes ET, 4 portes OU, 13 portes OUI et 4 portes NON, avec 3 entrées: weathercock en 4 bit, binary_code en 4 bit et code_select, et comme sortie: not_code_display en 4bit.

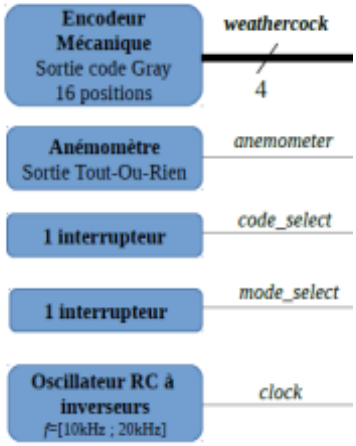
Encore une fois les résultats obtenus dans la simulation comportementale , correspondent aux résultats de la table de vérité, ce qui signifie que le comportement de notre fonction est correct.
Enfin avec la simulation Post-Fit en remarque un retard de 7,5 ns

Bilan des résultats pour l'ensemble du sous-système de traitement :

Bloc	Macro-cellules/64	Nombre entrées-sorties/33	Temps de réponse (ns)
weather_system	55	28	/
decoder	20	25	7.5
gray_bin_4bit XOR	4	8	7.5 .
gray_decoder_4bit	16	20	6.7
mux_2x1x4bit	4	13	7.5
inv_4bit	4	8	6,7
speed_counter	54	22	11
transcoder_speed	20	25	7,5
mux_2x1x20bit	/	/	/

A travers ce tableau on observe que presque toutes les Macro-cellules, ainsi que les entrées et les sorties sont utilisées, pour la programmation du weather_system.
La fonction la plus lente mesurée est le speed counter avec 11 ns de retard, et la fonction mux_2*1*20 bit n'as pas pu être mesurée dû aux trop grand nombre de macro-cellules utilisées.

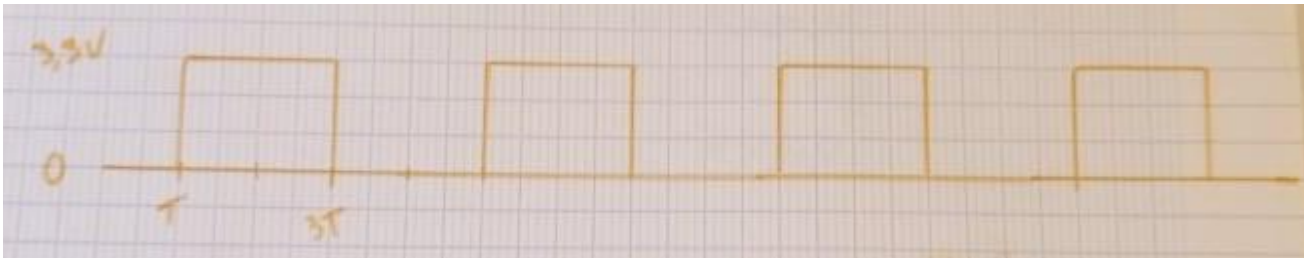
5. INTERFACE DU SOUS-SYSTÈME DE TRAITEMENT AVEC LES SOUS-SYSTÈMES D'ENTRÉE ET SORTIE



Synoptique du sous-système d'entrée.

Allure des signaux :

Sortie de poids faible de la roue codeuse (girouette):
Notre encodeur mécanique alimenté en 3.3V renvoie, selon la position de la girouette, une sortie logique 1 à 3.3V ou 0 à 0V sur chacun des 4 bits.
Ainsi le bits de poids faible, si on suppose un temps T durant lequel la roue reste à chaque position, devrait valoir 0 durant T puis 3.3V durant 2T puis 0V durant 2T puis 3.3V durant 2T etc…



Sortie du bloc oscillateur:
En sortie du bloc oscillateur nous obtenons un signal carré de fréquence 15kHz et de rapport cyclique 50%.

Caractéristiques en tension :

On peut relever dans la documentation "Mixed I/O voltages compatible with 1.5V, 1.8V, 2.5V, and 3.3V logic levels", or nos signaux d'entrées, donc en sortie de la roue codeuse, sont de valeurs logiques de 3.3V. Ainsi les caractéristiques en tension des signaux sont compatibles.

Interfaçage avec le sous-système de sortie :

Dans la documentation des bar-graphs on peut lire que la tension les traversants ne doit pas dépasser 2.4V. Or sans résistance, on les alimenterait en 3.3V. Pour pallier cela on ajoute des résistances de 330Ω pour les alimenter autour de 1.8V, la valeur pour laquelle les bar-graphs commencent à scintiller.

6. CONCLUSION

Pour conclure, nous avons rempli le cahier des charges. Les résultats des simulations du sous-système de traitement correspondent parfaitement aux résultats attendus.

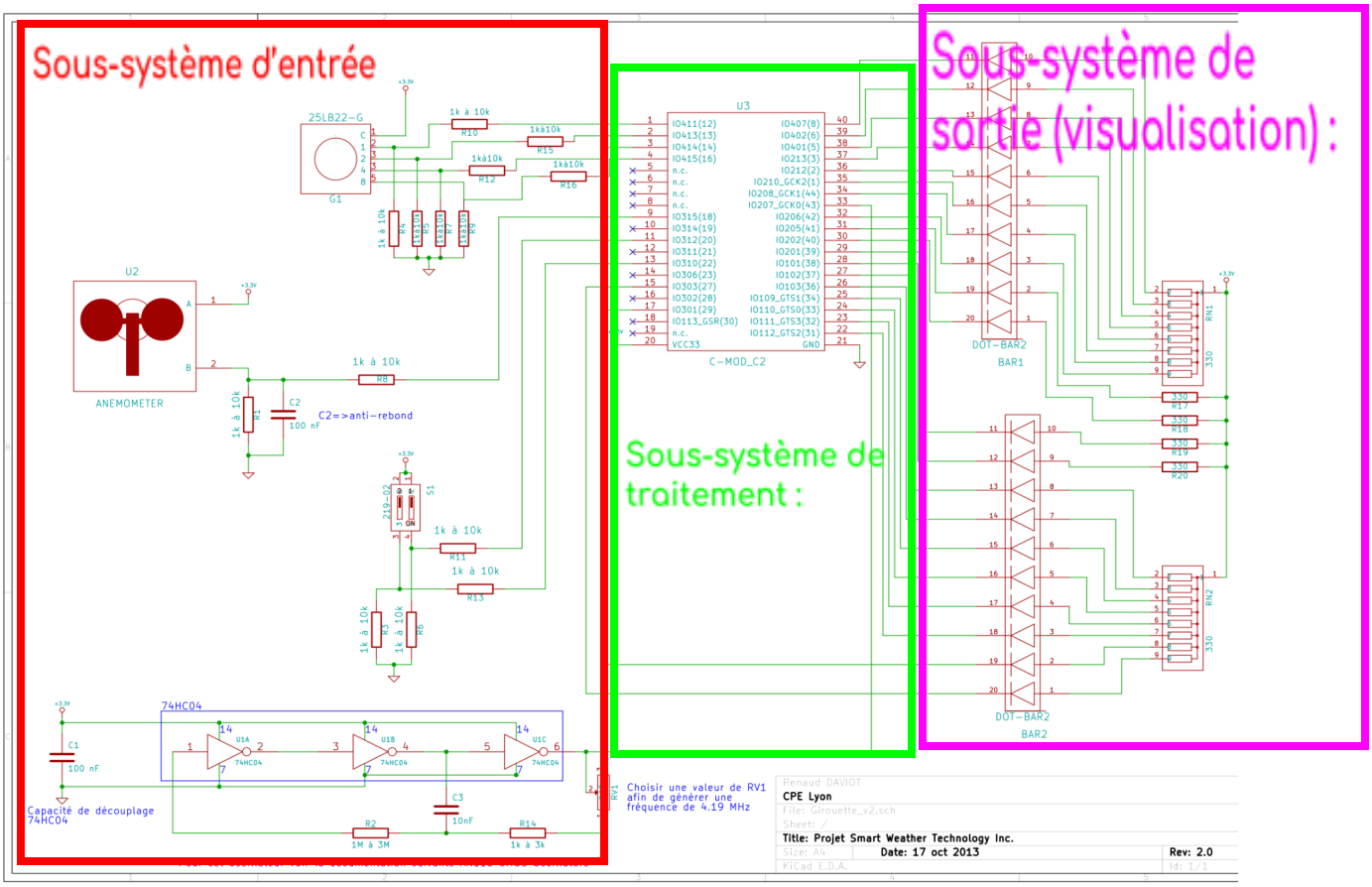
Cependant, nous avons rencontré une difficulté pour le montage et notamment pendant le branchement de l'oscillateur. Nous avons dû générer le signal avec un GBF.

Le système était tout de même fonctionnel en mode GIROUETTE, aussi bien avec l'affichage GRAY qu'avec l'affichage BINAIRE NATUREL.

7. ANNEXES

Table des matières:

Schéma électronique complet :



Liste des composants :

Elément	Composant	Valeur
Encodeur mécanique	Résistances : R4, R5, R7, R9, R10, R12, R15, R16	1 kΩ
Anémomètre à coupelles	Résistances : R1, R8	4.7 kΩ
Anémomètre à coupelles	Condensateur : C2	330 nF
Interrupteurs microswitchs	Résistances : R3, R6, R11, R13	4.7 kΩ
Oscillateur	Résistance R2	1 MΩ
Oscillateur	Résistance R14	1 kΩ

Oscillateur	Potentiomètre RV1	2.2 kΩ
Oscillateur	Condensateur C3	10 nF
Oscillateur	Condensateur C1	100 nF
Bar-Graph	Réseaux de résistances parallèles : RR1, RR2	330Ω
Bar-Graph	Résistances : R17, R18, R19, R20	330Ω

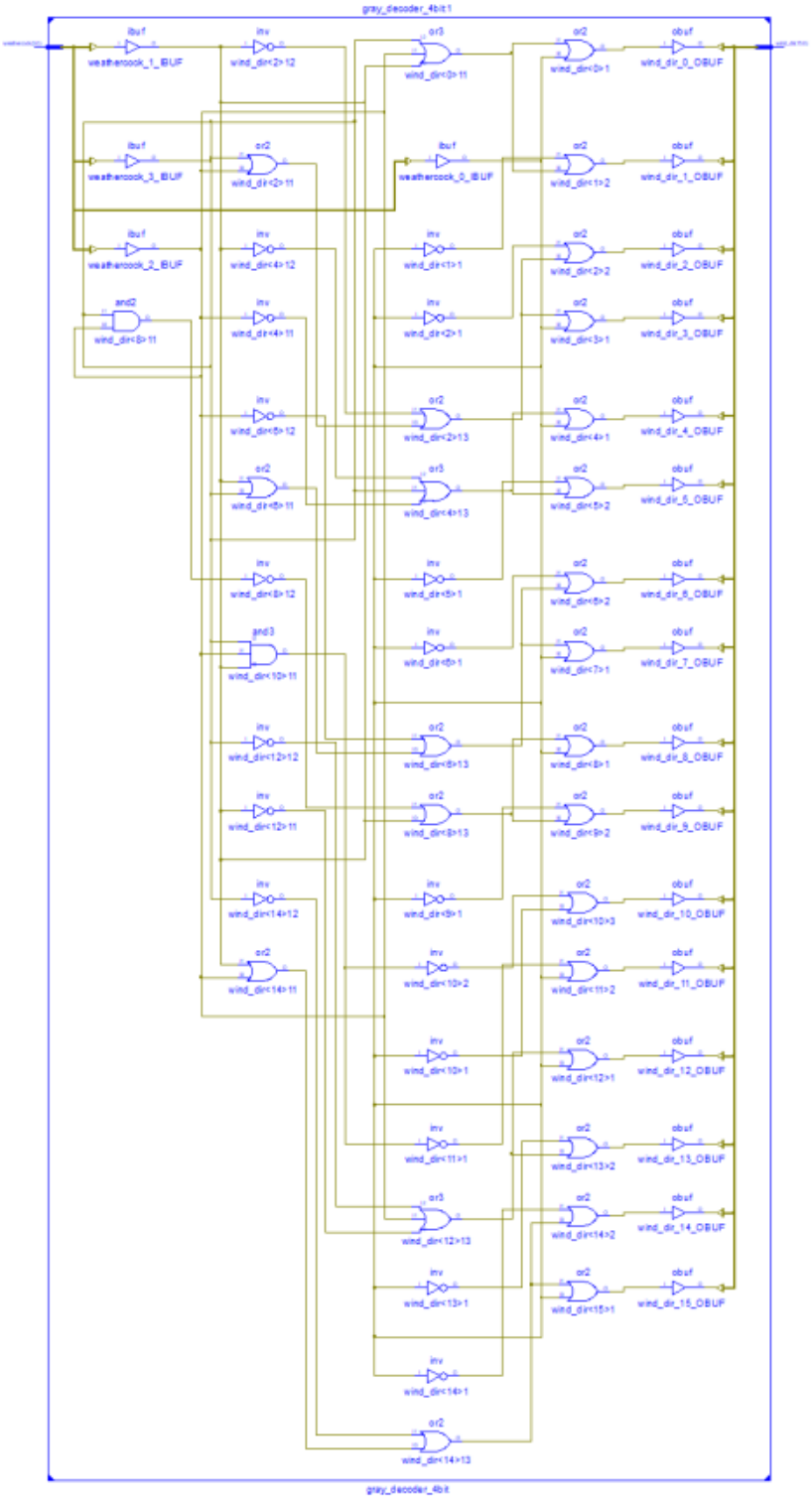
Fonction gray_decoder_4bit:

Impression du fichier source(.vhd):

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity gray_decoder_4bit is
    Port ( weathercock : in  STD_LOGIC_VECTOR (3 downto 0);
           wind_dir : out  STD_LOGIC_VECTOR (15 downto 0));
end gray_decoder_4bit;

architecture Behavioral of gray_decoder_4bit is

begin

wind_dir <= "1111111111111110" when (weathercock="0000") else
                "1111111111111101" when (weathercock="0001") else
                "1111111111111011" when (weathercock="0011") else
                "1111111111110111" when (weathercock="0010") else
                "1111111111101111" when (weathercock="0110") else
                "1111111111011111" when (weathercock="0111") else
                "1111111110111111" when (weathercock="0101") else
                "1111111101111111" when (weathercock="0100") else
                "1111111011111111" when (weathercock="1100") else
                "1111110111111111" when (weathercock="1101") else
                "1111101111111111" when (weathercock="1111") else
                "1111011111111111" when (weathercock="1110") else
                "1110111111111111" when (weathercock="1010") else
                "1101111111111111" when (weathercock="1011") else
                "1011111111111111" when (weathercock="1001") else
                "0111111111111111" when (weathercock="1000") else
                "1111111111111111";

end Behavioral;
```

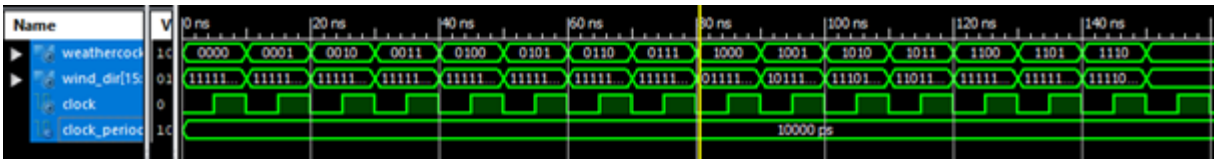
Cette fonction a été directement programmée à partir de la table de vérité.

Vue schématique (vue Technologique):



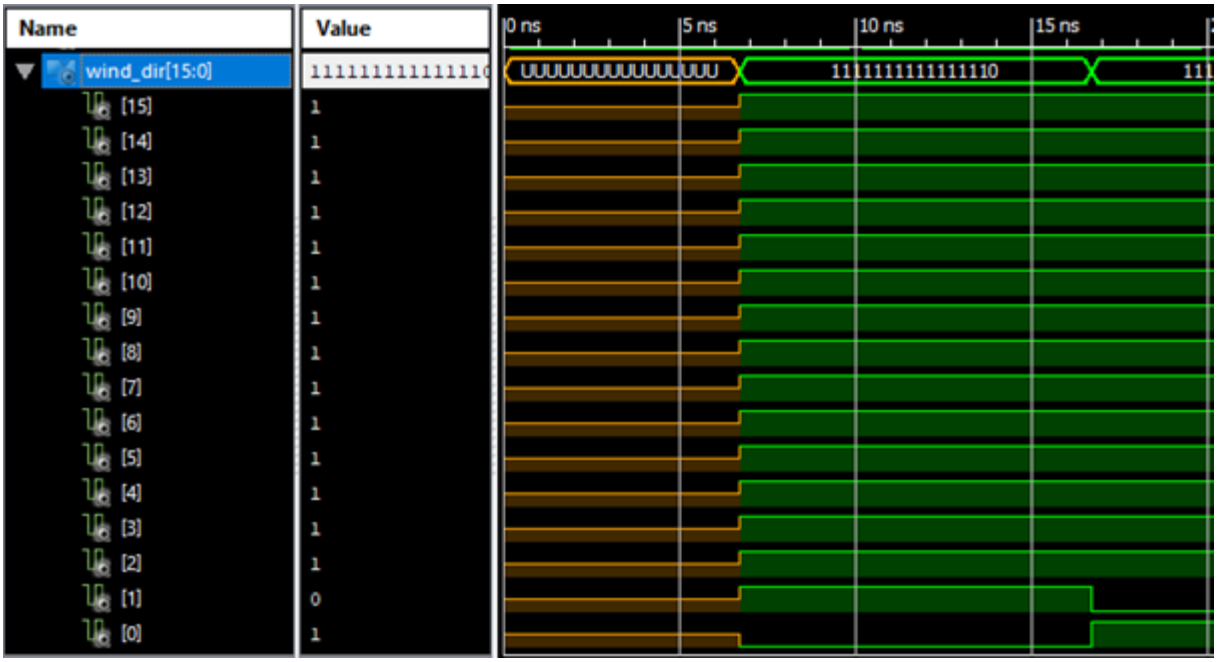
Une version plus optimisée aurait pu être utilisée afin d'éviter la complexité de ce schéma.

Impression de la simulation comportementale (Behavorial):



Les résultats obtenus sont cohérents avec la table de vérité

Impression de la simulation réelle (Post-fit):



Toutes les sorties de la fonction ont le même retard de 6,7 ns..

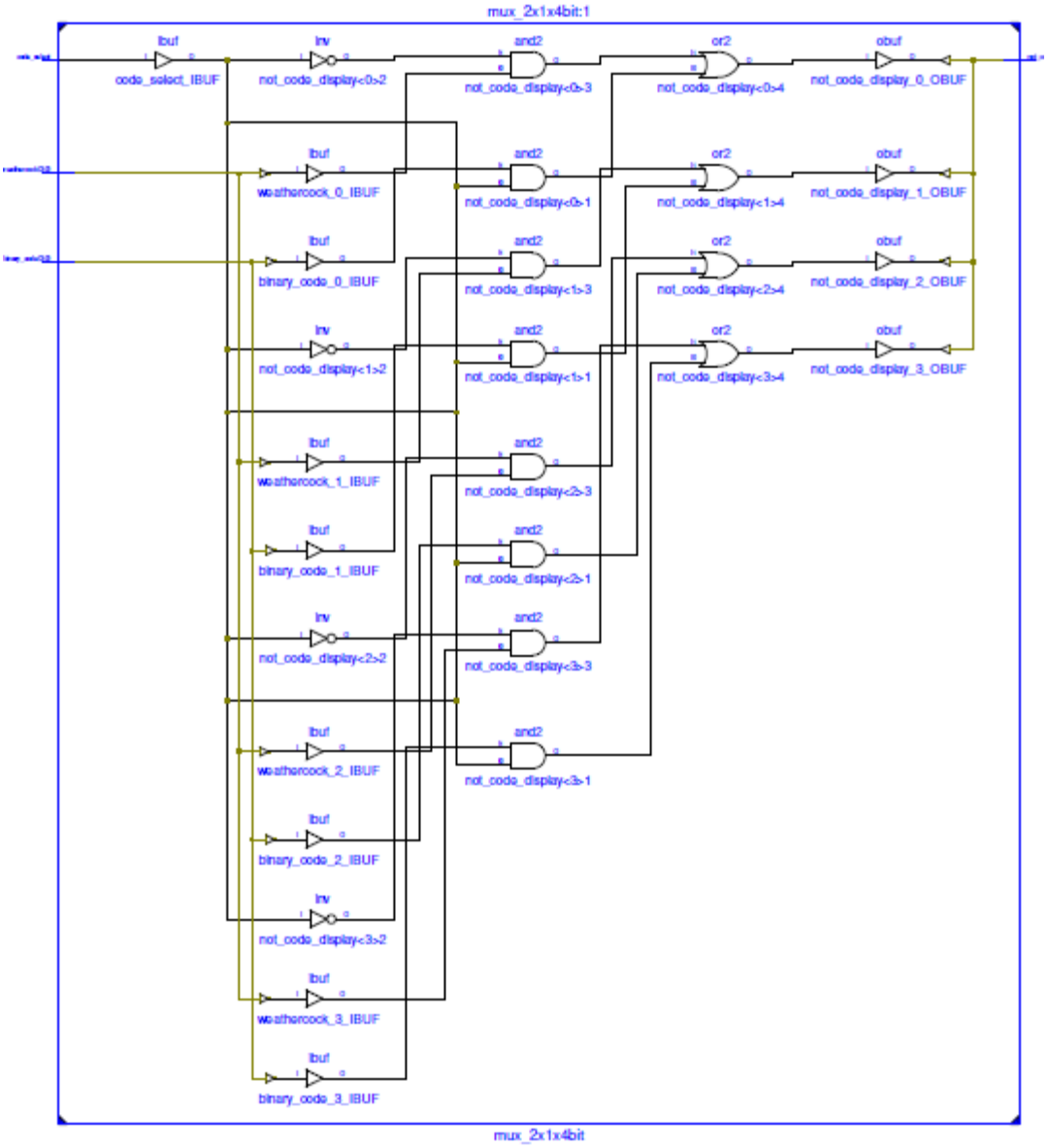
Fonction mux_2x1x4bit:

Impression du fichier source(.vhd):

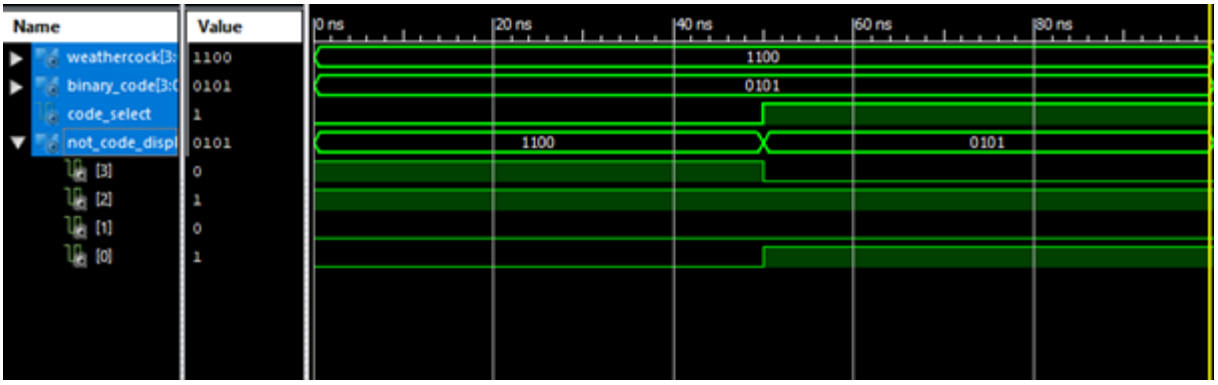
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity mux_2x1x4bit is
    Port ( weathercock : in  STD_LOGIC_VECTOR (3 downto 0);
           binary_code : in  STD_LOGIC_VECTOR (3 downto 0);
           not_code_display : out  STD_LOGIC_VECTOR (3 downto 0);
           code_select : in  STD_LOGIC);
end mux_2x1x4bit;

architecture Behavioral of mux_2x1x4bit is

begin


not_code_display<=binary_code when code_select='1' else weathercock;



end Behavioral;
```

Cette fonction est programmé à donner en sortie binary_code pour code_select=1 ou weathercock si code_select=0.

Vue schématique (vue Technologique):

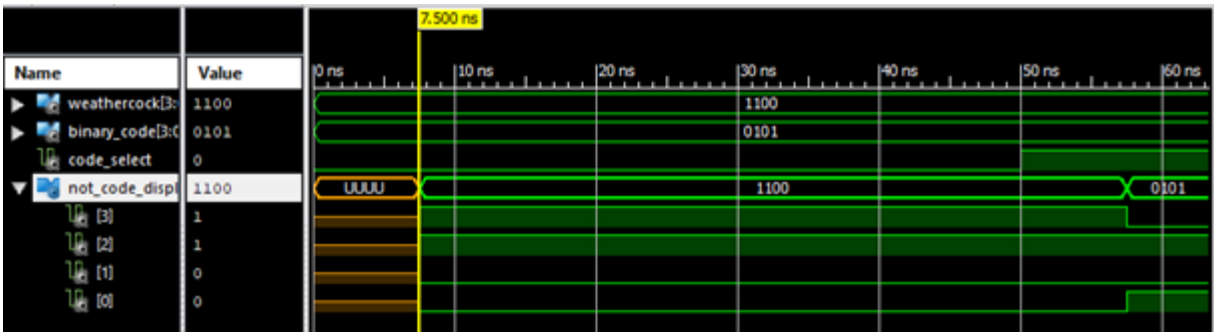


Impression de la simulation comportementale (Behavorial):



Dans cette simulation, seule la sélection du code voulu a été testé, sans passer par toutes les possibilités, ce qui n'aurait pas un grand intérêt dans ce cas.

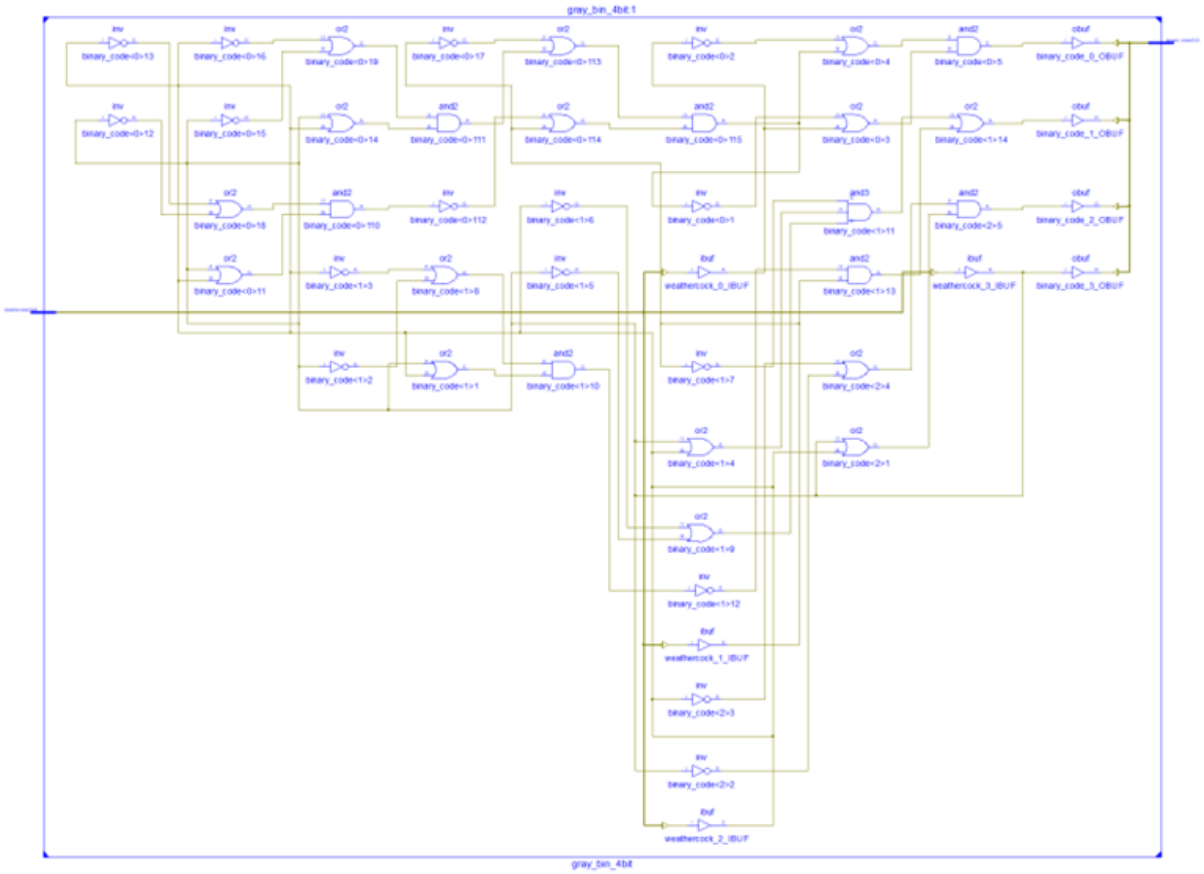
Impression de la simulation réelle (Post-fit):



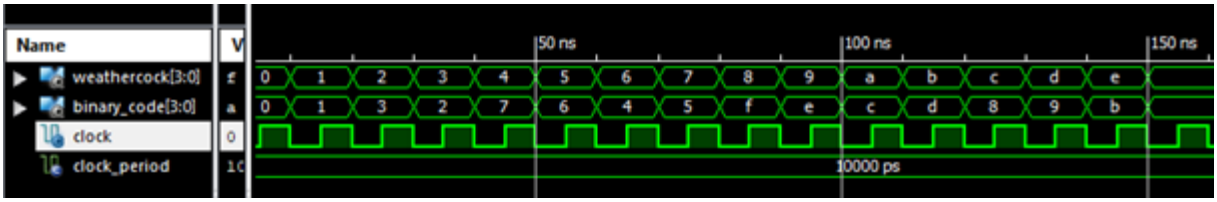
On obtient donc un retard de 7,5 ns.

Fonction « gray_bin_4bit »:

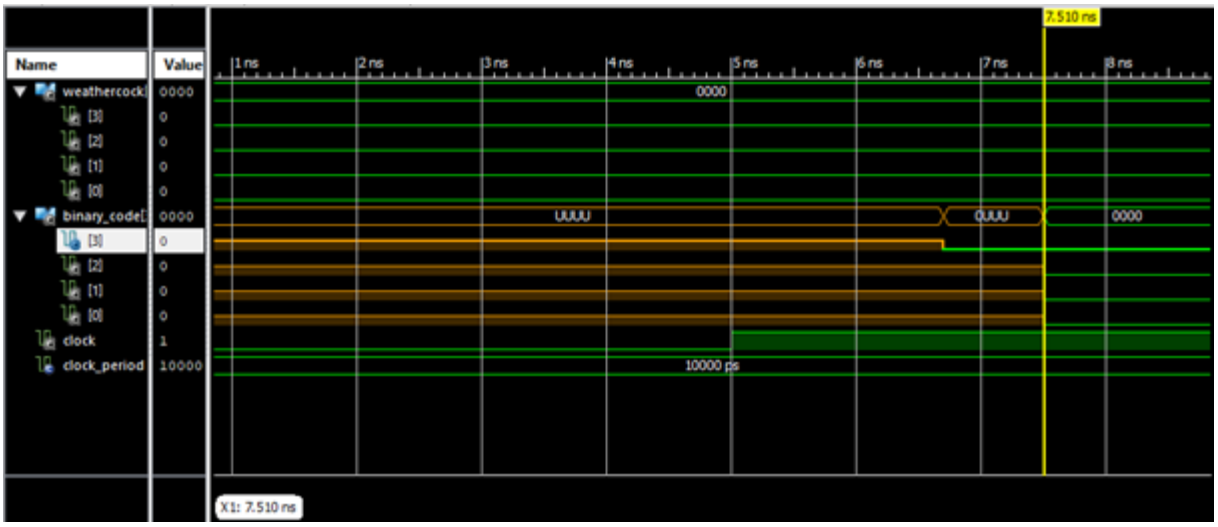
Vue schématique (vue Technologique):



Impression de la simulation comportementale (Behavorial):



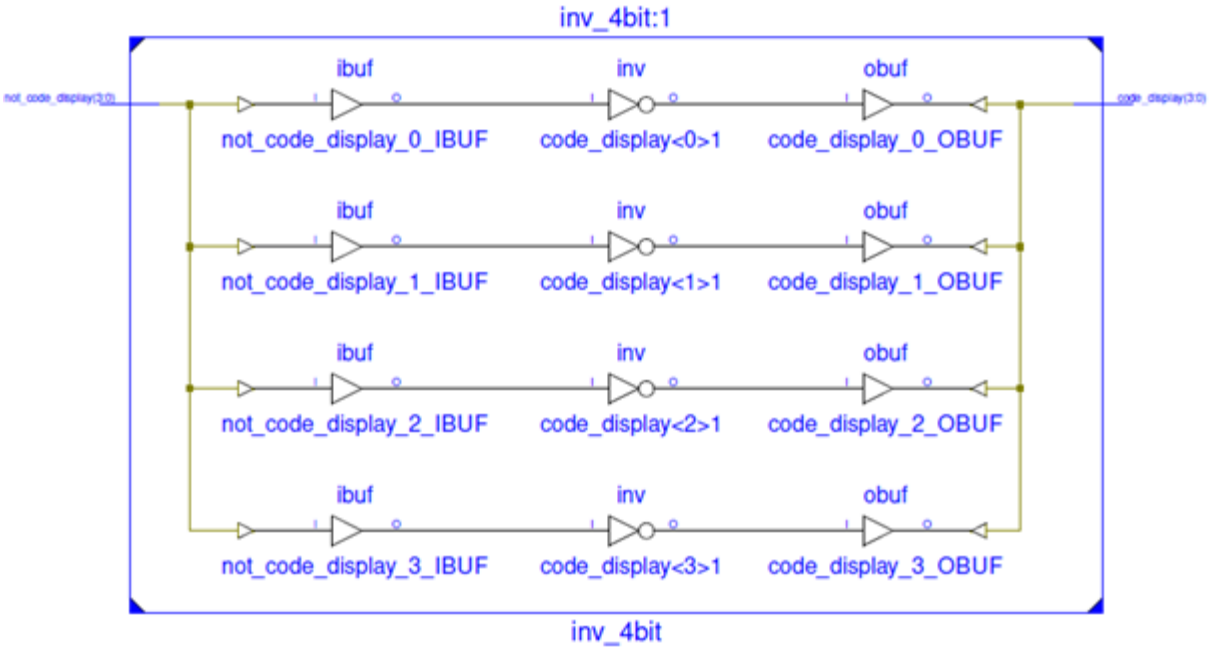
Impression de la simulation réelle (Post-fit):



Le dernier bit arrive plus rapidement que les autres, puisqu' il prend presque directement le signal de l'entrée et interagit moins avec le reste du circuit que les autres sorties.

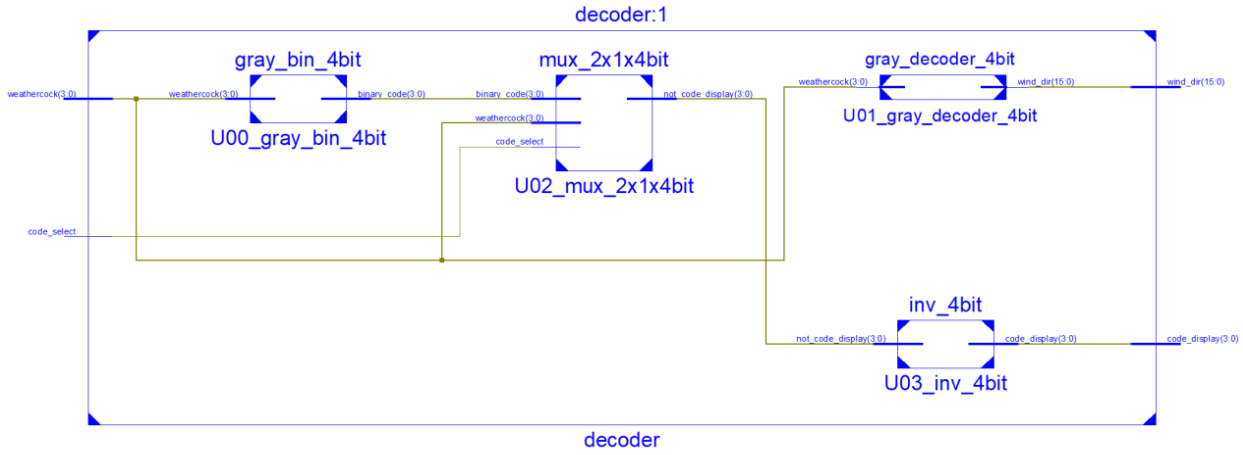
Fonction « inv_4bit »:

Vue schématique (vue Technologique):



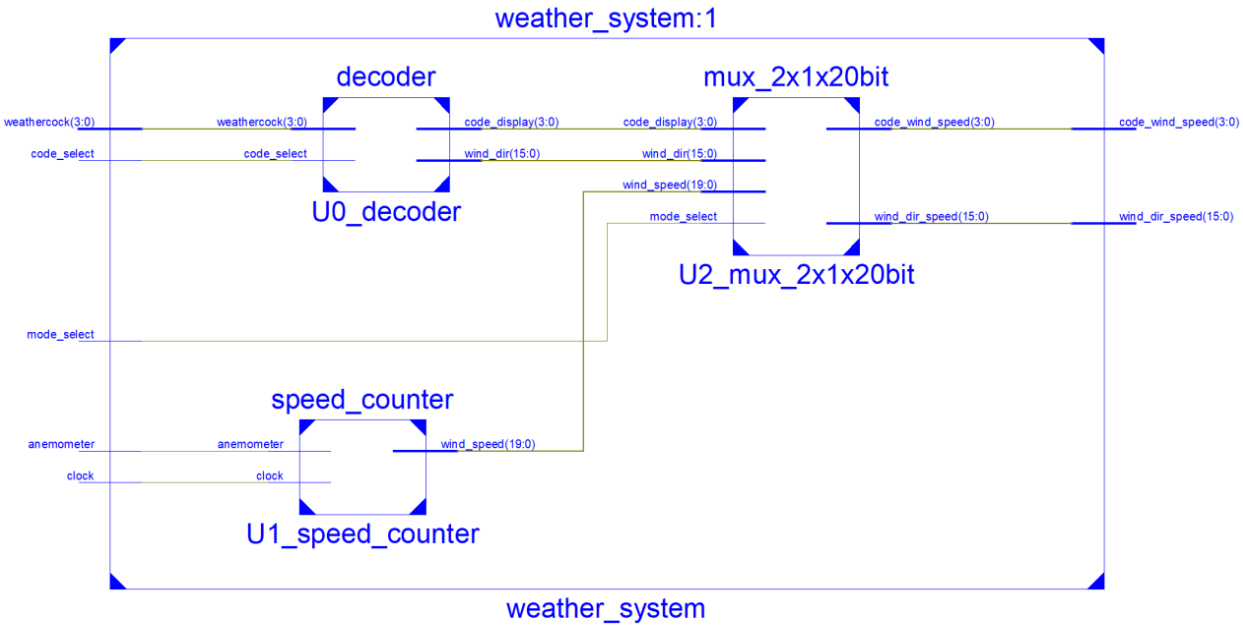
Fonction « decoder »:

Vue schématique RTL:



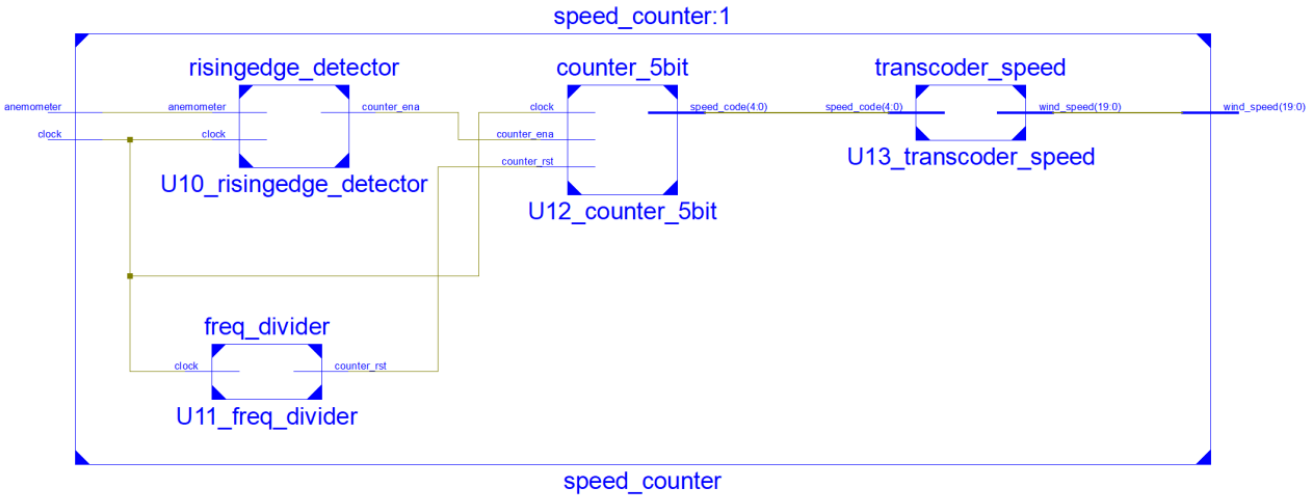
Fonction « weather_system »:

Vue schématique RTL:



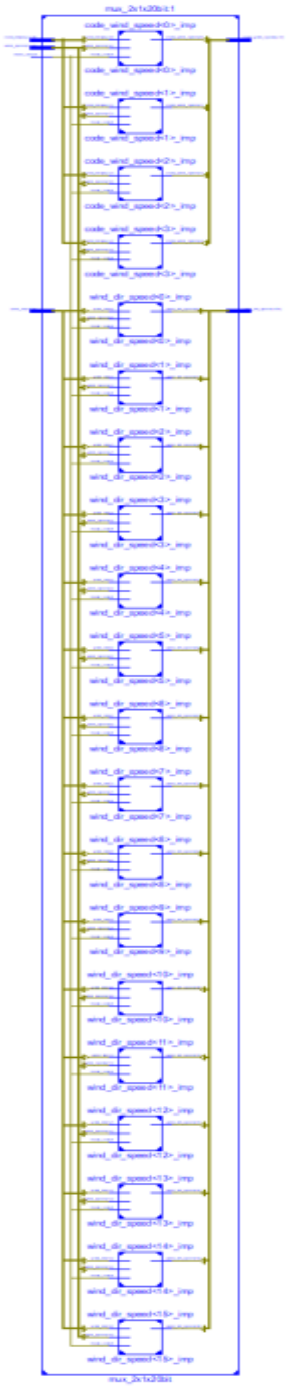
Fonction « speed_counter »:

Vue schématique RTL:

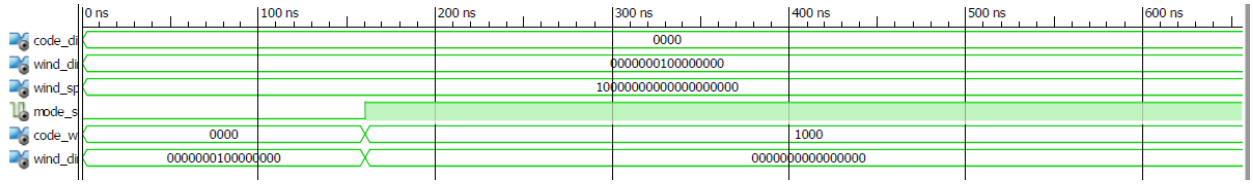


Fonction « mux_2x1x20bit »:

Vue schématique RTL:

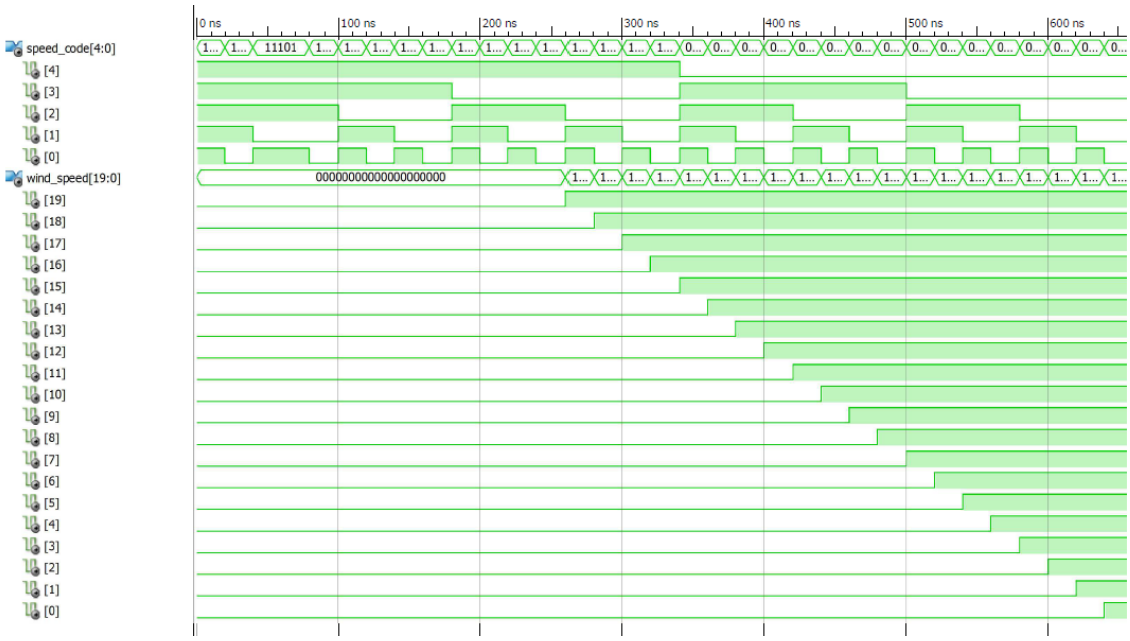


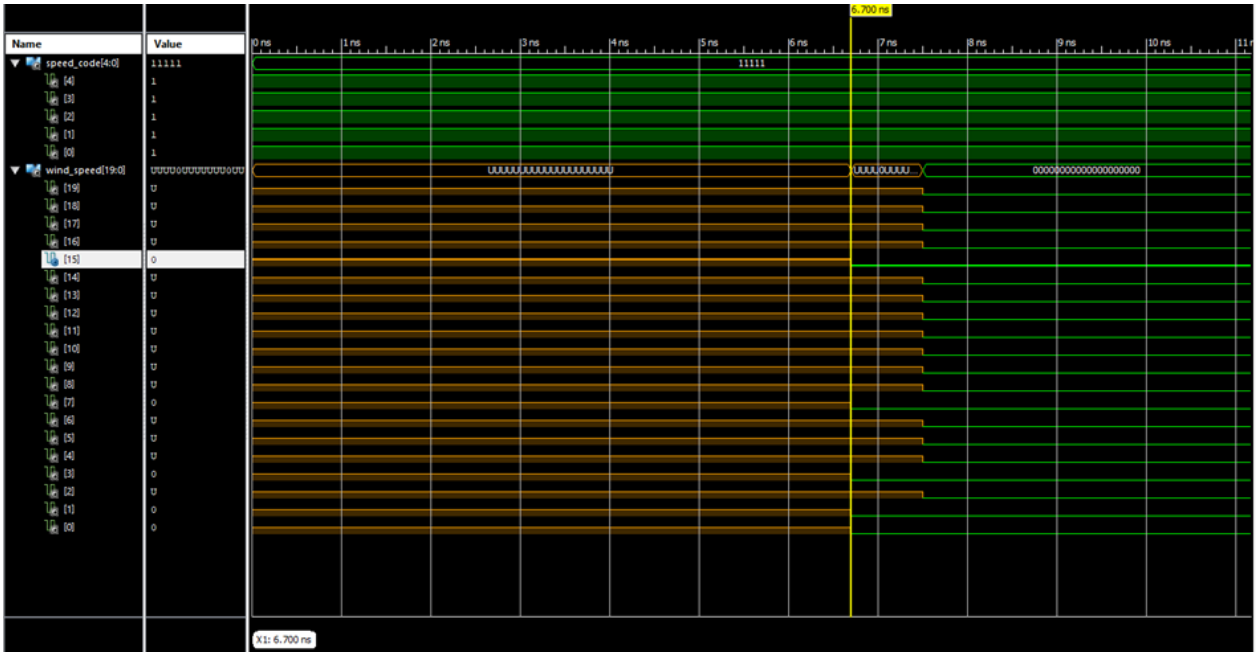
Impression de la simulation comportementale (Behavorial):



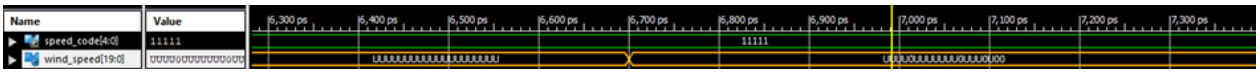
Fonction « transcodeur_speed »:

Impression de la simulation comportementale (Behavorial):





On remarque 2 retards, à 6.7ns aucune information n'est arrivée (Valeur U pour tous les bits = inconnu) en sortie, tandis qu'a 7.5ns elles le sont toutes.



Entre 6.7ns et 7.5ns, on observe en sortie que certains bits valent 0, donc qu'une partie du système est déjà arrivée à la sortie.

Timing Constraints

Constraint Name	Requirement (ns)	Delay (ns)	Paths	Paths Failing
AUTO_TS_F2F	0.0	0.0	0	0
AUTO_TS_P2P	0.0	7.5	82	82
AUTO_TS_P2F	0.0	0.0	0	0
AUTO_TS_F2P	0.0	0.0	0	0

On remarque effectivement qu'il y a 2 temps de réponse différents :

speed_code<3>	wind_speed<8>	7.500
speed_code<3>	wind_speed<9>	7.500
speed_code<4>	wind_speed<10>	7.500
speed_code<4>	wind_speed<11>	7.500
speed_code<4>	wind_speed<12>	7.500
speed_code<4>	wind_speed<13>	7.500
speed_code<4>	wind_speed<14>	7.500
speed_code<4>	wind_speed<16>	7.500
speed_code<4>	wind_speed<17>	7.500
speed_code<4>	wind_speed<18>	7.500
speed_code<4>	wind_speed<19>	7.500
speed_code<4>	wind_speed<2>	7.500
speed_code<4>	wind_speed<4>	7.500
speed_code<4>	wind_speed<5>	7.500
speed_code<4>	wind_speed<6>	7.500
speed_code<4>	wind_speed<8>	7.500
speed_code<4>	wind_speed<9>	7.500
speed_code<0>	wind_speed<0>	6.700
speed_code<1>	wind_speed<0>	6.700
speed_code<1>	wind_speed<1>	6.700
speed_code<2>	wind_speed<0>	6.700
speed_code<2>	wind_speed<1>	6.700
speed_code<2>	wind_speed<3>	6.700
speed_code<3>	wind_speed<0>	6.700
speed_code<3>	wind_speed<1>	6.700
speed_code<3>	wind_speed<3>	6.700
speed_code<3>	wind_speed<7>	6.700
speed_code<4>	wind_speed<0>	6.700
speed_code<4>	wind_speed<15>	6.700
speed_code<4>	wind_speed<1>	6.700

```
    speed_code(1) AND speed_code(0));
wind_speed(7) <= (NOT speed_code(4) AND NOT speed_code(3));
wind_speed(8) <= ((NOT speed_code(4) AND NOT speed_code(3))
    OR (NOT speed_code(4) AND NOT speed_code(2) AND NOT speed_code(1) AND
    NOT speed_code(0)));
wind_speed(9) <= ((NOT speed_code(4) AND NOT speed_code(3))
    OR (NOT speed_code(4) AND NOT speed_code(2) AND NOT speed_code(1)));
wind_speed(10) <= ((NOT speed_code(4) AND NOT speed_code(3))
    OR (NOT speed_code(4) AND NOT speed_code(2) AND NOT speed_code(1))
    OR (NOT speed_code(4) AND NOT speed_code(2) AND NOT speed_code(0)));
wind_speed(11) <= ((NOT speed_code(4) AND NOT speed_code(3))
    OR (NOT speed_code(4) AND NOT speed_code(2)));
wind_speed(12) <= ((NOT speed_code(4) AND NOT speed_code(3))
    OR (NOT speed_code(4) AND NOT speed_code(2))
    OR (NOT speed_code(4) AND NOT speed_code(1) AND NOT speed_code(0)));
wind_speed(13) <= NOT (((speed_code(4))
    OR (speed_code(3) AND speed_code(2) AND speed_code(1))));
wind_speed(14) <= NOT (((speed_code(4))
    OR (speed_code(3) AND speed_code(2) AND speed_code(1) AND
    speed_code(0))));
wind_speed(15) <= NOT speed_code(4);
```

Parmi les sorties les plus rapides, on remarque la 7 et la 15 dont les équations sont de plus très courtes. Malgré la différence de longueur d'équation, leurs temps de réponse sont les mêmes car c'est la technologie CMOS qui est utilisée dans les 2 cas et les portes NOT et NOR possèdent toutes les 2 un seul étage de transistor.

En revanche la sortie 13 passe par plus de portes logiques donc son temps de propagation est plus long (7.5ns).

Caractéristiques de la fonction :

Nombre de macro-cellules : 20/64 (32%) ; nombre de broches d'entrées-sorties : 25/33 (75%) ; temps de réponse : 7.5ns.

Macrocells Used	Pterms Used	Registers Used	Pins Used	Function Block Inputs Used
20/64 (32%)	29/224 (13%)	0/64 (0%)	25/33 (76%)	13/160 (9%)

Feuilles de mesure:

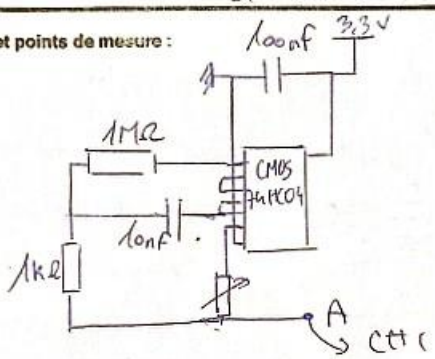
C.P.E. Lyon **FEUILLE DE MESURES AU LABORATOIRE**

Noms : **Groupe :** C

Date : 03/03/22 **Poste :** 3

Objet de la manipulation : Vérifier le bon fonctionnement de l'oscillateur.

Schéma du montage et points de mesure :



Protocole de travail (si nécessaire) :

On cherche à obtenir un signal carré de fréquence comprise entre 12 et 15 kHz.

Mesures + Conditions de mesure : Conditions : Couplage CC
Sonde x1
Sensibilité verticale : 1V
sensibilité horizontale : 25µs

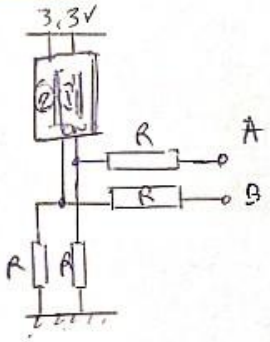
C.P.E. Lyon **FEUILLE DE MESURES AU LABORATOIRE**

Noms : **Groupe :** C

Date : **Poste :** 3

Objet de la manipulation : Vérifier le fonctionnement de l'interrupteur micro. switchs.

Schéma du montage et points de mesure :



R = 1 kΩ

Protocole de travail (si nécessaire) :

- Relever les tensions en A et B dans les 4 positions possibles.

Mesures + Conditions de mesure :

Conditions { couplage CC ; Sonde x1 ; Sensibilité verticale : 1V ; horizontale : 250 µs

1	2	A	B
H	H	3,3V	3,3V
H	L	3,3V	0V
L	H	0V	3,3V
L	L	0V	0V

CONCLUSION : L'interrupteur fonctionne.

C.P.E. Lyon **FEUILLE DE MESURES AU LABORATOIRE**

Noms :

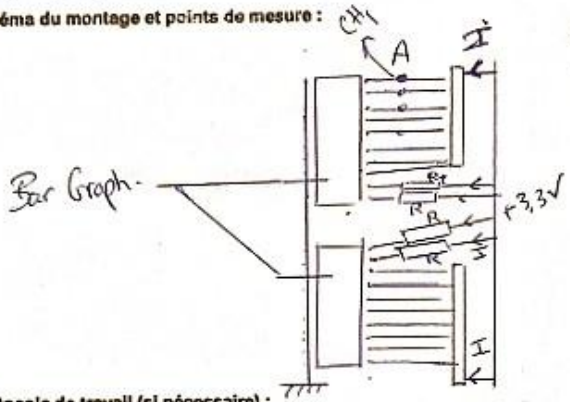
Date :

Groupe : C

Poste : 3

Objet de la manipulation : Vérification du bon fonctionnement des bars graphs à l'état bas.

Schéma du montage et points de mesure :



4x Resistance de 330 Ω (R)
2x Reseaux de résistance 330 Ω.
2x Bar Graph.

I = 0,005 V.

Protocole de travail (si nécessaire) :

- Vérifier que les diodes s'allument.

Mesures + Conditions de mesure :

Conditions : Sonde x1
Sensibilité verticale : 1V
horizontale : 50µs
Couplage AC

On mesure une tension de 1,8V aux différents points entre les resistances et les bar graphs.

Conclusion : Les diodes s'allument bien à l'état bas.
$V_{Diode} = 3,3 - RI = 3,3 - 1,65 = 1,65\ V \sim 1,8V > V_{seuil}$

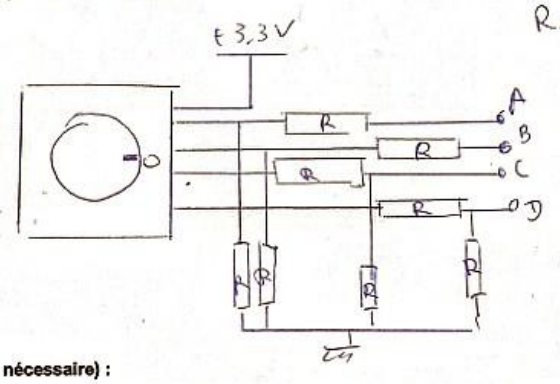
C.P.E. Lyon **FEUILLE DE MESURES AU LABORATOIRE**

Noms : Kleitz / Cornaton / Michalopoulos / Hayotte **Groupe :** C

Date : 03/01/22 **Poste :** 3

Objet de la manipulation : Vérifier le bon fonctionnement de l'encodeur mécanique

Schéma du montage et points de mesure :



Protocole de travail (si nécessaire) :

Pour chaque position relever les tensions aux points A, B, C, D.

Mesures + Conditions de mesure :

Conditions : Couplage CC.
Sonde x1
Sensibilité verticale : 1V.
horizontale : 25µs.

On mesure : en position 0 :
A = B = C = D = 0V...

A	B	C	D	(Position 0)
0V	0V	0V	0V	
3,3V	0V	0V	0V	
3,3V	3,3V	0V	0V	
0V	3,3V	0V	0V	
0V	3,3V	3,3V	0V	
3,3V	3,3V	3,3V	0V	

A	B	C	D
3,3V	0V	3,3V	0V
0V	0V	3,3V	0V
0V	0V	3,3V	3,3V
3,3V	0V	3,3V	3,3V
3,3V	3,3V	3,3V	3,3V
0V	3,3V	3,3V	3,3V
0V	3,3V	0V	3,3V
3,3V	3,3V	0V	3,3V

A	B	C	D
3,3V	0V	0V	3,3V
0V	0V	0V	3,3V

Conclusion : l'encodeur mécanique fonctionne.