UNIVERSITÉ DE SHERBROOKE
Faculté de génie
Département de génie électrique et génie informatique

RAPPORT APP1

Logique combinatoire
GEN420, GEN430

Présenté à
Équipe de formateurs de la session S4

Présenté par
Alexis Guérard – guea0902
Raphael Bouchard – bour0703

Sherbrooke – 17 janvier 2024

TABLE DES MATIÈRES

LISTE DES FIGURES

LISTE DES TABLEAUX

1. DÉMARCHE POUR LA RÉALISATION DU THERMO2BIN

1.1 GÉNÉRATION DES ÉQUATIONS ET CALCUL DE LA FRÉQUENCE D'OPÉRATION DU MODULE THERMO2BIN

La méthodologie pour obtenir les spécifications du module thermo2bin a impliqué l'utilisation d'une fonction logique pour l'erreur et de trois autres pour la conversion du thermométrique au binaire, afin de simplifier le traitement. Ces deux blocs ont été appliqués à plusieurs reprises pour couvrir les 12 bits encodés thermométriquement, aboutissant à une valeur binaire en 4 bits. Pour déterminer les équations simplifiées, l'approche initiale a consisté à identifier les entrées, constituées de blocs de 3 bits, et les sorties (1 bit pour l'erreur et 3 bits pour la transformation). Une table de vérité a ensuite été créée en utilisant ces valeurs. Un tableau de Karnaugh a été généré pour chaque valeur de sortie, et les équations ont pu être déduites en regroupant les valeurs dans ces tableaux, comme présenté dans l'annexe. Pour F_0, la valeur est de 0 puisqu'il n'y a aucun moment où le bit est de 1. Pour F_1, l'équation trouvé est la suivante :

$$F_1 = BC$$

Pour F_2 :

$$F_2 = A \oplus B \oplus C$$

Et pour l'erreur e :

$$e = A\,\bar{B} + B\,\bar{C}$$

En supposant que nos portes logiques ont un délai unique de 5 ns, notre circuit ne peut opérer à 20 MHz. En effet, à cette fréquence, la période est 50 ns. Le chemin le plus long dans notre schéma contient 14 porte logique. Alors 14 fois 5 ns égal 70 ns et c'est donc plus grand que 50 ns.

1.2 IMPLÉMENTATION DE SCHÉMAS/CODE VHDL

L'approche utilisée pour configurer le module thermo2bin de l'APP a consisté à diviser les 12 bits encodés thermométriquement en quatre sous-groupes de 3 bits. Cette méthode permet de décomposer ce module complexe en sous-modules plus simples à traiter. Pour chaque sous-groupe de 3 bits, nous avons créé une table de vérité et un tableau de Karnaugh afin de

déterminer les fonctions logiques nécessaires pour convertir les bits encodés thermométriquement en une valeur binaire.

Les résultats obtenus pour les quatre sous-groupes ont ensuite été additionnés à l'aide de l'additionneur 4 bits, permettant d'obtenir le nombre binaire correct en fonction de ce qui a été reçu en entrée. Pour faciliter cette opération, un bit de 0 a été ajouté au groupe de 3 bits. Le sous-module responsable de ces opérations prend donc en entrée 3 bits, et sa sortie est un ensemble de 4 bits.

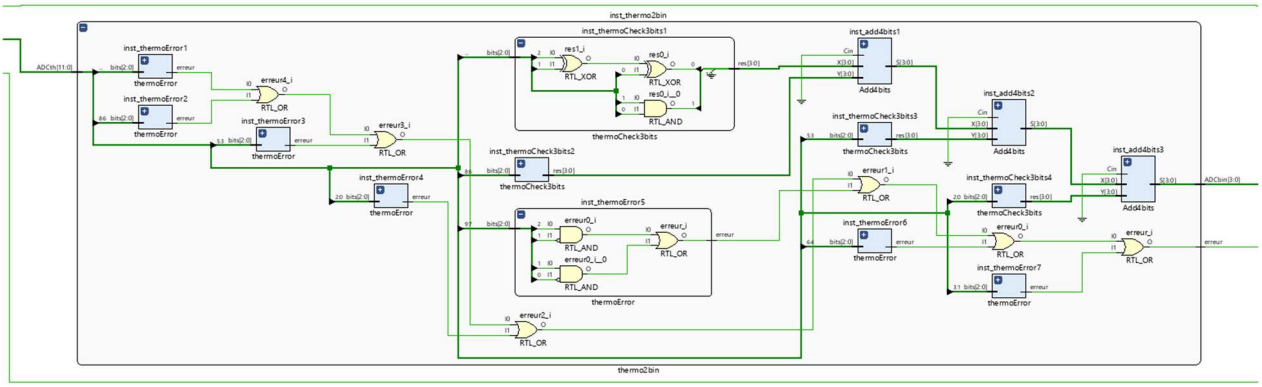

Figure 1 : Schéma Bloc du module Thermo2bin

Pour vérifier d'éventuelles erreurs, un autre sous-module a été utilisé. En utilisant une table de vérité et un tableau de Karnaugh, une fonction logique a été déterminée pour détecter les erreurs en envoyant un signal de 1 ou de 0. Pour garantir une détection fiable des erreurs, ce sous-module est répété sept fois. En effet, la vérification est effectuée sur les quatre sous-groupes de 3 bits que nous avons isolés pour la conversion, et elle est répétée trois autres fois pour vérifier l'absence d'erreurs entre ces sous-groupes. La figure ci-dessous illustre clairement la méthodologie utilisée :

Figure 2 : Exemple visuel de la vérification d'erreur

Ensuite, à l'aide de l'opérateur logique "OU", les résultats des sept vérifications sont évalués. Si l'une ou plusieurs erreurs sont détectées, la sortie "erreur" prendra la valeur 1.

2. SIMULATION DU PROJET COMPLET

Pour nous assurer de couvrir tous les cas de figure importants sans explorer les 2^12 possibilités d'entrée du Thermo2Bin, nous avons décidé d'incorporer les scénarios les plus favorables et les plus défavorables dans nos entrées. En effet, en introduisant les scénarios les plus critiques, notre module sera en mesure de repérer tout type d'erreur. Dans le cas du Thermo2Bin, le scénario le plus défavorable se produit lorsque nous avons une séquence de bits majoritairement à 1, avec un 0 isolé dans cette chaîne, par exemple "110111111111". Ceci s'explique par le fait qu'il est plus efficace pour notre module de détection d'erreur de repérer des bits à 0 que d'en trouver un.

Figure 3 : Chronogrammes de nos tests

La figure ci-dessus illustre les résultats de notre simulation du projet complet. Nos tests se déroulent correctement, et les vérifications d'erreurs (carré rouge) affichent une erreur sur chacun des sept segments, démontrant ainsi que la méthode de vérification des erreurs sur le thermo2bin est appropriée. De plus, toutes les entrées valides (carré jaune) fonctionnent correctement, que ce soit l'affichage en décimal, hexadécimal et la valeur moins 5. Les LED fonctionnent également correctement (carré vert). Il y a tout le temps seulement une LED d'allumé. Une erreur est également affichée lorsque BTN0 et BTN1 sont poussés.

3. COMPATIBILITÉ DES SIGNAUX ENTRE LE FPGA ET LA DEL D2

NC7SP04	
V_{OH}	1.1 V
V_{OL}	0.1 V

74V1T04	
V_{IH}	2 V
V_{IL}	0.8 V

Tableau 1 : Tableau de compatibilité des signaux entre le NC7SP04 et le 74V1T04

Le problème principal expliquant pourquoi la DEL D2 ne s'allume pas, même lorsque le FPGA génère une valeur logique de « 1 », se retrouve dans l'incompatibilité des signaux entre ces deux modules. En effet, le 74V1T04 nécessite une tension d'entrée élevée (V_{IH}) qui n'est pas atteinte par le NC7SP04. Le V_{OH} du NC7SP04 est inférieur au V_{IN} du 74V1T04, ce qui signifie que le 74V1T04 atteint jamais sa tension d'entrée pour fonctionner correctement. Ainsi, les signaux envoyés par le FPGA n'atteignent jamais la DEL D2.

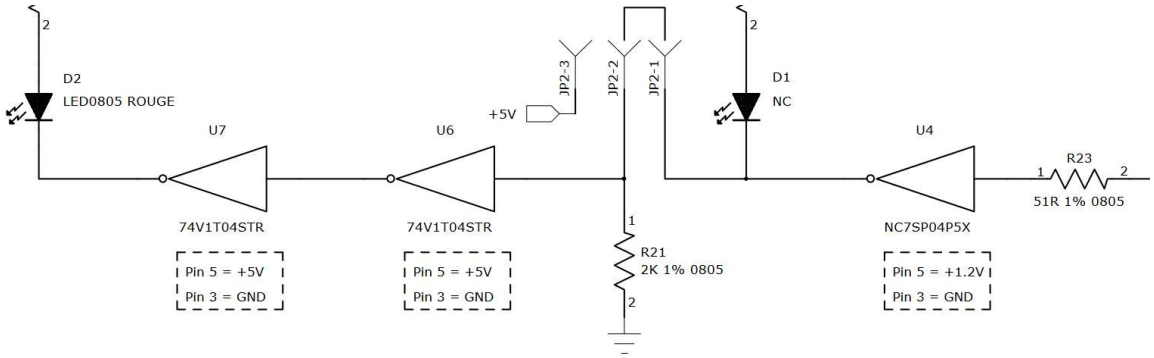

Figure 4 : Carte ADCThermométrique - schéma

Annexe A SCHÉMA BLOCS, TABLES DE VÉRITÉ ET TABLEAUX DE KARNAUGH

A	B	C	F_0	F_1	F_2	E
0	0	0	0	0	0	0
0	0	1	0	0	1	0
0	1	0	D	D	D	1
0	1	1	0	1	0	0
1	0	0	D	D	D	1
1	0	1	D	D	D	1
1	1	0	D	D	D	1
1	1	1	0	1	1	0

Tableau 2 : Table de vérité du Thermo2Bin

C \ AB	00	01	11	10
0	0	D	D	D
1	0	0	0	D

Tableau 3 : Tableau de Karnaugh de F_0

C \ AB	00	01	11	10
0	0	D	D	D
1	0	1	1	D

Tableau 4 : Tableau de Karnaugh de F_1

C \ AB	00	01	11	10
0	0	D	D	D
1	1	0	1	D

Tableau 5 : Tableau de Karnaugh de F_2

C	AB	00	01	11	10
0		0	1	1	1
1		0	0	0	1

Figure 5 : Tableau de Karnaugh de E

Annexe B CODE VHDL DU THERMO2BIN

Code VHDL pour passer de thermométrique 12 bits à binaire 4 bits

```
entity thermoCheck3bits is

    Port ( bits : in STD_LOGIC_VECTOR (2 downto 0);

           res : out STD_LOGIC_VECTOR (3 downto 0));

end thermoCheck3bits;

architecture Behavioral of thermoCheck3bits is

begin

    res(3) <= '0';

    res(2) <= '0';

    res(1) <= bits(1) and bits(0);

    res(0) <= bits(2) xor bits(1) xor bits(0);

end Behavioral;
```

Code VHDL pour vérifier les erreurs du Thermo2Bin

```
entity thermoError is

    Port ( bits : in STD_LOGIC_VECTOR (2 downto 0);

           erreur : out STD_LOGIC);

end thermoError;

architecture Behavioral of thermoError is

begin

    erreur <= (bits(2) and (not bits(1))) or (bits(1) and (not bits(0)));

end Behavioral;
```

Code VHDL du module Thermo2Bin

```
architecture Behavioral of thermo2bin is
component thermoCheck3bits is
    Port ( bits : in STD_LOGIC_VECTOR (2 downto 0);
           res : out STD_LOGIC_VECTOR (3 downto 0));
end component;
component thermoError is
    Port ( bits : in STD_LOGIC_VECTOR (2 downto 0);
```

```
           erreur : out STD_LOGIC);
end component;
component Add4bits is
    Port ( X : in STD_LOGIC_VECTOR (3 downto 0);
           Y : in STD_LOGIC_VECTOR (3 downto 0);
           Cin : in STD_LOGIC;
           Cout : out STD_LOGIC;
           S : out STD_LOGIC_VECTOR (3 downto 0));
end component;
begin
    inst_thermoError1 : thermoError
        port map (
            bits => ADCth (11 downto 9),
            erreur => er1);
     inst_thermoError2 : thermoError
        port map (
            bits => ADCth (8 downto 6),
            erreur => er2);
    inst_thermoError3 : thermoError
        port map (
            bits => ADCth (5 downto 3),
            erreur => er3);
    inst_thermoError4 : thermoError
        port map (
            bits => ADCth (2 downto 0),
            erreur => er4);
    inst_thermoError5 : thermoError
        port map (
            bits => ADCth (9 downto 7),
            erreur => er5);
    inst_thermoError6 : thermoError
        port map (
            bits => ADCth (6 downto 4),
            erreur => er6);
    inst_thermoError7 : thermoError
        port map (
            bits => ADCth (3 downto 1),
            erreur => er7);
     erreur <= er1 or er2 or er3 or er4 or er5 or er6 or er7;
     inst_thermoCheck3bits1 : thermoCheck3bits
        port map (
            bits => Adcth (11 downto 9),
```

```
            res => group1);
      inst_thermoCheck3bits2 : thermoCheck3bits
         port map (
            bits => Adcth (8 downto 6),
            res => group2);
      inst_thermoCheck3bits3 : thermoCheck3bits
         port map (
            bits => Adcth (5 downto 3),
            res => group3);
      inst_thermoCheck3bits4 : thermoCheck3bits
         port map (
            bits => Adcth (2 downto 0),
            res => group4);
      inst_add4bits1 : Add4bits
         port map (
            X => group1,
            Y => group2,
            Cin => '0',
            S => result1);
      inst_add4bits2 : Add4bits
         port map (
            X => result1,
            Y => group3,
            Cin => '0',
            S => result2);
      inst_add4bits3 : Add4bits
         port map (
            X => result2,
            Y => group4,
            Cin => '0',
            S => ADCbin);
end Behavioral;
```