UNIVERSITÉ DE SHERBROOKE

UNIVERSITÉ DE SHERBROOOKE

Faculté de génie

Département de génie électrique et génie informatique

APP 1 : Logique combinatoire

GEN430 01,GEN430 02,GEN420 01,GEN420 02

Présenté à

Marc-André Tétrault et Réjean Fontaine

Présenté par

Shawn Miller-Morneau

Alexis Juteau

Sherbrooke - 18 janvier 2023

Table des matières

Table des figures

1 Démarches thermo2bin

1.1 Génération des équations et calcul de la fréquence d'opération du module thermo2bin

Cette section vise à décoder un code thermométrique de 12 bits en un signal non signé de 4 bits. Pour ce faire, le module thermo2bin sera composé de 5 sous-modules pour tester le code thermométrique et de 2 additionneurs. Le sous-module de test prend 4 bits en entrée et sort les mêmes 4 bits plus l'erreur. Ce module est inséré tous les 2 bits pour minimiser le niveau d'erreur du signal. Une fois que tous les bits ont été testés, les additionneurs de 4 bits additionnent les 8 premiers bits et le résultat est additionné aux 4 derniers bits restants, donnant ainsi une sortie de 4 bits. Le schéma en bloc est présenté en annexe4.1.

Pour arriver à cette solution de gestion des 4 bits, il est nécessaire de développer une table de vérité pour évaluer les équations des 3 bits de sorties et de la gestion d'erreur. La table est disponible en annexe sous le titre "Table de vérité Thermo2bin". Le bit le plus significatif de la sortie est ignoré car avec 4 bits d'entrées du code thermométrique, il ne peut pas dépasser la valeur 4. La sortie comprend alors 3 bits plus le bit d'erreur. Pour simplifier ces sorties, la table de Karnaugh est appliquée à chaque sortie. Les résultats sont présentés en annexe 4.3 suivis du code VHDL 4.4.

En supposant que toutes les portes logiques du schéma ont un délai unique (tpHL et tpLH) de 5 ns, il ne sera pas possible de fonctionner à 20 MHz. Puisque la période de fonctionnement est de 50 ns, le délai du Thermo2bin prend plus de 50 ns pour que le signal passe à travers. Pour arriver à cette conclusion, il faut examiner le passage du signal le plus long. C'est avec la gestion d'erreur et la mémoire de l'additionneur 4 bits que le traitement est le plus long.

1.2 Implémentation de schémas/code VHDL

En ce qui concerne l'architecture de Thermo2bin le schéma en bloc à la partie 4.1 de l'annexe représente sa structure. Le code VHDL est constitué de deux sous modules :
Thermo2bin_4bits : Ce module converti 4 bits thermométrique en 3 bits binaire.
add_4bits : Ce module additionne la valeur de Thermo2bin_4bits. Qui plus est, 5 modules Thermo2bin_4bits, dont 2 sont utiliser seulement pour la gestion d'erreur, permettent de convertir entièrement le code thermométrique de 12 bit(11 :0) en 4 bit(3 :0) avec l'aide des additionneurs.

2 Démarche de vérification

2.1 Résultats simulation

Pour valider l'intégration de tous les modules, les entrées (S1, S2, BTN0, BTN1 et ADCth) et les sorties (DEL2, DEL3, SSD et LED) devront être simulées. Pour ce faire, lors de la simulation, il y aura différentes sections qui testeront toutes les entrées. Le chronogramme de la Figure 1 démontre la valeur des sorties en fonction des entrées préconfigurées.

2.2 Liste de tests thermo2bin

Pour s'assurer de bien tester la fonctionnalité du système sans tester 2^{12} possibilitées, on peut séparer les tests avec les possibilités de réussite et les possibilitées d'échecs. En ce qui concerne les possibilités de réussite, 12 états thermométrique peuvent être testés pour assurer la fonctionalité. En ce qui concerne les possibilités d'échecs, pour éviter d'Avoir une énorme quantité de test, certaines méthodes ont été adoptés : thermométrique inverse, packet de 4 bits de 0 avec des 1 aux extrémités, ainsi que discontinuité dans les valeurs(0 entre des 1).

2.3 Chronogramme

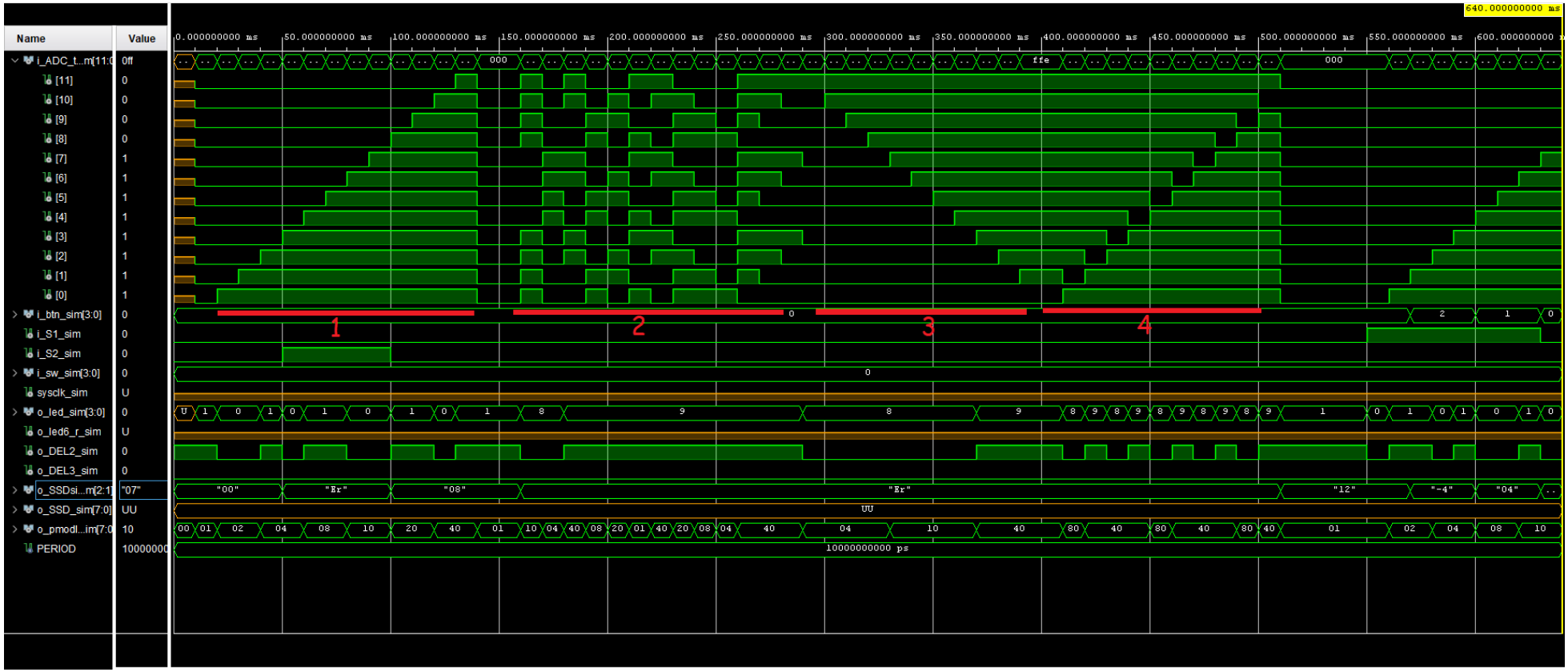


Figure 1 – Chronogramme

2.4 Commentaire réussite/échec

La section 1 du chronogramme affiche tous les états thermométriques possibles. La partie 2 test les états d'échec avec des 4 bits et des paires de bit pour tester les états non fonctionnels et savoir si le code fonctionne comme prévu. Pour la partie 3, c'est thermométrique inverse et pour la partie 4 c'Est pour tester la discontinuité dans le code thermométrique. Avec les chronogrammes on peut voir les états où on obtient une erreur.

3 Démarche d'analyse de compatibilité

Les signaux du FPGA pour la DEL D2 ne sont pas compatibles. En se basant sur les datasheets, des composantes 74V1T04STR et NC7SP04P5X. Il est possible d'extraire les valeurs hautes et les valeurs basses des tensions(VIH et VIL). Pour 74V1T04STR la minimum niveau haut à l'entrée (VIH) est de 2V et la sortie obtenue du NC7SP04P5X au niveau au est de 0.65*Vcc, sois 0.65*1.2V. Donc, il ne peut jamais obtenir le niveau haut à U6. Étant donnée la valeur de la source de U4(74V1T04STR) le résultat démontre une incompatibilité des signaux entre le FPGA et la DEL D2.

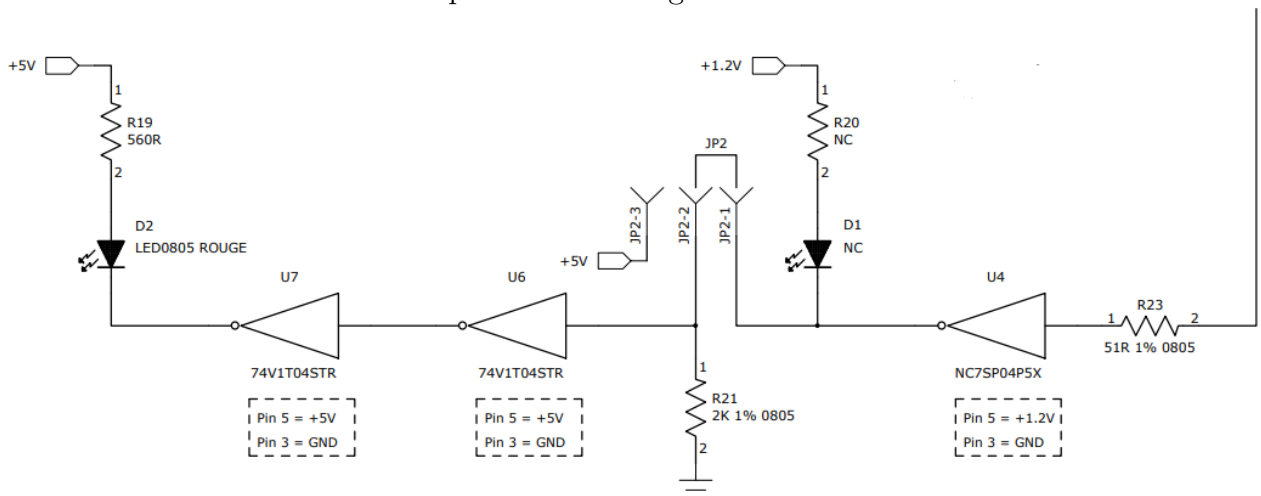
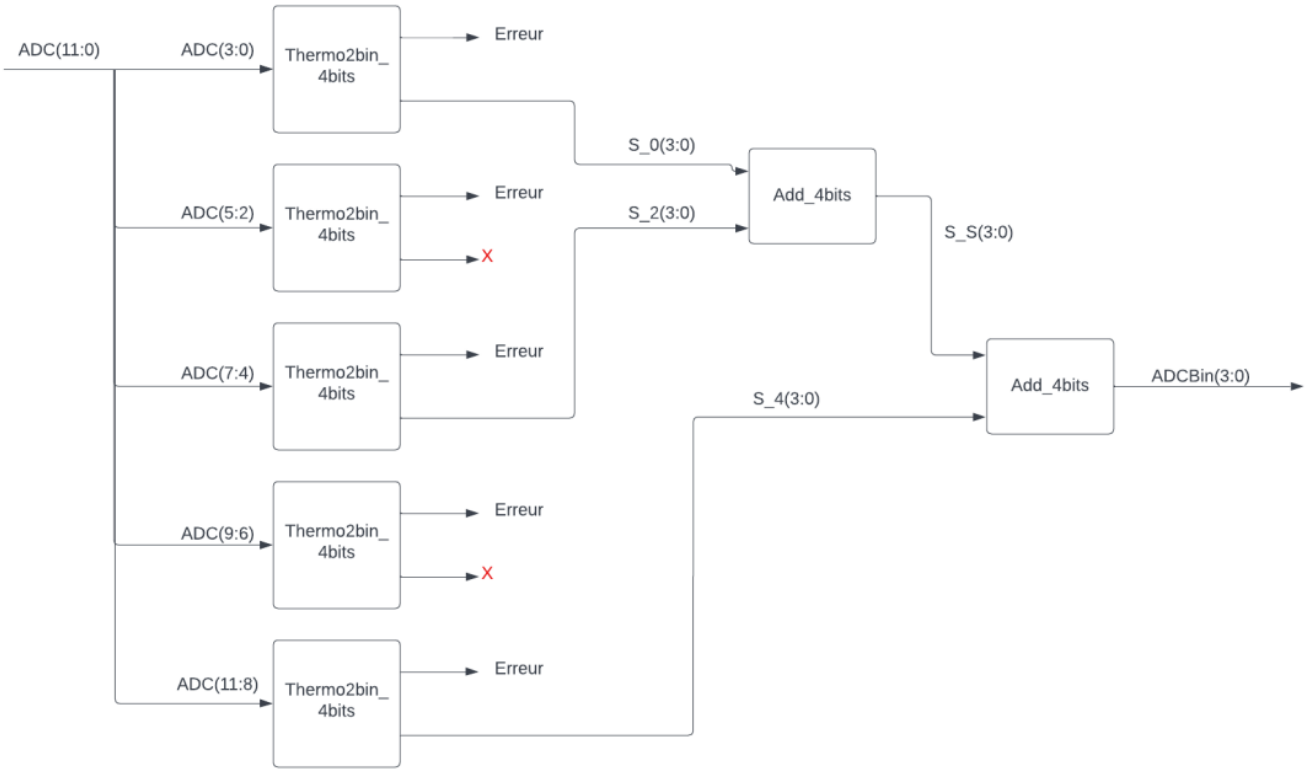


FIGURE 2 – Circuit de la LED D2

4 Annexe

4.1 Schéma en bloc de Thermo2bin

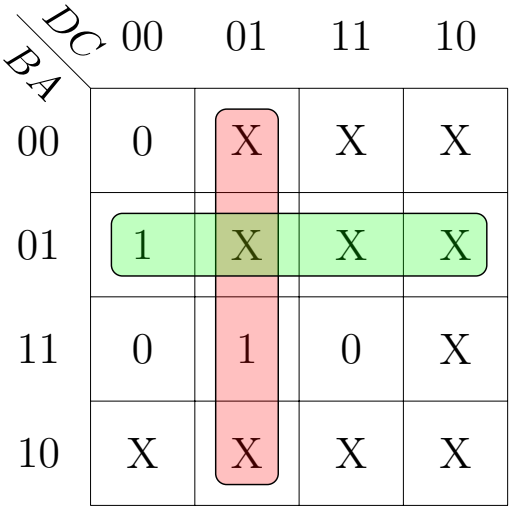


4.2 Table de vérité Thermo2bin

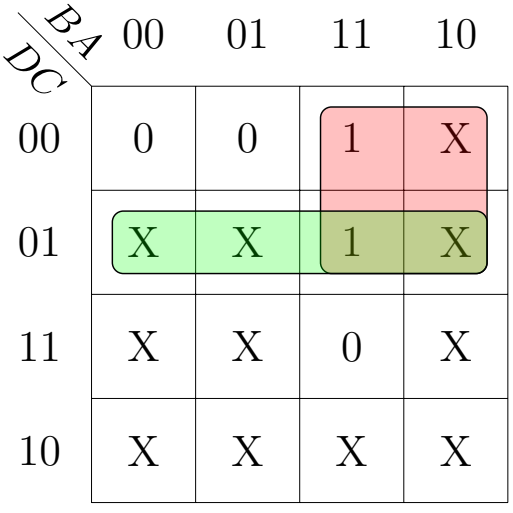
D	C	B	A	S2	S1	S0
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	0	-	-	-
0	0	1	1	0	1	0
0	1	0	0	-	-	-
0	1	0	1	-	-	-
0	1	1	0	-	-	-
0	1	1	1	0	1	1
1	0	0	0	-	-	-
1	0	0	1	-	-	-
1	0	1	0	-	-	-
1	0	1	1	-	-	-
1	1	0	0	-	-	-
1	1	0	1	-	-	-
1	1	1	0	-	-	-
1	1	1	1	1	0	0

4.3 Tables de Karnaugh Thermo2bin

$$\boxed{S_0 = \overline{D}C + \overline{B}A}$$



$$\boxed{S_1 = B\overline{D} + \overline{D}C}$$



$$\boxed{S_2 = D}$$

DC \ BA	00	01	11	10
00	0	0	0	X
01	X	X	0	X
11	X	X	1	X
10	X	X	X	X

$$\boxed{Erreur = CB' + DC' + BA' + DB' + DC'A'}$$

DC \ BA	00	01	11	10
00	0	0	0	1
01	1	1	0	1
11	1	1	0	1
10	1	1	1	1

4.4 VHDL Code Thermo2bin et sous-module

4.5 Thermo2bin

```
entity Thermo2bin is
    Port ( thermo_in : in STD_LOGIC_VECTOR (11 downto 0);
           thermo_out : out STD_LOGIC_VECTOR (3 downto 0);
           thermo_er : out STD_LOGIC);
end Thermo2bin;

architecture Behavioral of Thermo2bin is

component Add4bits is
    port ( X_0 : in STD_LOGIC;
           X_1 : in STD_LOGIC;
           X_2 : in STD_LOGIC;
           X_3 : in STD_LOGIC;
           Y_0 : in STD_LOGIC;
           Y_1 : in STD_LOGIC;
           Y_2 : in STD_LOGIC;
           Y_3 : in STD_LOGIC;
           S_0 : out STD_LOGIC;
           S_1 : out STD_LOGIC;
           S_2 : out STD_LOGIC;
           S_3 : out STD_LOGIC;
           Ci : in STD_LOGIC;
           C0 : out STD_LOGIC
         );
    end component;

component Thermo2Bin_4bits is
    Port ( Thermo_in_4bits : in STD_LOGIC_VECTOR (3 downto 0);
```

```
            Thermo_out_4bits : out STD_LOGIC_VECTOR (3 downto 0);
            Er_out_4bits : out STD_LOGIC
            );
    end component;

SIGNAL S_0  : std_logic_vector(3 downto 0);
SIGNAL S_1  : std_logic_vector(3 downto 0);
SIGNAL S_2  : std_logic_vector(3 downto 0);
SIGNAL S_3  : std_logic_vector(3 downto 0);
SIGNAL S_4  : std_logic_vector(3 downto 0);
SIGNAL S_S  : std_logic_vector(3 downto 0);

Signal er_0 : std_logic;
Signal er_1 : std_logic;
Signal er_2 : std_logic;
Signal er_3 : std_logic;
Signal er_4 : std_logic;

begin

inst_Thermo2Bin_4bits_0 : Thermo2Bin_4bits
    Port map (
    Thermo_in_4bits => thermo_in(3 downto 0),
    Thermo_out_4bits => S_0,
    Er_out_4bits => er_0
             );

inst_Thermo2Bin_4bits_1 : Thermo2Bin_4bits
    Port map (
    Thermo_in_4bits => thermo_in(5 downto 2),
    Thermo_out_4bits => S_1,
    Er_out_4bits => er_1
```

```
            );

inst_Thermo2Bin_4bits_2 : Thermo2Bin_4bits
    Port map (
    Thermo_in_4bits => thermo_in(7 downto 4),
    Thermo_out_4bits => S_2,
    Er_out_4bits => er_2
            );

inst_Thermo2Bin_4bits_3 : Thermo2Bin_4bits
    Port map (
    Thermo_in_4bits => thermo_in(9 downto 6),
    Thermo_out_4bits => S_3,
    Er_out_4bits => er_3
            );

inst_Thermo2Bin_4bits_4 : Thermo2Bin_4bits
    Port map (
    Thermo_in_4bits => thermo_in(11 downto 8),
    Thermo_out_4bits => S_4,
    Er_out_4bits => er_4
            );

inst_add1 : Add4bits
    Port map (
            X_3     => S_0(3),
            X_2     => S_0(2),
            X_1     => S_0(1),
            X_0     => S_0(0),
            Y_3     => S_2(3),
            Y_2     => S_2(2),
            Y_1     => S_2(1),
```

```
            Y_0     => S_2(0),
            Ci      => '0',
            S_0     => S_S(0),
            S_1     => S_S(1),
            S_2     => S_S(2),
            S_3     => S_S(3)
             );

inst_add2 : Add4bits
    Port map (
            X_3     => S_4(3),
            X_2     => S_4(2),
            X_1     => S_4(1),
            X_0     => S_4(0),
            Y_3     => S_S(3),
            Y_2     => S_S(2),
            Y_1     => S_S(1),
            Y_0     => S_S(0),
            Ci      => '0',
            S_0     => thermo_out(0),
            S_1     => thermo_out(1),
            S_2     => thermo_out(2),
            S_3     => thermo_out(3)
             );

thermo_er <= er_0 OR er_1 OR er_2 OR er_3 OR er_4;

end Behavioral;
```

4.6 Sous-module Thermo2bin

```
entity Thermo2Bin_4bits is
    Port ( Thermo_in_4bits : in STD_LOGIC_VECTOR (3 downto 0);
           Thermo_out_4bits : out STD_LOGIC_VECTOR (3 downto 0);
           Er_out_4bits : out STD_LOGIC);
end Thermo2Bin_4bits;

architecture Behavioral of Thermo2Bin_4bits is
Signal S_0 : STD_LOGIC;
Signal S_1 : STD_LOGIC;
Signal S_2 : STD_LOGIC;
Signal err : STD_LOGIC;
begin
Er_out_4bits <= ((Thermo_in_4bits(2) AND (NOT Thermo_in_4bits(1)))
            OR (Thermo_in_4bits(3) AND (NOT Thermo_in_4bits(2)))
            OR (Thermo_in_4bits(1) AND (NOT Thermo_in_4bits(0)))
            OR (Thermo_in_4bits(3) AND (NOT Thermo_in_4bits(1)))
            OR (Thermo_in_4bits(3) AND (NOT Thermo_in_4bits(2))
            AND (NOT Thermo_in_4bits(0))) OR
            (Thermo_in_4bits(1) AND (NOT Thermo_in_4bits(2))
            AND (NOT Thermo_in_4bits(0))));

S_0 <= (NOT Thermo_in_4bits(3) AND Thermo_in_4bits(2)) OR (NOT Thermo_in_4bits(1)
AND Thermo_in_4bits(0));
S_1 <= (NOT Thermo_in_4bits(3) AND Thermo_in_4bits(1)) OR (NOT Thermo_in_4bits(3)
AND Thermo_in_4bits(2));
S_2 <= Thermo_in_4bits(3) OR (Thermo_in_4bits(3) AND Thermo_in_4bits(1))
OR (Thermo_in_4bits(3) AND Thermo_in_4bits(0));

Thermo_out_4bits <=  '0' & S_2 & S_1 & S_0;
end Behavioral;
```

Références

[1] Réjean Fontaine, Marc-André Tétrault, Guide étudiant Logique combinatoire, Département de génie électrique et de génie informatique - Faculté de génie Université de Sherbrooke

[2] John Wakerly, 2007,Pearson ; 4th edition, Digital Design : Principles and practices