INF3500 – Hiver 2022

Exercices #3 – Modélisation et vérification de circuits combinatoires

0301/0302 Description d'un circuit combinatoire en VHDL; Concepts intermédiaires de VHDL

1. Donner le code VHDL synthétisable pour une porte NOR (non-OU) à nombre d'entrées arbitraire ≥ 3. Utiliser une description comportementale.

2. Donnez le code VHDL synthétisable d'un module combinatoire qui accepte en entrée un vecteur de W bits, W >= 8, et qui a une sortie F qui doit être à '1' quand le vecteur contient au moins quatre bits à '1' et quatre bits à '0'.

3. Tony Stark, riche entrepreneur en robotique, possède un vaste manoir situé sur une des îles de la Rivière-des-Milles-Iles. Le manoir est entouré d'une haute clôture surveillée par un réseau de caméras intelligentes, numérotées 0, 1, 2, 3, etc. Les caméras sont capables de détecter automatiquement la présence d'un intrus, jour et nuit. Quand un intrus est détecté, une caméra active un signal d'alerte sur un fil qui lui est dédié. M. Stark a besoin d'une boîte d'interface pour relier toutes ces caméras. La boîte doit activer un signal d'alarme quand au moins une caméra détecte un intrus. Elle doit aussi donner le numéro de la caméra qui détecte un intrus par un nombre binaire. M. Stark, craint par ses ennemis, s'attend à ce qu'il n'y ait jamais plus d'un intrus à la fois; mais, comme tout bon ingénieur, il veut aussi un signal d'erreur qui s'active si jamais plus d'une caméra rapportait un intrus. Donnez une architecture en VHDL synthétisable pour la boîte d'interface de M. Stark, qui correspond à la déclaration d'entité suivante

   a. Donnez un code valide uniquement pour 4 caméras.
   b. Donnez un code valide pour un nombre arbitraire de caméras.

```
library IEEE;
use IEEE.std_logic_1164.all;
use ieee.numeric_std.all;
use ieee.math_real.all;

entity museeSecurite3 is
    generic (
        N : positive := 4
    );
    port (
        alerte_Camera : in std_logic_vector(N - 1 downto 0) ;
        alarme_Intrus : out std_logic;
        erreur : out std_logic;
        code_Camera : out unsigned(integer(ceil(log2(real(N)))) - 1 downto 0)
    );
end;
```

0303 Banc d'essai pour un circuit combinatoire

4. Donnez un exemple d'utilisation possible des quatre niveaux de la clause `severity` dans un banc d'essai.

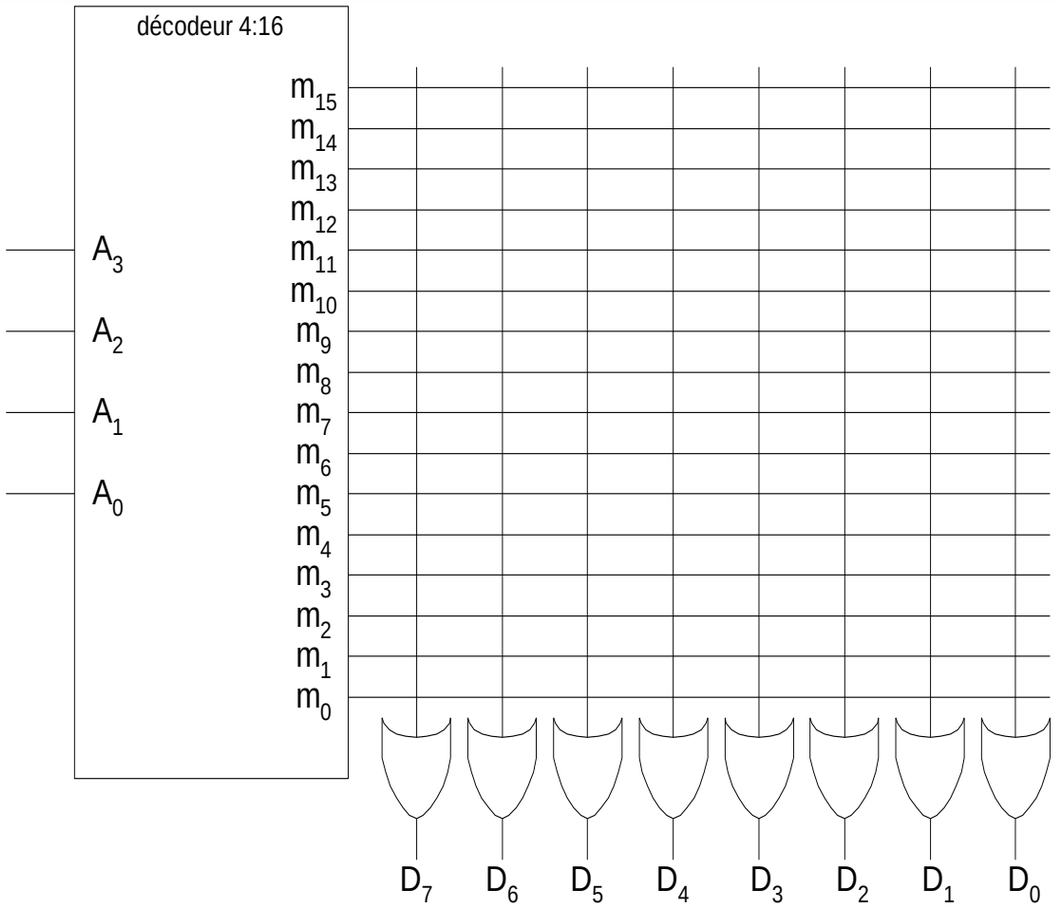
5. La fonction parité est vraie si le nombre de bits en entrée égal à '1' est pair. Par exemple, la fonction est vraie pour les vecteurs {0000, 1111, 1001} et fausse pour {0001, 0111}. Considérez la déclaration d'entité suivante en VHDL. Donnez le code VHDL d'un banc d'essai qui stimule et vérifie complètement le module pour déterminer la parité selon une entrée a. de 4 bits et b. d'un nombre arbitraire de bits.

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

entity parite is
    generic ( W : positive := 4); -- le nombre de bits d'entrée
    port (
        I : in std_logic_vector(W - 1 downto 0);
        F : out std_logic  -- '1' pour parité paire, '0' pour parité impaire
    );
end parite;
```

0304 Synthèse d'un circuit combinatoire à partir de sa description en VHDL

6. Programmez le PROM suivant pour implémenter le code VHDL qui suit. Étiquetez correctement les ports d'entrée et de sortie, et placez des boulets (•) aux intersections de lignes à relier ensemble.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity module12 is
   port (
      x : in unsigned(3 downto 0);
      triangle : out unsigned(3 downto 0);
      carre2 : out std_logic;
      pulse4 : out std_logic
   );
end module12;

architecture arch of module12 is
begin
   carre2 <= x(1);
   pulse4 <= x(1) and x(0);

   process(x) is
   begin
      if x < 5 then
         triangle <= x;
      elsif x < 9 then
         triangle <= 8 - x;
      elsif x < 13 then
         triangle <= x - 8;
      else
         triangle <= 16 - x;
      end if;
   end process;

end arch;
```



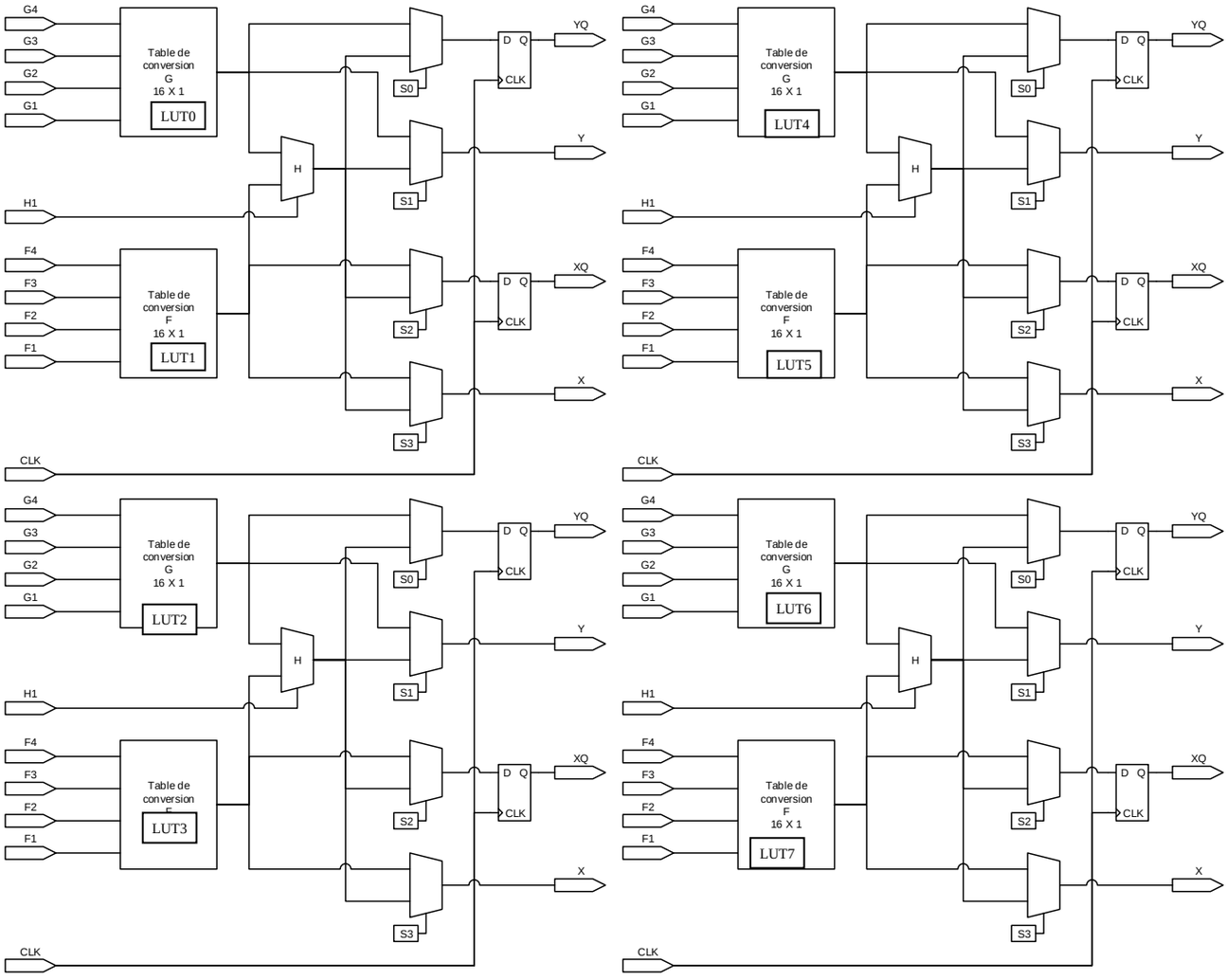
7. Montrez comment implémenter une fonction NON-OU à 4 entrées et une fonction NON-OU à 6 entrées sur un FPGA. Montrez le résultat de la synthèse sur un des diagrammes de FPGA donnés aux pages suivantes.

8. Questions de discussion.
   a. Quel genre de description de module combinatoire est la plus facile à synthétiser, une description structurale, par flot de données ou comportementale ?
   b. Laquelle des trois descriptions mène à une implémentation qui prend le moins de ressources ?

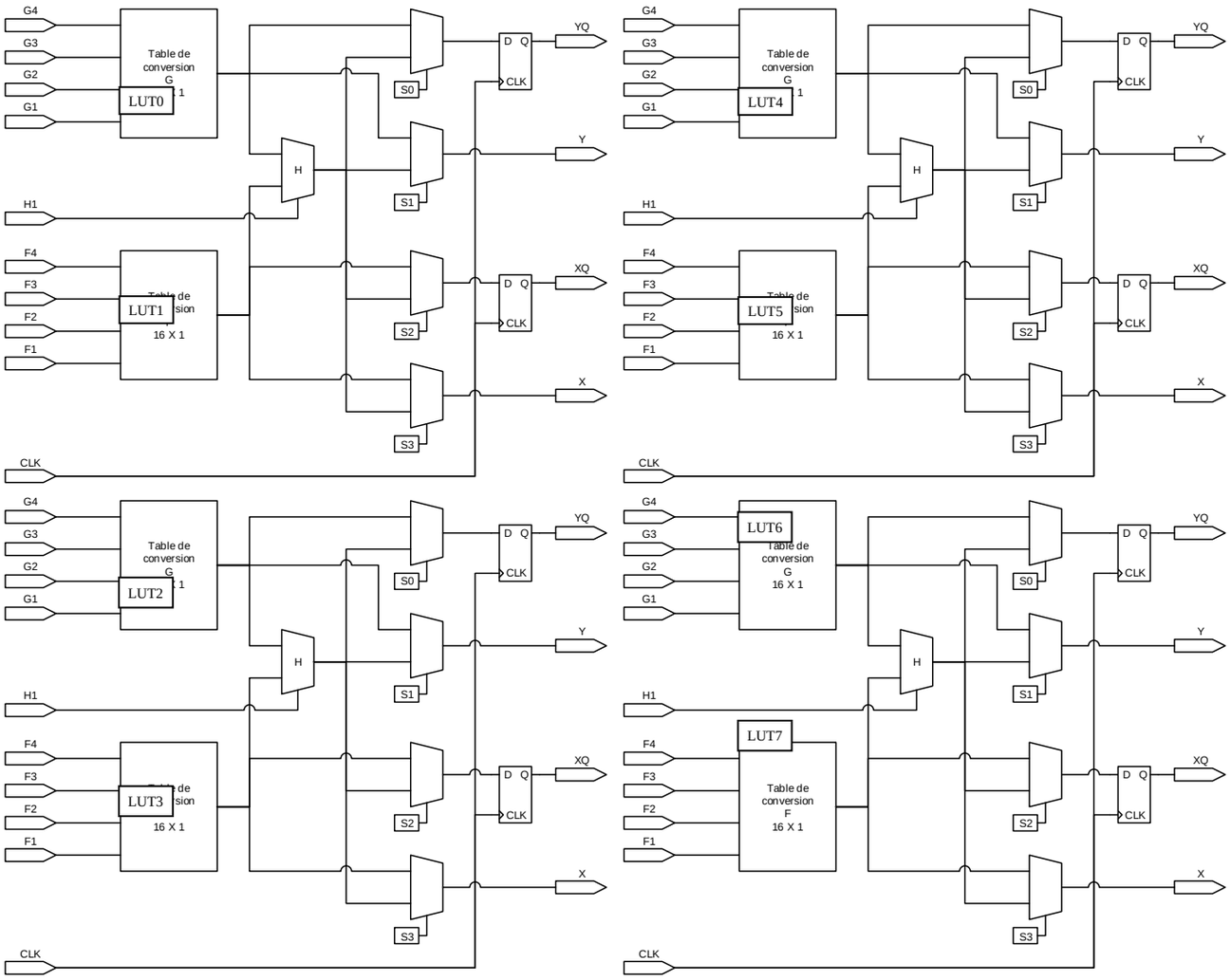
9. Considérez l'énoncé `assert`, par exemple pour vérifier les valeurs d'une constante définie par un énoncé `generic` :

```
assert W > 3 report "Il faut avoir W > 3" severity failure;
```

   a. La norme 1076.6-2004 permet-elle son utilisation dans la description d'un module synthétisable?
   b. Le synthétiseur de Vivado WebPack permet-il son utilisation dans la description d'un module synthétisable?



G4/F4	G3/F3	G2/F2	G1/F1	LUT7	LUT6	LUT5	LUT4	LUT3	LUT2	LUT1	LUT0
0	0	0	0								
0	0	0	1								
0	0	1	0								
0	0	1	1								
0	1	0	0								
0	1	0	1								
0	1	1	0								
0	1	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								



G4/F4	G3/F3	G2/F2	G1/F1	LUT7	LUT6	LUT5	LUT4	LUT3	LUT2	LUT1	LUT0
0	0	0	0								
0	0	0	1								
0	0	1	0								
0	0	1	1								
0	1	0	0								
0	1	0	1								
0	1	1	0								
0	1	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								

Solutions

1. Code :

```
entity porteNOU is
    generic (
        W : positive := 4 -- le nombre d'entrées de la porte NOU
    );
    port (
        I : in std_logic_vector(W - 1 downto 0);
        F : out std_logic
    );
end porteNOU;

architecture comportementale of porteNOU is
begin
    process (I)
    variable sortie : std_logic;
    begin
        sortie := '0';
        for k in W - 1 downto 0 loop
            sortie := sortie or I(k);
        end loop;
        F <= not(sortie);
    end process;
end comportementale;
```

2. Code :

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

entity quatre_deux_et_uns is
    generic (
        W : positive := 12 -- le nombre de bits d'entrée
    );
    port (
        I : in std_logic_vector(W - 1 downto 0);
        F : out std_logic
    );
end quatre_deux_et_uns;

architecture comportementale of quatre_deux_et_uns is
begin
     process (I)
    variable compte_0 : natural range 0 to W;
    variable compte_1 : natural range 0 to W;
    begin
        compte_0 := 0;
        compte_1 := 0;
        for k in W - 1 downto 0 loop
            if I(k) = '0' then
                compte_0 := compte_0 + 1;
            end if;
            if I(k) = '1' then
                compte_1 := compte_1 + 1;
            end if;
        end loop;

        if (compte_0 >= 4 and compte_1 >= 4) then
            F <= '1';
        else
            F <= '0';
        end if;

    end process;
end comportementale;
```

3. Code (plusieurs solutions possibles)

```
architecture arch of museeSecurite3 is
begin
    assert N = 4 report "architecture valable uniquement pour N = 4" severity failure;
    alarme_Intrus <= alerte_Camera(3) or alerte_Camera(2) or alerte_Camera(1) or
alerte_Camera(0);

    with alerte_Camera select
    code_Camera <=
    "00" when "0001",
```

```
    "01" when "0010",
    "10" when "0100",
    "11" when "1000",
    "00" when others;

    erreur <= (alerte_Camera(3) and alerte_Camera(2))
           or (alerte_Camera(3) and alerte_Camera(1))
           or (alerte_Camera(3) and alerte_Camera(0))
           or (alerte_Camera(2) and alerte_Camera(1))
           or (alerte_Camera(2) and alerte_Camera(0))
           or (alerte_Camera(1) and alerte_Camera(0));

end arch;

architecture archflexible of museeSecurite3 is
begin
    process(alerte_Camera)
    variable compte : natural range 0 to N;
    begin
        compte := 0; -- les variables sont persistentes,
                  --  il faut les réinitialiser à chaque itération
        alarme_Intrus <= '0'; -- il faut une valeur par défaut
        for k in N - 1 downto 0 loop
            if alerte_Camera(k) = '1' then
                code_Camera <= to_unsigned(k, code_Camera'length);
                alarme_Intrus <= '1';
                compte := compte + 1;
            end if;
        end loop;
        if compte >= 2 then
            erreur <= '1';
        else
            erreur <= '0';
        end if;
    end process;

end archflexible;
```

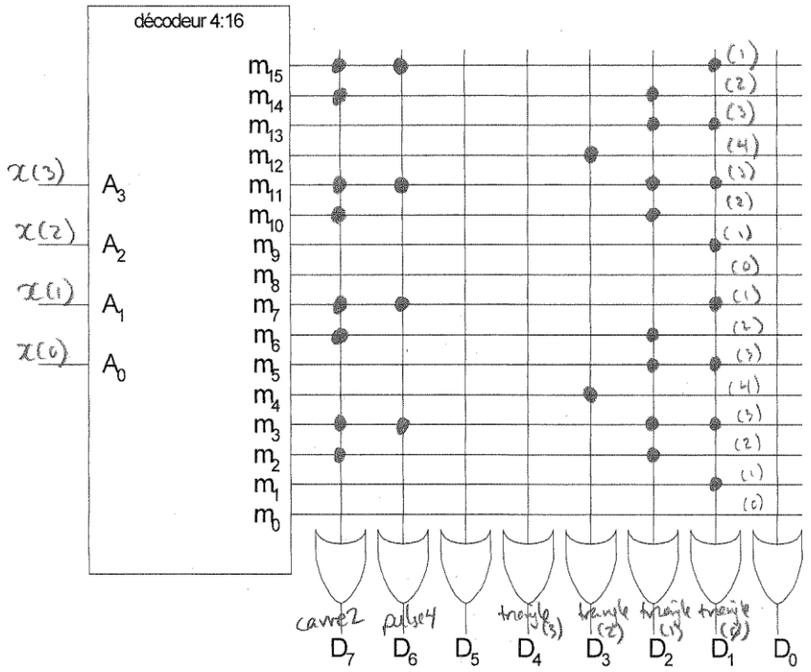
4. À discuter en classe, Note, Warning, Error, Failure

5. Solution

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use work.all;

entity parite_TB is
    generic (W : positive := 5); -- le nombre de bits d'entrée de l'UUT
end parite_TB;

architecture arch1 of parite_TB is

    signal vecteurDeTest : std_logic_vector(W - 1 downto 0);
    signal reponse : std_logic;

    -- fonction de vérification de parité
    function paritePaire(entree: std_logic_vector(W - 1 downto 0)) return boolean is
    variable somme : natural := 0;
    begin
        for k in 0 to W - 1 loop
            if entree(k) = '1' then
                somme := somme + 1;
            end if;
        end loop;
        -- la parité est paire ssi
        -- la somme des '1' dans le vecteur en entrée est un nombre pair
        return (somme mod 2 = 0);
    end paritePaire;

begin

    -- instanciation du module à vérifier
    UUT : entity parite(arch1) generic map (W) port map (vecteurDeTest, reponse);

    -- application exhaustive des vecteurs de test et vérification des réponses
    process
```

```
    begin
        for k in 0 to 2 ** W - 1 loop
            vecteurDeTest <= std_logic_vector(to_unsigned(k, W));
--          report integer'image(k);
            wait for 10 ns;
            assert (paritePaire(vecteurDeTest) = (reponse = '0'))
                report "erreur pour l'entrée " & integer'image(k) severity error;
        end loop;
        report "simulation terminée" severity failure;
    end process;

end arch1;
```

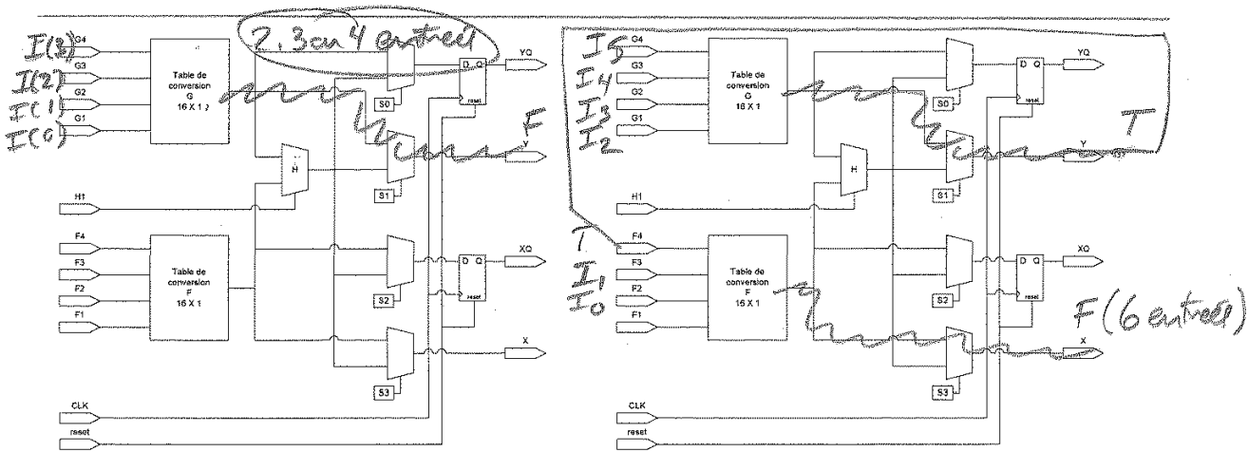
6. Solution



7. Solution

Pour la fonction NON-OU à 2, 3 ou 4 entrées, une seule LUT suffit. Les entrées sont appliquées à la LUT qui calcule la fonction logique. Par exemple, pour 3 entrées, on utiliserait seulement I(2), I(1) et I(0). On fixerait l'entrée G4 à 0. Le contenu de la LUT (16 bits) serait, dans l'ordre (du minterme le moins significatif au plus significatif), {1,0,0,0,0,0,0,0,-,-,-,-,-,-,-,-}.

Pour la fonction NON-OU à plus de 4 entrées, il faut utiliser plus d'une LUT. On montre ici le cas avec 6 entrées. La première LUT réalise la fonction OU des entrées I5, I4, I3, et I2 (dans le signal T). Ce signal est routé vers une deuxième LUT qui a aussi pour entrée les signaux I1 et I0, et la fonction NON-OU de ces trois signaux est réalisée comme expliqué au paragraphe précédent.



8. À discuter en classe.

9. À vérifier et discuter en classe.