INF3500 : Conception et réalisation de systèmes numériques

Examen intra

Lundi 25 octobre 2010

Durée: 2h.
Pondération: 20%.
Documentation: Une feuille recto verso 8.5”×11” ou A4 permise.
Calculatrice: Programmable permise.
Directives particulières:

- Ordinateurs interdits.
- Répondre à toutes les questions, la valeur de chaque question est indiquée.
- Répondre dans le cahier fourni et soumettre les pages du questionnaire indiquées.
- Ne pas remettre le questionnaire.
- Ne posez pas de questions. En cas de doute sur le sens d’une question, énoncez clairement toute supposition que vous faites.

Question 1. (4 points)

Donnez une architecture pour les entités suivantes en VHDL synthétisable, en respectant les spécifications données en commentaires.

a. (2 points)

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

-- Cette bascule JK a le comportement suivant.
-- Les valeurs des signaux de contrôle J et K déterminent l'état de la bascule
-- lors d'une transition positive du signal d'horloge:
--  si JK == 00, pas de changement
--  si JK == 01, Q passe à 0
--  si JK == 10, Q passe à 1
--  si JK == 11, Q est inversé
entity basculeJK is
port (
   clk, J, K : in std_logic;
   Q : out std_logic
);
end basculeJK;
```

Solution

```
architecture arch of basculeJK is
begin
   process(CLK, reset)
   variable JK : std_logic_vector(1 downto 0);
   variable Qint : std_logic := 'X';
   begin
      JK := (J, K);
      if rising_edge(CLK) then
         case JK is
            when "01" => Qint := '0';
            when "10" => Qint := '1';
            when "11" => Qint := not(Qint);
            when others => Qint := Qint;
         end case;
      end if;
      Q <= Qint;
   end process;
end arch;
```

b. (2 points)

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

-- Ce compteur a le comportement suivant.
-- Le compte s'effectue sur une transition négative du signal d'horloge.
-- Le compte est activé quand go == '1'.
-- Le signal UD détermine si le compte est vers le haut ('1') ou vers le bas ('0').
-- Le compte est remis à zéro de façon asynchrone quand reset = '1';
entity compteurUD is
generic (W : integer := 8);
port (
   clk, reset, go, UD : in std_logic;
   compte : out signed(W - 1 downto 0)
);
end compteurUD;
```

Solution

```
architecture arch of compteurUD is
begin
   process(CLK)
   variable compte_int : signed(W - 1 downto 0);
   begin
      if reset = '1' then
         compte_int := (others => '0');
      else
         if falling_edge(CLK) then
            if go = '1' then
               if UD = '1' then
                  compte_int := compte_int + 1;
               else
                  compte_int := compte_int - 1;
               end if;
            end if;
         end if;
      end if;
      compte <= compte_int;
   end process;
end arch;
```

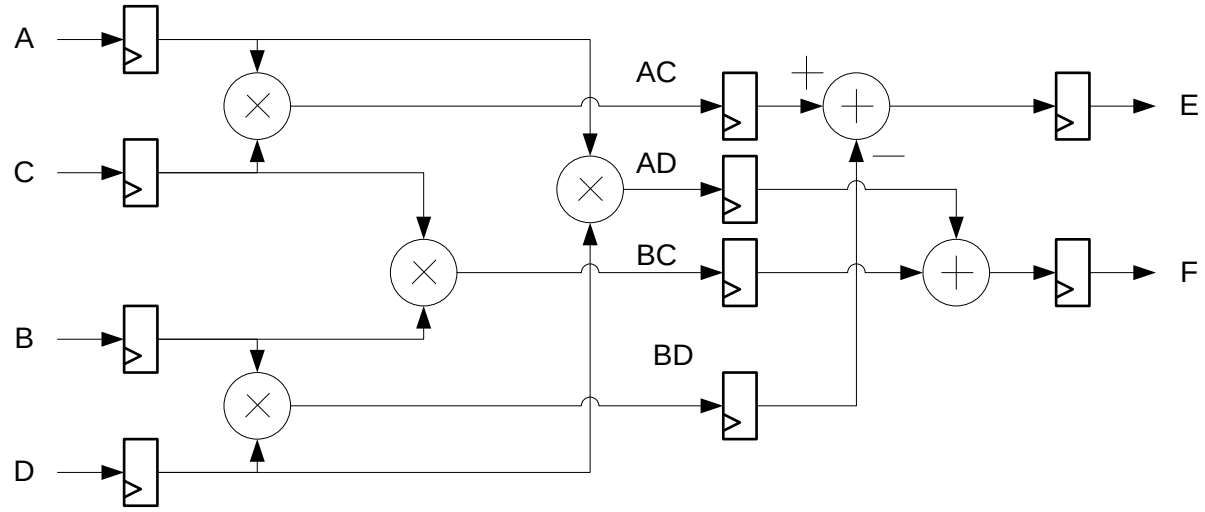
Question 2. (4 points)

Considérez le problème de la conception d'un multiplicateur complexe devant être implémenté sur FPGA.

Les entrées du circuit sont A, B, C, D, représentant les nombres Z1 = A + jB et Z2 = C + jD. Les sorties du circuit sont E et F représentant Z3 = E + jF, avec Z3 = Z1 × Z2.

Le diagramme suivant décrit un multiplicateur complexe. Les entrées sont des valeurs entières signées exprimées avec 16 bits. Les sorties des multiplicateurs sont signées et exprimées sur 32 bits, et les sorties des additionneurs sont signées et exprimées sur 33 bits. Tous les registres partagent une même horloge `CLK` et un signal de réinitialisation synchrone `reset`.

Donnez le code VHDL synthétisable correspondant à ce diagramme.



Solution

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

entity multiplicateurcomplexe is
    generic (
        W : positive := 16 -- nombre de bits des opérandes d'entrée
    );
    port (
        clk, reset : in std_logic;
        A, B, C, D : in signed(W - 1 downto 0);
        E, F : out signed(2 * W downto 0)
    );
end multiplicateurcomplexe;

architecture arch of multiplicateurcomplexe is
signal At, Bt, Ct, Dt : signed(W - 1 downto 0);
signal AC, AD, BC, BD : signed(2 * W - 1 downto 0);
begin

    process(clk, reset)
    begin
        if rising_edge(clk) then
            if reset = '1' then
                At <= (others => '0'); Bt <= (others => '0');
                Ct <= (others => '0'); Dt <= (others => '0');
                AC <= (others => '0'); BC <= (others => '0');
                AD <= (others => '0'); BD <= (others => '0');
                E <= (others => '0'); F <= (others => '0');
            else
                At <= A; Bt <= B; Ct <= C; Dt <= D;
                AC <= At * Ct; AD <= At * Dt;
                BC <= Bt * Ct; BD <= Bt * Dt;
                E <= resize(AC, E'length) - resize(BD, E'length);
                F <= resize(AD, F'length) + resize(BC, F'length);
            end if;
        end if;
    end process;

end arch;
```

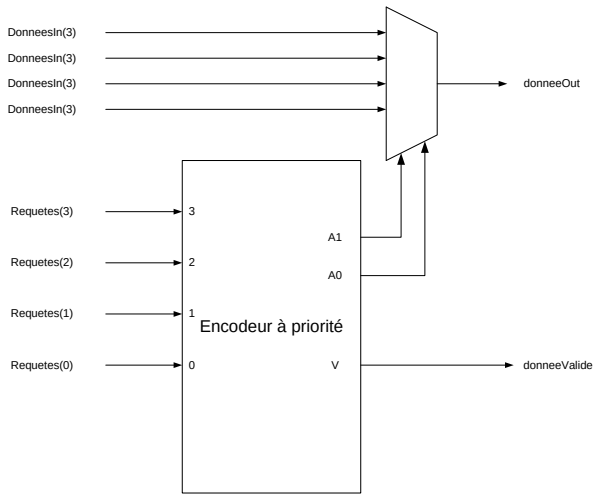
Question 3. (2 points)

Considérez le code VHDL suivant pour un arbitre de bus rudimentaire.

Donnez un schéma du circuit correspondant à ce code.

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

entity arbitreBus is
    generic (n : positive := 2);
    port(
        requetes : in std_logic_vector(2 ** n - 1 downto 0);
        donneesIn : in std_logic_vector(2 ** n - 1 downto 0);
        donneeOut, donneeValide : out std_logic
    );
end arbitreBus;

architecture arch of arbitreBus is
signal choix : unsigned(n - 1 downto 0);
begin

    process(requetes, donneesIn)
    begin
        donneeValide <= '0';
        for k in 2 ** n - 1 downto 0 loop
            if requetes(k) = '1' then
                choix <= to_unsigned(k, n);
                donneeValide <= '1';
                exit;
            end if;
        end loop;
    end process;

    process(choix, donneesIn)
    begin
        donneeOut <= donneesIn(to_integer(choix));
    end process;

end arch;
```

Solution

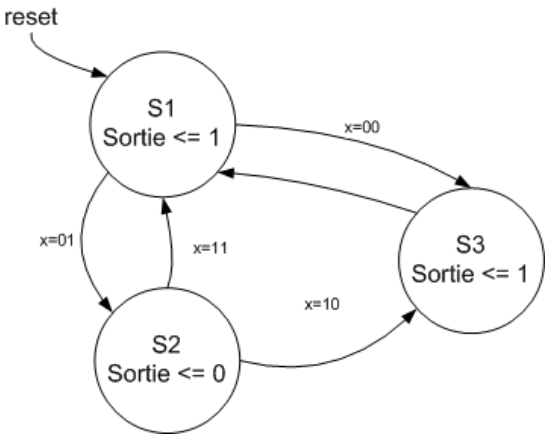


Question 4. (2 points)

Considérez le code VHDL suivant. Donnez un diagramme d'états correspondant.

```
library IEEE;
use IEEE.std_logic_1164.all;

entity machineetats is
    port (
        reset, CLK : in STD_LOGIC;
        x : in STD_LOGIC_VECTOR(1 downto 0);
        sortie : out STD_LOGIC
    );
end machineetats;

architecture arch of machineetats is
type type_etat is (S1, S2, S3);
signal etat : type_etat := S1;
begin

    process(CLK, reset) is
    begin
        if (reset = '0') then
           etat <= S1;
        elsif (rising_edge(CLK)) then
           case etat is
              when S1 =>
                 if x = "00" then
                    etat <= S3;
                 elsif x = "01" then
                    etat <= S2;
                 end if;
              when S2 =>
                 if x = "10" then
                    etat <= S3;
                 elsif x = "11" then
                    etat <= S1;
                 end if;
              when S3 =>
                 etat <= S1;
           end case;
        end if;
    end process;

    process(x, etat)
    begin
        case etat is
           when S1 | S3 =>
              sortie <= '1';
           when S2 =>
              sortie <= '0';
        end case;
    end process;

end arch;
```

Solution



Question 5. (4 points)

Réponses brèves.

a. (0.5 point) Placez les technologies de logique programmable suivantes en ordre chronologique croissant d'apparition sur le marché : CPLD, EPROM, FPGA, GAL, LSI, MSI, PLA/PAL, SSI

b. (0.5 point) Donnez une différence architecturale importante entre un CPLD et un GAL.

c. (0.5 point) Donnez une différence architecturale importante entre un CPLD et un FPGA.

d. (0.5 point) Nommez deux technologies de programmation utilisées dans des FPGAs présentement sur le marché.

e. (1 point) Énumérez quatre considérations d'implémentation qui devraient être prises en compte lors de la conception d'un système numérique.

f. (1 point) Donnez un diagramme simplifié du flot de conception que vous avez suivi dans les laboratoires du cours à ce jour.

Solution

a. SSI, MSI, LSI, EPROM, PLA/PAL, GAL, CPLD, FPGA

b. un CPLD incorpore effectivement plusieurs PAL/PLA ou GALs sur une seule puce; ajout d'un réseau d'interconnections dans un CPLD

c. blocs fonctionnels de grande taille vs CLBs réalisant des fonctions simples; un réseau d'interconnexions central vs connexions distribuées

d. SRAM, anti-fusibles, mémoire Flash

e. puissance consommée, précision des calculs, taille du circuit, taux de traitement

f. fig. 4.1 des notes de cours simplifiée : code VHDL et schémas -> vérification par simulation -> synthèse -> implémentation -> génération de fichier de configuration et programmation du FPGA

Question 6. (4 points)

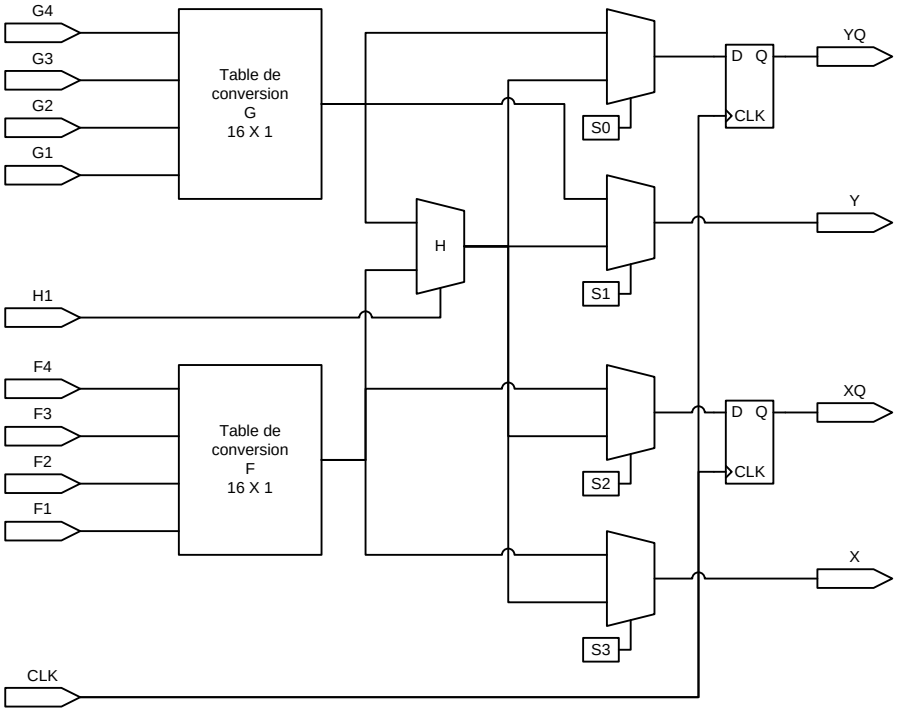
Considérez le code VHDL suivant. Montrez, sur le diagramme fourni, un résultat possible de la synthèse et de l'implémentation de ce code sur un FPGA Virtex II Pro. Indiquez directement sur le dessin où chaque signal se situe ainsi que les interconnections entre les blocs. Indiquez dans les tables de vérité fournies le contenu de chacune des tables de conversion que vous utilisez.

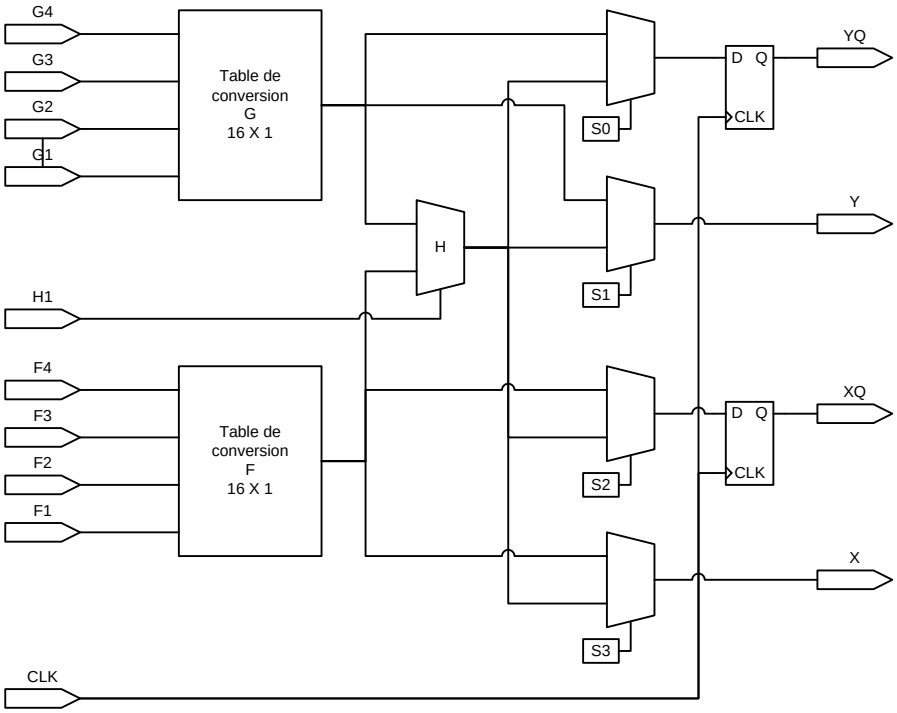
Écrivez votre nom et matricule et remettez les deux feuilles dans votre cahier d'examen.

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity compteurJohnson is
   generic (
      W : positive := 3
   );
   port (
      reset, clk: in std_logic;
      compte : out std_logic_vector(W - 1 downto 0);
      alerte : out std_logic
   );
end compteurJohnson;

architecture arch1 of compteurJohnson is
begin

   process(clk, reset)
   variable c : std_logic_vector(W - 1 downto 0);
   begin

      if (rising_edge(clk)) then
         if reset = '1' then
            c := (others => '0');
         else
            c := c(W - 2 downto 0) & not(c(W - 1));
         end if;
      end if;

      if c = "111" then
         alerte <= '1';
      else
         alerte <= '0';
      end if;

      compte <= c;

   end process;

end arch1;
```

Question 6 - Annotez cette page et remettez-la avec votre cahier d'examen

Nom : _____ Matricule : _____





Question 6 - Annotez cette page et remettez-la avec votre cahier d'examen

Nom : _____ Matricule : _____

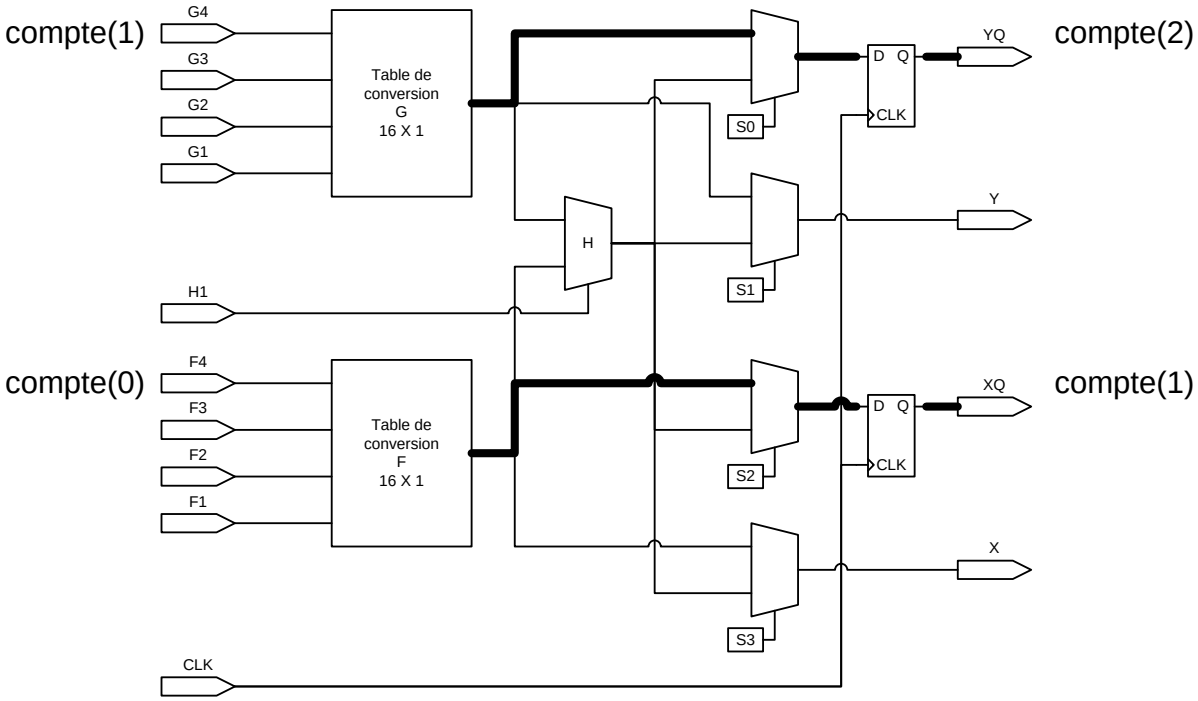
G4 ( )	G3 ( )	G2 ( )	G1 ( )	G ( )
0	0	0	0	
0	0	0	1	
0	0	1	0	
0	0	1	1	
0	1	0	0	
0	1	0	1	
0	1	1	0	
0	1	1	1	
1	0	0	0	
1	0	0	1	
1	0	1	0	
1	0	1	1	
1	1	0	0	
1	1	0	1	
1	1	1	0	
1	1	1	1	

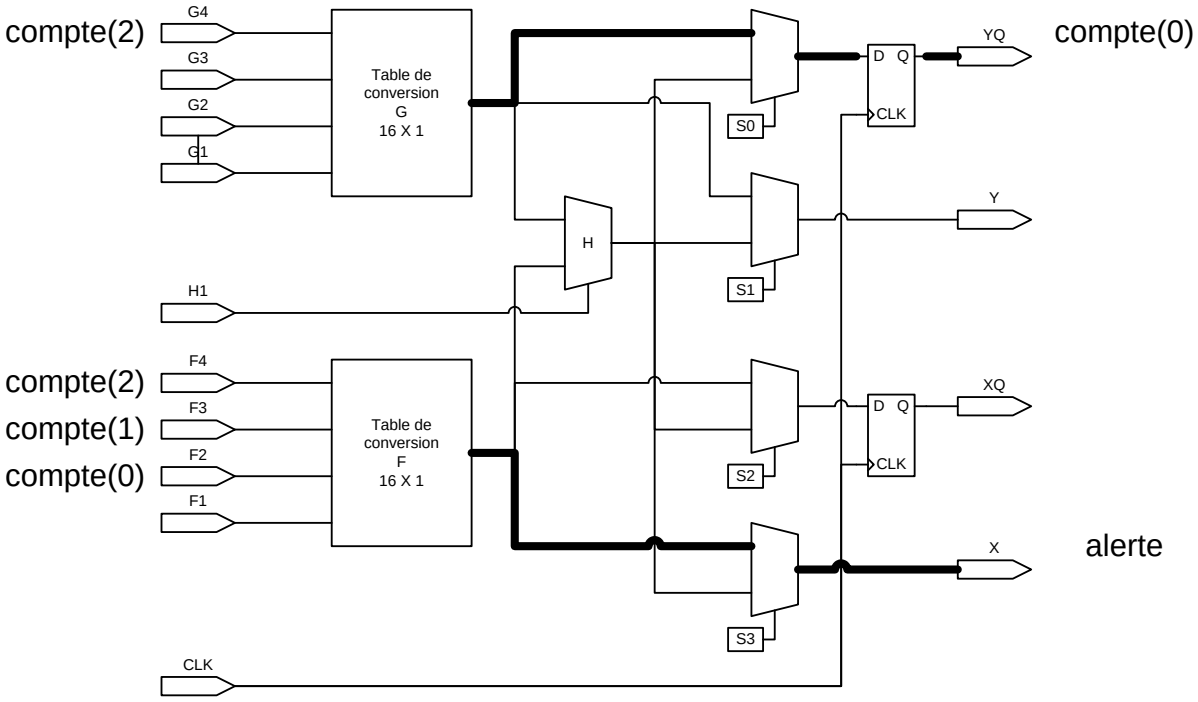
F4 ( )	F3 ( )	F2 ( )	F1 ( )	F ( )
0	0	0	0	
0	0	0	1	
0	0	1	0	
0	0	1	1	
0	1	0	0	
0	1	0	1	
0	1	1	0	
0	1	1	1	
1	0	0	0	
1	0	0	1	
1	0	1	0	
1	0	1	1	
1	1	0	0	
1	1	0	1	
1	1	1	0	
1	1	1	1	

G4 ( )	G3 ( )	G2 ( )	G1 ( )	G ( )
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	

F4 ( )	F3 ( )	F2 ( )	F1 ( )	F ( )
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	

Solution


compte(1)
G4
G3
G2
G1
Table de conversion G 16 X 1
H
H1
compte(0)
F4
F3
F2
F1
Table de conversion F 16 X 1
CLK
S0
S1
S2
S3
D Q
CLK
YQ
Y
XQ
X
compte(2)
compte(1)


compte(2)
G4
G3
G2
G1
Table de conversion G 16 X 1
H
H1
compte(2)
compte(1)
compte(0)
F4
F3
F2
F1
Table de conversion F 16 X 1
CLK
S0
S1
S2
S3
D Q
CLK
YQ
Y
XQ
X
compte(0)
alerte

Solution

G4 ( )	G3 ( )	G2 ( )	G1 ( )	G ( )
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

F4 ( )	F3 ( )	F2 ( )	F1 ( )	F ( )
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

G4 ( )	G3 ( )	G2 ( )	G1 ( )	G ( )
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

F4 ( )	F3 ( )	F2 ( )	F1 ( )	F ( )
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1