INF3500 : Conception et réalisation de systèmes numériques

Examen intra

Vendredi 19 février 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 la page du questionnaire indiquée.
- 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.

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

-- La sortie de ce compteur est remise à zéro de façon synchrone quand reset = '1'.
-- Le compteur est incrémenté par la valeur 'delta' à chaque coup d'horloge.
entity compteurSpecial is
    generic (
    W : positive := 5
    );
    port (
    reset, clk: in std_logic;
    delta : in unsigned(W - 1 downto 0);
    compte : out unsigned(W - 1 downto 0)
    );
end compteurSpecial;
```

Une solution possible :

```
architecture arch of compteurSpecial is
begin

    process(clk, reset)
    variable c : unsigned(W - 1 downto 0);
    begin
        if (rising_edge(clk)) then
            if reset = '1' then
                c := (others => '0');
            else
                c := c + delta;
            end if;
        end if;
        compte <= c;
    end process;

end arch;
```

b.

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

-- Ce registre est remis à zéro de façon asynchrone quand reset = '1'.
-- La valeur du port D est chargée sur une transition positive d'horloge
-- quand charge = '1'. Si charge = '0' et que div2 = '1', alors le contenu
-- du registre est décalé d'une position vers la droite sur une transition
-- positive d'horloge.

entity registre is
    generic (
    W : integer := 8 -- nombre de bits du registre
    );
    port (
    reset, clk, charge, div2 : in std_logic;
    D : in unsigned(W - 1 downto 0);
    Q : out unsigned(W - 1 downto 0)
    );
end registre;
```

Une solution possible :

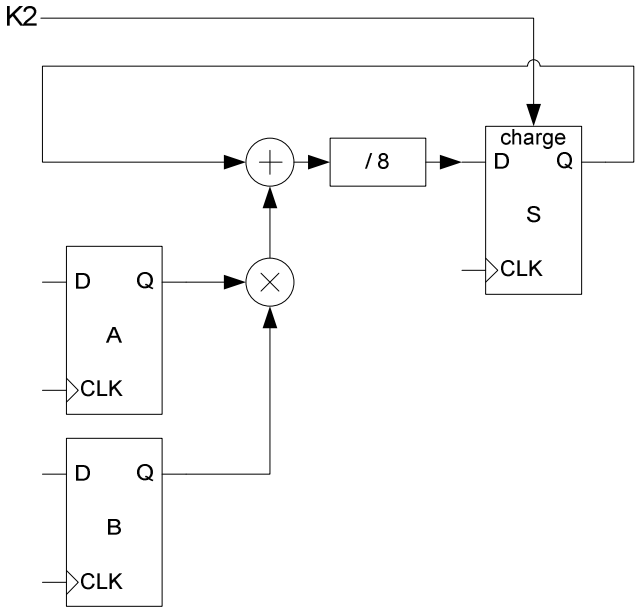
```
architecture arch of registre is
begin
    process (CLK, reset)
    variable Qt : unsigned(W - 1 downto 0);
    begin
        if reset = '1' then
            Qt := (others => '0');
         elsif rising_edge(clk) then
             if charge = '1' then
               Qt := D;
            elsif div2 = '1' then
               Qt := '0' & Qt(W - 1 downto 1);
            end if;
        end if;
        Q <= Qt;
    end process;
end arch;
```

Question 2. (2 points)

Donnez un diagramme d'un chemin des données implémentant les micro-opérations suivantes.

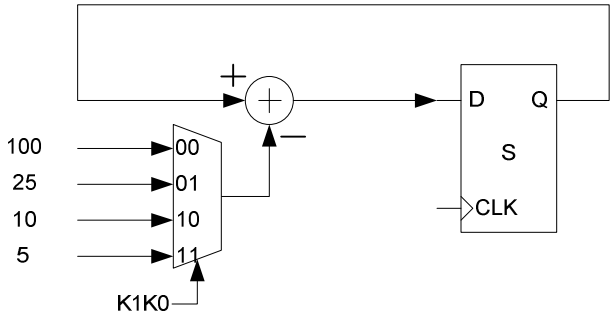
a. K2: S ← (S + A * B) / 8;

Une solution possible :



b. K1'K0': S ← S – 100; K1'K0: S ← S – 25; K1K0': S ← S – 10; K1K0: S ← S – 5;

Une solution possible :



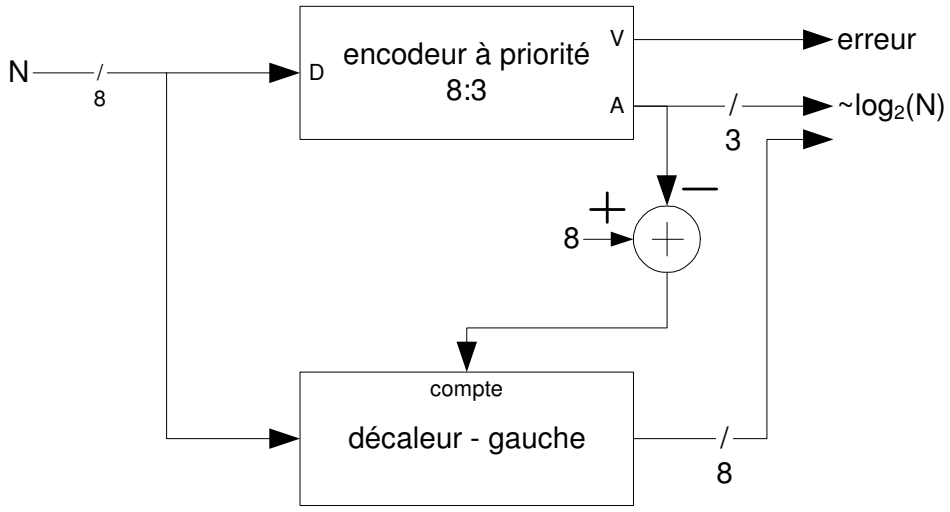
Question 3. (3 points)

Considérez le code VHDL suivant. Le circuit accepte en entrée un nombre non signé encodé sur 8 bits. Il donne une approximation du logarithme à base 2 du nombre. Si le nombre est égal à 0, un signal spécial d'erreur est activé.

Donnez un diagramme correspondant composé de modules combinatoires (multiplexeurs, décodeurs, etc.) et d'unités fonctionnelles (additionneurs, multiplicateurs, comparateurs, etc.) tel qu'il pourrait être généré par un processus de synthèse.

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

entity log2approx is
    port (
        N : in unsigned(7 downto 0);
        log2N : out unsigned(10 downto 0);
        erreur : out std_logic
    );
end log2approx;

architecture arch of log2approx is
signal partieentiere : integer range 0 to 7 := 0;
signal partiefract : unsigned(7 downto 0) := (others => '0');
begin
    process(N)
    begin
        erreur <= '1';
        partieentiere <= 0;
        for k in 7 downto 0 loop
            if N(k) = '1' then
                partieentiere <= k;
                erreur <= '0';
                exit;
            end if;
        end loop;
    end process;
    partiefract <= shift_left(N, 8 - partieentiere);
    log2N <= to_unsigned(partieentiere, 3) & partiefract;
end arch;
```

Solution :



Question 4. (4 points)

Pour l'entité `log2approx` décrite à la question précédente, composez un banc d'essai qui effectue une stimulation exhaustive, puis qui rapporte la différence maximale et la différence moyenne entre l'approximation générée par le module et la valeur exacte du logarithme. Le banc de test doit aussi vérifier si le bit d'erreur fonctionne correctement.

Rappel : Le package `math_real` inclut une fonction `log2()` qui prend en paramètre un réel et qui retourne son logarithme en base 2 sous la forme d'un réel.

Complétez le code VHDL suivant en donnant l'architecture.

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

entity log2approx_tb is
end log2approx_tb;

architecture TB_ARCHITECTURE of log2approx_tb is
--- votre code ici

begin
--- votre code ici

end TB_ARCHITECTURE;
```

Une solution possible:

```
architecture TB_ARCHITECTURE of log2approx_tb is

signal N : UNSIGNED(7 downto 0);
signal log2N : UNSIGNED(10 downto 0);
signal erreur : STD_LOGIC;

begin

UUT : entity log2approx(arch) port map (N, log2N, erreur);

process
variable log2N_vrai, difference, diffmax, sommediff : real;
begin
   diffmax := 0.0;
   sommediff := 0.0;
   for k in 0 to 255 loop
      N <= to_unsigned(k, N'length);
      wait for 10 ns;

      if k = 0 then
         assert erreur = '1' report "bit d'erreur incorrect pour N = 0!"
         severity failure;
      else
         assert erreur = '0' report "bit d'erreur incorrect pour N > 0!"
         severity failure;

         log2N_vrai := log2(real(to_integer(N)));
         difference := abs(real(to_integer(log2N)) / (2.0 ** 8) - log2N_vrai);
         if difference > diffmax then
            diffmax := difference;
         end if;
         sommediff := sommediff + difference;
```

```
            report "N: " & integer'image(to_integer(N)) &
            ", log2N: " & real'image(real(to_integer(log2N)) / (2.0 ** 8)) &
            ", log2N_vrai: " & real'image(log2N_vrai) &
            ", difference: " & real'image(difference);

         end if;
      end loop;
      report "différence maximale: " & real'image(diffmax);
      report "différence moyenne: " & real'image(sommediff / 255.0);
      report "simulation terminée" severity failure;
   end process;

   end TB_ARCHITECTURE;
```

Question 5. (3 points)

Réponses brèves.

a. Lequel des énoncés suivants est vrai concernant des entités et architectures en VHDL?

i. un module peu avoir plusieurs déclarations d'entité mais une seule architecture;

ii. un module peut avoir seulement une déclaration d'entité mais plusieurs architectures;

iii. un module peut avoir seulement une déclaration d'entité et une seule architecture;

iv. un module peut avoir plusieurs déclarations d'entité et plusieurs architectures;

Solution : ii. Une entité, plusieurs architectures

b. Il y a quatre catégories d'objets en VHDL, dont `signal` et `file`. Quelles sont les deux autres?

Solution : `variable` et `constant`

c. L'architecture d'un certain modèle VHDL contient exclusivement des assignations de signaux concurrentes, choisies et conditionnelles. Avec quel style de description de circuit ce modèle est-il décrit?

Solution : flot de données

d. Lequel/lesquels des énoncés suivants est/sont vrai(s) concernant VHDL?

i. Les processus d'une architecture s'exécutent de façon concurrente.

ii. À l'intérieur d'un processus, les assignations à des signaux sont faites de façon concurrente.

iii. À l'intérieur d'un processus, les assignations à des variables sont faites de façon séquentielle.

iv. Dans une boucle ne comportant pas d'énoncé `wait`, chaque itération prend effectivement un temps nul.

v. Dans une architecture, une assignation concurrente à un signal est une façon compacte d'écrire un processus.

Solution : i, iii, iv, v.

e. Énumérez quatre (4) éléments qu'on peut retrouver sur un FPGA présentement en vente sur le marché, à part des blocs de logique programmable (CLB) et de leur contenu.

Solution : blocs d'entrées-sorties (IOB), réseau d'interconnexions, circuits de routage rapide des retenues, blocs de mémoire intégrée, blocs de fonctions arithmétiques avancées (multiplicateurs, multiplicateurs-accumulateurs), microprocesseurs fixes, circuits de génération et de distribution d'horloge.

f. Énumérez deux des composantes fondamentales qu'on peut retrouver à l'intérieur d'un circuit logique programmable complexe (CPLD) présentement en vente sur le marché.

Solution : blocs fonctionnels (effectivement des PALs), macrocellules (contenant un élément à mémoire programmable), cellules d'entrées-sorties et réseau d'interconnexions

Question 6. (4 points)

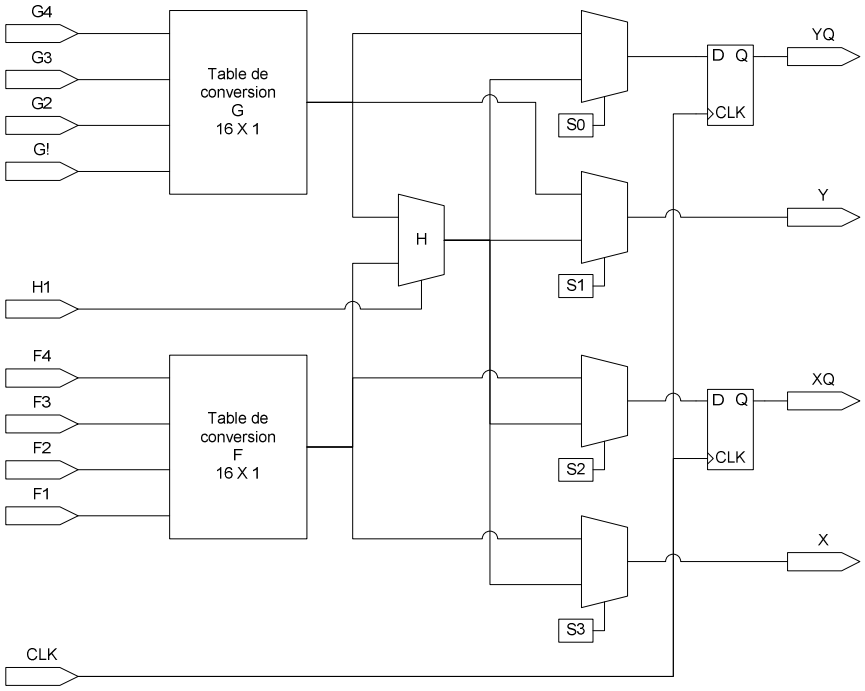
Considérez le code VHDL suivant. Montrez, sur le diagramme suivant d'une partie d'un FPGA, un résultat possible de la synthèse et de l'implémentation de ce code. Indiquez directement sur le dessin où chaque signal se situe ainsi que les interconnections entre les blocs. Indiquez dans les tables de vérité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

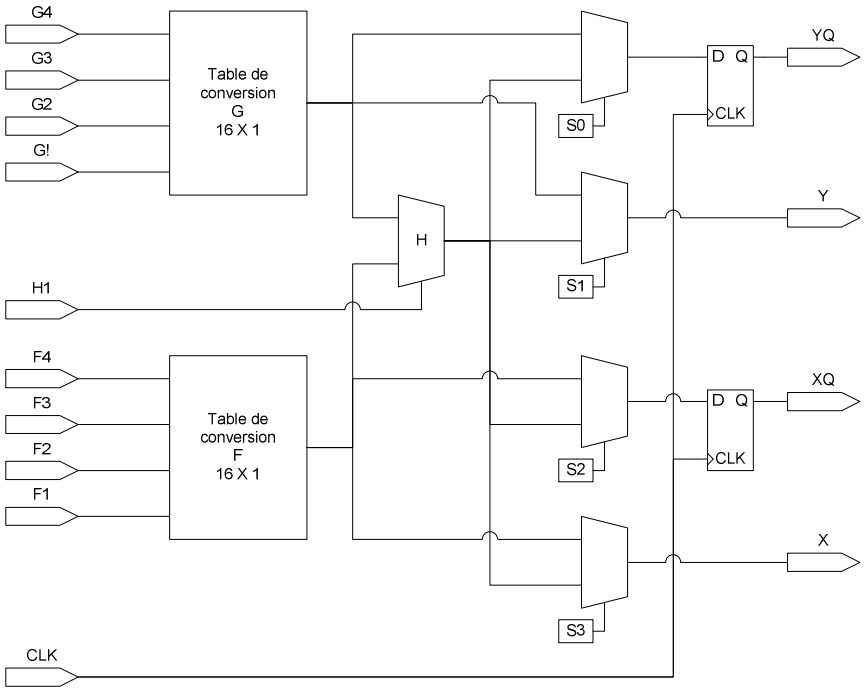
Remettez les deux feuilles dans votre cahier d'examen.

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

entity module2 is
   port (
      CLK, A, B, C, D, E : in std_logic;
      F : out std_logic
   );
end module2;

architecture arch of module2 is
signal S : std_logic_vector(1 downto 0);
begin

   process (CLK)
   begin
      if rising_edge(CLK) then
         S(0) <= A or B or C or D;
         S(1) <= A and B and not(E);
      end if;
   end process;

   process (S, A, C, D)
   begin
      if S(0) = '1' then
         F <= A or not(D);
      else
         F <= S(1) or C or D;
      end if;
   end process;

end arch;
```

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

Nom : _____ Matricule : _____





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

Nom : _____ Matricule : _____

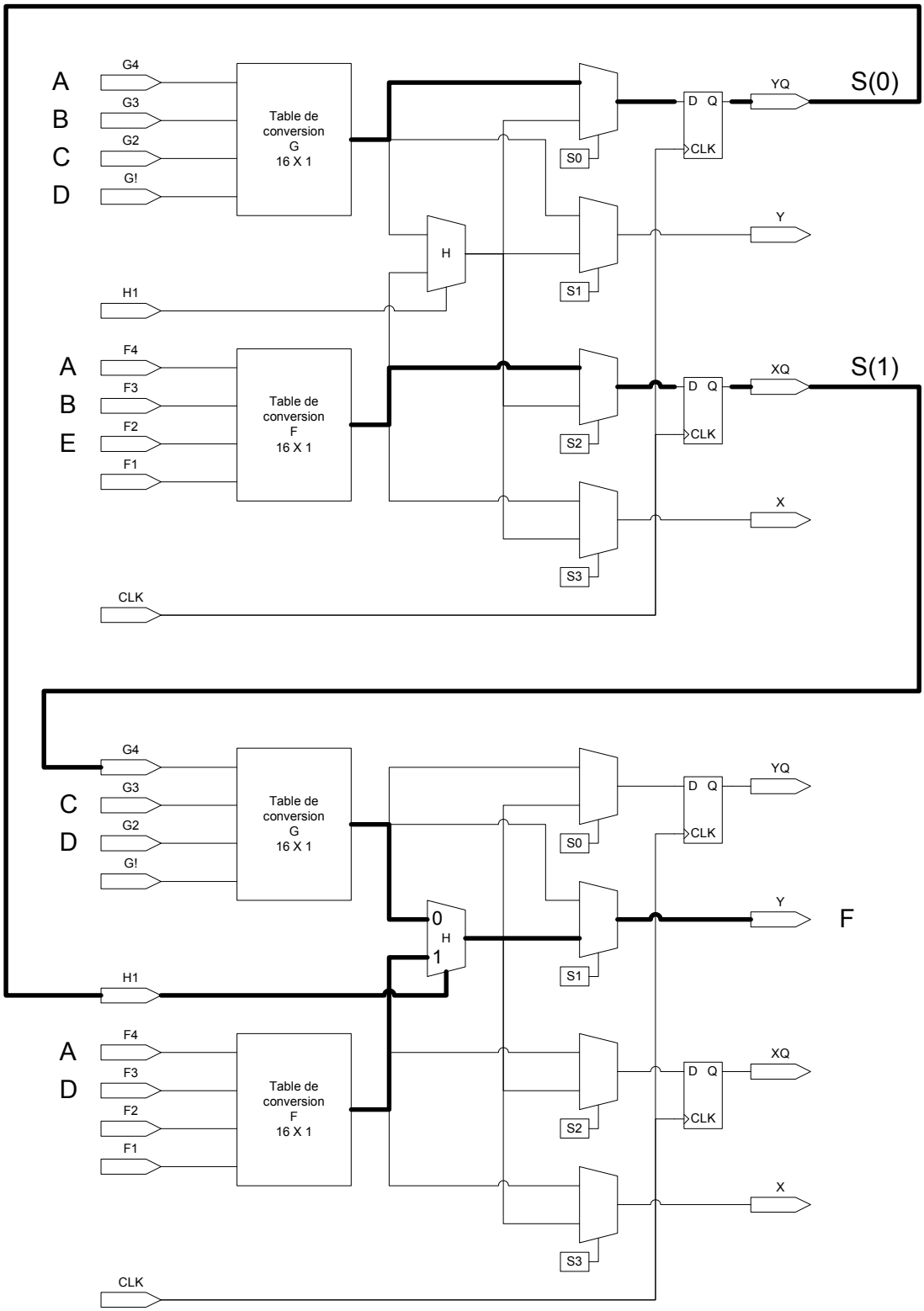
G4 ( )	G3 ( )	G2 ( )	G1 ( )	G ( )
0	0	0	0	
0	0	0	1	
0	0	1	0	
0	0	1	1	
0	1	0	0	
0	1	0	1	
0	1	1	0	
0	1	1	1	
1	0	0	0	
1	0	0	1	
1	0	1	0	
1	0	1	1	
1	1	0	0	
1	1	0	1	
1	1	1	0	
1	1	1	1	

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 :


A
B
C
D
G4
G3
G2
G!
Table de conversion G 16 X 1
H
H1
S0
S1
S2
S3
D Q
CLK
YQ
Y
XQ
X
S(0)
A
B
E
F4
F3
F2
F1
Table de conversion F 16 X 1
S(1)
C
D
A
D
0
1
F

G4 (A)	G3 (B)	G2 (C)	G1 (D)	G (S(0))
0	0	0	0	0
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	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 (A)	F3 (B)	F2 (E)	F1 (NC)	F (S(1))
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	1
1	1	0	1	1
1	1	1	0	0
1	1	1	1	0

G4 (S(1))	G3 (C)	G2 (D)	G1 (NC)	G ( )
0	0	0	0	0
0	0	0	1	0
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	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 (A)	F3 (D)	F2 (NC)	F1 (NC)	F ( )
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
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