INF3500 : CONCEPTION ET RÉALISATION DE SYSTÈMES NUMÉRIQUES

Contrôle périodique #2 (Formatif) – 11 avril 2024

Durée : 1.5 heure.

Pondération: 0%.

Nom	
Prénom	
Matricule	

Directives

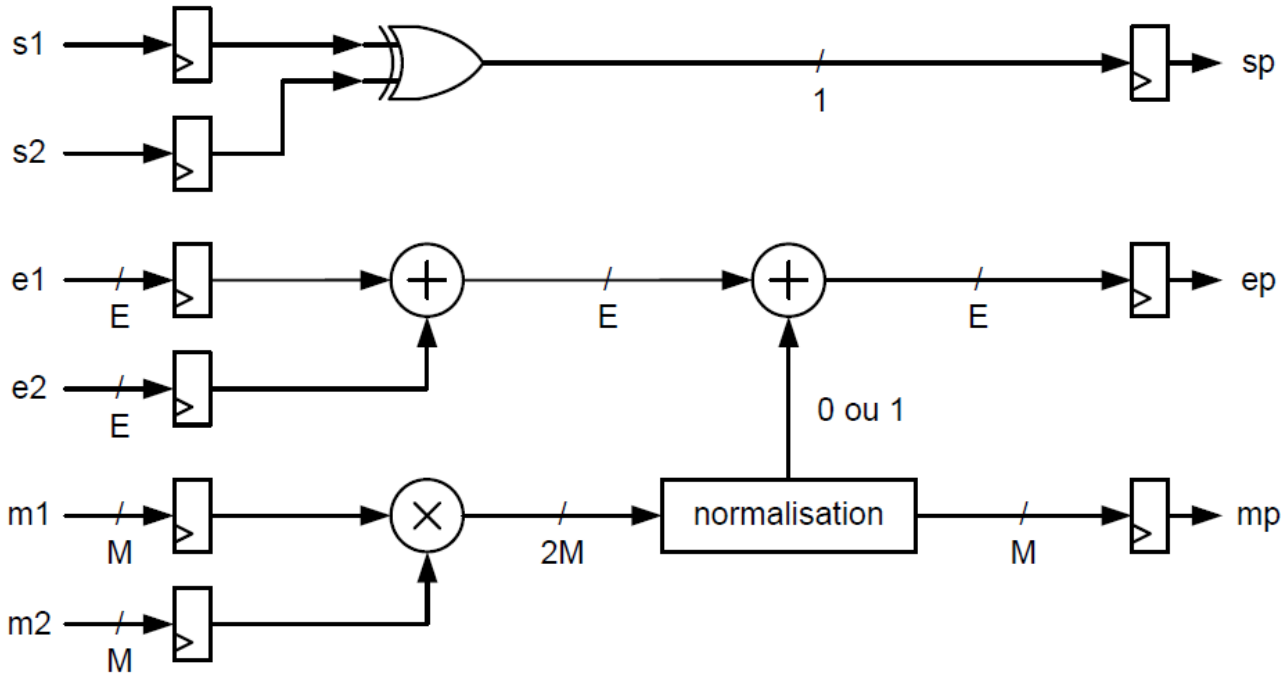
- Une feuille de note recto verso 8.5”×11” ou A4 permise.
- Calculatrice programmable permise.
- Ordinateurs interdits.
- Appareils mobiles interdits.
- Répondre à toutes les questions, la valeur de chaque question est indiquée.
- Répondre sur le questionnaire et le remettre. Au besoin, utilisez le verso des feuilles.
- Ne posez pas de questions. En cas de doute sur le sens d’une question, énoncez clairement vos suppositions.

Réservé au correcteur

Q1		/8
Q2		/3
Q3		/3
Q4		/3
Q5		/3
Total		/20

Q1 (8 points)

Considérez le diagramme suivant d'un multiplieur de nombres à virgule flottante.



Après implémentation sur un FPGA de la série 7 de Xilinx, on a caractérisé les différentes parties du circuit comme suit.

	Symbole	Valeur (ns)
Délai intrinsèque – Bascules	td	0.45
Temps préparation – Bascules	tsu	0.25
Temps de maintien – Bascules	th	0.1
Délai combinatoire – Porte OUX	toux	0.5
Délai combinatoire – Additionneur	tadd	2.6
Délai combinatoire – Multiplicateur	tmul	5.5
Délai combinatoire – Normalisation	tnor	3.1

A la suite de l'étape de placement et routage, on estime que les fils d'interconnexion entre chacun des blocs ont des délais de ti = 0.15 ns chacun.

Q1.1 Identifiez le chemin critique du circuit et donnez la fréquence maximale d'opération.
Donnez la latence du circuit en ns et le débit du circuit en résultats (multiplications) par seconde.
Montrez vos calculs.

Fréquence maximale		MHz
Latence		ns
Débit		Multiplications/Seconde

Q1.2 On désire maximiser le débit.
Indiquer, sur le diagramme de la page précédente, comment ajouter une ou des étage(s) de pipeline au circuit afin d'obtenir un débit maximum.
Vous n'avez pas la possibilité de modifier les fonctions additionneur, multiplieur et normalisation.
Donnez la nouvelle fréquence maximale d'opération, latence et débit du circuit.
Montrez vos calculs.

Fréquence maximale		MHz
Latence		ns
Débit		Multiplications/Seconde

Q1.3 Complétez le code VHDL partiel du multiplieur de nombres à virgule flottante.
Donnez le code VHDL pour la version avec pipeline telle que proposée en Q1.2.
Ne donnez que le code pour la sortie `sp`.

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

entity cp2q1_sp is
     port(
          clk     : in std_logic ;
          s1, s2  : in std_logic ;
          sp      : out std_logic );
end cp2q1_sp ;

architecture cp_avec_pipeline of cp2q1_sp is
```

```
end cp_avec_pipeline ;
```

Q1.4 Pour chacune des entrées du circuits telles que définies ci-dessous, donnez les valeurs à tester selon l'analyse des valeurs limites.

```
s1, s2 : in  std_logic;              -- bits de signe des entrées
e1, e2 : in  signed(6 downto 0);     -- exposants
m1, m2 : in  unsigned(7 downto 0);   -- mantisses
```

s1 et s2
e1 et e2
m1 et m2

Combien de vecteurs de tests sont nécessaires pour effectuer la vérification selon l'analyse des valeurs limites?

Q2 (3 points)

Considérez le code VHDL suivant pour un module combinatoire et son banc de test associé.

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

entity add3 is
      port (Cin  : in std_logic;
            X    : in std_logic;
            Y    : in std_logic;
            Cout : out std_logic;
            S    : out std_logic);
end add3;

architecture cp2q2 of add3 is
   signal SXY : std_logic;
begin

      SXY <= X xor Y ;

      S <= SXY xor Cin ;

      process (all)
      begin
         if ((SXY and Cin) or (X and Y)) then
            Cout <= '1' ;
         else
            Cout <= '0' ;
         end if ;
      end process ;

end cp2q2;
```

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

entity add3TBsimple is
end add3TBsimple;

architecture cp2q2 of add3TBsimple is
   signal Cin, X, Y, Cout, S : std_logic;
begin

   UUT : entity work.add3(cp2q1
         port map (Cin, X, Y, Cout, S);

   Cin <= '0' after 0 ns, '1' after 10 ns;
   X   <= '1' after 0 ns;
   Y   <= '0' after 0 ns, '1' after 10 ns;

end cp2q2;
```

Donnez la valeur de tous les ports et signaux internes du module combinatoire en fonction du temps, en tenant compte des délais deltas. Toutes les lignes du tableau pourraient ne pas être requises.

Temps	Delta	SXY	Cin	X	Y	Cout	S
0 ns	0	U	U	U	U	U	U
0 ns	1						
0 ns	2						
0 ns	3						
0 ns	4						
10 ns	0						
10 ns	1						
10 ns	2						
10 ns	3						
10 ns	4						

Q3 (3 points)

Pour chacun des cas suivants, indiquez la valeur de la variable V et du signal S suite à l'exécution du *process*.

```
architecture cas1 of sigvar is
    signal S : integer := 0;
begin
    process
        variable V : integer := 0 ;
    begin
        for i in 0 to 4 loop
            V := V + 1 ;
            S <= V ;
        end loop ;
        report "V = " & to_string(V);
        report "S = " & to_string(S);
        wait ;
    end process;
end cas1 ;
```

V = _____

S = _____

```
architecture cas2 of sigvar is
    signal S : integer := 0;
begin
    process
        variable V : natural ;
    begin
        for i in 0 to 9 loop
            S <= S + 1;
        end loop ;
        wait for 10 ns ;
        report "V = " & to_string(V);
        report "S = " & to_string(S);
        wait ;
    end process;
end cas2 ;
```

V = _____

S = _____

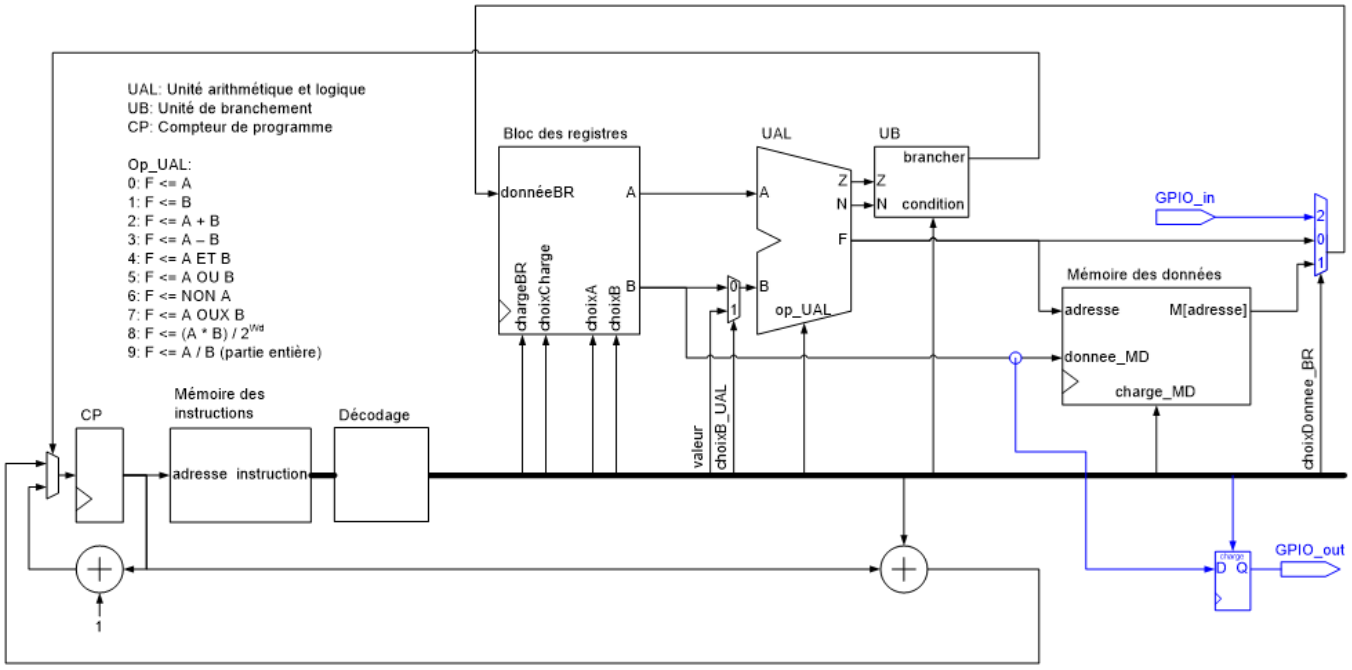
```
architecture cas3 of sigvar is
    signal S : integer := 0;
begin
    process
        variable V : positive;
    begin
        for i in 2 downto 0 loop
            V := V + 1 ;
            S <= S + V ;
            wait for 10 ns ;
        end loop ;
        report "V = " & to_string(V);
        report "S = " & to_string(S);
        wait ;
    end process;
end cas3 ;
```

V = _____

S = _____

Q4 (3 points)

Considérez le diagramme du processeur PolyRISC.



Q4.1 Indiquez si chaque opération ou groupe d'opérations simultanées suivants est possible.
Si oui, donnez la valeur des signaux de contrôle correspondants. Si non, en donner la raison.

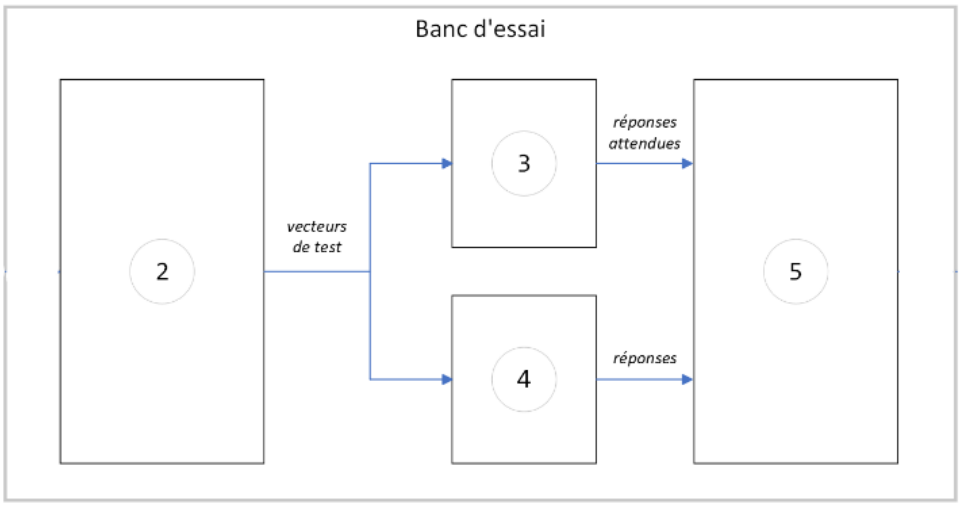
	Opération	*Possible (oui/non)*	chargeBR	choixCharge	choixA	choixB	valeur	choixB_UAL	op_UAL	charge_MD	choixDonnee_BR	charge_GPIO_out
1	`R9 := M[R8+R11] ;`											
2	`R13 := R6+R2 ;` `MD[R6+R2] := R6;`											
3	`R8 := R0 ET 64;` `MD[R1+64] := R3;`											
4	`R8 := GPIO_in ;` `GPIO_out := R8 ;`											
5	`MD[R0+R2] := GPIO_in`											

Q5 (3 points)

Q5.1 Vrai ou faux? Une couverture de code de 100% des énoncés pour un ensemble de vecteurs de test garantit que le circuit rencontre toutes ses spécifications. Justifiez votre réponse.

Q5.2 On doit synchroniser un signal entre deux domaines d'horloge différents. En supposant que la fréquence CLK2 est beaucoup plus grande que la fréquence CLK1, montrez sur le schéma comment minimiser le risque de métastabilité pour un signal allant du domaine CLK1 vers le domaine CLK2.

Q5.3 Considérez le diagramme de banc d'essai suivant. Identifiez la fonction effectuée par chacun des blocs.



Bloc 2	
Bloc 3	
Bloc 4	
Bloc 5	

CETTE PAGE NE SERA PAS CORRIGÉE. À UTILISER COMME BROUILLON.

CETTE PAGE NE SERA PAS CORRIGÉE. À UTILISER COMME BROUILLON.

Solutions

Q1

Q1.1

Le chemin critique va d'une des bascules m, à travers le multiplieur, le bloc de normalisation, l'additionneur, puis la bascule pour ep.

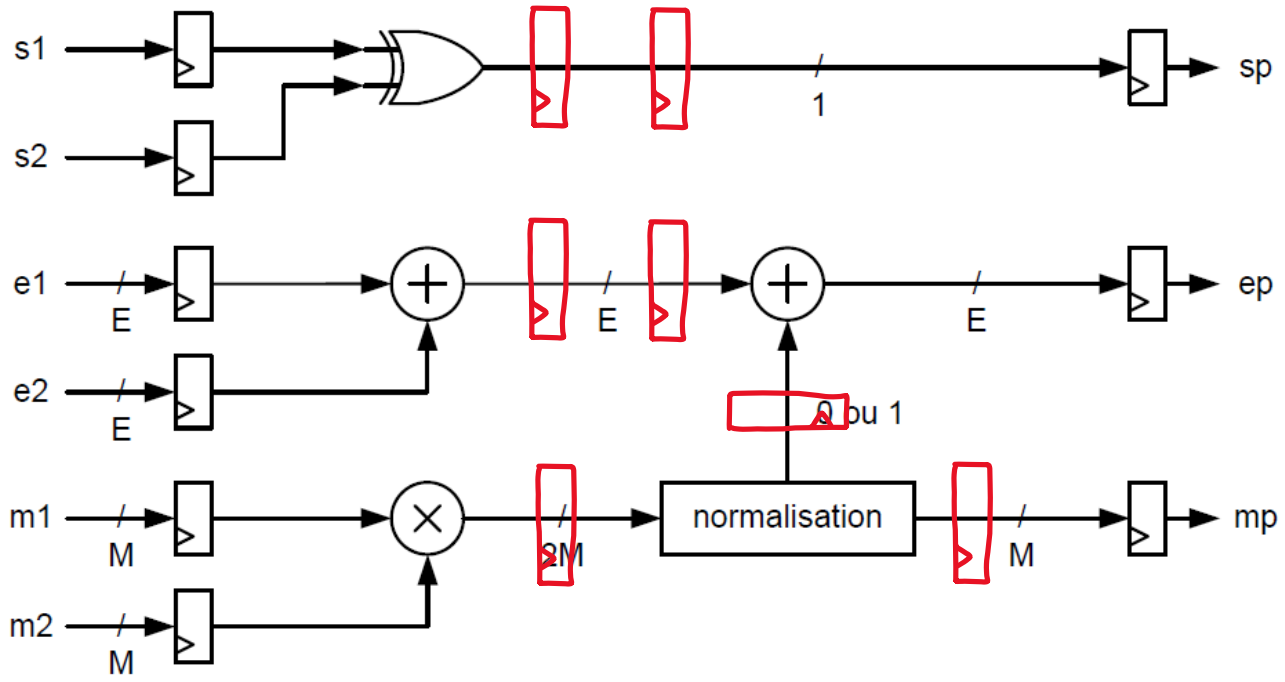
```
Tmin = td + tmul + tnor + tadd + 4*ti + tsu
Tmin = 0.45 + 5.5 + 3.1 + 2.6 + 4*0.15 + 0.25 = 12.5 ns
Fmax = 1/Tmin = 80 MHz
```

La latence du circuit est d'un seul cycle (sans compter la bascule à l'entrée).

```
Latence = 12.5 ns
Débit = 80M Mult/Seconde
```

Q1.2

Afin d'atteindre un débit maximal, il faut placer des registres de pipelines entre chacun des blocs combinatoires (additionneur, multiplicateur, normalisation), et ensuite équilibrer les autres chemins.



Le nouveau chemin critique va d'une des bascules m, à travers le multiplieur puis le nouveau registre qui suit.

```
Tmin = td + tmul + 2*ti + tsu
Tmin = 0.45 + 5.5 + 2*0.15 + 0.25 = 6.5 ns
Fmax = 1/Tmin = 154 MHz
```

La latence du circuit est maintenant de trois cycles.

```
Latence = 3 * 6.5 ns = 19.5 ns
Débit = 154M Mult/Seconde
```

Q1.3

```
-- sans pipeline

architecture cp of cp2q1_sp is
   signal s1i, s2i : std_logic ;
begin

   process (clk)
   begin
      if rising_edge(clk) then
         s1i <= s1 ;
         s2i <= s2 ;
         sp <= s1i xor s2i ;
      end if ;
   end process ;

end cp ;

-- avec pipeline

architecture cp_avec_pipeline of cp2q1_sp is
   constant N : natural := 2 ; -- nombre etages pipeline
   signal s1i, s2i : std_logic ;
   signal spi : std_logic_vector (0 to N-1) ;
begin

   process (clk)
   begin
      if rising_edge(clk) then
           s1i <= s1 ;
           s2i <= s2 ;
           spi(0) <= s1i xor s2i ;
           for i in 1 to N-1 loop
                spi(i) <= spi(i-1) ;
           end loop ;
           sp <= spi(N-1) ;
      end if ;
   end process ;

end cp_avec_pipeline ;
```

Q1.4

Pour les entrées s1 et s2, {0, 1}.

Pour les entrées e1 et e2, en entier : {-64, -63, 0, 62, 63}

En signed : {1000000, 1000001, 0000000, 0111110, 0111111}

Pour les entrées m1 et m2 : (0, 1, 170, 254, 255}

En unsigned {00000000, 00000001, 10101010, 11111110, 11111111}

Il faudrait prendre 2 x 2 x 5 × 5 × 5 × 5 = 2500 vecteurs de test.

Q2

Time	Delta	SXY	Cin	X	Y	Cout	S
0 ps	0	U	U	U	U	U	U
0 ps	1	U	0	1	0	0	U
0 ps	2	1	0	1	0	0	U
0 ps	3	1	0	1	0	0	1
10000 ps	0	1	1	1	1	0	1
10000 ps	1	0	1	1	1	1	0
10000 ps	2	0	1	1	1	1	1

Q3

cas1 : V = 5 S = 0

cas2 : V = 0 S = 1

cas3 : V = 4 S = 9

Q4

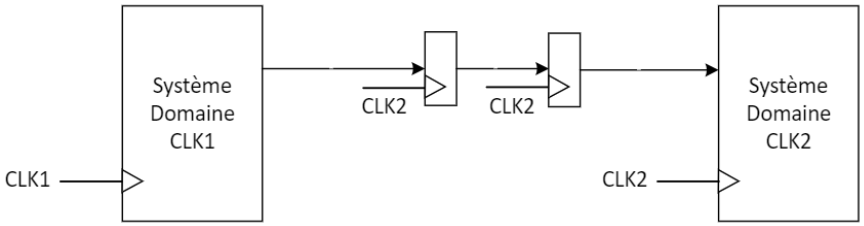
	Opération	*Possible (oui/non)*	chargeBR	choixCharge	choixA	choixB	valeur	choixB_UAL	Op_UAL	Charge_MD	choixDonnee_BR	Charge_GPIO_out
1	`R9 := M[R8+R11] ;`	oui	1	9	8	11	-	0	2	0	1	0
2	`R13 := R6+R2 ;` `M[R6+R2] := R6;`	oui	1	13	2	6	-	0	2	1	0	0
3	`R8 := R0 ET 64;` `M[R1+64] := R3;`	non	L'UAL n'a qu'une sortie et ne peut donc pas faire l'addition et l'opération logique ET en même temps.									
4	`R8 := GPIO_in ;` `GPIO_out := R8 ;`	oui	1	8	-	8	-	-	-	0	2	1
5	`MD[R0+R2] := GPIO_in`	non	On ne peut pas écrire directement GPIO_in en mémoire.									

Q5

Q5.1

Faux.
La couverture de code n'indique que si certaines situations ont été exercées ou non, sans égard à la fonctionnalité du système.

Q5.2



Q5.3

Bloc 2 Génération des vecteurs de test.

Bloc 3 Génération des réponses attendues aux vecteurs de test.

Bloc 4 Instanciation du circuit à vérifier (UUT).

Bloc 5 Comparaison des réponses du circuit aux réponses attendues, et indiquer toute différence entre les deux par une condition d'erreur;