

---

## INF3500 : CONCEPTION ET RÉALISATION DE SYSTÈMES NUMÉRIQUES

Contrôle périodique #2 (Formatif) – 1 décembre 2023

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.

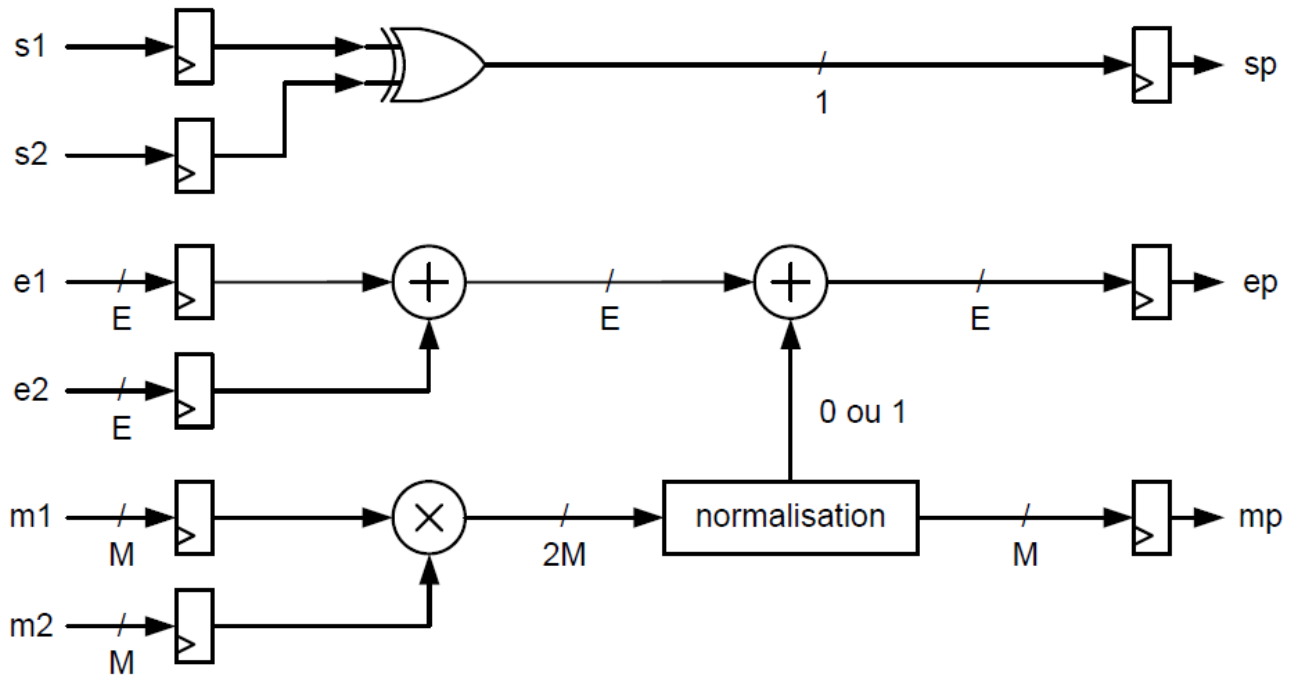
### Réservé au correcteur

---

Q1		/6
Q2		/3
Q3		/4
Q4		/5
Q5		/2
T		/20

**Q1 (6 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	$t_d$	0.45
Temps préparation – Bascules	$t_{su}$	0.25
Temps de maintien – Bascules	$t_h$	0.1
Délai combinatoire – Porte OUX	$t_{oux}$	0.5
Délai combinatoire – Additionneur	$t_{add}$	2.6
Délai combinatoire – Multiplicateur	$t_{mul}$	5.5
Délai combinatoire – Normalisation	$t_{nor}$	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  $t_i = 0.15$  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 par seconde. Montrez vos calculs.

[illegible]

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
```

[illegible]

```
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 (4 points)**

Considérez le code VHDL suivant :

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

entity cp2q3 is
    port (clk : in std_logic ;
          X : in unsigned(3 downto 0) ;
          Y : in unsigned(2 downto 0) ;
          M : out unsigned(3 downto 0)) ;
end cp2q3;

architecture cp of cp2q3 is
begin

    process (clk)
    begin
        if rising_edge(clk) then
            M <= X + resize(Y*2,M'length) ;
        end if ;
    end process;

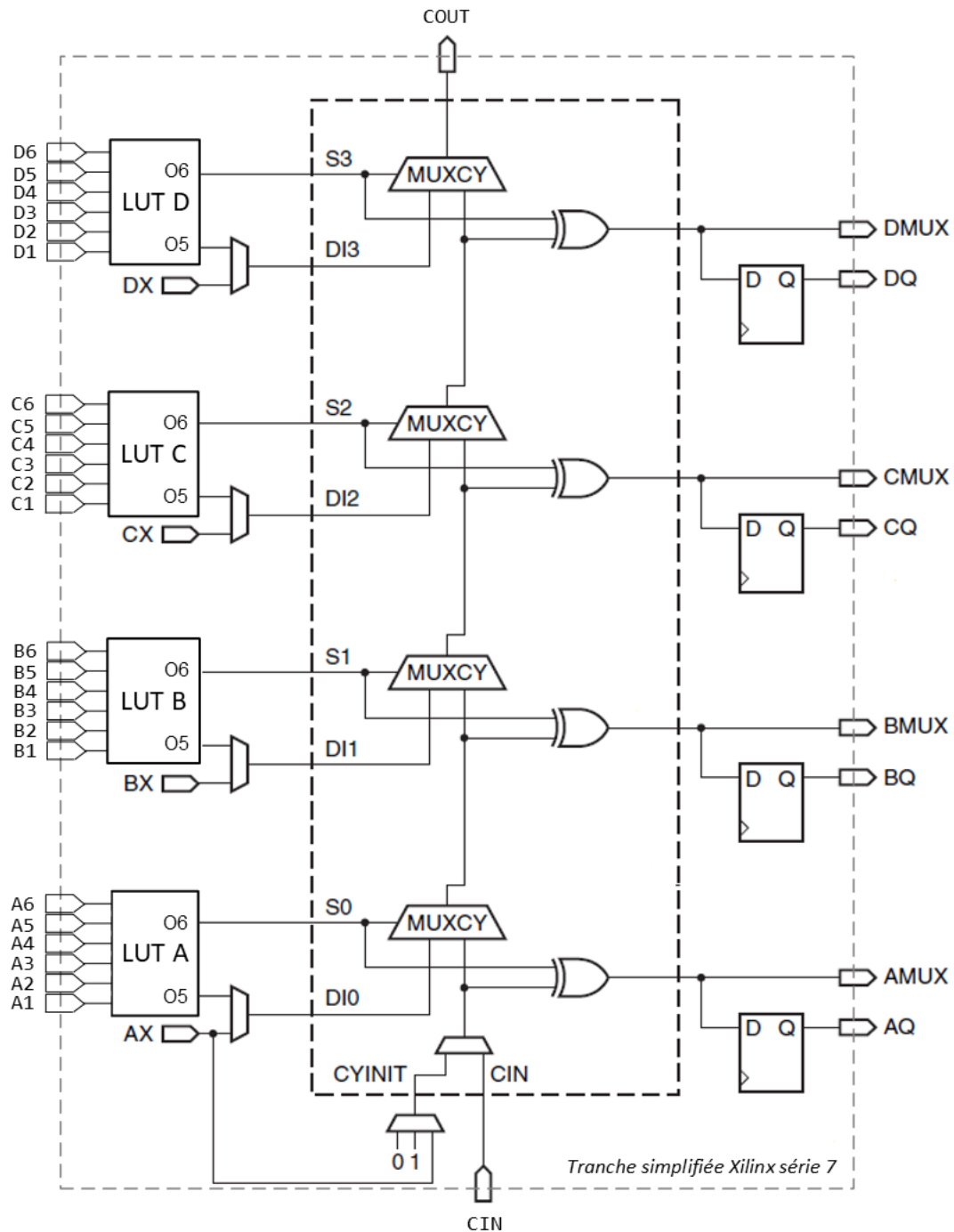
end cp ;
```

**Q3.1** En utilisant le schéma de la page suivante, montrez un résultat possible de synthèse pour ce circuit. Le schéma représente en version simplifiée une tranche d'un FPGA Xilinx de la série 7.

- Identifiez clairement sur le dessin où vous placez les entrées et les sorties du circuit. Vous n'avez pas à placer le signal d'entrée *clk*.
- Identifiez clairement sur le dessin les chemins utilisés entre les entrées et les sorties

**Q3.2** Donnez la configuration des LUTs afin d'implémenter la fonction désirée. Exprimez votre réponse sous formes d'équations booléennes pour chacune des sorties O6 et O5. Laissez la case vide si non utilisée.

LUT D	O6 =	O5 =
LUT C	O6 =	O5 =
LUT B	O6 =	O5 =
LUT A	O6 =	O5 =



**Q4 (5 points)**

Une suite de Fibonacci est une suite d'entiers dans laquelle chaque terme est la somme des deux termes qui le précèdent.

$$F_0 = 0, F_1 = 1 \text{ et } F_n = F_{n-1} + F_{n-2} \text{ pour } n \geq 2.$$

$F_0$	$F_1$	$F_2$	$F_3$	$F_4$	$F_5$	$F_6$	$F_7$	$F_8$	$F_9$	$F_{10}$	$F_{11}$	$F_{12}$	...
0	1	1	2	3	5	8	13	21	34	55	89	144	...

On désire concevoir un circuit qui calcule la somme de tous les termes impairs de la suite de Fibonacci dont les valeurs sont plus petites que  $2^{16}$ .

Le circuit a une entrée : INIT (1 bit), et deux sorties : S (17 bits) et FINI (1 bit). Lorsque INIT= '1', le circuit est forcé dans son état initial, soit FN=  $F_1$  et FP=  $F_0$ . Toutes les opérations sont synchronisées par un signal CLK implicite, vous n'avez pas besoin de l'inclure dans votre schéma. A chaque front montant de l'horloge, le prochain terme est calculé et accumulé dans la somme S s'il est impair. La sortie FINI doit prendre la valeur 1 quand les calculs sont terminés, ce qui indique que la sortie S contient la somme finale.

Vous aurez besoin des registres suivants :

Registre	Dimension	Valeur initiale	Description
FN	16 bits	1 ( $F_1$ )	Le terme courant de la suite
FP	16 bits	0 ( $F_0$ )	Le terme précédent de la suite
S	17 bits	0	La somme des termes impairs de la suite
FINI	1 bit	0	Indique que le calcul est terminé ( $FN \geq 2^{16}$ )

**Q4.1** Donnez un diagramme de chemin des données qui implémente ce circuit (répondez dans l'encadré de la page suivante).

**Q4.2** Estimez le nombre de ressources nécessaires en termes de tables de conversion (LUT), de bascules D (FF) et de tranches DSP48 pour implémenter votre circuit sur un FPGA Xilinx de la série 7. Justifiez vos réponses.

LUT

---



---



---

Bascules D (FF)

---



---



---

DSP48

---



---



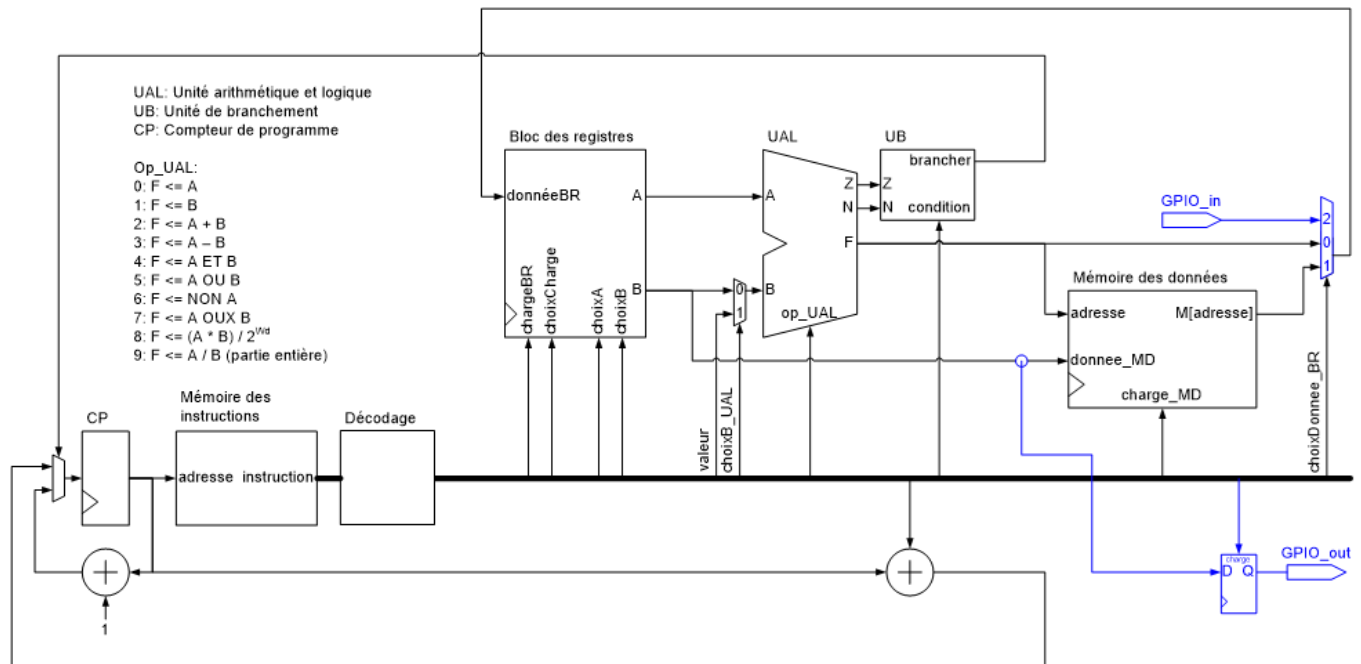
---



### Réponse Q4.1

**Q5 (2 points)**

Considérez le diagramme du processeur PolyRISC.



**Q5.1** 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											

*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.

$$T_{min} = t_d + t_{mul} + t_{nor} + t_{add} + 4 \cdot t_i + t_{su}$$

$$T_{min} = 0.45 + 5.5 + 3.1 + 2.6 + 4 \cdot 0.15 + 0.25 = 12.5 \text{ ns}$$

$$F_{max} = 1/T_{min} = 80 \text{ MHz}$$

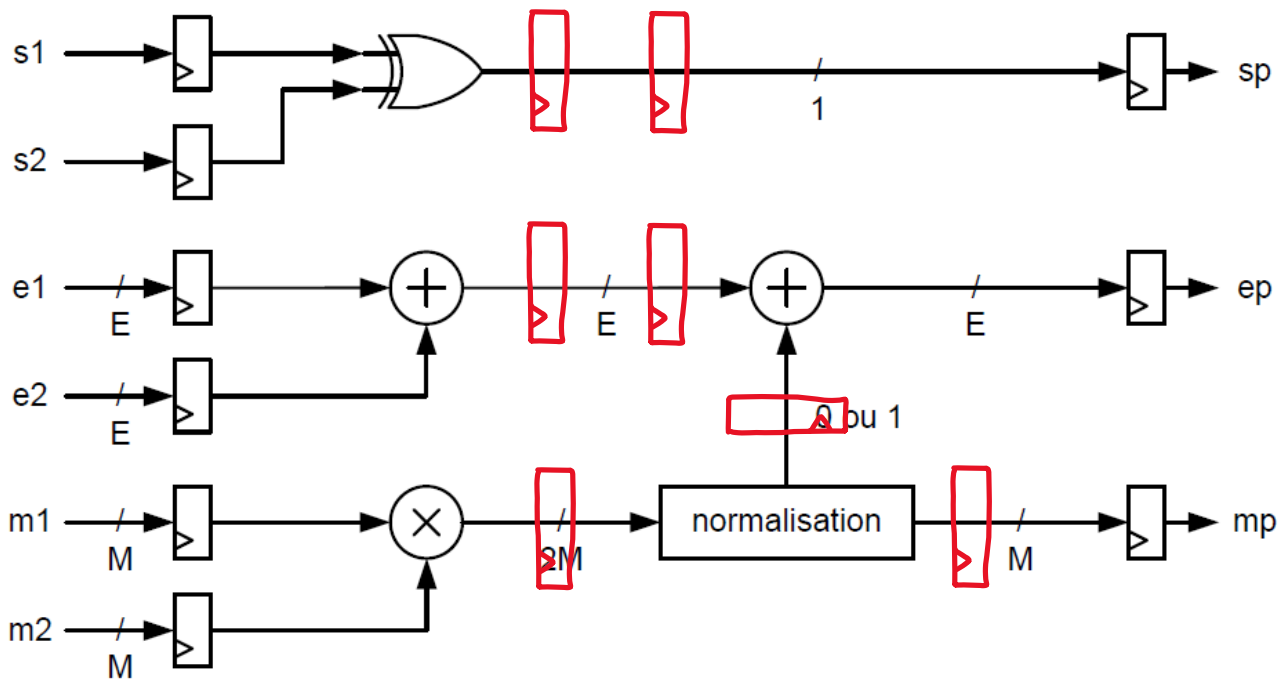
La latence du circuit est d'un seul cycle (sans compter la bascule à l'entrée).

$$\text{Latence} = 12.5 \text{ ns}$$

$$\text{Débit} = 80 \text{ M Mult/Seconde}$$

#### Q1.2

Afin d'atteindre un débit maximal, il faut placer des registres de pipelines entre chacun des blocs combinatoires (additionneur, multiplieur, 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.

$$T_{min} = t_d + t_{mul} + 2 \cdot t_i + t_{su}$$

$$T_{min} = 0.45 + 5.5 + 2 \cdot 0.15 + 0.25 = 6.5 \text{ ns}$$

$$F_{max} = 1/T_{min} = 154 \text{ MHz}$$

La latence du circuit est maintenant de trois cycles.

$$\text{Latence} = 3 \cdot 6.5 \text{ ns} = 19.5 \text{ ns}$$

$$\text{Débit} = 154 \text{ M Mult/Seconde}$$

## Q1.3

```

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

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

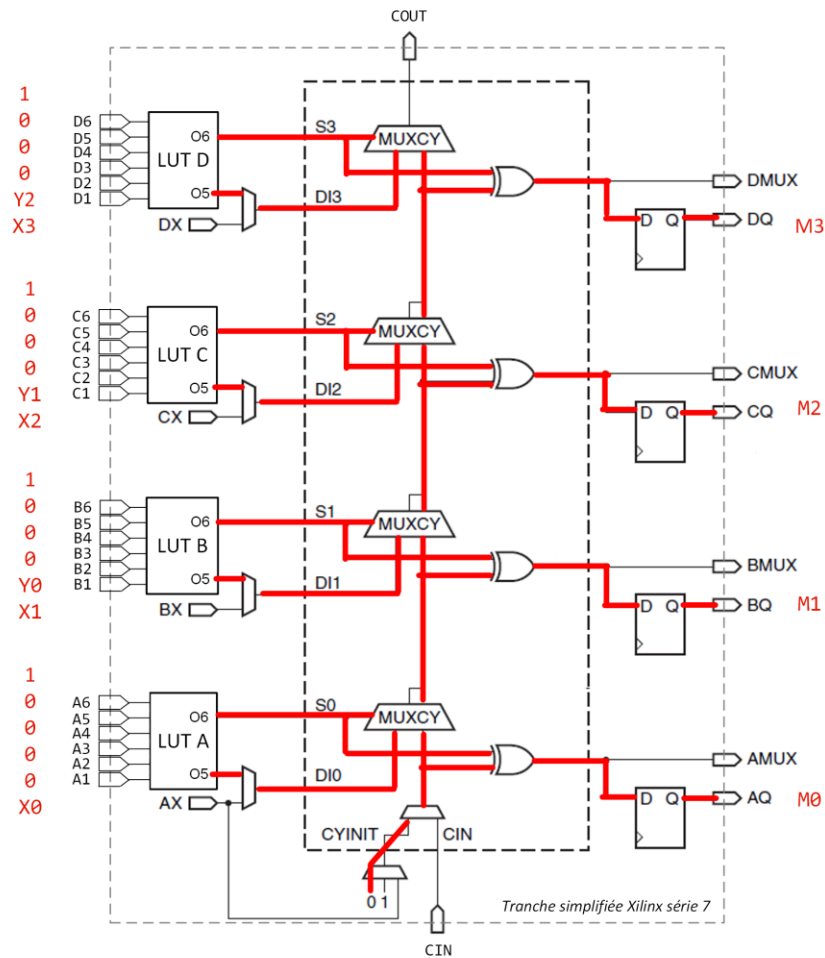
En unsigned {00000000, 00000001, 10101010, 11111110, 11111111}

Il faudrait prendre  $2 \times 2 \times 5 \times 5 \times 5 \times 5 = 2500$  vecteurs de test.

## Q2

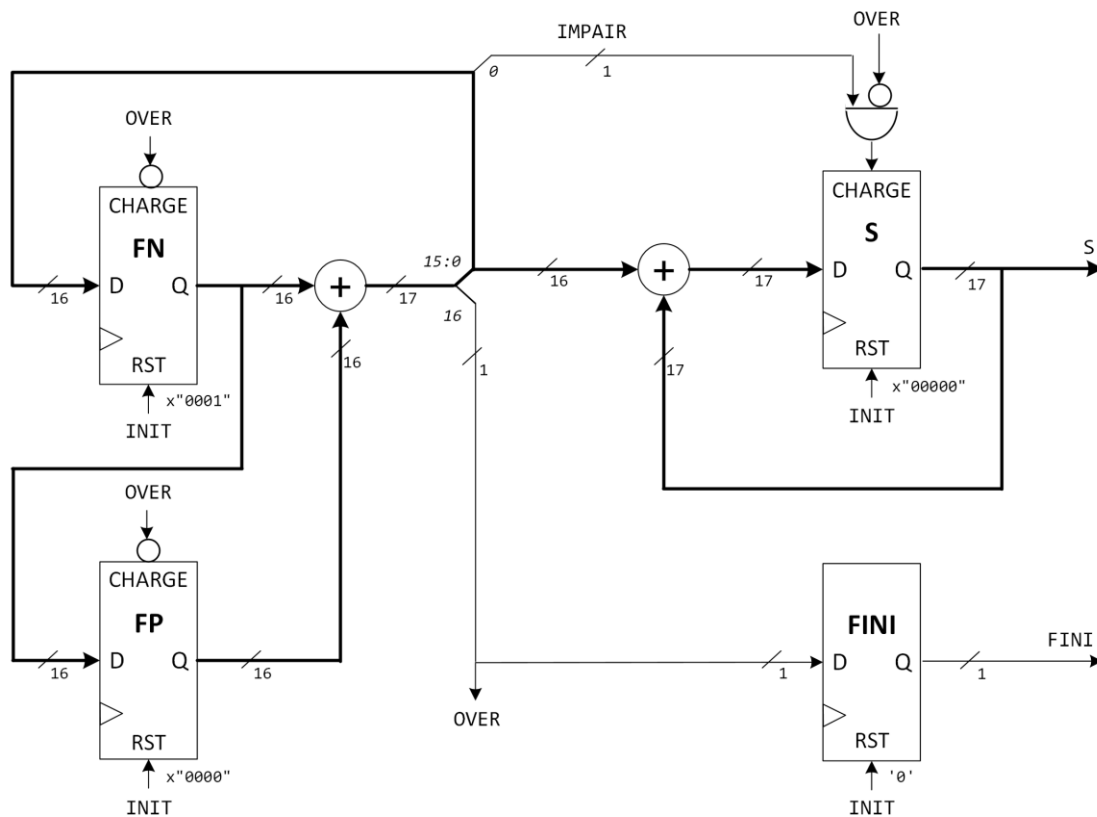
Time	Delta	 SXY	 Cin	 X	 Y	 Cout	 S
0 ps	0	U	U	U	U	U	U
0 ps	1	U	0	1	0	0	U
0 ps	2	1	0	1	0	0	U
0 ps	3	1	0	1	0	0	1
10000 ps	0	1	1	1	1	0	1
10000 ps	1	0	1	1	1	1	0
10000 ps	2	0	1	1	1	1	1

## Q3



LUT D	$O6 = X3 \text{ XOR } Y2$	$O5 = Y2$
LUT C	$O6 = X2 \text{ XOR } Y1$	$O5 = Y1$
LUT B	$O6 = X1 \text{ XOR } Y0$	$O5 = Y0$
LUT A	$O6 = X0$	$O5 = 0$

## Q4



## LUT

34 LUT total.

Additionneur FN (16 LUT), Additionneur S (17 LUT)

Contrôle charge S (1 LUT)

## Bascules D (FF)

50 FF

FN (16) + FP (16) + S (17) + FINI (1)

## DSP48

0 DSP48

(autres solutions possibles - ex. avec multiplexeurs).



## Q5

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.									