Nom : _____ Matricule : _____

INF3500 : Conception et réalisation de systèmes numériques

Examen intra #1 – 15 février 2021

1. Conditions de l'examen

- Cet examen est calibré pour une durée de 60 minutes. Une période additionnelle de 15 minutes est allouée pour la remise de l'examen dans Moodle.
- L'examen est à livres ouverts, toutes les sources sont permises. Si vous utilisez du matériel trouvé en ligne ou ailleurs, citez bien vos sources dans votre réponse.
- L'examen doit être fait individuellement. Toute forme de communication avec d'autres personnes que le surveillant est interdite pendant l'examen. Une attestation à cet effet vous sera demandée dans la dernière question.
- La pondération de l'examen est de 10 %. La valeur de chaque question est indiquée.

2. Communication avec le surveillant pendant l'examen

- Le surveillant est disponible via le serveur Discord du cours pour aider à résoudre les problèmes techniques. Si le serveur Discord du cours ne fonctionne pas, envoyez un courriel à pierre.langlois@polymtl.ca.
- Ne posez pas de questions au surveillant concernant les questions de l'examen. Aucune réponse ne sera donnée. En cas de doute sur le sens d'une question, faites une supposition raisonnable, énoncez-la clairement dans votre réponse et poursuivez.

3. À propos de la remise

- Remettez vos réponses dans l'activité devoir de Moodle avant 16 h (heure de Montréal).
- Les réponses en retard sont permises mais seront pénalisées de 1 point (10 %) par minute après 16 h.
- Les réponses doivent être remises dans deux fichiers :
  - un seul fichier .pdf pour les questions 1, 2, 3 et 5
  - un seul fichier .vhd pour la question 4.
- Sauvegardez souvent votre travail. Déposez régulièrement vos fichiers les plus récents dans Moodle.
- Dans le document .pdf, en plus des réponses tapées et des tableaux remplis, il est acceptable d'inclure des captures d'écran, des diagrammes faits à la main, du texte écrit à la main etc.
- Pour les images incluses, assurez-vous :
  - qu'elles soient orientées correctement, bien cadrées, bien rognées et bien alignées;
  - qu'elles soient nettes partout, que l'éclairage soit uniforme et que le contraste soit suffisamment élevé;
  - d'en limiter la taille – le document au complet doit faire moins de 5 Mo.
- Commencez la réponse à chaque question sur une nouvelle page.

À l'usage des correcteurs :

Q1	Q2	Q3	Q4	Q5 ok ?	Total sur 10

Question 1. (3 points)

C'est l'été 2021 et vous travaillez à distance pour la compagnie FPGA3500 Solutions Consultants Inc. La présidente vous appelle par Zoom et vous demande votre avis pour choisir parmi trois technologies pour l'implémentation d'un processeur pour un système de détection du virus de la Covid-19 par Test d'amplification des Acides Nucléiques (TAN) en temps réel. Les données de comparaison des trois technologies sont présentées au tableau suivant.

Technologie	Frais fixes supplémentaires	Coût unitaire	Fréquence d'horloge max.	Ressources de calcul (blocs logiques)	Livraison
A. Logique fixe	625 k$	3 $	900 MHz	10000	6 mois
B. FPGA de moyenne gamme	0 $	140 $	550 MHz	3000	2 mois
C. FPGA bas de gamme	0 $	25 $	225 MHz	1000	1 mois

Le système doit traiter des données à un débit de 750×10^6 résultats par seconde. Le processeur produit un résultat par cycle d'horloge. Il est possible d'en instancier plusieurs en parallèle et ainsi de multiplier le débit. Le processeur occupe un espace équivalent à 1325 blocs logiques. La présidente prévoit que 1000 systèmes pourraient être vendus dans un mois, 25000 dans deux mois et seulement 2000 dans 8 mois parce qu'on s'attend à ce que la pandémie soit alors passée.

Quelle technologie privilégiez-vous ? Pourquoi ?

Énoncez clairement toutes les suppositions raisonnables que vous faites et montrez tous vos calculs.

Si vous le souhaitez, vous pouvez consigner vos calculs et réponses numériques dans le tableau suivant. Vous pouvez ajouter ou retrancher des lignes.

Catégorie	A. Logique fixe	B. FPGA moyenne gamme	C. FPGA bas de gamme

Question 2. (1 point)

Faites la conception d'un circuit CMOS qui implémente le code VHDL suivant.

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

entity combinatoire2a is
    port (
        x1, x2, x3 : in std_logic;
        y2 : out std_logic
    );
end combinatoire2a;

architecture arch of combinatoire2a is

signal p3, p4 : std_logic;

begin

    y2 <= p3 and p4;
    p3 <= not x1 or not x2;
    p4 <= x1 or not x2 or not x3;

end arch;
```

Donnez le diagramme d'un circuit CMOS qui implémente ce module. Utilisez le moins de transistors possible. Les entrées x1, x2 et x3 ne sont disponibles qu'en valeur naturelle non complémentée.

Question 3. (3 points)

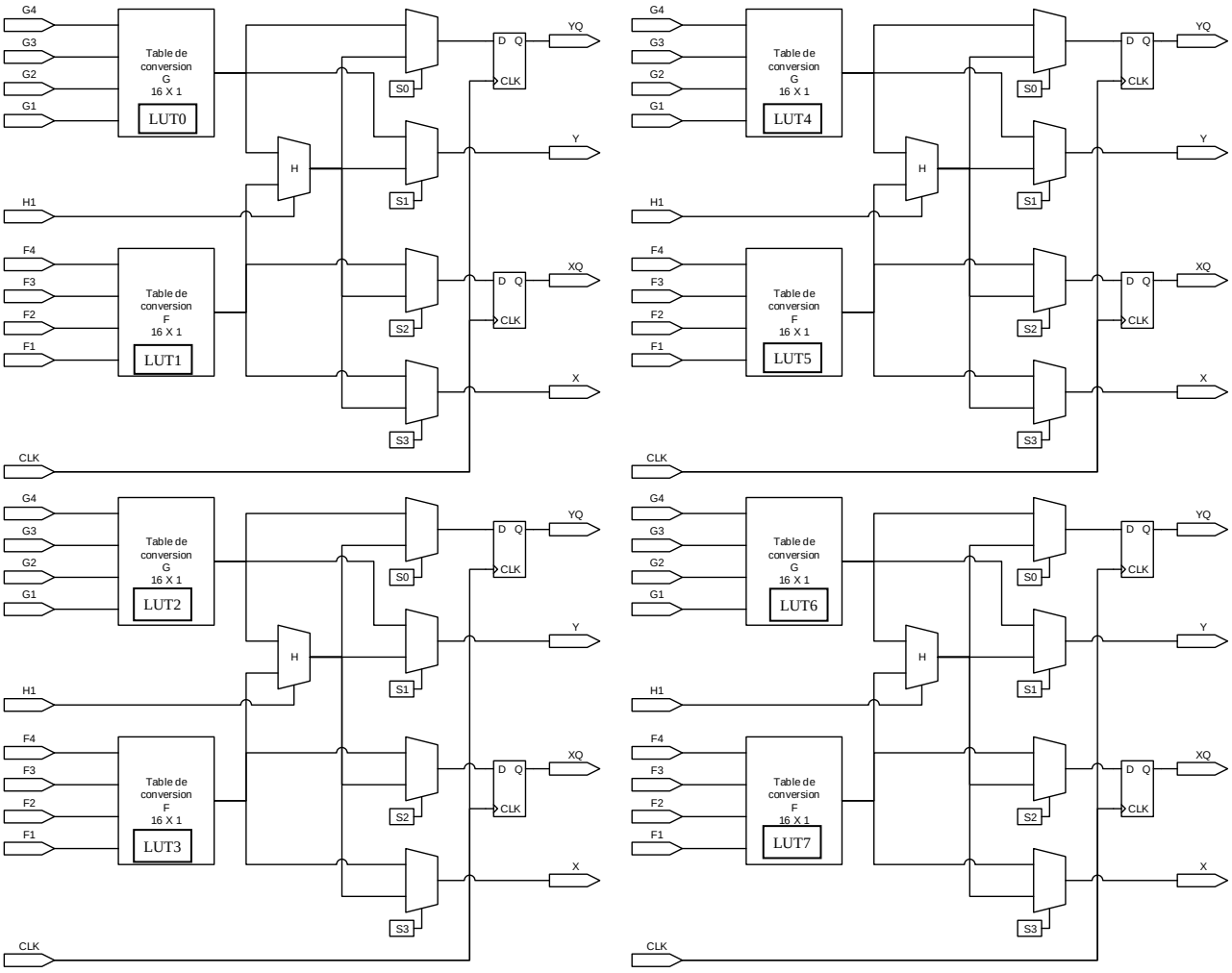
Annotez le diagramme de la page suivante pour montrer comment implémenter le code VHDL combinatoire suivant.

Indiquez clairement les entrées, les sorties, et le cheminement des signaux à l'intérieur et entre les tranches du modèle de FPGA.

Il est possible d'effectuer des connexions de la sortie d'une tranche (par exemple, d'une sortie Y) vers l'entrée d'une table de conversion (*Look-Up Table* – LUT) d'une autre tranche (par exemple, une entrée F).

Montrez le contenu de chaque LUT que vous utilisez dans le tableau.

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

entity combinatoire5 is
  port (
        s, t : in std_logic;
        op : in unsigned(2 downto 0);
        A, B, C : out std_logic
    );
end combinatoire5;

architecture arch of combinatoire5 is
begin

    process(all)
    begin
        case to_integer(op(1 downto 0)) is
            when 0 => A <= s or t;
            when 1 => A <= s and t;
            when 2 => A <= s xor t;
            when others => A <= not(s);
        end case;
    end process;

    process(all)
    begin
        if op(2) = '1' then
            B <= s;
        else
            B <= t;
        end if;
    end process;

    C <= (s xor t) or (op(2) and op(1) and op(0));

end arch;
```



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								

Question 4. (3 points)

Faites la conception d'un circuit combinatoire qui calcule la priorité d'une personne à recevoir le vaccin contre la Covid-19 et modélisez-le en VHDL.

Le circuit a une sortie, `priorite`, qui encode avec un nombre non signé de N bits la priorité correspondante au risque le plus élevé de la personne. Il a une entrée qui est un vecteur de 2 ** N bits, `FR`, qui représentent les *facteurs de risques* de la personne à évaluer. Par exemple, pour N = 2 on aurait 4 facteurs de risque, qui pourraient être, en ordre décroissant de risque :

- FR niveau 0 : vivre dans un centre d'hébergement et de soins de longue durée (CHSLD) | priorité "00";
- FR niveau 1 : être un travailleur de la santé | priorité "01";
- FR niveau 2 : être âgé de plus de 60 ans | priorité "10";
- FR niveau 3 : avoir une maladie chronique | priorité "11".

Une personne peut avoir plus d'un facteur de risque à la fois, mais alors c'est le facteur le plus important qui détermine la priorité. Par exemple, une personne de plus de 60 ans avec une maladie chronique (FR = "0011") aurait une priorité de 2 ("10"). Si elle vit en CHSLD (FR = "1011"), sa priorité serait de 0 ("00"). Une personne peut aussi n'avoir aucun facteur de risque (FR = "0000"), auquel cas une sortie spéciale `population_generale` d'un seul bit est activée et le code de la priorité est alors sans importance (*don't care*). Cette personne recevrait donc son vaccin après que toutes les personnes ayant des facteurs de risque aient été vaccinées.

La table de vérité suivante énumère les sorties pour le cas N = 2 :

FR(0)	FR(1)	FR(2)	FR(3)	population_generale	priorite(1)	priorite(0)
0	0	0	0	1	-	-
0	0	0	1	0	1	1
0	0	1	0	0	1	0
0	0	1	1	0	1	0
0	1	0	0	0	0	1
0	1	0	1	0	0	1
0	1	1	0	0	0	1
0	1	1	1	0	0	1
1	0	0	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	0	0	0
1	0	1	1	0	0	0
1	1	0	0	0	0	0
1	1	0	1	0	0	0
1	1	1	0	0	0	0
1	1	1	1	0	0	0

Pour un maximum de 2 points sur 3 : donnez une architecture synthétisable pour l'entité suivante qui fonctionne correctement pour N = 2, c'est-à-dire qu'il y a 4 facteurs de risques et 4 niveaux de priorité.

Pour un maximum de 3 points sur 3 : donnez une architecture synthétisable pour l'entité suivante qui fonctionne correctement pour toute valeur de N égale ou inférieure 5, c'est-à-dire avec au maximum 32 facteurs de risques et 32 niveaux de priorité.

Remettez votre fichier `vaccin_qc_priorite.vhd` complété. Ne modifiez pas le nom du fichier, le nom de l'entité, le `generic`, les ports, ni le nom de l'architecture. Donnez du code de la meilleure qualité possible.

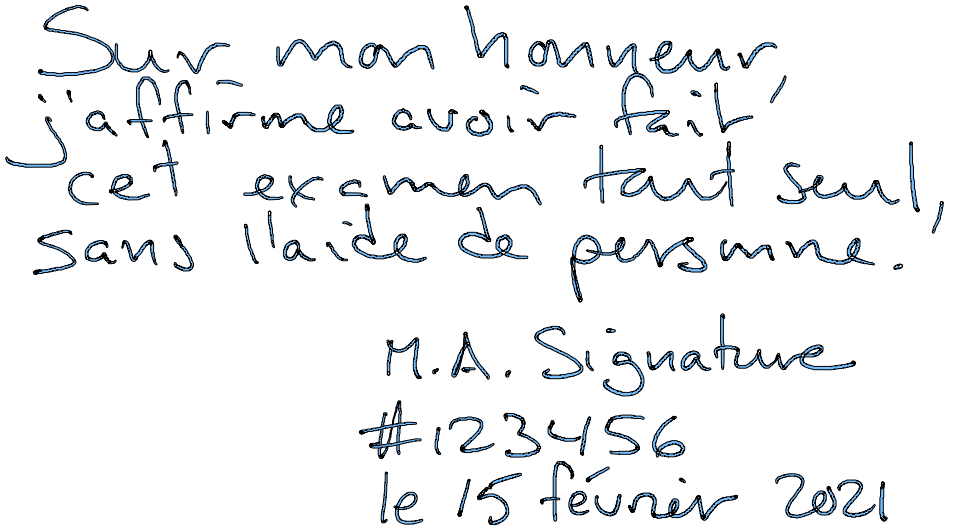
```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity vaccin_qc_priorite is
    generic (N : positive := 2); -- nombre de bits pour exprimer la priorité
    port (
    FR : in std_logic_vector(0 to 2 ** N - 1); -- les facteurs de risque
                                               -- en ordre croissant à partir du bit 0
    priorite : out unsigned(N - 1 downto 0);
    population_generale : out std_logic
    );
end vaccin_qc_priorite;

architecture arch of vaccin_qc_priorite is

-- votre code ici

begin

-- votre code ici

end arch;
```

Question 5. (0 points)

Déclaration sur l'honneur

Produisez une affirmation solennelle sur votre honneur, écrite à la main, signée et datée, à l'effet que vous avez complété cet examen seul/e et sans l'aide de personne.

Joignez une image de votre affirmation en guise de réponse. Voir l'exemple ci-dessous.



Solutions

#1 options d'implémentation

Technologie	# de processeurs	# de puces	coûts pour 28 k systèmes	notes
A. Logique fixe	Fmax = 900 MHz > 750 MHz, donc un seul processeur suffit => 1325 blocs logiques.	Une seule puce suffit, 1325 blocs < 10000	625 k$ + 28 k × 3 $= 709 k$	Problème de délais : on ne peut pas attendre 6 mois Solution pas retenue
B. FPGA de moyenne gamme	Fmax = 550 MHz, donc il faut deux processeurs, => 2650 blocs logiques.	Une seule puce suffit, 2650 blocs < 3000 blocs.	28 k × 140 $ = 3.9 M$	Solution possible Le système tient sur une seule puce.
C. FPGA bas de gamme	Fmax = 225 MHz, donc il faut 750 / 225 = 4 processeurs, => 5300 blocs logiques.	Il faut six puces, sur lesquelles on pourrait répartir les 4 processeurs.	28 k × 25 $ × 6 = 4.2 M$	Solution possible mais pas retenue. Grand risque de complexité à avoir 6 puces différentes pour le même système.

La solution B semble préférable.

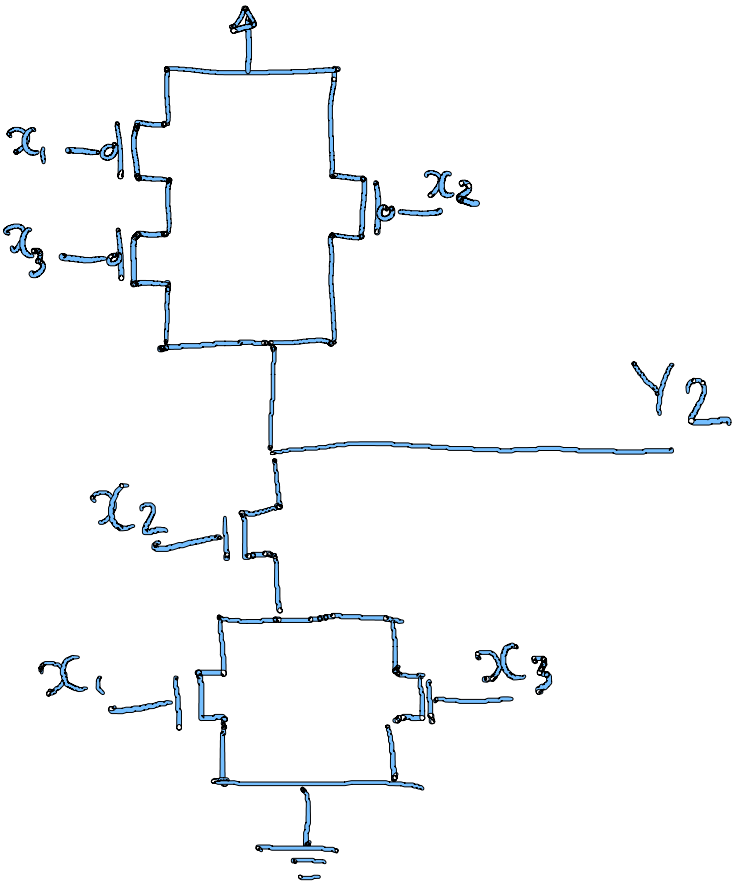
On pourrait aussi prendre la solution B pour la livraison de 25000 items, coûts de 3.5 M$, et la solution A pour les 2000 autres systèmes, coûts de 631 k$, total 4.1 M$, mais ça ne semble pas en valoir la peine ici.

#2

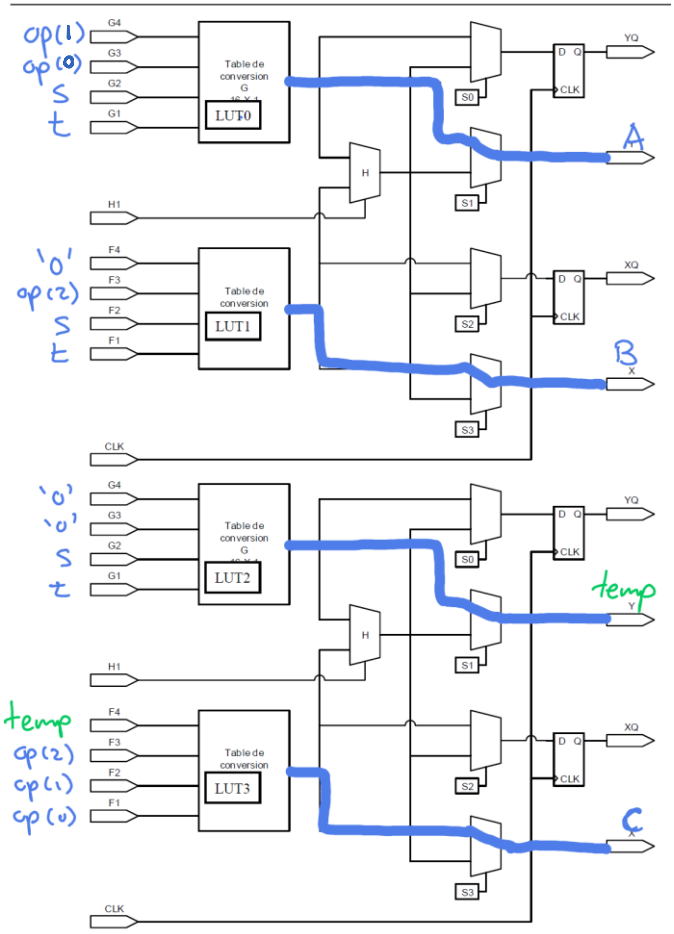
On a y2 = (x1' + x2')(x1 + x2' + x3')
= x1'x1 + x1'x2' + x1'x3' + x1x2' + x2'x2' + x2'x3'
= 0 + x2'(x1' + x1 + 1 + x3') + x1'x3
= x2' + x1'x3'

Donc y2' = (x2' + x1'x3')' = x2(x1 + x3), ce qui donne le circuit suivant.

On peut aussi partir du début d'une autre manière :
y2 = (x1' + x2')(x1 + x2' + x3')
y2' = x1x2 + x1'x2x3
= x2(x1 + x1'x3)
= x2(x1 + x3) [par l'identité A + A'B = A + B]



#3 Une solution possible



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

#4 Solution

Pour N = 2, on obtient de la table de vérité :
population_generale <= non(ou(FR));
priorite(1) = FR(0)'FR(1)'
priorite(0) = FR(0)'FR(1) + FR(0)'FR(1)'FR(2)' = FR(0)'(FR(1) + FR(2)')

On a les codes suivants pour N = 2 et N général.

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

entity vaccin_qc_priorite is
    generic (N : positive := 2); -- nombre de bits pour exprimer la priorité
    port (
    FR : in std_logic_vector(0 to 2 ** N - 1); -- les facteurs de risque, en ordre croissant
à partir du bit 0
    priorite : out unsigned(N - 1 downto 0);
    population_generale : out std_logic
    );
end vaccin_qc_priorite;



architecture arch_N_2 of vaccin_qc_priorite is
begin

    assert N = 2 report "cette architecture ne fonctionne que pour N = 2" severity failure;
    population_generale <= nor(FR);
    priorite(1) <= not(FR(0)) and not(FR(1));
    priorite(0) <= not(FR(0)) and (FR(1) or not(FR(2)));

end arch_N_2;



architecture arch_N of vaccin_qc_priorite is
begin

    process(all)
    begin
        population_generale <= '1';     -- valeur par défaut
        priorite <= (others => '1');    -- valeur par défaut sans importance
        for k in FR'range loop
            if FR(k) = '1' then
                priorite <= to_unsigned(k, priorite'length);
                population_generale <= '0';
                exit;
            end if;
        end loop;
    end process;

end arch_N;
```