Nom : _____ Matricule : _____

INF3500 : Conception et réalisation de systèmes numériques

Examen intra #1 – devoir de rattrapage pédagogique – 21 février 2021

1. Conditions de l'examen

- Ce devoir de rattrapage peut permettre d'augmenter votre note de la moitié des points entre votre note et 10 / 10 : si votre note au contrôle était de 6 / 10, vous pouvez aller chercher jusqu'à 2 points supplémentaires.
- Le devoir doit être remis au plus tard à 14 h 45 le lundi 8 mars. Aucun retard ne sera accepté.
- Le devoir 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.
- Le devoir doit être fait individuellement. Vous ne pouvez pas demander d'aide sur les questions du devoir. Une attestation à cet effet vous sera demandée dans la dernière question.

2. À propos de la remise

- 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.
- 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'automne 2021 et vous travaillez (encore) 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 des variants 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	825 k$	5 $	900 MHz	10000	6 mois
B. FPGA de moyenne gamme	0 $	160 $	550 MHz	3800	2 mois
C. FPGA bas de gamme	0 $	35 $	225 MHz	800	1 mois

Le système doit traiter des données à un débit de 850×10^6 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.

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 combinatoire2b is
    port (
        x1, x2, x3 : in std_logic;
        y2 : out std_logic
    );
end combinatoire2a;

architecture arch of combinatoire2b is

signal p3, p4 : std_logic;

begin

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

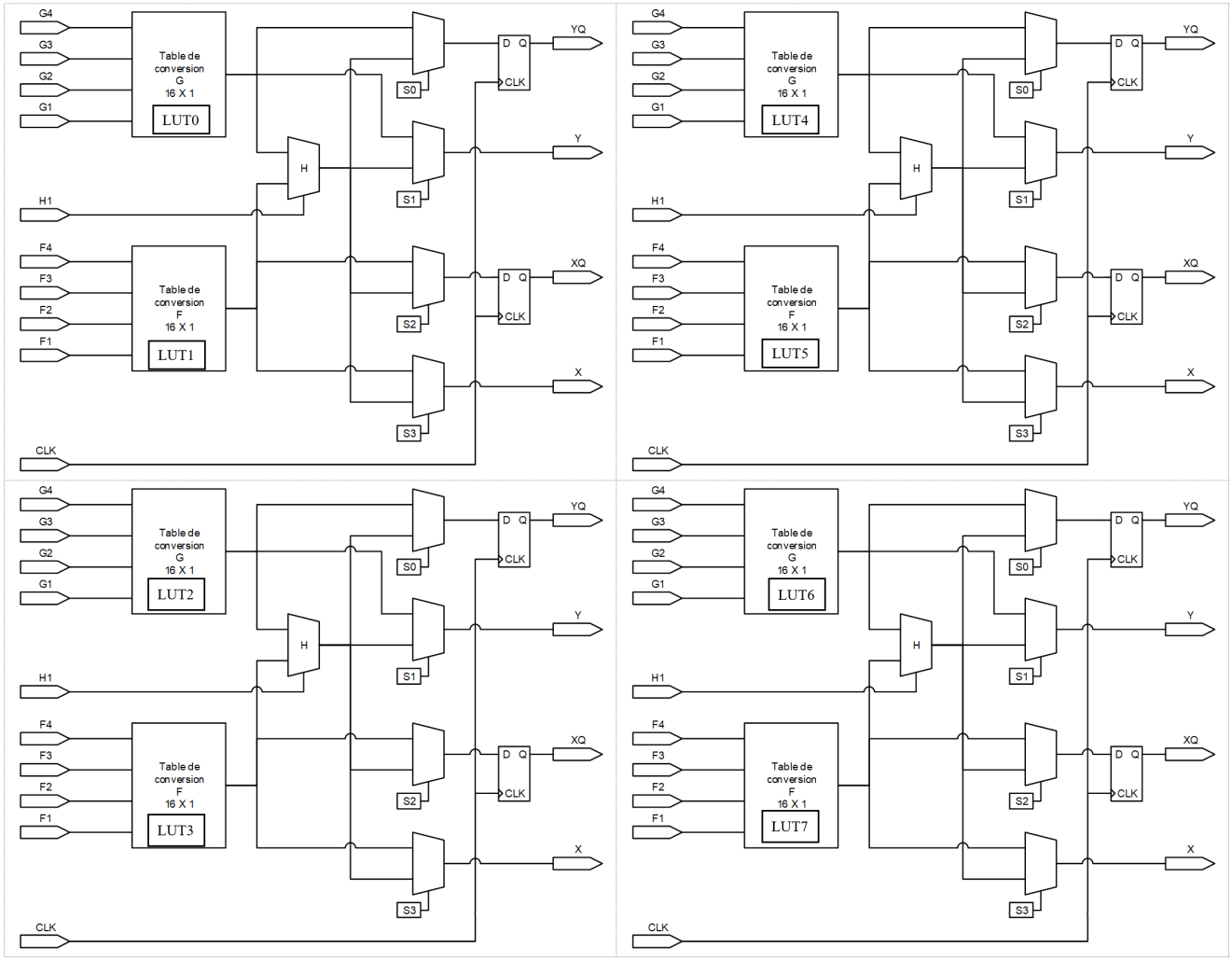
Annotez le diagramme de la page suivante pour montrer comment implémenter le code VHDL combinatoire suivant.

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 combinatoire5b is
  port (
        s, t : in std_logic;
        op : in unsigned(2 downto 0);
        A, B, C : out std_logic
    );
end combinatoire5;

architecture arch of combinatoire5b is
begin

    process(all)
    begin
        case to_integer(op(1 downto 0)) is
            when 0 => A <= s nor t;
            when 1 => A <= s nand t;
            when 2 => A <= s xnor t;
            when others => A <= not(t);
        end case;
    end process;

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

    C <= (s xor t) and (op(2) or op(1) or 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 3 : vivre dans un centre d'hébergement et de soins de longue durée (CHSLD) | priorité "11";
- FR niveau 2 : être un travailleur de la santé | priorité "10";
- FR niveau 1 : être âgé de plus de 60 ans | priorité "01";
- FR niveau 0 : avoir une maladie chronique | priorité "00".

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 1 ("01"). Si elle vit en CHSLD (FR = "1011"), sa priorité serait de 3 ("11"). 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(3)	FR(2)	FR(1)	FR(0)	population_generale	priorite(1)	priorite(0)
0	0	0	0	1	-	-
0	0	0	1	0	0	0
0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	1	0	0	0	1	0
0	1	0	1	0	1	0
0	1	1	0	0	1	0
0	1	1	1	0	1	0
1	0	0	0	0	1	1
1	0	0	1	0	1	1
1	0	1	0	0	1	1
1	0	1	1	0	1	1
1	1	0	0	0	1	1
1	1	0	1	0	1	1
1	1	1	0	0	1	1
1	1	1	1	0	1	1

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.

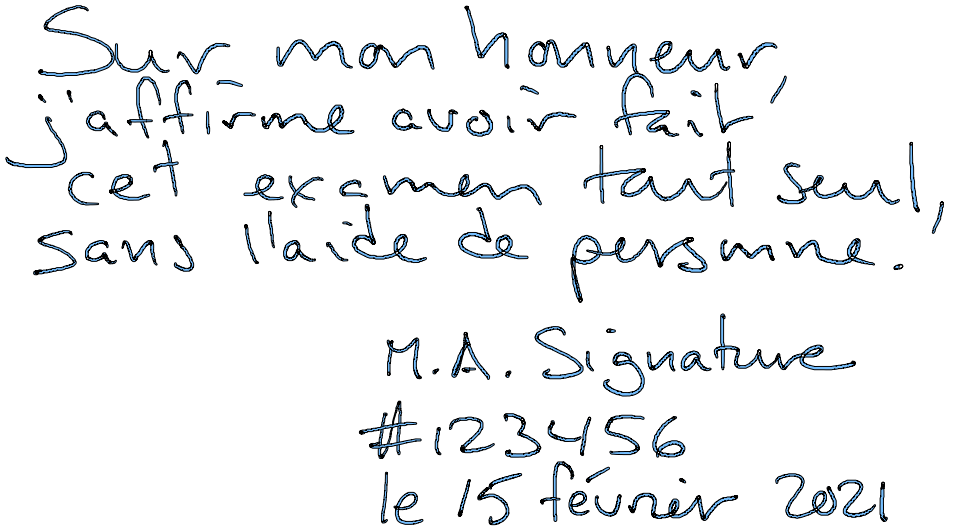
```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity vaccin_qc_priorite_v2 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 décroissant à partir du bit 0
    priorite : out unsigned(N - 1 downto 0);
    population_generale : out std_logic
    );
end vaccin_qc_priorite_v2;

architecture arch of vaccin_qc_priorite_v2 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 ce devoir seul/e et sans l'aide de personne.

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



Solutions

Q1.

Catégorie	A. Logique fixe	B. FPGA moyenne gamme	C. FPGA bas de gamme
# processeurs en parallèle	1 seul (900 MHz > 850 MHz ok)	2 @ 550 MHz chacun = 1100 MHz > 850 MHz	4 @ 225 MHz = 900 MHz > 850 MHz
# puces	1325 blocs 1 seule puce	2650 blocs < 3800 max 1 seule puce	Plafond(5300 blocs / 800 blocs / puce) = 7 puces
Coûts unitaires / système	5 $	160 $	7 × 35 $ = 245 $
Coûts fixes supplémentaires (On suppose que les coûts fixes de développement sont les mêmes pour les trois technologies)	825 k$	0 $	0 $
Coûts totaux	825 k$ + n × 5 $	n × 160 $	n × 245 $
Livraison	6 mois	2 mois	1 mois
Point d'équilibre	n_équilibre = 825000 / (160 – 5) = 5322		

Discussion

Le FPGA bas de gamme n'est vraiment pas intéressant parce qu'il nécessite sept puces par système. Oui on l'obtiendrait plus rapidement, mais la complexité supplémentaire de fabrication (×7) paraît impossible à justifier.

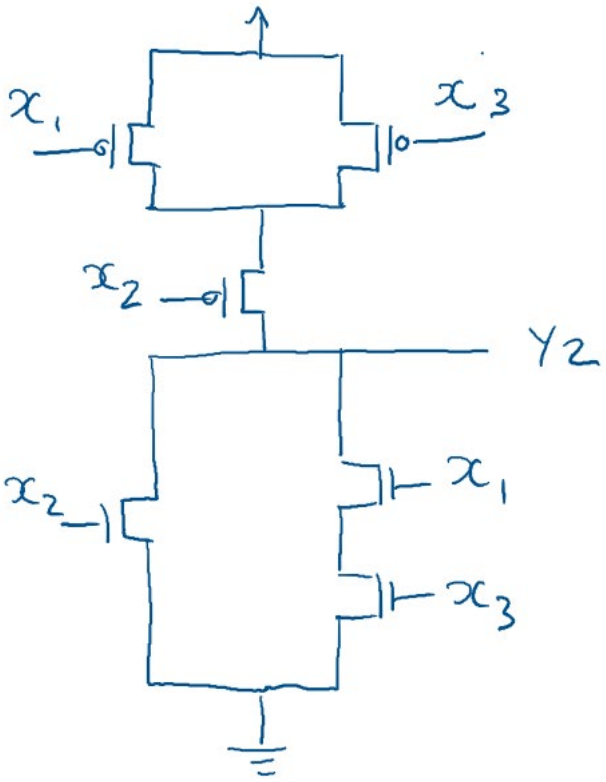
Le nombre de systèmes commandés n'est pas spécifié dans la question. À partir de 5500 systèmes environ, la logique fixe deviendrait avantageuse, du point de vue des coûts, par rapport à la solution FPGA de moyenne gamme. On suppose bien sûr qu'il n'y aurait aucune erreur de design nécessitant une nouvelle commande et de nouveaux frais de 825 k$. En pratique il faudrait une commande d'au moins 7 k, 8 k ou 10 k systèmes pour que le risque en vaille la peine. Il y a environ 1300 hôpitaux au Canada, donc si on pense pouvoir vendre le système à l'échelle planétaire, alors la logique fixe pourrait être intéressante.

On peut se questionner aussi sur les délais. La logique fixe ne serait disponible qu'au printemps 2022. On espère que la pandémie sera sous contrôle à ce moment-là. Si de nouveaux variants entrent en jeu et que la pandémie devait s'étirer sur plusieurs années, alors la logique fixe pourrait être un bon choix. Sinon, le FPGA de moyenne gamme est vraiment préférable.

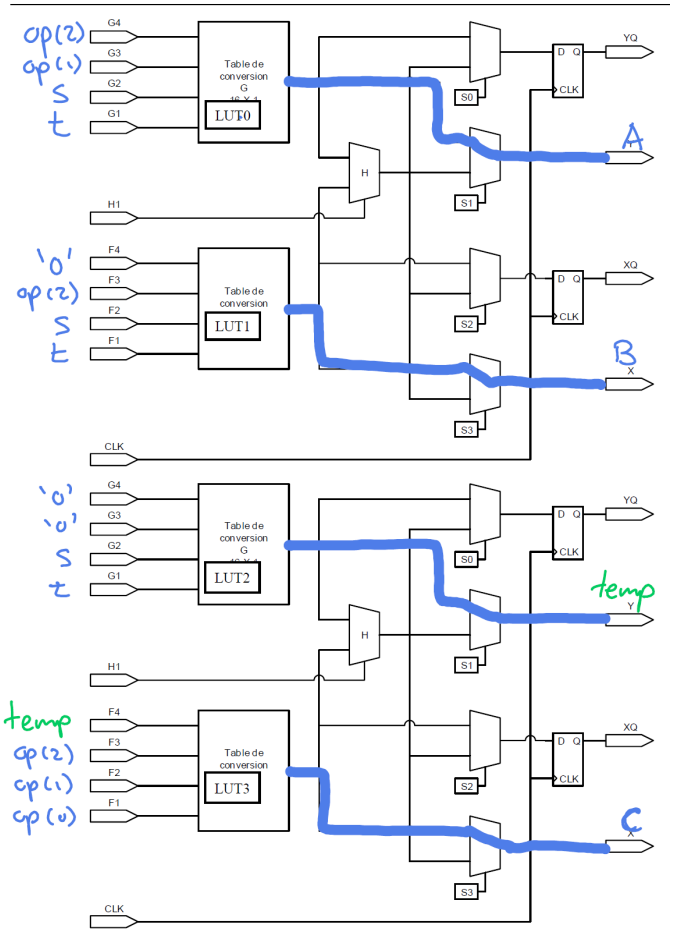
Pour conclure, il apparaît prudent de choisir le FPGA de moyenne gamme pour commencer, par exemple pour quelques centaines ou milliers de systèmes, selon la demande attendue. Si les prévisions de vente sont révisées significativement à la hausse (> 6-7 k systèmes) et que les délais (> 6 mois) sont acceptables, alors la solution en logique fixe serait la meilleure.

Q2.

$$Y_2 = x_1' x_2' + x_1 x_2' x_3'$$
$$= x_2' (x_1' + x_1 x_3')$$
$$= x_2' (x_1' + x_3')$$
$$Y_2' = x_2 + x_1 x_3$$



Q3



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	0	1					0	1	1	0
0	0	1	0					0	1	1	0
0	0	1	1					0	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	-	1	0
1	0	0	0					0	-	-	1
1	0	0	1					1	-	-	0
1	0	1	0					1	-	-	0
1	0	1	1					1	-	-	1
1	1	0	0					1	-	-	1
1	1	0	1					1	-	-	0
1	1	1	0					1	-	-	1
1	1	1	1					1	-	-	0

Q4

```
--------------------------------------------------------------------------------------------------
--
-- INF3500 hiver 2021
-- contrôle #1 - rattrapage
--
-- 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_v2 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
décroissant à partir du bit 0
    priorite : out unsigned(N - 1 downto 0);
    population_generale : out std_logic
    );
end vaccin_qc_priorite_v2;

architecture arch of vaccin_qc_priorite_v2 is

-- votre code ici

begin

-- votre code ici

end arch;


architecture solution_N_2 of vaccin_qc_priorite_v2 is
begin

    assert N = 2 report "cette architecture ne fonctionne que pour N = 2" severity failure;
    population_generale <= nor(FR);
    priorite(1) <= FR(3) or FR(2);
    priorite(0) <= FR(3) or (not(FR(2)) and FR(1));

end solution_N_2 ;

architecture solution_N of vaccin_qc_priorite_v2 is
begin

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

end solution_N;
```

```
--------------------------------------------------------------------------------
--
-- banc d'essai
--
--------------------------------------------------------------------------------

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

entity vaccin_qc_priorite_v2_tb is
    generic (
    N : positive := 3 -- nombre de bits pour exprimer la priorité
    );
end vaccin_qc_priorite_v2_tb;

architecture arch of vaccin_qc_priorite_v2_tb is

signal FR : std_logic_vector(0 to 2 ** N - 1); -- les facteurs de risque, en ordre
décroissant à partir du bit 0
signal priorite : unsigned(N - 1 downto 0);
signal population_generale : std_logic;

-- inversion de l'ordre des bits
-- par Jonathan Bromley, nov. 2003
-- https://groups.google.com/g/comp.lang.vhdl/c/eBZQXrw2Ngk/m/4H7oL8hdHMcJ?pli=1
-- consulté 2021-04-03;
-- édition mineure pour les espacements
function reverse_any_vector (a: in std_logic_vector) return std_logic_vector is
variable result: std_logic_vector(a'RANGE);
alias aa: std_logic_vector(a'REVERSE_RANGE) is a;
begin
    for i in a'RANGE loop
        result(i) := aa(i);
    end loop;
    return result;
end;

begin

--    UUT : entity vaccin_qc_priorite_v2(solution_N_2) generic map (N) port map (FR,
priorite, population_generale);
    UUT : entity vaccin_qc_priorite_v2(solution_N) generic map (N) port map (FR, priorite,
population_generale);
--    UUT : entity vaccin_qc_priorite_v2(arch) generic map (N) port map (FR, priorite,
population_generale);

    process
  constant kmax : integer := 2 ** (2 ** N) - 1;
  begin
    for k in 0 to kmax loop  -- application exhaustive des vecteurs de test
            FR <= reverse_any_vector(std_logic_vector(to_unsigned(k, FR'length)));
            wait for 10 ns;
    end loop;
    report "simulation terminée" severity failure;
  end process;

end arch;
```