Question 1. On souhaite décrire en VHDL une fonction logique dont la table de vérité est donnée ci-dessous. Quelles descriptions VHDL correspondent à cette fonction ?

abc	s
xx0	0
001	0
101	1
011	1
111	1

(a)

```
ENTITY myFunc IS
  PORT(
    a, b, c: IN std_logic;
    s: OUT std_logic
  );
END myFunc;

ARCHITECTURE ar OF myFunc IS
BEGIN
  s <= a OR (b AND c);
END ar;
```

(b)

```
ENTITY myFunc IS
  PORT(
    a, b, c: IN std_logic;
    s: OUT std_logic
  );
END myFunc;

ARCHITECTURE ar OF myFunc IS
BEGIN
  s <= c AND (a OR b);
END ar;
```

(c)

```
ENTITY myFunc IS
  PORT(
    a, b, c: IN std_logic;
    s: OUT std_logic
  );
END myFunc;

ARCHITECTURE ar OF myFunc IS
  SIGNAL abc: std_logic_vector(2 DOWNTO 0);
BEGIN
  abc <= a & b & c;
  WITH abc SELECT
  s <=
    '1' WHEN "011" | "101" | "111",
    '0' WHEN OTHERS;
END ar;
```

(d)

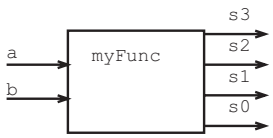
```
ENTITY myFunc IS
  PORT(
    a, b, c: IN std_logic;
    s: OUT std_logic
  );
END myFunc;

ARCHITECTURE ar OF myFunc IS
  SIGNAL ab : std_logic;
BEGIN
  s <= ab AND c;
  ab <= a OR b;
END ar;
```

Question 2. A quelles schémas structurels correspond la déclaration d'entité VHDL ci-dessous ?

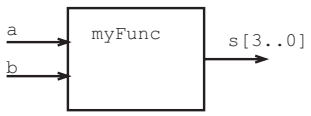
```
ENTITY myFunc IS
  PORT(
    a, b: IN std_logic;
    s: OUT std_logic_vector(3 downto 0)
  );
END myFunc;
```

(a)



(b)

(c)



(d)

(e)

(f)

Question 3. La description VHDL ci-dessous est un décodeur. Parmi les propositions ci-dessous, lesquelles sont vrais ?

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY decoder IS
  PORT(
    a: IN std_logic_vector(2 DOWNTO 0);
    b: OUT std_logic_vector(5 DOWNTO 0)
  );
END decoder;

ARCHITECTURE ar OF decoder IS
BEGIN
  WITH a SELECT
  b <=
    "000001" WHEN "000",
    "000010" WHEN "001",
    "000100" WHEN "010",
    "001000" WHEN "011",
    "010000" WHEN "100",
    "100000" WHEN "101",
    "000000" WHEN OTHERS;
END ar;
```

(a) Le signal *a* représente l'adresse de la sortie active.

(b) Le signal *a* représente le nombre de sorties actives.

(c) L'instruction *with... select* est une instruction concurrente de VHDL.

(d) Si le signal *a* vaut "110", aucune sortie n'est active.

(e) Avec un tel décodeur, il est possible de faire n'importe quel opérateur logique qui comporte au plus 3 entrées et 6 sorties.

(f) Ici, le cas *OTHERS* est obligatoire si l'on veut être certain de fabriquer un circuit combinatoire.

(g) Ici, le cas *OTHERS* est facultatif puisque toutes les combinaisons possibles ont été évoquées.

(h) Il est préférable, mais pas obligatoire, d'utiliser le cas *OTHERS* si l'on veut générer une structure combinatoire.

Question 4. Parmi les propositions ci-dessous qui se rapportent à la description VHDL suivante, lesquelles sont vraies ?

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY myFunc IS
   PORT(
      a: IN std_logic_vector(7 DOWNTO 0);
      b: IN std_logic_vector(2 DOWNTO 0);
      c: OUT std_logic
   );
END myFunc ;

ARCHITECTURE ar OF myFunc IS
BEGIN
   WITH b SELECT
   c <=
      a(7) WHEN "111",
      a(6) WHEN "110",
      a(5) WHEN "101",
      a(4) WHEN "100",
      a(3) WHEN "011",
      a(2) WHEN "010",
      a(1) WHEN "001",
      a(0) WHEN "000",
      '0'  WHEN OTHERS;
END ar;
```

(a) Avec un tel circuit, il est possible de faire n'importe quel opérateur logique qui comporte au plus 3 entrées et une sortie.

(b) Avec un tel circuit, il est possible de faire n'importe quel opérateur logique qui comporte au plus 8 entrées et une sortie.

(c) La description ci-dessus est un multiplexeur 8 vers 1.

(d) La description ci-dessus est un décodeur 1 parmi 8.

(e) La description ci-dessus est un décodeur 3 parmi 8.

(f) La description ci-dessus est un multiplexeur 3 vers 1.

(g) Le cas *OTHERS* n'a aucune chance de se présenter ici, on aurait pu l'enlever.

(h) Même si le cas OTHERS n'a aucune chance de se présenter, il ne faut pas le supprimer sinon la description ne correspond plus à un circuit combinatoire.

(i) Le signal *a* correspond à l'adresse de l'entrée sélectionnée pour être en sortie.

(j) Le signal *b* correspond à l'adresse de l'entrée sélectionnée pour être en sortie.