ENIG

GCR2
2022/2023

Systèmes Electroniques Embarqués

Série de TD N°2

Exercice 1 :

1) Quelles sont les familles des PLD.
2) Quelles sont les critères de choix d'un PLD.
3) De quoi est constitué une PAL.
4) Quelles sont les types des blocs logiques dans les FPGA.

Exercice 2 :

1) On souhaite décrire en VHDL une fonction logique dont la table de vérité est donnée cidessous. 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;

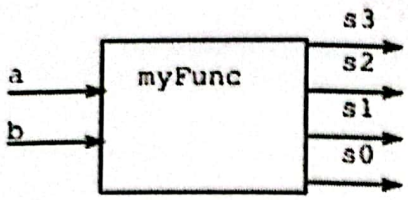
ARCHITECTURE ar OF myFunc IS
  SIGNAL ab : std_logic;
BEGIN
  s <= ab AND c;
  ab <= a OR b;
END ar;
```

2) A quels schémas structurels correspond la déclaration d'entité VHDL ci-dessous ?

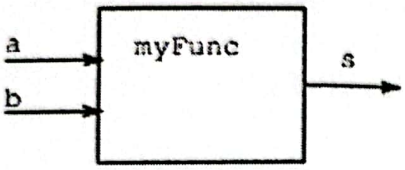
CamScanner

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

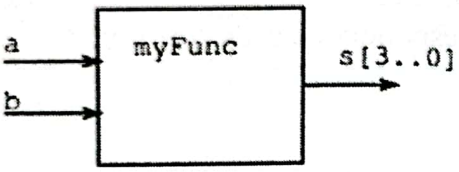
(a)



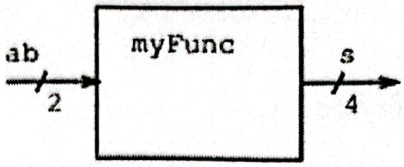
(b)



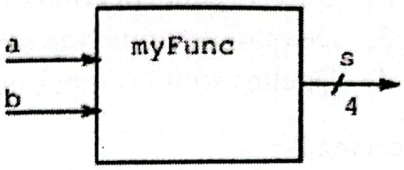
(c)



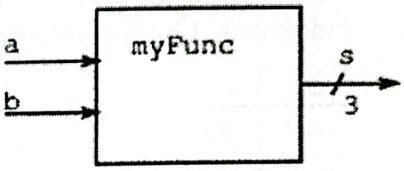
(d)



(e)



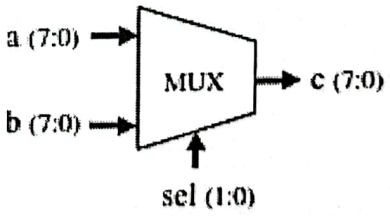
(f)



Exercice 3 :

Ecrire un programme VHDL conformément à :

```
5  ENTITY mux IS
6     PORT ( __ , __ : ___ STD_LOGIC_VECTOR (7 DOWNTO 0);
7            sel : IN ____________________________ ;
8            ___ : OUT STD_LOGIC_VECTOR (7 DOWNTO 0));
9  END _____ ;
10 ---------------------------------------
11 ARCHITECTURE example OF _____IS
12 BEGIN
13    PROCESS (a, b, ____ )
14    BEGIN
15       IF (sel = "00") THEN
16          c <= "00000000";
17       ELSIF (___________) THEN
18          c <= a;
19       _____ (sel = "10") THEN
20          c <= __;
21       ELSE
22          c <= (OTHERS => '__');
23       END ___ ;
24    END __________ ;
25 END __________ ;
26 ---------------------------------------
```



sel	c
00	0
01	a
10	b
11	Z

CamScanner

Exercice 4 :

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;

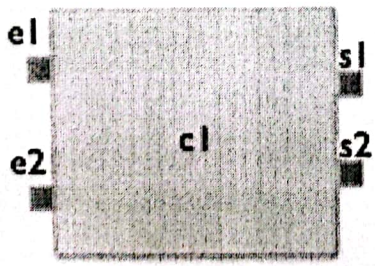
ARCHITECTURE ar OF decoder IS
BEGIN
  WITH a SELECT
  b <=
    "000001" WHEN "000",
    "000010" WHEN "001",
    "000100" WHEN "010",
    "001000" WHEN "011",
    "010000" WHEN "100",
    "100000" WHEN "101",
    "000000" WHEN OTHERS;
END ar;
```

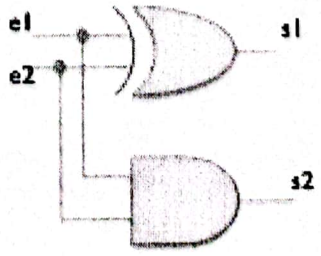
a) Le signal *a* représente l'adresse de la sortie active.
b) Le signal *a* représente le nombre de sorties actives.
c) L'instruction *with... select* est une instruction concurrente de VHDL.
d) Si le signal *a* vaut "110", aucune sortie n'est active.
e) Avec un tel décodeur, il est possible de faire 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.

©

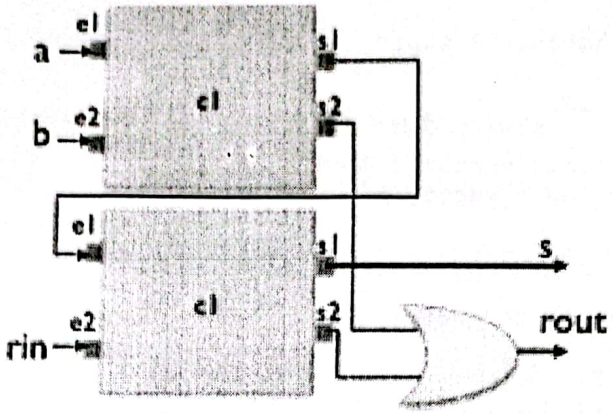
Exercice 5 :

1) Ecrire un programme VHDL réalisant le fonctionnement du composant C1 du circuit de la figure suivante :



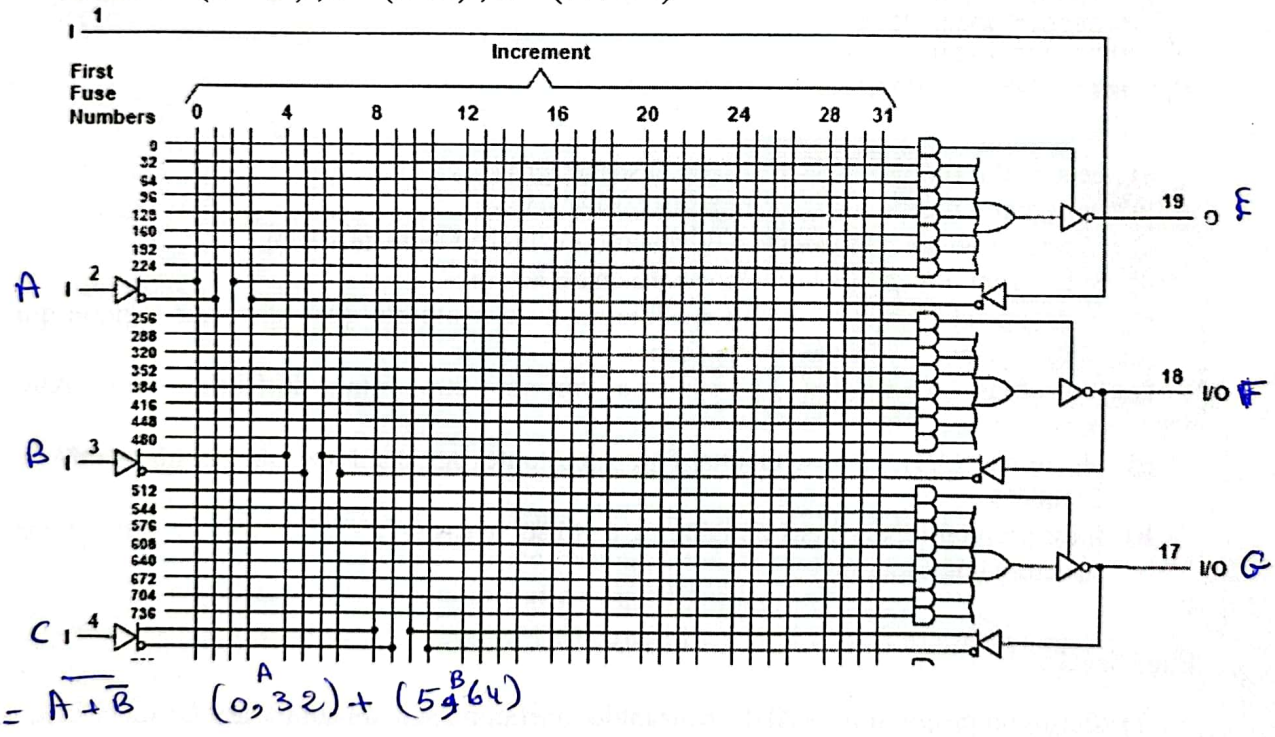


1) Ecrire un programme VHDL réalisant le fonctionnement du circuit de la figure suivante :



Exercice 6 :

Montrez comment implémenter les fonctions logiques suivantes sur l'extrait de circuit PAL donné. $E = (\overline{A + B})$; $F = (\overline{ABC})$; $G = (\overline{A + BC})$



$E = \overline{A + B}$ (0, 32) + (5, 64)

GCR2
2022/2023

Systèmes Electroniques Embarqués

Série de TD N°2

Exercice 1 :

1) Quelles sont les familles des PLD.
2) Quelles sont les critères de choix d'un PLD.
3) De quoi est constitué une PAL.
4) Quelles sont les types des blocs logiques dans les FPGA.

Exercice 2 :

1) On souhaite décrire en VHDL une fonction logique dont la table de vérité est donnée cidessous. 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)

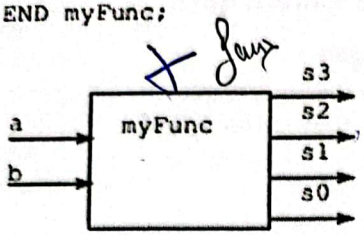
```
ENTITY myFunc IS
  PORT(
    a, b, c: IN std_logic;
    s: OUT std_logic
  );
END myFunc;

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

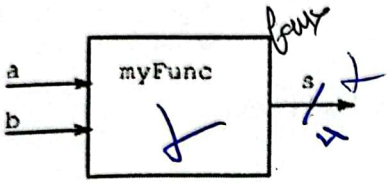
2) A quels schémas structurels correspond la déclaration d'entité VHDL ci-dessous ?

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

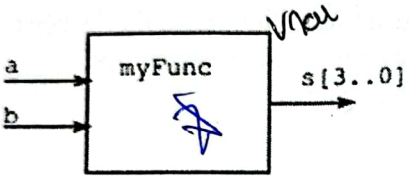
(a)



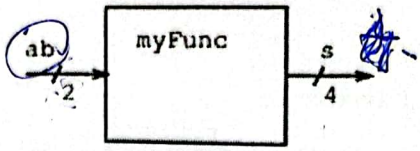
(b)



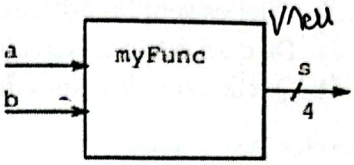
(c)



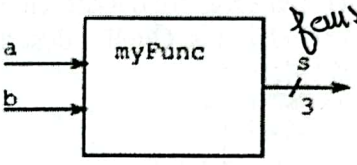
(d)



(e)



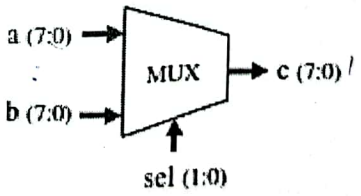
(f)



Exercice 3 :

Ecrire un programme VHDL conformément à :

```
5  ENTITY mux IS
6     PORT ( a , b : IN STD_LOGIC_VECTOR (7 DOWNTO 0);
7            sel : IN std_logic_vector(1 downto 0)
8            c : OUT STD_LOGIC_VECTOR (7 DOWNTO 0));
9  END mux;
10 ----------------------------------------
11 ARCHITECTURE example OF mux IS
12 BEGIN
13    PROCESS (a, b, sel)
14    BEGIN
15       IF (sel = "00") THEN
16          c <= "00000000";
17       ELSIF (sel="01") THEN
18          c <= a;
19       elseif (sel = "10") THEN
20          c <= b;
21       ELSE
22          c <= (OTHERS => 'Z');
23       END IF ;
24    END Process ;
25 END example ;
26 ----------------------------------------
```



sel	c
00	0
01	a
10	b
11	Z

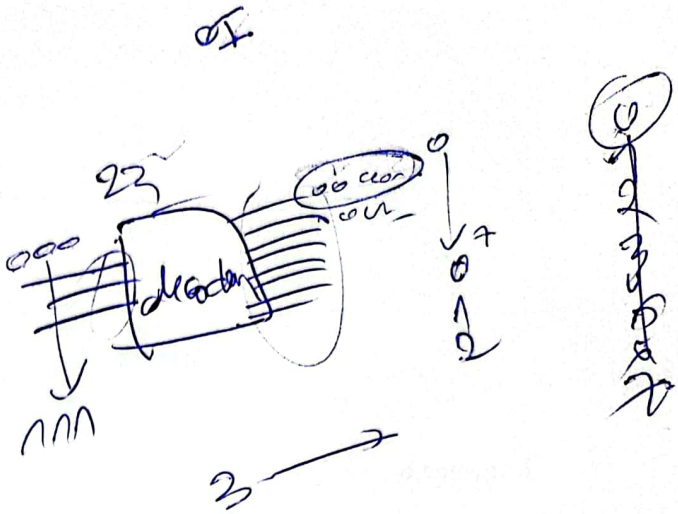
Exercice 4 :

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.

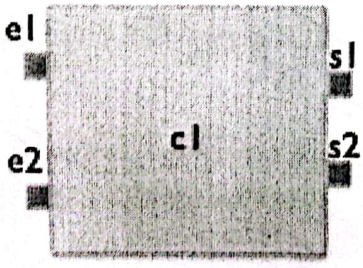
g) Ici, le cas *OTHERS* est facultatif puisque toutes les combinaisons possibles ont été évoquées.

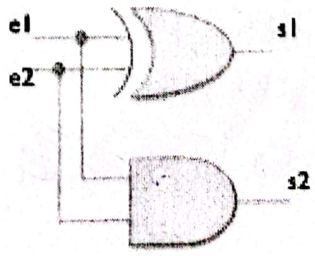
h) Il est préférable, mais pas obligatoire, d'utiliser le cas *OTHERS* si l'on veut générer une structure combinatoire.

©

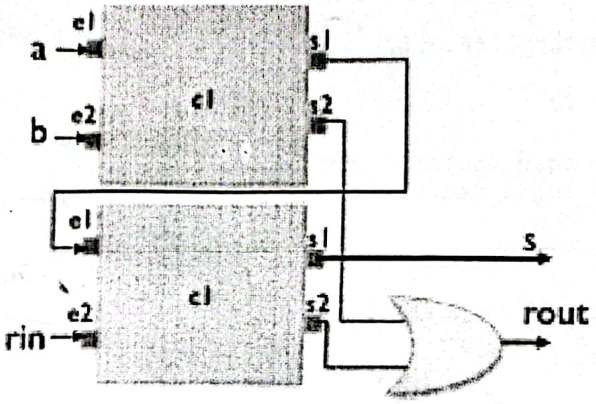
Exercice 5 :

1) Ecrire un programme VHDL réalisant le fonctionnement du composant C1 du circuit de la figure suivante :



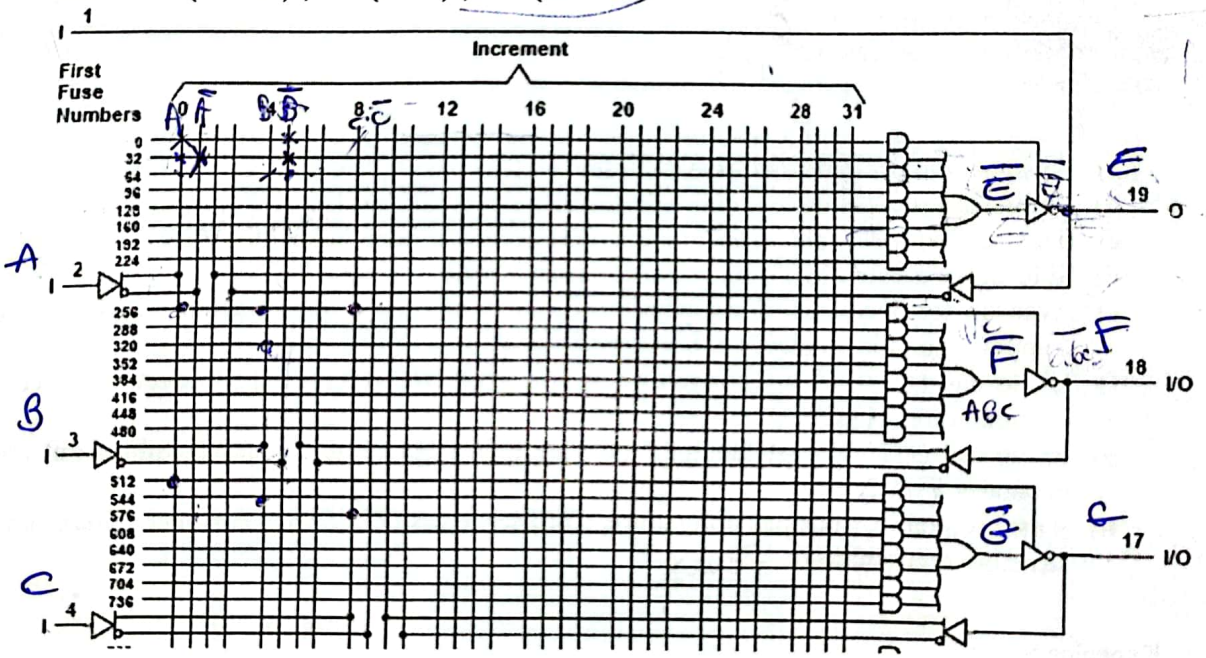


1) Ecrire un programme VHDL réalisant le fonctionnement du circuit de la figure suivante :



Exercice 6 :

Montrez comment implémenter les fonctions logiques suivantes sur l'extrait de circuit PAL donné. $E = \overline{(A + B')}$; $F = \overline{(ABC)}$; $G = \overline{(A + BC)}$



$\overline{E} = \overline{(A+B)} = \overline{A} \cdot \overline{B}$

$A = (256, 0)$

$B = (256, 4)$

$C = (256, 8)$

$\overline{A + \overline{B} + \overline{C}}$:

$G = \overline{ABC} = \overline{A + B + C}$

$\overline{E} = \overline{ABC} = \overline{A + B + C}$

$\overline{A +}$

$E = (A + B')' \rightsquigarrow A = (0, 0)$, $\overline{B} = (2, 6)$.

$\Rightarrow E = (A + B')'$

GCR2
2022/2023

Systèmes Electroniques Embarqués

Série de TD N°2

Exercice 1 :

1) Quelles sont les familles des PLD. PAL / GAL / EPLD / FPGA
2) Quelles sont les critères de choix d'un PLD. Prix / vitesse de fonct / nombre de portes / consommat°
3) De quoi est constitué une PAL. zone de fusibilité / desentrée / structure de sortie
4) Quelles sont les types des blocs logiques dans les FPGA. ~~matrice de~~ Macrocellule / Bloc à multiplexage / cellule symétrique les LUT

Exercice 2 :

1) On souhaite décrire en VHDL une fonction logique dont la table de vérité est donnée cidessous. Quelles descriptions VHDL correspondent à cette fonction ?

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

A ou B ifsift J1 — A et B x0 = 0

(a) Faux

```
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) Vrai

```
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) Vrai

```
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) Faux

```
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;
```

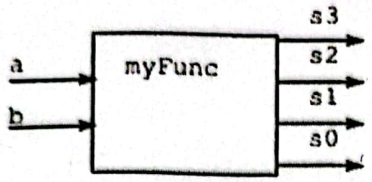
par défaut ab = 0

2) A quels schémas structurels correspond la déclaration d'entité VHDL ci-dessous ?

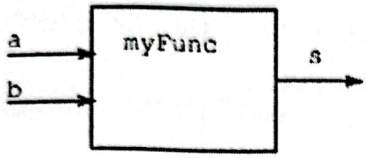
CamScanner

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

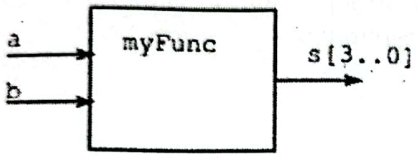
(a) Vrai



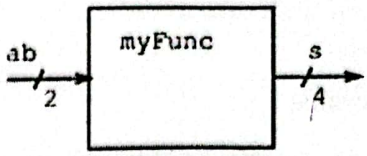
(b) Faux



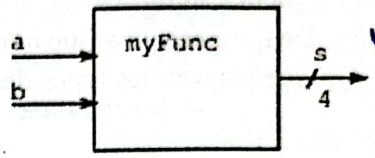
(c) Vrai



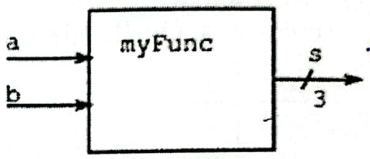
(d) F-



(e) Vrai



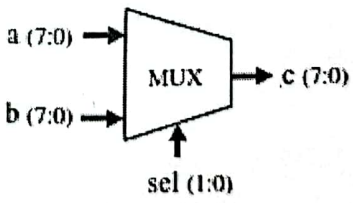
(f) faux



Exercice 3 :

Ecrire un programme VHDL conformément à :

```
5  ENTITY mux IS
6     PORT ( a , b : IN STD_LOGIC_VECTOR (7 DOWNTO 0);
7            sel : IN STD_LOGIC_VECTOR (1 downto 0)
8            c : OUT STD_LOGIC_VECTOR (7 DOWNTO 0));
9  END mux ;
10 ----------------------------------------
11 ARCHITECTURE example OF mux IS
12 BEGIN
13    PROCESS (a, b, sel )
14    BEGIN
15       IF (sel = "00") THEN
16          c <= "00000000";
17       ELSIF (sel="01" ) THEN
18          c <= a;
19       elsif (sel = "10") THEN
20          c <= b;
21       ELSE
22          c <= (OTHERS => 'Z');
23       END if ;
24    END process ;
25 END example ;
26 ----------------------------------------
```



sel	c
00	0
01	a
10	b
11	Z

CamScanner

Exercice 4 :

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;
```

6
7

a) Le signal *a* représente l'adresse de la sortie active. Faux (nbre de sortie active)
b) Le signal *a* représente le nombre de sorties actives. oui FAUX
c) L'instruction *with... select* est une instruction concurrente de VHDL. oui
d) Si le signal *a* vaut "110", aucune sortie n'est active. ~~Faux~~ Vrai
e) Avec un tel décodeur, il est possible de faire n'importe quel opérateur logique qui comporte au plus 3 entrées et 6 sorties. Faux car 6 et 7 donne pas valeur
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. F
h) Il est préférable, mais pas obligatoire, d'utiliser le cas *OTHERS* si l'on veut générer une structure combinatoire. F

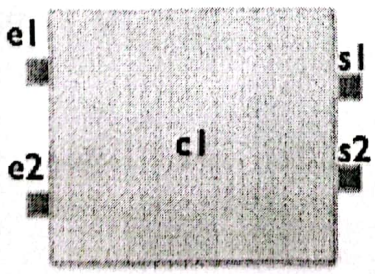
©

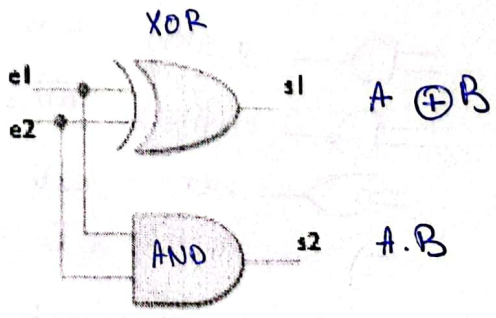
Exercice 5 :

1) Ecrire un programme VHDL réalisant le fonctionnement du composant C1 du circuit de la figure suivante : Demi-additionneur

fait la somme de 2 nbs binaires sur un bit chacun et donne de sortie

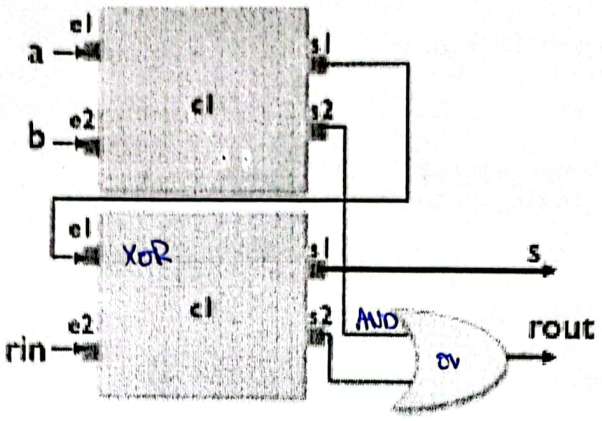
e_1	e_2	s_2	s_1
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0





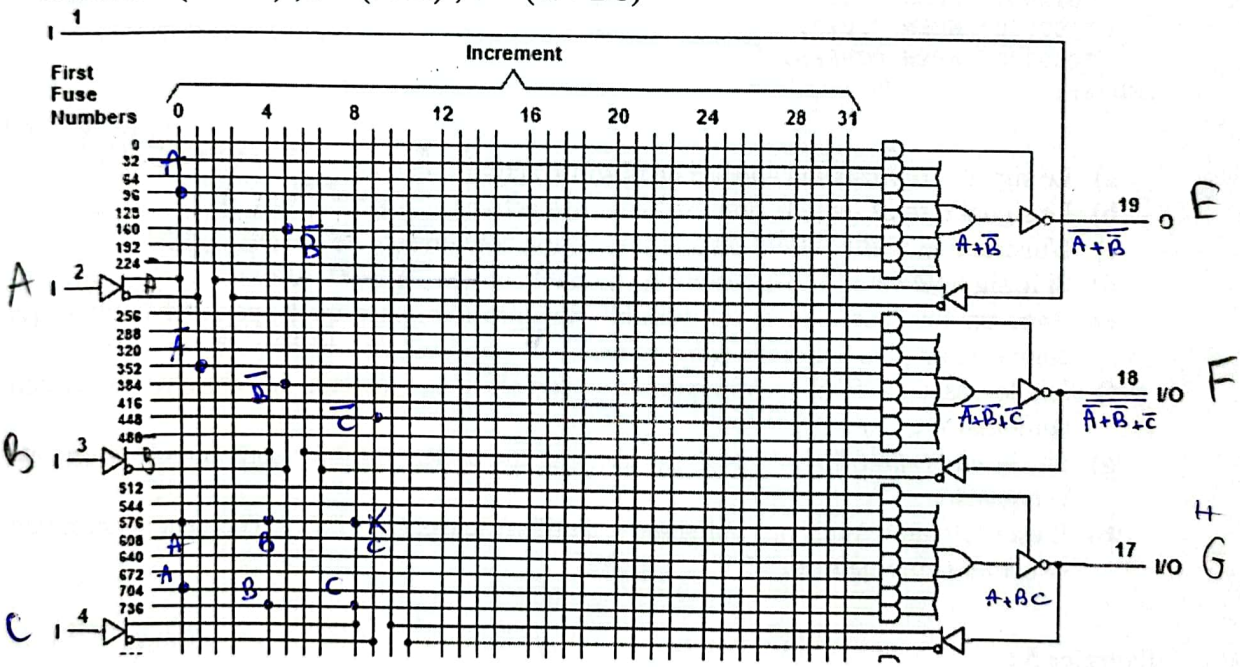
CamScanner

1) Ecrire un programme VHDL réalisant le fonctionnement du circuit de la figure suivante :



Exercice 6 :

Montrez comment implémenter les fonctions logiques suivantes sur l'extrait de circuit PAL donné. E = (A + B')'; F = (ABC)'; G = (A + BC)'



: mot $\bar{a}$

: AND $a.b$

: NAND $\overline{ab} = \bar{a} + \bar{b}$

: or $a+b$

: NOR $\bar{a}.\bar{b}$

: XOR $a \oplus b$

$F = \overline{\bar{A} + \bar{B} + \bar{C}}$

$H = ABC$

TD2

Ex1

1/ - PAL
- GAL
- EPLD
- FPGA

2/ - Vitesse de fonctionnement
- nbre de portes
- Consommation
- Technologie
- Prix

3/ Matricielle
- Bloc { matrice de fonctionnement / structure de sortie

Les signaux { - les entrée / les sorties
- clock / Reset

4/ - Macro cellule
- Bloc à Multiplexage
- cellule symetrique
- les LUT

Ex2

1/

EX 5 :

1) Library ieee;
use ieee.std_logic_1164.ALL;)*

Entity halfadder is Port (e1, e2 : IN std_logic; S1, S2 : out std_logic);
END halfadder

Architecture an of halfadder is begin
S_1 <= (e_1 XOR e_2)
S_2 <= (e_1 AND e_2)
END an

2) *
Entity fulladder is Port (a, b, rin : IN std_logic ; s, rout : out std_logic;)
End fulladder

Architecture behavior of fulladder is
begin
S <= (a xor b) xor rin
rout <= (a AND b) OR (rin AND (a xor b))
End behavior

autre_arc :
Architect Structure of fulladder is
signal : S_1 - sum
S_2 - sum
retenue R_1, R_2
begin
halfadder1 : Entity component
Port MAP : (a, b, S_1 - sum, R_1);
half adder 2 : Entity component
Port MAP (S_1 - sum, S_2 - sum, R_1, R_2)

CamScanner

nout < = R_1 ∂ R_2
End Structure

```
int vin ;
float temperature ;
include <liquide_crystal.h> ;
liquidcrystal.lcd( 12, 11, 5, 4, 3, 2) ;

void setup() {
lcd.begin( 16,2)
Lcd.print("temperature :") ;
}

vin = analog read( A0) ;
temperature =
LCD. setCursor( 3,1)
LCD.print(" Temperature") ;
LCD.print ("celcius") ;
LCD.print ( temperature ) ;
```

PLD : pgmable logical devices

→ circuit logique pgmable ± cplexe

Remplace l'associat° de plusieurs boites logique

↳ Simplificat° de cablage

↳ cout reduit

↳ facile à utilisé

↳ risque de panne

Time To Market

Cahier de charge → cout produc°

→ performance

→ consommat°

→ integrat°

→ evolut° de produ

Alternative → ASIC (App Specific Integrated Circuit)

* Choix de fonderie
* conceptº du circuit
* integrat°
* circuit dédié à 1 app
→ Haute performance

→ Circuit à R/W pgmable

* integrat° limitée
→ circuit pour 1 app
↳ Technologie figée / non reprogrammable

Avantage PLD → facilite pgmat°

→ Cout et temps ↘

→ prise de risque ↘

→ prototypage =

→ Support logiciel

Consommat° →

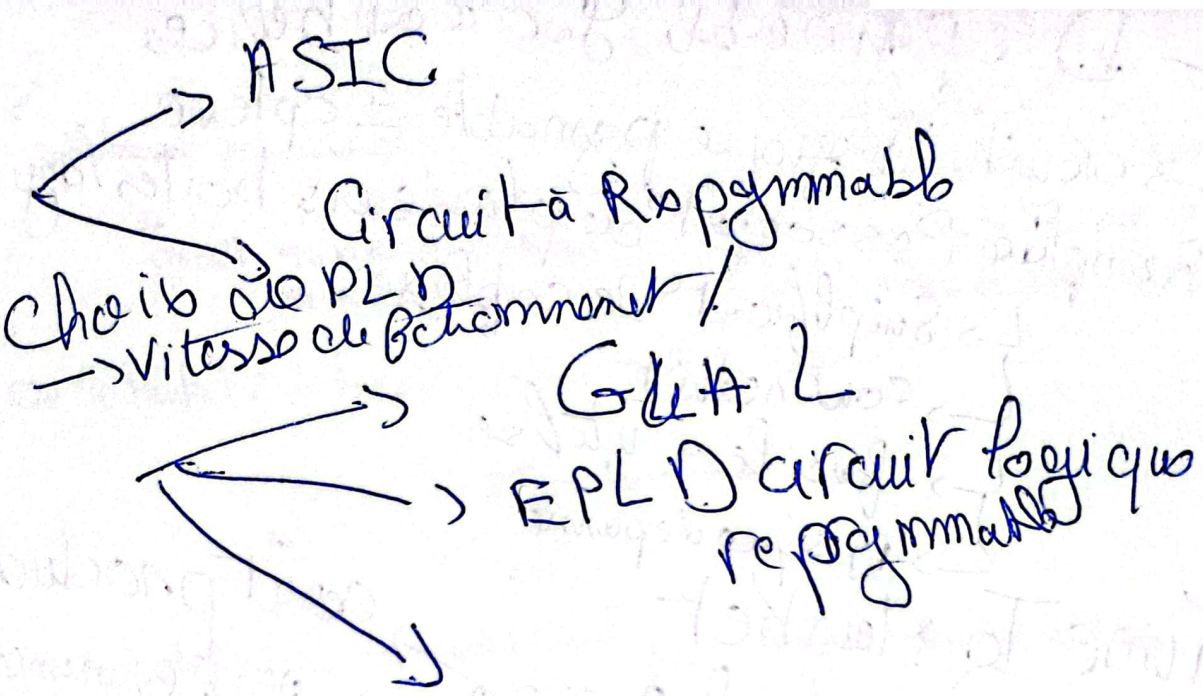
prix →

→ PAL

→ GAL

→ EPLD

→ FPGA


ASIC
Circuit à Rpgmmabb
Choix de PLD
→ Vitesse de fonctionnement
GAL
EPLD circuit logique
reprogrammable

PLD :

↓

circuits logique programmables

⇓

circuit qui associe plusieurs boites logiques.

+ simplificat° de cablage
+ diminit° des coûts
+ Encombrement réduit
+ Prise de risque Reduit.

Time To Market : le temps de fabrication et du vente

Type des circuits

ASIC

Application specific Integrated circuit :

- réalise une seule fct bien spécifique (Non programmable)

PLD

- integrat° limitée < ASIC
- un circuit programmé pour une app
- niveau d'intégration moindre aux ASIC
- consommation plus élevé
- prix élevés, faible disponibilité

Les familles des PLD

PAL :

Programmable Array Logic

GAL

Generic Array logic : PAL effaçable Electriquement

EPLD

Enasable programmable : de grande capacité

FPGA

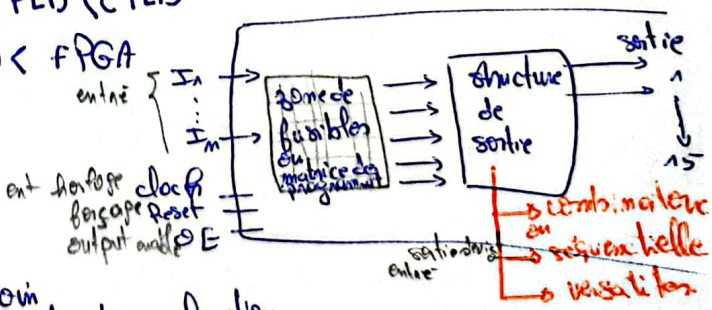
forecasting Programmable Gate Array

⇓

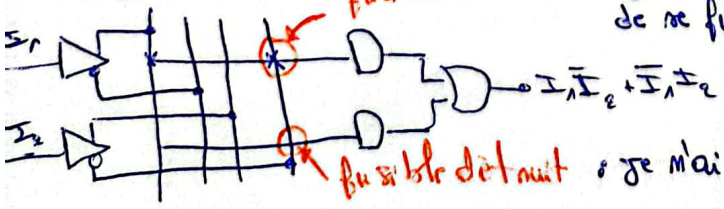
ouvrir des nouvelles voies de recherche : des méthodologie de conception des syst reconfigurables

+ Critère de choix des PLD :

- vitesse de fonctionnement : HSI FPGA < PLD < CPLD
- Nombre de portes : HSI < PLD < CPLD < FPGA
- Consommation
- technologie
- Prix.



Technique de PAL : fusible intact : j'ai besoin de ce fusible dans ma fonction



fusible détruit : je n'ai pas besoin