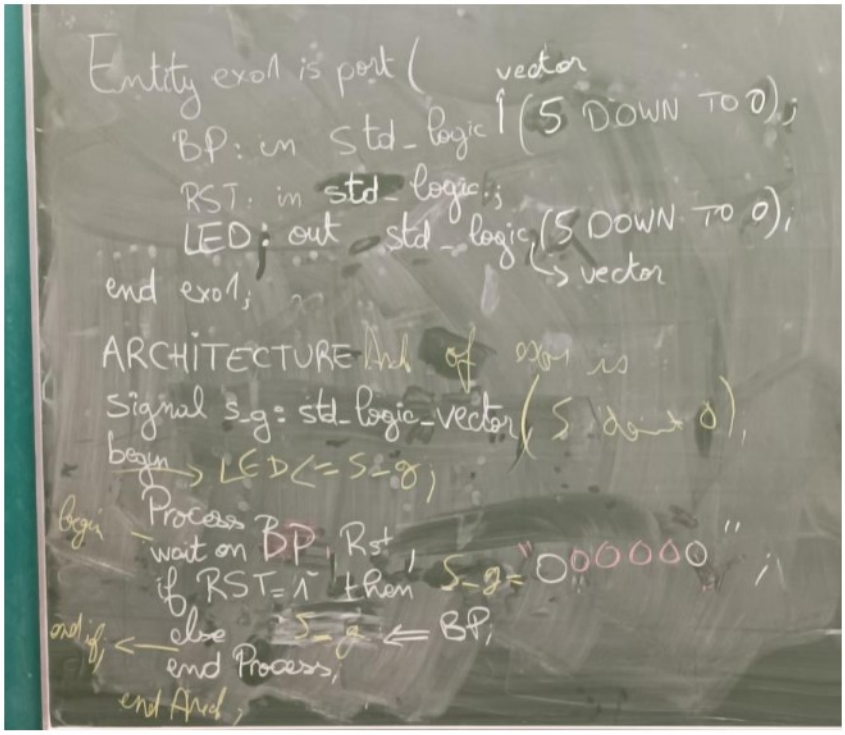
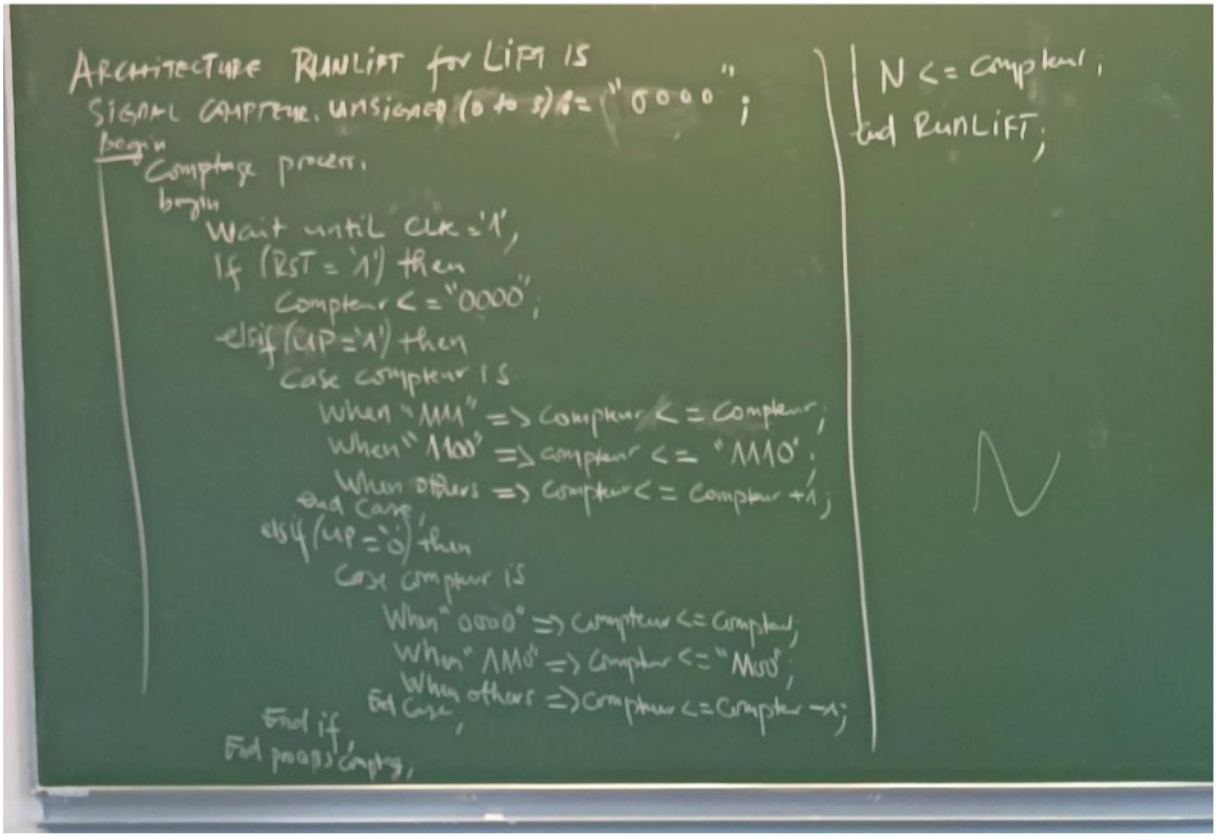
Correction du TD 3 VHDL

Correction Exercice 1 :



Correction Exercice 2 :



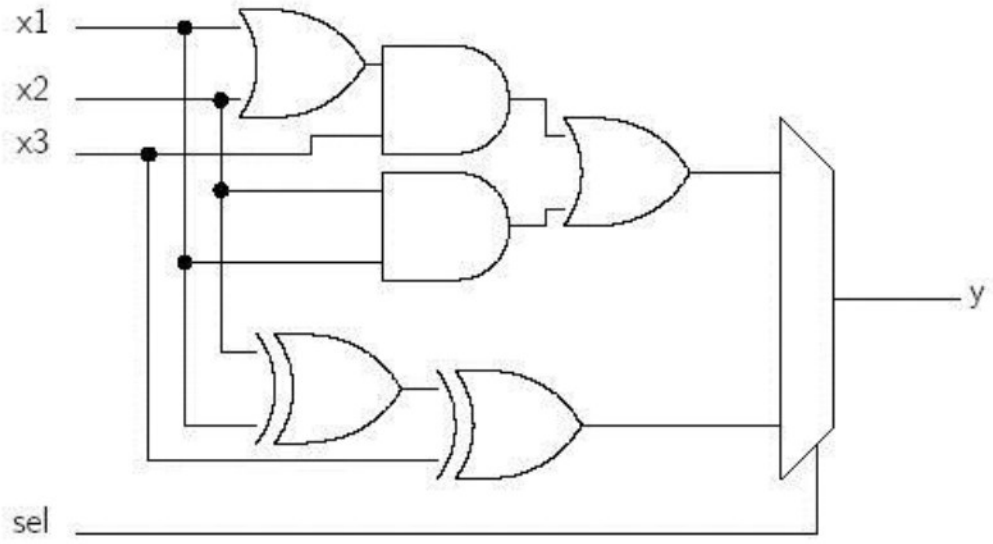
Correction du TD 4 VHDL

Correction Exercice 1 :

```
Library ieee;
use ieee.std_logic_1164.all;
entity comparateur8bits is port
        (A, B: in std_logic_vector (7 downto 0);
        EQUA, SUPE, INFE: out std_logic);
end comparateur8bits;

architecture ARCH_comparateur8bits of comparateur8bits is
Begin
EQUA<='1' when A=B else '0';
SUPE<='1' when A>B else '0';
INFE<='1' when A<B else '0';
end ARCH_comparateur8bits;
```

Correction Exercice 2 :

1- Le schéma correspondant



2- C'est une description comportementale, puisqu'il n'y a aucune composante.
C'est un processus combinatoire car il inclut toutes les entrées (signaux lus) dans la liste de sensitivité. De plus, toutes les valeurs possibles de sel sont traitées.

3 - C'est un plein additionneur avec multiplexage des sorties somme et retenue.

4-
```
architecture archi2 of exercice2 is
signal a, b, c, d, e, f: std ulogic;
signal ytemp: std logic;
begin
a <= x1 or x3;
```

```
b <= x1 and x3;
c <= x2 and a;
d <= b or c;
e <= x1 xor x2;
f <= x3 xor e;
P1: process (d, f, sel)
begin
if sel='0' then
ytemp <= d;
else
ytemp <= 'z';
end if;
end process P1;
P2: process (d, f, sel)
begin
if sel='1' then
ytemp <= f;
else
ytemp <= 'z';
end if;
end process P2;
y <= ytemp;
end architecture archi;
```

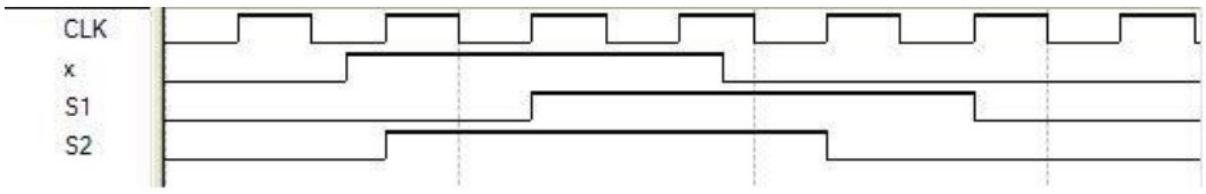
Correction Exercice 3 :

```
Library ieee;
use ieee.std_logic_1164.all;
-- entity statement
Entity Moore_Machine is port
(x, CLK : in  BIT; S1,S2 : out BIT);
End Moore_Machine;
-- architecture statement

Architecture FSM of  Moore_Machine is
type STATE_TYPE  is (Etat0, Etat1, Etat2, Etat3);
signal STATE : STATE_TYPE := Etat0;
signal S :bit_vector (1 downto 0);
begin
--block funtion
NEXT_STATE: block (CLK'event and CLK='1')
begin
-- guarded conditional concurrent signal assignment statement
STATE <= guarded Etat0 when (STATE=Etat0 and x='0') else
                                Etat1 when (STATE=Etat0 and x='1') else
                                Etat2 when (STATE=Etat1 and x='1') else
                                Etat3 when (STATE=Etat2 and x='0') else
                                Etat2 when (STATE=Etat2 and x='1') else
                                Etat0 when (STATE=Etat3 and x='0') else
            Etat1 when (STATE=Etat3 and x='1') else
```

```
            Etat3 when (STATE=Etat1 and x='0') else
            STATE;
end block NEXT_STATE;
-- unguarded selected concurrent signal assignment statement

with STATE select
S <= "00" when Etat0,
    "10" when Etat1,
    "11" when Etat2,
    "01" when Etat3,
    "00" when others;

S1<=S(0);
S2<=S(1);
End FSM;
```

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

--library SYNTH ;
--use SYNTH.vhdlsynth.all ;

----- Définition de l'entité----
ENTITY diagramme IS
PORT (raz, clk :IN STD_LOGIC;  -- Ne pas oublier remise à 0 et horloge !
                x :IN bit;
               S1,S2 :OUT STD_LOGIC);
END diagramme;

------ Définir L'architecture de description d'état ----------------
ARCHITECTURE arch_diagramme OF diagramme IS
        TYPE etat_4 IS (A, B, C, D);      --définir une liste des états...
  SIGNAL etat,etat_suiv :etat_4 := A;  --et 2 signaux: états courant et –-suivant,
contenant la valeur d'un état de la liste (initialisée à M0) .
  signal S :std_logic_vector (1 downto 0);
BEGIN
        definir_etat:PROCESS( raz, clk) -- "definir_etat":label optionnel
                BEGIN
                        If raz = '1' THEN
                                etat <= A;                  --Le PACKAGE std_logic_1164
                        ELSIF rising_edge(clk) THEN          --en permet l'utilisation.
                            etat <= etat_suiv ;
                        END IF;                                      --Mise à jour de la
variable
        END PROCESS definir_etat;

---------- Définir les états des sorties----------
```

```
sorties : process (etat, x)          --Le PROCESS doit contenir une...
        BEGIN                                          --structure de CAS unique
dépendant
                    --de la variable d'état.
CASE etat IS
  --Le signal de l'état suivant n'est
--attribué que dans la structure CASE

                    WHEN A => S <= "00";

                    If x ='1' then etat_suiv <= B;
                    elsif x ='0' then etat_suiv <= A;
                    end if;

                    WHEN B => S <= "10";

                    If x ='1' then etat_suiv <= C;
                    elsif x ='0' then etat_suiv <= D;
                    end if;

                    WHEN C => S <= "11";

                    If x ='1' then etat_suiv <= C;
                    elsif x ='0' then etat_suiv <= D;
                    end if;

                    WHEN D => S <= "01";

                    If x ='1' then etat_suiv <= B;
                    elsif x ='0' then etat_suiv <= A;
                    end if;
                    END CASE;
END process sorties ;
S1<=S(0);
S2<=S(1);

END arch_diagramme ;          -- ne pas oublier de terminer l'architecture !
```



Correction Exercice 5 :

```
library ieee ;
use  ieee.std_logic_1164.ALL;

entity MAE is              --MAE = Machine A Etat
  port ( E,RST,HOR         : in   STD_LOGIC ;
         S                 : out STD_LOGIC  );
end MAE;
architecture COMPORTEMENT of MAE is
  signal REG_ETAT                : STD_LOGIC_VECTOR(2 downto 0);
  process (HOR,RST)
  begin
  if RST='0' then REG_ETAT <= "000";
                elsif (HOR'event and HOR='1') then
                                case REG_ETAT is
                                when "000" => S = '0' ;
                                                if E = '1' then REG_ETAT <= "001";
                                                        else REG_ETAT <= "000";
                                                end if ;
                                when "001" => S = '0' ;
                                                if E = '1' then REG_ETAT <= "010";
                                                        else REG_ETAT <= "000";
                                                end if ;
                                when "010" => S = '0' ;
                                                if E = '1' then REG_ETAT <= "011";
                                                        else REG_ETAT <= "000";
                                                end if ;
                                when "011" => S = '1' ;
                                                if E = '1' then REG_ETAT <= "011";
                                                        else REG_ETAT <= "100";
                                                end if ;
                                when "100" => S = '1' ;
                                                if E = '1' then REG_ETAT <= "011";
                                                        else REG_ETAT <= "101";
                                                end if ;
                                when "101" => S = '0' ;
                                                if E = '1' then REG_ETAT <= "011";
                                                        else REG_ETAT <= "000";
                                                end if ;
                                when others => REG_ETAT <= "00";
                                end case ;
  end if ;
  end process ;
end COMPORTEMENT ;
```