
# EC de programmation des composants embarqués

# Langage VHDL

Rendu de projet

Fait par :

- NEMIR DOUAOUDA Chaima
- BELAIBOUDE Kahina

Supervisé par :

- Mr .Ali CHÉRIF

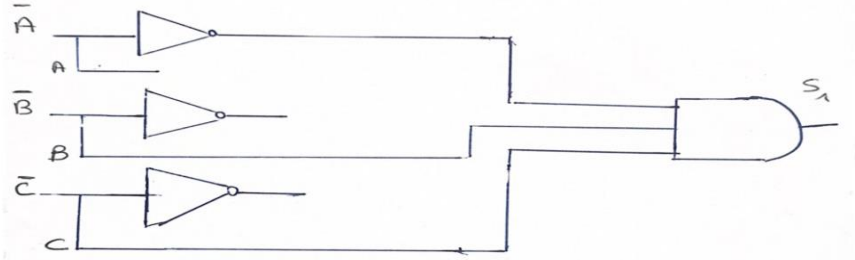
## Exercice 01:

Exercice 01 :

1- Découper Le circuit en 4 "mini circuit" :

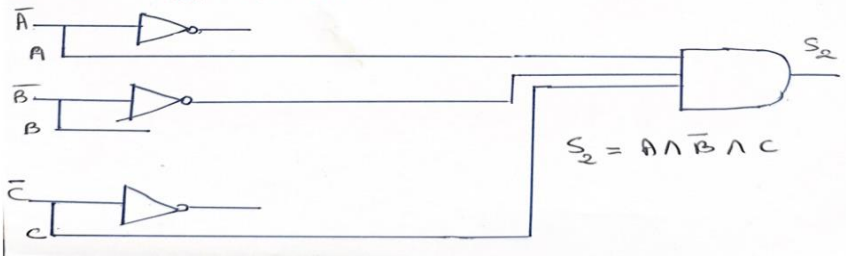
$S = S_1 \vee S_2 \vee S_3 \vee S_4$

↳ 1er mini circuit :

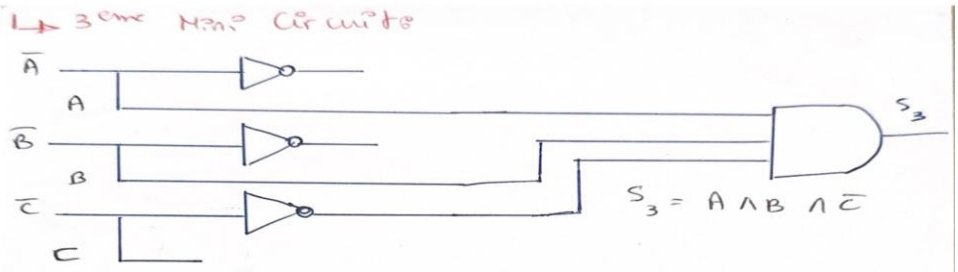



$S_1 = \bar{A} \wedge B \wedge C$

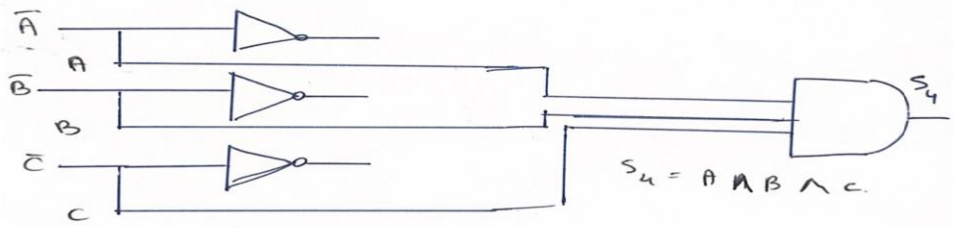
↳ 2ème Mini circuit :




↳ 3ème Mini Circuit :

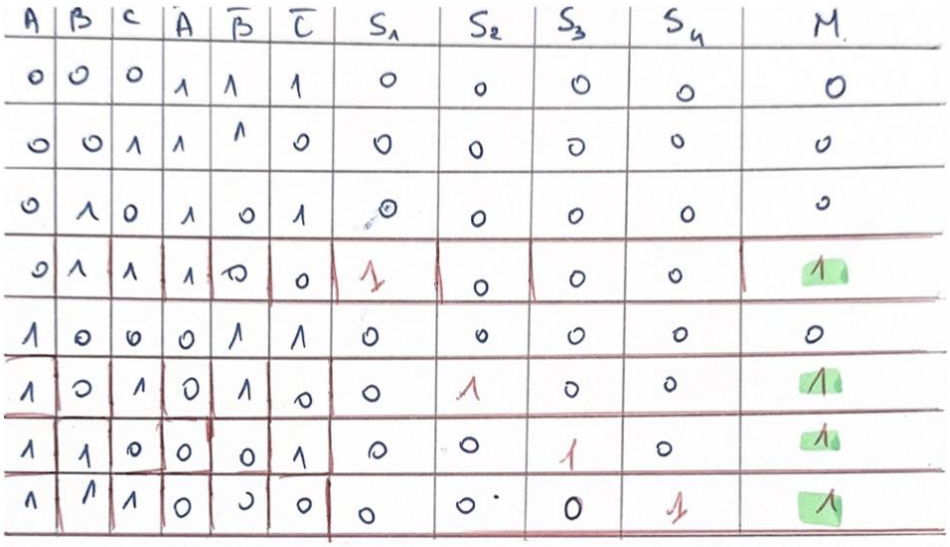



↳ 4ème Mini Circuit :




$M = S_1 \vee S_2 \vee S_3 \vee S_4$

$M = (\bar{A} \wedge B \wedge C) \vee (A \wedge \bar{B} \wedge C) \vee (A \wedge B \wedge \bar{C}) \wedge (A \wedge B \wedge C)$




( On a utilisé une extension VHDL sur vscode pour la compilation on a utiliser le compilateur de ghdl en utilisant des ligne de commande sur un terminal et pour afficher les signaux et le waves on a utilisé GTKWave )

Ça se traduit en VHDL par:

e.vhdl > Archi_Cicruit (Circuit) > @line13

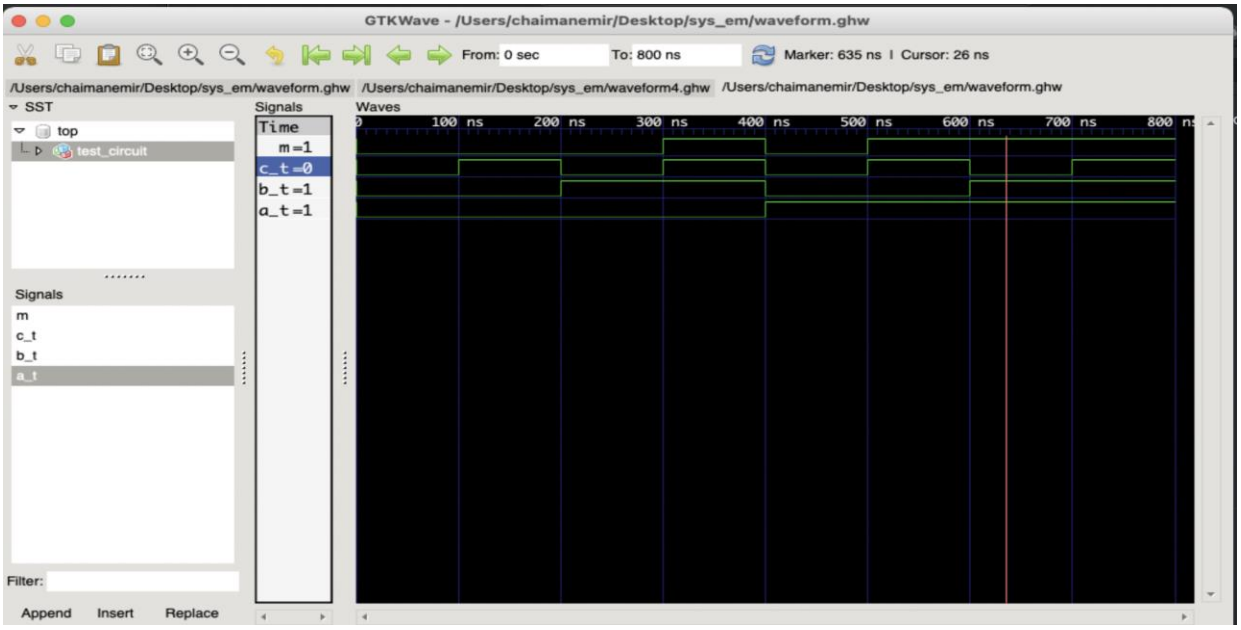
```vhdl
library ieee;
use ieee.std_logic_1164.all;

entity Circuit is
  port (
    A, B, C : in std_logic;
    M : out std_logic
  );
end entity Circuit;

architecture Archi_Cicruit of Circuit is
begin
  process (A, B, C)
  begin
    if (((not A) and B and C) = '1' or (A and (not B) and C) = '1' or (A and B and (not C)) = '1'  or (A and B
      M <= '1';  -- La sortie M est activée
    else
      M <= '0';  -- La sortie M est désactivée
    end if;
  end process;
end architecture Archi_Cicruit;
```

À l'aide de la librairie ieee on a définit un seul composant ‘Circuit ‘avec 3 portes à l’aide des opérateurs logiques AND, OR et NOT.

Pour le tester et implimenter la simulation on a créer un fichier e1_test.vhdl à l'aide de la table de vérité,

```
e1_test.vhdl > behavior (test_circuit) > stimulus
 4  entity test_circuit is
 5      end test_circuit;
 6      architecture behavior of test_circuit is
 7          component Circuit is
 8              Port (
 9                  A, B, C : in std_logic;
10                  M : out std_logic
11              );
12          end component;
13          signal A_t, B_t , C_t, M : std_logic;
14      begin
15          uut: Circuit port map (A_t, B_t, C_t, M);
16          stimulus: process
17          begin
18              A_t <= '0'; B_t <= '0'; C_t <= '0'; wait for 100 ns;
19              assert (M = '0');
20              A_t <= '0'; B_t <= '0'; C_t <= '1'; wait for 100 ns;
21              assert (M = '0');
22              A_t <= '0'; B_t <= '1'; C_t <= '0'; wait for 100 ns;
23              assert (M = '0');
24              A_t <= '0'; B_t <= '1'; C_t <= '1'; wait for 100 ns;
25              assert (M = '1');
26              A_t <= '1'; B_t <= '0'; C_t <= '0'; wait for 100 ns;
27              assert (M = '0');
28              A_t <= '1'; B_t <= '0'; C_t <= '1'; wait for 100 ns;
29              assert (M = '1');
30              A_t <= '1'; B_t <= '1'; C_t <= '0'; wait for 100 ns;
31              assert (M = '1');
32              A_t <= '1'; B_t <= '1'; C_t <= '1'; wait for 100 ns;
33              assert (M = '1');
34              wait;
35          end process;
36      end behavior;
```

le résulat du test de simulation :



## Exercice 2 :

Programmation de comparateur :

| A | B | A>B | A<B | A=B |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 |

On peut conclure de la table de vérité que l'on a 2 entrée  A et B, et 3 sorties: A>B , A<B et A=B

Ça se traduit en VHDL par :

Users > chaimanemir > Desktop > e2.vhdl > ...

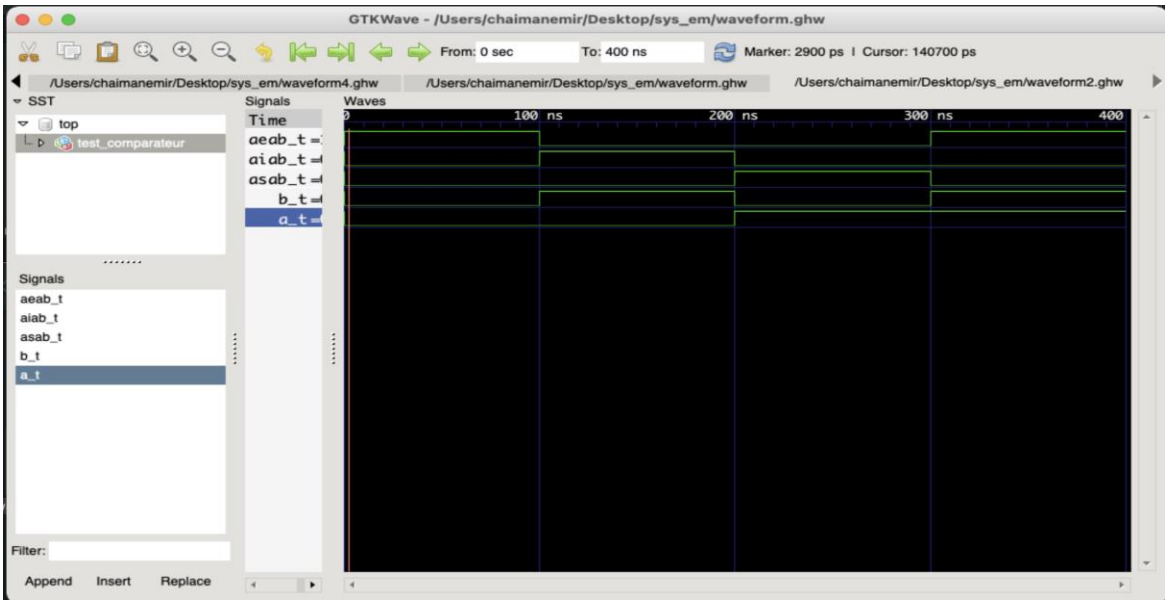
```vhdl
library ieee;
use ieee.std_logic_1164.all;

entity COMPARATEUR is
  port(
    A,B : in std_logic;
    AsaB, AiaB, AeaB : out std_logic
);
end COMPARATEUR;
architecture Archi_comp of comparateur is
begin
    asab <= '1' when a > b else '0';
    aiab <= '1' when a < b else '0';
    aeab <= '1' when a = b else '0';
end Archi_comp;
```

À l'aide de la librairie ieee on a définit le comparateur avec 2 portes d'entrés A er B et 3 sorties : AsaB, AiaB et AeaB à l'aide des opérateurs de comparaison < , > et =.

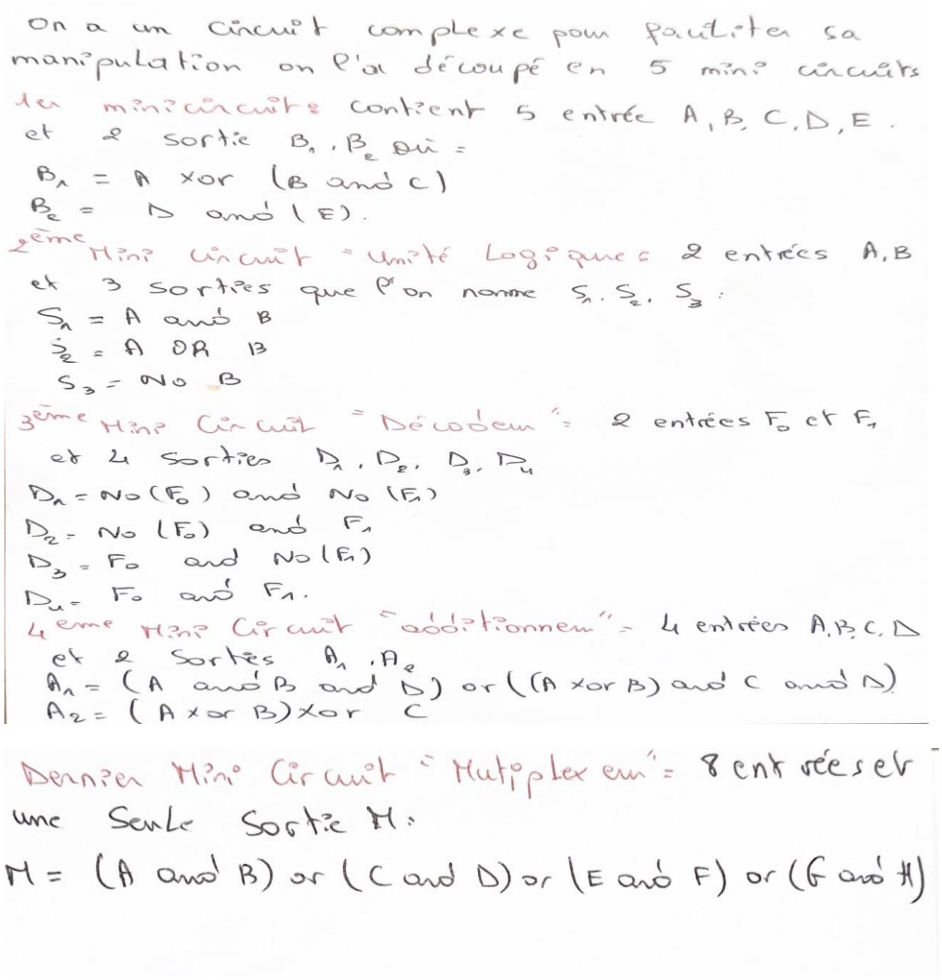
Pour le tester et implimenter la simulation on a créer un fichier e2_test.vhdl à l'aide de la table de vérité donnée.

```vhdl
e2_test.vhdl > behavior (test_comparateur)
3
4   entity test_comparateur is
5   end test_comparateur;
6   architecture behavior of test_comparateur is
7     component comparateur is
8       port (
9         A, B : in std_logic;
10        AsaB, AiaB, AeaB : out std_logic
11      );
12    end component;
13    signal A_t, B_t, AsaB_t, AiaB_t, AeaB_t : std_logic;
14  begin
15    uut: comparateur port map (A_t, B_t, AsaB_t, AiaB_t, AeaB_t);
16    stimulus: process
17    begin
18      A_t <= '0';
19      B_t <= '0';
20      wait for 100 ns;
21      assert (AsaB_t = '0');
22      assert (AiaB_t = '0');
23      assert (AeaB_t = '1');
24      A_t <= '0';
25      B_t <= '1';
26      wait for 100 ns;
27      assert (AsaB_t = '0');
28      assert (AiaB_t = '1');
29      assert (AeaB_t = '0');
30      A_t <= '1';
31      B_t <= '0';
32      wait for 100 ns;
33      assert (AsaB_t = '1');
34      assert (AiaB_t = '0');
35      assert (AeaB_t = '0');
```

Le résultat de test de simulation:



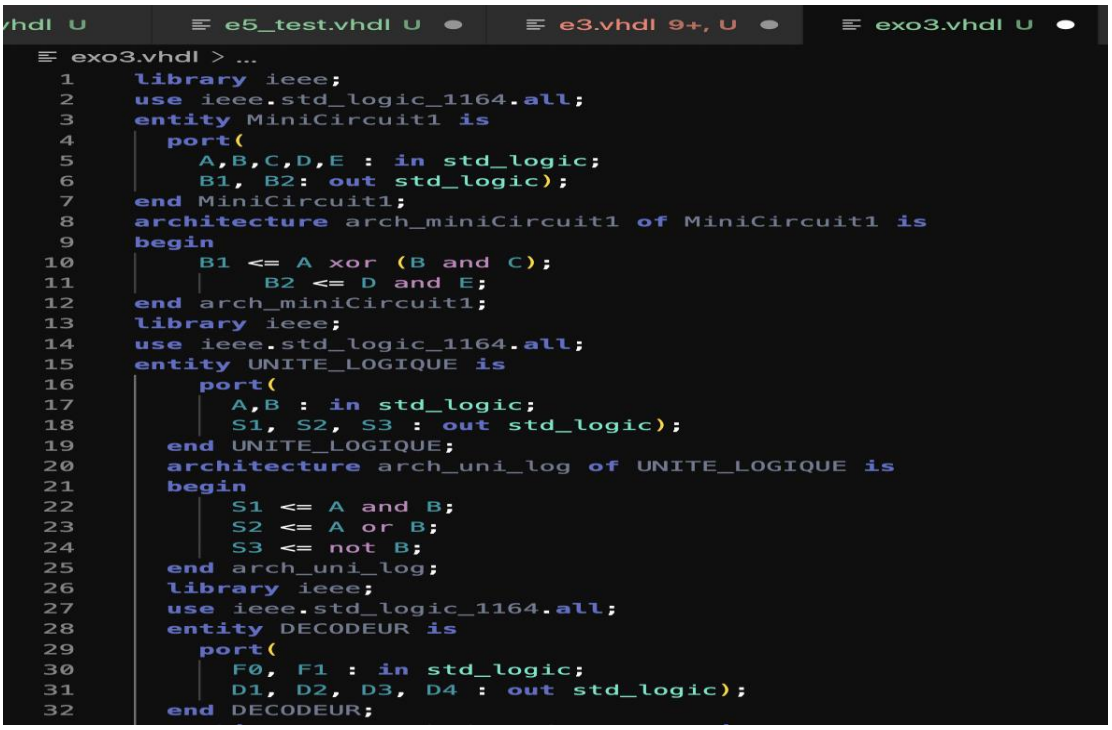

## Exercice 03:



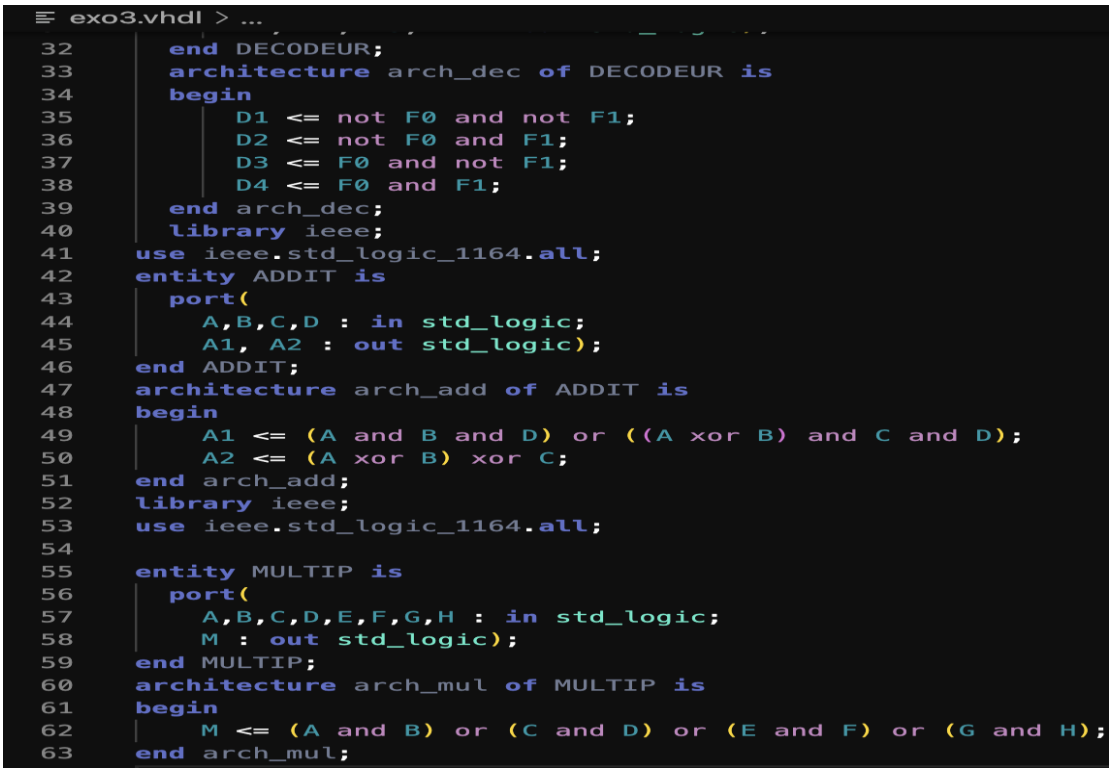




Ça se traduit en VHDL en :



```
vhdl U    e5_test.vhdl U    e3.vhdl 9+, U    exo3.vhdl U
exo3.vhdl > ...
library ieee;
use ieee.std_logic_1164.all;
entity MiniCircuit1 is
  port(
    A,B,C,D,E : in std_logic;
    B1, B2: out std_logic);
end MiniCircuit1;
architecture arch_miniCircuit1 of MiniCircuit1 is
begin
    B1 <= A xor (B and C);
        B2 <= D and E;
end arch_miniCircuit1;
library ieee;
use ieee.std_logic_1164.all;
entity UNITE_LOGIQUE is
    port(
      A,B : in std_logic;
      S1, S2, S3 : out std_logic);
  end UNITE_LOGIQUE;
  architecture arch_uni_log of UNITE_LOGIQUE is
  begin
      S1 <= A and B;
      S2 <= A or B;
      S3 <= not B;
  end arch_uni_log;
  library ieee;
  use ieee.std_logic_1164.all;
  entity DECODEUR is
    port(
      F0, F1 : in std_logic;
      D1, D2, D3, D4 : out std_logic);
  end DECODEUR;
```



```
exo3.vhdl > ...
  end DECODEUR;
  architecture arch_dec of DECODEUR is
  begin
      D1 <= not F0 and not F1;
      D2 <= not F0 and F1;
      D3 <= F0 and not F1;
      D4 <= F0 and F1;
  end arch_dec;
  library ieee;
use ieee.std_logic_1164.all;
entity ADDIT is
  port(
    A,B,C,D : in std_logic;
    A1, A2 : out std_logic);
end ADDIT;
architecture arch_add of ADDIT is
begin
    A1 <= (A and B and D) or ((A xor B) and C and D);
    A2 <= (A xor B) xor C;
end arch_add;
library ieee;
use ieee.std_logic_1164.all;

entity MULTIP is
  port(
    A,B,C,D,E,F,G,H : in std_logic;
    M : out std_logic);
end MULTIP;
architecture arch_mul of MULTIP is
begin
    M <= (A and B) or (C and D) or (E and F) or (G and H);
end arch_mul;
```

exo3.vhdl > arch_Circuit_Complexe (MAIN) > MULTIP

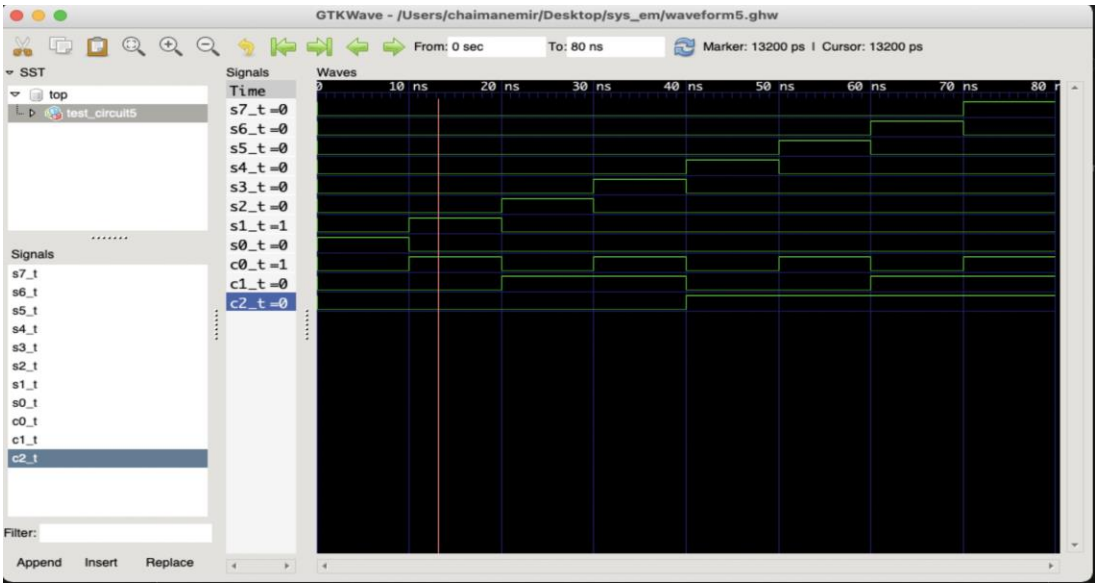
```vhdl
architecture arch_Circuit_Complexe of MAIN is
    component MiniCircuit1 is
    port(   A,B,C,D,E : in std_logic;
            B1, B2: out std_logic);
    end component;
    component UNITE_LOGIQUE is
    port(   A: in std_logic;
            B: in std_logic;
            S1: out std_logic;
        S2: out std_logic;
            S3: out std_logic );
    end component;
    component ADDIT is
    port(   A,B,C,D : in std_logic;
            A1, A2 : out std_logic);
    end component;
    component MULTIP is
    port(   A,B,C,D,E,F,G,H : in std_logic;
            M : out std_logic);
    end component;
    component DECODEUR is
    port(   F0, F1 : in std_logic;
            D1, D2, D3, D4 : out std_logic);
    end component;
    signal B1, B2 : std_logic;
    signal UL1, UL2, UL3 : std_logic;
    signal D1, D2, D3, D4 : std_logic;
    signal A1 : std_logic;
begin
    Porte1: MiniCircuit1 port map (A=>INVA, B=>A, C=>ENA, D=>B, E=>ENB, B1=>B1, B2=>B2);
    Porte2: UNITE_LOGIQUE port map (A=>B1, B=>B2, S1=>UL1, S2=>UL2, S3=>UL3);
    Porte3: DECODEUR port map (F0=>F0, F1=>F1, D1=>D1, D2=>D2, D3=>D3, D4=>D4);
    Porte4: ADDIT port map (A=>B1, B=>B2, C=>Cin, D=>D4, A1=>C_out, A2=>A1);
```

Pour le tester:

e3_test.vhdl > behavior (CircuitComplex_test) > @line23

```vhdl
entity CircuitComplex_test is
end CircuitComplex_test;
architecture behavior of CircuitComplex_test is
  signal A, B, Cin, F0, F1, INVA, ENA, ENB : std_logic;
  signal C_out, X : std_logic;
begin
  uut: entity work.MAIN
    port map (
      A => A,
      B => B,
      Cin => Cin,
      F0 => F0,
      F1 => F1,
      INVA => INVA,
      ENA => ENA,
      ENB => ENB,
      C_out => C_out,
      X => X);
  process
  begin
    A <= '0';
    B <= '1';
    Cin <= '0';
    F0 <= '0';
    F1 <= '1';
    INVA <= '1';
    ENA <= '1';
    ENB <= '0';
    wait for 10 ns;
    report "C_out = " & std_logic'image(C_out);
    report "X = " & std_logic'image(X);
    wait;
  end process;
```

Le résultat de test:




## Exercice 04:

Programmer un décompte circulaire  front déscendant d'horloge de 20 à à ensuite de 0 à 20 :

On a défini un composant CompteurCirculaire contenant 'count_max 'qui définit la valeur 20 ' hlg 'qui représente l'horloge,' r' est une entrée de réintialisation et 'compteur' est la sortie qui représente la valeur actuelle de décompte.

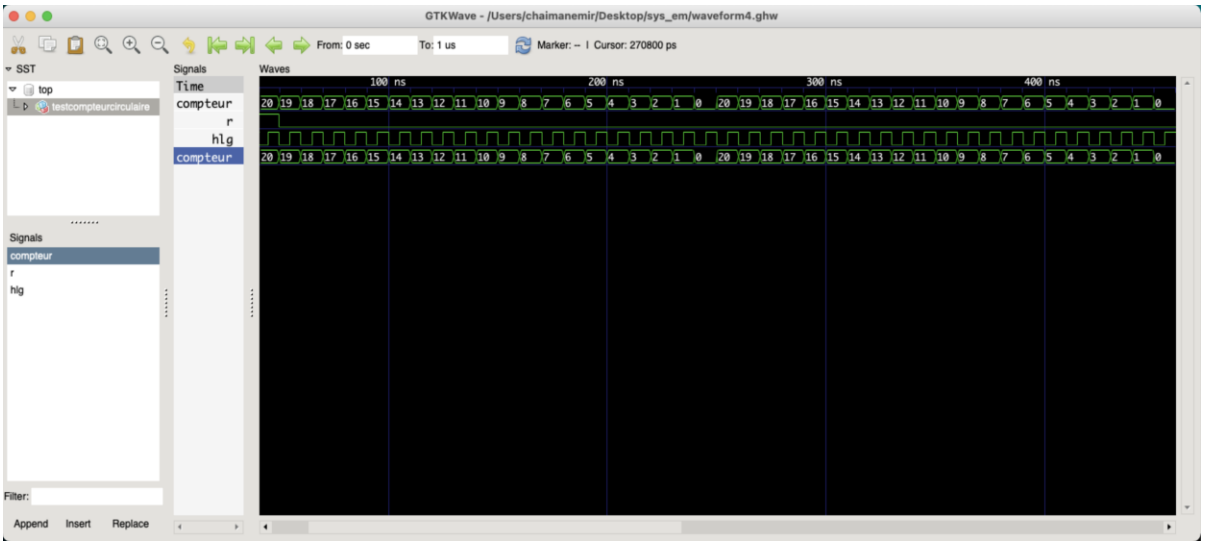
Le processus 'Behavioral' gère le processus de décompte, une fois le compteur atteint 0, il est réinitaliser à la valeur maximal: 20.

```vhdl
e4.vhdl > Behavioral (CompteurCirculaire) > @line18
1   library IEEE;
2   use IEEE.STD_LOGIC_1164.ALL;
3   use IEEE.NUMERIC_STD.ALL;
4
5   entity CompteurCirculaire is
6       generic (
7           MAX_COUNT : natural := 20
8       );
9       port (
10          hlg : in STD_LOGIC;
11          r : in STD_LOGIC;
12          compteur : out natural range 0 to MAX_COUNT
13      );
14  end CompteurCirculaire;
15  architecture Behavioral of CompteurCirculaire is
16      signal count : natural range 0 to MAX_COUNT;
17  begin
18      process (hlg, r)
19      begin
20          if r = '1' then
21              count <= MAX_COUNT;
22          elsif falling_edge(hlg) then
23              if count = 0 then
24                  count <= MAX_COUNT;
25              else
26                  count <= count - 1;
27              end if;
28          end if;
29      end process;
30      compteur <= count;
31  end Behavioral;
```

Pour le tester :

```vhdl
e4_test.vhdl > Behavioral (TestCompteurCirculaire)
12      signal compteur : natural;
13  begin
14      uut: entity work.CompteurCirculaire
15          generic map (
16              MAX_COUNT => 20
17          )
18          port map (
19              hlg => hlg,
20              r => r,
21              compteur => compteur
22          );
23      clk_process: process
24      begin
25          while now < 1000 ns loop
26              hlg <= '0';
27              wait for CLK_PERIOD / 2;
28              hlg <= '1';
29              wait for CLK_PERIOD / 2;
30          end loop;
31          wait;
32      end process;
33
34      reset_process: process
35      begin
36          r <= '1';
37          wait for 50 ns;
38          r <= '0';
39          wait;
40      end process;
41      display_process: process
42      begin
43          while now < 1000 ns loop
44              wait for CLK_PERIOD;
```

Le résultat de test :




## Exercice 05:

On a un circuit qui se compose de 3 entrées C2, C1 et C0 et 8 sortie S0, S1, ..,S7

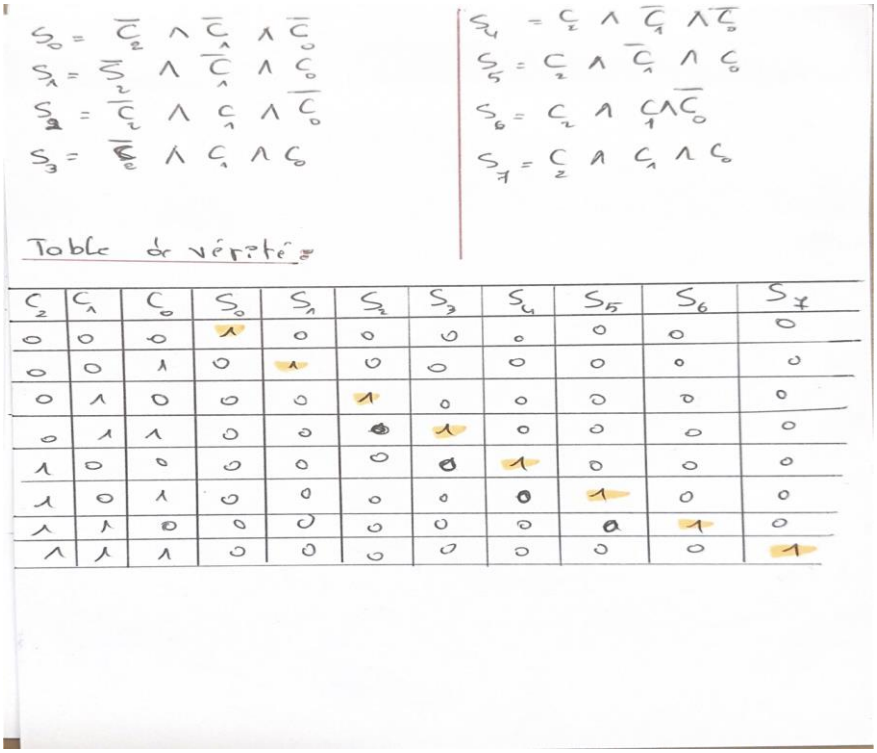


Table de vérité

| $C_2$ | $C_1$ | $C_0$ | $S_0$ | $S_1$ | $S_2$ | $S_3$ | $S_4$ | $S_5$ | $S_6$ | $S_7$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

Ça se traduit en VHDL par :

e5.vhdl > behavior (Circuit5)

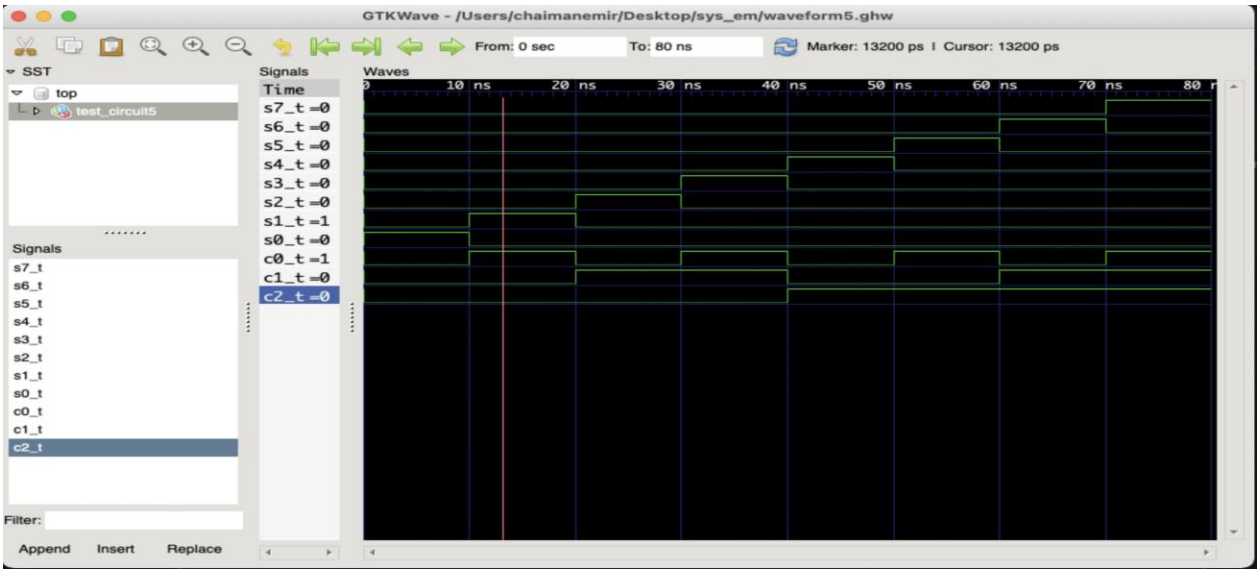
```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Circuit5 is
    Port (
        C2, C1, C0 : in std_logic;
        S0, S1, S2, S3, S4, S5, S6, S7 : out std_logic
    );
end Circuit5;

architecture behavior of Circuit5 is
begin
    S0 <= not C2 and not C1 and not C0;
    S1 <= not C2 and not C1 and C0;
    S2 <= not C2 and C1 and not C0;
    S3 <= not C2 and C1 and C0;
    S4 <= C2 and not C1 and not C0;
    S5 <= C2 and not C1 and C0;
    S6 <= C2 and C1 and not C0;
    S7 <= C2 and C1 and C0;
end behavior;
```

Pour le tester :

e5_test.vhdl > behavior (Test_Circuit5) > uut

```vhdl
architecture behavior of Test_Circuit5 is
    component Circuit5 is
        Port (
            C2, C1, C0 : in std_logic;
            S0, S1, S2, S3, S4, S5, S6, S7 : out std_logic
        );
    end component;

    signal C2_t, C1_t, C0_t : std_logic;
    signal S0_t, S1_t, S2_t, S3_t, S4_t, S5_t, S6_t, S7_t : std_logic;

begin
    uut: Circuit5 port map (C2_t, C1_t, C0_t, S0_t, S1_t, S2_t, S3_t, S4_t, S5_t, S6_t, S7_t);
    stimulus: process
    begin
        C2_t <= '0'; C1_t <= '0'; C0_t <= '0';wait for 10 ns;
        assert (S0_t = '1' and S1_t = '0' and S2_t = '0' and S3_t = '0' and S4_t = '0' and S5_t = '0' and S6_t

        C2_t <= '0'; C1_t <= '0'; C0_t <= '1';wait for 10 ns;
        assert (S0_t = '0' and S1_t = '1' and S2_t = '0' and S3_t = '0' and S4_t = '0' and S5_t = '0' and S6_t

        C2_t <= '0'; C1_t <= '1'; C0_t <= '0';wait for 10 ns;
        assert (S0_t = '0' and S1_t = '0' and S2_t = '1' and S3_t = '0' and S4_t = '0' and S5_t = '0' and S6_t

        C2_t <= '0'; C1_t <= '1'; C0_t <= '1';wait for 10 ns;
        assert (S0_t = '0' and S1_t = '0' and S2_t = '0' and S3_t = '1' and S4_t = '0' and S5_t = '0' and S6_t

        C2_t <= '1'; C1_t <= '0'; C0_t <= '0';wait for 10 ns;
        assert (S0_t = '0' and S1_t = '0' and S2_t = '0' and S3_t = '0' and S4_t = '1' and S5_t = '0' and S6_t

        C2_t <= '1'; C1_t <= '0'; C0_t <= '1';wait for 10 ns;
        assert (S0_t = '0' and S1_t = '0' and S2_t = '0' and S3_t = '0' and S4_t = '0' and S5_t = '1' and S6_t
```

Le résultat de simulation:


GTKWave - /Users/chaimanemir/Desktop/sys_em/waveform5.ghw
From: 0 sec
To: 80 ns
Marker: 13200 ps | Cursor: 13200 ps
SST
top
test_circuit5
Signals
s7_t
s6_t
s5_t
s4_t
s3_t
s2_t
s1_t
s0_t
c0_t
c1_t
c2_t
Filter:
Append
Insert
Replace
Signals
Time
s7_t =0
s6_t =0
s5_t =0
s4_t =0
s3_t =0
s2_t =0
s1_t =1
s0_t =0
c0_t =1
c1_t =0
c2_t =0
Waves
0
10 ns
20 ns
30 ns
40 ns
50 ns
60 ns
70 ns