Rapport du Projet VHDL
-Réalisation d'une Serrure Électronique-

- **But de la manipulation:**
  Le but de ce projet est de concevoir une serrure électronique à code à 4 chiffres, connectée à un clavier 16 touches via un bus 16 bits BusExt. Le système doit être capable de reconnaître le code utilisateur entré par l'utilisateur et de déclencher l'ouverture de la serrure si le code est correct. Le projet implique la conception de chaque bloc constituant la serrure en VHDL et la validation de leur fonctionnement via des simulations.

- **Réalisation:**

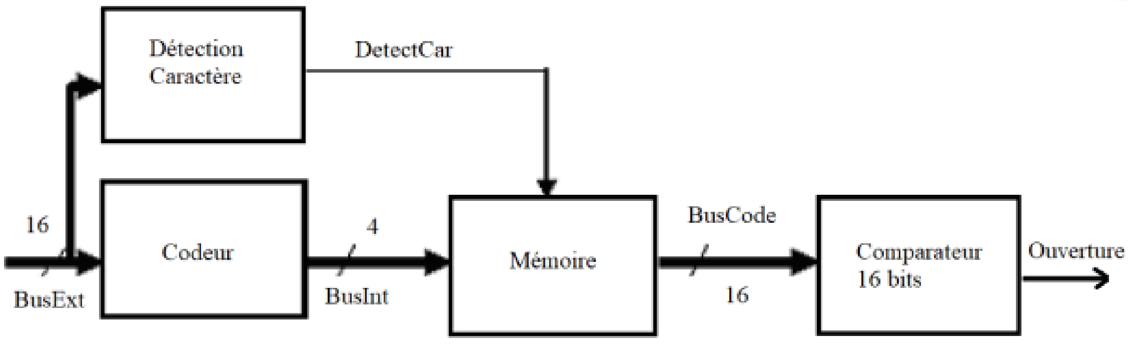
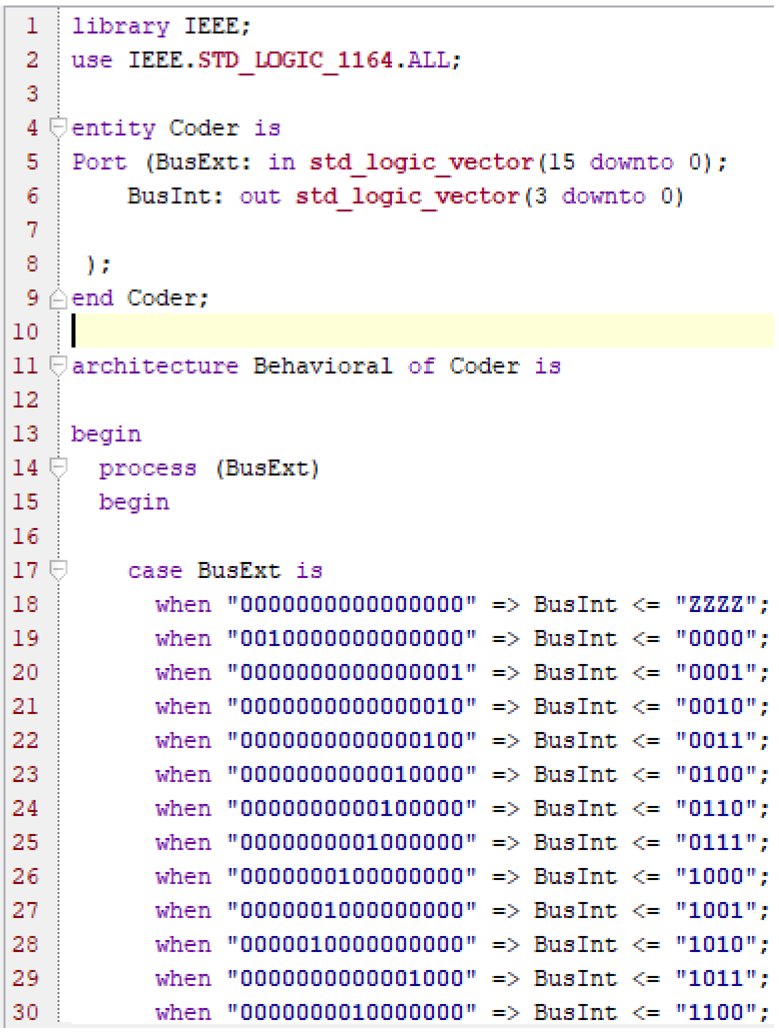


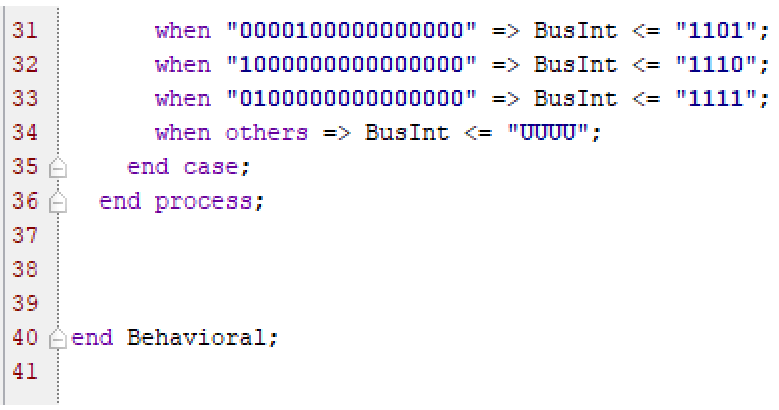
Figure 2 : Architecture globale du système

1)Codeur:

-Description VHDL:



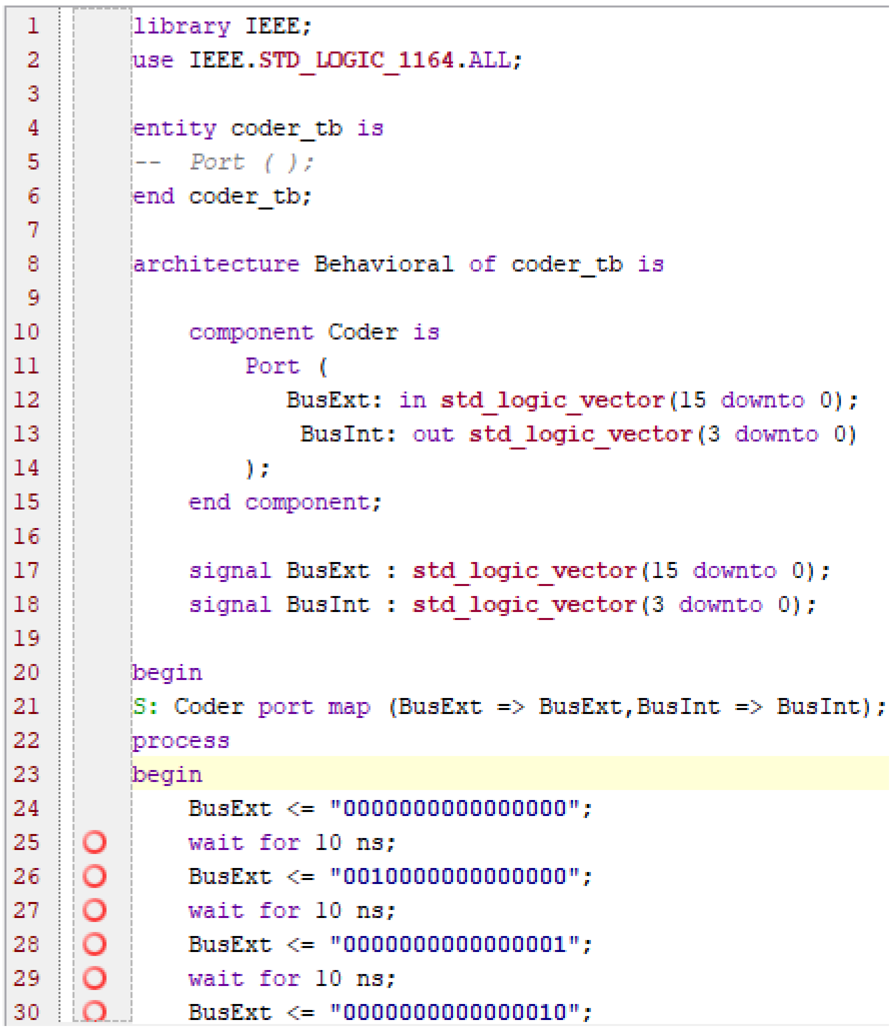
```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Coder is
Port (BusExt: in std_logic_vector(15 downto 0);
    BusInt: out std_logic_vector(3 downto 0)

 );
end Coder;

architecture Behavioral of Coder is

begin
  process (BusExt)
  begin

    case BusExt is
      when "0000000000000000" => BusInt <= "ZZZZ";
      when "0010000000000000" => BusInt <= "0000";
      when "0000000000000001" => BusInt <= "0001";
      when "0000000000000010" => BusInt <= "0010";
      when "0000000000000100" => BusInt <= "0011";
      when "0000000000010000" => BusInt <= "0100";
      when "0000000000100000" => BusInt <= "0110";
      when "0000000001000000" => BusInt <= "0111";
      when "0000000100000000" => BusInt <= "1000";
      when "0000001000000000" => BusInt <= "1001";
      when "0000010000000000" => BusInt <= "1010";
      when "0000000000001000" => BusInt <= "1011";
      when "0000000010000000" => BusInt <= "1100";
```



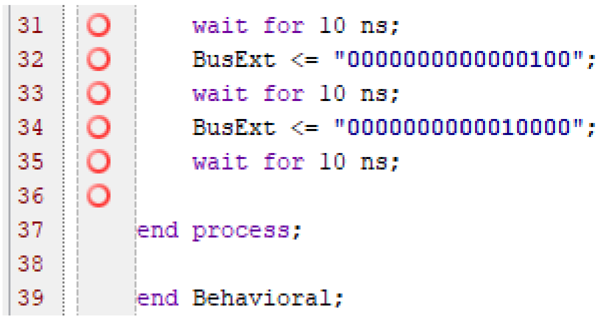
```vhdl
      when "0000100000000000" => BusInt <= "1101";
      when "1000000000000000" => BusInt <= "1110";
      when "0100000000000000" => BusInt <= "1111";
      when others => BusInt <= "UUUU";
    end case;
  end process;



end Behavioral;

```

-Simulation:

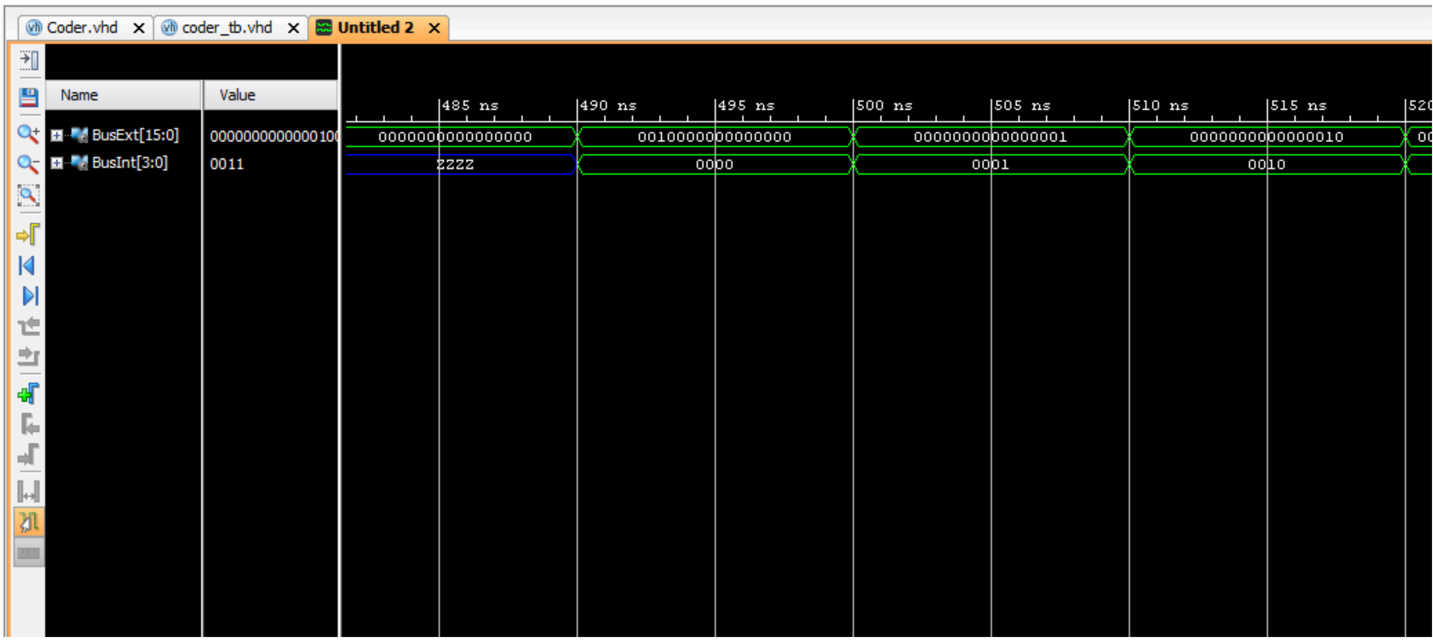
TestBench:



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

entity coder_tb is
--  Port ( );
end coder_tb;

architecture Behavioral of coder_tb is

    component Coder is
        Port (
           BusExt: in std_logic_vector(15 downto 0);
            BusInt: out std_logic_vector(3 downto 0)
        );
    end component;

    signal BusExt : std_logic_vector(15 downto 0);
    signal BusInt : std_logic_vector(3 downto 0);

begin
S: Coder port map (BusExt => BusExt,BusInt => BusInt);
process
begin
    BusExt <= "0000000000000000";
    wait for 10 ns;
    BusExt <= "0010000000000000";
    wait for 10 ns;
    BusExt <= "0000000000000001";
    wait for 10 ns;
    BusExt <= "0000000000000010";
```



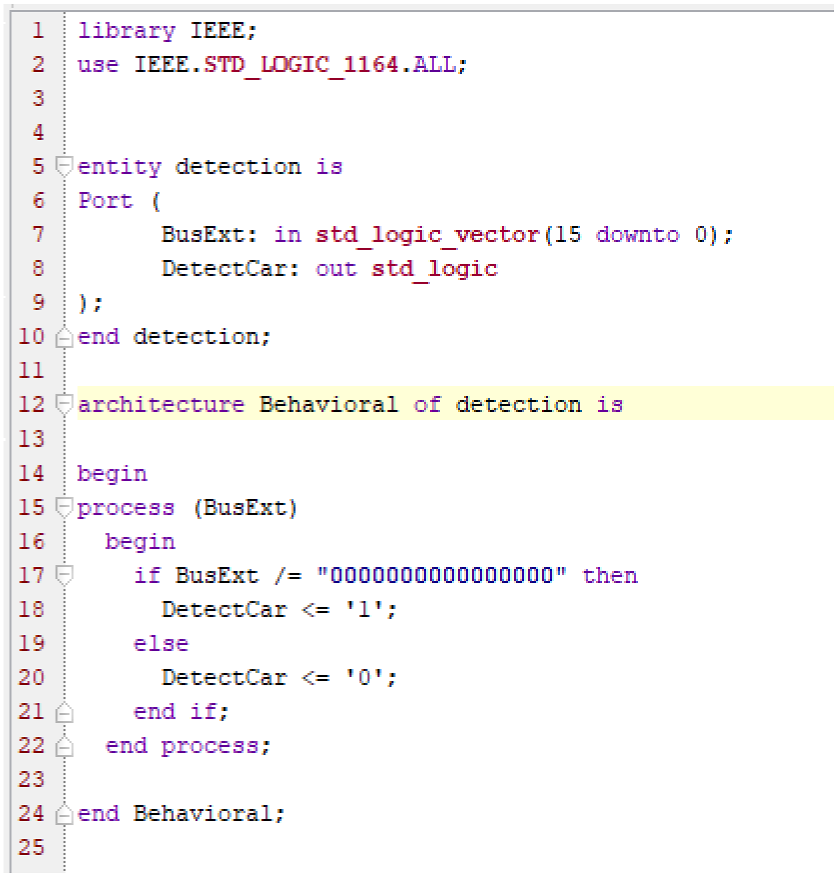
```vhdl
    wait for 10 ns;
    BusExt <= "0000000000000100";
    wait for 10 ns;
    BusExt <= "0000000000010000";
    wait for 10 ns;

end process;

end Behavioral;
```



```
BusExt[15:0]   0000000000000100   0000000000000000   0010000000000000   0000000000000001   0000000000000010   00
BusInt[3:0]    0011               ZZZZ               0000               0001               0010
```

2)Détection:

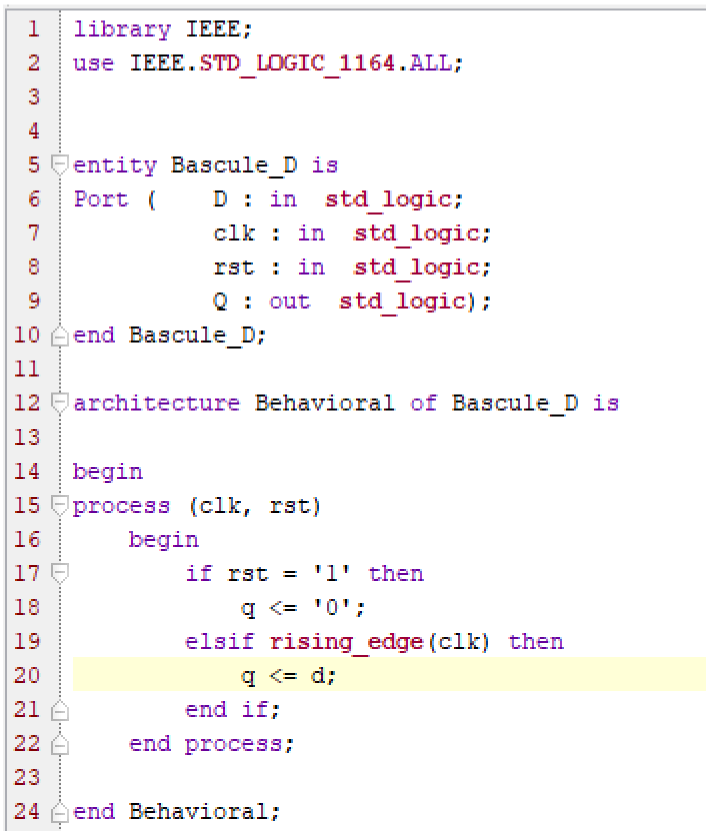
-Description VHDL:



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


entity detection is
Port (
      BusExt: in std_logic_vector(15 downto 0);
      DetectCar: out std_logic
);
end detection;

architecture Behavioral of detection is

begin
process (BusExt)
  begin
    if BusExt /= "0000000000000000" then
      DetectCar <= '1';
    else
      DetectCar <= '0';
    end if;
  end process;

end Behavioral;

```

3)Bascule D active sur le front montant:

-Description VHDL:



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


entity Bascule_D is
Port (    D : in  std_logic;
          clk : in  std_logic;
          rst : in  std_logic;
          Q : out  std_logic);
end Bascule_D;

architecture Behavioral of Bascule_D is

begin
process (clk, rst)
    begin
        if rst = '1' then
            q <= '0';
        elsif rising_edge(clk) then
            q <= d;
        end if;
    end process;

end Behavioral;
```

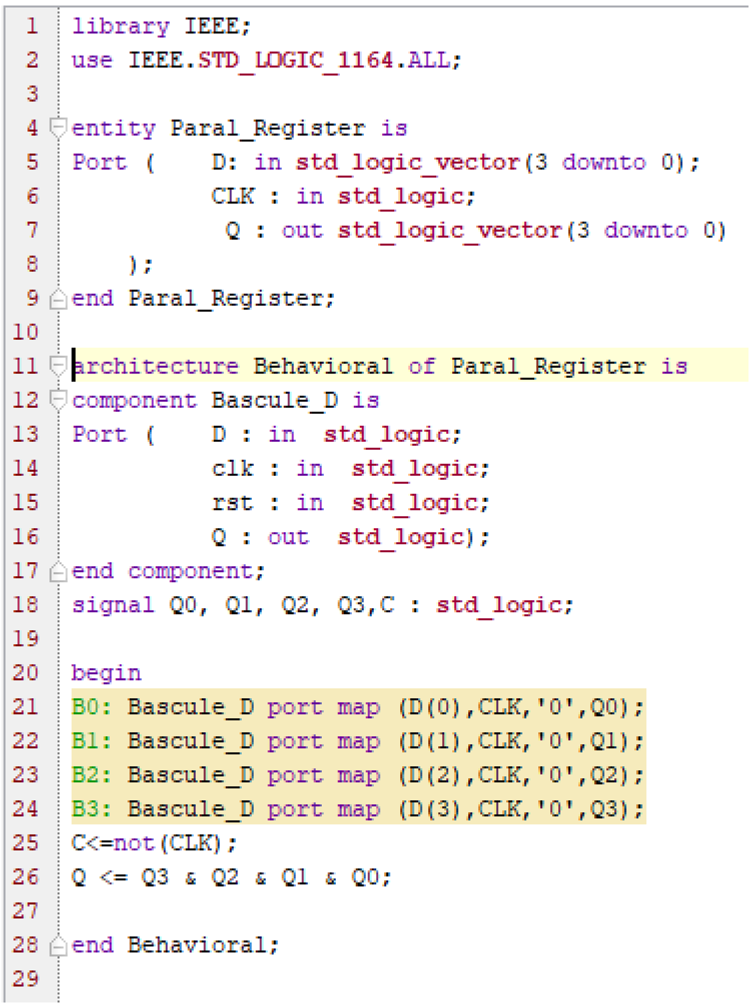
4)Registre à chargement parallèle:

-Le registre actif sur front stocke des données sur le front montant (rising edge) d'un signal d'horloge. Il peut être construit à l'aide de 4 bascules D, où chaque bascule stocke un bit de données d'entrée.

-Table de vérité:

D(3)	D(2)	D(1)	D(0)	CLK	S(3)	S(2)	S(1)	S(0)
x	x	x	x	0	S(3)	S(2)	S(1)	S(0)
D(3)	D(2)	D(1)	D(0)	↑	D(3)	D(2)	D(1)	D(0)

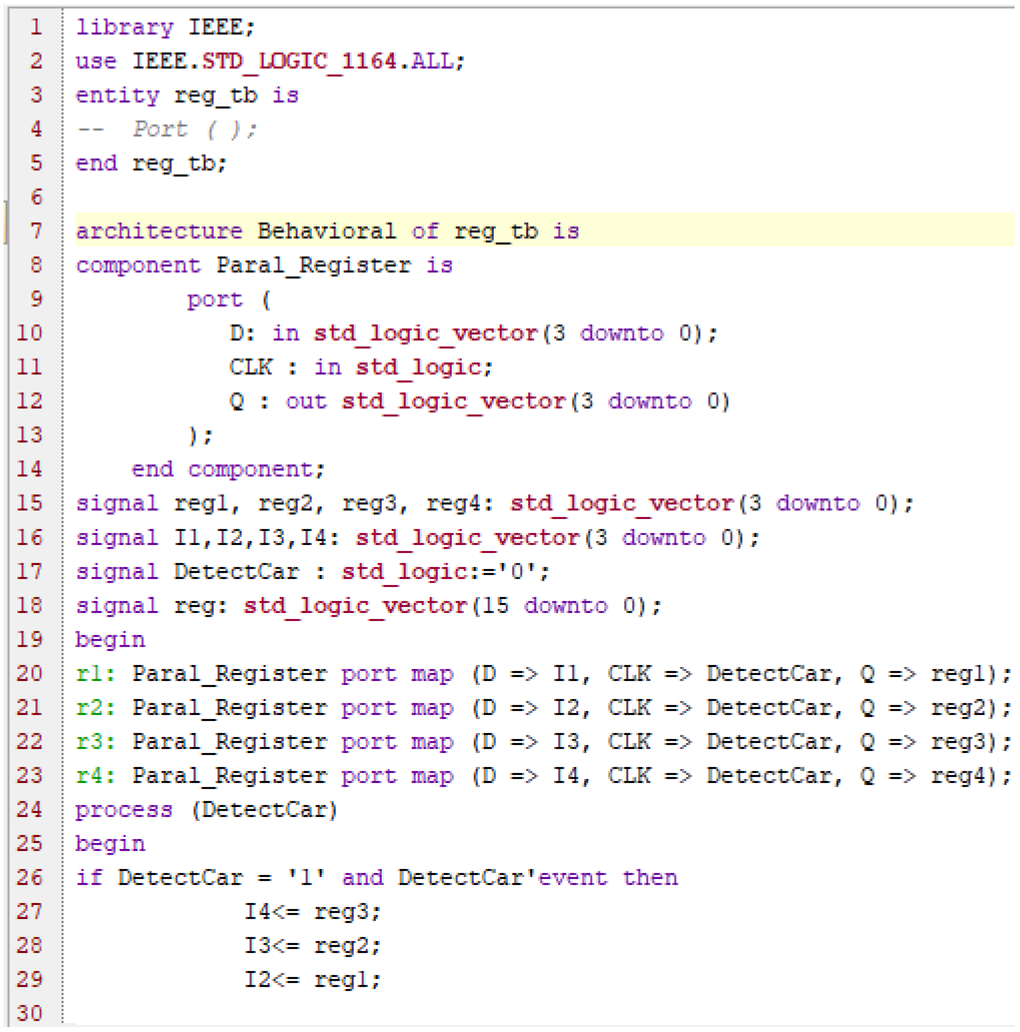
-Description VHDL:



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

entity Paral_Register is
Port (    D: in std_logic_vector(3 downto 0);
          CLK : in std_logic;
           Q : out std_logic_vector(3 downto 0)
    );
end Paral_Register;

architecture Behavioral of Paral_Register is
component Bascule_D is
Port (    D : in  std_logic;
          clk : in  std_logic;
          rst : in  std_logic;
          Q : out  std_logic);
end component;
signal Q0, Q1, Q2, Q3,C : std_logic;

begin
B0: Bascule_D port map (D(0),CLK,'0',Q0);
B1: Bascule_D port map (D(1),CLK,'0',Q1);
B2: Bascule_D port map (D(2),CLK,'0',Q2);
B3: Bascule_D port map (D(3),CLK,'0',Q3);
C<=not(CLK);
Q <= Q3 & Q2 & Q1 & Q0;

end Behavioral;
```

-Simulation:

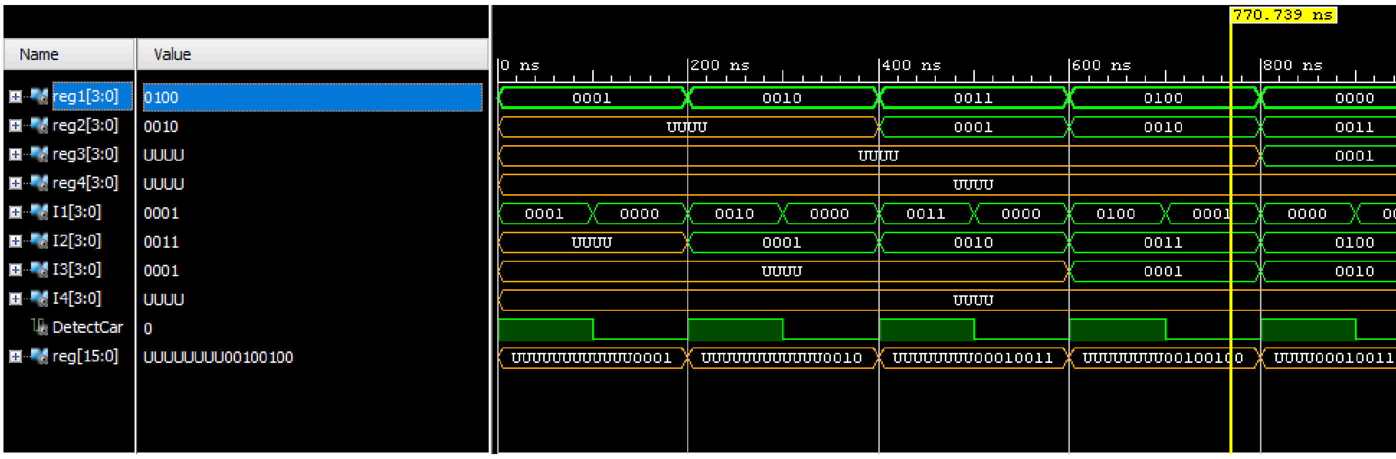
TestBench:



```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity reg_tb is
--  Port ( );
end reg_tb;

architecture Behavioral of reg_tb is
component Paral_Register is
        port (
           D: in std_logic_vector(3 downto 0);
           CLK : in std_logic;
           Q : out std_logic_vector(3 downto 0)
        );
    end component;
signal reg1, reg2, reg3, reg4: std_logic_vector(3 downto 0);
signal I1,I2,I3,I4: std_logic_vector(3 downto 0);
signal DetectCar : std_logic:='0';
signal reg: std_logic_vector(15 downto 0);
begin
r1: Paral_Register port map (D => I1, CLK => DetectCar, Q => reg1);
r2: Paral_Register port map (D => I2, CLK => DetectCar, Q => reg2);
r3: Paral_Register port map (D => I3, CLK => DetectCar, Q => reg3);
r4: Paral_Register port map (D => I4, CLK => DetectCar, Q => reg4);
process (DetectCar)
begin
if DetectCar = '1' and DetectCar'event then
            I4<= reg3;
            I3<= reg2;
            I2<= reg1;

```

```
end if;

    end process;

    reg <= reg4 & reg3 & reg2 & reg1;
process
begin
DetectCar<= not(DetectCar);
I1 <= "0001";
wait for 10ns;
DetectCar<= not(DetectCar);
I1 <= "0000";
wait for 100ns;
DetectCar<= not(DetectCar);
I1 <= "0010";
wait for 10ns;
DetectCar<= not(DetectCar);
I1 <= "0000";
wait for 100ns;
DetectCar<= not(DetectCar);
```

```
wait for 10ns;
DetectCar<= not(DetectCar);
I1 <= "0000";
wait for 100ns;
DetectCar<= not(DetectCar);
I1 <= "0100";
wait for 10ns;
end process;

end Behavioral;
```

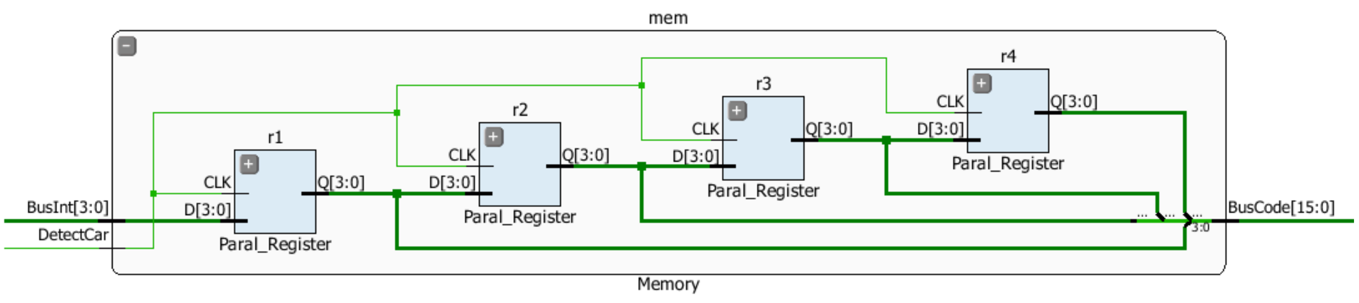


Name	Value
reg1[3:0]	0100
reg2[3:0]	0010
reg3[3:0]	UUUU
reg4[3:0]	UUUU
I1[3:0]	0001
I2[3:0]	0011
I3[3:0]	0001
I4[3:0]	UUUU
DetectCar	0
reg[15:0]	UUUUUUUU00100100

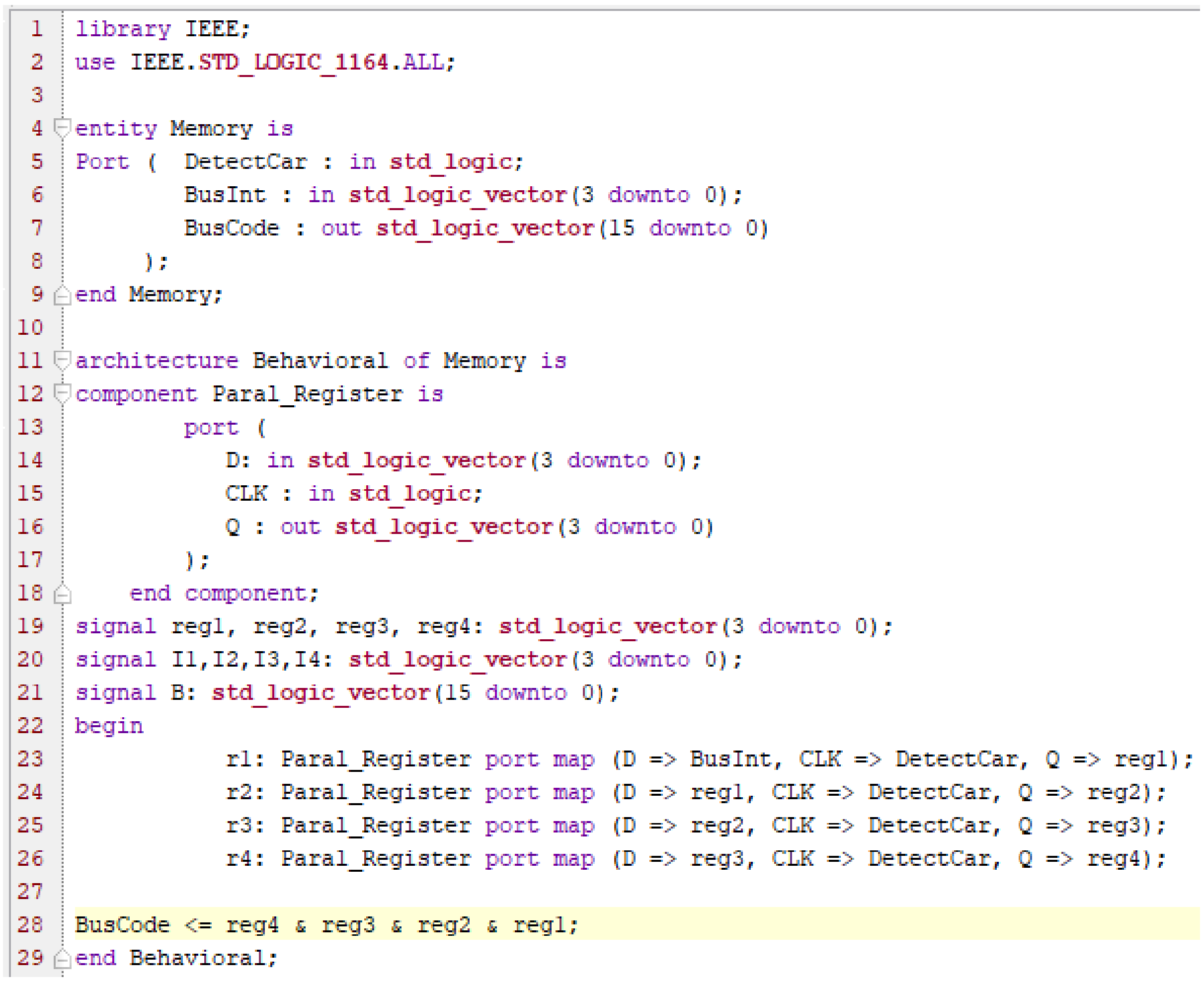
5)Stockage du code:

- le signal clk des registres doit être relié au signal DetectCar, pour que les registres effectuent une lecture parallèle synchronisée avec le signal DetectCar.

-Schéma du mémoire:



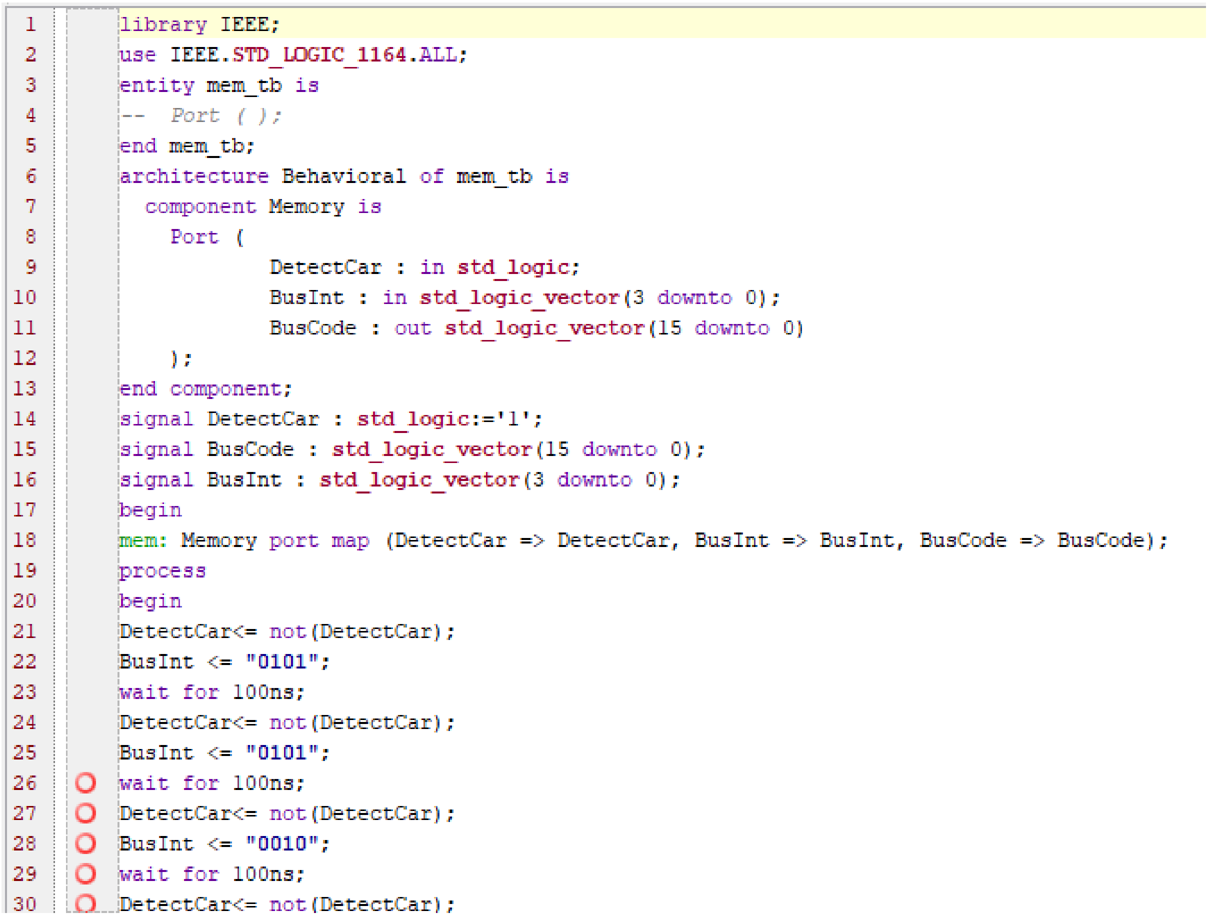
-Description VHDL:



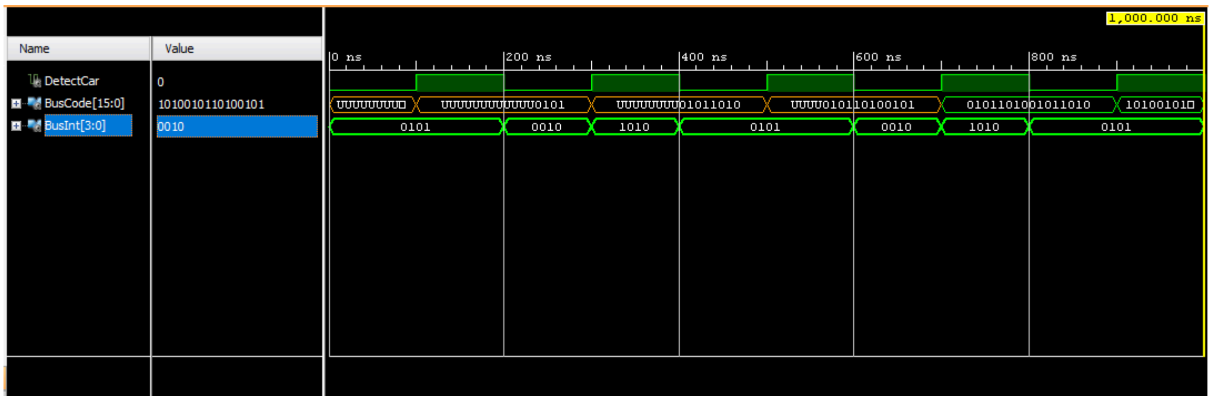
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Memory is
Port (  DetectCar : in std_logic;
        BusInt : in std_logic_vector(3 downto 0);
        BusCode : out std_logic_vector(15 downto 0)
     );
end Memory;

architecture Behavioral of Memory is
component Paral_Register is
        port (
           D: in std_logic_vector(3 downto 0);
           CLK : in std_logic;
           Q : out std_logic_vector(3 downto 0)
        );
    end component;
signal reg1, reg2, reg3, reg4: std_logic_vector(3 downto 0);
signal I1,I2,I3,I4: std_logic_vector(3 downto 0);
signal B: std_logic_vector(15 downto 0);
begin
           r1: Paral_Register port map (D => BusInt, CLK => DetectCar, Q => reg1);
           r2: Paral_Register port map (D => reg1, CLK => DetectCar, Q => reg2);
           r3: Paral_Register port map (D => reg2, CLK => DetectCar, Q => reg3);
           r4: Paral_Register port map (D => reg3, CLK => DetectCar, Q => reg4);

BusCode <= reg4 & reg3 & reg2 & reg1;
end Behavioral;
```

-Simulation:

TestBench:



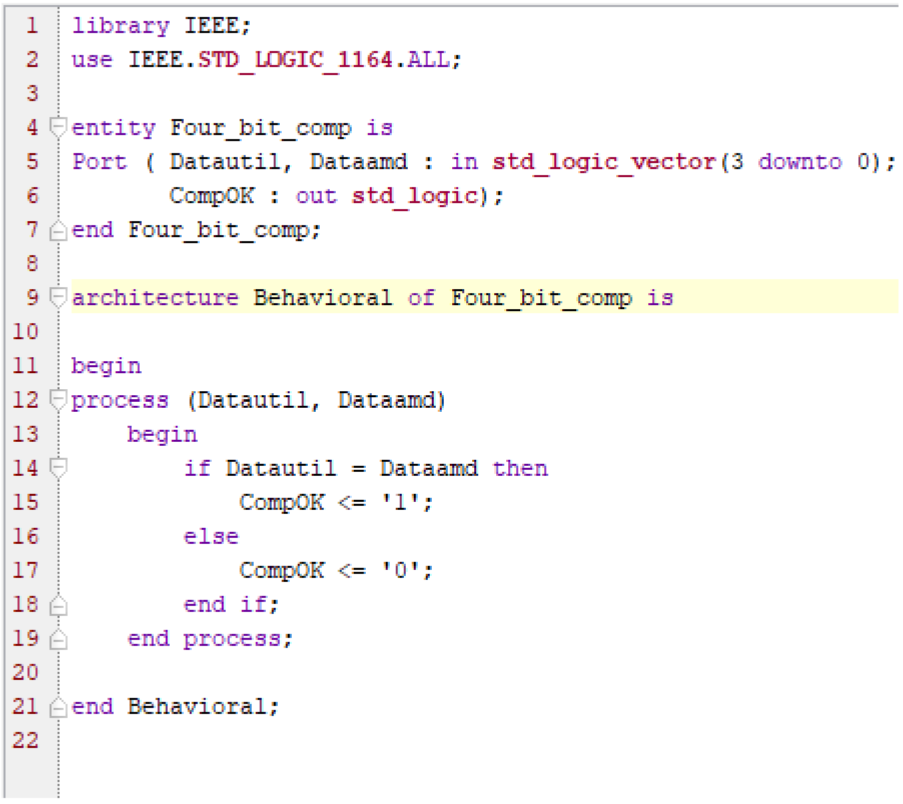
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity mem_tb is
--  Port ( );
end mem_tb;
architecture Behavioral of mem_tb is
  component Memory is
    Port (
            DetectCar : in std_logic;
            BusInt : in std_logic_vector(3 downto 0);
            BusCode : out std_logic_vector(15 downto 0)
    );
end component;
signal DetectCar : std_logic:='1';
signal BusCode : std_logic_vector(15 downto 0);
signal BusInt : std_logic_vector(3 downto 0);
begin
mem: Memory port map (DetectCar => DetectCar, BusInt => BusInt, BusCode => BusCode);
process
begin
DetectCar<= not(DetectCar);
BusInt <= "0101";
wait for 100ns;
DetectCar<= not(DetectCar);
BusInt <= "0101";
wait for 100ns;
DetectCar<= not(DetectCar);
BusInt <= "0010";
wait for 100ns;
DetectCar<= not(DetectCar);
```



6)Comparateur 4 bits:

-Le signal de sortie du comparateur dépend uniquement des signaux d'entrée actuels, Après la dernière bouton le signal BusExt revient à “0000000000000000” et la sortie du comparateur reste à ’1’ si on veut de le remettre à ‘0’ il suffit d'appuyer sur n'importe quelle bouton du clavier.

-Description VHDL:

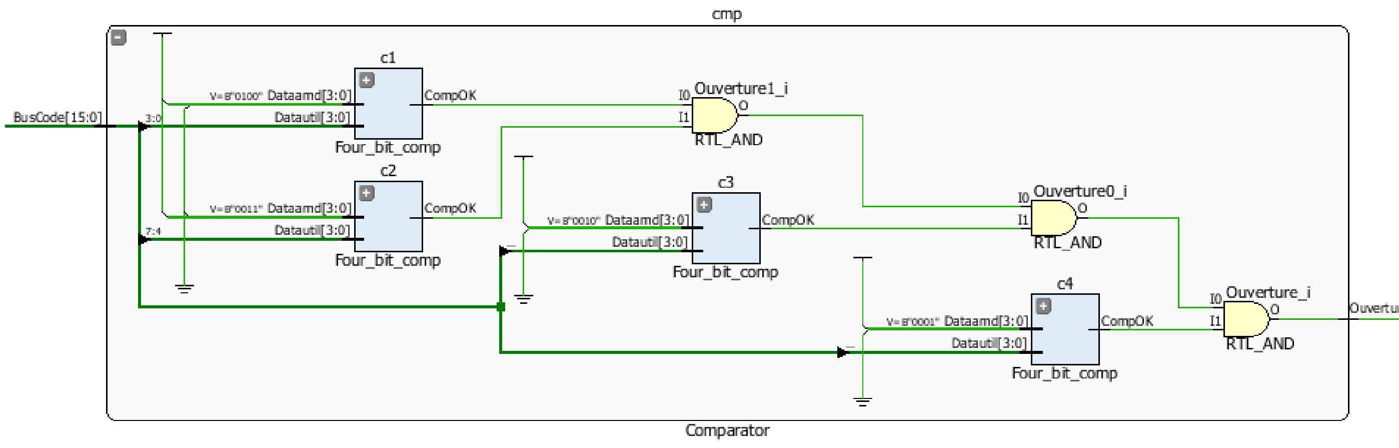


```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Four_bit_comp is
Port ( Datautil, Dataamd : in std_logic_vector(3 downto 0);
       CompOK : out std_logic);
end Four_bit_comp;

architecture Behavioral of Four_bit_comp is

begin
process (Datautil, Dataamd)
    begin
        if Datautil = Dataamd then
            CompOK <= '1';
        else
            CompOK <= '0';
        end if;
    end process;

end Behavioral;

```

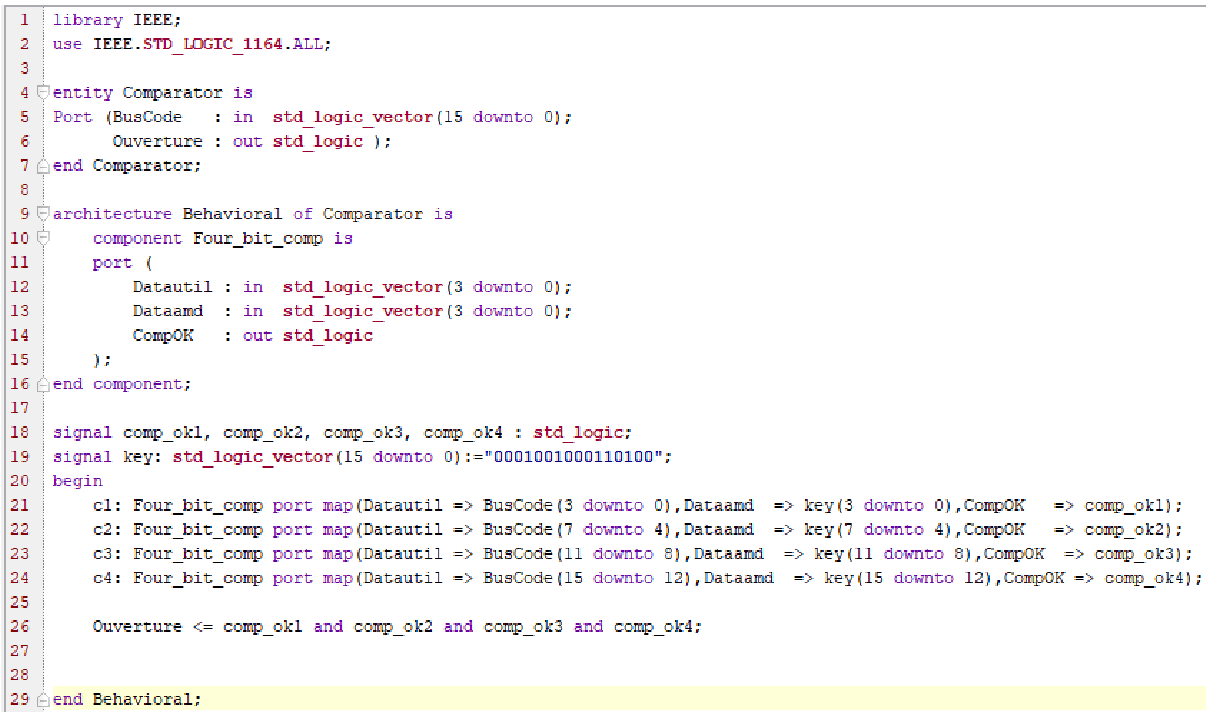
6) Comparateur 16 bits:

-Schéma du comparateur 16 bits:

Il faut ajouter 3 ports AND reliant les sorties des comparateurs à 4 bits.



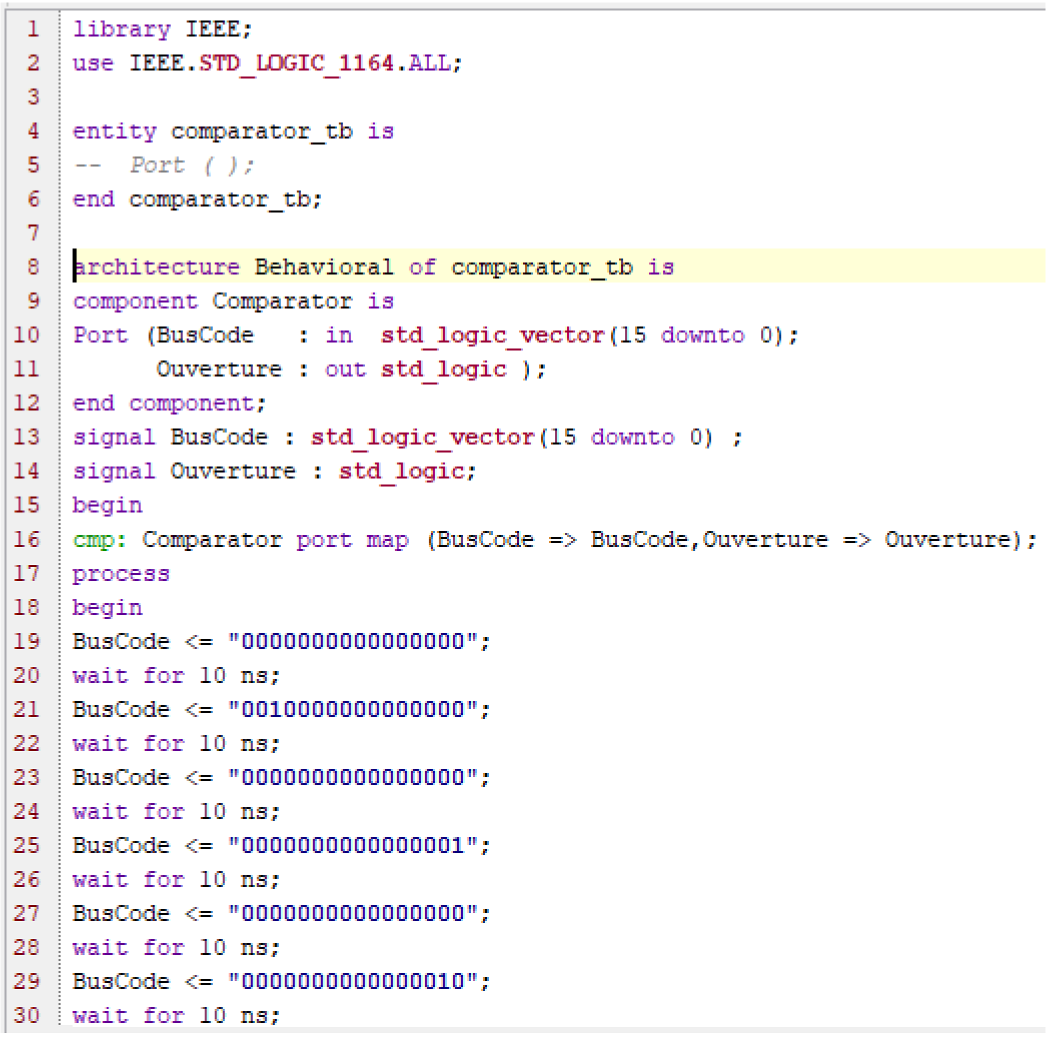
-Description VHDL:



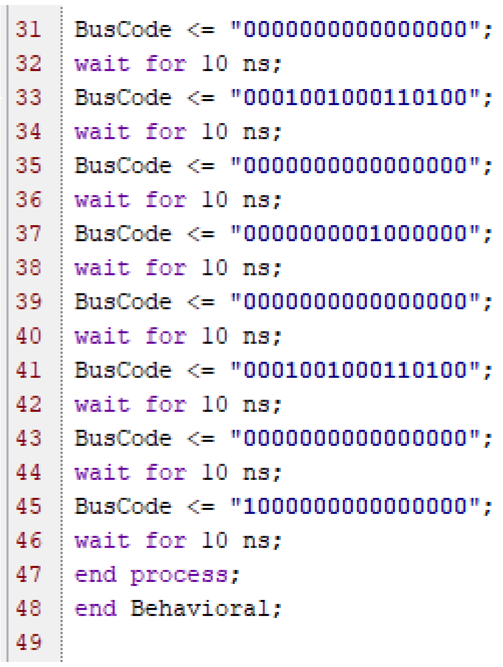
```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Comparator is
Port (BusCode    : in  std_logic_vector(15 downto 0);
      Ouverture : out std_logic );
end Comparator;

architecture Behavioral of Comparator is
    component Four_bit_comp is
    port (
        Datautil  : in  std_logic_vector(3 downto 0);
        Dataamd   : in  std_logic_vector(3 downto 0);
        CompOK    : out std_logic
    );
end component;

signal comp_ok1, comp_ok2, comp_ok3, comp_ok4 : std_logic;
signal key: std_logic_vector(15 downto 0):="0001001000110100";
begin
    c1: Four_bit_comp port map(Datautil => BusCode(3 downto 0),Dataamd  => key(3 downto 0),CompOK   => comp_ok1);
    c2: Four_bit_comp port map(Datautil => BusCode(7 downto 4),Dataamd  => key(7 downto 4),CompOK   => comp_ok2);
    c3: Four_bit_comp port map(Datautil => BusCode(11 downto 8),Dataamd  => key(11 downto 8),CompOK  => comp_ok3);
    c4: Four_bit_comp port map(Datautil => BusCode(15 downto 12),Dataamd  => key(15 downto 12),CompOK => comp_ok4);

    Ouverture <= comp_ok1 and comp_ok2 and comp_ok3 and comp_ok4;


end Behavioral;
```

-Simulation:

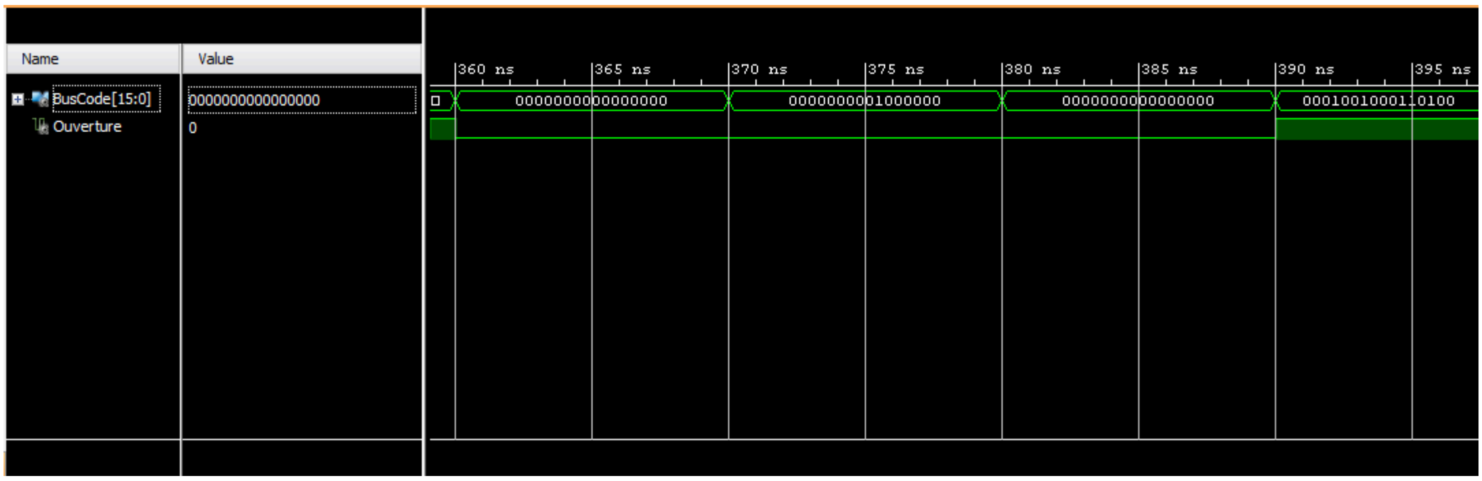
TestBench:



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

entity comparator_tb is
--  Port ( );
end comparator_tb;

architecture Behavioral of comparator_tb is
component Comparator is
Port (BusCode    : in  std_logic_vector(15 downto 0);
      Ouverture : out std_logic );
end component;
signal BusCode : std_logic_vector(15 downto 0) ;
signal Ouverture : std_logic;
begin
cmp: Comparator port map (BusCode => BusCode,Ouverture => Ouverture);
process
begin
BusCode <= "0000000000000000";
wait for 10 ns;
BusCode <= "0010000000000000";
wait for 10 ns;
BusCode <= "0000000000000000";
wait for 10 ns;
BusCode <= "0000000000000001";
wait for 10 ns;
BusCode <= "0000000000000000";
wait for 10 ns;
BusCode <= "0000000000000010";
wait for 10 ns;
```



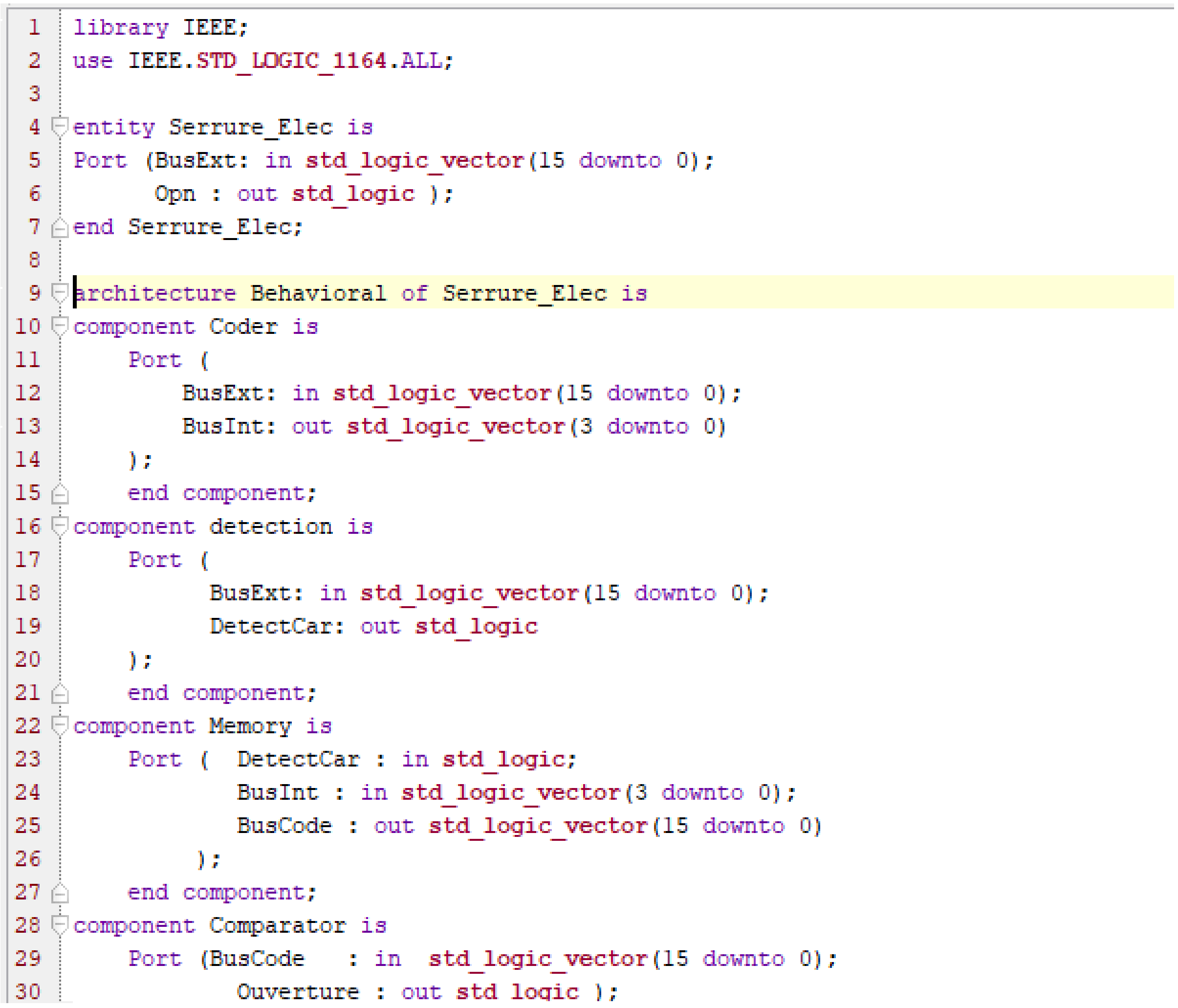
```vhdl
BusCode <= "0000000000000000";
wait for 10 ns;
BusCode <= "0001001000110100";
wait for 10 ns;
BusCode <= "0000000000000000";
wait for 10 ns;
BusCode <= "0000000001000000";
wait for 10 ns;
BusCode <= "0000000000000000";
wait for 10 ns;
BusCode <= "0001001000110100";
wait for 10 ns;
BusCode <= "0000000000000000";
wait for 10 ns;
BusCode <= "1000000000000000";
wait for 10 ns;
end process;
end Behavioral;

```



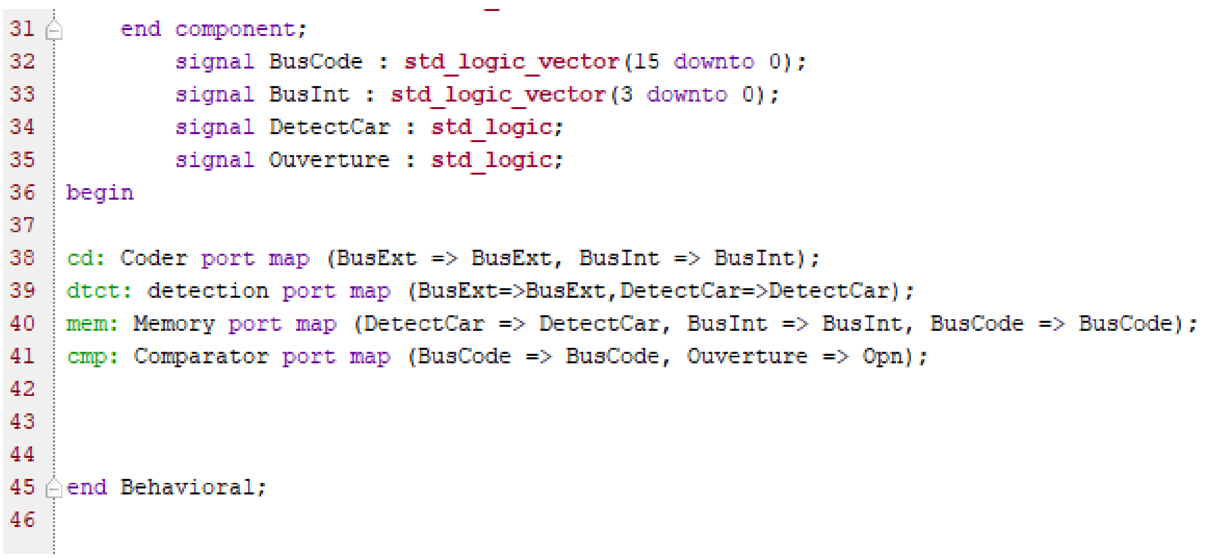
7)Assemblage:

-Schéma:

-Description VHDL:



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

entity Serrure_Elec is
Port (BusExt: in std_logic_vector(15 downto 0);
      Opn : out std_logic );
end Serrure_Elec;

architecture Behavioral of Serrure_Elec is
component Coder is
    Port (
        BusExt: in std_logic_vector(15 downto 0);
        BusInt: out std_logic_vector(3 downto 0)
    );
    end component;
component detection is
    Port (
          BusExt: in std_logic_vector(15 downto 0);
          DetectCar: out std_logic
    );
    end component;
component Memory is
    Port (  DetectCar : in std_logic;
            BusInt : in std_logic_vector(3 downto 0);
            BusCode : out std_logic_vector(15 downto 0)
         );
    end component;
component Comparator is
    Port (BusCode    : in  std_logic_vector(15 downto 0);
            Ouverture : out std_logic );
```

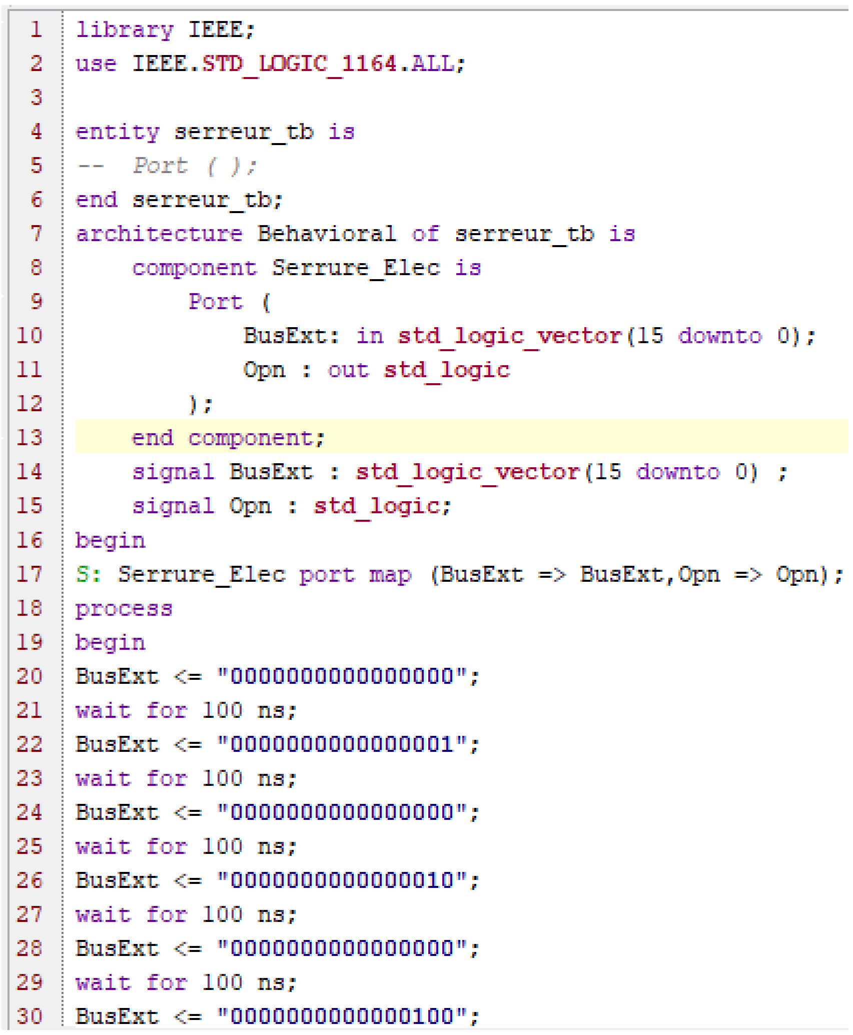


```vhdl
    end component;
        signal BusCode : std_logic_vector(15 downto 0);
        signal BusInt : std_logic_vector(3 downto 0);
        signal DetectCar : std_logic;
        signal Ouverture : std_logic;
begin

cd: Coder port map (BusExt => BusExt, BusInt => BusInt);
dtct: detection port map (BusExt=>BusExt,DetectCar=>DetectCar);
mem: Memory port map (DetectCar => DetectCar, BusInt => BusInt, BusCode => BusCode);
cmp: Comparator port map (BusCode => BusCode, Ouverture => Opn);



end Behavioral;

```

-Simulation:

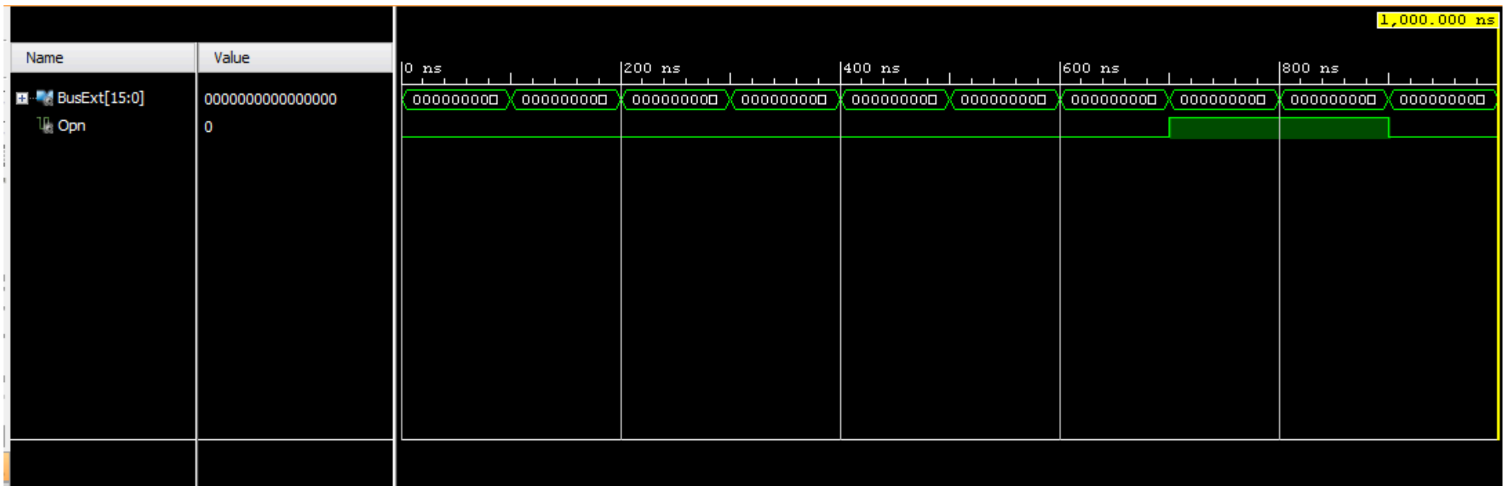
TestBench:

-Dans le testbench je donne un valeur à “0000000000000000” à BusExt entre 2 autres valeurs pour assimiler l'appui sur les boutons et permet le DetectCar d'alterner entre ‘0’ et ‘1’.



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

entity serreur_tb is
--  Port ( );
end serreur_tb;
architecture Behavioral of serreur_tb is
    component Serrure_Elec is
        Port (
            BusExt: in std_logic_vector(15 downto 0);
            Opn : out std_logic
        );
    end component;
    signal BusExt : std_logic_vector(15 downto 0) ;
    signal Opn : std_logic;
begin
S: Serrure_Elec port map (BusExt => BusExt,Opn => Opn);
process
begin
BusExt <= "0000000000000000";
wait for 100 ns;
BusExt <= "0000000000000001";
wait for 100 ns;
BusExt <= "0000000000000000";
wait for 100 ns;
BusExt <= "0000000000000010";
wait for 100 ns;
BusExt <= "0000000000000000";
wait for 100 ns;
BusExt <= "0000000000000100";
```

```
wait for 100 ns;
BusExt <= "0000000000000000";
wait for 100 ns;
BusExt <= "0000000000010000";
wait for 100 ns;
end process;

end Behavioral;

```



```
Name            Value
BusExt[15:0]    0000000000000000
Opn             0
```