

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:**

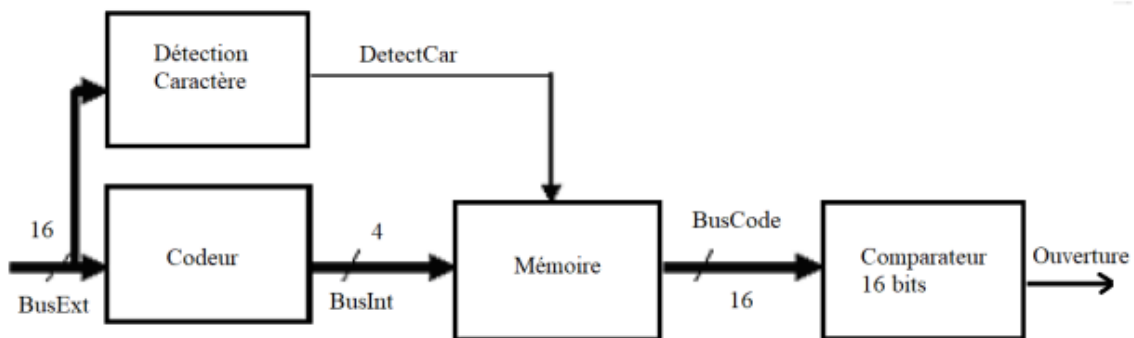


Figure 2 : Architecture globale du système

1)Codeur:

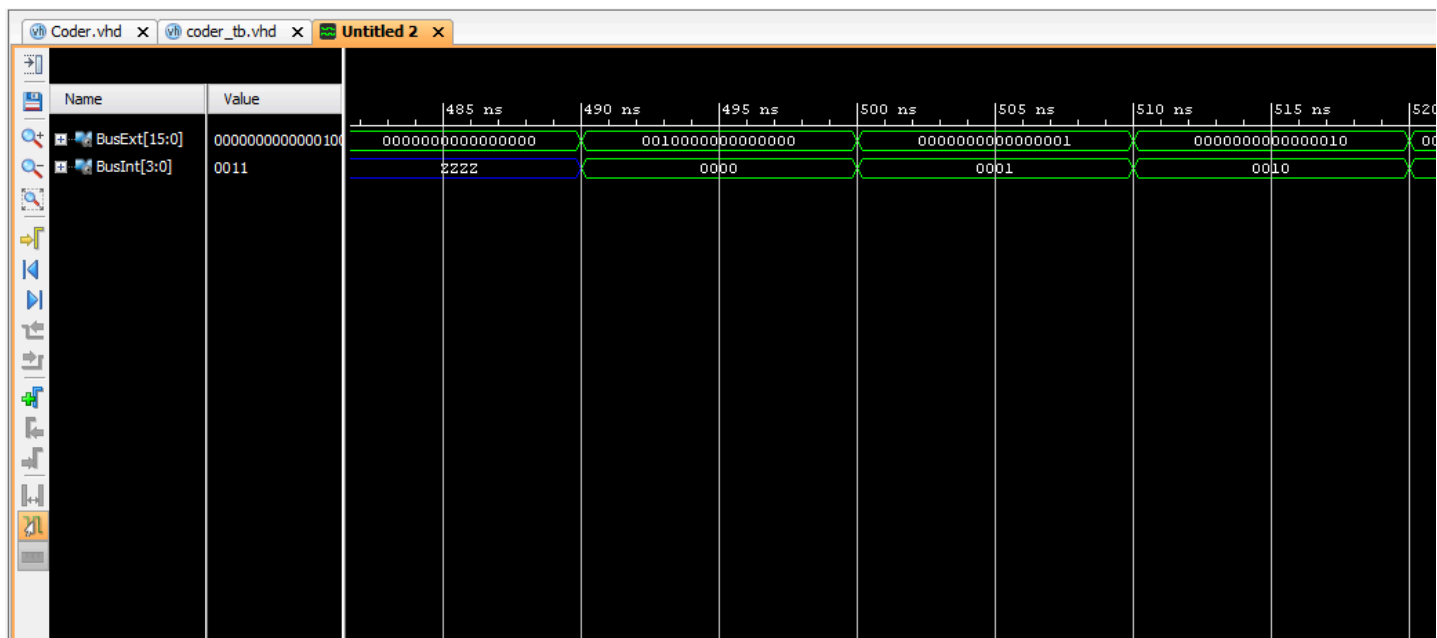
-Description VHDL:

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  entity Coder is
5  Port (BusExt: in std_logic_vector(15 downto 0);
6        BusInt: out std_logic_vector(3 downto 0)
7
8  );
9  end Coder;
10
11 architecture Behavioral of Coder is
12
13 begin
14   process (BusExt)
15   begin
16
17     case BusExt is
18       when "0000000000000000" => BusInt <= "ZZZZ";
19       when "0010000000000000" => BusInt <= "0000";
20       when "0000000000000001" => BusInt <= "0001";
21       when "0000000000000010" => BusInt <= "0010";
22       when "0000000000000100" => BusInt <= "0011";
23       when "0000000000001000" => BusInt <= "0100";
24       when "0000000000010000" => BusInt <= "0110";
25       when "0000000001000000" => BusInt <= "0111";
26       when "0000000100000000" => BusInt <= "1000";
27       when "0000001000000000" => BusInt <= "1001";
28       when "0000010000000000" => BusInt <= "1010";
29       when "0000000000001000" => BusInt <= "1011";
30       when "0000000010000000" => BusInt <= "1100";
31
32       when "0000100000000000" => BusInt <= "1101";
33       when "1000000000000000" => BusInt <= "1110";
34       when "0100000000000000" => BusInt <= "1111";
35       when others => BusInt <= "UUUU";
36     end case;
37   end process;
38
39 end Behavioral;
40
41
```

-Simulation:

TestBench:

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  entity coder_tb is
5  -- Port ( );
6  end coder_tb;
7
8  architecture Behavioral of coder_tb is
9
10     component Coder is
11         Port (
12             BusExt: in std_logic_vector(15 downto 0);
13             BusInt: out std_logic_vector(3 downto 0)
14         );
15     end component;
16
17     signal BusExt : std_logic_vector(15 downto 0);
18     signal BusInt : std_logic_vector(3 downto 0);
19
20 begin
21     S: Coder port map (BusExt => BusExt, BusInt => BusInt);
22     process
23     begin
24         BusExt <= "0000000000000000";
25         wait for 10 ns;
26         BusExt <= "0010000000000000";
27         wait for 10 ns;
28         BusExt <= "0000000000000001";
29         wait for 10 ns;
30         BusExt <= "0000000000000010";
31
32         wait for 10 ns;
33         BusExt <= "0000000000000100";
34         wait for 10 ns;
35         BusExt <= "00000000000010000";
36         wait for 10 ns;
37     end process;
38
39 end Behavioral;
```



2)Détection:

-Description VHDL:

```

1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4
5  entity detection is
6  Port (
7      BusExt: in std_logic_vector(15 downto 0);
8      DetectCar: out std_logic
9  );
10 end detection;
11
12 architecture Behavioral of detection is
13
14 begin
15 process (BusExt)
16 begin
17     if BusExt /= "0000000000000000" then
18         DetectCar <= '1';
19     else
20         DetectCar <= '0';
21     end if;
22 end process;
23
24 end Behavioral;
25

```

3)Bascule D active sur le front montant:

-Description VHDL:

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4
5  entity Bascule_D is
6  Port (    D : in  std_logic;
7          clk : in  std_logic;
8          rst : in  std_logic;
9          Q : out std_logic);
10 end Bascule_D;
11
12 architecture Behavioral of Bascule_D is
13
14 begin
15 process (clk, rst)
16     begin
17         if rst = '1' then
18             q <= '0';
19         elsif rising_edge(clk) then
20             q <= d;
21         end if;
22     end process;
23
24 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:

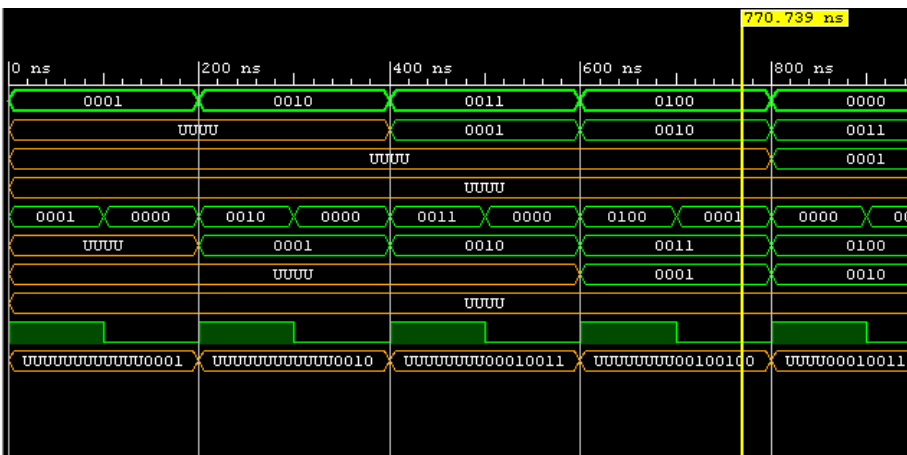
```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  entity Paral_Register is
5  Port (    D: in std_logic_vector(3 downto 0);
6          CLK : in std_logic;
7          Q : out std_logic_vector(3 downto 0)
8          );
9  end Paral_Register;
10
11  architecture Behavioral of Paral_Register is
12  component Bascule_D is
13  Port (    D : in  std_logic;
14          clk : in  std_logic;
15          rst : in  std_logic;
16          Q : out  std_logic);
17  end component;
18  signal Q0, Q1, Q2, Q3,C : std_logic;
19
20  begin
21  B0: Bascule_D port map (D(0),CLK,'0',Q0);
22  B1: Bascule_D port map (D(1),CLK,'0',Q1);
23  B2: Bascule_D port map (D(2),CLK,'0',Q2);
24  B3: Bascule_D port map (D(3),CLK,'0',Q3);
25  C<=not(CLK);
26  Q <= Q3 & Q2 & Q1 & Q0;
27
28  end Behavioral;
29
```

-Simulation:

TestBench:

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3  entity reg_tb is
4  -- Port ( );
5  end reg_tb;
6
7  architecture Behavioral of reg_tb is
8  component Paral_Register is
9      port (
10         D: in std_logic_vector(3 downto 0);
11         CLK : in std_logic;
12         Q : out std_logic_vector(3 downto 0)
13     );
14 end component;
15 signal reg1, reg2, reg3, reg4: std_logic_vector(3 downto 0);
16 signal I1,I2,I3,I4: std_logic_vector(3 downto 0);
17 signal DetectCar : std_logic:='0';
18 signal reg: std_logic_vector(15 downto 0);
19 begin
20 r1: Paral_Register port map (D => I1, CLK => DetectCar, Q => reg1);
21 r2: Paral_Register port map (D => I2, CLK => DetectCar, Q => reg2);
22 r3: Paral_Register port map (D => I3, CLK => DetectCar, Q => reg3);
23 r4: Paral_Register port map (D => I4, CLK => DetectCar, Q => reg4);
24 process (DetectCar)
25 begin
26 if DetectCar = '1' and DetectCar'event then
27     I4<= reg3;
28     I3<= reg2;
29     I2<= reg1;
30
```

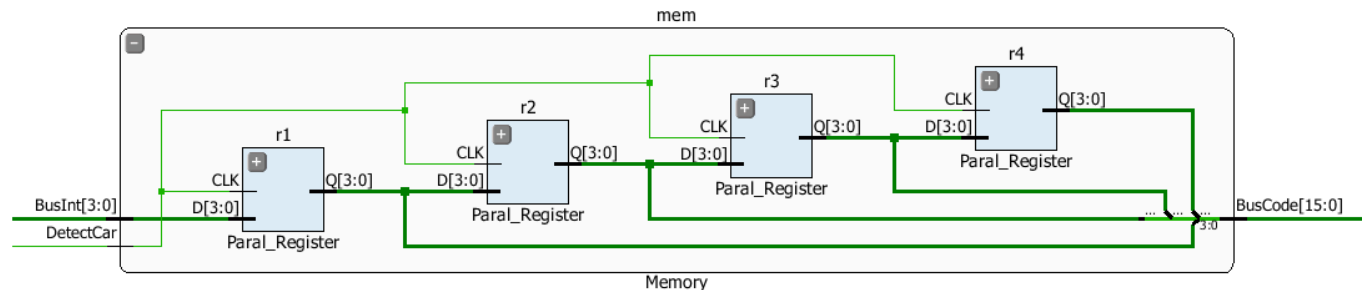
Name	Value
reg1[3:0]	0100
reg2[3:0]	0010
reg3[3:0]	UUUU
reg4[3:0]	UUUU
I1[3:0]	0001
I2[3:0]	0011
I3[3:0]	0001
I4[3:0]	UUUU
DetectCar	0
reg[15:0]	UUUUUUUU00100100



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:

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  entity Memory is
5  Port ( DetectCar : in std_logic;
6         BusInt : in std_logic_vector(3 downto 0);
7         BusCode : out std_logic_vector(15 downto 0)
8       );
9  end Memory;
10
11 architecture Behavioral of Memory is
12 component Paral_Register is
13     port (
14         D: in std_logic_vector(3 downto 0);
15         CLK : in std_logic;
16         Q : out std_logic_vector(3 downto 0)
17     );
18 end component;
19 signal reg1, reg2, reg3, reg4: std_logic_vector(3 downto 0);
20 signal I1,I2,I3,I4: std_logic_vector(3 downto 0);
21 signal B: std_logic_vector(15 downto 0);
22 begin
23     r1: Paral_Register port map (D => BusInt, CLK => DetectCar, Q => reg1);
24     r2: Paral_Register port map (D => reg1, CLK => DetectCar, Q => reg2);
25     r3: Paral_Register port map (D => reg2, CLK => DetectCar, Q => reg3);
26     r4: Paral_Register port map (D => reg3, CLK => DetectCar, Q => reg4);
27
28     BusCode <= reg4 & reg3 & reg2 & reg1;
29 end Behavioral;
```

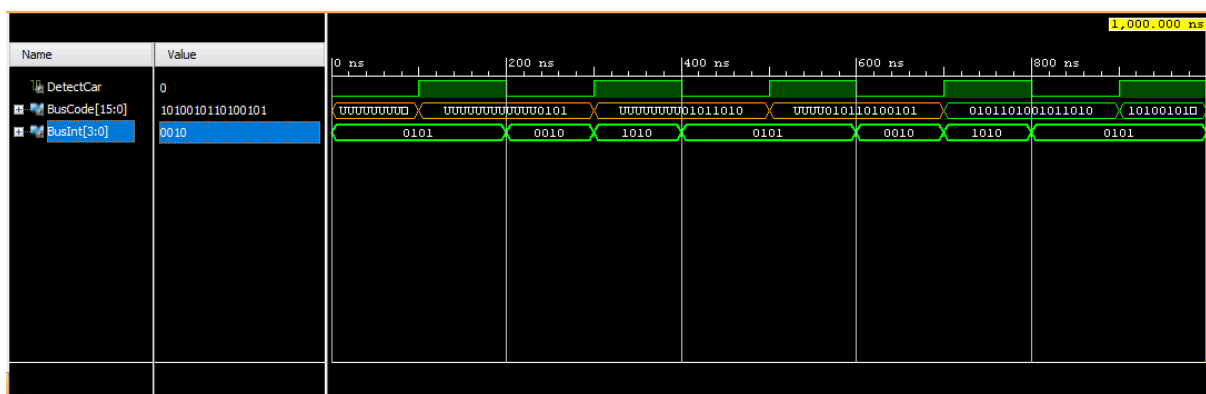
-Simulation:

TestBench:

```

1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3  entity mem_tb is
4  -- Port ( );
5  end mem_tb;
6  architecture Behavioral of mem_tb is
7      component Memory is
8          Port (
9              DetectCar : in std_logic;
10             BusInt : in std_logic_vector(3 downto 0);
11             BusCode : out std_logic_vector(15 downto 0)
12          );
13      end component;
14      signal DetectCar : std_logic := '1';
15      signal BusCode : std_logic_vector(15 downto 0);
16      signal BusInt : std_logic_vector(3 downto 0);
17      begin
18      mem: Memory port map (DetectCar => DetectCar, BusInt => BusInt, BusCode => BusCode);
19      process
20      begin
21          DetectCar<= not(DetectCar);
22          BusInt <= "0101";
23          wait for 100ns;
24          DetectCar<= not(DetectCar);
25          BusInt <= "0101";
26          ○ wait for 100ns;
27          ○ DetectCar<= not(DetectCar);
28          ○ BusInt <= "0010";
29          ○ wait for 100ns;
30          ○ 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.

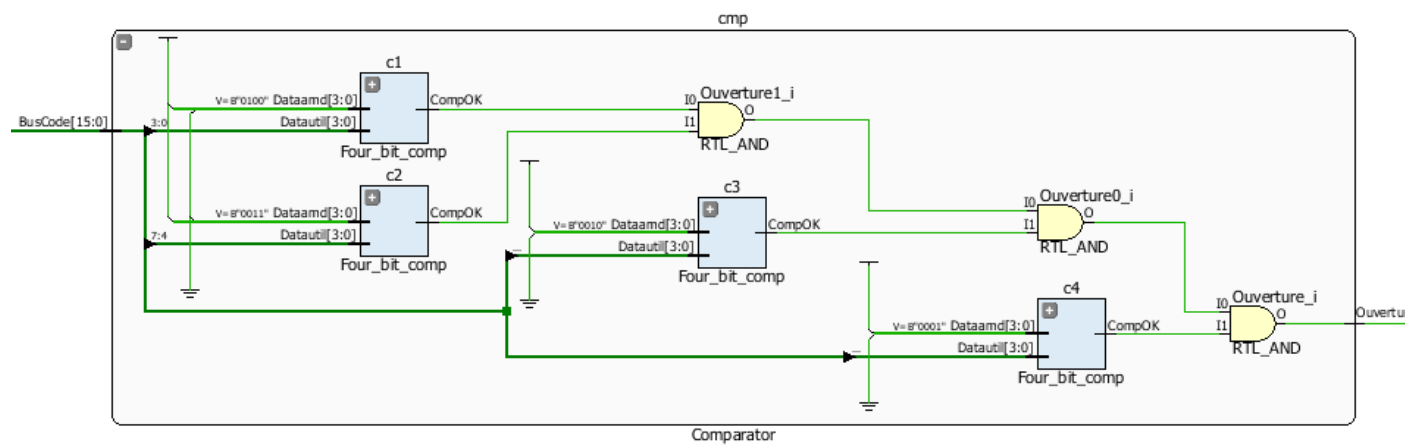
-Description VHDL:

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  entity Four_bit_comp is
5  Port ( Datautil, Dataamd : in std_logic_vector(3 downto 0);
6        CompOK : out std_logic);
7  end Four_bit_comp;
8
9  architecture Behavioral of Four_bit_comp is
10
11  begin
12  process (Datautil, Dataamd)
13      begin
14          if Datautil = Dataamd then
15              CompOK <= '1';
16          else
17              CompOK <= '0';
18          end if;
19      end process;
20
21  end Behavioral;
22
```

6) Comparateur 16 bits:

-Schéma du comparateur 16 bits:

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



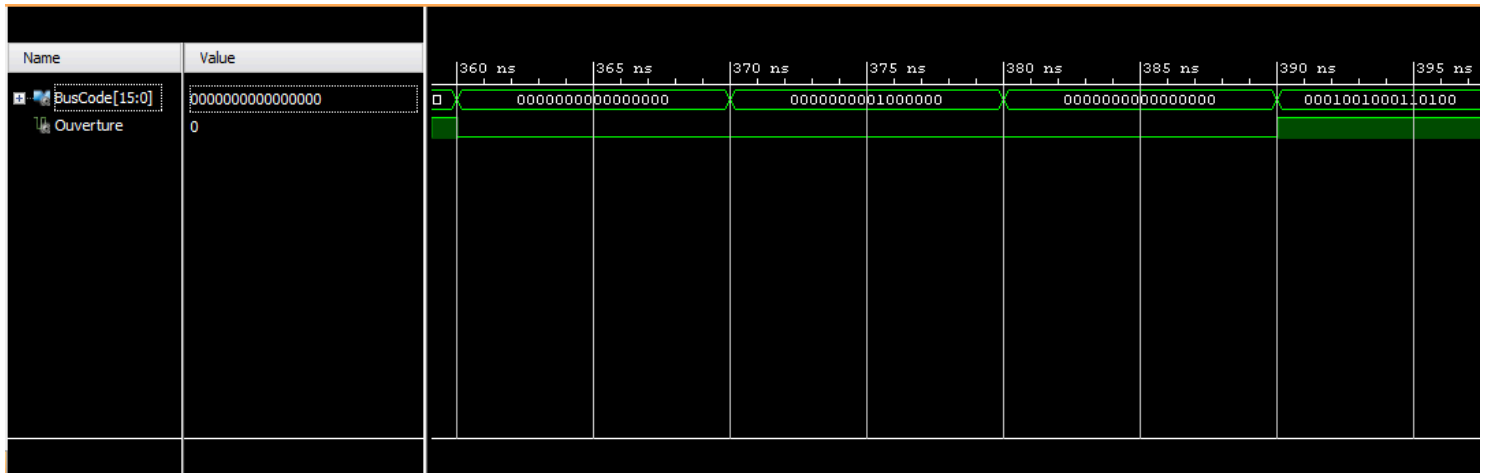
-Description VHDL:

```
1 library IEEE;
2 use IEEE.STD_LOGIC_1164.ALL;
3
4 entity Comparator is
5   Port (BusCode : in std_logic_vector(15 downto 0);
6         Ouverture : out std_logic );
7 end Comparator;
8
9 architecture Behavioral of Comparator is
10   component Four_bit_comp is
11     port (
12       Datautil : in std_logic_vector(3 downto 0);
13       Dataamd : in std_logic_vector(3 downto 0);
14       CompOK : out std_logic
15     );
16   end component;
17
18   signal comp_ok1, comp_ok2, comp_ok3, comp_ok4 : std_logic;
19   signal key: std_logic_vector(15 downto 0) := "0001001000110100";
20 begin
21   c1: Four_bit_comp port map(Datautil => BusCode(3 downto 0), Dataamd => key(3 downto 0), CompOK => comp_ok1);
22   c2: Four_bit_comp port map(Datautil => BusCode(7 downto 4), Dataamd => key(7 downto 4), CompOK => comp_ok2);
23   c3: Four_bit_comp port map(Datautil => BusCode(11 downto 8), Dataamd => key(11 downto 8), CompOK => comp_ok3);
24   c4: Four_bit_comp port map(Datautil => BusCode(15 downto 12), Dataamd => key(15 downto 12), CompOK => comp_ok4);
25
26   Ouverture <= comp_ok1 and comp_ok2 and comp_ok3 and comp_ok4;
27
28
29 end Behavioral;
```

-Simulation:

TestBench:

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  entity comparator_tb is
5  -- Port ( );
6  end comparator_tb;
7
8  architecture Behavioral of comparator_tb is
9  component Comparator is
10 Port (BusCode : in std_logic_vector(15 downto 0);
11       Ouverture : out std_logic );
12 end component;
13 signal BusCode : std_logic_vector(15 downto 0) ;
14 signal Ouverture : std_logic;
15 begin
16 cmp: Comparator port map (BusCode => BusCode,Ouverture => Ouverture);
17 process
18 begin
19 BusCode <= "0000000000000000";
20 wait for 10 ns;
21 BusCode <= "0010000000000000";
22 wait for 10 ns;
23 BusCode <= "0000000000000000";
24 wait for 10 ns;
25 BusCode <= "0000000000000001";
26 wait for 10 ns;
27 BusCode <= "0000000000000000";
28 wait for 10 ns;
29 BusCode <= "0000000000000010";
30 wait for 10 ns;
31 BusCode <= "0000000000000000";
32 wait for 10 ns;
33 BusCode <= "0001001000110100";
34 wait for 10 ns;
35 BusCode <= "0000000000000000";
36 wait for 10 ns;
37 BusCode <= "0000000001000000";
38 wait for 10 ns;
39 BusCode <= "0000000000000000";
40 wait for 10 ns;
41 BusCode <= "0001001000110100";
42 wait for 10 ns;
43 BusCode <= "0000000000000000";
44 wait for 10 ns;
45 BusCode <= "1000000000000000";
46 wait for 10 ns;
47 end process;
48 end Behavioral;
49
```



7)Assemblage:

-Schéma:

-Description VHDL:

```

1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  entity Serrure_Elec is
5  Port (BusExt: in std_logic_vector(15 downto 0);
6        Opn : out std_logic );
7  end Serrure_Elec;
8
9  architecture Behavioral of Serrure_Elec is
10 component Coder is
11     Port (
12         BusExt: in std_logic_vector(15 downto 0);
13         BusInt: out std_logic_vector(3 downto 0)
14     );
15 end component;
16 component detection is
17     Port (
18         BusExt: in std_logic_vector(15 downto 0);
19         DetectCar: out std_logic
20     );
21 end component;
22 component Memory is
23     Port ( DetectCar : in std_logic;
24           BusInt : in std_logic_vector(3 downto 0);
25           BusCode : out std_logic_vector(15 downto 0)
26     );
27 end component;
28 component Comparator is
29     Port (BusCode : in std_logic_vector(15 downto 0);
30           Ouverture : out std_logic );

```

```

31 end component;
32 signal BusCode : std_logic_vector(15 downto 0);
33 signal BusInt : std_logic_vector(3 downto 0);
34 signal DetectCar : std_logic;
35 signal Ouverture : std_logic;
36 begin
37
38 cd: Coder port map (BusExt => BusExt, BusInt => BusInt);
39 dtct: detection port map (BusExt=>BusExt,DetectCar=>DetectCar);
40 mem: Memory port map (DetectCar => DetectCar, BusInt => BusInt, BusCode => BusCode);
41 cmp: Comparator port map (BusCode => BusCode, Ouverture => Opn);
42
43
44
45 end Behavioral;
46

```

-Simulation:

TestBench:

-Dans le testbench je donne une 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'.

```

1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  entity serreur_tb is
5  -- Port ( );
6  end serreur_tb;
7  architecture Behavioral of serreur_tb is
8      component Serrure_Elec is
9          Port (
10             BusExt: in std_logic_vector(15 downto 0);
11             Opn : out std_logic
12          );
13      end component;
14      signal BusExt : std_logic_vector(15 downto 0) ;
15      signal Opn : std_logic;
16  begin
17      S: Serrure_Elec port map (BusExt => BusExt,Opn => Opn);
18  process
19  begin
20      BusExt <= "0000000000000000";
21      wait for 100 ns;
22      BusExt <= "0000000000000001";
23      wait for 100 ns;
24      BusExt <= "0000000000000000";
25      wait for 100 ns;
26      BusExt <= "0000000000000010";
27      wait for 100 ns;
28      BusExt <= "0000000000000000";
29      wait for 100 ns;
30      BusExt <= "0000000000000100";

```

```

31 wait for 100 ns;
32 BusExt <= "0000000000000000";
33 wait for 100 ns;
34 BusExt <= "0000000000010000";
35 wait for 100 ns;
36 end process;
37
38 end Behavioral;
39

```

