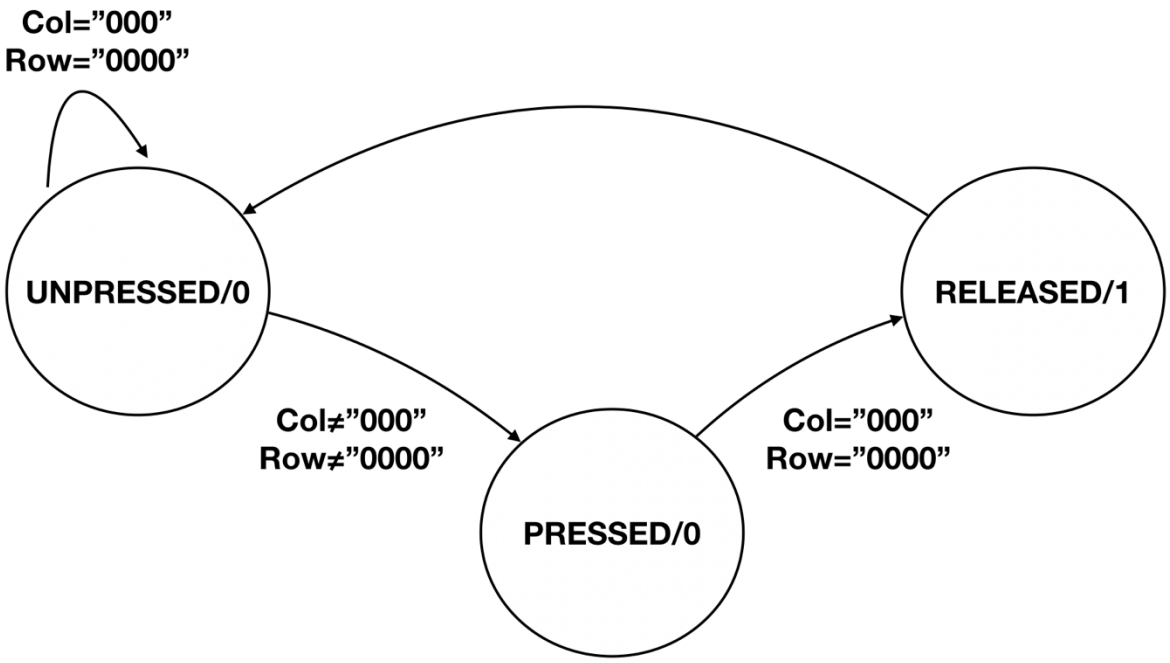
Keyboard:

- Questo modulo si occupa di registrare l'avvenuta pressione di un tasto ed opera nel seguente modo: prende in input la combinazione righe colonne di una matrice 4x3 (consideriamo il tastierino numerico come tale) , finché la combinazione riga="0000" colonna= "000" (che codifico come tasto non premuto) nessun tasto viene registrato, la pressione di un tasto verrà registrata correttamente quando riga ≠ "0000" e colonna ≠ "000" cioè ad avvenuto rilascio del pulsante premuto seguendo il seguente automa :



Codice:

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date:    17:09:46 01/05/2019
-- Design Name:
-- Module Name:    keyboard - Behavioral
-- Project Name:
-- Target Devices:
-- Tool versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------
```

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

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
library UNISIM;
use UNISIM.VComponents.all;

entity keyboard is
port (
                                        col : in std_logic_vector (0 to 2);
                                        row : in std_logic_vector (0 to 3);
                                        clk : in std_logic;
                                        rst : in std_logic;
                                        key : out  STD_LOGIC
                                        );
end keyboard;

architecture Behavioral of keyboard is
        type state is (notpressed, pressed, released);
                signal current_state,next_state : state;
                begin

        current_state_register: process(next_state,rst,clk)
        begin
        if rising_edge(clk) then
                if (rst = '1') then
                        current_state <= notpressed;
                else
                        current_state <= next_state;
                end if;
        end if;
        end process;

        process (clk)
                begin
                case current_state is

                        when notpressed =>
                                key <= '0';
                                if ((col /="000") and (row /="0000")) then
                                        next_state <= pressed;
                                else
                                        next_state <= notpressed;
                                end if;

                                when pressed =>
                                        key <= '0';
                                if ((col /="000") and (row /="0000")) then
                                        next_state <= pressed;
                                else
                                        next_state <= released;
                                end if;
```

```
                                        when released =>
                                                key <= '1';
                                                next_state <= notpressed;

                                        end case;
                        end process;

end Behavioral;
```

Test Bench:

simulo la pressione e la registrazione di un pulsante, prima gli input riga-colonna sono “0000” e “000” (nessun pulsante premuto) successivamente viene rilevata la pressione del pulsante ed infinite col rilascio del pulsante viene registrato l’inserimento di un pulsante.

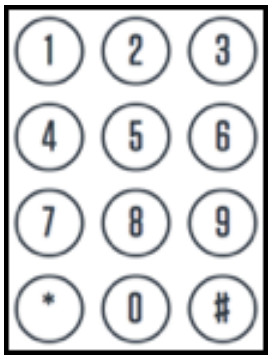
```
--------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date:   11:11:24 02/18/2019
-- Design Name:
-- Module Name:   /home/ise/Xilinx Projects/doorlock/keyboard_tb.vhd
-- Project Name:  doorlock
-- Target Device:
-- Tool versions:
-- Description:
--
-- VHDL Test Bench Created by ISE for module: keyboard
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
-- Notes:
-- This testbench has been automatically generated using types std_logic and
-- std_logic_vector for the ports of the unit under test.  Xilinx recommends
-- that these types always be used for the top-level I/O of a design in order
-- to guarantee that the testbench will bind correctly to the post-implementation
-- simulation model.
--------------------------------------------------------------------------------
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--USE ieee.numeric_std.ALL;

ENTITY keyboard_tb IS
END keyboard_tb;

ARCHITECTURE behavior OF keyboard_tb IS

   -- Component Declaration for the Unit Under Test (UUT)

   COMPONENT keyboard
```

```
    PORT(
         col : IN  std_logic_vector(0 to 2);
         row : IN  std_logic_vector(0 to 3);
         clk : IN  std_logic;
         rst : IN  std_logic;
         key : OUT  std_logic
        );
    END COMPONENT;


   --Inputs
   signal col : std_logic_vector(0 to 2) := (others => '0');
   signal row : std_logic_vector(0 to 3) := (others => '0');
   signal clk : std_logic := '0';
   signal rst : std_logic := '0';

              --Outputs
   signal key : std_logic;

   -- Clock period definitions
   constant clk_period : time := 10 ns;

BEGIN

              -- Instantiate the Unit Under Test (UUT)
   uut: keyboard PORT MAP (
          col => col,
          row => row,
          clk => clk,
          rst => rst,
          key => key
        );

   -- Clock process definitions
   clk_process :process
   begin
                          clk <= '0';
                          wait for clk_period/2;
                          clk <= '1';
                          wait for clk_period/2;
   end process;

   -- Stimulus process
   stim_proc: process
   begin
     -- hold reset state for 100 ns.
     wait for 100 ns;

     wait for clk_period*10;

      --insert stimulus here
                          col<="000";
                          row<="0000";
     wait for clk_period*10;
                          col<="001";
                          row<="0010";
                          wait for clk_period*10;
                          col<="000";
```

```
                    row<="0000";
                    wait;
  end process;
END;
```

Matrix_2_value:

Il componente matrix_2_value si occupa decodificare la combinazione righe-colonne della matrice ed ottenere il numero binario corrispondente seguendo la seguente codifica :
Consideriamo il seguente tastierino numerico come esempio , “1” corrisponderà alla combinazione riga = “1000” colonna=”100” ed in binario otterremo “0001” mentre “4” corrisponderà a riga=”0100” colonna=”100” ed in binario otterremo “0100” e così via.
La combinazione riga=”0000” e colonna = “000” viene considerata come nessun tasto premuto.



Codice:

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date:    11:53:10 02/18/2019
-- Design Name:
-- Module Name:    matrix_2_value - Behavioral
-- Project Name:
-- Target Devices:
-- Tool versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
```

```
--library UNISIM;
--use UNISIM.VComponents.all;

entity matrix_2_value is
   Port ( row : in  STD_LOGIC_VECTOR (0 to 3);
          col : in  STD_LOGIC_VECTOR (0 to 2);
          num : out  STD_LOGIC_VECTOR (0 to 3));
end matrix_2_value;

architecture Behavioral of matrix_2_value is
            begin
            process (row,col)
            begin
            if row = "1000" then
                        if  col = "100" then
                                    num <= "0001";
                        elsif  col = "010" then
                                    num <= "0010";
                        elsif col = "001" then
                                    num <= "0011";
                                    else
                        num <= "1111";
                        end if;
            elsif  row = "0100" then
                                    if  col = "100" then
                                    num <= "0100";
                                                elsif  col = "010" then
                                                num <= "0101";
                                    elsif col = "100" then
                                                num  <= "0110";
                                    else
                        num <= "1111";
                                                end if;

            elsif row = "0010" then
                                                if  col = "100" then
                                                            num <= "0111";
                                                elsif  col = "010" then
                                                            num <= "1000";
                                                elsif col = "001" then
                                                            num <= "1001";
                                                else
                                                num <= "1111";
                                                end if;

                        elsif row = "0001" then
                                    if  col = "100" then
                                    num <= "1010";
                        elsif  col = "010" then
                                    num <= "0000";
                        elsif col = "001" then
                                    num  <= "1011";
                                    else
                        num <= "1111";
                        end if;
                        end if;
                        end process;
```

```
end Behavioral;
```

Test Bench:

```
-- Company:
-- Engineer:
--
-- Create Date:   14:47:31 02/18/2019
-- Design Name:
-- Module Name:   /home/ise/Xilinx Projects/doorlock/matrix_2_value_tb.vhd
-- Project Name:  doorlock
-- Target Device:
-- Tool versions:
-- Description:
--
-- VHDL Test Bench Created by ISE for module: matrix_2_value
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
-- Notes:
-- This testbench has been automatically generated using types std_logic and
-- std_logic_vector for the ports of the unit under test.  Xilinx recommends
-- that these types always be used for the top-level I/O of a design in order
-- to guarantee that the testbench will bind correctly to the post-implementation
-- simulation model.
--------------------------------------------------------------------------------
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--USE ieee.numeric_std.ALL;

ENTITY matrix_2_value_tb IS
END matrix_2_value_tb;

ARCHITECTURE behavior OF matrix_2_value_tb IS

   -- Component Declaration for the Unit Under Test (UUT)

   COMPONENT matrix_2_value
   PORT(
        row : IN  std_logic_vector(0 to 3);
        col : IN  std_logic_vector(0 to 2);
        num : OUT  std_logic_vector(0 to 3)
       );
   END COMPONENT;


  --Inputs
  signal row : std_logic_vector(0 to 3) := (others => '0');
  signal col : std_logic_vector(0 to 2) := (others => '0');
```

```
        --Outputs
  signal num : std_logic_vector(0 to 3);
  -- No clocks detected in port list. Replace <clock> below with
  -- appropriate port name


BEGIN

        -- Instantiate the Unit Under Test (UUT)
  uut: matrix_2_value PORT MAP (
       row => row,
       col => col,
       num => num
      );

  -- Stimulus process
  stim_proc: process
  begin
    wait for 100 ns;
          row <= "1000";
          col <= "100";
   wait for 100 ns;
          row <= "0010";
          col <= "010";
   wait for 100 ns;
           row <= "1000";
          col <= "001";
   wait for 100 ns;
           row <= "0100";
          col <= "100";
   wait for 100 ns;
           row <= "0001";
          col <= "001";
   wait for 100 ns;
    wait;
  end process;

END;
```

CONTATORE:

Ho istanziato lo stesso contatore diverse volte in quanto avevo necessità di effettuare conteggi diversi, il codice del contatore è sempre lo stesso. Un contatore si occupa di contare il numero di cifre inserite, uno di contare il numero di tentativi effettuato e l'ultimo contatore si occupa di andare ad effettuare il conteggio per il match, che si incrementa solo quando il confronto tra la singola cifra inserita e la cifra della combinazione risulta uguale, in modo che se questo contatore è <4 allora il codice inserito risulta errato.

Codice :

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date:    15:06:44 02/18/2019
-- Design Name:
-- Module Name:    counter - Behavioral
```

```
-- Project Name:
-- Target Devices:
-- Tool versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity counter is
  port(
            clk: in std_logic;
            rst: in std_logic;
            increase : in std_logic;
            Output: out std_logic_vector(0 to 3));
end counter;

architecture Behavioral of counter is
  signal temp: std_logic_vector(0 to 3);
begin   process(clk,rst)
  begin
    if rst='1' then
      temp <= "0000";
    elsif(rising_edge(clk)) then
                                if (increase = '1') then
          temp <= temp + 1;
                                          else
                                                  temp<= temp;
        end if;
    end if;
  end process;
  Output <= temp;
end Behavioral;
```

Test Bench:

```
--------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
```

```
-- Create Date:   16:38:11 02/18/2019
-- Design Name:
-- Module Name:   /home/ise/Xilinx Projects/doorlock/counter_tb.vhd
-- Project Name:  doorlock
-- Target Device:
-- Tool versions:
-- Description:
--
-- VHDL Test Bench Created by ISE for module: counter
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
-- Notes:
-- This testbench has been automatically generated using types std_logic and
-- std_logic_vector for the ports of the unit under test.  Xilinx recommends
-- that these types always be used for the top-level I/O of a design in order
-- to guarantee that the testbench will bind correctly to the post-implementation
-- simulation model.
--------------------------------------------------------------------------------
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--USE ieee.numeric_std.ALL;

ENTITY counter_tb IS
END counter_tb;

ARCHITECTURE behavior OF counter_tb IS

   -- Component Declaration for the Unit Under Test (UUT)

   COMPONENT counter
   PORT(
       clk : IN  std_logic;
       rst : IN  std_logic;
       increase : IN  std_logic;
       Output : OUT  std_logic_vector(0 to 3)
      );
   END COMPONENT;


  --Inputs
  signal clk : std_logic := '0';
  signal rst : std_logic := '0';
  signal increase : std_logic := '0';

            --Outputs
  signal Output : std_logic_vector(0 to 3);

  -- Clock period definitions
  constant clk_period : time := 10 ns;
```

```
BEGIN

        -- Instantiate the Unit Under Test (UUT)
  uut: counter PORT MAP (
       clk => clk,
       rst => rst,
       increase => increase,
       Output => Output
      );

  -- Clock process definitions
  clk_process :process
  begin
                clk <= '0';
                wait for clk_period/2;
                clk <= '1';
                wait for clk_period/2;
  end process;


  -- Stimulus process
  stim_proc: process
  begin
    -- hold reset state for 100 ns.
                rst <= '1';
    wait for 100 ns;
                rst<='0';
          increase <= '1';
    wait for clk_period*10;

    -- insert stimulus here

    wait;
  end process;

END;
```

ROM:

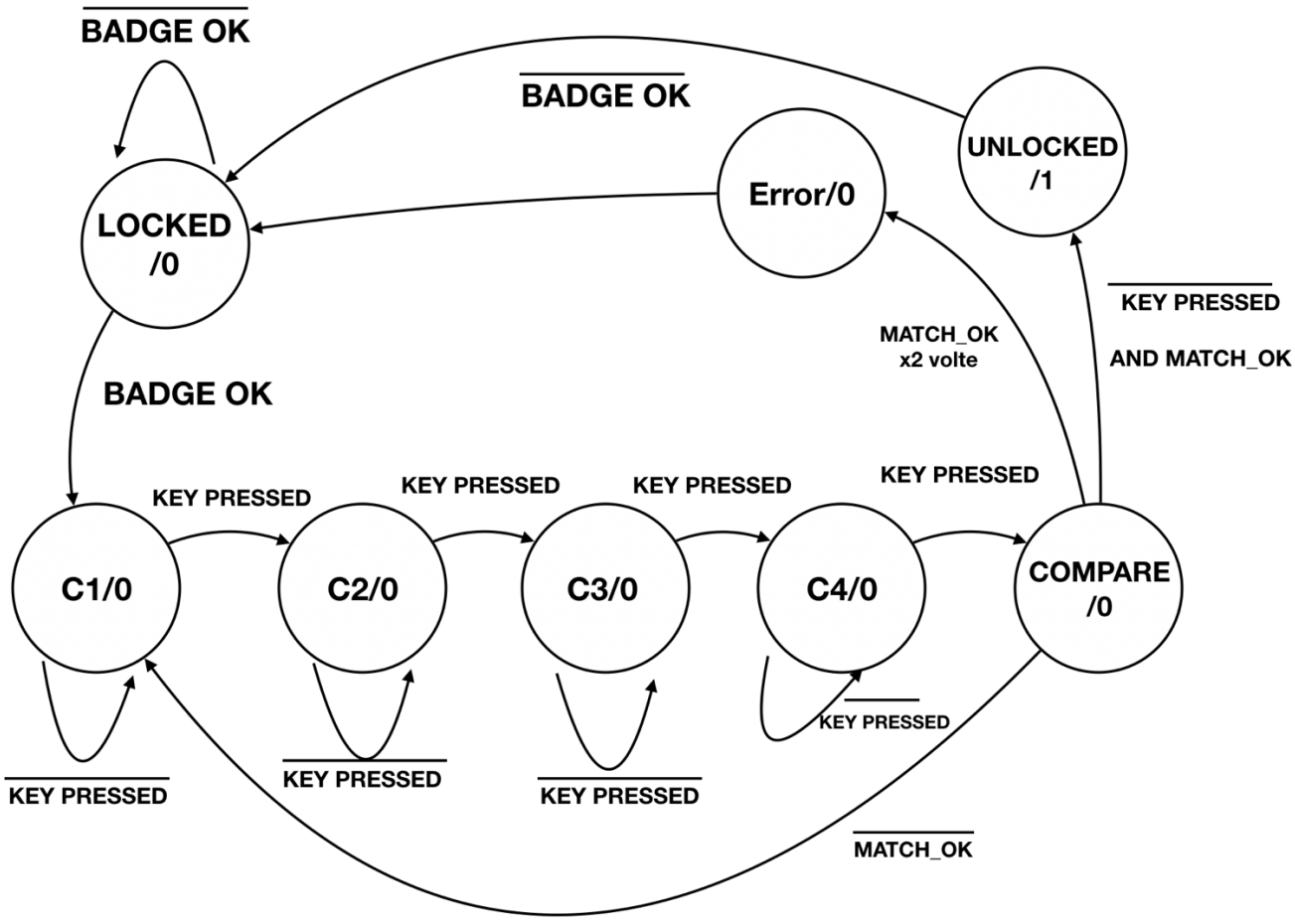
ROM è un acronimo di read only memory e sta ad indicare una memoria in cui non è possibile scrivere alcunché ma è possibile soltanto leggere i dati al suo interno, ho implementato una rom per memorizzare la sequenza corretta che apre la porta.

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date:    16:45:12 02/18/2019
-- Design Name:
-- Module Name:    rom - Behavioral
-- Project Name:
-- Target Devices:
-- Tool versions:
-- Description:
--
-- Dependencies:
--
```

```
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity rom is
port (CLK : in std_logic;
     EN : in std_logic;
     ADDR : in std_logic_vector(3 downto 0);
     DATA : out std_logic_vector (3 downto 0)
                          );
end rom;

architecture syn of rom is
   type rom_type is array (3 downto 0) of std_logic_vector(3 downto 0);
   signal ROM : rom_type:= ("1000","0100","0010","0001");

   signal rdata : std_logic_vector (3 downto 0);
begin

   rdata <= ROM(conv_integer(ADDR));

   process (CLK)
   begin
      if (falling_edge(clk)) then
         if (EN = '1') then
            DATA <= rdata;
         end if;
      end if;
   end process;

end syn;
```

Macchina a stati finiti:

Ho usato il seguente automa per rappresentare la macchina a stati finiti nel mio progetto :



Gli stati C1-C2-C3-C4 sono gli stati in cui la macchina si aspetta delle cifre, nello stato locked c'è bisogno di passare il badge per proseguire verso C1. Nello stato compare viene confrontata la sequenza inserita dall'utente

---

-- Company:

```
-- Engineer:
--
-- Create Date:    09:35:51 02/19/2019
-- Design Name:
-- Module Name:    fsm - Behavioral
-- Project Name:
-- Target Devices:
-- Tool versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity fsm is
   Port ( clk : in  STD_LOGIC;
                                keypressed : in std_logic;
                                badge : in std_logic_vector (1 downto 0);
                                count_in : in std_logic_vector (3 downto 0);
                                match_ok : in std_logic_vector (3 downto 0);
                                increase_num : out std_logic;
                                increase_try : out std_logic;
                                reset_try : out std_logic;
                                reset_num : out std_logic;
                                rst_match : out std_logic;
                                enable_rom : out std_logic;
                                enable_match : out std_logic;
                                unlock : out std_logic;
                                error_system : out std_logic;
          rst : in  STD_LOGIC
                                );
end fsm;

architecture Behavioral of fsm is
            type state is (locked, c1, c2, c3, c4, unlocked,error,compare);
            signal current_state,next_state : state;

            begin
                        current_state_register: process(next_state,rst,clk) --start
                                    begin
                                                if rising_edge(clk) then
                                                            if (rst = '1') then
                                                            current_state <= locked;
```

```
                        else
                                current_state <= next_state;
                        end if;
                end if;
end process;

process (clk) -- macchina sequenziale a stati finiti
begin
case current_state is
        when locked =>
                unlock <= '0';
                if (badge = "10") then
                        next_state <= c1;
                        increase_num <= '1';
                        increase_try <= '0';
                        enable_rom <= '1';
                        enable_match <= '0';
                        reset_num <= '1';
                        reset_try <= '1';
                        error_system <= '0';
                        rst_match <= '1';
                else
                        next_state <= locked;
                        increase_num <= '0';
                        increase_try <= '0';
                        enable_rom <= '1';
                        enable_match <= '0';
                        reset_num <= '1';
                        reset_try <= '1';
                        error_system <= '0';
                        rst_match <= '1';
                end if;

        when c1 =>
                unlock <= '0';
                if ((badge = "00" or badge = "10") and keypressed = '1') then
                        next_state <= c2;
                        increase_num <= '1';
                        increase_try <= '0';
                        enable_rom <= '1';
                        enable_match <= '1';
                        reset_num <= '0';
                        reset_try <= '0';
                        rst_match <= '0';
                else
                        next_state <= c1;
                        increase_num <= '0';
                        increase_try <= '0';
                        enable_rom <= '1';
                        enable_match <= '0';
                        reset_num <= '0';
                        reset_try <= '0';
                        rst_match <= '0';
                end if;

        when c2 =>
                unlock <= '0';
                if ((badge = "00" or badge = "10") and keypressed = '1') then
```

```
                next_state <= c3;
                increase_num <= '1';
                enable_rom <= '1';
                reset_num <= '0';
                enable_match <= '1';
                else
                next_state <= c2;
                increase_num <= '0';
                enable_rom <= '1';
                reset_num <= '0';
                enable_match <= '0';
        end if;

when c3 =>
        unlock <= '0';
        if ((badge = "00" or badge = "10") and keypressed = '1') then
                next_state <= c4;
                increase_num <= '1';
                enable_rom <= '1';
                enable_match <= '1';
                else
                next_state <= c3;
                increase_num <= '0';
                enable_rom <= '1';
                enable_match <= '0';
        end if;

when c4 =>
        unlock <= '0';
        if ((badge = "00" or badge = "10") and keypressed = '1') then
                        next_state <= compare;
                        increase_num <= '0';
                        enable_rom <= '1';
                        enable_match <= '1';
                        reset_num <='1';
                elsif (keypressed = '0') then
                        next_state <= c4;
                        increase_num <= '0';
                        enable_rom <= '1';
                        enable_match <= '0';
                else
                        next_state <= c4;

        end if;

when compare =>
        unlock <= '0';
        if (match_ok = "0011" and (badge = "00" or badge = "10") and keypressed = '0') then
                        next_state <= unlocked;
                        rst_match <= '1';
                else
                        rst_match <= '1';
                        increase_try <= '1';
                        if (count_in /="0001") then
                                next_state <= c1;
                        else
                                next_state <= error;
```

```
                                        end if;
                                end if;

                        when error =>
                                unlock <= '0';
                                error_system <='1';
                                increase_try <= '0';
                                reset_try <= '1';

                        when unlocked =>
                                unlock <= '1';
                                if (badge = "01") then
                                        error_system <= '1';
                                        reset_try <= '0';
                                else
                                        next_state <=unlocked;
                                end if;
                end case;
end process;

end Behavioral;
```

Test Bench:

```
--------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date:   15:28:55 02/19/2019
-- Design Name:
-- Module Name:   /home/ise/Xilinx Projects/doorlock/fsm_tb.vhd
-- Project Name:  doorlock
-- Target Device:
-- Tool versions:
-- Description:
--
-- VHDL Test Bench Created by ISE for module: fsm
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
-- Notes:
-- This testbench has been automatically generated using types std_logic and
-- std_logic_vector for the ports of the unit under test.  Xilinx recommends
-- that these types always be used for the top-level I/O of a design in order
-- to guarantee that the testbench will bind correctly to the post-implementation
-- simulation model.
--------------------------------------------------------------------------------
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
```

```
--USE ieee.numeric_std.ALL;

ENTITY fsm_tb IS
END fsm_tb;

ARCHITECTURE behavior OF fsm_tb IS

   -- Component Declaration for the Unit Under Test (UUT)

   COMPONENT fsm
   PORT(
       clk : IN  std_logic;
       keypressed : IN  std_logic;
       badge : IN  std_logic_vector(1 downto 0);
       count_in : IN  std_logic_vector(3 downto 0);
       match_ok : IN  std_logic_vector(3 downto 0);
       increase_num : OUT  std_logic;
       increase_try : OUT  std_logic;
       reset_try : OUT  std_logic;
       reset_num : OUT  std_logic;
       enable_rom : OUT  std_logic;
       enable_match : OUT  std_logic;
       unlock : OUT  std_logic;
       error_system : OUT  std_logic;
       rst : IN  std_logic
      );
   END COMPONENT;


  --Inputs
  signal clk : std_logic := '0';
  signal keypressed : std_logic := '0';
  signal badge : std_logic_vector(1 downto 0) := (others => '0');
  signal count_in : std_logic_vector(3 downto 0) := (others => '0');
  signal match_ok : std_logic_vector(3 downto 0) := (others => '0');
  signal rst : std_logic := '0';

            --Outputs
  signal increase_num : std_logic;
  signal increase_try : std_logic;
  signal reset_try : std_logic;
  signal reset_num : std_logic;
  signal enable_rom : std_logic;
  signal enable_match : std_logic;
  signal unlock : std_logic;
  signal error_system : std_logic;

  -- Clock period definitions
  constant clk_period : time := 10 ns;

BEGIN

            -- Instantiate the Unit Under Test (UUT)
  uut: fsm PORT MAP (
        clk => clk,
        keypressed => keypressed,
        badge => badge,
        count_in => count_in,
```

```
        match_ok => match_ok,
        increase_num => increase_num,
        increase_try => increase_try,
        reset_try => reset_try,
        reset_num => reset_num,
        enable_rom => enable_rom,
        enable_match => enable_match,
        unlock => unlock,
        error_system => error_system,
        rst => rst
      );

  -- Clock process definitions
  clk_process :process
  begin
                        clk <= '0';
                        wait for clk_period/2;
                        clk <= '1';
                        wait for clk_period/2;
  end process;


  -- Stimulus process
  stim_proc: process
   begin
                        rst <='1';
    -- hold reset state for 100 ns.
    wait for 100 ns;
                        rst <= '0';
                        badge <= "10";
                        keypressed <= '1';

    wait for clk_period*5;
            count_in <= "0001";
             wait for clk_period*5;
             rst <='1';
             wait for 10 ns;
             rst <='0';
    -- insert stimulus here

    wait;
  end process;

END;
```

MATCH:

Fa un confronto tra I dati inseriti tramite il tastierino numerico ed i dati memorizzati restituendo 1 se questo confronto è risultato positivo o 0 se è risultato negativo.

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date:    11:22:14 02/19/2019
```

```
-- Design Name:
-- Module Name:    match - Behavioral
-- Project Name:
-- Target Devices:
-- Tool versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity match is
   Port ( data : in  STD_LOGIC_VECTOR (3 downto 0);
          num : in  STD_LOGIC_VECTOR (3 downto 0);
          enable : in  STD_LOGIC;
          match_ok : out  STD_LOGIC);
end match;

architecture Behavioral of match is

begin
            process (enable,data,num)
            begin
                        if (enable='1') then
                                    if (data = num) then
                                                match_ok <= '1';
                                    else
                                                match_ok <='0';
                                    end if;
                        else
                                    match_ok<='0';
                        end if;
            end process;

end Behavioral;
```

Test Bench:

```
-- Tool versions:
-- Description:
--
-- VHDL Test Bench Created by ISE for module: match
```

```
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
-- Notes:
-- This testbench has been automatically generated using types std_logic and
-- std_logic_vector for the ports of the unit under test.  Xilinx recommends
-- that these types always be used for the top-level I/O of a design in order
-- to guarantee that the testbench will bind correctly to the post-implementation
-- simulation model.
--------------------------------------------------------------------------------
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--USE ieee.numeric_std.ALL;

ENTITY match_tb IS
END match_tb;

ARCHITECTURE behavior OF match_tb IS

   -- Component Declaration for the Unit Under Test (UUT)

   COMPONENT match
   PORT(
        data : IN  std_logic_vector(3 downto 0);
        num : IN  std_logic_vector(3 downto 0);
        enable : IN  std_logic;
        match_ok : OUT  std_logic
       );
   END COMPONENT;


  --Inputs
  signal data : std_logic_vector(3 downto 0) := (others => '0');
  signal num : std_logic_vector(3 downto 0) := (others => '0');
  signal enable : std_logic := '0';

            --Outputs
  signal match_ok : std_logic;
  -- No clocks detected in port list. Replace <clock> below with
  -- appropriate port name



BEGIN

            -- Instantiate the Unit Under Test (UUT)
  uut: match PORT MAP (
         data => data,
         num => num,
         enable => enable,
         match_ok => match_ok
```

```
      );


  -- Stimulus process
  stim_proc: process
  begin
    -- hold reset state for 100 ns.
    wait for 100 ns;
                        enable <='1';
                        data <= "0010";
                        num<= "0010";
                        wait for 100 ns;
    -- insert stimulus here
                        enable <='0';
                        data <= "0010";
                        num<= "0010";
                        wait for 100 ns;
                        enable <='1';
                        data <= "0110";
                        num<= "0010";
    wait;
  end process;

END;
```

FLAG:

serve a resettare tutto nello stato error, in modo da ritornare nello stato di partenza locked.

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date:    14:43:27 02/19/2019
-- Design Name:
-- Module Name:    flag_rst - Behavioral
-- Project Name:
-- Target Devices:
-- Tool versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
```

```
--library UNISIM;
--use UNISIM.VComponents.all;

entity flag_rst is
   Port ( error_system : in  STD_LOGIC;
          rst : out  STD_LOGIC);
end flag_rst;

architecture Behavioral of flag_rst is

begin
            with error_system select rst <=
                        '1' when '1',
                        '0' when '0',
                        '0' when others;
end Behavioral;
```

DOOR LOCK:

Nel modulo principale della mia macchina ho eseguito diversi port map in per usare gli altri moduli.

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date:    16:06:03 01/05/2019
-- Design Name:
-- Module Name:    door_lock - Behavioral
-- Project Name:
-- Target Devices:
-- Tool versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;



entity door_lock is
            Port ( badge : in  STD_LOGIC_vector (1 downto 0);
```

```
                                        col : in std_logic_vector (0 to 2);
                                        row : in std_logic_vector (0 to 3);
                                        clk : in std_logic;
                                        rst : in std_logic;
                                        unlock : out  STD_LOGIC
                                );
end door_lock;

architecture Behavioral of door_lock is
        signal keypressed : std_logic;
        signal rst_try : std_logic;
        signal rst_num : std_logic;
        signal rst_match : std_logic;
        signal increase_try : std_logic;
        signal increase_num : std_logic;
        signal count_in : std_logic_vector(3 downto 0);
        signal address_rom : std_logic_vector(3 downto 0);
        signal enable : std_logic;
        signal data_out : std_logic_vector(3 downto 0);
        signal num : STD_LOGIC_VECTOR (0 to 3);
        signal match_ok : std_logic;
        signal enable_match : std_logic;
        signal error_system : std_logic;
        signal allert : std_logic;
        signal match_out : std_logic_vector (3 downto 0);

        component keyboard port (
                                        col : in std_logic_vector (0 to 2);
                                        row : in std_logic_vector (0 to 3);
                                        clk : in std_logic;
                                        rst : in std_logic;
                                        key : out  STD_LOGIC
                                        );
        end component;

        component rom is
port (CLK : in std_logic;
    EN : in std_logic;
    ADDR : in std_logic_vector(3 downto 0);
    DATA : out std_logic_vector(3 downto 0))
                ;
end component;

component counter is
  port(
            clk: in std_logic;
            rst: in std_logic;
            increase : in std_logic;
            Output: out std_logic_vector(3 downto 0));
end component;

component matrix_2_value is
   Port ( row : in  STD_LOGIC_VECTOR (0 to 3);
       col : in  STD_LOGIC_VECTOR (0 to 2);
       num : out  STD_LOGIC_VECTOR (0 to 3));
end component;

component fsm is
```

```
   Port ( clk : in  STD_LOGIC;
                                keypressed : in std_logic;
                                badge : in std_logic_vector (1 downto 0);
                                count_in : in std_logic_vector (3 downto 0);
                                match_ok : in std_logic_vector (3 downto 0);
                                increase_num : out std_logic;
                                increase_try : out std_logic;
                                reset_try : out std_logic;
                                reset_num : out std_logic;
                                rst_match : out std_logic;
                                enable_rom : out std_logic;
                                enable_match : out std_logic;
                                unlock : out std_logic;
                                error_system : out std_logic;
          rst : in  STD_LOGIC
                                );
end component;

component match is
   Port ( data : in  STD_LOGIC_VECTOR (3 downto 0);
          num : in  STD_LOGIC_VECTOR (3 downto 0);
          enable : in  STD_LOGIC;
          match_ok : out  STD_LOGIC);
end component;

component flag_rst is
   Port ( error_system : in  STD_LOGIC;
          rst : out  STD_LOGIC);
end component;

component match_append is
   Port ( match_ok : in  STD_LOGIC;
          enable : in  STD_LOGIC;
                                last_value : in std_logic;
          correct_code : out  STD_LOGIC);
end component;
             begin

             input : keyboard port map (col,row,clk,rst,keypressed);
             value : matrix_2_value port map (row,col,num);
             try_counter  : counter port map (clk,rst_try,increase_try,count_in);
             num_counter : counter port map (clk,rst_num,increase_num,address_rom);
             memory : rom port map (clk,enable,address_rom,data_out);
             machine : fsm port map
(clk,keypressed,badge,count_in,match_out,increase_num,increase_try,rst_try,rst_num,rst_match,enable,enable_mat
ch,unlock,error_system,allert);
             compare : match port map (data_out,num,enable_match,match_ok);
             flag : flag_rst port map (error_system,allert);
             match_counter  : counter port map (clk,rst_match,match_ok,match_out);

end Behavioral;
```

Test Bench:

Nel file di test mostro il funzionamento completo della macchina. Prima viene inserita una serie di ingressi errata in modo da testare la gestione dei tentativi della macchina (è possibile tentare massimo 2 volte prima di dover reinserire il badge) quindi prima la macchina torna nello stato C1 dove si aspetta di nuovo l'inserimento della prima cifra, una

volta inserita la combinazione corretta la porta si apre e si richiude correttamente al seguito del passaggio del badge da destra.

```
--------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date:   16:04:29 02/19/2019
-- Design Name:
-- Module Name:   /home/ise/Xilinx Projects/doorlock/door_lock_tb.vhd
-- Project Name:  doorlock
-- Target Device:
-- Tool versions:
-- Description:
--
-- VHDL Test Bench Created by ISE for module: door_lock
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
-- Notes:
-- This testbench has been automatically generated using types std_logic and
-- std_logic_vector for the ports of the unit under test.  Xilinx recommends
-- that these types always be used for the top-level I/O of a design in order
-- to guarantee that the testbench will bind correctly to the post-implementation
-- simulation model.
--------------------------------------------------------------------------------
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--USE ieee.numeric_std.ALL;

ENTITY door_lock_tb IS
END door_lock_tb;

ARCHITECTURE behavior OF door_lock_tb IS

   -- Component Declaration for the Unit Under Test (UUT)

   COMPONENT door_lock
   PORT(
        badge : IN  std_logic_vector(1 downto 0);
        col : IN  std_logic_vector(0 to 2);
        row : IN  std_logic_vector(0 to 3);
        clk : IN  std_logic;
        rst : IN  std_logic;
        unlock : OUT  std_logic
       );
   END COMPONENT;


  --Inputs
  signal badge : std_logic_vector(1 downto 0) := (others => '0');
```

```
  signal col : std_logic_vector(0 to 2) := (others => '0');
  signal row : std_logic_vector(0 to 3) := (others => '0');
  signal clk : std_logic := '0';
  signal rst : std_logic := '0';

          --Outputs
  signal unlock : std_logic;

  -- Clock period definitions
  constant clk_period : time := 10 ns;

BEGIN

          -- Instantiate the Unit Under Test (UUT)
  uut: door_lock PORT MAP (
       badge => badge,
       col => col,
       row => row,
       clk => clk,
       rst => rst,
       unlock => unlock
      );

  -- Clock process definitions
  clk_process :process
  begin
                      clk <= '0';
                      wait for clk_period/2;
                      clk <= '1';
                      wait for clk_period/2;
  end process;


  -- Stimulus process
  stim_proc: process
  begin
    -- hold reset state for 100 ns.
                      rst <= '1';
    wait for 100 ns;
                      rst <= '0';
                      badge <= "10";
                      row <= "1000";
                      col <= "100";
    wait for clk_period*10;
                      row <= "0000";
                      col <= "000";
    wait for clk_period*10;
                      row <= "0001";
                      col <= "100";
                          wait for clk_period*10;
                      row <= "0000";
                      col <= "000";
    wait for clk_period*10;
                      row <= "1000";
                      col <= "100";
                      wait for clk_period*10;
                      row <= "0000";
                      col <= "000";
```

```
    wait for clk_period*10;
                        row <= "1000";
                        col <= "010";
                        wait for clk_period*10;
                        row <= "0000";
                        col <= "000";
                        wait for clk_period*10;
                        row <= "1000";
                        col <= "100";
    wait for clk_period*10;
                        row <= "0000";
                        col <= "000";
    wait for clk_period*10;
                        row <= "1000";
                        col <= "010";
                            wait for clk_period*10;
                        row <= "0000";
                        col <= "000";
    wait for clk_period*10;
                        row <= "0100";
                        col <= "100";
                            wait for clk_period*10;
                        row <= "0000";
                        col <= "000";
    wait for clk_period*10;
                        row <= "0010";
                        col <= "010";
                        wait for clk_period*10;
                        row <= "0000";
                        col <= "000";
                        wait for clk_period*10;
                        badge <= "01";
                        wait for clk_period*10;
    wait;
  end process;

END;
```