Electronic Security Management of Basic Module Based on VHDL Language

Yuan Huang[1, a, *], Fujun Wu[2]

[1]School of Wuhan University of Technology, Wuhan, China

[2]School of Northeastern University, Qinhuangdao, China

[a]huang_yuan1@163.com

*Corresponding author

Keywords: Electronic Security Management, Basic Module, VHDL Language

Abstract: The electronic security management has an important impact in our life. Managing electronic security in an effective way is becoming more and more important. In this paper VHDL language is used to manage the button module and display module of the electronic security system which are the basic module to optimize performance and improve its operating efficiency.

1. Introduction

The electronic security management has an important impact in our life. Managing electronic security in an effective way is becoming more and more important. The button module and display module are the basic modules in the electronic security management. In this paper VHDL language is used to manage the button module and display module of the electronic security system to optimize performance and improve its operating efficiency. We will select the button module and the display module for research, stipulate the function of each module and the interface between each module, and combine the modules with the schematic language to simulate and complete the electronic security management of the basic module.

2. Distribution management of electronic security systems in the button module

The input part of the button module includes a clock signal, a reset signal, and a 16-bit key input signal; the output part transmits the key output signal to the control module. We use the button module to improve the management of safety performance and improve operational efficiency.

The following is the VHDL language source for the button module.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity button_model is
    Port (
        CLK             :in std_logic;
        RSTn            :in std_logic;
        Pin_In          :in std_logic_vector(15 downto 0);
        Pin_Out         :out std_logic_vector(15 downto 0)
        );
end button_model;
architecture button_model of button_model is
  signal rPin_Out    :std_logic_vector(15 downto 0);
  signal LastPin_In  :std_logic_vector(15 downto 0);
    signal t:              integer range 0 to 15;
begin
```

  DOI: 10.25236/icemit.2019.017

```
process(CLK,RSTn)
begin
	if RSTn='0' then
		rPin_Out <= "0000000000000000";
	elsif CLK'event and CLK='1'then
		LastPin_In <= Pin_In;
		if Pin_In(0)='0' and LastPin_In(0)='1' then
			rPin_Out(0) <= '1';
		else
			rPin_Out(0) <= '0';
		end if;
		if Pin_In(1)='0' and LastPin_In(1)='1' then
			rPin_Out(1) <= '1';
		else
			rPin_Out(1) <= '0';
		end if;
		if Pin_In(1)='0' and LastPin_In(1)='1' then
			rPin_Out(1) <= '1';
		else
			rPin_Out(1) <= '0';
		end if;
		if Pin_In(2)='0' and LastPin_In(2)='1' then
			rPin_Out(2) <= '1';
		else
			rPin_Out(2) <= '0';
		end if;
		if Pin_In(3)='0' and LastPin_In(3)='1' then
			rPin_Out(3) <= '1';
		else
			rPin_Out(3) <= '0';
		end if;
		if Pin_In(4)='0' and LastPin_In(4)='1' then
			rPin_Out(4) <= '1';
		else
			rPin_Out(4) <= '0';
		end if;
		if Pin_In(5)='0' and LastPin_In(5)='1' then
			rPin_Out(5) <= '1';
		else
			rPin_Out(5) <= '0';
		end if;
		if Pin_In(6)='0' and LastPin_In(6)='1' then
			rPin_Out(6) <= '1';
		else
			rPin_Out(6) <= '0';
		end if;
		if Pin_In(7)='0' and LastPin_In(7)='1' then
			rPin_Out(7) <= '1';
		else
			rPin_Out(7) <= '0';
		end if;
		if Pin_In(8)='0' and LastPin_In(8)='1' then
```

```
            rPin_Out(8) <= '1';
        else
            rPin_Out(8) <= '0';
        end if;
        if Pin_In(9)='0' and LastPin_In(9)='1' then
            rPin_Out(9) <= '1';
        else
            rPin_Out(9) <= '0';
        end if;
        if Pin_In(10)='0' and LastPin_In(10)='1' then
            rPin_Out(10) <= '1';
        else
            rPin_Out(10) <= '0';
        end if;
        if Pin_In(11)='0' and LastPin_In(11)='1' then
            rPin_Out(11) <= '1';
        else
            rPin_Out(11) <= '0';
        end if;
        if Pin_In(12)='0' and LastPin_In(12)='1' then
            rPin_Out(12) <= '1';
        else
            rPin_Out(12) <= '0';
        end if;
        if Pin_In(13)='0' and LastPin_In(13)='1' then
            rPin_Out(13) <= '1';
        else
            rPin_Out(13) <= '0';
        end if;
        if Pin_In(14)='0' and LastPin_In(14)='1' then
            rPin_Out(14) <= '1';
        else
            rPin_Out(14) <= '0';
        end if;
        if Pin_In(15)='0' and LastPin_In(15)='1' then
            rPin_Out(15) <= '1';
        else
            rPin_Out(15) <= '0';
        end if;
    end if;
  end process;
  Pin_Out <= rPin_Out;
end button_model;
```

The timing simulation is followed.

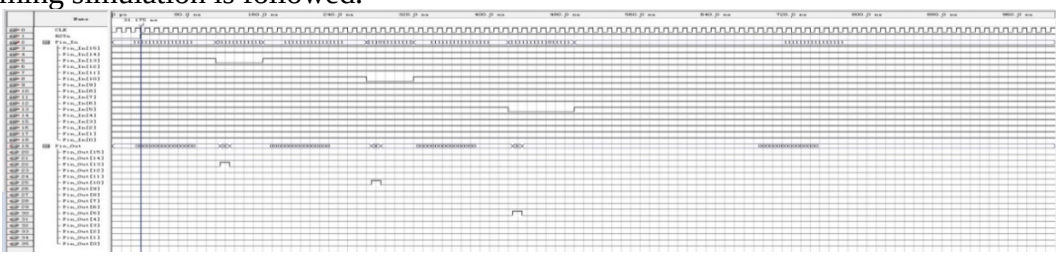

Figure 1. The timing simulation in the button module management.

3. Distribution management of electronic security systems in the display module

We use the code program to design the display module to achieve system visibility and improve management efficiency. The display module includes a dynamic scanning circuit and a twelve-segment digital tube display circuit. And the timing simulation is followed.

The following is the VHDL language source for the display module.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity smg_model is
    Port (
        CLK                 :in std_logic;
        Number_Data             :in std_logic_vector(47 downto 0 );
        Row_Scan_Sig        :out std_logic_vector(7 downto 0);
        Column_Scan_Sig     :out std_logic_vector(11 downto 0)
      );
end smg_model;
architecture smg_model of smg_model is
  type ram is array(0 to 9) of std_logic_vector(7 downto 0);
  constant smg :ram :=(
        (x"C0"),(x"F9"),(x"A4"),(x"B0"),(x"99"),(x"92"),(x"82"),(x"F8"),(x"80"),(x"90")
--0-9
                          );
    signal rData :std_logic_vector(7 downto 0);
    signal rColumn_Scan_Sig :std_logic_vector(11 downto 0);
    signal t: integer range 0 to 15;
begin
  process(CLK)
    variable Count1: integer range 0 to 2000;
  begin
    if CLK'event and CLK='1'then
        if Count1=10 then
            Count1 := 0;
            if t=11 then
                t <= 0;
            else
                t <= t + 1;
            end if;
        else
            Count1 := Count1 + 1;
        end if;
        case t is
            when 0 =>    rData <= smg(conv_integer(Number_Data(3 downto 0)) );
                         rColumn_Scan_Sig <= "111111111110";
            when 1 =>    rData <= smg(conv_integer(Number_Data(7 downto 4)) );
                         rColumn_Scan_Sig <= "111111111101";
            when 2 =>    rData <= smg(conv_integer(Number_Data(11 downto 8)) );
                         rColumn_Scan_Sig <= "111111111011";
            when 3 =>    rData <= smg(conv_integer(Number_Data(15 downto 12)) );
                         rColumn_Scan_Sig <= "111111110111";
            when 4 =>    rData <= smg(conv_integer(Number_Data(19 downto 16)) );
```

```
                    rColumn_Scan_Sig <= "111111101111";
            when 5 =>   rData <= smg(conv_integer(Number_Data(23 downto 20)) );
                    rColumn_Scan_Sig <= "111111011111";
            when 6 =>   rData <= smg(conv_integer(Number_Data(27 downto 24)) );
                    rColumn_Scan_Sig <= "111110111111";
            when 7 =>   rData <= smg(conv_integer(Number_Data(31 downto 28)) );
                    rColumn_Scan_Sig <= "111101111111";
            when 8 =>   rData <= smg(conv_integer(Number_Data(35 downto 32)) );
                    rColumn_Scan_Sig <= "111011111111";
            when 9 =>   rData <= smg(conv_integer(Number_Data(39 downto 36)) );
                    rColumn_Scan_Sig <= "110111111111";
            when 10 =>    rData <= smg(conv_integer(Number_Data(43 downto 40)) );
                    rColumn_Scan_Sig <= "101111111111";
            when 11 =>    rData <= smg(conv_integer(Number_Data(47 downto 44)) );
                    rColumn_Scan_Sig <= "011111111111";
            when others=>
                    rData <= "11111111";
                    rColumn_Scan_Sig <= "111111111111";
        end case;
    end if;
  end process;
Row_Scan_Sig <= rData;
Column_Scan_Sig <= rColumn_Scan_Sig;
end smg_model;
```

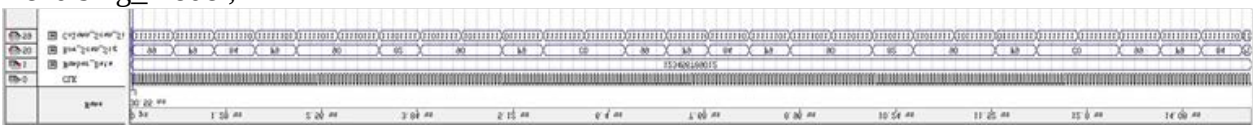

Figure 2. The timing simulation in the display module management.

4. Conclusion

Through the program coding, the button module has the functions of a clock input circuit, a reset circuit and a key input circuit. The display module mainly provides the management of the dynamic scanning circuit and the twelve-segment digital tube display circuit. We can manage the electronic security effectively in this way and provide an example of case for relevant management.

References

[1] H Jizu, Electronic Design Automation Based on VHDL Language and Its Application, Information Communication, 11 (2016) 74-75.

[2] X Han, Li Jia. VHDL language for electronic design automation and its application, Computer fans, 10 (2018) 233.

[3] L.Yingjie, Digital Clock Design Based on VHDL, Modern Economic Information, 12 (2018) 43.