时钟分频

----------------------------------------------------------------------------------

-- Engineer: switch\_swq

-- Create Date: 2024/04/18 13:02:24

----------------------------------------------------------------------------------

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity clk\_div is

generic(

DIV\_NUM:integer:=10000

);

Port (

rst:in std\_logic;

clk\_in:in std\_logic;

clk\_out:out std\_logic

);

end clk\_div;

architecture Behavioral of clk\_div is

begin

process(rst,clk\_in)

variable count:integer:=0;

begin

if(rst='1')then count:=0;

elsif(clk\_in='1'and clk\_in'event)then

count:=count+1;

if(count<=DIV\_NUM/2)then

clk\_out<='1';

elsif(count>DIV\_NUM/2 and count<DIV\_NUM)then

clk\_out<='0';

elsif(count>=DIV\_NUM)then

count:=0;

end if;

end if;

end process;

end Behavioral;

按键消抖

----------------------------------------------------------------------------------

-- Engineer: switch\_swq

-- Create Date: 2024/04/18 13:02:24

----------------------------------------------------------------------------------

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity key\_stroke is

generic(CLK\_FRE:integer:=100000000);

Port (

clk:in std\_logic;

reset:in std\_logic;

key\_in:in std\_logic;

output:out std\_logic

);

end key\_stroke;

architecture Behavioral of key\_stroke is

type states is(s0,s1,s2,s3);

signal state:states;

begin

process(reset,clk,key\_in)

variable count\_num:integer:=3\*CLK\_FRE/1000;--delay 3ms

variable count:integer:=0;

begin

if reset='0'then

state<=s0;

count:=0;

output<='0';

elsif(clk='1'and clk'event)then

case state is

when s0=>if(key\_in='1')then count:=0;output<='0';state<=s1;end if;

when s1=>

count:=count+1;

if (count>=count\_num) then state<=s2; end if;

when s2=>

if(key\_in='1')then output<='1';state<=s3;

--if(key\_in='1')then output<='1';state<=s0;--one clk cycle

elsif(key\_in='0')then state<=s0;

end if;

when s3=>if(key\_in='0')then state<=s0;end if;

end case;

end if;

end process;

end Behavioral;

数码管显示

----------------------------------------------------------------------------------

-- Engineer: switch\_swq

-- Create Date: 2024/04/18 13:02:24

----------------------------------------------------------------------------------

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.STD\_LOGIC\_UNSIGNED.ALL;

entity seg\_dis is

port(

rst,clk:in std\_logic;

data\_in\_A,data\_in\_B,data\_in\_C,data\_in\_D:in std\_logic\_vector(15 downto 0);

seg\_dig:out std\_logic\_vector(15 downto 0);

seg\_data:out std\_logic\_vector(7 downto 0)--管脚连接顺序：dp,g,f,e,d,c,b,a

);

end seg\_dis;

architecture Behavioral of seg\_dis is

signal data\_in\_line: std\_logic\_vector(3 downto 0);

type states is(s0,s1,s2,s3,s4,s5,s6,s7,s8,s9,s10,s11,s12,s13,s14,s15);

signal state,next\_state:states;

begin

process(rst,clk)

begin

if(rst='1')then state<=s0;

elsif(clk='1'and clk'event)then

state<=next\_state;

end if;

end process;

process(state)

begin

case state is

when s0 => seg\_dig<=not"0000000000000001"; data\_in\_line<=data\_in\_A(3 downto 0); next\_state<=s1;

when s1 => seg\_dig<=not"0000000000000010"; data\_in\_line<=data\_in\_A(7 downto 4); next\_state<=s2;

when s2 => seg\_dig<=not"0000000000000100"; data\_in\_line<=data\_in\_A(11 downto 8); next\_state<=s3;

when s3 => seg\_dig<=not"0000000000001000"; data\_in\_line<=data\_in\_A(15 downto 12); next\_state<=s4;

when s4 => seg\_dig<=not"0000000000010000"; data\_in\_line<=data\_in\_B(3 downto 0); next\_state<=s5;

when s5 => seg\_dig<=not"0000000000100000"; data\_in\_line<=data\_in\_B(7 downto 4); next\_state<=s6;

when s6 => seg\_dig<=not"0000000001000000"; data\_in\_line<=data\_in\_B(11 downto 8); next\_state<=s7;

when s7 => seg\_dig<=not"0000000010000000"; data\_in\_line<=data\_in\_B(15 downto 12); next\_state<=s8;

when s8 => seg\_dig<=not"0000000100000000"; data\_in\_line<=data\_in\_C(3 downto 0); next\_state<=s9;

when s9 => seg\_dig<=not"0000001000000000"; data\_in\_line<=data\_in\_C(7 downto 4); next\_state<=s10;

when s10=> seg\_dig<=not"0000010000000000"; data\_in\_line<=data\_in\_C(11 downto 8); next\_state<=s11;

when s11=> seg\_dig<=not"0000100000000000"; data\_in\_line<=data\_in\_C(15 downto 12); next\_state<=s12;

when s12=> seg\_dig<=not"0001000000000000"; data\_in\_line<=data\_in\_D(3 downto 0); next\_state<=s13;

when s13=> seg\_dig<=not"0010000000000000"; data\_in\_line<=data\_in\_D(7 downto 4); next\_state<=s14;

when s14=> seg\_dig<=not"0100000000000000"; data\_in\_line<=data\_in\_D(11 downto 8); next\_state<=s15;

when s15=> seg\_dig<=not"1000000000000000"; data\_in\_line<=data\_in\_D(15 downto 12); next\_state<=s0;

when others=>next\_state<=s0;

end case;

end process;

process(data\_in\_line)

begin

case data\_in\_line is

when "0000"=>seg\_data<=not"00111111";

when "0001"=>seg\_data<=not"00000110";

when "0010"=>seg\_data<=not"01011011";

when "0011"=>seg\_data<=not"01001111";

when "0100"=>seg\_data<=not"01100110";

when "0101"=>seg\_data<=not"01101101";

when "0110"=>seg\_data<=not"01111101";

when "0111"=>seg\_data<=not"00000111";

when "1000"=>seg\_data<=not"01111111";

when "1001"=>seg\_data<=not"01101111";

when "1010"=>seg\_data<=not"01110111";

when "1011"=>seg\_data<=not"01111100";

when "1100"=>seg\_data<=not"00111001";

when "1101"=>seg\_data<=not"01011110";

when "1110"=>seg\_data<=not"01111001";

when "1111"=>seg\_data<=not"01110001";

when others=>seg\_data<=not"11111111";

end case;

end process;

end Behavioral;

4——16

----------------------------------------------------------------------------------

-- Engineer: switch\_swq

-- Create Date: 2024/04/18 13:02:24

----------------------------------------------------------------------------------

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity decoder4\_16 is

Port (

sel\_in:in std\_logic\_vector(3 downto 0);

sel\_out:out std\_logic\_vector(15 downto 0)

);

end decoder4\_16;

architecture Behavioral of decoder4\_16 is

begin

process(sel\_in)

begin

case sel\_in is

when "0000"=>sel\_out<=not"0000000000000001";

when "0001"=>sel\_out<=not"0000000000000010";

when "0010"=>sel\_out<=not"0000000000000100";

when "0011"=>sel\_out<=not"0000000000001000";

when "0100"=>sel\_out<=not"0000000000010000";

when "0101"=>sel\_out<=not"0000000000100000";

when "0110"=>sel\_out<=not"0000000001000000";

when "0111"=>sel\_out<=not"0000000010000000";

when "1000"=>sel\_out<=not"0000000100000000";

when "1001"=>sel\_out<=not"0000001000000000";

when "1010"=>sel\_out<=not"0000010000000000";

when "1011"=>sel\_out<=not"0000100000000000";

when "1100"=>sel\_out<=not"0001000000000000";

when "1101"=>sel\_out<=not"0010000000000000";

when "1110"=>sel\_out<=not"0100000000000000";

when "1111"=>sel\_out<=not"1000000000000000";

when others=>sel\_out<=not"1111111111111111";

end case;

end process;

end Behavioral;