library IEEE;

use IEEE.STD\_LOGIC\_1164.all;

use IEEE.STD\_LOGIC\_unsigned.all;

entity final\_project is

port(

clk50 : in STD\_LOGIC;

clr : in STD\_LOGIC;

PS2C : in STD\_LOGIC;

PS2D : in STD\_LOGIC;

keyval1 : out STD\_LOGIC\_VECTOR(7 downto 0);

keyval2 : out STD\_LOGIC\_VECTOR(7 downto 0);

keyval3 : out STD\_LOGIC\_VECTOR(7 downto 0);

);

end final\_project;

architecture final\_project of final\_project is

type state\_type is (start, wtclklo1, wtclkhi1, getkey1, wtclklo2, wtclkhi2, getkey2, breakey, wtclklo3, wtclkhi3, getkey3);

signal state: state\_type;

signal PS2Cf, PS2Df: std\_logic;

signal ps2c\_filter, ps2d\_filter: std\_logic\_vector(7 downto 0);

signal shift1, shift2, shift3: std\_logic\_vector(10 downto 0);

signal keyval1s, keyval2s, keyval3s: std\_logic\_vector(7 downto 0);

signal bit\_count: std\_logic\_vector(3 downto 0);

signal bit\_count\_max: std\_logic\_vector(3 downto 0) := "1011";

begin

filter: process(clk50, clr)

begin

if clr = '1' then

ps2c\_filter <= (others => '0');

ps2d\_filter <= (others => '0');

PS2Cf <= '1';

PS2Df <= '1';

elsif clk50'event and clk50 = '1' then

ps2c\_filter(7) <= PS2C;

ps2c\_filter(6 downto 0) <= ps2d\_filter(7 downto 1);

ps2d\_filter(7) <= PS2D;

ps2d\_filter(6 downto 0) <= ps2d\_filter(7 downto 1);

if ps2c\_filter = X"FF" then

PS2Cf <= '1';

elsif ps2c\_filter = X"00" then

PS2Cf <= '0';

end if;

if ps2d\_filter = X"FF" then

PS2Df <= '1';

elsif ps2d\_filter = X"00" then

PS2Df <= '0';

end if;

end if;

end process filter;

skey: process(clk50, clr)

begin

if(clr = '1') then

state <= start;

bitcount <= (others => '0');

shift1 <= (others => '0');

shift2 <= (others => '0');

shift3 <= (others => '0');

keyval1s <= (others => '0');

keyval2s <= (others => '0');

keyval3s <= (others => '0');

elsif (clk50'event and clk50 = '1') then

case state is

when start =>

if PS2Df = '1' then

state <= start;

else

state <= wtclklo1;

end if;

when wtclklo1 =>

if bit\_count < bit\_count\_max then

if PS2Cf = '1' then

state <= wtclklo1;

else

state <= wtclkhi1;

shift1 <= PS2Df & shift1(10 downto 1);

end if;

else

state <= getkey1;

end if;

when wtclkhi1 =>

if PS2Cf = '0' then

state <= wtclkhi1;

else

state <= wtclklo1;

bit\_count <= bit\_count +1;

end if;

when getkey1 =>

keyval1s <= shift1(8 downto 1);

bit\_count <= (others => '0');

state <= wtclklo2;

when wtclklo2 =>

if bit\_count < bit\_count\_max then

if PS2Cf = '1' then

state <= wtclklo2;

else

state <= wtclkhi2;

shift2 <= PS2Df & shift2(10 downto 1);

end if;

else

state <= getkey2;

end if;

when wtclkhi2 =>

if PS2Cf = '0' then

state <= wtclkhi2;

else

state <= wtclklo2;

bit\_count <= bit\_count +1;

end if;

when getkey2 =>

keyval2s <= shift2(8 downto 1);

bit\_count <= (others => '0');

state <= breakey;

when breakey =>

if keyval2s = X"F0" then

state <= wtclklo3;

else

if keyval1s = X"E0" then

state <= wtclklo1;

else

state <= wtclklo2;

end if;

end if;

when wtclklo3 =>

if bit\_count < bit\_count\_max then

if PS2Cf = '1' then

state <= wtclklo3;

else

state <= wtclkhi3;

shift3 <= PS2Df & shift3(10 downto 1);

end if;

else

state <= getkey3;

end if;

when wtclkhi3 =>

if PS2Cf = '0' then

state <= wtclkhi3;

else

state <= wtclklo3;

bit\_count <= bit\_count +1;

end if;

when getkey3 =>

keyval3s <= shift3(8 downto 1);

bit\_count <= (others => '0');

state <= wtclklo1;

end case;

end if;

end process skey;

keyval1 <= keyval1s;

keyval2 <= keyval2s;

keyval3 <= keyval3s;

end final\_project;