FPGA系統設計\_第九章\_程序處理

**10867024 廖育賢**

# **objective- THE PROBLEM AND PURPOSE**

結合按鍵處理、狀態控制及波形顯示功能，建立一個五鍵輸入的電子密碼鎖電路。

# **procedure – DESIGN methods**

## 外部電路配置:

使用基本測試模組， 利用4組按鍵XC(3~0)並以8組LED燈輸出訊號XB(7~0) 。

7個密碼 : 1233011

## 內部電路設計:

此電路分成6個Block:

1. 自由計數器 : 18位元，產生128Hz 和16Hz赫茲的抽樣訊號。

2. 按鍵解碼電路 : 將四個按鍵解為00-01-10-11四碼。

3. 彈跳消除電路 : 標準的按鍵彈跳消除電路，抽樣頻率128Hz。

4. 按鍵微分電路 : 將彈跳消除後的輸出改成微分型式。

5. 狀態控制電路 : 控制密碼輸入程序，由按鍵訊號出發。

6. 移位暫存電路 : 當密碼輸入完畢，用移位暫存器製造出鼓掌型式的波形輸出。

# **simulation results**

## program codes

### **ch09.vhd**

library IEEE;

use IEEE.std\_logic\_1164.all;

use IEEE.std\_logic\_arith.all;

use IEEE.std\_logic\_unsigned.all;

entity CH09 is

port

( CLK : in STD\_LOGIC;

XA : out STD\_LOGIC\_VECTOR (3 downto 0);

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

XC : in STD\_LOGIC\_VECTOR (3 downto 0)

);

end CH09;

architecture CH09\_ARCH of CH09 is

signal SAMPLE,DISPLAY : STD\_LOGIC;

signal KEY,FLT,DIF,DOUT : STD\_LOGIC;

signal DIN : STD\_LOGIC\_VECTOR (1 downto 0);

begin

FREE\_COUNTER : block

signal Q : STD\_LOGIC\_VECTOR (19 downto 0);

signal D1,D2 : STD\_LOGIC;

begin

process (CLK)

begin

if CLK'event and CLK='1' then --CLK rising

D1 <= Q(16); --delay

D2 <= Q(19); --delay

Q <= Q+1; --UP counter

end if;

end process;

SAMPLE <= Q(16) and not D1; --differential (125Hz)

DISPLAY <= Q(19) and not D2; --differential ( 16Hz)

end block FREE\_COUNTER;

KEYBOARD\_DECODING : block

begin

KEY <= not(XC(0) and XC(1) and XC(2) and XC(3)); --any keyin

DIN <= "00" when XC="1110" else --<0> for XC(0)

"01" when XC="1101" else --<1> for XC(1)

"10" when XC="1011" else --<2> for XC(2)

"11"; --<3> for XC(3)

end block KEYBOARD\_DECODING;

DEBOUNCING : block

signal D0,D1 : STD\_LOGIC;

begin

process (CLK)

begin

if CLK'event and CLK='1' then --CLK rising

if SAMPLE='1' then --sampling control

D1<=D0; D0<=KEY; --delay

FLT <= ((D0 and D1) or FLT) and (D0 or D1); --RS-F/F

end if;

end if;

end process;

end block DEBOUNCING;

DIFFERENTIAL : block

signal D0,D1 : STD\_LOGIC;

begin

process (CLK)

begin

if CLK'event and CLK='1' then --CLK rising

D1<=D0; D0<=FLT; --delay

end if;

end process;

DIF <= D0 and not D1; --differential

end block DIFFERENTIAL;

STATE\_MACHINE : block

type STATE is ( S0,S1,S2,S3,S4,S5,S6,S7 ); --STATE definition

attribute enum\_encoding of

STATE : type is "000 001 010 011 100 101 110 111";

signal EC : STD\_LOGIC;

signal NOW,NXT : STATE;

begin

process (CLK) --sequential logic

begin

if CLK'event and CLK='1' then --CLK rising

if EC='1' then --CLK enable

NOW <= NXT; --state to next

end if;

end if;

end process;

process (NOW,DIN) --combinational logic

begin

case NOW is

when S0 => --in state S0

DOUT <= '0';

if DIN="01" then --check <1>

NXT <= S1;

else

NXT <= S0;

end if;

when S1 => --in state S1

DOUT <= '0';

if DIN="10" then --check <2>

NXT <= S2;

else

NXT <= S0;

end if;

when S2 => --in state S2

DOUT <= '0';

if DIN="11" then --check <3>

NXT <= S3;

else

NXT <= S0;

end if;

when S3 => --in state S3

DOUT <= '0';

if DIN="11" then --check <3>

NXT <= S4;

else

NXT <= S0;

end if;

when S4 => --in state S4

DOUT <= '0';

if DIN="00" then --check <0>

NXT <= S5;

else

NXT <= S0;

end if;

when S5 => --in state S5

DOUT <= '0';

if DIN="01" then --check <1>

NXT <= S7;

else

NXT <= S0;

end if;

when S7 => --in state S2

DOUT <= '0';

if DIN="01" then --check <1>

NXT <= S6;

else

NXT <= S0;

end if;

when S6 => --in state S6

DOUT <= '1'; --signal OK

NXT <= S0;

when others => --in others

DOUT <= '0';

NXT <= S0;

end case;

end process;

EC <= DIF; --sample signal

XA <= "1111" when NOW=S0 else --state monitor

"1110" when NOW=S1 else

"1101" when NOW=S2 else

"1100" when NOW=S3 else

"1011" when NOW=S4 else

"1010" when NOW=S5 else

"1001" when NOW=S6 else

"1111";

end block STATE\_MACHINE;

SHIFT\_REGISTER : block

signal STS : STD\_LOGIC\_VECTOR (7 downto 0);

signal RST,EC : STD\_LOGIC;

begin

process (CLK,RST)

begin

if RST='1' then --reset status

STS <= "00000001"; --to S0

elsif CLK'event and CLK='1' then --CLK rising

if EC='1' then

STS <= STS(6 downto 0) & STS(7); --rotate left

end if;

end if;

end process;

RST <= not DOUT; --RST signal

EC <= DOUT and DISPLAY; --EC signal

XB(0) <= not(DOUT and (STS(0) or STS(7))); --LED output

XB(1) <= not(DOUT and (STS(1) or STS(6)));

XB(2) <= not(DOUT and (STS(2) or STS(5)));

XB(3) <= not(DOUT and (STS(3) or STS(4)));

XB(4) <= not(DOUT and (STS(4) or STS(3)));

XB(5) <= not(DOUT and (STS(5) or STS(2)));

XB(6) <= not(DOUT and (STS(6) or STS(1)));

XB(7) <= not(DOUT and (STS(7) or STS(0)));

end block SHIFT\_REGISTER;

end CH09\_ARCH;

### **CH09.UCF**

NET CLK LOC = B12;

NET XA<0> LOC = N2;

NET XA<1> LOC = T1;

NET XA<2> LOC = T2;

NET XA<3> LOC = U2;

NET XB<0> LOC = M3;

NET XB<1> LOC = M4;

NET XB<2> LOC = W3;

NET XB<3> LOC = W4;

NET XB<4> LOC = Y3;

NET XB<5> LOC = Y4;

NET XB<6> LOC = W6;

NET XB<7> LOC = Y6;

NET XC<0> LOC = V1;

NET XC<1> LOC = V2;

NET XC<2> LOC = W1;

NET XC<3> LOC = W2;

## observations

### 1.Now = S1

