FPGA系統設計\_第十二章\_密碼鎖專題

**10867024 廖育賢**

# **objective- THE PROBLEM AND PURPOSE**

這次的實驗應用了使用了鍵盤輸入與LED顯示的功能。

# **procedure – DESIGN methods**

## 外部電路配置:

4\*4 按鍵矩陣 :

1. 0~9 數字: 輸入值右方移入按鍵數字。
2. 按鍵退位 : 消除前一個數字輸入。
3. 按鍵清除 : 清除目前輸入值。
4. 密碼清除 : 密碼清除成”0000”。
5. 密碼更改 : 將目前數字設為新密碼。
6. 所謂復歸 : 密碼鎖上鎖。
7. 密碼檢查 : 檢查輸入與密碼是否吻合，符合即開鎖。

## 內部電路設計:

1. 鍵盤次序處理。
2. 按鍵解碼電路。
3. 按鍵暫存電路。
4. 談跳消除電路。
5. 按鍵微分電路。
6. 自由計數電路。
7. 數字選擇電路。
8. 七段式查表電路。
9. 數字輸入電路。
10. 密碼更改電路。
11. 密碼檢查電路。

# **simulation results**

## program codes

### **ch012.vhd**

--The IEEE standard 1164 package, declares std\_logic, rising\_edge(), etc.

library IEEE;

use IEEE.std\_logic\_1164.all;

use IEEE.std\_logic\_arith.all;

use IEEE.std\_logic\_unsigned.all;

--

--\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- CH12B : electronic code lock (test #B)

-- designed by Pei-Chong Tang, Feb. 1999

--\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

--

entity CH12B is

port

( CLK : in STD\_LOGIC; --4MHz clock

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

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

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

);

end CH12B;

architecture CH12B\_ARCH of CH12B is

component KEYBOARD

port

( CLK : in STD\_LOGIC; --system clock

SAMPLE : in STD\_LOGIC; --sample point

SEQ : in STD\_LOGIC\_VECTOR(1 downto 0); --scanning sequence

INP : in STD\_LOGIC\_VECTOR(3 downto 0); --keyboard input

CODE : out STD\_LOGIC\_VECTOR(3 downto 0); --keyboard output

KIN : out STD\_LOGIC --keyboard strike

);

end component;

component DISPLAY

port

( CLK : in STD\_LOGIC; --system clock

REG : in STD\_LOGIC\_VECTOR (15 downto 0); --digit data

SEG : out STD\_LOGIC\_VECTOR (6 downto 0); --segment output

SEQ : out STD\_LOGIC\_VECTOR (1 downto 0); --segment output

SAMPLE,BLINK : out STD\_LOGIC --sample and blink

);

end component;

signal ACC : STD\_LOGIC\_VECTOR (15 downto 0);

signal REG : STD\_LOGIC\_VECTOR (15 downto 0);

signal SEG : STD\_LOGIC\_VECTOR (6 downto 0);

signal ENB : STD\_LOGIC\_VECTOR (3 downto 0);

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

signal CODE : STD\_LOGIC\_VECTOR (3 downto 0);

signal FUN : STD\_LOGIC\_VECTOR (6 downto 0);

signal KIN,SAMPLE,UNLOCK,BLINK : STD\_LOGIC;

begin

--

--\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

--

-- system connection for the experiment

--

SYSTEM\_CONNECT : block

signal INP : STD\_LOGIC\_VECTOR (3 downto 0);

begin

U1: KEYBOARD port map (CLK, SAMPLE, SEQ, INP, CODE, KIN);

U2: DISPLAY port map (CLK, ACC, SEG, SEQ, SAMPLE, BLINK);

INP <= not XC;

XA <= not ENB;

XB(7) <= UNLOCK and BLINK;

XB(6 downto 0) <= SEG;

ENB <= "0001" when SEQ=0 else

"0010" when SEQ=1 else

"0100" when SEQ=2 else

"1000" when SEQ=3 else

"0000";

FUN <= "0000010" when CODE=10 else

"0000100" when CODE=11 else

"0001000" when CODE=12 else

"0010000" when CODE=13 else

"0100000" when CODE=14 else

"1000000" when CODE=15 else

"0000001";

end block SYSTEM\_CONNECT;

--

--\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- keyin process

--

KEYIN\_PROCESS : block

signal RST,EC,DIR : STD\_LOGIC;

begin

process (CLK,RST)

begin

if RST='1' then --CLK reset

ACC <= "0000000000000000";

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

if EC='1' then --CLK enable

if DIR='1' then

ACC <= ACC(11 downto 0) & CODE; --shift left

else

ACC <= "0000" & ACC(15 downto 4); --shift right

end if;

end if;

end if;

end process;

RST <= KIN and FUN(2); --RESET

EC <= KIN and (FUN(0) or FUN(1)); --NUMBER or BACK

DIR <= FUN(0); --NUMBER

end block KEYIN\_PROCESS;

--

--\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- setting process

--

SETTING\_PROCESS : block

signal RST,EC : STD\_LOGIC;

begin

process (CLK,RST)

begin

if RST='1' then --data reset

REG <= "0000000000000000";

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

if EC='1' then --CLK enable

REG <= ACC; --data latch

end if;

end if;

end process;

RST <= KIN and FUN(3); --RESET CODE

EC <= KIN and FUN(4) and UNLOCK; --SET CODE

end block SETTING\_PROCESS;

--

--\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- compare process

--

COMPARE\_PROCESS : block

signal RST,EC,Q : STD\_LOGIC;

begin

process (CLK,RST)

begin

if RST='1' then --data reset

Q <= '0'; --data reset

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

if EC='1' then --CLK enable

Q <= '1'; --RS Flip/Flop

end if;

end if;

end process;

RST <= KIN and FUN(5); --RESET LOCK

EC <= '1' when KIN='1' and FUN(6)='1' and REG=ACC else

'0';

UNLOCK <= Q;

end block COMPARE\_PROCESS;

--

--\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- end of architechture

--

end CH12B\_ARCH;

--The IEEE standard 1164 package, declares std\_logic, rising\_edge(), etc.

library IEEE;

use IEEE.std\_logic\_1164.all;

use IEEE.std\_logic\_arith.all;

use IEEE.std\_logic\_unsigned.all;

--

--\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- DISPLAY : 4-digit LED display control

-- designed by Pei-Chong Tang, Feb. 1999

--\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

--

entity DISPLAY is

port

( CLK : in STD\_LOGIC; --system clock

REG : in STD\_LOGIC\_VECTOR (15 downto 0); --digit data

SEG : out STD\_LOGIC\_VECTOR (6 downto 0); --segment output

SEQ : out STD\_LOGIC\_VECTOR (1 downto 0); --segment output

SAMPLE,BLINK : out STD\_LOGIC --sample and blink

);

end DISPLAY;

architecture DISPLAY\_ARCH of DISPLAY is

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

signal DB : STD\_LOGIC\_VECTOR (3 downto 0);

begin

--

--\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- 22-bit free counter

--

FREE\_COUNTER : block

signal Q : STD\_LOGIC\_VECTOR (21 downto 0);

signal D : STD\_LOGIC;

begin

process (CLK)

begin

if CLK'event and CLK='1' then

D <= Q(15);

Q <= Q+1;

end if;

end process;

BLINK <= Q(21);

SAMPLE <= not Q(15) and D;

SCAN <= Q(17 downto 16);

SEQ <= SCAN;

end block FREE\_COUNTER;

--

--\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- scanning sequence and bus multiplexer

--

MULTIPLEXER : block

begin

DB <= REG(15 downto 12) when SCAN=3 else "ZZZZ";

DB <= REG(11 downto 8) when SCAN=2 else "ZZZZ";

DB <= REG(7 downto 4) when SCAN=1 else "ZZZZ";

DB <= REG(3 downto 0) when SCAN=0 else "ZZZZ";

end block MULTIPLEXER;

--

--\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- seven-segment LED transformation

--

SEVEN\_SEGMENT : block

begin

--GFEDCBA

SEG <= "0111111" when DB=0 else

"0000110" when DB=1 else

"1011011" when DB=2 else

"1001111" when DB=3 else

"1100110" when DB=4 else

"1101101" when DB=5 else

"1111101" when DB=6 else

"0000111" when DB=7 else

"1111111" when DB=8 else

"1101111" when DB=9 else

"1110111" when DB=10 else

"1111100" when DB=11 else

"0111001" when DB=12 else

"1011110" when DB=13 else

"1111001" when DB=14 else

"1110001" when DB=15 else

"0000000";

end block SEVEN\_SEGMENT;

--

--\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- end of architechture

--

end DISPLAY\_ARCH;

--The IEEE standard 1164 package, declares std\_logic, rising\_edge(), etc.

library IEEE;

use IEEE.std\_logic\_1164.all;

use IEEE.std\_logic\_arith.all;

use IEEE.std\_logic\_unsigned.all;

--

--\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- KEYBOARD : one-bit keyboard debouncing

-- designed by Pei-Chong Tang, Feb. 1999

--\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

--

entity KEYBOARD is

port

( CLK : in STD\_LOGIC; --system clock

SAMPLE : in STD\_LOGIC; --sample point

SEQ : in STD\_LOGIC\_VECTOR(1 downto 0); --scanning sequence

INP : in STD\_LOGIC\_VECTOR(3 downto 0); --keyboard input

CODE : out STD\_LOGIC\_VECTOR(3 downto 0); --keyboard output

KIN : out STD\_LOGIC --keyboard strike

);

end KEYBOARD;

architecture KEYBOARD\_ARCH of KEYBOARD is

signal KBUF : STD\_LOGIC\_VECTOR (4 downto 0);

signal PLS1,PLS2,PLS3 : STD\_LOGIC;

signal KEY,FLT : STD\_LOGIC;

begin

--

--\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

--

-- keyboard sequence control

--

KEYBOARD\_SEQUENCE : block

begin

process (CLK)

begin

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

PLS3<=PLS2; PLS2<=PLS1; PLS1<=SAMPLE; --delay

end if;

end process;

end block KEYBOARD\_SEQUENCE;

--

--\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

--

-- keyboard decoding

--

KEYBOARD\_DECODING : block

signal D : STD\_LOGIC\_VECTOR (2 downto 0);

signal EC,RST : STD\_LOGIC;

begin

process (CLK,RST)

begin

if RST='1' then

KBUF <= "00000";

elsif CLK'event and CLK='1' then

if EC='1' then

KBUF <= D & (not SEQ);

end if;

end if;

end process;

D(2 downto 0) <= "100" when INP="0001" else

"101" when INP="0010" else

"110" when INP="0100" else

"111" when INP="1000" else

"000";

EC <= PLS1 and D(2);

RST <= PLS3 and not SEQ(1) and not SEQ(0);

end block KEYBOARD\_DECODING;

--

--\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

--

-- keyboard latch

--

KEYBOARD\_LATCH : block

signal EC : STD\_LOGIC;

begin

process (CLK)

begin

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

if EC='1' then --CLK enable

CODE <= KBUF(3 downto 0); --latch code

KEY <= KBUF(4);

end if;

end if;

end process;

EC <= PLS2 and not SEQ(1) and not SEQ(0);

end block KEYBOARD\_LATCH;

--

--\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

--

-- keyboard debouncing

--

KEYBOARD\_DEBOUNCING : block

signal D0,D1,EC : STD\_LOGIC;

begin

process (CLK)

begin

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

if EC='1' then --CLK enable

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

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

end if;

end if;

end process;

EC <= PLS3 and not SEQ(1) and not SEQ(0);

end block KEYBOARD\_DEBOUNCING;

--

--\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

--

-- differential signal

--

KEYBOARD\_DIFF : 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;

KIN <= D0 and not D1; --differential

end block KEYBOARD\_DIFF;

--

--\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- end of architechture

--

end KEYBOARD\_ARCH;

### **CH012.UCF**

##############################################

# I/O mapping for USB 3s1000 FPGA board #

##############################################

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.設定密碼 1234:

### 

### 2.輸入錯誤密碼 2340:

### 

### 3.密碼檢查 1234 (正確):

