FPGA系統設計\_第十五章\_溫度控制專題

**10867024 廖育賢**

# **objective- THE PROBLEM AND PURPOSE**

溫度測量屬於類比電路的範疇，類比電路的複雜性與精度規格有著絕對的關係；此實驗設計採基本規格，學習溫度量測和控制界面技術，包括類比數位轉換、開關式與比例式的迴受控制。

# **procedure – DESIGN methods**

## 外部電路配置:

選擇LED鍵盤模組及溫度控制模組。溫度控制模組包含一個由PWM訊號驅動之功率電阻，可當成電熱絲來運用，當電阻兩端施加12伏特電源時，就會逐漸加熱。另外在功率電阻下安裝AD590來量測溫度，訊號經放大之後由ADC轉換器送回FPGA元件。

## 內部電路設計:

XA(3~0) : 連接四位數字的4位元掃描控制。

XB(7~0) : 連接七段式LED數字顯示的8位源控制。

XC(3~0) : 4位元的按鍵矩陣輸入。

YA(0) : 功率電阻的PWM驅動訊號，1:加熱、0:關閉。

YA(3~2) : ADC轉換器的控制訊號，YA(2):RD、YA(3):WR。

YB(7~0) : ADC轉換器的8位元轉換結果。

# **simulation results**

## program codes

### **ch15.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;

--This package is a version of the Synopsys package and has been

-- optimized for use with the Express compiler.

--library SYNOPSYS;

--use SYNOPSYS.attributes.all;

--

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

-- CH15C : temperature control (test #C)

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

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

--

entity CH15C is

port

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

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

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

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

YA : out STD\_LOGIC\_VECTOR (3 downto 0); --PWM output

YB : in STD\_LOGIC\_VECTOR (7 downto 0) --photo input

);

end CH15C;

architecture CH15C\_ARCH of CH15C is

component MONITOR

port

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

DB : inout STD\_LOGIC\_VECTOR (7 downto 0); --data bus

A : out STD\_LOGIC\_VECTOR (3 downto 0); --address bus

RD : out STD\_LOGIC; --read control

SWR : out STD\_LOGIC; --write control

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

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

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

FREQ: out STD\_LOGIC --test frequency

);

end component;

component ADC

port

( AOP : out STD\_LOGIC\_VECTOR (7 downto 0); --ADC data

AIN : in STD\_LOGIC\_VECTOR (7 downto 0); --ADC input

STC : out STD\_LOGIC; --start of conv.

ENB : out STD\_LOGIC; --read enable

FREQ : in STD\_LOGIC; --sample point

CLK : in STD\_LOGIC --system clock

);

end component;

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

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

signal RD, SWR : STD\_LOGIC;

signal AOP,AIN : STD\_LOGIC\_VECTOR (7 downto 0);

signal CMD,PWM : STD\_LOGIC\_VECTOR (7 downto 0);

signal CS0,PWR : STD\_LOGIC;

begin

--

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

--

-- system connection for the experiment

--

SYSTEM\_CONNECT : block

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

signal FREQ,STC,ENB : STD\_LOGIC;

begin

U1: MONITOR port map (CLK, DB, A, RD, SWR, XA, XB, XC, FREQ);

U2: ADC port map (AOP, AIN, STC, ENB, FREQ, CLK);

YA(0) <= PWR;

YA(1) <= '1';

YA(2) <= ENB;

YA(3) <= STC;

AIN <= YB;

CS0 <= not A(3) and not A(2) and not A(1) and not A(0);

end block SYSTEM\_CONNECT;

--

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

-- command register

--

COMMAND\_REGISTER : block

signal T,EC : STD\_LOGIC;

begin

process (CLK)

begin

if CLK'event and CLK='1' then

if EC='1' then

CMD <= DB;

end if;

end if;

end process;

DB <= AOP when T='1' else "ZZZZZZZZ";

EC <= SWR and CS0;

T <= not RD and CS0;

end block COMMAND\_REGISTER;

--

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

-- temperature control

--

TEMP\_CONTROL : block

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

begin

DIF <= (CMD-AOP) when CMD>AOP else

"00000000";

PWM <= DIF;

end block TEMP\_CONTROL;

--

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

-- PWM generator

--

PWM\_GENERATOR : block

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

begin

process (CLK)

begin

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

Q <= Q-1; --down counter

end if;

end process;

PWR <= '1' when PWM>Q else '0'; --PWM output

end block PWM\_GENERATOR;

--

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

-- end of architechture

--

end CH15C\_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;

--This package is a version of the Synopsys package and has been

-- optimized for use with the Express compiler.

--library SYNOPSYS;

--use SYNOPSYS.attributes.all;

--

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

-- MONITOR : system monitor with LED and keyboard

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

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

--

entity MONITOR is

port

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

DB : inout STD\_LOGIC\_VECTOR (7 downto 0); --data bus

A : out STD\_LOGIC\_VECTOR (3 downto 0); --address bus

RD : out STD\_LOGIC; --read control

SWR : out STD\_LOGIC; --write control

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

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

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

FREQ: out STD\_LOGIC --test frequency

);

end MONITOR;

architecture MONITOR\_ARCH of MONITOR 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;

signal REG : STD\_LOGIC\_VECTOR (12 downto 0);

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

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

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

signal KIN,SAMPLE,BLINK : STD\_LOGIC;

begin

--

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

-- system connection for LED & keyboard interface

--

SYSTEM\_CONNECT : block

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

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

begin

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

DB <= REG(7 downto 0) when REG(12)='1' else

"ZZZZZZZZ";

INP <= not XC;

XA <= not ENB;

XB(7) <= (SEQ(1) and SEQ(0) and not REG(12) and BLINK) or

(SEQ(1) and not SEQ(0));

XB(6 downto 0) <= SEG when SEQ(1 downto 0)/=3 else "0000000";

ENB <= "0001" when SEQ=0 else

"0010" when SEQ=1 else

"0100" when SEQ=2 else

"1000" when SEQ=3 else

"0000";

RD <= REG(12);

SWR <= REG(12) and KIN and

not CODE(3) and CODE(2) and not CODE(1) and not CODE(0);

A <= REG(11 downto 8);

end block SYSTEM\_CONNECT;

--

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

-- 22-bit free counter

--

FREE\_COUNTER : block

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

signal DLY1,DLY2 : STD\_LOGIC;

begin

process (CLK)

begin

if CLK'event and CLK='1' then

DLY1 <= Q(15);

DLY2 <= Q(13);

Q <= Q+1;

end if;

end process;

BLINK <= Q(21);

SAMPLE <= not Q(15) and DLY1;

FREQ <= not Q(13) and DLY2;

SEQ <= Q(17 downto 16);

end block FREE\_COUNTER;

--

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

--

--

DIGIT0\_MONITOR : block --\*\*\*\*\* digit #0 \*\*\*\*\*

signal EC,LD,DIR : STD\_LOGIC;

begin

process (CLK)

begin

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

if EC='1' then --CLK enable

if LD='1' then

REG(3 downto 0) <= DB (3 downto 0); --load register

elsif DIR='0' then

REG(3 downto 0) <= REG(3 downto 0) + 1; --up counter

else

REG(3 downto 0) <= REG(3 downto 0) - 1; --down counter

end if;

end if;

end if;

end process;

EC <= (SAMPLE and not REG(12)) or

(KIN and not CODE(3) and CODE(1) and CODE(0));

LD <= SAMPLE and not REG(12);

DIR <= CODE(2);

end block DIGIT0\_MONITOR;

DIGIT1\_MONITOR : block --\*\*\*\*\* digit #1 \*\*\*\*\*

signal EC,LD,DIR : STD\_LOGIC;

begin

process (CLK)

begin

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

if EC='1' then --CLK enable

if LD='1' then

REG(7 downto 4) <= DB (7 downto 4); --load register

elsif DIR='0' then

REG(7 downto 4) <= REG(7 downto 4) + 1; --up counter

else

REG(7 downto 4) <= REG(7 downto 4) - 1; --down counter

end if;

end if;

end if;

end process;

EC <= (SAMPLE and not REG(12)) or

(KIN and not CODE(3) and CODE(1) and not CODE(0));

LD <= SAMPLE and not REG(12);

DIR <= CODE(2);

end block DIGIT1\_MONITOR;

DIGIT2\_MONITOR : block --\*\*\*\*\* digit #2 \*\*\*\*\*

signal EC,DIR : STD\_LOGIC;

begin

process (CLK)

begin

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

if EC='1' then --CLK enable

if DIR='0' then

REG(11 downto 8) <= REG(11 downto 8) + 1; --up counter

else

REG(11 downto 8) <= REG(11 downto 8) - 1; --down counter

end if;

end if;

end if;

end process;

EC <= KIN and not CODE(3) and not CODE(1) and CODE(0);

DIR <= CODE(2);

end block DIGIT2\_MONITOR;

DIGIT3\_MONITOR : block --\*\*\*\*\* digit #3 \*\*\*\*\*

signal EC : STD\_LOGIC;

begin

process (CLK)

begin

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

if EC='1' then --CLK enable

REG(12) <= not REG(12); --T-type F/F

end if;

end if;

end process;

EC <= KIN and not CODE(3) and not CODE(2) and not CODE(1) and not CODE(0);

end block DIGIT3\_MONITOR;

--

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

-- seven-segment LED transformation

--

SEVEN\_SEGMENT : block

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

begin

D <= REG(11 downto 8) when SEQ=2 else "ZZZZ";

D <= REG(7 downto 4) when SEQ=1 else "ZZZZ";

D <= REG(3 downto 0) when SEQ=0 else "ZZZZ";

--GFEDCBA

SEG <= "0111111" when D=0 else

"0000110" when D=1 else

"1011011" when D=2 else

"1001111" when D=3 else

"1100110" when D=4 else

"1101101" when D=5 else

"1111101" when D=6 else

"0000111" when D=7 else

"1111111" when D=8 else

"1101111" when D=9 else

"1110111" when D=10 else

"1111100" when D=11 else

"0111001" when D=12 else

"1011110" when D=13 else

"1111001" when D=14 else

"1110001" when D=15 else

"0000000";

end block SEVEN\_SEGMENT;

--

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

-- end of architechture

--

end MONITOR\_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;

--This package is a version of the Synopsys package and has been

-- optimized for use with the Express compiler.

--library SYNOPSYS;

--use SYNOPSYS.attributes.all;

--

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

-- ADC : 8-BIT ADC input

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

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

--

entity ADC is

port

( AOP : out STD\_LOGIC\_VECTOR (7 downto 0); --ADC data

AIN : in STD\_LOGIC\_VECTOR (7 downto 0); --ADC input

STC : out STD\_LOGIC; --start of conv.

ENB : out STD\_LOGIC; --read enable

FREQ : in STD\_LOGIC; --sample point

CLK : in STD\_LOGIC --system clock

);

end ADC;

architecture ADC\_ARCH of ADC is

signal D0,D1,D2,D3 : STD\_LOGIC;

begin

--

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

--

-- time sequence generator

--

TIME\_SEQUENCE : block

begin

process (CLK)

begin

if CLK'event and CLK='1' then

D3<=D2; D2<=D1; D1<=D0; D0<=FREQ;

end if;

end process;

ENB <= not (FREQ or D0 or D1);

STC <= not D3;

end block TIME\_SEQUENCE;

--

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

--

-- 8-bit analog-to-digital input

--

ADC\_FETCH : block

signal EC : STD\_LOGIC;

begin

process (CLK)

begin

if CLK'event and CLK='1' then

if EC='1' then

AOP <= AIN;

end if;

end if;

end process;

EC <= D1;

end block ADC\_FETCH;

--

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

-- end of architechture

--

end ADC\_ARCH;

### **CH15.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;

NET YA<0> LOC = W10;

NET YA<1> LOC = Y10;

NET YA<2> LOC = W9;

NET YA<3> LOC = W8;

NET YB<0> LOC = U3;

NET YB<1> LOC = T4;

NET YB<2> LOC = N3;

NET YB<3> LOC = N4;

NET YB<4> LOC = T5;

NET YB<5> LOC = U4;

NET YB<6> LOC = V3;

NET YB<7> LOC = V4;

## observations

### 功率電阻加熱:

