FPGA系統設計\_第十四章\_運動控制專題

**10867024 廖育賢**

# **objective- THE PROBLEM AND PURPOSE**

運動控制專題探討直流馬達的介面控制技術，包括脈寬調變電路及光電解碼界面。配合實驗模組，逐步建立開環路速度控制、位置量測和速度量測實驗。

# **procedure – DESIGN methods**

## 外部電路配置:

包含LED鍵盤模組和運動控制模組，運動控制模組:散熱風扇及相關驅動器。XA(3~0): 連接四位數字的4位元掃描控制，其中XA(3)最高位；XB(7~0): 連接七段式LED數字顯示的8位元控制；XC(3~0): 4位元的按鍵矩陣輸入；YA(1~0): YA(0)為風扇PWM驅動訊號，YA(1)為光電盤的光源控制；YC(1~0) : 光電盤兩相訊號輸入，YC(0)為A相，YC(1)為B相。

## 內部電路設計:

1. 數位濾波電路(A、B相)
2. 乘四解碼電路
3. 正反計數器電路
4. 速度計算電路
5. 匯流排界面電路

# **simulation results**

## program codes

### **ch014.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;

--

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

-- CH14C : motion control (test #C)

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

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

--

entity CH14C 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 (1 downto 0); --PWM output

YC : in STD\_LOGIC\_VECTOR (1 downto 0) --photo input

);

end CH14C;

architecture CH14C\_ARCH of CH14C 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 PWM

port

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

OP : out STD\_LOGIC; --PWM output

CS : in STD\_LOGIC; --chip select

SWR : in STD\_LOGIC; --sync write

CLK : in STD\_LOGIC --system clock

);

end component;

component SPEED

port

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

PA,PB : in STD\_LOGIC; --photo input

CS : in STD\_LOGIC\_VECTOR (1 downto 0); --chip select

RD : in STD\_LOGIC; --read signal

CLK : in STD\_LOGIC; --system clock

FREQ : in STD\_LOGIC --sample point

);

end component;

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

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

signal RD,SWR : STD\_LOGIC;

begin

--

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

--

-- system connection for the experiment

--

SYSTEM\_CONNECT : block

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

signal OP,PA,PB,FREQ : STD\_LOGIC;

begin

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

U2: PWM port map (DB, OP, CS(0), SWR, CLK);

U3: SPEED port map (DB, PA, PB, CS, RD, CLK, FREQ);

YA(0) <= OP;

YA(1) <= '1';

PA <= YC(0);

PB <= YC(1);

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

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

end block SYSTEM\_CONNECT;

--

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

-- end of architechture

--

end CH14C\_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;

--

--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;

--

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

-- PHOTO : photo-encoder decoding circuit

-- (CLK, PHTA,PHTB,PHTZ, MODE, ZCLR, RD)

-- ==> (DBUS, ZERO)

--

-- designed by Pei-Chong Tang, Dec. 1998

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

--

entity PHOTO is

port

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

PA,PB : in STD\_LOGIC; --photo input

CS : in STD\_LOGIC; --chip select

RD : in STD\_LOGIC; --read signal

CLK : in STD\_LOGIC --system clock

);

end PHOTO;

architecture PHOTO\_ARCH of PHOTO is

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

signal PHA,PHB : STD\_LOGIC;

signal DIR,PLS : STD\_LOGIC;

begin

--

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

-- low-pass digital filter

-- (PA, PB)

-- ==> (PHA, PHB)

--

FILTER\_A : block -- digital filter for #A

signal D0,D1,D2 : STD\_LOGIC;

begin

process (CLK)

begin

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

D2<=D1; D1<=D0; D0<=PA; --delay 3 clocks

PHA <= (D0 and D1 and D2) or ((D0 or D1 or D2) and PHA);

end if;

end process;

end block FILTER\_A;

FILTER\_B : block -- digital filter for #B

signal D0,D1,D2 : STD\_LOGIC;

begin

process (CLK)

begin

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

D2<=D1; D1<=D0; D0<=PB; --delay 3 clocks

PHB <= (D0 and D1 and D2) or ((D0 or D1 or D2) and PHB);

end if;

end process;

end block FILTER\_B;

--

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

-- up/down decoding

-- (PHA, PHB)

-- ==> (DIR, PLS)

--

DECODER : block

signal DLA,DLB : STD\_LOGIC;

begin

process (CLK)

begin

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

DLA<=PHA; DLB<=PHB; -- delay 1-clock

end if;

end process;

DIR <= (not PHA and DLA and not PHB)

or ( PHA and not DLA and PHB)

or (not PHB and DLB and PHA)

or ( PHB and not DLB and not PHA);

PLS <= (not PHA and DLA) or (PHA and not DLA)

or (not PHB and DLB) or (PHB and not DLB);

end block DECODER;

--

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

-- up/down counter

-- (CLK, MODE, PLS, DIR)

-- ==> (CNT)

--

COUNTER : block

signal EC : STD\_LOGIC;

begin

process (CLK)

begin

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

if EC='1' then --clock enable

if DIR='1' then --up/down control

CNT <= CNT - 1; --down counter

else

CNT <= CNT + 1; --up counter

end if;

end if;

end if;

end process;

EC <= PLS;

end block COUNTER;

--

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

-- bus interface

-- (CNT, RD)

-- ==> (DBUS)

--

BUS\_INTERFACE : block

signal T : STD\_LOGIC;

begin

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

T <= not RD and CS;

end block BUS\_INTERFACE;

--

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

-- end of architecture

--

end PHOTO\_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;

--

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

-- PWM : Pulse-Width-Modulation (PWM) genetation

-- (CLK, DB, SWR, CS)

-- ==> (OP)

--

-- designed by Pei-Chong Tang, Dec. 1998

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

--

entity PWM is

port

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

OP : out STD\_LOGIC; --PWM output

CS : in STD\_LOGIC; --chip select

SWR : in STD\_LOGIC; --sync write

CLK : in STD\_LOGIC --system clock

);

end PWM;

architecture PWM\_ARCH of PWM is

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

begin

--

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

-- down counter

-- (CLK)

-- ==> (CNT)

--

COUNTER : block

signal Q : STD\_LOGIC\_VECTOR (9 downto 0);

begin

process (CLK)

begin

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

Q <= Q-1; --down counter

end if;

end process;

CNT <= Q(9 downto 2);

end block COUNTER;

--

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

-- data-latch and comparator

-- (DB, SWR, CNT)

-- ==> (OP)

--

PWM\_COMPARATOR : block

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

signal EC : STD\_LOGIC;

begin

process (CLK)

begin

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

if EC='1' then --CLK enable

Q <= DB; --D-type register

end if;

end if;

end process;

EC <= SWR and CS; --CLK enable

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

end block PWM\_COMPARATOR;

--

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

-- end of architecture

--

end PWM\_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;

--

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

-- SPEED : photo-encoder decoding circuit

-- (CLK, PHTA,PHTB,PHTZ, MODE, ZCLR, RD)

-- ==> (DBUS, ZERO)

--

-- designed by Pei-Chong Tang, Dec. 1998

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

--

entity SPEED is

port

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

PA,PB : in STD\_LOGIC; --photo input

CS : in STD\_LOGIC\_VECTOR (1 downto 0); --chip select

RD : in STD\_LOGIC; --read signal

CLK : in STD\_LOGIC; --system clock

FREQ : in STD\_LOGIC --sample point

);

end SPEED;

architecture SPEED\_ARCH of SPEED is

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

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

signal PHA,PHB : STD\_LOGIC;

signal DIR,PLS : STD\_LOGIC;

begin

--

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

-- low-pass digital filter

-- (PA, PB)

-- ==> (PHA, PHB)

--

FILTER\_A : block -- digital filter for #A

signal D0,D1,D2 : STD\_LOGIC;

begin

process (CLK)

begin

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

D2<=D1; D1<=D0; D0<=PA; --delay 3 clocks

PHA <= (D0 and D1 and D2) or ((D0 or D1 or D2) and PHA);

end if;

end process;

end block FILTER\_A;

FILTER\_B : block -- digital filter for #B

signal D0,D1,D2 : STD\_LOGIC;

begin

process (CLK)

begin

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

D2<=D1; D1<=D0; D0<=PB; --delay 3 clocks

PHB <= (D0 and D1 and D2) or ((D0 or D1 or D2) and PHB);

end if;

end process;

end block FILTER\_B;

--

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

-- up/down decoding

-- (PHA, PHB)

-- ==> (DIR, PLS)

--

DECODER : block

signal DLA,DLB : STD\_LOGIC;

begin

process (CLK)

begin

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

DLA<=PHA; DLB<=PHB; -- delay 1-clock

end if;

end process;

DIR <= (not PHA and DLA and not PHB)

or ( PHA and not DLA and PHB)

or (not PHB and DLB and PHA)

or ( PHB and not DLB and not PHA);

PLS <= (not PHA and DLA) or (PHA and not DLA)

or (not PHB and DLB) or (PHB and not DLB);

end block DECODER;

--

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

-- up/down counter

-- (CLK, MODE, PLS, DIR)

-- ==> (CNT)

--

COUNTER : block

signal EC : STD\_LOGIC;

begin

process (CLK)

begin

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

if EC='1' then --clock enable

if DIR='1' then --up/down control

CNT <= CNT - 1; --down counter

else

CNT <= CNT + 1; --up counter

end if;

end if;

end if;

end process;

EC <= PLS;

end block COUNTER;

--

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

-- speed measurement

--

SPEED\_MEASUREMENT : block

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

signal EC : STD\_LOGIC;

begin

process (CLK)

begin

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

if EC='1' then

DIF <= CNT - DLY; --difference

DLY <= CNT; --delay

end if;

end if;

end process;

EC <= FREQ;

end block SPEED\_MEASUREMENT;

--

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

-- bus interface

-- (CNT, RD)

-- ==> (DBUS)

--

BUS\_INTERFACE : block

signal T0,T1 : STD\_LOGIC;

begin

DB <= CNT when T0='1' else "ZZZZZZZZ";

DB <= DIF when T1='1' else "ZZZZZZZZ";

T0 <= not RD and CS(0);

T1 <= not RD and CS(1);

end block BUS\_INTERFACE;

--

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

-- end of architecture

--

end SPEED\_ARCH;

### **CH014.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;

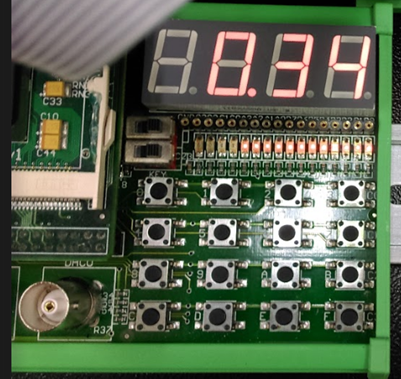
NET YA<1> LOC = Y10;

NET YC<0> LOC = Y11;

NET YC<1> LOC = W11;

## observations

### 1.位置控制:



### 2.速度控制:

