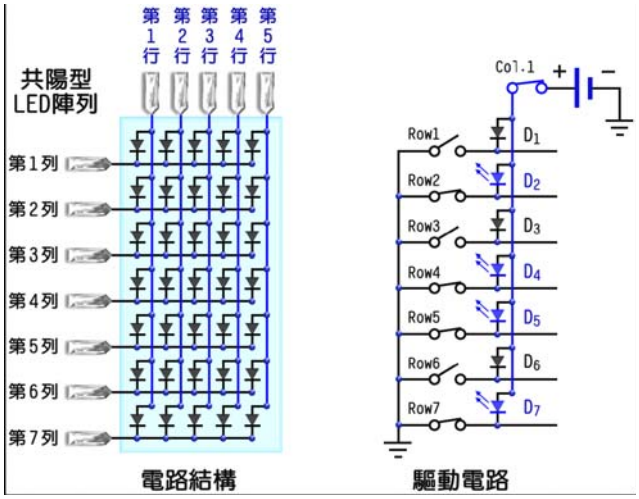
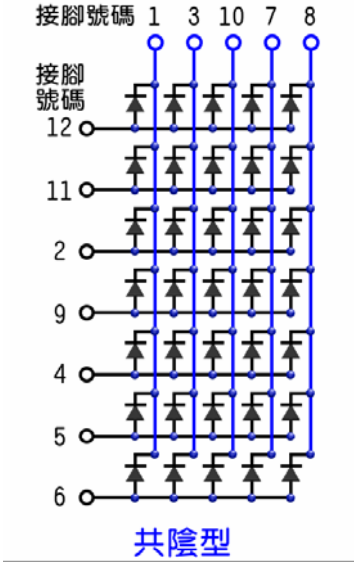
LED陣列：

所謂LED 陣列是將多個LED 以矩陣方式排列，而成為一個零件，其中各LED 的接腳，以規律性連接，如下左圖所示為共陽型5×7 LED 陣列，右圖為共陰型(職訓中心實驗板為共陰型)內部電路架構：





對於共陽型5×7 LED 陣列而言，每行LED 的陽極連接在一起，即為行接腳(column)、而每列LED 的陰極連接在一起，即為列接腳(row)。通常是站在行的角度來看，所以稱為「共陽型」(common anode，簡稱**CA**)。若要點亮其中的LED，則需行的信號與列的信號交集，例如要第1 行、第2 列的LED(即D_2)亮，則須將Col. 1 接腳接到電源(VCC)、Row 2 接腳接地，才能形成一個順向迴路，該LED 才會亮，如上右圖所示。送到行接腳的信號為**掃描信號**，在上圖裡，五個行信號之中，只有一個為高態，其餘為低態，稱為**高態掃描**。換言之，任何時間裡，只有一行的LED可能會亮。而所要點亮的信號，則由列接腳送入低態信號。當信號切換的速度夠快時，我們將感覺到整個LED 陣列是亮的，而不只亮其中一行而已。

若連接到行接腳的是LED 的陰極， 則稱為「共陰型」(common cathode，簡稱**CC**)。若要點亮這種LED 陣列，其行接腳必須採低態掃瞄，而列接腳，則為高態信號。

5x7 點矩陣顯示器練習

matrix_test.vhd

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;

ENTITY matrix_test IS
      PORT(   CLK        :IN STD_LOGIC;
              col_s:out std_logic_vector(0 to 4);
              row_s:out std_logic_vector(0 to 6));
END matrix_test;

ARCHITECTURE a OF matrix_test IS
signal cnt: std_logic_vector (2 downto 0);
BEGIN
PROCESS(clk)
      BEGIN
       IF clk'event and clk='1' then
              CASE cnt IS
                     WHEN "000" => col_s<="01111";row_s<="1000000";
                     WHEN "001" => col_s<="10111";row_s<="0100000";
                     WHEN "010" => col_s<="11011";row_s<="0010000";
                     WHEN "011" => col_s<="11101";row_s<="0001000";
                     WHEN "100" => col_s<="11110";row_s<="0000100";
                     WHEN "101" => col_s<="11100";row_s<="0000010";
                     WHEN "110" => col_s<="11000";row_s<="0000001";
                     WHEN others=> null;
              END CASE;
       end if;
END PROCESS;

-----------------------------------------------------------------------

PROCESS(clk)
      BEGIN
       IF clk'event and clk='1' then
              cnt<=cnt+1;
       end if;
END PROCESS;

END a;
```

5x7 LED 點矩陣為共陰型，故行掃描資料須接地 0，而列掃描資料須為 1，，LED 亮。

產生掃描信號

腳位號碼：

matrix_test@126	CLK
matrix_test@28	col_s0
matrix_test@29	col_s1
matrix_test@30	col_s2
matrix_test@31	col_s3
matrix_test@32	col_s4
matrix_test@18	row_s0
matrix_test@19	row_s1
matrix_test@20	row_s2
matrix_test@21	row_s3
matrix_test@22	row_s4
matrix_test@23	row_s5
matrix_test@26	row_s6

5x7 點矩陣顯示器顯示 0~9 A~F 練習

matrix_number.vhd

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;

ENTITY matrix_number IS
	PORT(	CLK		:IN STD_LOGIC;
		bcd	:in integer range 0 to 15;
		col_s:out std_logic_vector(0 to 4);
		row_s:out std_logic_vector(0 to 6));
END matrix_number;

ARCHITECTURE a OF matrix_number IS
signal cnt: integer range 4 downto 0;
BEGIN
process(clk)
begin
wait until clk='1';
case bcd is
when 0=>
	IF clk'event and clk='1' then
		CASE cnt IS
			WHEN 0 => col_s<="01111";row_s<="0111110";
			WHEN 1 => col_s<="10111";row_s<="1000001";
			WHEN 2 => col_s<="11011";row_s<="1000001";
			WHEN 3 => col_s<="11101";row_s<="1000001";
			WHEN 4 => col_s<="11110";row_s<="0111110";
			WHEN others=> null;
		END CASE;
	end if;
when 1=>
	IF clk'event and clk='1' then
		CASE cnt IS
			WHEN 0 => col_s<="01111";row_s<="0000001";
			WHEN 1 => col_s<="10111";row_s<="0100001";
			WHEN 2 => col_s<="11011";row_s<="1111111";
			WHEN 3 => col_s<="11101";row_s<="0000001";
			WHEN 4 => col_s<="11110";row_s<="0000001";
			WHEN others=> null;
		END CASE;
	end if;
when 2=>
	IF clk'event and clk='1' then
		CASE cnt IS
			WHEN 0 => col_s<="01111";row_s<="0100111";
			WHEN 1 => col_s<="10111";row_s<="1001001";
			WHEN 2 => col_s<="11011";row_s<="1001001";
			WHEN 3 => col_s<="11101";row_s<="1001001";
			WHEN 4 => col_s<="11110";row_s<="0110001";
			WHEN others=> null;
		END CASE;
	end if;
when 3=>
	IF clk'event and clk='1' then
		CASE cnt IS
			WHEN 0 => col_s<="01111";row_s<="0100010";
			WHEN 1 => col_s<="10111";row_s<="1001001";
			WHEN 2 => col_s<="11011";row_s<="1001001";
			WHEN 3 => col_s<="11101";row_s<="1001001";
			WHEN 4 => col_s<="11110";row_s<="0110110";
			WHEN others=> null;
		END CASE;
	end if;
when 4=>
	IF clk'event and clk='1' then
		CASE cnt IS
			WHEN 0 => col_s<="01111";row_s<="0000100";
			WHEN 1 => col_s<="10111";row_s<="0010100";
			WHEN 2 => col_s<="11011";row_s<="1000100";
			WHEN 3 => col_s<="11101";row_s<="1111111";
			WHEN 4 => col_s<="11110";row_s<="0000100";
			WHEN others=> null;
		END CASE;
	end if;
when 5=>
	IF clk'event and clk='1' then
		CASE cnt IS
			WHEN 0 => col_s<="01111";row_s<="1111010";
			WHEN 1 => col_s<="10111";row_s<="1001001";
			WHEN 2 => col_s<="11011";row_s<="1001001";
			WHEN 3 => col_s<="11101";row_s<="1001001";
			WHEN 4 => col_s<="11110";row_s<="1001110";
			WHEN others=> null;
		END CASE;
	end if;
when 6=>
	IF clk'event and clk='1' then
		CASE cnt IS
			WHEN 0 => col_s<="01111";row_s<="0111110";
			WHEN 1 => col_s<="10111";row_s<="1001001";
			WHEN 2 => col_s<="11011";row_s<="1001001";
			WHEN 3 => col_s<="11101";row_s<="1001001";
```

```
                WHEN 4 => col_s<="11110";row_s<="0100110";
                WHEN others=> null;
            END CASE;
    end if;
when 7=>
        IF clk'event and clk='1' then
            CASE cnt IS
                WHEN 0 => col_s<="01111";row_s<="1100000";
                WHEN 1 => col_s<="10111";row_s<="1000000";
                WHEN 2 => col_s<="11011";row_s<="1000000";
                WHEN 3 => col_s<="11101";row_s<="1000000";
                WHEN 4 => col_s<="11110";row_s<="1111111";
                WHEN others=> null;
            END CASE;
    end if;
when 8=>
        IF clk'event and clk='1' then
            CASE cnt IS
                WHEN 0 => col_s<="01111";row_s<="0110110";
                WHEN 1 => col_s<="10111";row_s<="1001001";
                WHEN 2 => col_s<="11011";row_s<="1001001";
                WHEN 3 => col_s<="11101";row_s<="1001001";
                WHEN 4 => col_s<="11110";row_s<="0110110";
                WHEN others=> null;
            END CASE;
    end if;
when 9=>
        IF clk'event and clk='1' then
            CASE cnt IS
                WHEN 0 => col_s<="01111";row_s<="0110010";
                WHEN 1 => col_s<="10111";row_s<="1001001";
                WHEN 2 => col_s<="11011";row_s<="1001001";
                WHEN 3 => col_s<="11101";row_s<="1001001";
                WHEN 4 => col_s<="11110";row_s<="0111110";
                WHEN others=> null;
            END CASE;
    end if;
when 10=>
        IF clk'event and clk='1' then
            CASE cnt IS
                WHEN 0 => col_s<="01111";row_s<="0011111";
                WHEN 1 => col_s<="10111";row_s<="0101000";
                WHEN 2 => col_s<="11011";row_s<="1001000";
                WHEN 3 => col_s<="11101";row_s<="0101000";
                WHEN 4 => col_s<="11110";row_s<="0011111";
                WHEN others=> null;
            END CASE;
    end if;
when 11=>
        IF clk'event and clk='1' then
            CASE cnt IS
                WHEN 0 => col_s<="01111";row_s<="1111111";
                WHEN 1 => col_s<="10111";row_s<="1001001";
                WHEN 2 => col_s<="11011";row_s<="1001001";
                WHEN 3 => col_s<="11101";row_s<="1001001";
                WHEN 4 => col_s<="11110";row_s<="0110110";
                WHEN others=> null;
            END CASE;
    end if;
when 12=>
        IF clk'event and clk='1' then
            CASE cnt IS
                WHEN 0 => col_s<="01111";row_s<="0111110";
                WHEN 1 => col_s<="10111";row_s<="1000001";
                WHEN 2 => col_s<="11011";row_s<="1000001";
                WHEN 3 => col_s<="11101";row_s<="1000001";
                WHEN 4 => col_s<="11110";row_s<="0100010";
                WHEN others=> null;
            END CASE;
    end if;
when 13=>
        IF clk'event and clk='1' then
            CASE cnt IS
                WHEN 0 => col_s<="01111";row_s<="1111111";
                WHEN 1 => col_s<="10111";row_s<="1000001";
                WHEN 2 => col_s<="11011";row_s<="1000001";
                WHEN 3 => col_s<="11101";row_s<="1000001";
                WHEN 4 => col_s<="11110";row_s<="0111110";
                WHEN others=> null;
            END CASE;
    end if;
when 14=>
        IF clk'event and clk='1' then
            CASE cnt IS
                WHEN 0 => col_s<="01111";row_s<="1111111";
                WHEN 1 => col_s<="10111";row_s<="1001001";
                WHEN 2 => col_s<="11011";row_s<="1001001";
                WHEN 3 => col_s<="11101";row_s<="1001001";
                WHEN 4 => col_s<="11110";row_s<="1001001";
                WHEN others=> null;
            END CASE;
    end if;
when 15=>
        IF clk'event and clk='1' then
            CASE cnt IS
                WHEN 0 => col_s<="01111";row_s<="1111111";
                WHEN 1 => col_s<="10111";row_s<="1001000";
                WHEN 2 => col_s<="11011";row_s<="1001000";
                WHEN 3 => col_s<="11101";row_s<="1001000";
                WHEN 4 => col_s<="11110";row_s<="1001000";
                WHEN others=> null;
            END CASE;
    end if;
when others=> null;
end case;
end process;
```

```
PROCESS(clk)
    BEGIN
     IF clk'event and clk='1' then
            if cnt=0 then
                cnt<=4;
            else cnt<=cnt-1;
            end if;
     end if;
END PROCESS;
END a;
```

利用 4x4 鍵盤輸入 5x7 點矩陣顯示器輸出練習

keyin.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;
entity keyin is
port(
	clk: in std_logic;
	y : in std_logic_vector(3 downto 0);
	x : out std_logic_vector(3 downto 0);
	col_s : out std_logic_vector(0 to 4);
	row_s: out std_logic_vector(0 to 6)) ;
end keyin;
architecture key of keyin is
signal bcd	:	integer range 0 to 15;
signal scan	:	integer range 3 downto 0;
--*************************************
component matrix_number
PORT(	CLK	:IN STD_LOGIC;
		bcd	:in integer range 0 to 15;
		col_s:out std_logic_vector(0 to 4);
		row_s:out std_logic_vector(0 to 6));
end component;
--*************************************

begin
--******************************
key_scan:
process(clk)
	begin
	  wait until clk='1' ;
	if y="0000" then
		if scan=0 then
			scan<=3;
		else scan<=scan-1;
		end if;
	end if;
end process key_scan;
--*************************
key_x_scan:
process(clk)
begin
	case scan is
		when 3 => x<="1000";
		when 2 => x<="0100";
		when 1 => x<="0010";
		when 0 => x<="0001";
		when others=> null;
	end case;
end process key_x_scan;
--*************************
```

5x7 LED 矩陣顯示器顯示電路之輸出入接腳宣告

```
key_read:
process(y)
begin
if scan=3 then
      case y is
              when "1000" => bcd<=3;
              when "0100" => bcd<=2;
              when "0010" => bcd<=1;
              when "0001" => bcd<=0;
              when others => null;
            end case;
      elsif scan=2 then
        case y is
              when "1000" => bcd<=7;
              when "0100" => bcd<=6;
              when "0010" => bcd<=5;
              when "0001" => bcd<=4;
              when others => null;
            end case;
      elsif scan=1 then
        case y is
              when "1000" => bcd<=11;
              when "0100" => bcd<=10;
              when "0010" => bcd<=9;
              when "0001" => bcd<=8;
              when others => null;
            end case;
      elsif scan=0 then
        case y is
              when "1000" => bcd<=15;
              when "0100" => bcd<=14;
              when "0010" => bcd<=13;
              when "0001" => bcd<=12;
              when others => null;
            end case;
      end if ;
end process key_read;
--******************************
number_decode:matrix_number
port map(clk,bcd,col_s,row_s);
end key    ;
```

呼叫 5x7 LED 矩陣顯示器顯示解碼電路

將上一個程式再改成用兩個副程式呼叫(component..port map)的方式，讓程式看來更為簡短且結構化

主程式 keyin_number.vhd 及 keyin_read_code.vhd、matrix_number.vhd 兩副程式

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;
entity keyin_number is
port(
      clk: in std_logic;
      y : in std_logic_vector(3 downto 0); --鍵盤 Y 列掃描讀入訊號
      x : out std_logic_vector(3 downto 0); --鍵盤 X 行掃描輸出訊號
      col_s : out std_logic_vector(0 to 4); --LED 行掃描輸出訊號
      row_s: out std_logic_vector(0 to 6)) ; --LED 列掃描輸出訊號
end keyin_number;
--************************************
architecture key of keyin_number is
signal bcd      :      integer range 0 to 15;
signal scan     :      integer range 3 downto 0;
--************************************
component keyin_read_code
port(
      clk: in std_logic;
      y : in std_logic_vector(3 downto 0);
      x : out std_logic_vector(3 downto 0);
      bcd  : out integer range 0 to 15);
end component;
--************************************
component matrix_number
PORT(    CLK        :IN STD_LOGIC;
                bcd    :in integer range 0 to 15;
                col_s:out std_logic_vector(0 to 4);
                row_s:out std_logic_vector(0 to 6));
end component;
--************************************
begin
read_keyin:keyin_read_code
port map(clk,y,x,bcd);
--****************************
number_decode:matrix_number
port map(clk,bcd,col_s,row_s);
end key    ;
```

鍵盤按鍵訊號編碼讀入

LED 矩陣訊號編碼掃瞄輸出

兩副程式之接腳對應關係

副程式 keyin_read_code.vhd
(鍵盤掃描輸入編碼)

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;
entity keyin_read_code is
port(
	clk: in std_logic;
	y : in std_logic_vector(3 downto 0);
	x : out std_logic_vector(3 downto 0);
	bcd  : out integer range 0 to 15);
end keyin_read_code;
architecture key of keyin_read_code is
signal scan	:	integer range 3 downto 0;

begin
--******************************
key_scan:
process(clk)
	begin
	  wait until clk='1' ;
	if y="0000" then
		if scan=0 then
			scan<=3;
		else scan<=scan-1;
		end if;
	end if;
end process key_scan;
--*************************
key_x_scan:
process(clk)
begin
	case scan is
		when 3 => x<="1000";
		when 2 => x<="0100";
		when 1 => x<="0010";
		when 0 => x<="0001";
		when others=> null;
	end case;
end process key_x_scan;
--*************************
key_read:
process(y)
begin
if scan=3 then
	case y is
		when "1000" => bcd<=3;
		when "0100" => bcd<=2;
		when "0010" => bcd<=1;
		when "0001" => bcd<=0;
		when others => null;
	      end case;
	elsif scan=2 then
	  case y is
		when "1000" => bcd<=7;
		when "0100" => bcd<=6;
		when "0010" => bcd<=5;
		when "0001" => bcd<=4;
		when others => null;
	      end case;
	elsif scan=1 then
	  case y is
		when "1000" => bcd<=11;
		when "0100" => bcd<=10;
		when "0010" => bcd<=9;
		when "0001" => bcd<=8;
		when others => null;
	      end case;
	elsif scan=0 then
	  case y is
		when "1000" => bcd<=15;
		when "0100" => bcd<=14;
		when "0010" => bcd<=13;
		when "0001" => bcd<=12;
		when others => null;
	      end case;
	end if ;
end process key_read;
end key	;
```

** DEVICE SUMMARY **

Chip/ POF	Device	Input Pins	Output Pins	Bidir Pins	Memory Bits	% Utilized	Memory LCs	% Utilized
keyin_number								
EPF10K10TC144-4		5	16	0	0	0 %	145	25 %
User Pins:		5	16	0				

Project Information d:\ex\matrix\keyin_number.rpt

** PIN/LOCATION/CHIP ASSIGNMENTS **

User Assignments	Actual Assignments (if different)	Node Name
keyin_number@126		clk
keyin_number@28		col_s0
keyin_number@29		col_s1
keyin_number@30		col_s2
keyin_number@31		col_s3
keyin_number@32		col_s4
keyin_number@18		row_s0
keyin_number@19		row_s1
keyin_number@20		row_s2
keyin_number@21		row_s3
keyin_number@22		row_s4
keyin_number@23		row_s5
keyin_number@26		row_s6
keyin_number@89		x0
keyin_number@90		x1
keyin_number@91		x2
keyin_number@92		x3
keyin_number@98		y0
keyin_number@97		y1
keyin_number@96		y2
keyin_number@95		y3

** DEVICE SUMMARY **

Chip/ POF	Device	Input Pins	Output Pins	Bidir Pins	Memory Bits	% Utilized	Memory LCs	% Utilized
keyin								
EPF10K10TC144-4		5	16	0	0	0 %	145	25 %
User Pins:		5	16	0				

Project Information d:\ex\matrix\keyin.rpt

** PIN/LOCATION/CHIP ASSIGNMENTS **

User Assignments	Actual Assignments (if different)	Node Name
keyin@126		clk
keyin@28		col_s0
keyin@29		col_s1
keyin@30		col_s2
keyin@31		col_s3
keyin@32		col_s4
keyin@18		row_s0
keyin@19		row_s1
keyin@20		row_s2
keyin@21		row_s3
keyin@22		row_s4
keyin@23		row_s5
keyin@26		row_s6
keyin@89		x0
keyin@90		x1
keyin@91		x2
keyin@92		x3
keyin@98		y0
keyin@97		y1
keyin@96		y2
keyin@95		y3

兩程式不同的寫法但編譯完後電路所使用的大小是一樣的

LED 矩陣移位練習(1)(由右到左整行輪流亮燈)

matrix_scan_test.vhd

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;

ENTITY matrix_scan_test IS
	PORT(	CLK		:IN STD_LOGIC;
		col_s:out std_logic_vector(0 to 4);
		row_s:out std_logic_vector(0 to 6));
END matrix_scan_test;

ARCHITECTURE shift OF matrix_scan_test IS
signal cnt: integer range 0 to 5;
signal clk_1hz,clk_1khz: std_logic;

--************************************
component mod_n1_n2hz
port ( clk: in std_logic;
		clk_1khz	: buffer std_logic;
		clk_1hz	: buffer std_logic);
end component;
--************************************
BEGIN
clk_mod:mod_n1_n2hz
port map(clk,clk_1khz,clk_1hz);


PROCESS(clk_1hz)
	BEGIN
	 IF clk_1hz'event and clk_1hz='1' then
		CASE cnt IS
			WHEN 0 => col_s<="01111";row_s<="1111111";
			WHEN 1 => col_s<="10111";row_s<="1111111";
			WHEN 2 => col_s<="11011";row_s<="1111111";
			WHEN 3 => col_s<="11101";row_s<="1111111";
			WHEN 4 => col_s<="11110";row_s<="1111111";
			WHEN others=> null;
		END CASE;
	 end if;
END PROCESS;

-----------------------------------------------------------------------

PROCESS
	BEGIN
wait until clk_1hz='1';
	IF cnt=4 then
		cnt<=0;
	 else cnt<=cnt+1;
	end if;
```

```
END PROCESS;

END shift;
```

LED 矩陣移位練習(2)(由左到右整行輪流亮燈)

matrix_scan_shift.vhd

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;

ENTITY matrix_scan_shift IS
      PORT(   CLK        :IN STD_LOGIC;
              col_s:buffer std_logic_vector(0 to 4);
              row_s:out std_logic_vector(0 to 6));
END matrix_scan_shift;

ARCHITECTURE shift OF matrix_scan_shift IS
signal clk_1hz,clk_1khz: std_logic;
--signal s,sh : std_logic_vector(4 downto 0) ;

--************************************
component mod_n1_n2hz
port ( clk: in std_logic;
            clk_1khz      : buffer std_logic;
            clk_1hz      : buffer std_logic);
end component;
--************************************
BEGIN
clk_mod:mod_n1_n2hz
port map(clk,clk_1khz,clk_1hz);
--****************************
PROCESS
  BEGIN
  wait until clk_1hz='1';
      if col_s="00000" then
            col_s<="01111" ;
      else col_s<=col_s(1 to 4)&col_s(0); --移位
      row_s<="1111111";
      end if;
  END PROCESS;
END shift;
```

將系統之 20MHZ 訊號除頻成 1KHZ 及 1HZ

LED 矩陣移位練習(3)(數字 0~9 A~F 由右向左移)
主程式 matrix_shift.vhd，副程式 matrix_scan_mdu.vhd

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;
USE IEEE.std_logic_unsigned.all;
USE IEEE.std_logic_arith.all;

ENTITY matrix_shift IS
    PORT(
        clk: IN    STD_LOGIC;    --LED 矩陣掃瞄訊號
        shift: IN STD_LOGIC;     --移位停止訊號（LED 矩陣不顯示）
        GX: OUT STD_LOGIC_VECTOR(0 to 6);    --Y 列接腳
        CH_OUT: OUT STD_LOGIC_VECTOR(0 to 4)    --X 行接腳
        );
    END matrix_shift;

ARCHITECTURE beh OF matrix_shift IS

  component matrix_scan_mdu
  PORT(
        fin: IN    STD_LOGIC;
        chin: in STD_LOGIC_VECTOR(0 to 34);
        change: in STD_LOGIC;
        gxx: OUT STD_LOGIC_VECTOR(0 to 6);
        chout: OUT STD_LOGIC_VECTOR(0 to 4));
  END component;
```

LED 矩陣掃描顯示電路(副程式)

```
 CONSTANT h0:STD_LOGIC_VECTOR(0 TO 34):="10001"
                                        &"01110"
                                        &"01110"
                                        &"01110"
                                        &"01110"
                                        &"01110"
                                        &"10001";
 CONSTANT h1:STD_LOGIC_VECTOR(0 TO 34):="11011"
                                        &"10011"
                                        &"11011"
                                        &"11011"
                                        &"11011"
                                        &"11011"
                                        &"00000";
 CONSTANT h2:STD_LOGIC_VECTOR(0 TO 34):="10001"
                                        &"01110"
                                        &"11110"
                                        &"10001"
                                        &"01111"
                                        &"01111"
                                        &"00000";
 CONSTANT h3:STD_LOGIC_VECTOR(0 TO 34):="10001"
                                        &"01110"
                                        &"11110"
                                        &"10001"
```

顯示字型之編碼，排列成矩陣形狀，為行(X 軸)之資料，因為是共陰極接線，所以 0 為顯示(亮)，1 為不亮。

```
                                                    &"11110"
                                                    &"01110"
                                                    &"10001";
CONSTANT h4:STD_LOGIC_VECTOR(0 TO 34):="11001"
                                                    &"11101"
                                                    &"10101"
                                                    &"11101"
                                                    &"00000"
                                                    &"11101"
                                                    &"11101";
CONSTANT h5:STD_LOGIC_VECTOR(0 TO 34):="00000"
                                                    &"01111"
                                                    &"01111"
                                                    &"00000"
                                                    &"11110"
                                                    &"01110"
                                                    &"10001";
CONSTANT h6:STD_LOGIC_VECTOR(0 TO 34):="10001"
                                                    &"01110"
                                                    &"01111"
                                                    &"00001"
                                                    &"01110"
                                                    &"01110"
                                                    &"10001";
CONSTANT h7:STD_LOGIC_VECTOR(0 TO 34):="00000"
                                                    &"01110"
                                                    &"11110"
                                                    &"11110"
                                                    &"11110"
                                                    &"11110"
                                                    &"11110";
CONSTANT h8:STD_LOGIC_VECTOR(0 TO 34):="10001"
                                                    &"01110"
                                                    &"01110"
                                                    &"10001"
                                                    &"01110"
                                                    &"01110"
                                                    &"10001";
CONSTANT h9:STD_LOGIC_VECTOR(0 TO 34):="10001"
                                                    &"01110"
                                                    &"01110"
                                                    &"10000"
                                                    &"11110"
                                                    &"01110"
                                                    &"10001";
 CONSTANT a:STD_LOGIC_VECTOR(0 TO 34):="11011"
                                                    &"10101"
                                                    &"01110"
                                                    &"00000"
                                                    &"01110"
                                                    &"01110"
                                                    &"01110";
```

顯示字型之編碼，排列成矩陣形狀，為行(X 軸)之資料，因為是共陰極接線，所以 0 為顯示(亮)，1 為不亮。

```
CONSTANT b:STD_LOGIC_VECTOR(0 TO 34):="00001"
                                      &"01110"
                                      &"01110"
                                      &"00001"
                                      &"01110"
                                      &"01110"
                                      &"00001";
CONSTANT c:STD_LOGIC_VECTOR(0 TO 34):="10001"
                                      &"01110"
                                      &"01111"
                                      &"01111"
                                      &"01111"
                                      &"01110"
                                      &"10001";
CONSTANT d:STD_LOGIC_VECTOR(0 TO 34):="00001"
                                      &"01110"
                                      &"01110"
                                      &"01110"
                                      &"01110"
                                      &"01110"
                                      &"00001";
CONSTANT e:STD_LOGIC_VECTOR(0 TO 34):="00000"
                                      &"01111"
                                      &"01111"
                                      &"00000"
                                      &"01111"
                                      &"01111"
                                      &"00000";
CONSTANT f:STD_LOGIC_VECTOR(0 TO 34):="00000"
                                      &"01111"
                                      &"01111"
                                      &"00000"
                                      &"01111"
                                      &"01111"
                                      &"01111";

signal f_row,f_shift:std_logic;
signal ch_f,ch_b:STD_LOGIC_VECTOR(0 TO 34);
signal fram_index:STD_LOGIC_VECTOR(0 TO 3);

BEGIN


   freq:process(clk)
      variable modu:std_logic_vector(23 downto 0);
   begin
      if clk='1' and clk'event then
         modu:=modu-1;
      end if;
      f_row<=modu(8);
      f_shift<=modu(22);
   end process freq;
```

顯示字型之編碼，排列成矩陣形狀，為行(X 軸)之資料，因為是共陰極接線，所以 0 為顯示(亮)，1 為不亮。

除頻訊號：掃描及移位訊號

```
shifting:PROCESS(f_shift)
variable sh_index:STD_LOGIC_VECTOR(0 TO 2);
begin
if f_shift='0' and f_shift'event then
  if shift='0' then
        if sh_index>=7 then
             if fram_index>=15 then
                  fram_index<="0000";sh_index:="000";
             else
                  fram_index<=fram_index+1;sh_index:="000";
             end if;
        else
             sh_index:=sh_index+1;    --資料位移訊號
        end if;
        case sh_index is
        when "001"=>
        ch_f( 0 to   4)<=ch_f( 1 to   4)&'1';
        ch_f( 5 to   9)<=ch_f( 6 to   9)&'1';
        ch_f(10 to 14)<=ch_f(11 to 14)&'1';
        ch_f(15 to 19)<=ch_f(16 to 19)&'1';
        ch_f(20 to 24)<=ch_f(21 to 24)&'1';
        ch_f(25 to 29)<=ch_f(26 to 29)&'1';
        ch_f(30 to 34)<=ch_f(31 to 34)&'1';
        when "010"=>
        ch_f( 0 to   4)<=ch_f( 1 to   4)&ch_b(0);
        ch_f( 5 to   9)<=ch_f( 6 to   9)&ch_b(5);
        ch_f(10 to 14)<=ch_f(11 to 14)&ch_b(10);
        ch_f(15 to 19)<=ch_f(16 to 19)&ch_b(15);
        ch_f(20 to 24)<=ch_f(21 to 24)&ch_b(20);
        ch_f(25 to 29)<=ch_f(26 to 29)&ch_b(25);
        ch_f(30 to 34)<=ch_f(31 to 34)&ch_b(30);
        when "011"=>
        ch_f( 0 to   4)<=ch_f( 1 to   4)&ch_b(1);
        ch_f( 5 to   9)<=ch_f( 6 to   9)&ch_b(6);
        ch_f(10 to 14)<=ch_f(11 to 14)&ch_b(11);
        ch_f(15 to 19)<=ch_f(16 to 19)&ch_b(16);
        ch_f(20 to 24)<=ch_f(21 to 24)&ch_b(21);
        ch_f(25 to 29)<=ch_f(26 to 29)&ch_b(26);
        ch_f(30 to 34)<=ch_f(31 to 34)&ch_b(31);
        when "100"=>
        ch_f( 0 to   4)<=ch_f( 1 to   4)&ch_b(2);
        ch_f( 5 to   9)<=ch_f( 6 to   9)&ch_b(7);
        ch_f(10 to 14)<=ch_f(11 to 14)&ch_b(12);
        ch_f(15 to 19)<=ch_f(16 to 19)&ch_b(17);
        ch_f(20 to 24)<=ch_f(21 to 24)&ch_b(22);
        ch_f(25 to 29)<=ch_f(26 to 29)&ch_b(27);
        ch_f(30 to 34)<=ch_f(31 to 34)&ch_b(32);
        when "101"=>
        ch_f( 0 to   4)<=ch_f( 1 to   4)&ch_b(3);
        ch_f( 5 to   9)<=ch_f( 6 to   9)&ch_b(8);
```

字型顯示順序之訊號

先移進一行空白行

將字型最左一行資料移入最右邊之 LED 矩陣

將字型左邊第二行資料移入最右邊之 LED 矩陣，以下依序將資料移入最右邊矩陣，而原本資料則左移一行

```
            ch_f(10 to 14)<=ch_f(11 to 14)&ch_b(13);
            ch_f(15 to 19)<=ch_f(16 to 19)&ch_b(18);
            ch_f(20 to 24)<=ch_f(21 to 24)&ch_b(23);
            ch_f(25 to 29)<=ch_f(26 to 29)&ch_b(28);
            ch_f(30 to 34)<=ch_f(31 to 34)&ch_b(33);
            when "110"=>
            ch_f( 0 to    4)<=ch_f( 1 to    4)&ch_b(4);
            ch_f( 5 to    9)<=ch_f( 6 to    9)&ch_b(9);
            ch_f(10 to 14)<=ch_f(11 to 14)&ch_b(14);
            ch_f(15 to 19)<=ch_f(16 to 19)&ch_b(19);
            ch_f(20 to 24)<=ch_f(21 to 24)&ch_b(24);
            ch_f(25 to 29)<=ch_f(26 to 29)&ch_b(29);
            ch_f(30 to 34)<=ch_f(31 to 34)&ch_b(34);
            when others=>null;
            end case;
        end if;
      end if;
  end process shifting;

  change_frame:process(fram_index)
     begin
        case fram_index is
        when "0000"=>ch_b<=h0;
        when "0001"=>ch_b<=h1;
        when "0010"=>ch_b<=h2;
        when "0011"=>ch_b<=h3;
        when "0100"=>ch_b<=h4;
        when "0101"=>ch_b<=h5;
        when "0110"=>ch_b<=h6;
        when "0111"=>ch_b<=h7;
        when "1000"=>ch_b<=h8;
        when "1001"=>ch_b<=h9;
        when "1010"=>ch_b<=a;
        when "1011"=>ch_b<=b;
        when "1100"=>ch_b<=c;
        when "1101"=>ch_b<=d;
        when "1110"=>ch_b<=e;
        when "1111"=>ch_b<=f;
        when others=>null;
        end case;
  end process change_frame;
```

顯示移位字型之順序

```
scan_wt:matrix_scan_mdu port map(f_row,ch_f,shift,gx,ch_out);
```

--呼叫 LED 矩陣掃描顯示電路

```
END beh;
```

上一程式之副程式

副程式 matrix_scan_mdu.vhd

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;
USE IEEE.std_logic_unsigned.all;
USE IEEE.std_logic_arith.all;

ENTITY matrix_scan_mdu IS
	PORT(
		fin: IN	STD_LOGIC;                          --掃描頻率
		chin: in STD_LOGIC_VECTOR(0 to 34);    --字型資料串
		change: in STD_LOGIC;                          --停止移位按鍵
		gxx: OUT STD_LOGIC_VECTOR(0 to 6);    --列掃描訊號，高態動作
		chout: OUT STD_LOGIC_VECTOR(0 to 4)    --行資料訊號，低態動作
		);
	END matrix_scan_mdu;

ARCHITECTURE beh OF matrix_scan_mdu IS
   type r_state is (rs0,rs1,rs2,rs3,rs4,rs5,rs6);
   signal pre_state,next_state:r_state;
   signal change_r:STD_LOGIC;

BEGIN

to_next_state:process(fin)
begin
   if fin='0' and fin'event then
        pre_state<=next_state;
   end if;
end process to_next_state;
```

觸發到下一個訊號

列掃瞄之狀態機設計法

```
scan_5_7:process
begin
if change='0' then change_r<='1';
else change_r<='0';
end if;
     case pre_state is
     when rs0      =>if change_r='1' then
                             next_state<=rs1;
                             gxx<="1000000";
                             chout<=chin(0 to 4);
                        else
                             next_state<=rs0;
                             gxx<="0000000";    --當壓下停止移位鍵，LED 矩陣不顯示
                        end if;
       when rs1      =>if change_r='1' then
                             next_state<=rs2;
                             gxx<="0100000";
                             chout<=chin(5 to 9);
                        else
```

```
                        gxx<="0000000";
                        next_state<=rs1;
                    end if;
    when rs2    =>if change_r='1' then
                        next_state<=rs3;
                        gxx<="0010000";
                        chout<=chin(10 to 14);
                    else
                        gxx<="0000000";
                        next_state<=rs2;
                    end if;
    when rs3    =>if change_r='1' then
                        next_state<=rs4;
                        gxx<="0001000";
                        chout<=chin(15 to 19);
                    else
                        gxx<="0000000";
                        next_state<=rs3;
                    end if;
    when rs4    =>if change_r='1' then
                        next_state<=rs5;
                        gxx<="0000100";
                        chout<=chin(20 to 24);
                    else
                        gxx<="0000000";
                        next_state<=rs4;
                    end if;
    when rs5    =>if change_r='1' then
                        next_state<=rs6;
                        gxx<="0000010";
                        chout<=chin(25 to 29);
                    else
                        gxx<="0000000";
                        next_state<=rs5;
                    end if;
    when rs6    =>if change_r='1' then
                        next_state<=rs0;
                        gxx<="0000001";
                        chout<=chin(30 to 34);
                      else
                        gxx<="0000000";
                        next_state<=rs6;
                      end if;
    end case;
end process scan_5_7;
END beh;
```

LED 矩陣移位練習(3)(數字 0~9 A~F 由右向左移)
主程式 matrix_shift_a.vhd，副程式 scanwt_mdu.vhd

加入掃描加權電路，讓 LED 矩陣之 LED 顯示不會因為同一時間掃瞄輸出之點亮顆數不同，造成 LED 顯示出亮度不一樣之問題，方法是再同一掃描時間如亮 1 顆燈則掃瞄 1 次，亮兩顆則重複掃描 2 次，3 科 3 次，則能達到史 LED 所顯示之亮度均勻之目的。

主程式與上一程式相同，只修正副程式之輸出入點數目

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;
USE IEEE.std_logic_unsigned.all;
USE IEEE.std_logic_arith.all;

ENTITY matrix_shift_a IS
        PORT(
           clk: IN        STD_LOGIC;
           shift: IN STD_LOGIC;
           GX: OUT STD_LOGIC_VECTOR(0 to 6);
           CH_OUT: OUT STD_LOGIC_VECTOR(0 to 4)
           );
        END matrix_shift_a;

ARCHITECTURE beh OF matrix_shift_a IS

  component scanwt_mdu
  PORT(
           fin: IN        STD_LOGIC;
           chin: in STD_LOGIC_VECTOR(0 to 34);
           gxx: OUT STD_LOGIC_VECTOR(0 to 6);
           chout: OUT STD_LOGIC_VECTOR(0 to 4));
  END component;

 CONSTANT h0:STD_LOGIC_VECTOR(0 TO 34):="10001"
                                        &"01110"
                                        &"01110"
                                        &"01110"
                                        &"01110"
                                        &"01110"
                                        &"10001";
 CONSTANT h1:STD_LOGIC_VECTOR(0 TO 34):="11011"
                                        &"10011"
                                        &"11011"
                                        &"11011"
                                        &"11011"
                                        &"11011"
                                        &"00000";
 CONSTANT h2:STD_LOGIC_VECTOR(0 TO 34):="10001"
                                        &"01110"
                                        &"11110"
                                        &"10001"
                                        &"01111"
                                        &"01111"
                                        &"00000";
 CONSTANT h3:STD_LOGIC_VECTOR(0 TO 34):="10001"
                                        &"01110"
                                        &"11110"
                                        &"10001"
                                        &"11110"
                                        &"01110"
                                        &"10001";
 CONSTANT h4:STD_LOGIC_VECTOR(0 TO 34):="11001"
                                        &"11101"
                                        &"10101"
                                        &"11101"
                                        &"00000"
                                        &"11101"
                                        &"11101";
 CONSTANT h5:STD_LOGIC_VECTOR(0 TO 34):="00000"
                                        &"01111"
                                        &"01111"
                                        &"00000"
                                        &"11110"
                                        &"01110"
                                        &"10001";
 CONSTANT h6:STD_LOGIC_VECTOR(0 TO 34):="10001"
                                        &"01110"
                                        &"01111"
                                        &"00001"
                                        &"01110"
                                        &"01110"
                                        &"10001";
 CONSTANT h7:STD_LOGIC_VECTOR(0 TO 34):="00000"
                                        &"01110"
                                        &"11110"
                                        &"11110"
                                        &"11110"
                                        &"11110"
                                        &"11110";
 CONSTANT h8:STD_LOGIC_VECTOR(0 TO 34):="10001"
                                        &"01110"
                                        &"01110"
                                        &"10001"
                                        &"01110"
                                        &"01110"
                                        &"10001";
 CONSTANT h9:STD_LOGIC_VECTOR(0 TO 34):="10001"
                                        &"01110"
                                        &"01110"
                                        &"10000"
                                        &"11110"
                                        &"01110"
                                        &"10001";
 CONSTANT a:STD_LOGIC_VECTOR(0 TO 34):="11011"
                                        &"10101"
                                        &"01110"
                                        &"00000"
                                        &"01110"
                                        &"01110"
                                        &"01110";
 CONSTANT b:STD_LOGIC_VECTOR(0 TO 34):="00001"
                                        &"01110"
                                        &"01110"
                                        &"00001"
                                        &"01110"
                                        &"01110"
                                        &"00001";
 CONSTANT c:STD_LOGIC_VECTOR(0 TO 34):="10001"
                                        &"01110"
                                        &"01111"
                                        &"01111"
                                        &"01111"
                                        &"01110"
                                        &"10001";
 CONSTANT d:STD_LOGIC_VECTOR(0 TO 34):="00001"
                                        &"01110"
                                        &"01110"
                                        &"01110"
                                        &"01110"
                                        &"01110"
                                        &"00001";
 CONSTANT e:STD_LOGIC_VECTOR(0 TO 34):="00000"
                                        &"01111"
                                        &"01111"
                                        &"00000"
                                        &"01111"
                                        &"01111"
                                        &"00000";
 CONSTANT f:sTD_LOGIC_VECTOR(0 TO 34):="00000"
                                        &"01111"
                                        &"01111"
                                        &"00000"
```

```
                                        &"01111"
                                        &"01111"
                                        &"01111";

signal f_row,f_shift:std_logic;
signal ch_f,ch_b:STD_LOGIC_VECTOR(0 TO 34);
signal fram_index:STD_LOGIC_VECTOR(0 TO 3);

BEGIN


  freq:process(clk)
    variable modu:std_logic_vector(23 downto 0);
    begin
    if clk='1' and clk'event then
        modu:=modu-1;
    end if;
    f_row<=modu(8);
    f_shift<=modu(22);
    end process freq;


  shifting:PROCESS(f_shift)
  variable sh_index:STD_LOGIC_VECTOR(0 TO 2);
  begin
  if f_shift='0' and f_shift'event then
   if shift='0' then
        if sh_index>=7 then
            if fram_index>=15 then
                fram_index<="0000";sh_index:="000";
            else
                fram_index<=fram_index+1;sh_index:="000";
            end if;
        else
            sh_index:=sh_index+1;
        end if;
        case sh_index is
        when "001"=>
        ch_f( 0 to   4)<=ch_f( 1 to   4)&'1';
        ch_f( 5 to   9)<=ch_f( 6 to   9)&'1';
        ch_f(10 to 14)<=ch_f(11 to 14)&'1';
        ch_f(15 to 19)<=ch_f(16 to 19)&'1';
        ch_f(20 to 24)<=ch_f(21 to 24)&'1';
        ch_f(25 to 29)<=ch_f(26 to 29)&'1';
        ch_f(30 to 34)<=ch_f(31 to 34)&'1';
        when "010"=>
        ch_f( 0 to   4)<=ch_f( 1 to   4)&ch_b(0);
        ch_f( 5 to   9)<=ch_f( 6 to   9)&ch_b(5);
        ch_f(10 to 14)<=ch_f(11 to 14)&ch_b(10);
        ch_f(15 to 19)<=ch_f(16 to 19)&ch_b(15);
        ch_f(20 to 24)<=ch_f(21 to 24)&ch_b(20);
        ch_f(25 to 29)<=ch_f(26 to 29)&ch_b(25);
        ch_f(30 to 34)<=ch_f(31 to 34)&ch_b(30);
        when "011"=>
        ch_f( 0 to   4)<=ch_f( 1 to   4)&ch_b(1);
        ch_f( 5 to   9)<=ch_f( 6 to   9)&ch_b(6);
        ch_f(10 to 14)<=ch_f(11 to 14)&ch_b(11);
        ch_f(15 to 19)<=ch_f(16 to 19)&ch_b(16);
        ch_f(20 to 24)<=ch_f(21 to 24)&ch_b(21);
        ch_f(25 to 29)<=ch_f(26 to 29)&ch_b(26);
        ch_f(30 to 34)<=ch_f(31 to 34)&ch_b(31);
        when "100"=>
        ch_f( 0 to   4)<=ch_f( 1 to   4)&ch_b(2);
        ch_f( 5 to   9)<=ch_f( 6 to   9)&ch_b(7);
        ch_f(10 to 14)<=ch_f(11 to 14)&ch_b(12);
        ch_f(15 to 19)<=ch_f(16 to 19)&ch_b(17);
        ch_f(20 to 24)<=ch_f(21 to 24)&ch_b(22);
        ch_f(25 to 29)<=ch_f(26 to 29)&ch_b(27);
        ch_f(30 to 34)<=ch_f(31 to 34)&ch_b(32);
        when "101"=>
        ch_f( 0 to   4)<=ch_f( 1 to   4)&ch_b(3);
        ch_f( 5 to   9)<=ch_f( 6 to   9)&ch_b(8);
        ch_f(10 to 14)<=ch_f(11 to 14)&ch_b(13);
        ch_f(15 to 19)<=ch_f(16 to 19)&ch_b(18);
        ch_f(20 to 24)<=ch_f(21 to 24)&ch_b(23);
        ch_f(25 to 29)<=ch_f(26 to 29)&ch_b(28);
        ch_f(30 to 34)<=ch_f(31 to 34)&ch_b(33);
        when "110"=>
        ch_f( 0 to   4)<=ch_f( 1 to   4)&ch_b(4);
        ch_f( 5 to   9)<=ch_f( 6 to   9)&ch_b(9);
        ch_f(10 to 14)<=ch_f(11 to 14)&ch_b(14);
        ch_f(15 to 19)<=ch_f(16 to 19)&ch_b(19);
        ch_f(20 to 24)<=ch_f(21 to 24)&ch_b(24);
        ch_f(25 to 29)<=ch_f(26 to 29)&ch_b(29);
        ch_f(30 to 34)<=ch_f(31 to 34)&ch_b(34);
        when others=>null;
        end case;
      end if;
     end if;
 end process shifting;

 change_frame:process(fram_index)
   begin
     case fram_index is
     when "0000"=>ch_b<=h0;
     when "0001"=>ch_b<=h1;
     when "0010"=>ch_b<=h2;
     when "0011"=>ch_b<=h3;
     when "0100"=>ch_b<=h4;
     when "0101"=>ch_b<=h5;
     when "0110"=>ch_b<=h6;
     when "0111"=>ch_b<=h7;
     when "1000"=>ch_b<=h8;
     when "1001"=>ch_b<=h9;
     when "1010"=>ch_b<=a;
     when "1011"=>ch_b<=b;
     when "1100"=>ch_b<=c;
     when "1101"=>ch_b<=d;
     when "1110"=>ch_b<=e;
     when "1111"=>ch_b<=f;
     when others=>null;
     end case;
 end process change_frame;

scan_wt:scanwt_mdu port map(f_row,ch_f,gx,ch_out);

END beh;
```

上一程式之副程式

副程式 scanwt_mdu.vhd

加入掃描加權電路

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;
USE IEEE.std_logic_unsigned.all;
USE IEEE.std_logic_arith.all;

ENTITY scanwt_mdu IS
      PORT(
            fin: IN       STD_LOGIC;
            chin: in STD_LOGIC_VECTOR(0 to 34);
             gxx: OUT STD_LOGIC_VECTOR(0 to 6);
            chout: OUT STD_LOGIC_VECTOR(0 to 4)
            );
      END scanwt_mdu;

ARCHITECTURE beh OF scanwt_mdu IS
   type r_state is (rs0,rs1,rs2,rs3,rs4,rs5,rs6);
   signal pre_state,next_state:r_state;
   signal ch_wt:STD_LOGIC_VECTOR(0 TO 4);
   signal weight:STD_LOGIC_vector(3 downto 0);
   signal change_r:STD_LOGIC;

BEGIN

to_next_state:process(fin)
```

```
begin
   if fin='0' and fin'event then
      pre_state<=next_state;
   end if;
end process to_next_state;


scan_8_8:process
begin
    case pre_state is
    when rs0     =>if change_r='1' then
                      next_state<=rs1;
                   else
                      next_state<=rs0;
                   end if;
                   gxx<="1000000";
                   chout<=chin(0 to 4);
                   ch_wt<=chin(0 to 4);

     when rs1    =>if change_r='1' then
                      next_state<=rs2;
                   else
                      next_state<=rs1;
                   end if;
                   gxx<="0100000";
                   chout<=chin(5 to 9);
                   ch_wt<=chin(5 to 9);

     when rs2    =>if change_r='1' then
                      next_state<=rs3;
                   else
                      next_state<=rs2;
                   end if;
                   gxx<="0010000";
                   chout<=chin(10 to 14);
                   ch_wt<=chin(10 to 14);

     when rs3    =>if change_r='1' then
                      next_state<=rs4;
                   else
                      next_state<=rs3;
                   end if;
                   gxx<="0001000";
                   chout<=chin(15 to 19);
                   ch_wt<=chin(15 to 19);

     when rs4    =>if change_r='1' then
                      next_state<=rs5;
                   else
                      next_state<=rs4;
                   end if;
                   gxx<="0000100";
                   chout<=chin(20 to 24);
                   ch_wt<=chin(20 to 24);

     when rs5    =>if change_r='1' then
                      next_state<=rs6;
                   else
                      next_state<=rs5;
                   end if;
                   gxx<="0000010";
                   chout<=chin(25 to 29);
                   ch_wt<=chin(25 to 29);

     when rs6    =>if change_r='1' then
                      next_state<=rs0;
                   else
                      next_state<=rs6;
                   end if;
                   gxx<="0000001";
                   chout<=chin(30 to 34);
                   ch_wt<=chin(30 to 34);

    end case;
end process scan_8_8;

change_row:process(fin)
   variable row_stay:std_logic_vector(3 downto 0);
begin
  if fin='1' and fin'event then
     if row_stay>=weight   then
        row_stay:="0001";
        change_r<='1';
     else
        row_stay:=row_stay+1;
        change_r<='0';
     end if;
  end if;
end process change_row;

scan_weight:process(ch_wt)
   variable scan_time:STD_LOGIC_vector(3 downto 0);
begin

    scan_time:="0000";
    for j in 0 to 4 loop
       scan_time:=scan_time+not(ch_wt(j));
    end loop;
    weight<=scan_time;

end process scan_weight;

END beh;
```

掃瞄加權電路

將 row_stay 之值與每列亮燈數比較，如相等則到下一狀態繼續掃描下一列，如不等則重複掃描直到掃描次數等於亮燈數，才跳到下一狀態。

算出每一列中共有幾顆燈亮

LED 矩陣移位練習(3)(數字 0~9 A~F 由下往上移)

主程式 matrix_shift_b.vhd，副程式 atrix_scan_mdu_b.vhd

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;
USE IEEE.std_logic_unsigned.all;
USE IEEE.std_logic_arith.all;

ENTITY matrix_shift_b IS
        PORT(
            clk: IN     STD_LOGIC;
            shift: IN STD_LOGIC;
            GX: OUT STD_LOGIC_VECTOR(0 to 6);
            CH_OUT: OUT STD_LOGIC_VECTOR(0 to 4)
            );
        END matrix_shift_b;

ARCHITECTURE beh OF matrix_shift_b IS

   component matrix_scan_mdu_b
   PORT(
            fin: IN     STD_LOGIC;
            chin: in STD_LOGIC_VECTOR(0 to 34);
                change: in STD_LOGIC;
            gxx: OUT STD_LOGIC_VECTOR(0 to 6);
            chout: OUT STD_LOGIC_VECTOR(0 to 4));
   END component;

CONSTANT h0:STD_LOGIC_VECTOR(0 TO 34):=
  "0111110"
&"1000001"
&"1000001"
&"1000001"
&"0111110";
CONSTANT h1:STD_LOGIC_VECTOR(0 TO 34):=
  "0000001"
&"0100001"
&"1111111"
&"0000001"
&"0000001";
  CONSTANT h2:STD_LOGIC_VECTOR(0 TO 34):=
  "0100111"
&"1001001"
&"1001001"
&"1001001"
&"0110001";
  CONSTANT h3:STD_LOGIC_VECTOR(0 TO 34):=
  "0100010"
&"1001001"
&"1001001"
&"1001001"
&"0110110";
  CONSTANT h4:STD_LOGIC_VECTOR(0 TO 34):=
  "0000100"
&"0010100"
&"1000100"
&"1111111"
&"0000100";
  CONSTANT h5:STD_LOGIC_VECTOR(0 TO 34):=
  "1111010"
&"1001001"
&"1001001"
&"1001001"
&"1001110";
  CONSTANT h6:STD_LOGIC_VECTOR(0 TO 34):=
  "0111110"
&"1001001"
&"1001001"
&"1001001"
&"0100110";
  CONSTANT h7:STD_LOGIC_VECTOR(0 TO 34):=
  "1100000"
&"1000000"
&"1000000"
&"1000000"
&"1111111";
  CONSTANT h8:STD_LOGIC_VECTOR(0 TO 34):=
  "0110110"
&"1001001"
&"1001001"
&"1001001"
&"0110110";
  CONSTANT h9:STD_LOGIC_VECTOR(0 TO 34):=
  "0110010"
&"1001001"
&"1001001"
&"1001001"
&"0111110";
   CONSTANT a:STD_LOGIC_VECTOR(0 TO 34):=
  "0011111"
&"0101000"
&"1001000"
&"0101000"
&"0011111";
   CONSTANT b:STD_LOGIC_VECTOR(0 TO 34):=
  "1111111"
&"1001001"
&"1001001"
&"1001001"
&"0110110";
   CONSTANT c:STD_LOGIC_VECTOR(0 TO 34):=
  "0111110"
&"1000001"
&"1000001"
&"1000001"
&"0100010";
   CONSTANT d:STD_LOGIC_VECTOR(0 TO 34):=
  "1111111"
&"1000001"
&"1000001"
&"1000001"
&"0111110";
   CONSTANT e:STD_LOGIC_VECTOR(0 TO 34):=
  "1111111"
&"1001001"
&"1001001"
&"1001001"
&"1001001";
   CONSTANT f:sTD_LOGIC_VECTOR(0 TO 34):=
  "1111111"
&"1001000"
&"1001000"
&"1001000"
&"1001000";

   signal f_row,f_shift:std_logic;
   signal ch_f,ch_b:STD_LOGIC_VECTOR(0 TO 34);
   signal fram_index:STD_LOGIC_VECTOR(0 TO 3);

   BEGIN
freq:process(clk)
         variable modu:std_logic_vector(23 downto 0);
        begin
         if clk='1' and clk'event then
             modu:=modu-1;
```

```
    end if;
    f_row<=modu(8);
    f_shift<=modu(22);
  end process freq;

shifting:PROCESS(f_shift)
variable sh_index:STD_LOGIC_VECTOR(0 TO 2);
begin
  if f_shift='0' and f_shift'event then
        if shift='0' then
      if sh_index>=7 then
          if fram_index>=15 then
              fram_index<="0000";sh_index:="000";
          else
              fram_index<=fram_index+1;sh_index:="000" ;
          end if;
      else
              sh_index:=sh_index+1;
      end if;
  case sh_index is
      when "000"=>
      ch_f( 0 to    6)<=ch_f( 1 to    6)&"0";
      ch_f( 7 to    13)<=ch_f( 8 to    13)&"0";
      ch_f(14 to 20)<=ch_f(15 to 20)&"0";
      ch_f(21 to 27)<=ch_f(22 to 27)&"0";
      ch_f(28 to 34)<=ch_f(29 to 34)&"0";
      when "001"=>
      ch_f( 0 to    6)<=ch_f( 1 to    6)&ch_b(0);
      ch_f( 7 to    13)<=ch_f( 8 to    13)&ch_b(7);
      ch_f(14 to 20)<=ch_f(15 to 20)&ch_b(14);
      ch_f(21 to 27)<=ch_f(22 to 27)&ch_b(21);
      ch_f(28 to 34)<=ch_f(29 to 34)&ch_b(28);
      when "010"=>
      ch_f( 0 to    6)<=ch_f( 1 to    6)&ch_b(1);
      ch_f( 7 to    13)<=ch_f( 8 to    13)&ch_b(8);
      ch_f(14 to 20)<=ch_f(15 to 20)&ch_b(15);
      ch_f(21 to 27)<=ch_f(22 to 27)&ch_b(22);
      ch_f(28 to 34)<=ch_f(29 to 34)&ch_b(29);
      when "011"=>
      ch_f( 0 to    6)<=ch_f( 1 to    6)&ch_b(2);
      ch_f( 7 to    13)<=ch_f( 8 to    13)&ch_b(9);
      ch_f(14 to 20)<=ch_f(15 to 20)&ch_b(16);
      ch_f(21 to 27)<=ch_f(22 to 27)&ch_b(23);
      ch_f(28 to 34)<=ch_f(29 to 34)&ch_b(30);
      when "100"=>
      ch_f( 0 to    6)<=ch_f( 1 to    6)&ch_b(3);
      ch_f( 7 to    13)<=ch_f( 8 to    13)&ch_b(10);
      ch_f(14 to 20)<=ch_f(15 to 20)&ch_b(17);
      ch_f(21 to 27)<=ch_f(22 to 27)&ch_b(24);
      ch_f(28 to 34)<=ch_f(29 to 34)&ch_b(31);
      when "101"=>
      ch_f( 0 to    6)<=ch_f( 1 to    6)&ch_b(4);
      ch_f( 7 to    13)<=ch_f( 8 to    13)&ch_b(11);
      ch_f(14 to 20)<=ch_f(15 to 20)&ch_b(18);
      ch_f(21 to 27)<=ch_f(22 to 27)&ch_b(25);
      ch_f(28 to 34)<=ch_f(29 to 34)&ch_b(32);
      when "110"=>
      ch_f( 0 to    6)<=ch_f( 1 to    6)&ch_b(5);
      ch_f( 7 to    13)<=ch_f( 8 to    13)&ch_b(12);
      ch_f(14 to 20)<=ch_f(15 to 20)&ch_b(19);
      ch_f(21 to 27)<=ch_f(22 to 27)&ch_b(26);
      ch_f(28 to 34)<=ch_f(29 to 34)&ch_b(33);
          when "111"=>
      ch_f( 0 to    6)<=ch_f( 1 to    6)&ch_b(6);
      ch_f( 7 to    13)<=ch_f( 8 to    13)&ch_b(13);
      ch_f(14 to 20)<=ch_f(15 to 20)&ch_b(20);
      ch_f(21 to 27)<=ch_f(22 to 27)&ch_b(27);
      ch_f(28 to 34)<=ch_f(29 to 34)&ch_b(34);

when others=>null;
            end case;
        end if;
      end if;
  end process shifting;

  change_frame:process(fram_index)
    begin
      case fram_index is
      when "0000"=>ch_b<=h0;
      when "0001"=>ch_b<=h1;
      when "0010"=>ch_b<=h2;
      when "0011"=>ch_b<=h3;
      when "0100"=>ch_b<=h4;
      when "0101"=>ch_b<=h5;
      when "0110"=>ch_b<=h6;
      when "0111"=>ch_b<=h7;
      when "1000"=>ch_b<=h8;
      when "1001"=>ch_b<=h9;
      when "1010"=>ch_b<=a;
      when "1011"=>ch_b<=b;
      when "1100"=>ch_b<=c;
      when "1101"=>ch_b<=d;
      when "1110"=>ch_b<=e;
      when "1111"=>ch_b<=f;
      when others=>null;
      end case;
  end process change_frame;

scan_wt:matrix_scan_mdu_b port map(f_row,ch_f,shift,gx,ch_out);

END beh;
```

上一程式之副程式

副程式 matrix_scan_mdu_b.vhd

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;
USE IEEE.std_logic_unsigned.all;
USE IEEE.std_logic_arith.all;

ENTITY matrix_scan_mdu_b IS
      PORT(
            fin: IN     STD_LOGIC;
            chin: in STD_LOGIC_VECTOR(0 to 34);
            change: in STD_LOGIC;
            gxx: OUT STD_LOGIC_VECTOR(0 to 6);
            chout: OUT STD_LOGIC_VECTOR(0 to 4)
            );
      END matrix_scan_mdu_b;

ARCHITECTURE beh OF matrix_scan_mdu_b IS
   type r_state is (rs0,rs1,rs2,rs3,rs4);
   signal pre_state,next_state:r_state;
   signal change_r:STD_LOGIC;

BEGIN

to_next_state:process(fin)
begin
   if fin='0' and fin'event then
        pre_state<=next_state;
   end if;
end process to_next_state;


scan_5_7:process
begin
if change='0' then change_r<='1';
else change_r<='0';
end if;

case pre_state is
     when rs0      =>if change_r='1' then
                              next_state<=rs1;
                        else
                              next_state<=rs0;
                        end if;
                        gxx<=chin(0 to 6);
                        chout<="01111";
      when rs1      =>if change_r='1' then
                              next_state<=rs2;
                        else
                              next_state<=rs1;
                        end if;
                        gxx<=chin(7 to 13);
                        chout<="10111";
      when rs2      =>if change_r='1' then
                        next_state<=rs3;
                        else
                              next_state<=rs2;
                        end if;
                        gxx<=chin(14 to 20);
                        chout<="11011";
      when rs3      =>if change_r='1' then
                              next_state<=rs4;
                        else
                              next_state<=rs3;
                        end if;
                        gxx<=chin(21 to 27);
                        chout<="11101";
      when rs4      =>if change_r='1' then
                              next_state<=rs0;
                        else
                              next_state<=rs4;
                        end if;
                        gxx<=chin(28 to 34);
                        chout<="11110";
     end case;
end process scan_5_7;
END beh;
```