PWM 控制器硬體設計

1.實驗目的

利用程序控制器設計方法描述 PWM(Pulse Width Modulation)訊號產生器的動作行為，將 PWM 控制實際運用在類比電路上，練習以 PWM 訊號產器生控制改變發光二極體 LED 明亮程度。

2.實驗原理

PWM 為利用 on 與 off 兩種狀態在固定的週期中作切換，如圖 1 on 的時間與 off 的時間不同，達到 PWM 訊號產生的目的。

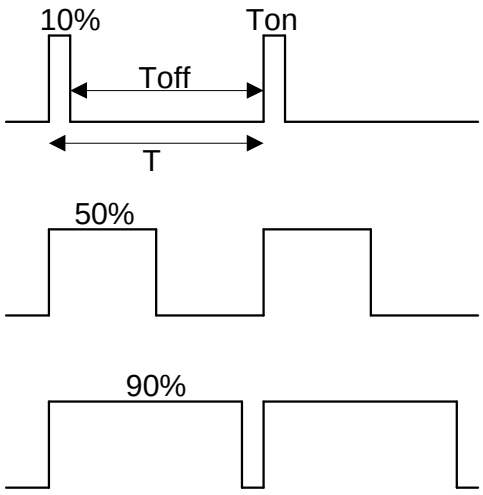

圖 1. 週期相同，Duty cycle 不同之 PWM 訊號。

例如以控制平均電壓為例：

平均電壓=Ton / (Ton+Toff) * 電源電壓

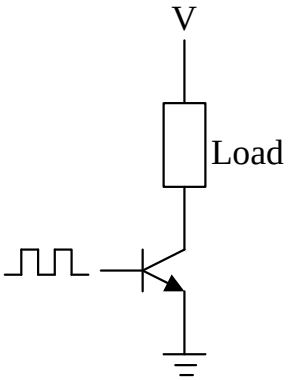

圖 2.平均電壓控制之負載驅動電路。

利用一計數器在計數過程中比較是否大於一 off value 若小於 off value 則 PWM 輸出 0 若大於 off value 則 PWM 輸出 1，如圖 3 所示。

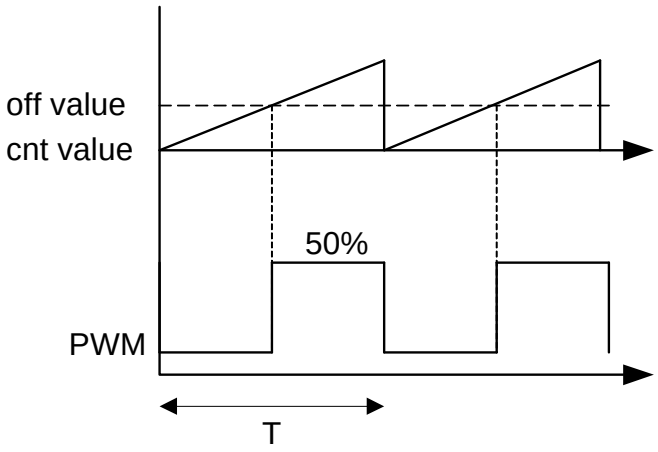

圖 3. PWM 訊號產生原理。

應用 GRAFCET 程序控制器設計方法描述 PWM 訊號產生器的行為如圖 4.。此 PWM 電路工作方式，首先設定計數資料 PWM 週期與 Duty cycle(PERIOD_I 、OFFT_I)，當狀態為 X0 時，PWM 的輸出為 0，CNT_REG=0，當狀態為 X1 時，PWM 的輸出為 0 並開始進行計數時直到計數器(CNT_REG)等於 OFFT_I，狀態由 X1 轉移至 X2，當狀態為 X2 時，PWM 輸出為 1，此時計數器(CNT_REG)繼續計數，直到計數器(CNT_REG)等於 PERIOD_REG，狀態由 X2 轉移至 X0，並清除計數器(CNT_REG)資料，重複 X0~X2 的動作。其合成電路如圖 5.，圖 6.為模擬結果。

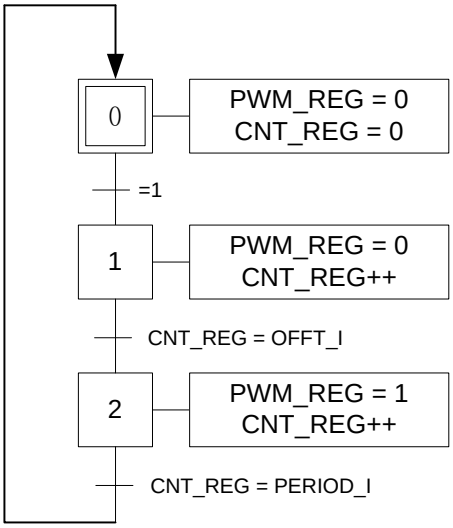

圖 4. PWM 訊號控制器設計。

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity pwm is
generic(CNT_WIDTH:integer:=7);
port(
	clk,rst:in std_logic;
	offt_i: in std_logic_vector(CNT_WIDTH downto 0);
	period_i: in std_logic_vector(CNT_WIDTH downto 0);
	cnt_out: out std_logic_vector(CNT_WIDTH downto 0);
	pwm_out: out std_logic
	);
end pwm;

ARCHITECTURE action OF pwm IS
	signal X0,X1,X2,pwm_reg: std_logic;
	signal cnt_reg:std_logic_vector(CNT_WIDTH downto 0);
begin
	process(clk,rst)
	begin
		if rst='0' then
		X0<='1';
		X1<='0';
		X2<='0';
	elsif clk'event and clk='1' then

			if X0='1'                              then X0<='0'; X1<='1';
		elsif X1='1' and cnt_reg=offt_i        then X1<='0'; X2<='1';
		elsif X2='1' and cnt_reg=period_i    then X2<='0'; X0<='1';
		end if;

			if X0='1' then pwm_reg<='0';cnt_reg<=(others=>'0');
		elsif X1='1' then pwm_reg<='0';cnt_reg<=cnt_reg+1;
		elsif X2='1' then pwm_reg<='1';cnt_reg<=cnt_reg+1;
		end if;
```

```
        end if;

        end process;

        cnt_out<=cnt_reg;
        pwm_out<=pwm_reg;

end action;
-----------------------------------------------
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity pwm_4 is
port(
        clk,rst:in std_logic;
        cnt_out: out std_logic_vector(3 downto 0);
        pwm_out: out std_logic
        );
end pwm_4;

ARCHITECTURE a OF pwm_4 IS

component pwm
generic(CNT_WIDTH:integer);
port(
        clk,rst:in std_logic;
        offt_i: in std_logic_vector(CNT_WIDTH downto 0);
        period_i: in std_logic_vector(CNT_WIDTH downto 0);
        cnt_out: out std_logic_vector(CNT_WIDTH downto 0);
        pwm_out: out std_logic
        );
end component;

begin
```

```
   U0:pwm generic map(3) port map(clk,rst,"0011","1000",cnt_out,pwm_out);
   --3 ="0011"
   --8 ="1000"
end a;
```

圖 5a. 應用程序控制器設計方法合成 PWM 之電路。

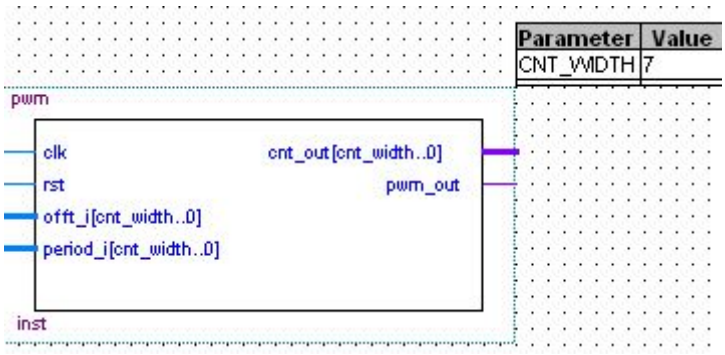

圖 5b.位元可參數化之 PWM 訊號產生器 I/O 訊號方塊圖。

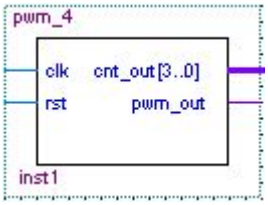

圖 5c. 4 位元 PWM 訊號產生器 I/O 訊號方塊圖。

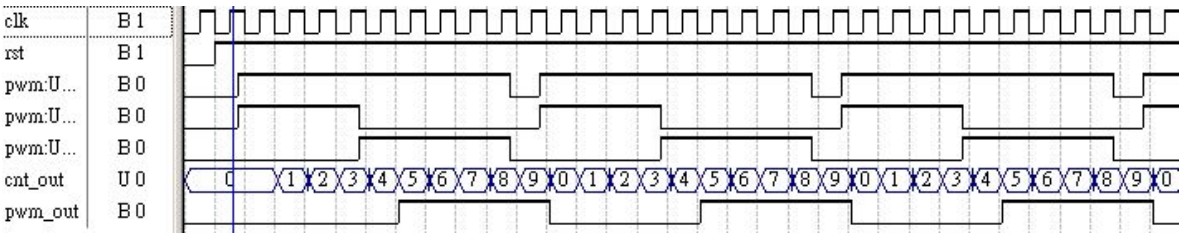

圖 6.當 Ton=5，T=10 時 之 PWM 產生器模擬。

3.立即演練

練習一、設計一可調整 LED 亮度之 PWM 控制電路。下表為接腳功能說明，請依照下列步驟完成此練習並下載至電路驗證。

表 1.接腳功能。

訊號名稱	功能
clk	50MHz
rst	系統重置/Button0
LED0	由點亮程度展示 PWM 訊號產生原理
Button1	調整 LED 亮度按鍵

步驟一、計算 PWM 之週期 T 與 DutyCycle(Ton)。

系統時脈是 50Mhz，若要產生週期為 20ms，Duty Cycle 為 10%的 PWM 訊號，則 Period_Time 與 Off_Time 的計數次數如下：

系統週期時間 1/50Mhz = 20ns

要達到 20ms 需要計數次數 20ms/20ns = 1,000,000 次

On_Time 的次數 1,000,000*10% = 100,000 次，即 Ton=2ms。

Off_Time 的次數 1,000,000- 100,000 = 900,000 次，即 Toff=18ms。

步驟二、.描述 PWM 訊號產生程序。

圖 7 為 PWM 產生器控制器程序行為描述，其中 PERIOD_I=1,000,000，OFFT_I 由圖 8 亮度調整控制器的 OFFT_I_REG 來設定。

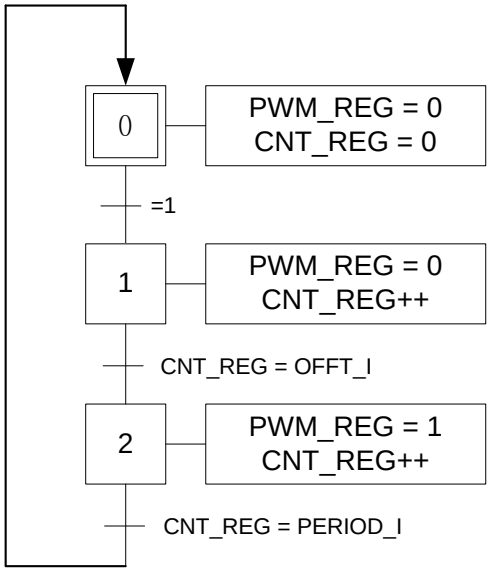

圖 7. PWM 訊號控制器設計。

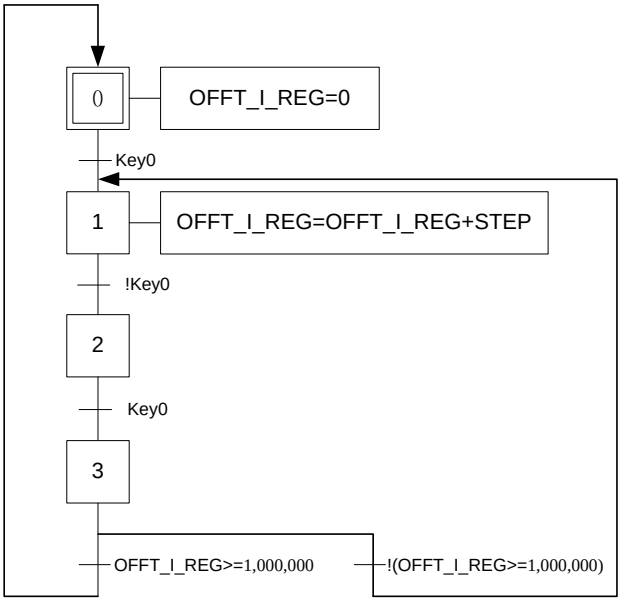

圖 8. Duty Cycle 調整控制器設計。

步驟三、電路合成。

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity pwm is
generic(CNT_WIDTH:integer:=7);
port(
      clk,rst:in std_logic;
      offt_i: in std_logic_vector(CNT_WIDTH downto 0);
      period_i: in std_logic_vector(CNT_WIDTH downto 0);
      cnt_out: out std_logic_vector(CNT_WIDTH downto 0);
      pwm_out: out std_logic
      );
end pwm;

ARCHITECTURE action OF pwm IS
      signal X0,X1,X2,pwm_reg: std_logic;
      signal cnt_reg:std_logic_vector(CNT_WIDTH downto 0);
begin
      process(clk,rst)
```

```
    begin
        if rst='0' then
        X0<='1';
        X1<='0';
        X2<='0';
    elsif clk'event and clk='1' then

            if X0='1'                           then X0<='0'; X1<='1';
        elsif X1='1' and cnt_reg=offt_i       then X1<='0'; X2<='1';
        elsif X2='1' and cnt_reg=period_i    then X2<='0'; X0<='1';
        end if;

            if X0='1' then pwm_reg<='0';cnt_reg<=(others=>'0');
        elsif X1='1' then pwm_reg<='0';cnt_reg<=cnt_reg+1;
        elsif X2='1' then pwm_reg<='1';cnt_reg<=cnt_reg+1;
        end if;

    end if;

    end process;

    cnt_out<=cnt_reg;
    pwm_out<=pwm_reg;

end action;
```

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity ledpwm is
generic(T:std_logic_vector(19 downto 0):="11110100001001000000");
--1,000,000="11110100001001000000"
port(
    clk,rst:in std_logic;
    key0     :in std_logic;
    cnt_out: out std_logic_vector(19 downto 0);
```

```
        pwm_out: out std_logic
        );
end ledpwm;

ARCHITECTURE a OF ledpwm IS

        signal X0,X1,X2,X3: std_logic;
        signal OFFT_I_REG:std_logic_vector(19 downto 0);

component pwm
generic(CNT_WIDTH:integer);
port(
        clk,rst:in std_logic;
        offt_i: in std_logic_vector(CNT_WIDTH downto 0);
        period_i: in std_logic_vector(CNT_WIDTH downto 0);
        cnt_out: out std_logic_vector(CNT_WIDTH downto 0);
        pwm_out: out std_logic
        );
end component;

begin

        process(clk,rst)
        begin
                if rst='0' then
                X0<='1';
                X1<='0';
                X2<='0';
                X3<='0';
        elsif clk'event and clk='1' then

                    if X0='1' and key0='0'                  then X0<='0'; X1<='1';
                elsif X1='1' and key0='1'           then X1<='0'; X2<='1';
                elsif X2='1' and key0='0'                   then X2<='0'; X3<='1';
                elsif X3='1' and OFFT_I_REG>=T          then X3<='0'; X0<='1';
                elsif X3='1' and not(OFFT_I_REG>=T)    then X3<='0'; X1<='1';
                end if;
```

```
            if X0='1' then OFFT_I_REG<=(others=>'0');
        elsif X1='1' then OFFT_I_REG<=OFFT_I_REG+1000;
        end if;

    end if;

    end process;

    U0:pwm generic map(19) port map(clk,rst,OFFT_I_REG,T,cnt_out,pwm_out);

end a;
```

圖 9. LED 亮度調整電路。

步驟四、 電路模擬與實驗板驗證。

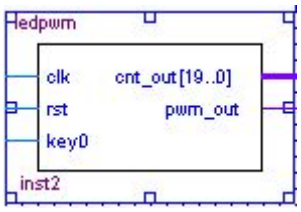

圖 10. LED 亮度調整電路訊號方塊圖。