目錄

一、　實驗小專題:DE0 FPGA應用

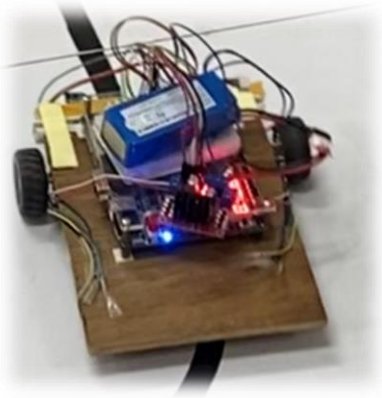
實驗架構:

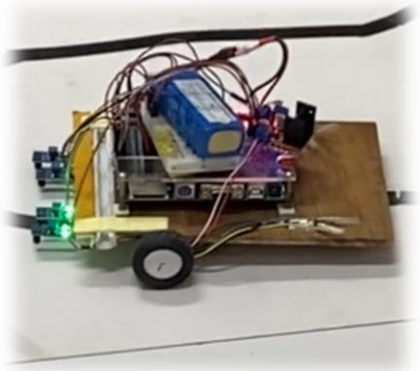
使用DE0 FPGA實驗板撰寫Verilog語法並且實作控制馬達與整合感測器並且對FPGA板進行整合與開發，並且使用直流馬達省點安靜、體積小的優點來製作。

設備說明:

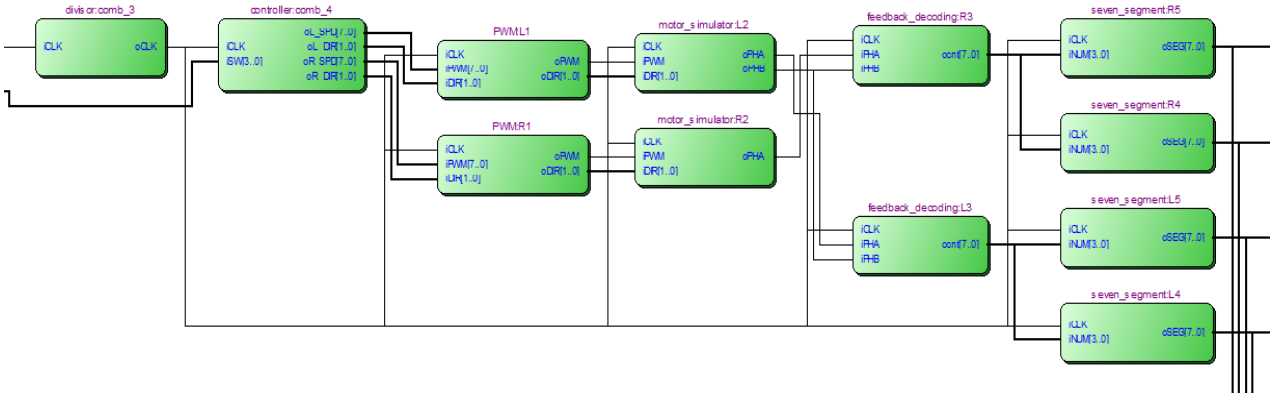
軟體:Quartus II 13.0 SP1、Nios II 13.0 SP1

硬體:電腦、DE0 FPGA 實驗板、L298N驅動器、直流馬達、杜邦線





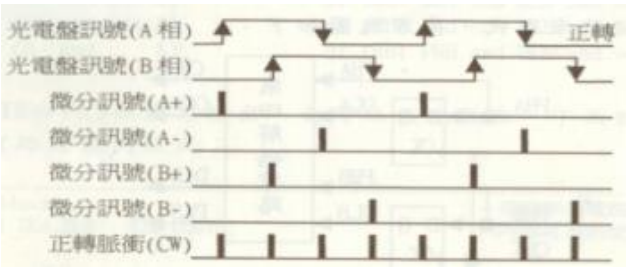
電路架構:

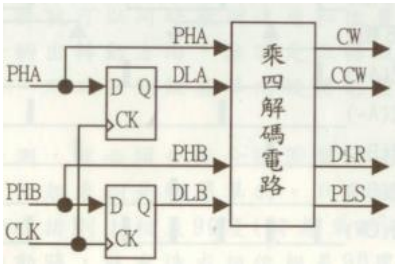


除頻器>訊號控制(放向速度)>PWM>馬達模擬>回授解碼電路>感測器

實驗困難點:

馬達編碼器為了量測方向，必須裝置兩組光電晶體，並讓兩組光電晶體在 排列上相差90度。圓盤轉動時，會造成相位相差90度的兩個方波訊號，而我們分別針對A/B兩相訊號，加以上緣微分和下緣微分而得到A+/A-/B+/B-的四個微分訊號，再將四個微分訊號整合成正轉脈衝CW訊號。這時光電盤每轉360度，就可以得到4個脈衝訊號輸出。





由於微分訊號都是由延遲訊號造成的，所以我們可以將微分訊號由延遲訊號來取代，經過D型暫存器後得到的DLA和DLB兩個延遲訊號接著的上緣和下緣微分訊號，就可以用這四個訊號來組合。

```
17  always@(posedge iCLK) begin
18      DLA <= iPHA;
19      DLB <= iPHB;
20  end
21
22  always@(posedge iCLK) begin
23
24  end
25  assign DIR = (iPHA &  DLA & ~iPHB)
26             | ( iPHA & ~DLA &  iPHB)
27             | (~iPHB &  DLB &  iPHA)
28             | ( iPHB & ~DLB & ~iPHA);
29
30  assign PLS = (~iPHA &  DLA)
31             | ( iPHA & ~DLA)
32             | (~iPHB &  DLB)
33             | ( iPHB & ~DLB);
34
35  assign oDIR = DIR;      //direction
36  assign oPLS = PLS;      //speed
```

feedback_decoding

二、　課堂專題：嵌入式系統計數器

設計一個可上下數的計數器，頻率約為 5Hz，以七段顯示器顯示四位數的上下計數。

➢若 SW[9]數值為 1，則七段顯示器以十進制進行計數；若 SW[9]數值為 0，則七段顯示器以十六進制進行計數。

➢若 SW[8]數值為 1，則七段顯示器進行上數；若 SW[8]數值為 0，則七段顯示器進行下數。

➢若SW[7]數值為 1，則進行計數；若 SW[7]數值為 0，則停止計數。

➢若SW[6]數值為 1，則七段顯示器顯示；若 SW[6]數值為 0，則七段顯示器不顯示。

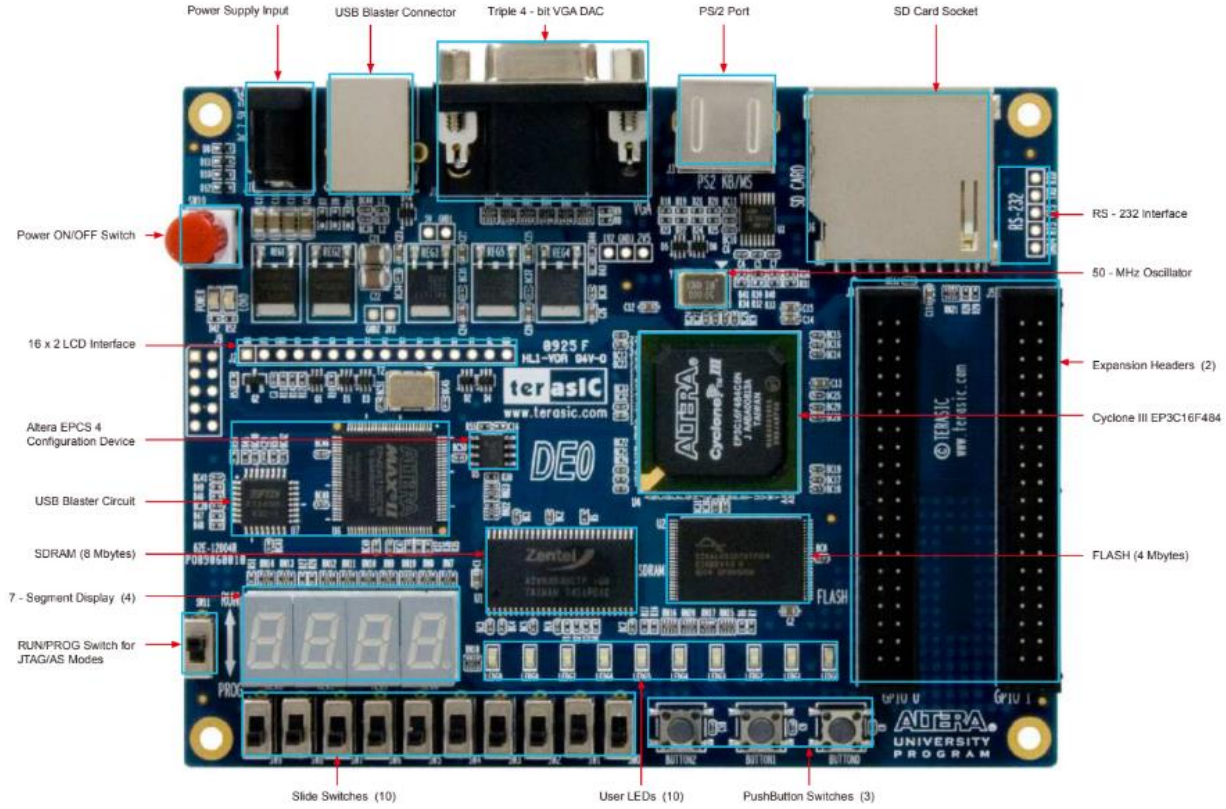
設備說明

軟體:Quartus II 13.0 SP1、Nios II 13.0 SP1

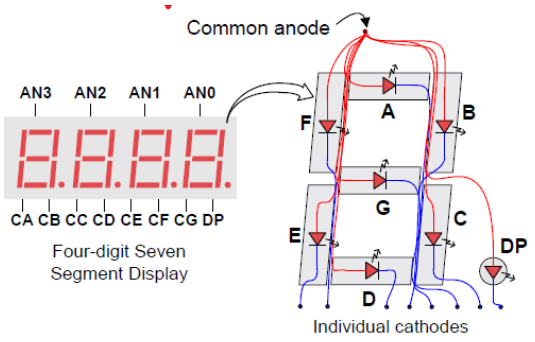
硬體:電腦、DE0 FPGA 實驗板

結果說明

如 Fig 1 程式碼 01、Fig 2 程式碼02、Fig 3 程式碼03所示。



指撥開關由左至右分別為
sw9sw8sw7sw6sw5sw4sw3sw2sw1sw0



字型	dp	g	f	e	d	c	b	a
0	1	1	0	0	0	0	0	0
1	1	1	1	1	1	0	0	1
2	1	0	1	0	0	1	0	0
3	1	0	1	1	0	0	0	0
4	1	0	0	1	1	0	0	1
5	1	0	0	1	0	0	1	0
6	1	0	0	0	0	0	1	0
7	1	1	1	1	1	0	0	0
8	1	0	0	0	0	0	0	0
9	1	0	0	1	0	0	0	0

由於此板是共陽極所以驅動共陽極七段顯示器 C語言16進位依序是0xC0、0xF9、0xA4、0xB0、0x99、0x92、0x82、0xF8、0x80、0x90 如Fig 3 程式碼 所示。

```c
#include "alt_types.h"
#include "sys/alt_irq.h"
#include "system.h"
#include <stdio.h>
#include <unistd.h>
#include <io.h>

static void sevenseg_set_hex(int hex);
static void sevenseg_set_dex(int dex);

int count=0;

 int main()
{
     // Set timer for 1 second
    IOWR(TIMER_BASE, 2, (short)(TIMER_FREQ/5& 0x0000ffff));
    IOWR(TIMER_BASE, 3, (short)((TIMER_FREQ/5>> 16) & 0x0000ffff));
    // Set timer running, no looping, no interrupts
    Timer_Interrupt_Enable();
     // Poll timer forever, print once per second
    while(1)
    {
        IOWR(TIMER_BASE, 1, 0x07);
    }

    return 0;

}

void TimerISR(void *context,alt_u32 id)
{
    printf("TimerISR\n");

    if(IORD(TIMER_BASE, 0)& 0x01)
        {
        if(IORD(SWITCHES_BASE,0)&0x80)//開始計數
            {
            if(IORD(SWITCHES_BASE,0)&0x100)//上數
                {
                if(count>=65535)
                    count=0;
                else
                    count++;
                }
```

Fig 1　程式碼 01

```
            else //下數
                {
                if(count<=0)
                    count=65535;
                else
                    count--;
                }
            }
        else //停止計數
            count+=0;
        if(IORD(SWITCHES_BASE,0)&0x40)//顯示七段
            {
            if(IORD(SWITCHES_BASE,0)&0x200)//10進位
                sevenseg_set_dex(count);
            else //16進位
                sevenseg_set_hex(count);
            }
        else//停止顯示
            IOWR(SEG7_BASE,0,0xFFFFFFFF);
        }

    IOWR(TIMER_BASE,0,0);

}

void Timer_Interrupt_Enable(void)
{
    int reg;

    alt_irq_register(TIMER_IRQ, 0, TimerISR);
    reg= IORD(TIMER_BASE, 1);
    reg= reg| 0x01;
    IOWR(TIMER_BASE, 1, reg);

    printf("\n\nTimer interrupt enabled.\n\n");

}
```

Fig 2　程式碼 02

```
static void sevenseg_set_hex(int hex)
{
    int hex1,hex2,hex3,hex4;

    hex1 = hex%16;
    hex2 = hex/16%16;
    hex3 = hex/256%16;
    hex4 = hex / 4096 % 16;

    static alt_u8 segments[16] = {
        0xC0, 0xF9, 0xA4, 0xB0, 0x99, 0x92, 0x82, 0xD8, 0x80, 0x90,
        0x88, 0x83, 0xC6, 0xA1, 0x86, 0x8E};

    alt_u32 data = segments[hex1]+(segments[hex2]<<8)+(segments[hex3]<<16)+(segments[hex4]<<24);
    IOWR(SEG7_BASE,0,data);

}

 static void sevenseg_set_dex(int dex)
 {
    int dex1,dex2,dex3,dex4;

    dex1 = dex%10;
    dex2 = dex/10%10;
    dex3 = dex/100%10;
    dex4 = dex / 1000 % 10;

    static alt_u8 segments[10] = {
        0xC0, 0xF9, 0xA4, 0xB0, 0x99, 0x92, 0x82, 0xD8, 0x80, 0x90};

    alt_u32 data = segments[dex1]+(segments[dex2]<<8)+(segments[dex3]<<16)+(segments[dex4]<<24);
    IOWR(SEG7_BASE,0,data);

 }
```

Fig 3　程式碼 03

程式設計說明

主程式內前四行為 timer 設定，應題目要求設定頻率為 5 Hz，所以TIMER_FREQ 要除五。如 Fig 1 程式碼 01 所示。

主程式第五行為呼叫中斷副函式。如 Fig 1 程式碼01 所示。

呼叫中斷副函式 Timer_Interrupt_Enable 後，第三行 alt_irq_register 代表不回傳任何數值，並且執行TimerISR 副函式。如 Fig 2 程式碼 02 所示。

副函式TimerISR 裡，第三行判斷計數時間是否結束，計數結束後，執行條件式內程式。如 Fig 1 程式碼 01 所示。

副函式TimerISR 裡，第五行為開始計數的條件式，當 Switch07 開啟，條件式裡的程式被編譯；Switch07 關閉時，else 裡程式被編譯，count 停止計數。如Fig 1 程式碼 01、Fig 2 程式碼 02。

副函式TimerISR 裡，Switch07 的條件式裡為一個上下數的條件判斷，當 Switch08 開啟時，count 加一、然後當超過 65535(十六進位為FFFF)時，count 重製從零開始上加。如 Fig 1 程式碼01 所示。當 Switch08 關閉時，執行 else 裡的程式 ---count 減一、然後當少於0 時，count 從65535(十六進位為FFFF)開始減少。如Fig 2 程式碼 02 所示。

在 Switch07 條件式結束後，又一個條件判斷為判斷是否顯示七段顯示器。當 Switch06 開啟，執行條件式裡的程式 ---顯示七段；Switch06 關閉時，執行else 裡的程式，對七段顯示器輸入數值零使七段顯示器不亮燈。如Fig 2 程式碼 02 所示。

Switch06 的條件式為七段顯示器是以十進位還是以十六進位顯示，當 Switch09 開啟時，執行條件式裡的程式 ---呼叫十進位的副函式；Switch09 關閉時，執行 else 裡的程式 --- 呼叫十六進位的副函式。如 Fig 2 程式碼 02 所示。

剩下的為副函式的部分，以十六進位為例。先宣告四個變數及一個陣列，其四個變數為四位數七段顯示器的各一個位數。而陣列裡則儲存著七段顯示器。

從 0 到F 的數值。最後將 data 的值輸出到 SEG7_BASE，就完成了十六進位的顯示。十位元同理。如 Fig 3 程式碼 03 所示。