Lab 2　輸入輸出控制與軟硬體共同設計

實驗主題一：SW 控制 LED 明滅

實驗主題二：NIOS II 結合 PWM 電路控制 LED 亮度變化

銘傳大學電腦與通訊工程學系

陳慶逸

壹、背景知識

1.為何使用 Nios II 處理器？

當今的嵌入式設計工程師面臨很棘手的挑戰：尋找一款能夠實現特性、成本、性能和生命週期完美組合的處理器。Nios II 系列 32 位元 RISC 嵌入式處理器具有超過 200 DMIP 的性能，在 FPGA 中實現成本只有 35 美分。由於處理器是軟核形式，具有很大的靈活性，您可以在多種系統設置組合中進行選擇，達到性能、特性和成本目標。Nios II 處理器具有以下特點：

(1)自定標準定制處理器

若採用 Nios II 處理器，將不會局限於預先製造的處理器技術，可根據自己的標準定制處理器，按照需要選擇合適的外部裝置、記憶體和介面。

(2)配置系統性能

Nios II 設計人員必須能夠更改其設計，加入多個 Nios II CPU、自定指令集、硬體加速器，以達到新的性能目標。採用 Nios II 處理器，可以通過 Avalon 匯流排來調整系統性能，實現大吞吐量應用。

(3)低成本實現

在選擇處理器時，為了實現需要的功能，可能要購買比實際所需數量多的處理器，也可能為了節省成本，而不得不購買比實際需要數量少的處理器。低成本、可定制 Nios II 處理器能夠幫助您解決這一難題。採用 Nios II 處理器，可以根據需要，設置功能，在價格低至 35 美分的 CycloneTM II FPGA 等低成本 Altera® 器件中實施。在單個 FPGA 中實現處理器、外部裝置、記憶體和 I/O 介面，可以降低系統總體成本。

(4)產品生存週期管理

為實現一個成功的產品，您需要將其儘快推向市場，增強其功能特性以延長使用時間，避免出現處理器逐漸過時。您可以在短時間內，將 Nios II 嵌入式處理器由最初概念設想轉為系統實現。這種基於 Nios II 處理器的系統具有永久免版稅設計許可，完全經得起時間考驗。此外，由於在 FPGA 中實現軟核處理器，因此可以方便實現現場硬體和軟體升級，產品能夠符合最新的規範、具備最新特性。

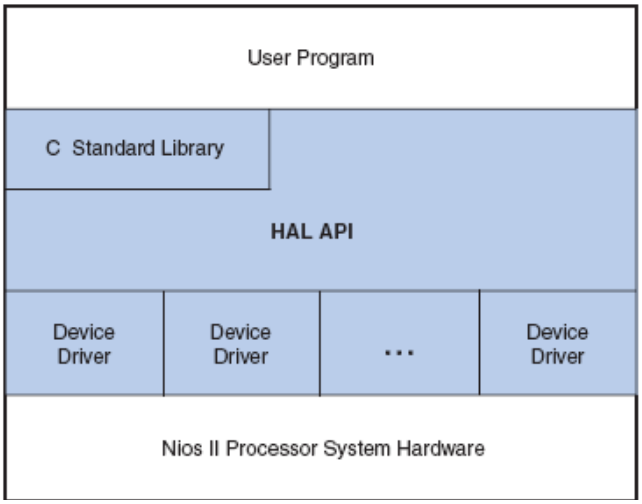
2.Nios II 處理器有三種的規格：

Nios II 處理器系列包括三種內核——快速（Nios II/f）、標準（Nios II/s）和經濟型（Nios II/e），每一型號都針對價格和性能範圍進行了最佳化（下表）。所有這些內核共用 32 位元指令集體系，與二進位碼 100%相容。使用 Altera 的 Quartus® II 設計軟體的 SOPC Builder 工具，可以在系統中輕鬆加入 Nios II 處理器。

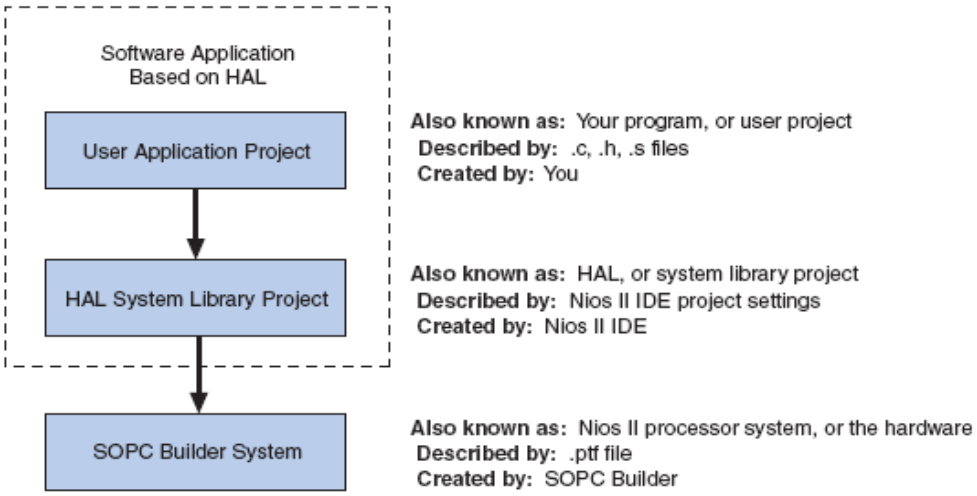
三種處理器內核			
特性	Nios II /f (快速)	Nios II /s (標準)	Nios II /e (經濟)
說明	針對最佳性能最佳化	比第一代 Nios CPU 最快的型號還要快，而體積比最小的還要小	針對最少的邏輯資源佔用最佳化
管線化	6 級	5 級	無
乘法器	1 週期	3 週期	軟體實現
分歧判斷	動態	靜態	無
指令快取	可設定	可可設定	無
資料快取	可設定	無	無
自定指令集	256 個	256 個	256 個

(資料來源：和春資工系線上教學實驗室;
http://www.altera.com.cn/literature/br/br_NiosII_SC.pdf)

3. The Layers of a HAL-Based System



4. The Nios II IDE Project Structure



5. NIOS II IDE 語法

#include <io.h>

- IOWR(Base, Regnum, Wdata)

  Base：周邊元件基底位置

  Regnum：暫存器號碼

  Wdata：寫入資料

  Example：IOWR(LED_PIO_BASE, 2, 0x10)

- Rdata = IORD(Base, Regnum)

  Base：周邊元件基底位置

  Regnum：暫存器號碼

  Rdata：從 IO 端讀取回來的資料

參考*_syslib[nios2]/ Device Drivers [sopc_builder]/

altera_avalon_pio/ inc/ altera_avalon_pio_regs.h

(1) IORD_ALTERA_AVALON_PIO_DATA(base)　等同於 IORD(base, 0)

(2) IORD_ALTERA_AVALON_PIO_DATA(BUTTON_PIO_BASE)等同於 IORD(BUTTON_PIO_BASE, 0)

(3) IOWR_ALTERA_AVALON_PIO_DATA(base,data)　等同於 IOWR(base, 0, data)

(4) IOWR_ALTERA_AVALON_PIO_DATA(LED_PIO_BASE,);　等同於 IOWR(LED_PIO_BASE, 0, led)

output 丟值

(1) IOWR_ALTERA_AVALON_PIO_DATA(base,data)等同於 IOWR(base, 0, data)

(2) IOWR_ALTERA_AVALON_PIO_DATA(LED_PIO_BASE, led);　等同於 IOWR(LED_PIO_BASE, 0, led)

offset	register	說明
0	data	將值丟到這裡
1	direction	只有雙向 IO 才會用到

input 讀值

(1)IORD_ALTERA_AVALON_PIO_DATA(base)　等同於 IORD(base, 0)

(2)Button=IORD_ALTERA_AVALON_PIO_DATA(BUTTON_PIO_BASE)　等同於 Button=IORD(BUTTON_PIO_BASE, 0)

offset	register	說明
0	data	從這裡讀取

1	direction	只有雙向 IO 才會用到

6. 變數型態宣告

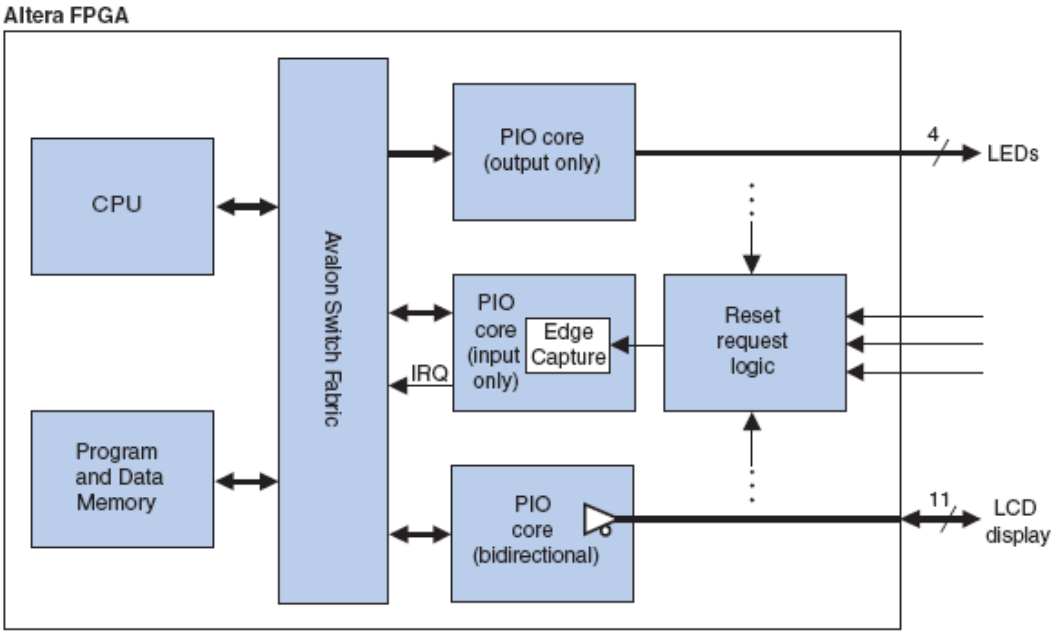
Table 4–1. The HAL Type Definitions

Type	Meaning
alt_8	Signed 8-bit integer.
alt_u8	Unsigned 8-bit integer.
alt_16	Signed 16-bit integer.
alt_u16	Unsigned 16-bit integer.
alt_32	Signed 32-bit integer.
alt_u32	Unsigned 32-bit integer.
alt_64	Signed 64-bit integer.
alt_u64	Unsigned 64-bit integer.

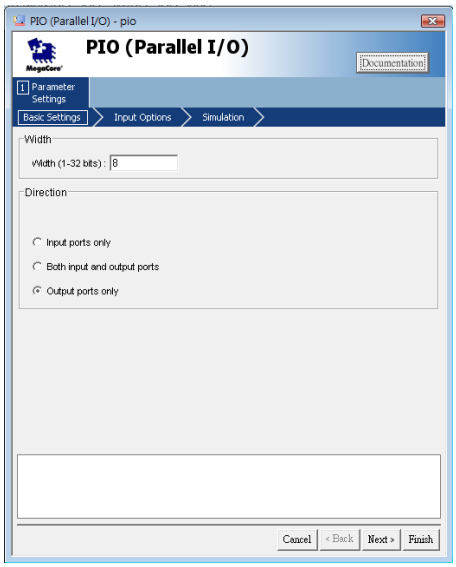
Table 4–2. GNU Toolchain Data Widths

Type	Meaning
char	8 bits.
short	16 bits.
long	32 bits.
int	32 bits.

7. An Example System Using Multiple PIO Cores



PIO Introduce:



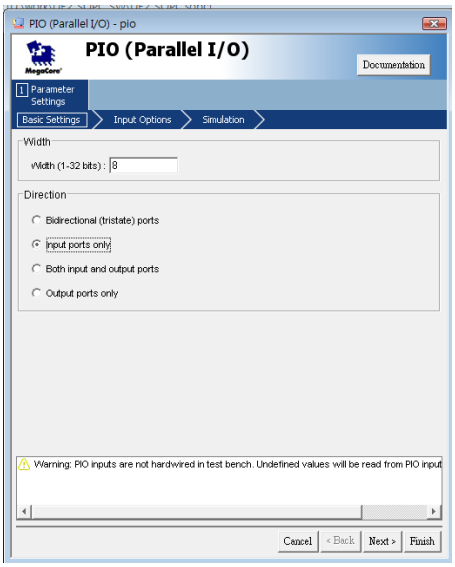


Table 11–2. Register Map for the PIO Core

Offset	Register Name		R/W	(n-1)	...	2	1	0
0	`data`	read access	R	Data value currently on PIO inputs				
		write access	W	New value to drive on PIO outputs				
1	`direction` *(1)*		R/W	Individual direction control for each I/O port. A value of 0 sets the direction to input; 1 sets the direction to output.				
2	`interruptmask` *(1)*		R/W	IRQ enable/disable for each input port. Setting a bit to 1 enables interrupts for the corresponding port.				
3	`edgecapture` *(1), (2)*		R/W	Edge detection for each input port.				

貳、實驗內容

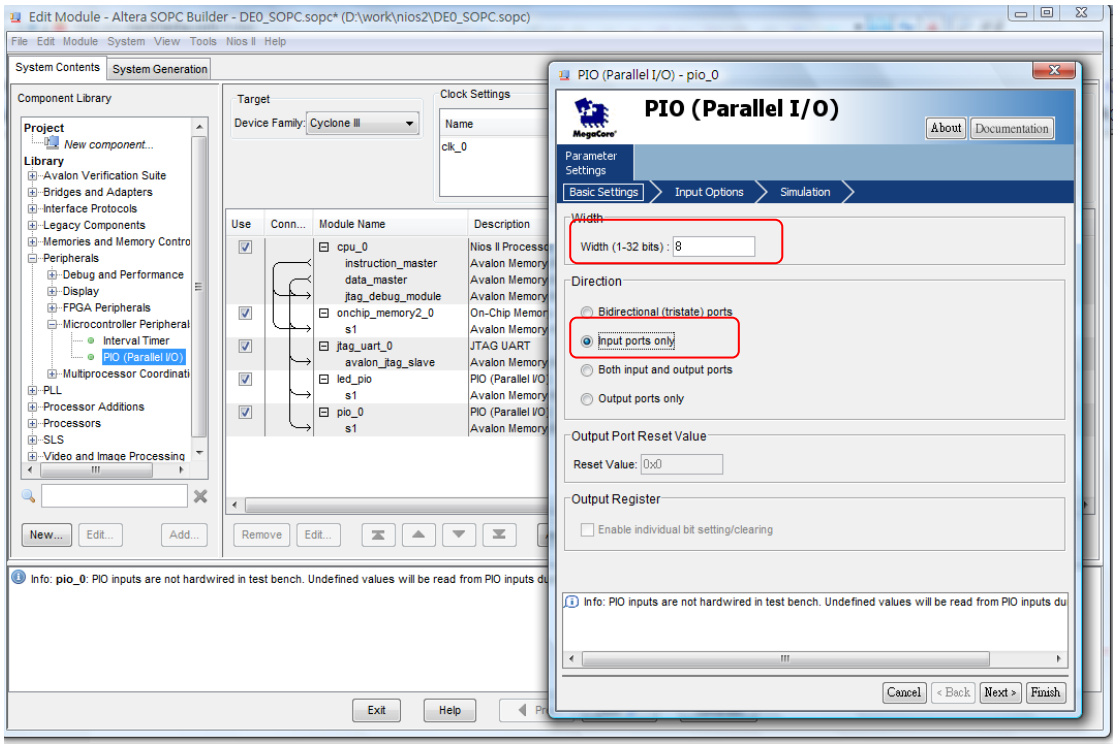
實驗一

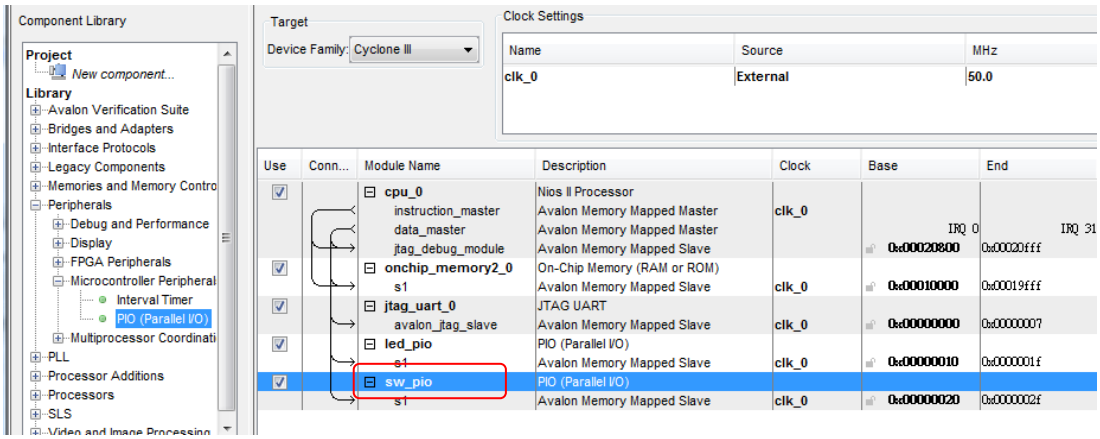
一、硬體設計(Nios II Hardware Development)

1. Create New Project

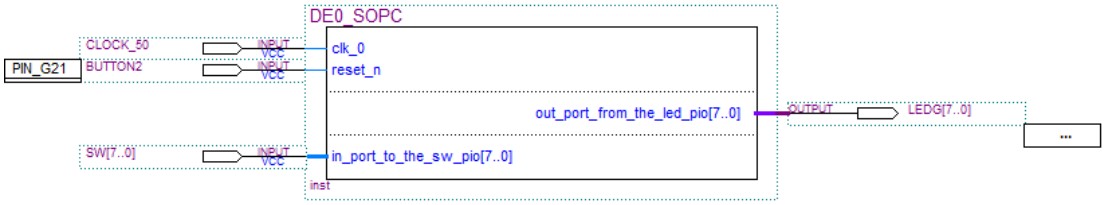
2. Using the SOPC Builder to Generate the Nios II System..

Add cpu, onchip_mem (40K bytes), jtag_uart, sysid, PIO (sw_pio, input, 8bit), PIO (led_pio, output, 8 bit)



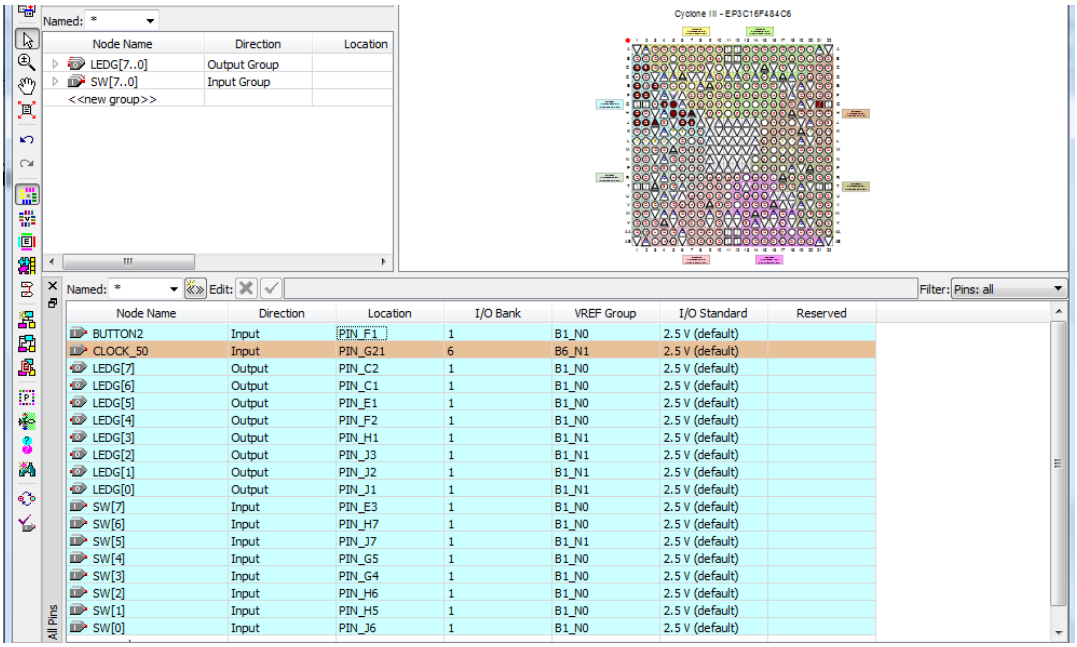


3. Integration of the Nios II System into the Quartus II Project



4. Pin Assignments & Download

輸入	腳位	輸出	腳位
SW[7]	PIN_E3 (SW[7])	LEDG[7]	PIN_C2 (LEDG[7])
SW[6]	PIN_H7 (SW[6])	LEDG[6]	PIN_C1 (LEDG[6])
SW[5]	PIN_J7 (SW[5])	LEDG[5]	PIN_E1 (LEDG[5])
SW[4]	PIN_G5 (SW[4])	LEDG[4]	PIN_F2 (LEDG[4])
SW[3]	PIN_G4 (SW[3])	LEDG[3]	PIN_H1 (LEDG[3])
SW[2]	PIN_H6 (SW[2])	LEDG[2]	PIN_J3 (LEDG[2])
SW[1]	PIN_H5 (SW[1])	LEDG[1]	PIN_J2 (LEDG[1])
SW[0]	PIN_J6 (SW[0])	LEDG[0]	PIN_J1 (LEDG[0])
Clock_50	PIN_G21		
BUTTON2	PIN_F1		


Named: *
Node Name
Direction
Location
LEDG[7..0]
Output Group
SW[7..0]
Input Group
<<new group>>
Cyclone III - EP3C16F484C6
Named: *
Edit:
Filter: Pins: all
Node Name
Direction
Location
I/O Bank
VREF Group
I/O Standard
Reserved
BUTTON2 Input PIN_F1 1 B1_N0 2.5 V (default)
CLOCK_50 Input PIN_G21 6 B6_N1 2.5 V (default)
LEDG[7] Output PIN_C2 1 B1_N0 2.5 V (default)
LEDG[6] Output PIN_C1 1 B1_N0 2.5 V (default)
LEDG[5] Output PIN_E1 1 B1_N0 2.5 V (default)
LEDG[4] Output PIN_F2 1 B1_N0 2.5 V (default)
LEDG[3] Output PIN_H1 1 B1_N1 2.5 V (default)
LEDG[2] Output PIN_J3 1 B1_N1 2.5 V (default)
LEDG[1] Output PIN_J2 1 B1_N1 2.5 V (default)
LEDG[0] Output PIN_J1 1 B1_N1 2.5 V (default)
SW[7] Input PIN_E3 1 B1_N0 2.5 V (default)
SW[6] Input PIN_H7 1 B1_N0 2.5 V (default)
SW[5] Input PIN_J7 1 B1_N1 2.5 V (default)
SW[4] Input PIN_G5 1 B1_N0 2.5 V (default)
SW[3] Input PIN_G4 1 B1_N0 2.5 V (default)
SW[2] Input PIN_H6 1 B1_N0 2.5 V (default)
SW[1] Input PIN_H5 1 B1_N0 2.5 V (default)
SW[0] Input PIN_J6 1 B1_N0 2.5 V (default)
All Pins

二、軟體設計

利用 switch 控制 LED 輸出

```
#include <io.h>
#include "system.h"

int main()
{ char sw_input;

  while(1)
 { sw_input=IORD(SW_PIO_BASE,0);
   IOWR(LED_PIO_BASE,0,sw_input); }
  return 0;
}
```

2. Build Project

3. Run As > Nios II Hardware

隨堂練習一：

1. 利用 if 敘述來判斷使用者從 DE0 實驗板指撥開關輸入的數值。當數值>=60 時，實驗板之 LED 為 11110000；當數值<60 時，實驗板之 LED 為 00001111。

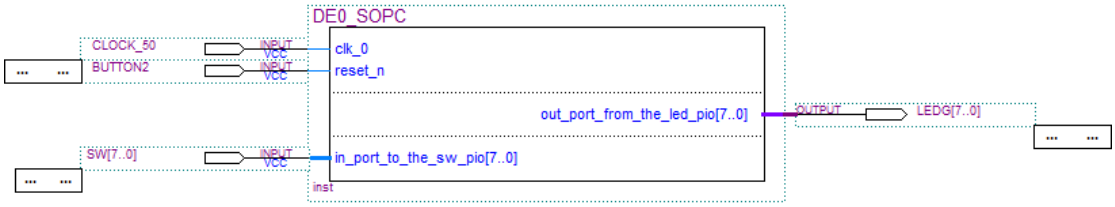
```
#include <io.h>
#include "system.h"

int main()
{ int ??????????;

  while(1)
 {
      ??????????
 }
  return 0;
}
```

實驗二、結合 NIOS II 與 PWM 硬體電路控制 LED 亮度變化

功能說明: 利用 SW[7..0]控制 LEDR[7..0]輸出，並決定 LEDR[17]的亮度變化。

Integration of the Nios II System and PWM symbol into the Quartus II Project

(1) Nios II System

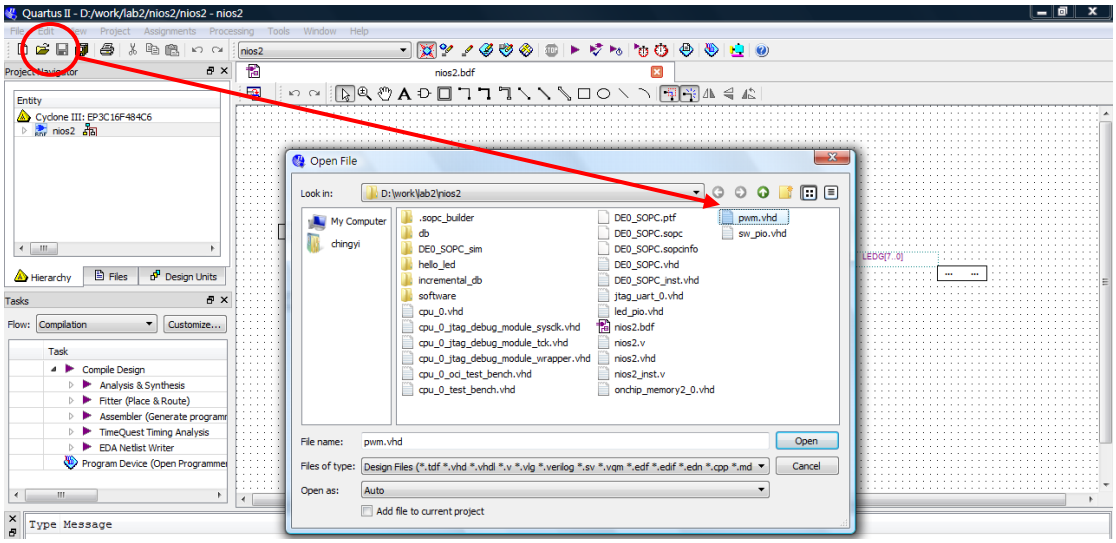


(2) 從 moodle 下載 pwm.vhd, 並存在 project 所在資料夾中

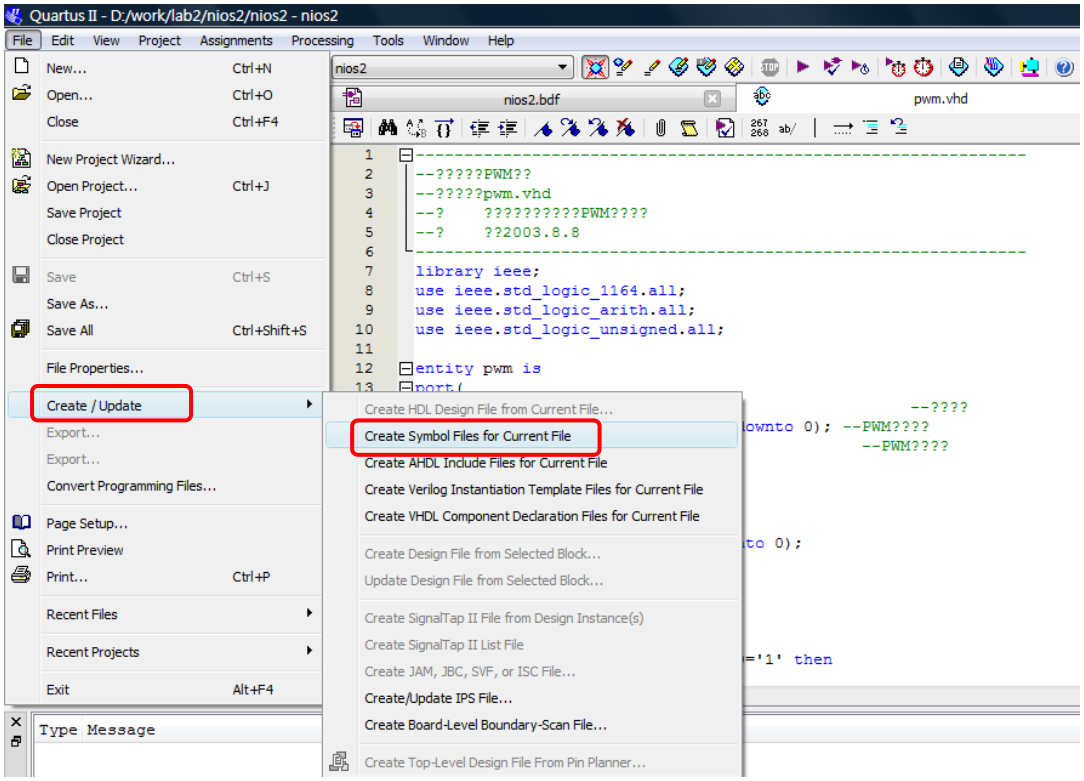
```
--------------------------------------------------------------------
--實驗名稱：PWM 實習
--檔案名稱：pwm.vhd
--功　　能：以指撥開關來調整 PWM 訊號輸出
--日　　期：2003.8.8
--------------------------------------------------------------------
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity pwm is
port(
        CLOCK_50 : in    std_logic;                              --系統頻率
        SW: in    std_logic_vector(7 downto 0); --PWM 控制訊號
        LED : out std_logic                                    --PWM 訊號輸出
    );
end pwm;

architecture a of pwm is
  signal B:std_logic_vector(7 downto 0);
begin

  ----- 下數計數器 -----
  process(CLOCK_50)
  begin
    if CLOCK_50'event and CLOCK_50='1' then
      B <= B-1;
    end if;
  end process;
```

```
  --比較器
  LED<='1' when SW > B else '0';

end a;
```

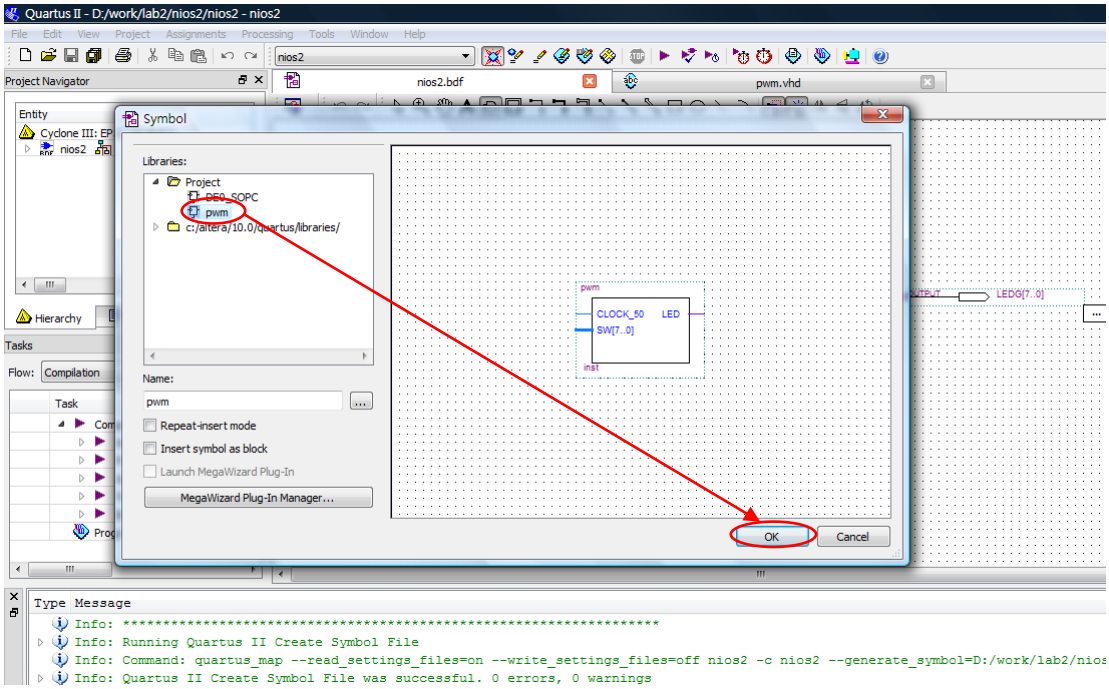
(3) open pwm.vhd



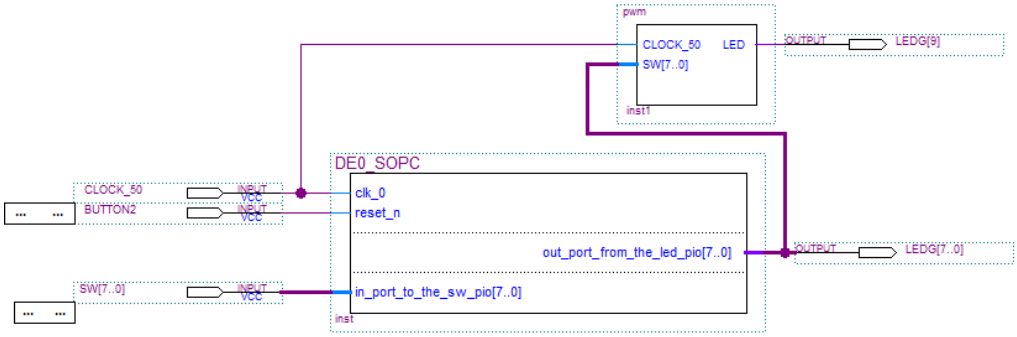
(4) File> Creat/update> Create Symbol Files for Current File



(5) 回到 top_design, 插入剛建立的 pwm symbol (滑鼠雙擊)



(6) 加上 output symbol (命名為 LEDG[9]), 並完成連線。



(7) Pin Assignments

輸入	腳位	輸出	腳位
SW[7]	PIN_E3 (SW[7])	LEDG[7]	PIN_C2
SW[6]	PIN_H7 (SW[6])	LEDG[6]	PIN_C1
SW[5]	PIN_J7 (SW[5])	LEDG[5]	PIN_E1
SW[4]	PIN_G5 (SW[4])	LEDG[4]	PIN_F2
SW[3]	PIN_G4 (SW[3])	LEDG[3]	PIN_H1
SW[2]	PIN_H6 (SW[2])	LEDG[2]	PIN_J3
SW[1]	PIN_H5 (SW[1])	LEDG[1]	PIN_J2
SW[0]	PIN_J6 (SW[0])	LEDG[0]	PIN_J1
Clock_50	PIN_G21		
BUTTON2	PIN_F1	LEDG[9]	PIN_B1

(8) download design。

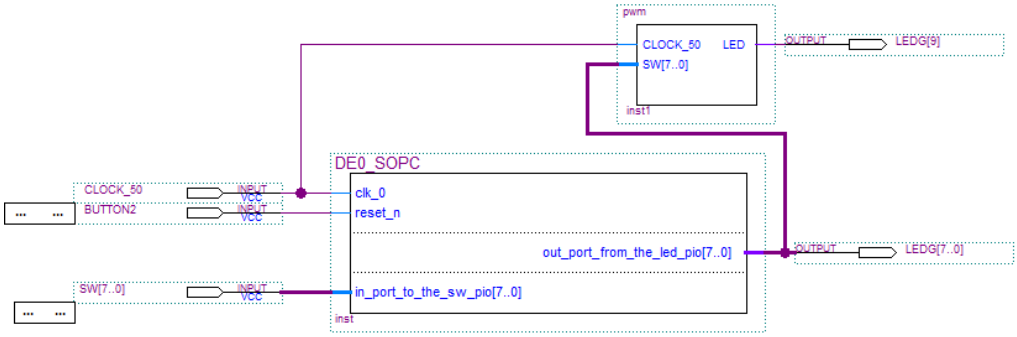
二、軟體設計

利用 switch 控制 LED 輸出

```
#include <io.h>
#include "system.h"

int main()
{ char sw_input;

  while(1)
 { sw_input=IORD(SW_PIO_BASE,0);
   IOWR(LED_PIO_BASE,0,sw_input); }
  return 0;
}
```

隨堂練習二：

1. 設計一個以 0.2s 計數的 00-FF 上數計數器，並將結果顯示在 LED[7..0]上。我們可以藉以觀察 pwm 的輸出是否由暗而亮變化。




```
#include <stdio.h>
#include <io.h>
#include "system.h"
#include "unistd.h"

int main()
{
    while(1)
    {
      ??????????     }
 return 0;
}
```

2. 設計一個以 0.02s 計數的上、下數自動切換之計數器。我們可以藉以觀察 pwm 的輸出是否由亮而暗、由暗而亮反覆動作。

```
#include <io.h>
#include "unistd.h"
#include "system.h"



int main()
{
  char dir = 0;
  int count;
```

```
  while (1)
  {

    if (dir)
    {
      for (??????????????????)
      {
         ?????????;
         ?????????;
         dir = 0;
      }
    }
    else
    {
      for (??????????????????)
      {
          ?????????;
         dir = 1;

      }
    }
 }

    return 0;
}
```