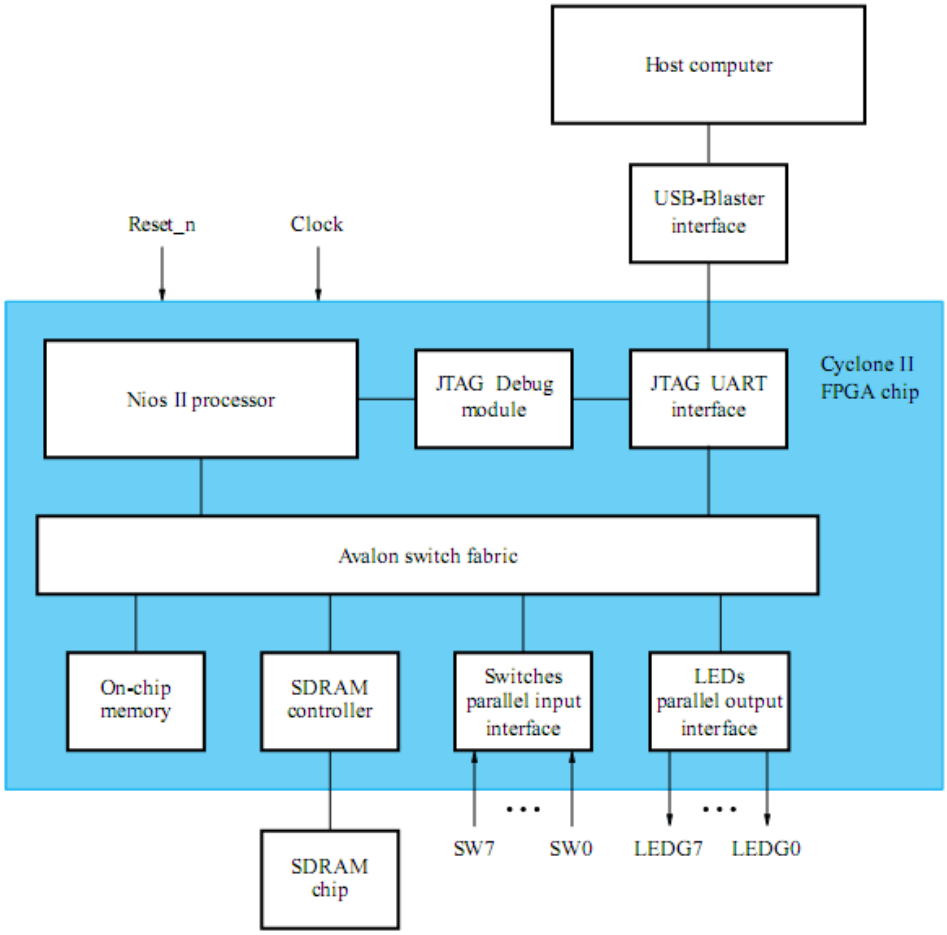
Lab 1　LED(PIO)控制實習

銘傳大學電腦與通訊工程學系

陳慶逸

一、背景知識

1. 下面為一個嵌入式系統，它包含了 Nios II CPU、On-Chip Memory、PIO、UART、Timer 和 Avalon Bus, 等等，這些是硬體的部份，也就是 SOC 的基礎。除了基本的 Nios II processor，以及與 host computer 溝通的 JTAG 外，剩下的周邊都必須透過 Avalon bus 做溝通，on-chip memory 負責放 C 的程式碼。



2. 如何在 NIOSII 中操作 PIO？

我們以 NIOS II IDE 下的 hello_led.c 範例程式來進一步說明如何在 NIOS II 中操作 PIO:

```
1    #include "system.h"
2    #include "altera_avalon_pio_regs.h"
3    #include "alt_types.h"
```

```
4
5    int main (void) __attribute__ ((weak, alias ("alt_main")));
6
7    int alt_main (void) {
8        alt_u8 led = 0x2;
9        alt_u8 dir = 0;
10       volatile int i;
11

12     while (1) {
13         if (led & 0x81)
14         dir = (dir ^ 0x1);
15
16         if (dir)
17            led = led >> 1;
18         else
19           led = led << 1;
20
21       IOWR_ALTERA_AVALON_PIO_DATA(LED_GREEN_BASE, led);
22
23         i =0;
24         while (i<200000)
25            i++;
26         }
28       return 0;
29 }
30
```

在 hello_led.c 範例程式中我們可以看到下面的語法：

IOWR_ALTERA_AVALON_PIO_DATA(LED_PIO_BASE, led);

其中 IOWR_ALTERA_AVALON_PIO_DATA 我們可在 altera_avalon_pio_regs.h 找到定義：

```
#include <io.h>
#define IORD_ALTERA_AVALON_PIO_DATA(base) IORD(base, 0)
#define IOWR_ALTERA_AVALON_PIO_DATA(base, data) IOWR(base, 0,
 data)
```

因此在 NIOSII 程式設計中，找們是可以引入#include <io.h>，直接使用 IORD 和 IOWR 來操作 PIO 的。

SOPC Builder 會分配每個 component 的 base address，當 Nios II EDS 產生 system library 時，使會將所有的 address 定義在\software\hello_world_0_syslib\Debug\system_description\system.h 下。例如

```
#define LED_PIO_BASE 0x00004000
```

其中 LED_PIO_BASE（IO 暫存器位址）即為 0x00004000

因此無論我們用 IOWR(LED_PIO_BASE, 0, led)或 IOWR(0x00004000, 0, led)的寫法，都可以代替 IOWR_ALTERA_AVALON_PIO_DATA(LED_PIO_BASE, led)。

二、硬體設計(Nios II Hardware Development)

1. Create New Project

1-1 Choose **Programs > Altera® > Quartus II 10.0**

1-2 Choose **New Project Wizard.**

1-2-1. Browse to the directory for your design Directory, type the Name of this project, and the Top-level Entity.

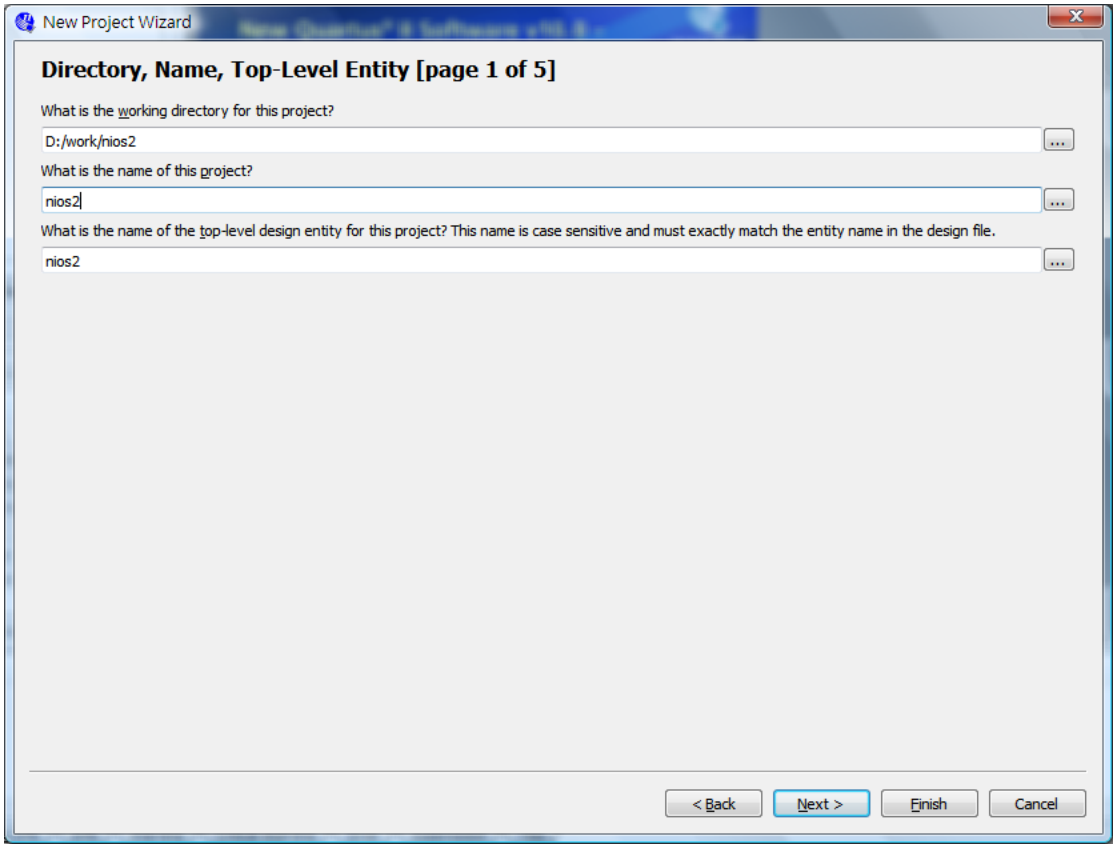


Fig.1 Make nios2 project

1-2-2. Family & Device Settings:

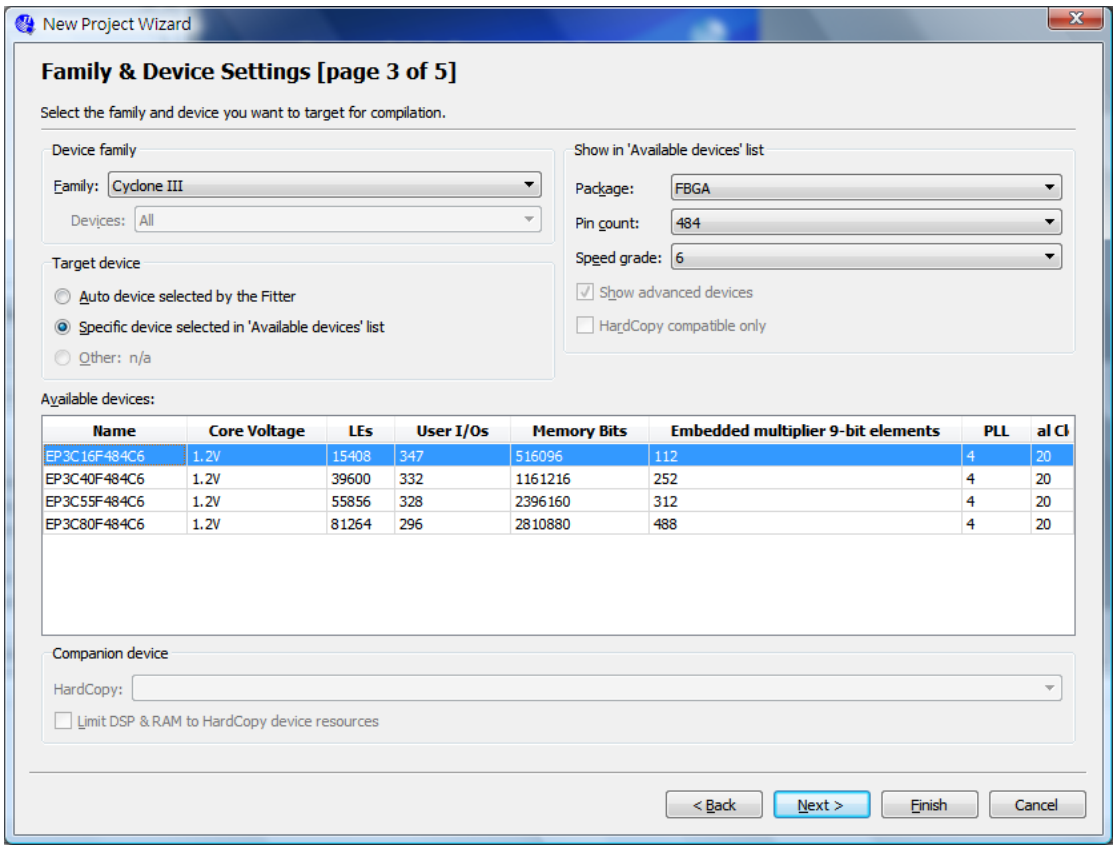
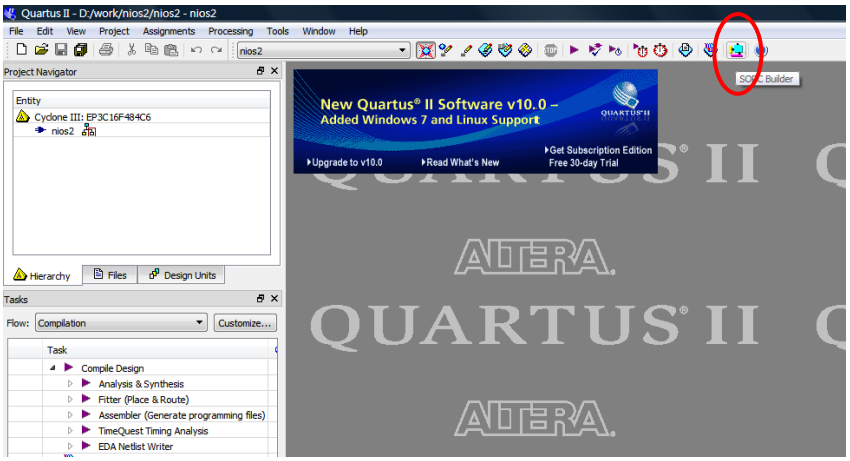


Fig.2 Choice the **CycloneIII EP3C16F484C6.**

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

2-1 Choose SOPC Builder (Tools menu), SOPC Builder displays the Create New System dialog box.



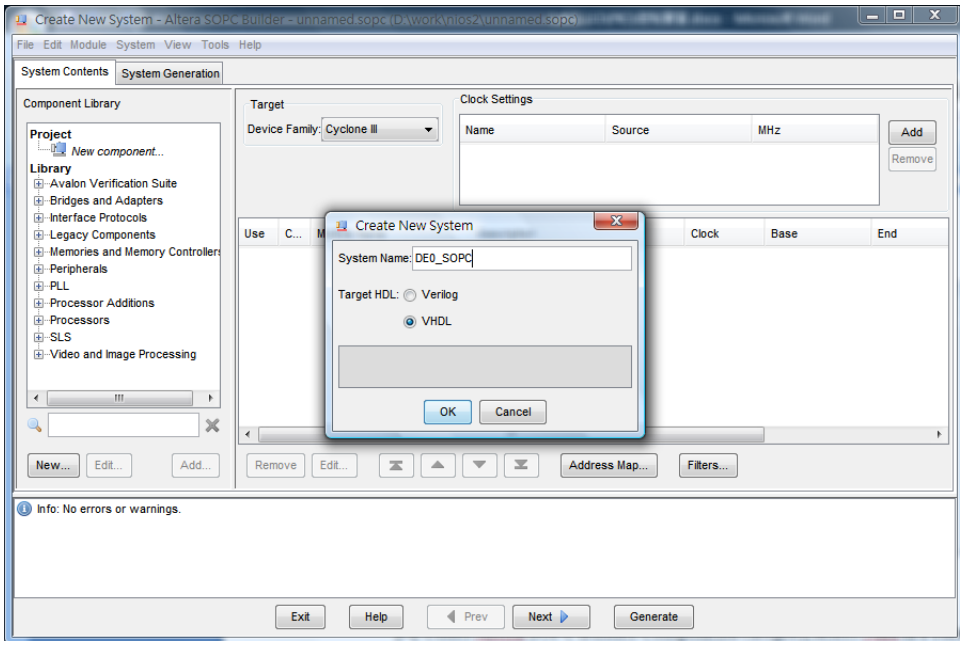


Fig.3 Type **DE0_SOPC** in the System Name field.

2-2 Under Altera SOPC Builder Components category, select **Nios II Processor**.

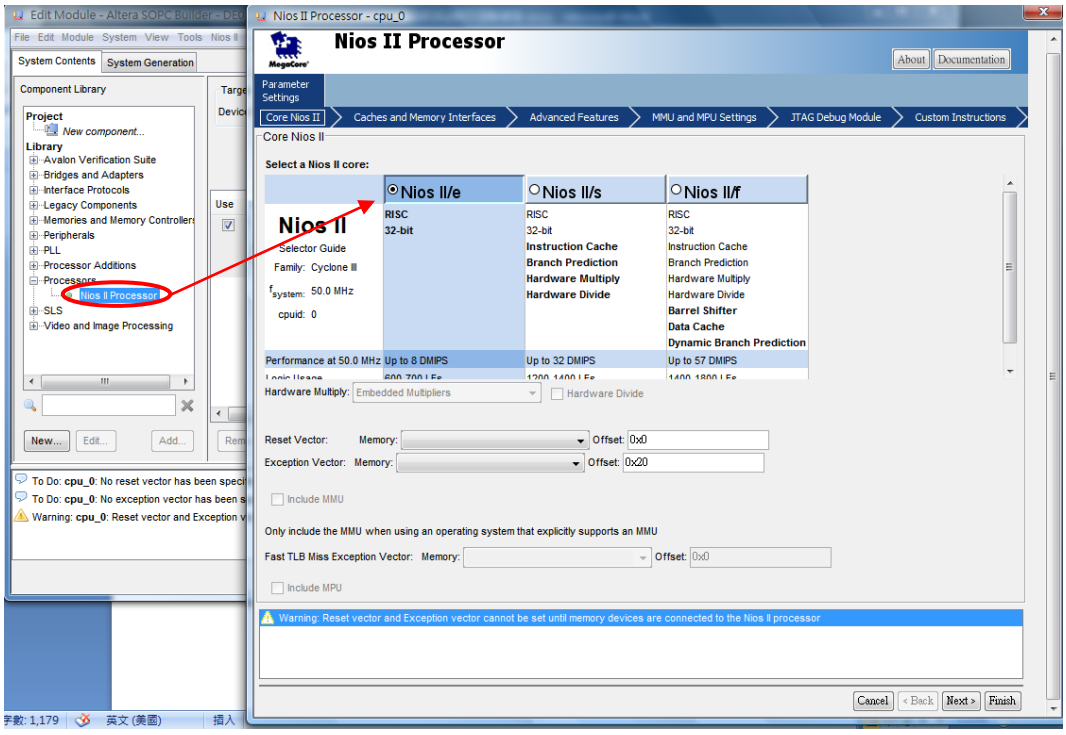


Fig. 4 Nios II Processor

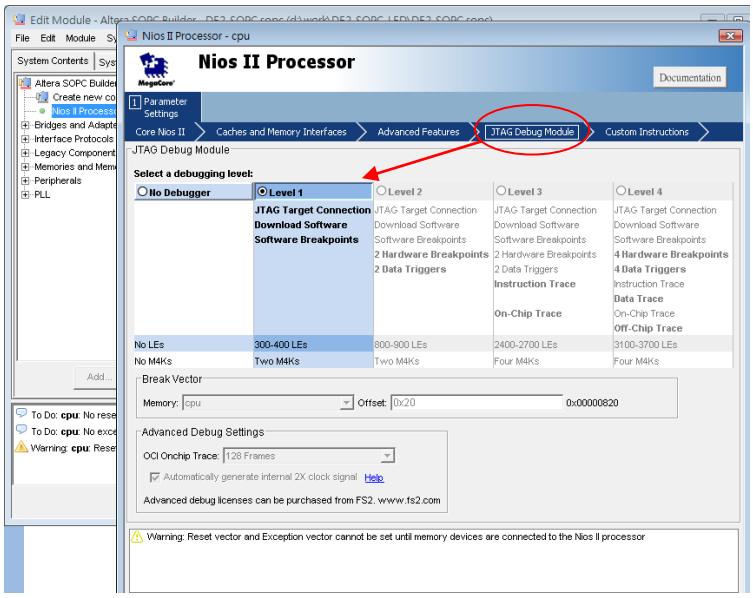


Fig. 5 Click the **JTAG Debug Module** tab and choose the “Level 1” degugging level.

2-3 Select **On-Chip Memory (RAM or ROM)** under the **Altera SOPC Builder** > Memory and Memory Controllers category. Type **40 KBytes** in the **Total Memory Size box** to specify a memory size of 40 Kbytes.

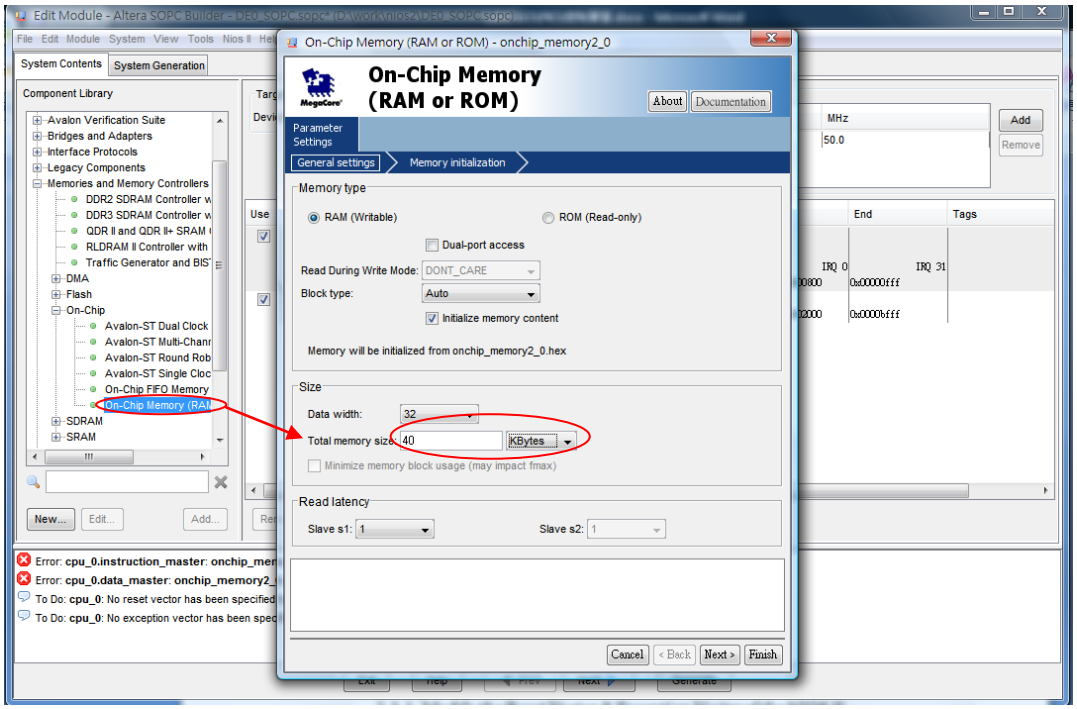


Fig.6 Add onchip_memory

2-3-1 Modify the Reset Vector & Exception Vector of the NIOS II

Processor.

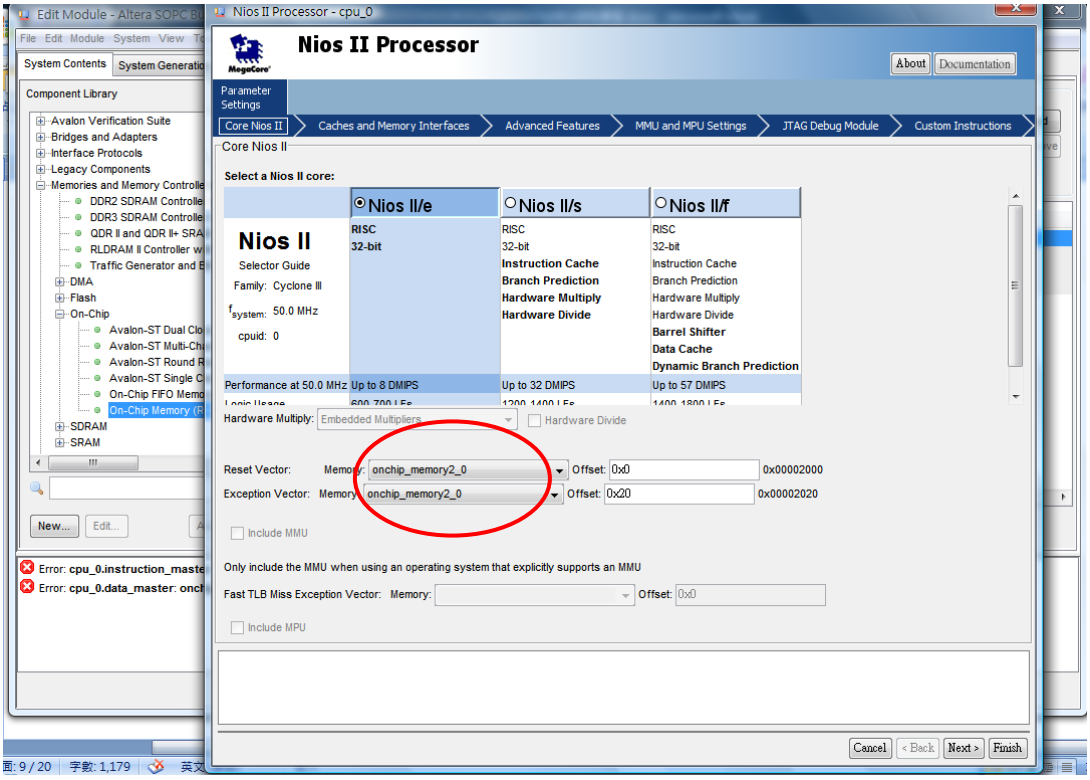


Fig.7 Modify the Reset Vector & Exception Vector

2-3-2 Choose **Auto-Assign Base Addresses** (System **menu**) to make SOPC Builder assign functional base addresses to each component in the system.

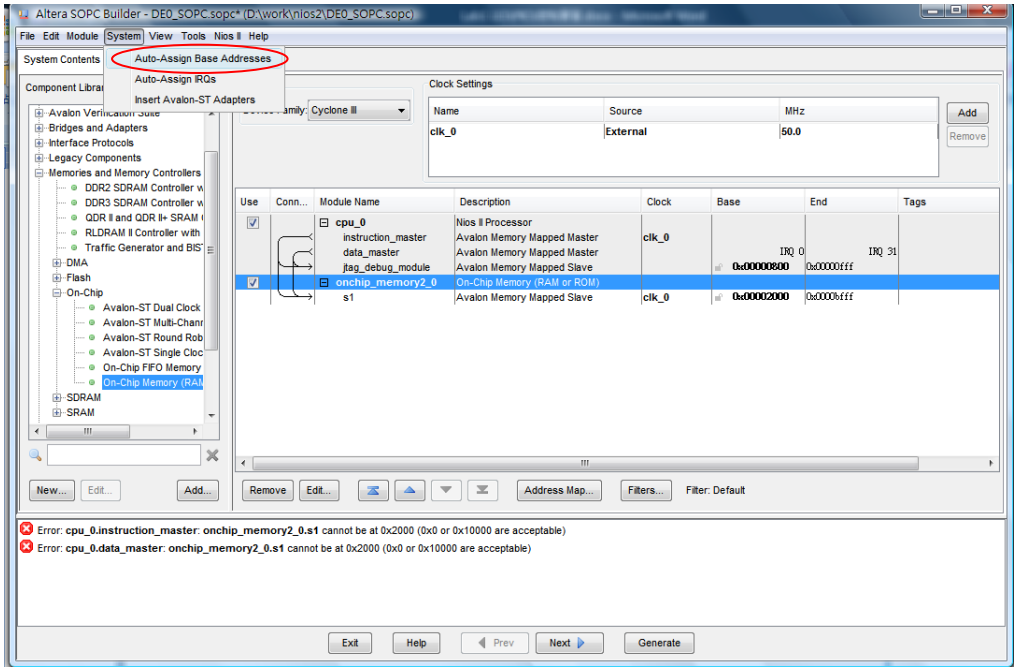
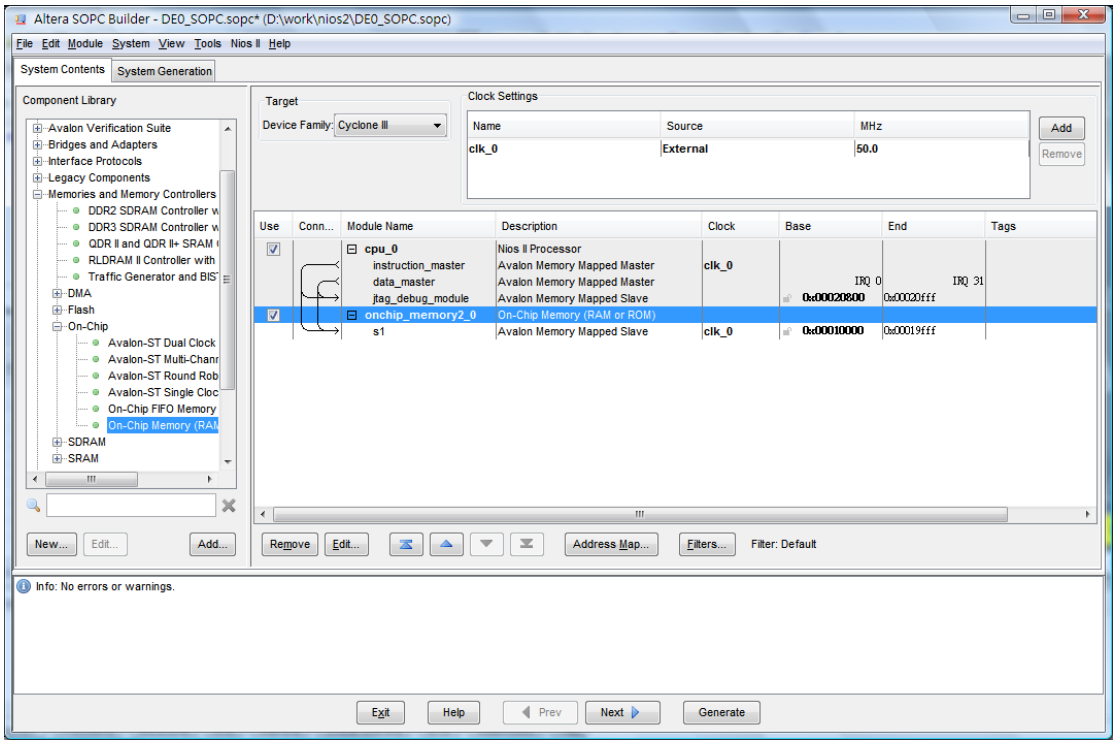


Fig.8 System/ Auto-Assign Base Addresses



2-4 Select **JTAG UART** under the **Altera SOPC Builder** > Interface Protocols > **Serial** category. Add JTAG_UART.

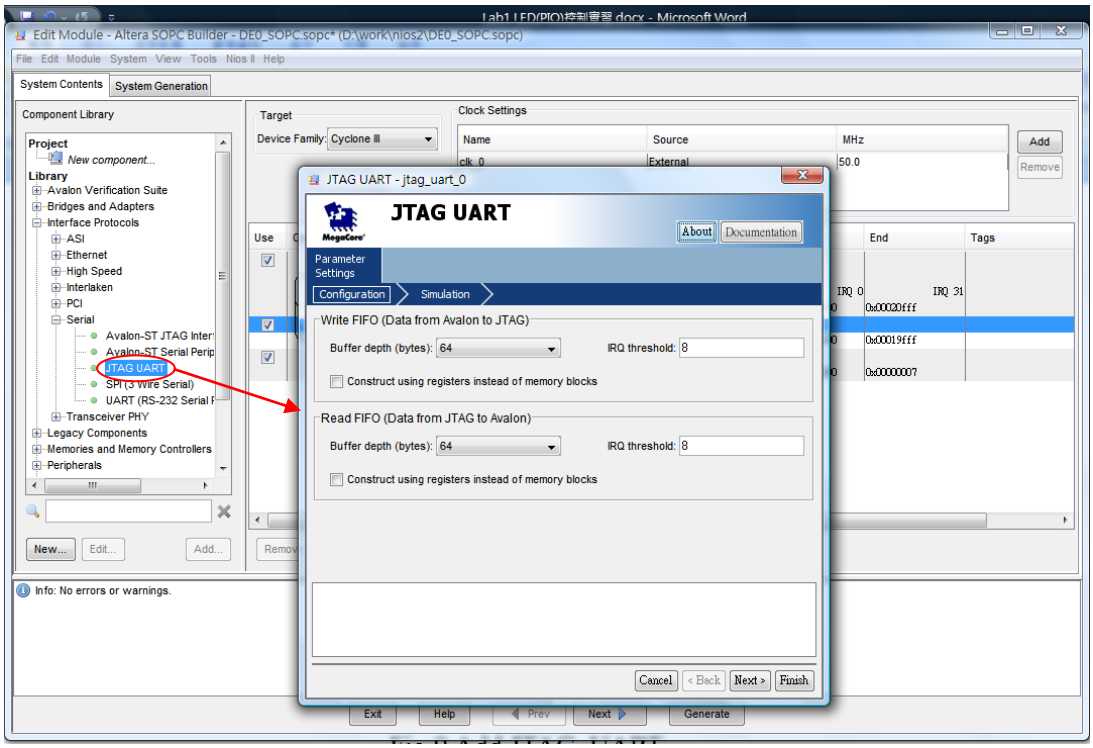


Fig.9 Add JTAG_UART

2-5 Select **PIO (Parallel I/O)** under the **Altera SOPC Builder**> **Peripherals > Microcontroller Peripherals** category. Add LED (PIO). Right-click **pio** and select Rename. Type **led_pio** and press Enter.

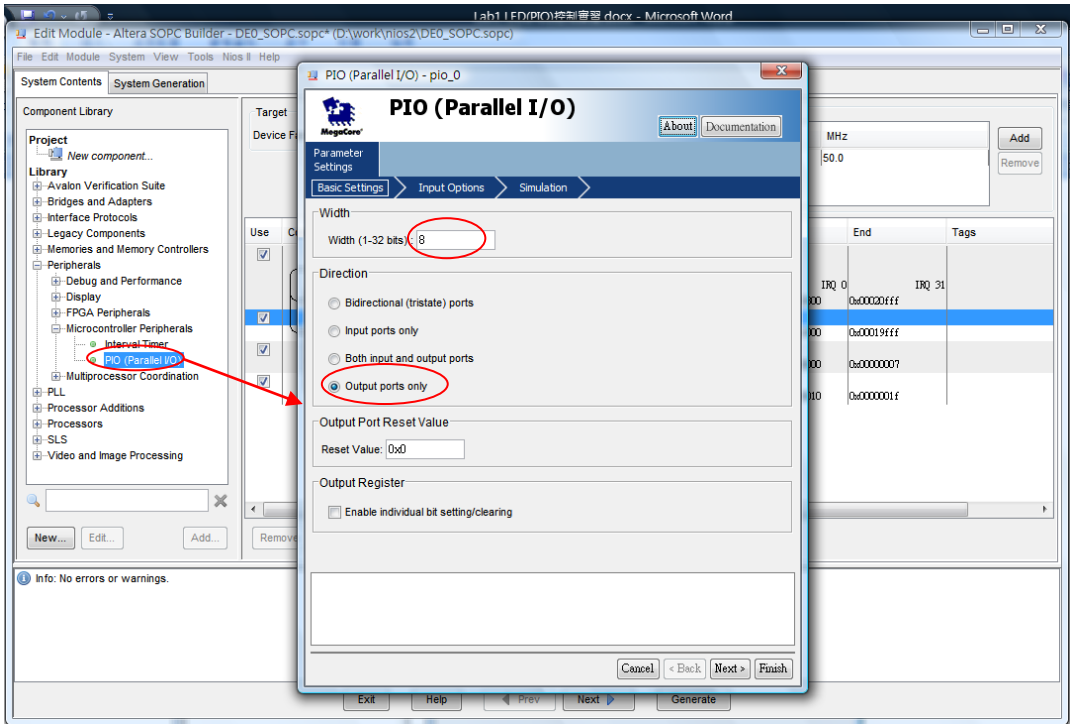


Fig.11 Add LED (PIO)

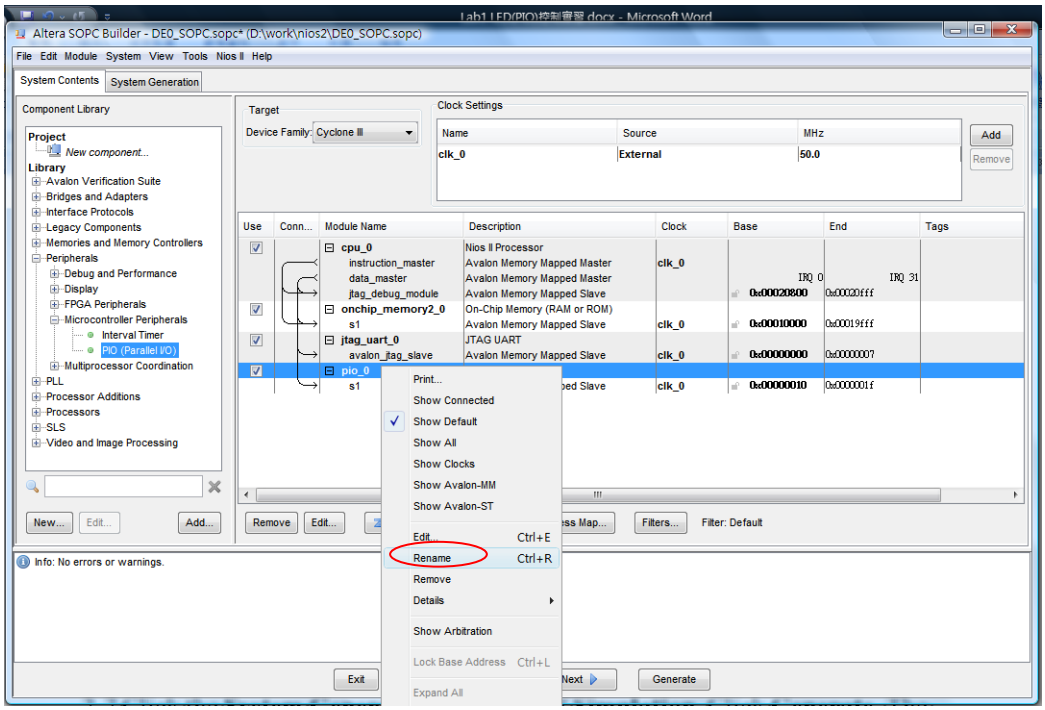
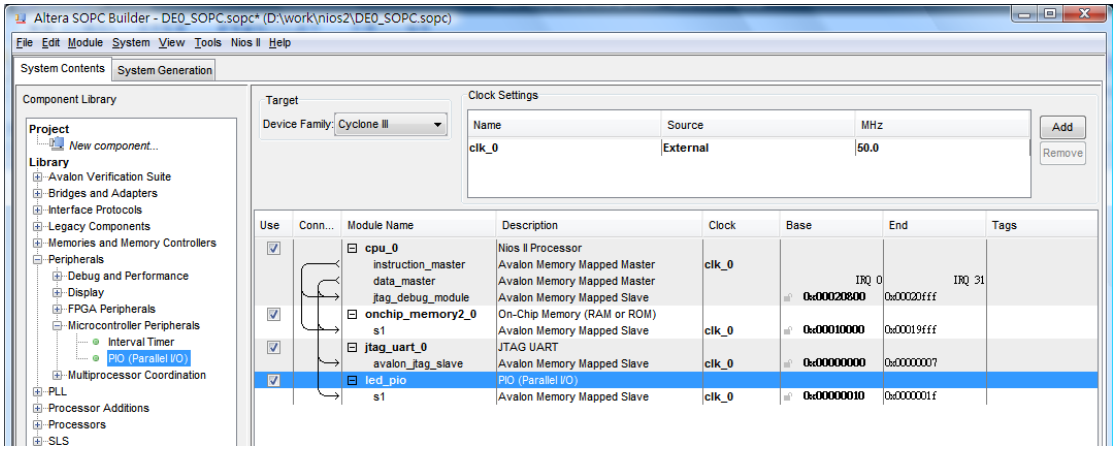


Fig.12 Select rename (led_pio)



2-6 Click the **System Generation** tab, turn off **Simulation**. Click **Generate**. The system generation process begins. The generation process can take several minutes. When it completes, the System Generation tab displays a message "**SUCCESS: SYSTEM GENERATION COMPLETED**".

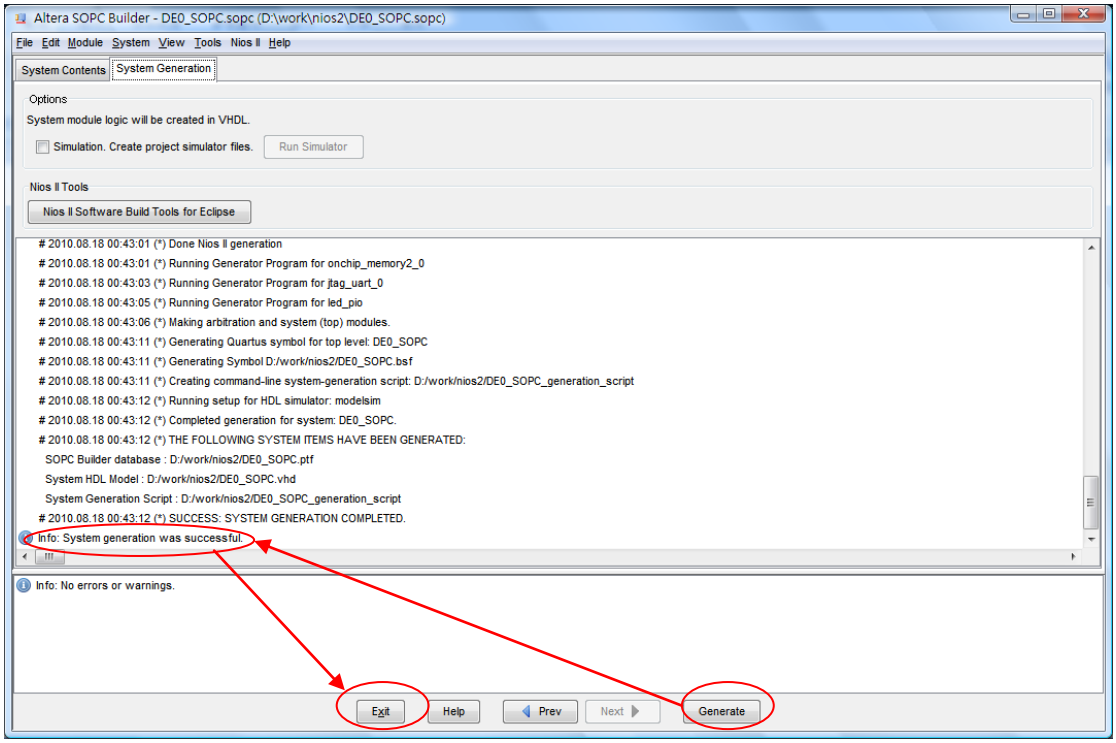


Fig. 13 Click **Generate**

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

3-1 Adding the Quartus II Symbol to the BDF:

3-1-1 File>New>Block Diagram/Schematic File

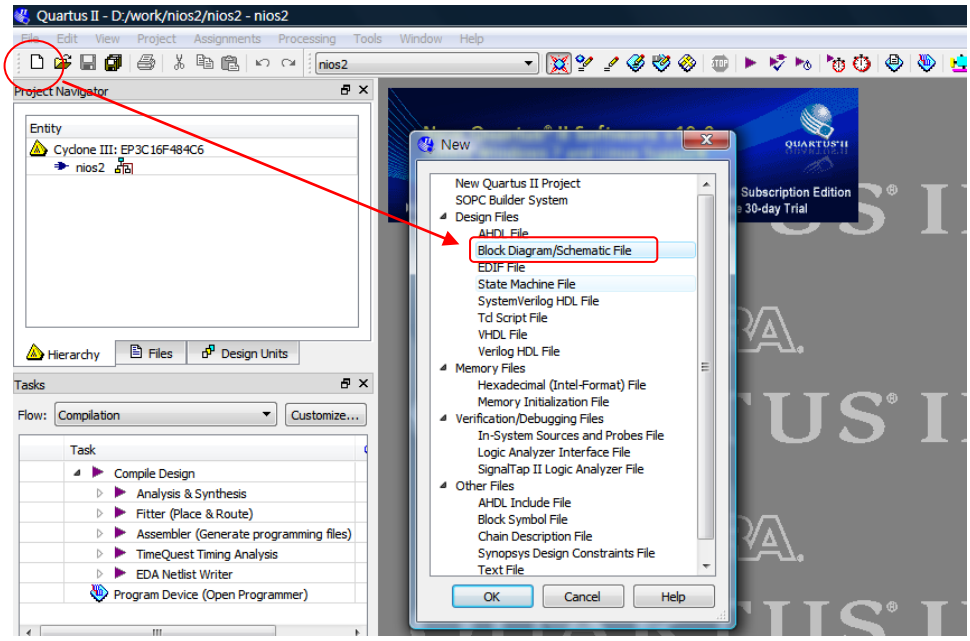
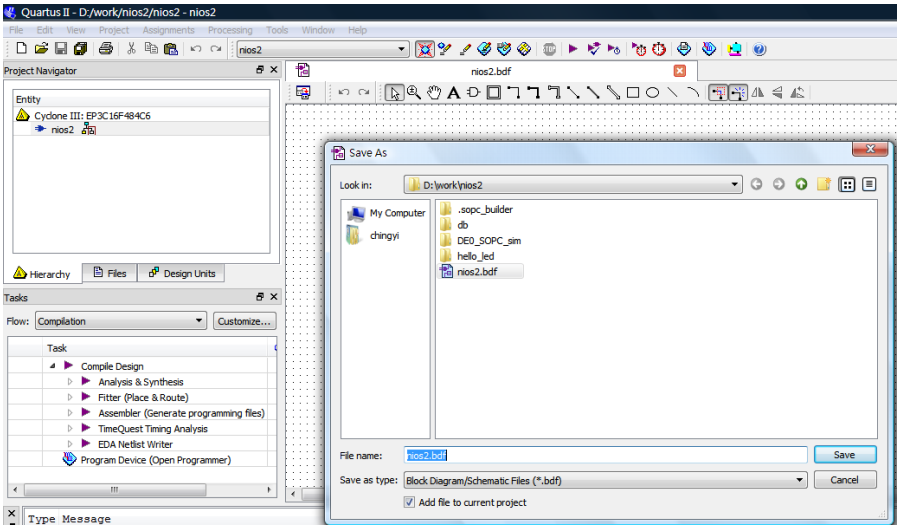


Fig.14 File>New>Block Diagram/Schematic File

3-1-2 File>Save as> nios2.bdf



3-1-2 Select Project >DE0_SOPC. The Symbol dialog box displays the DE0_SOPC symbol. Click OK.

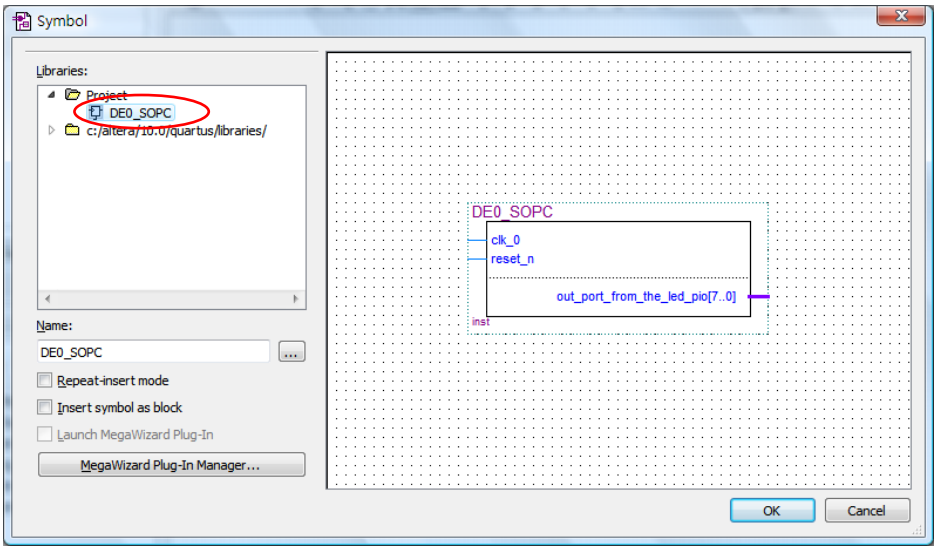


Fig.15 Add DE0_SOPC symbol

3-1-3 Right-click **DE0_SOPC** symbol and select “Generate pins for Symbol ports”:

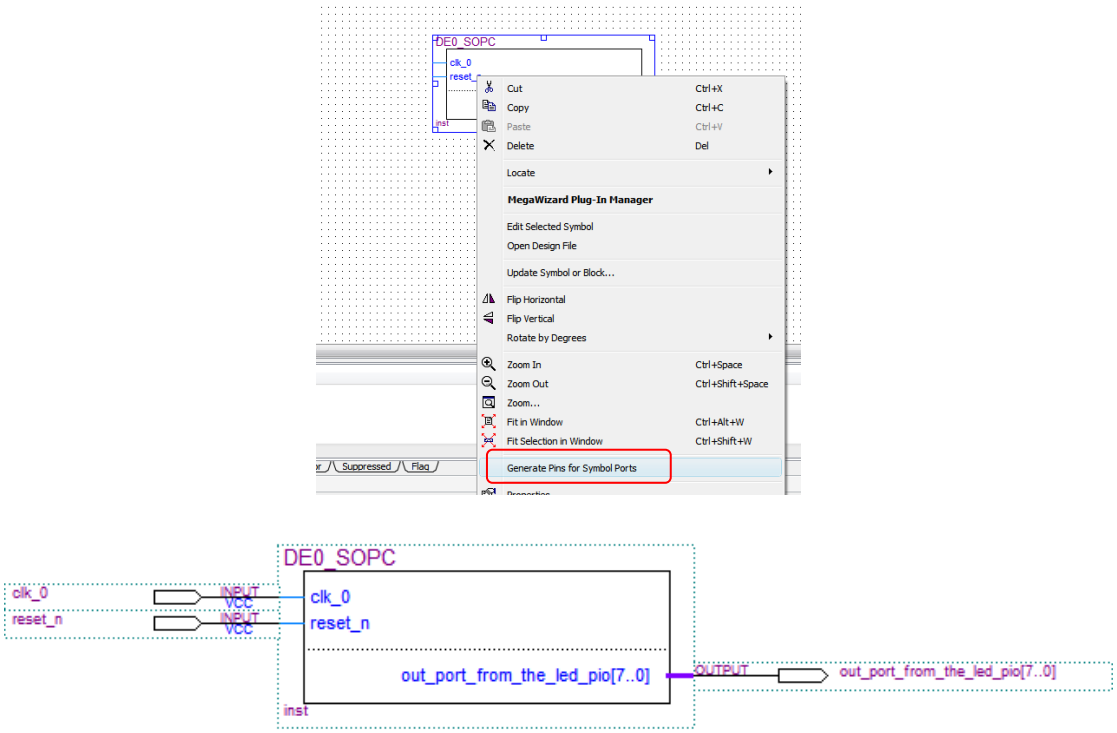


Fig.16 Generate pins for Symbol ports

3-1-4 Rename Symbol ports:

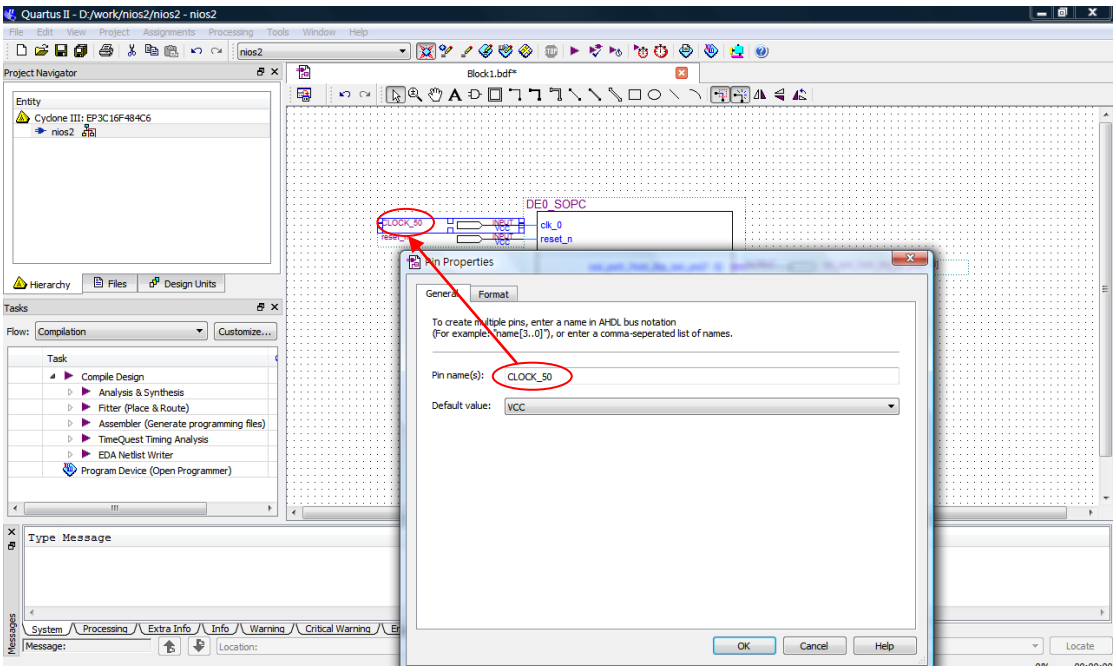


Fig.17 Rename Symbol ports

3-1-5 Finish Quartus II top design(DE2_SOPC_LED.bdf):

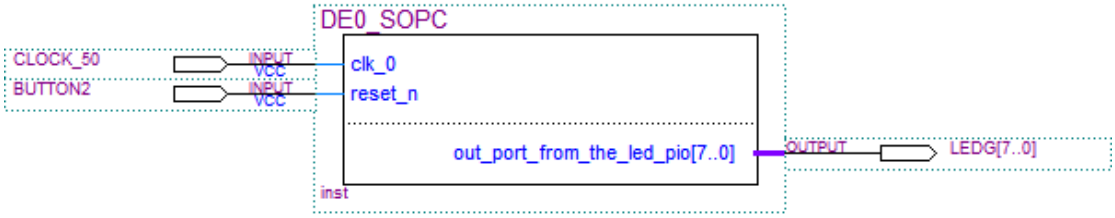
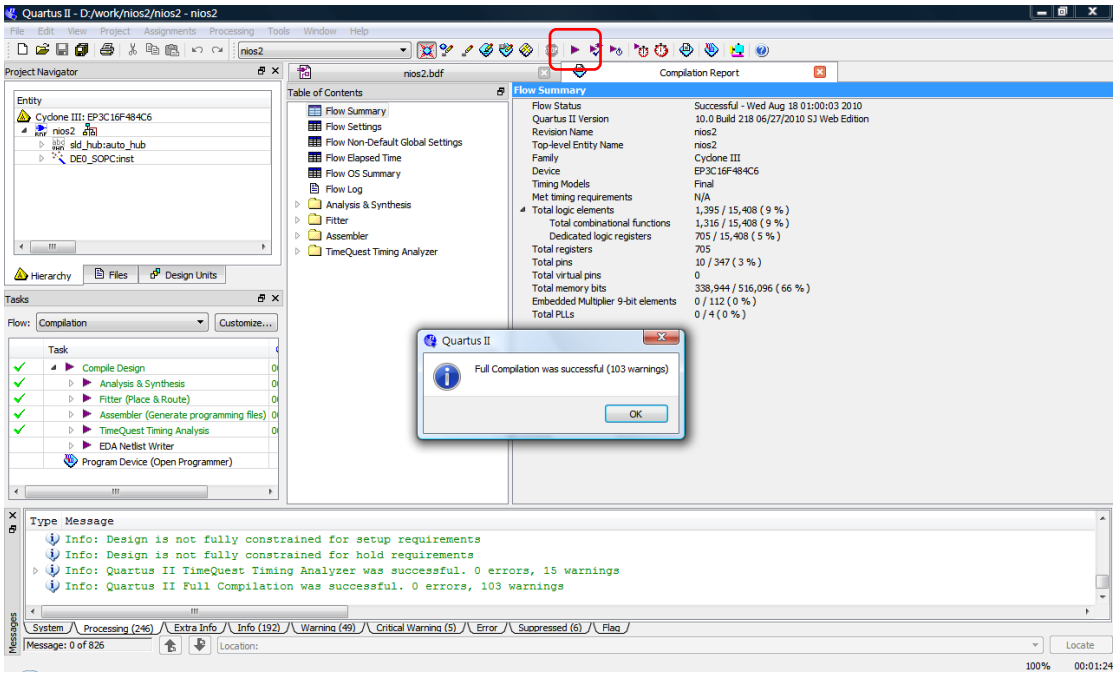


Fig.18 Top design(nios2.bdf)

3-1-6 Compiler the design



4. Pin Assignments & Download

4-1 Choose **Assignments > Pin Planner**

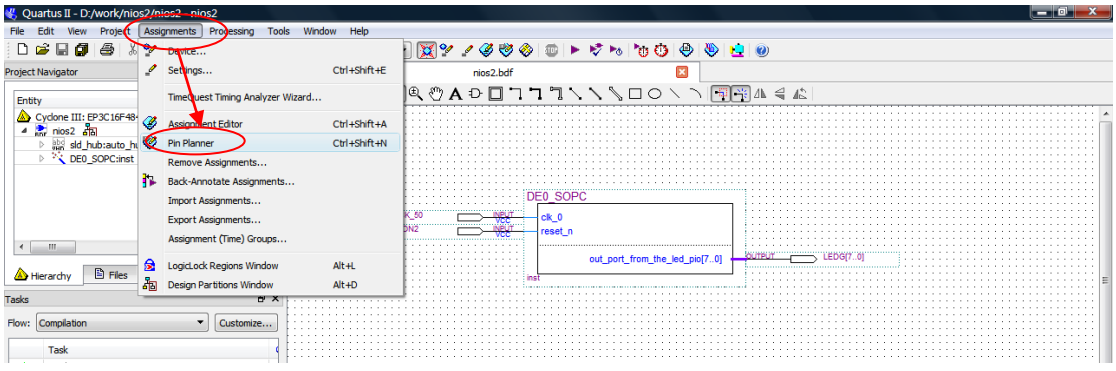


Fig.19 Pin Planner

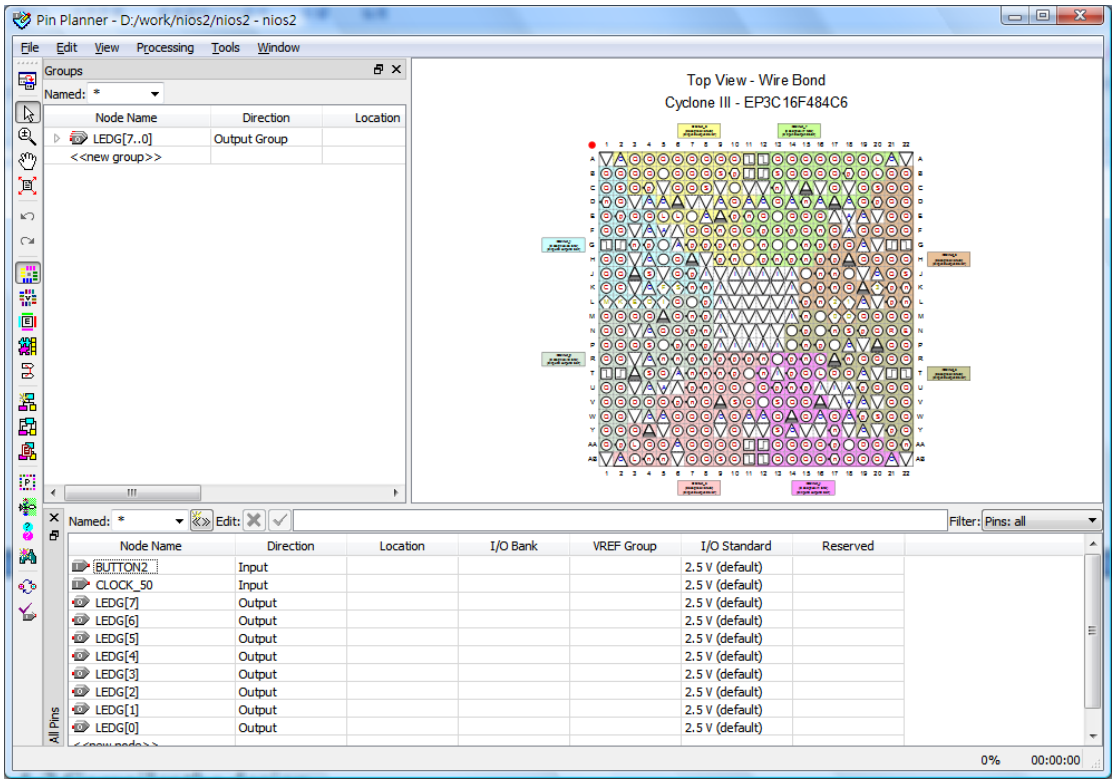
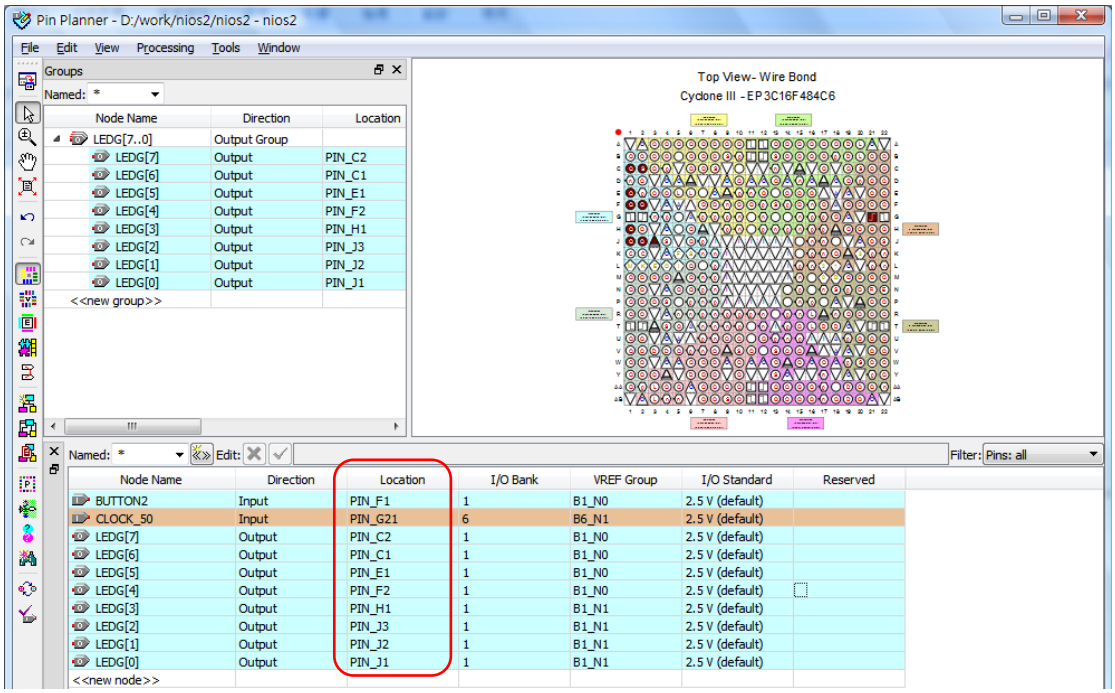


Fig.20 Pin Planner



4-2 Compiler the design.

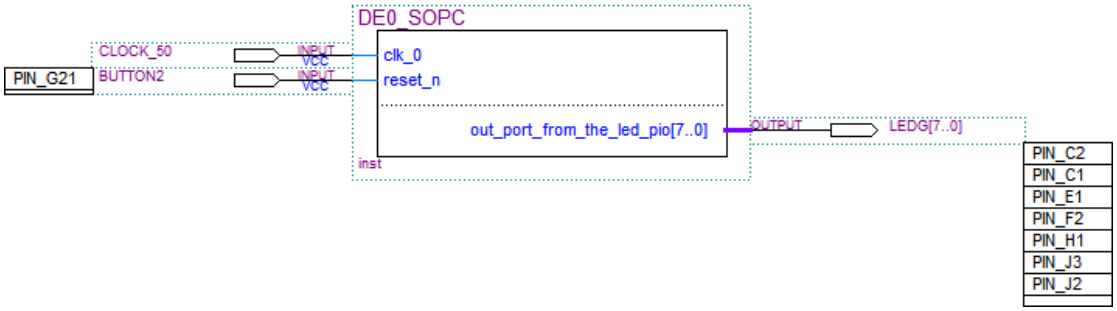


Fig.21 Compiler the design

4-3 download the hardware

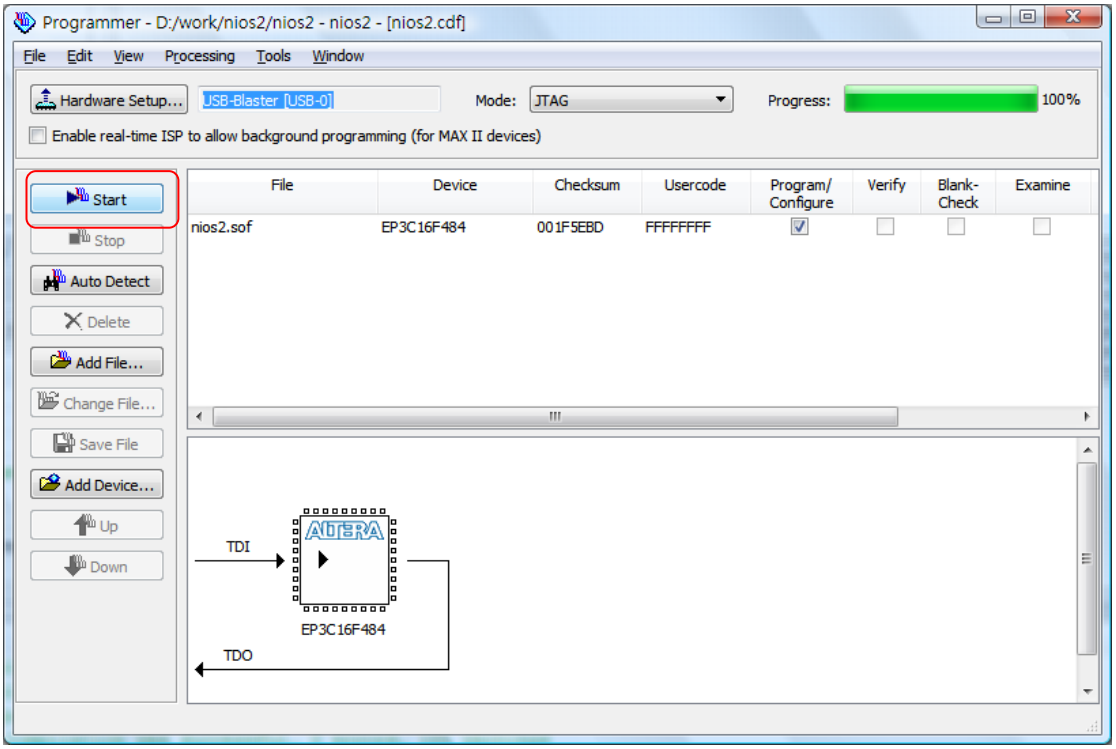
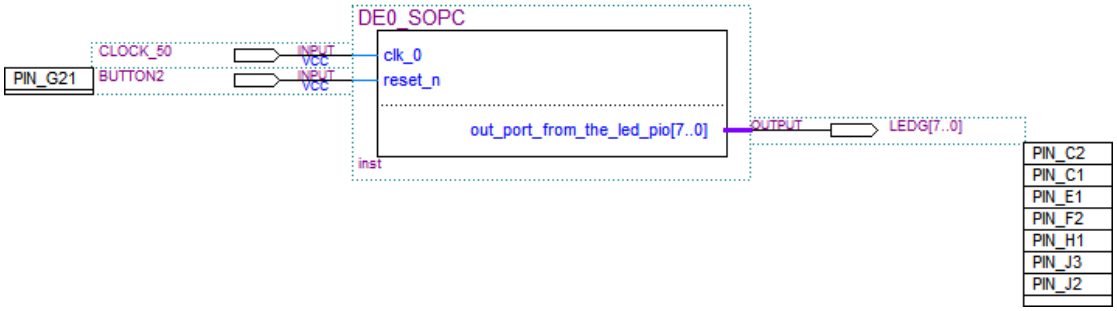


Fig.22 download the hardware

P.S.

Top design VHDL code:



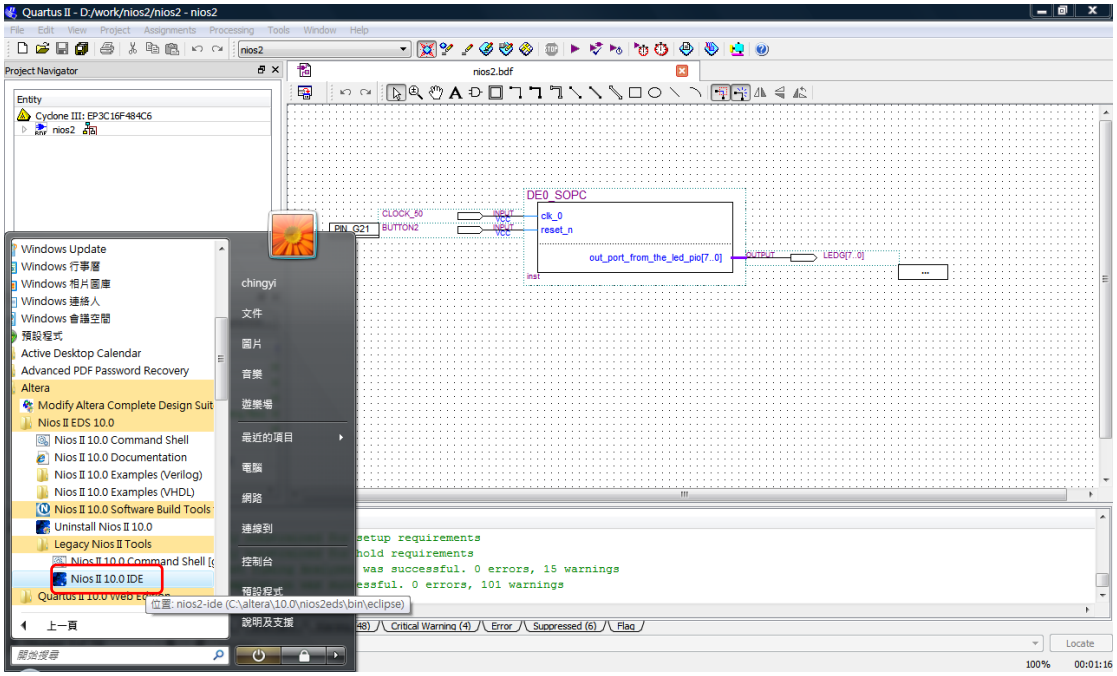
```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
LIBRARY work;

ENTITY nios2 IS
	PORT
	(
		CLOCK_50 :	IN	STD_LOGIC;
		BUTTON2 :	IN	STD_LOGIC;
		LEDG :	OUT	STD_LOGIC_VECTOR(7 DOWNTO 0)
	);
END nios2;

ARCHITECTURE bdf_type OF nios2 IS

COMPONENT de0_sopc
	PORT(clk_0 : IN STD_LOGIC;
		reset_n : IN STD_LOGIC;
		out_port_from_the_led_pio : OUT STD_LOGIC_VECTOR(7 DOWNTO 0)
	);
END COMPONENT;

BEGIN
b2v_inst : de0_sopc
PORT MAP(clk_0 => CLOCK_50,
		reset_n => BUTTON2,
		out_port_from_the_led_pio => LEDG);
END bdf_type;
```

Top design Verilog code:

```
module nios2(
	CLOCK_50,
	BUTTON2,
	LEDG
);

input wire	CLOCK_50;
input wire	BUTTON2;
output wire [7:0] LEDG;

DE0_SOPC	b2v_inst(
	.clk_0(CLOCK_50),
	.reset_n(BUTTON2),
	.out_port_from_the_led_pio(LEDG));

Endmodule
```

三、軟體設計

Choose **Programs>Altera>Legacy Nios II Tools>Nios II 10.0 IDE**



1. Create a New Nios II IDE Project

1-1 Choose File > switch workspace

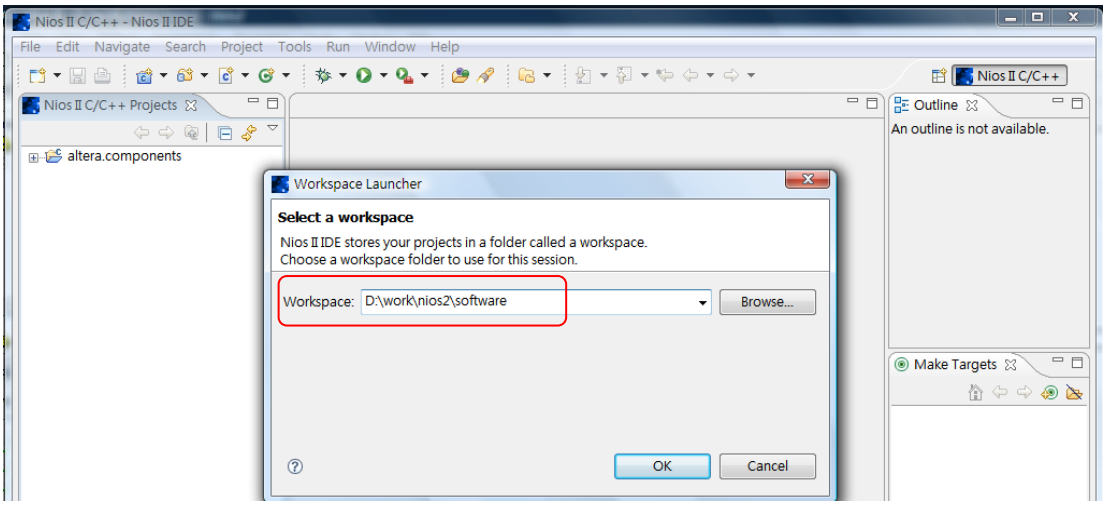
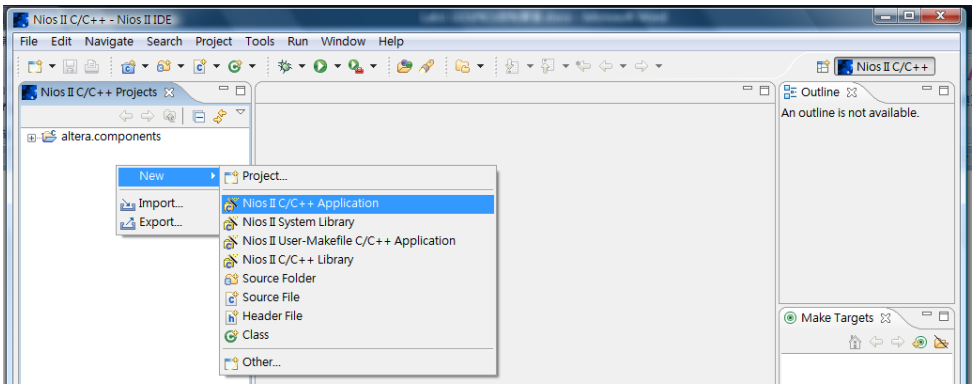


Fig.23 Switch workspace

1-2 Choose File >New >Nios C/C++ Application. The first page of New Project wizard opens.



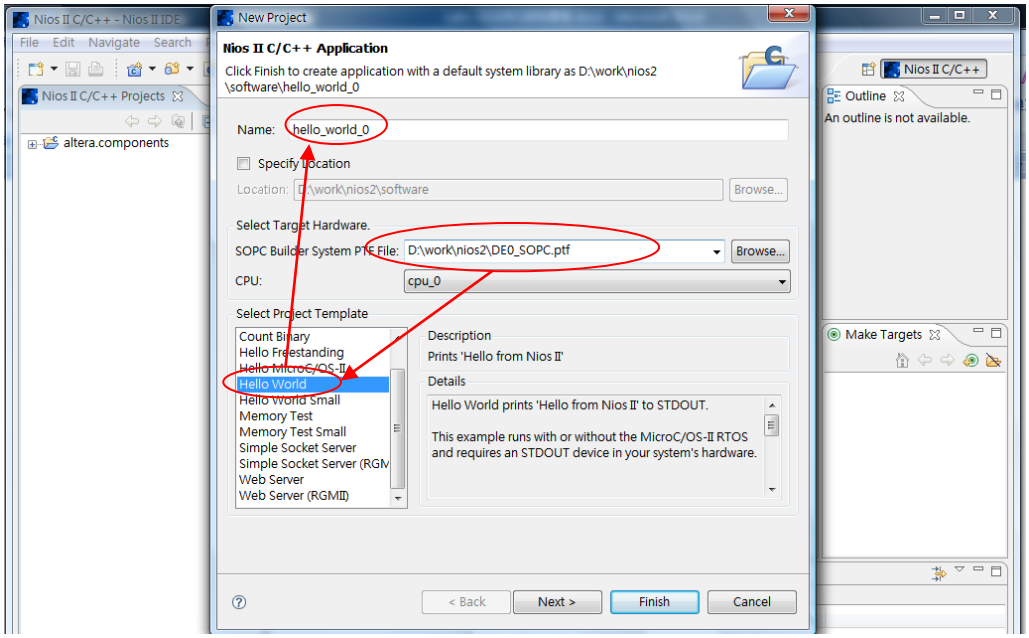


Fig.24 Nios C/C++ Application

1-3 hello_world.c

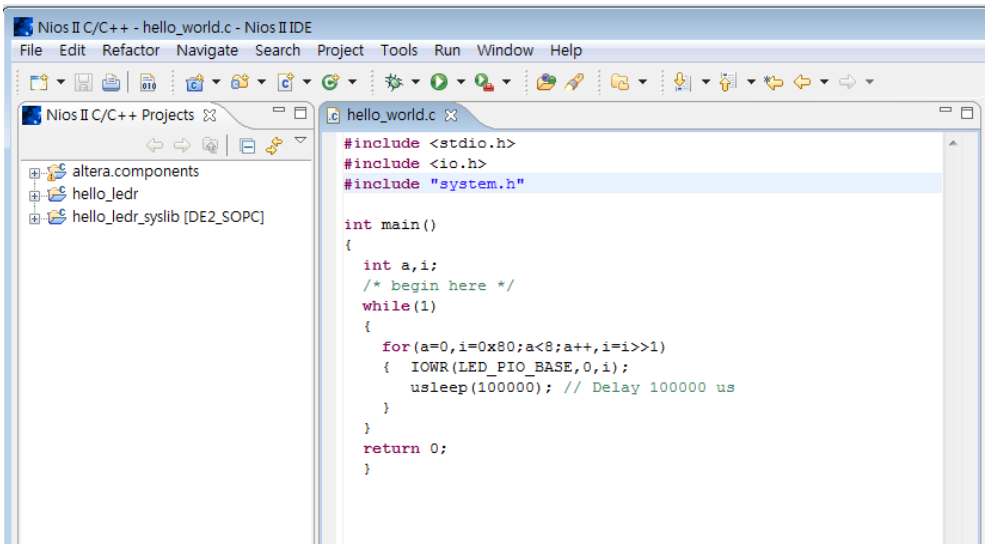


Fig.25 hello_world.c

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

int main()
{
   int a,i;
   /* begin here */
   while(1)
   {
        for(a=0,i=0x80;a<8;a++,i=i>>1)
        {  IOWR(LED_PIO_BASE,0,i);
           usleep(100000); // Delay 100000 us
        }
   }
  return 0;
}
```

P.S.

位元運算子主要功能在於對單一位元執行設定(為 1)、清除(為 0)、邏輯運算、移位等處理。這些位元運算子可以應用在所有的整數變數(int)和字元變數(char)。

位元運算子和它們的功能

&	Bitwise AND	兩個位元進行位元 AND 運算
|	Bitwise OR	兩個位元進行位元 OR 運算
^	Bitwise exclusive OR	兩個位元進行位元 XOR 運算
~	Complement	一個位元的位元 NOT 運算
<<	Shift left	位元向左移位
>>	Shift right	位元向右移位

1-4 Build Project

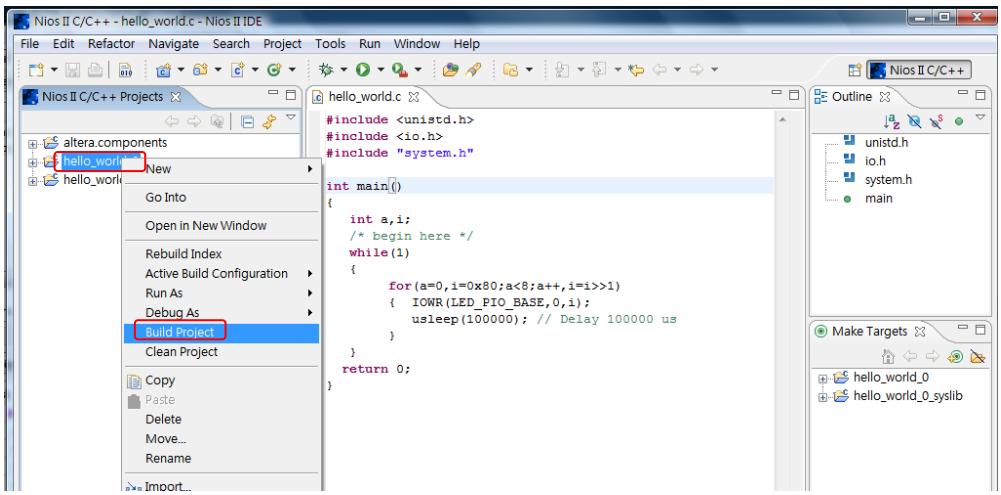


Fig.26 Build Project

1-5 Run As > Nios II Hardware

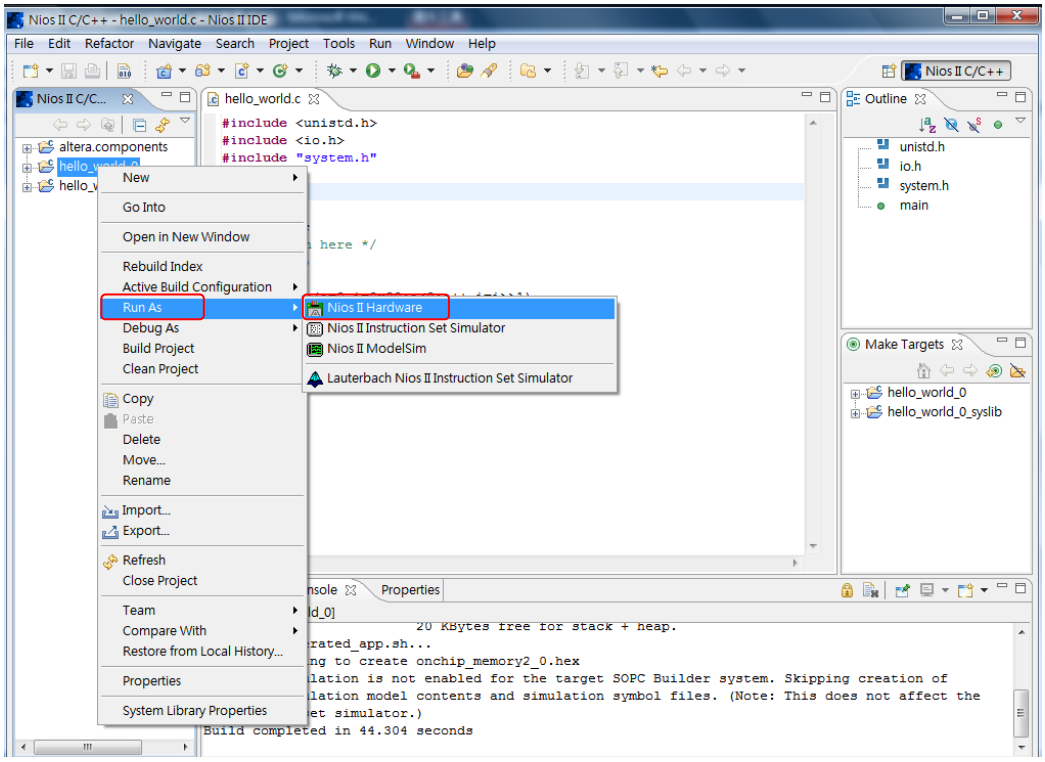


Fig.27 Run As Nios Hardware

四、結論與探討

1. 試述對軟硬體開發流程的認識。
2. 試述本實驗所用到的硬體資源。
3. 試思考如何分別從硬體和軟體部份來更改 LED 移動速度。
4. 試思考如何改變移動方式，如左旋、右旋等功能。

隨堂練習一：

1. 試在 DE0 實驗板上實現 LED 以 0.2s 閃爍的功能設計。
   提示： 0 xor 1 = 1;
   1 xor 1 = 0
   C 語言中，執行 XOR 的位元運算子為 “^”。

&	Bitwise AND	兩個位元進行位元 AND 運算
|	Bitwise OR	兩個位元進行位元 OR 運算
^	Bitwise exclusive OR	兩個位元進行位元 XOR 運算
~	Complement	一個位元的位元 NOT 運算
<<	Shift left	位元向左移位
>>	Shift right	位元向右移位

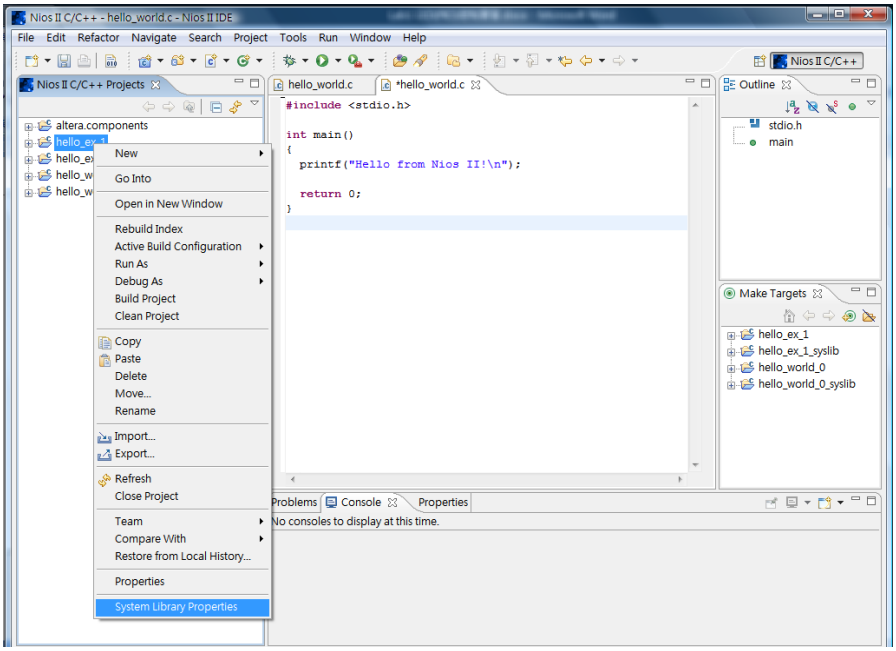
```
#include <io.h>
#include "system.h"
#include "unistd.h"


int main(void)
{
  int led_mask=0;
  while(1)
  {
    ????????????
  } // while
  return 0;
}
```

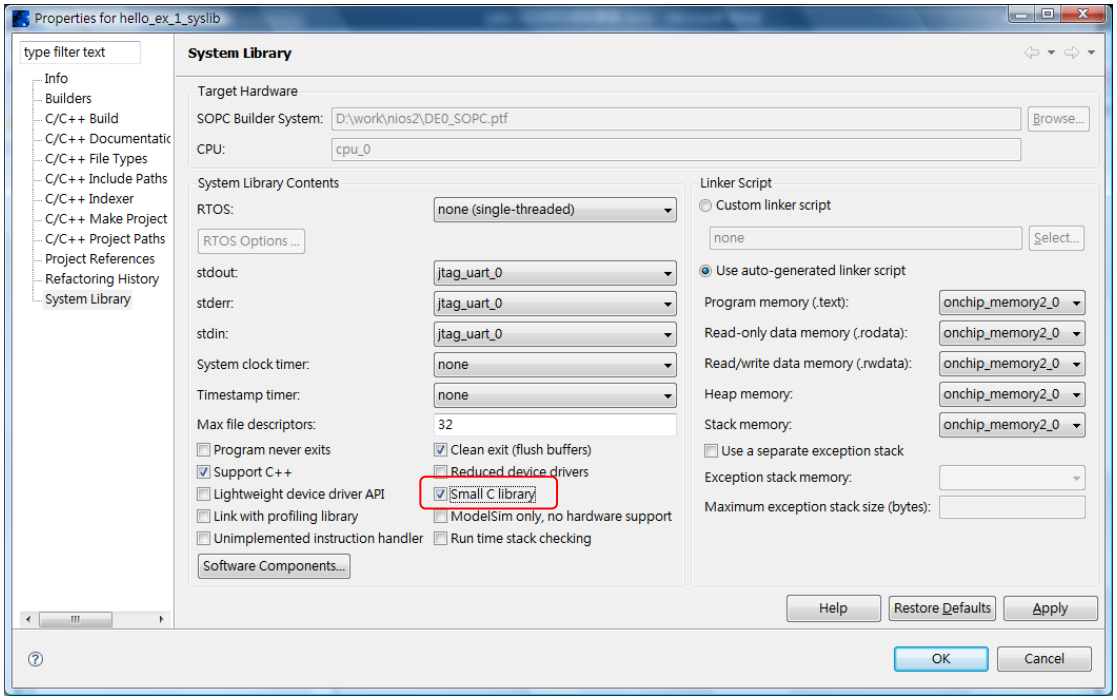
2. 試設計累加計數的功能(delay time : 1s)，並將結果顯示在 LED 上。
   提示： led ++ ;

```
#include <stdio.h>
#include <io.h>
```

```c
#include "system.h"
#include "unistd.h"

int main(void)
{
  char led=0;
  while(1)
  {
     ?????????????;
   }
  return 0;
}
```

此處，我們在 Nios II IDE 新增一個 project，並在 System library properties 下勾選 small C library 選項：





完成之後，編譯下面 Hello world 的程式即可成功(有 printf 敘述)，不會再出現 on-chip memory 太少的錯誤訊息。

```
#include <stdio.h>
```

```c
int main()
{
  printf("Hello from Nios II!\n");

  return 0;
}
```

因此，我們可以將 Nios II 平台當成我們所習慣的電腦來設計程式，例如在 NiosII 實現一個處理兩組位元作 AND 運算的簡單程式，並將結果顯示在 console 上：

```c
#include <stdio.h>

int main()
{
  int a,b,c;
  a=3; //0011
  b=7; //1111
  c=a & b; //(0011) AND (1111) = (0011)
  printf("c = %d",c);

  return 0;
}
```

P.S. 關於 printf 敘述的用法如下：

printf( ) 的列印格式、控制字元、修飾子

列印格式	輸出敘述
%c	字元
%s	字串
%d	十進位整數
%u	無號十進位整數
%o	無號八進位整數
%x	無號十六進位整數，以 0~f 表示
%X	無號十六進位整數，以 0~F 表示
%f	浮點數，小數點型式
%e	浮點數，指數 e 型式
%E	浮點數，指數 E 型式
%g	印出 %f 與 %e 較短者
%G	印出 %F 與 %E 較短者
%p	指標位址
%%	印出百分比符號

控制字元	功能
\a	警告音
\b	倒退
\f	換頁
\n	換行
\r	歸位
\t	跳格
\’	印出單引號
\』	印出雙引號
\\	反斜線
\/	斜線
\d	八進位 Ascii 碼
\x	十六進位 Ascii 碼

隨堂練習二：

1. 試判斷程式中 x,y,z 之值應為何(將結果顯示在 console 上)?

```
#include <stdio.h>

int main()
{
  int x,y,z;
  x=5;
  y=x++;
  z=++x;
  printf("x = %d\n",x);
  printf("y = %d\n",y);
  printf("z = %d\n",z);
  //IOWR(LED_PIO_BASE,0,c);

  return 0;
}
```

x = ?

y = ?

z = ?

2. 利用 for loop 設計一個累計 1+2+3…+10 的程式，並將結果顯示在 console 上。

```
#include <stdio.h>

int main()
{
    int sum = 0;
    int i;
    for (?????)
    {
      ??????????
    }

  printf("sum(1..10) = %d\n",sum);
```

```
  return 0;
}
```

Ans: sum(1..10) = 55