Software and Hardware Debugging

Introduction

This lab guides you through the process of setting debugging IP to desired signals in Vivado. The software debugging is based on the SDK to observe the values of variables and the memory that the switches changed the values. With the debugging IPs, the values of the internal signals can be observed in Logic Analyzer and compared with the values shown in SDK.

Objectives

After completing this lab, you will be able to:

- Use SDK Debugger to set break points and view the content of variables and memory
- Use Logic Analyzer to observe hardware signal and debug

Design Description

You will use the hardware design created in lab 4 to add debugging IP.

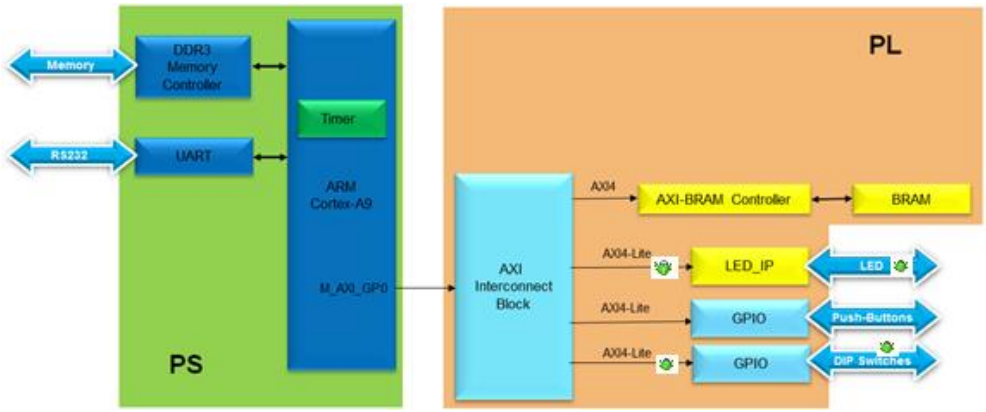

Figure 1

Design Flow

1 將 lab4 的專案另存為 lab6

2 選擇欲 Debug 之 Signal 並進行設定

2.1 點選 led_ip 的 S_AXI 和 LED[7:0]之後，右鍵點選 Mark Debug，如 Figure 2 所示。

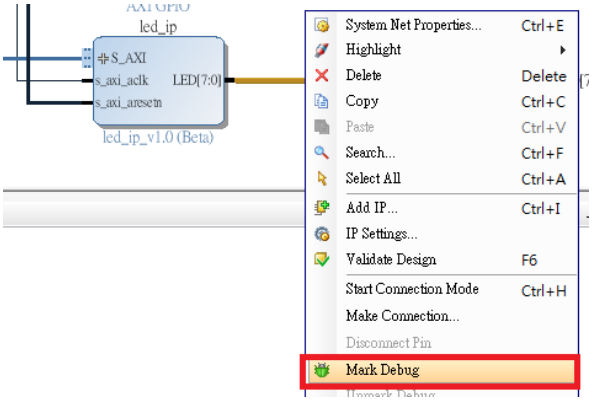

Figure 2

2.2 點選 switche 的 S_AXI 和 switches 之後，右鍵點選 Mark Debug，如 Figure 3 所示

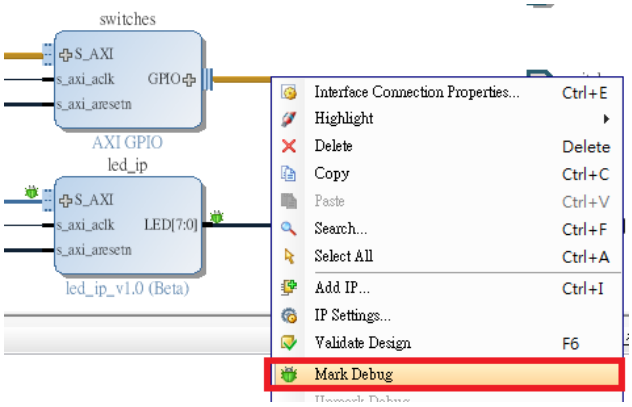

Figure 3

2.3 完成後如 Figure 4 所示，並存檔

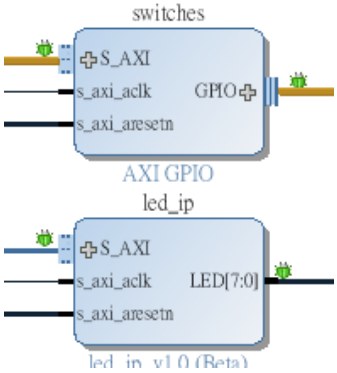

Figure 4

2.4 點選 Run Synthesis 進行合成，如 Figure 5

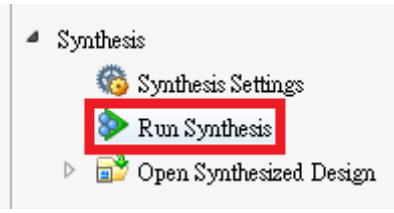

Figure 5

2.5 合成後點選 Open Synthesized Design

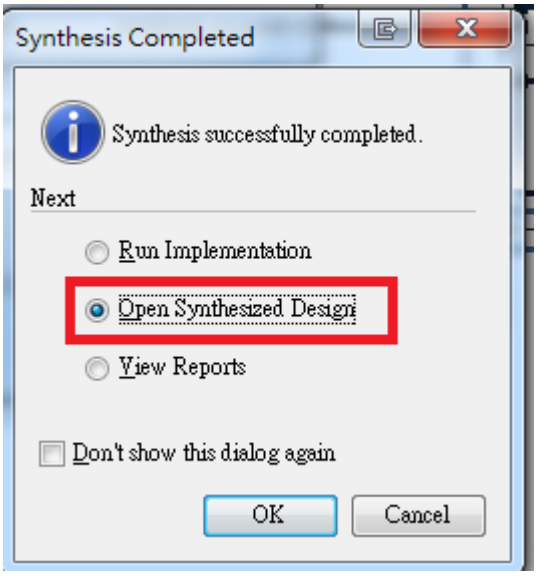

Figure 6

2.6 打開 Debug

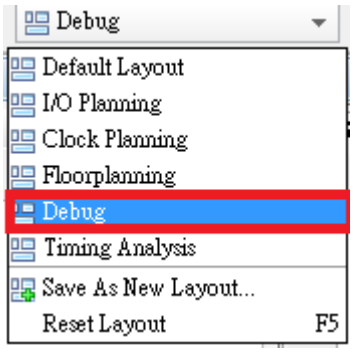

Figure 7

2.7 點選 Set Up Debug，並在跳出的視窗點選 Next，如 Figure 8 所示

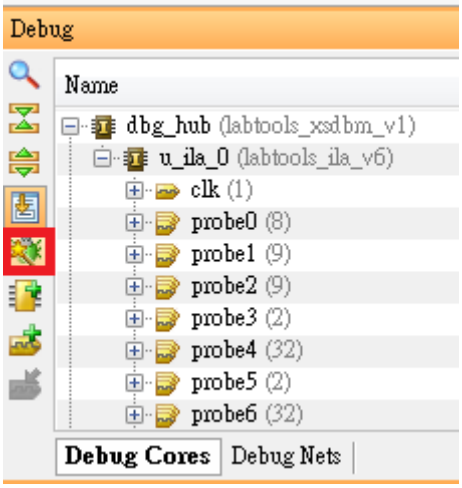

Figure 8

2.8 在 Figure 9 的視窗選擇第一個選項後，點選 Next

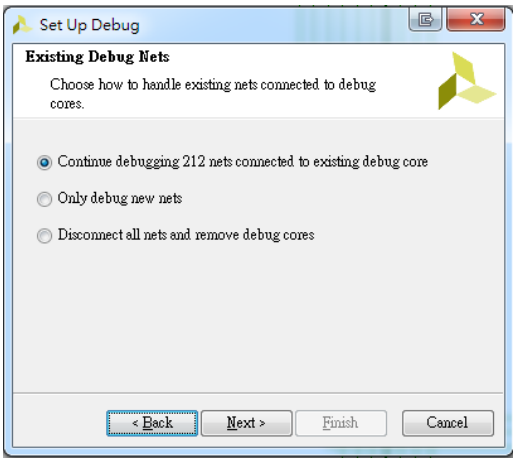

Figure 9

2.9 在新的視窗中檢查 Clock Domain 那欄是否有 undefined，如 Figure 10

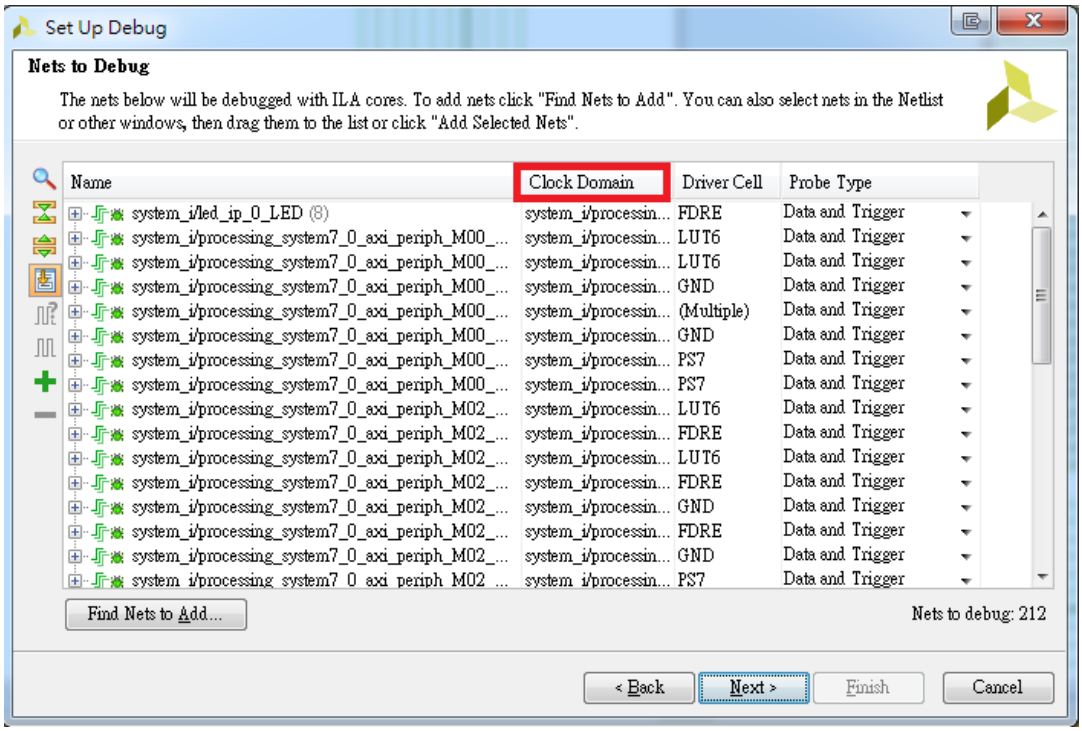

Figure 10

2.10 若有，點選該 net 並按右鍵選擇 Select Clock Domain…，如 Figure 11

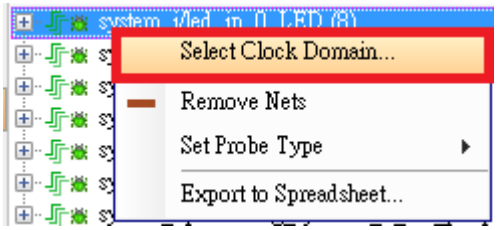

Figure 11

2.11 在跳出的視窗選擇 OK，在原 Figure 10 視窗點選 Next

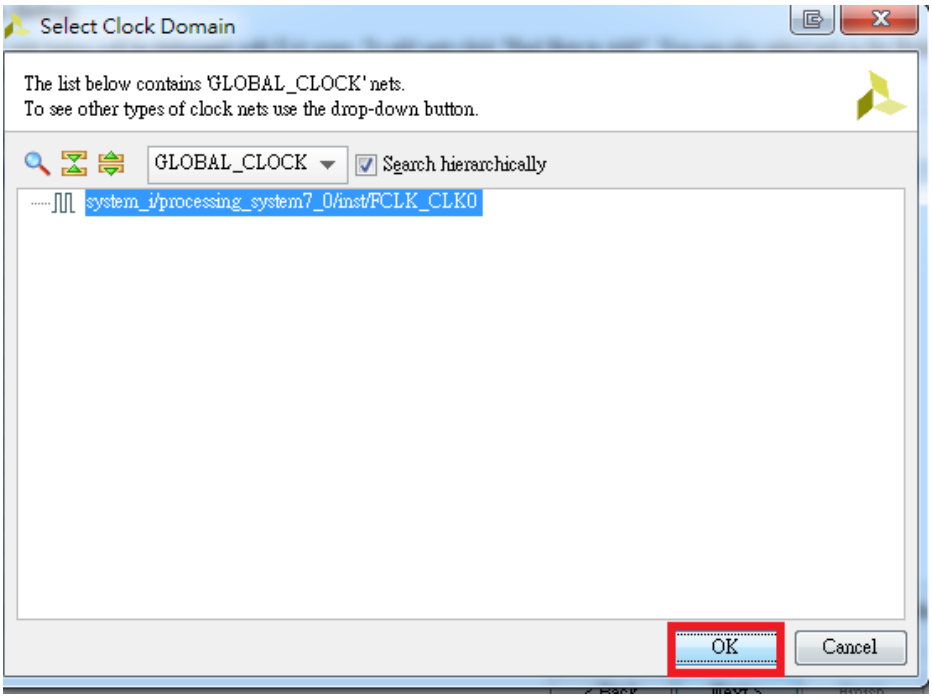

Figure 12

2.12 將 Capture control 和 Advanced trigger 打勾後點選 Next，在新視窗點選 Finish

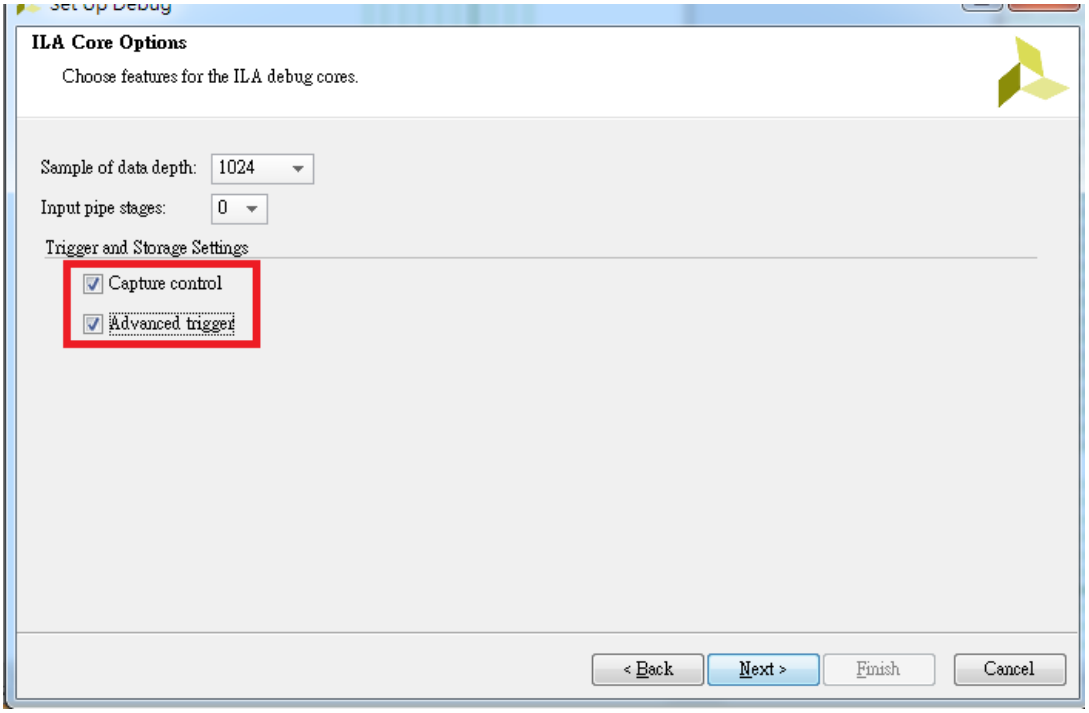

Figure 13

2.13 點選 Generate Bitstream

2.14 產生.bit 檔後，點選 File -> Export -> Export Hardware，在跳出的視窗勾選 Include bitstream 後點選 OK，如 Figure 14 所示。

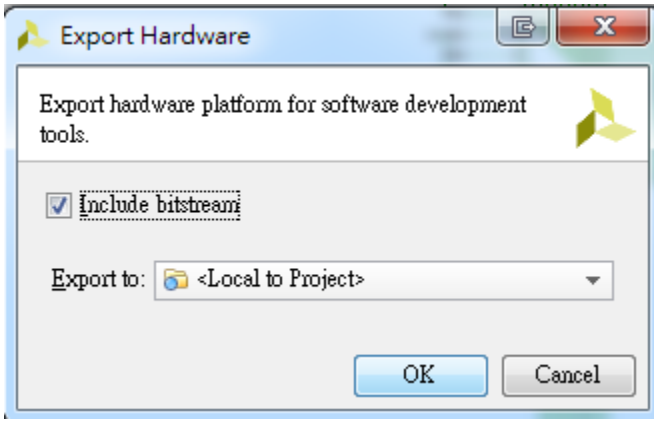

Figure 14

3 Software Debugging in SDK

3.1 點選 File -> Launch SDK，在跳出的視窗點選 OK

3.2 點選 File -> New -> Application Project，在視窗填入 lab6 並點選 Next，如 Figure 15

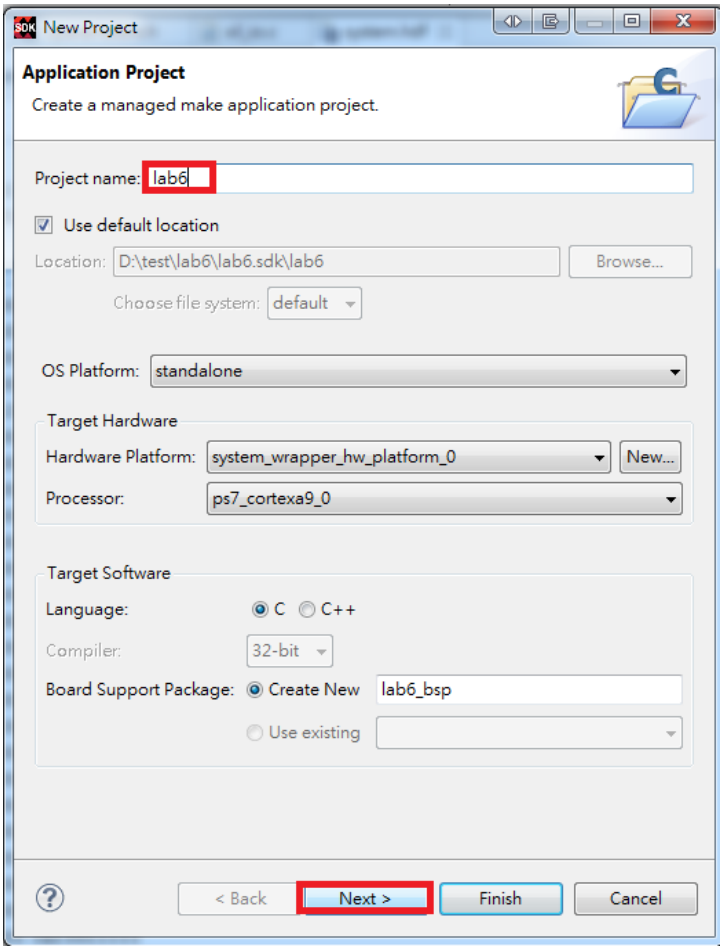

Figure 15

3.3 點選 Empty Application 再點選 Finish

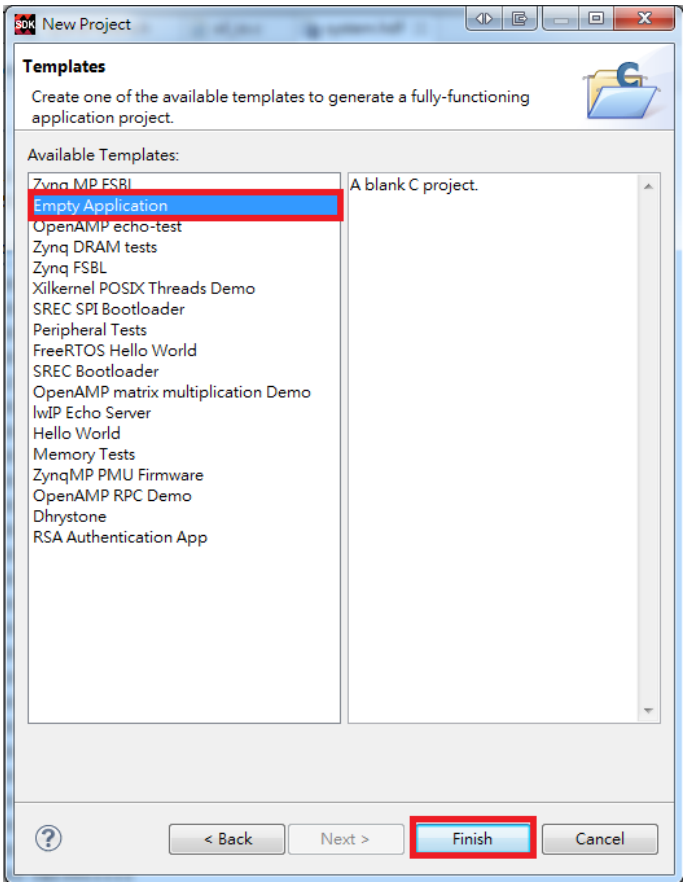

Figure 16

3.4 右健點選 lab6 下的 src 資料夾，選擇 Import，在跳出的視窗選擇 File System 再點選 Next

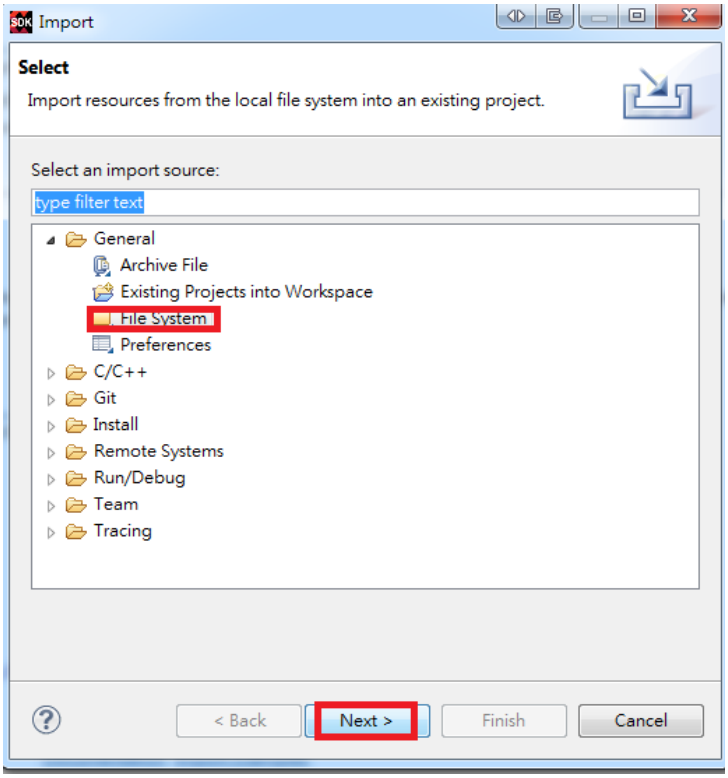

Figure 17

3.5 點選 Browse，選擇 2014_2zynq_sources/lab4，再點選確定

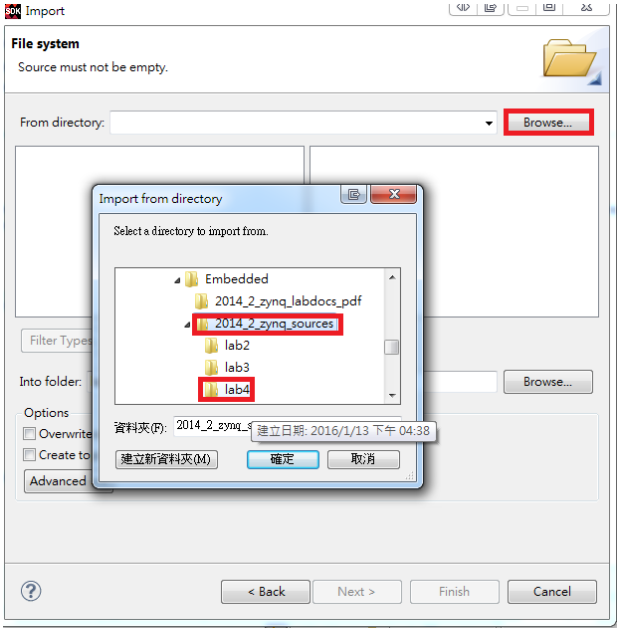

Figure 18

3.6 勾選 lab4_soln.c 後，點選 Finish

3.7 Xilinx Tools -> Program FPGA，點選 Program

3.8 將 for (i=0; i<9999999; i++)註解掉，如 Figure 19

```
        // output dip switches value on LED_ip device
        LED_IP_mWriteReg(XPAR_LED_IP_0_S_AXI_BASEADDR, 0, dip_check);

        //for (i=0; i<9999999; i++);
    }
}
```

Figure 19

3.9 右鍵點選 lab6，選擇 Debug As -> Launch on Hardware (GDB)，在跳出的視窗選擇 Yes

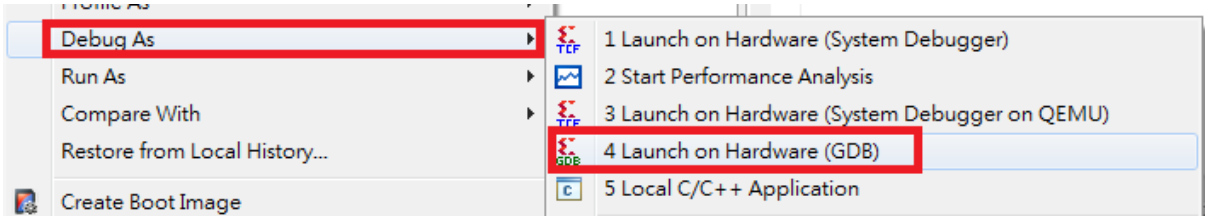

Figure 20

3.10 設定中斷點如 Figure 21

```
while (1)
{
    psb_check = XGpio_DiscreteRead(&push, 1);
    xil_printf("Push Buttons Status %x\r\n", psb_check);
    dip_check = XGpio_DiscreteRead(&dip, 1);
    xil_printf("DIP Switch Status %x\r\n", dip_check);

    // output dip switches value on LED_ip device
    LED_IP_mWriteReg(XPAR_LED_IP_0_S_AXI_BASEADDR, 0, dip_check);

    //for (i=0; i<9999999; i++);
}
```

Figure 21

3.11 新增 Memory 監測視窗如 Figure 22

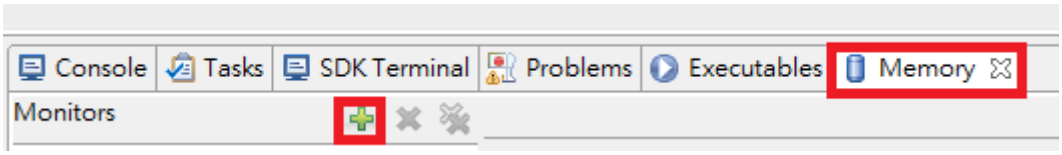

Figure 22

3.12 在新視窗填入 0x43C00000，再點選 OK

```
#define XPAR_LED_IP_0_DEVICE_ID 0
#define XPAR_LED_IP_0_S_AXI_BASEADDR 0x43C00000
#define XPAR_LED_IP_0_S_AXI_HIGHADDR 0x43C0FFFF
```

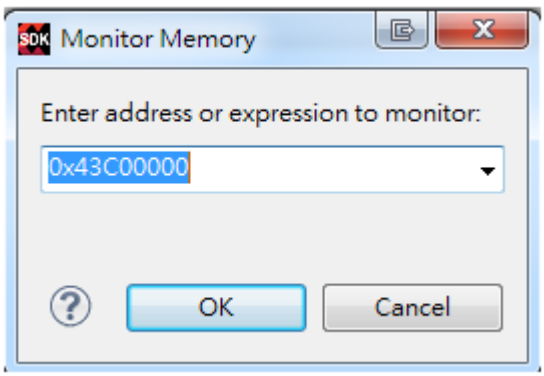

Figure 23

3.13 經由點選 和 並波動 switches，觀察參數值變化，如 Figure 24 和 Figure 25

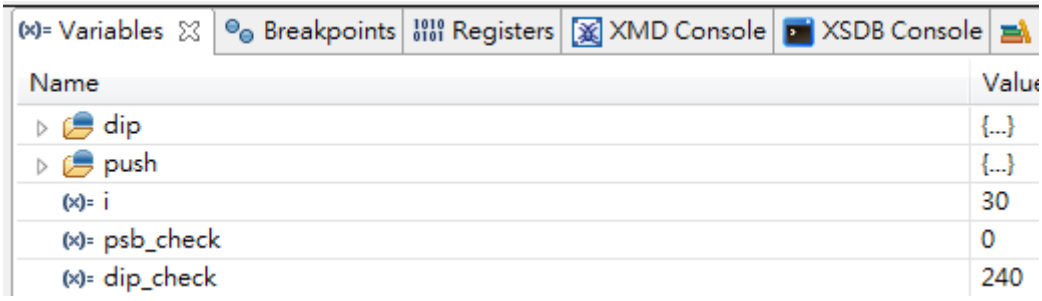

Figure 24

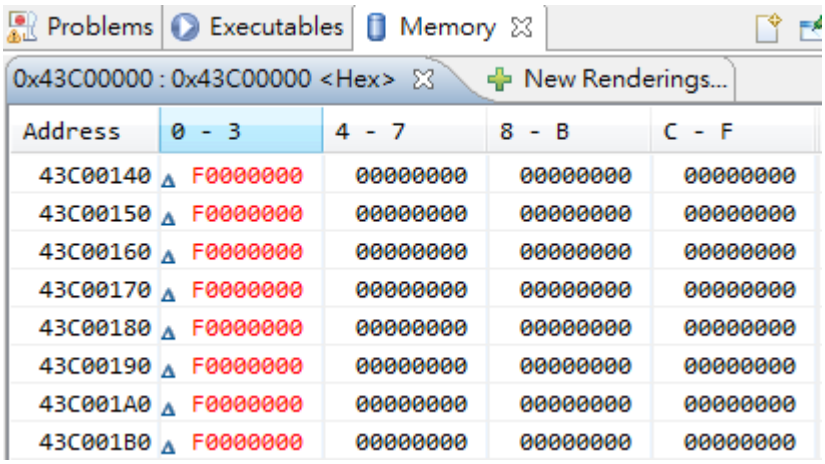

Figure 25

4 不關掉 SDK，回到 Vivado 並觀察各 debug 訊號的變化

4.1 點選 Open Hardware Manager，如 Figure 26

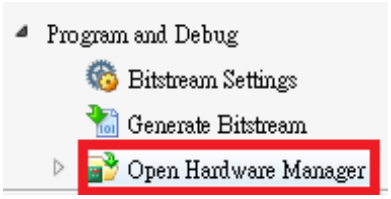

Figure 26

4.2 點選 Open target 如 Figure 27，再選擇 Open New Target

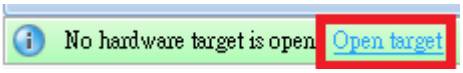

Figure 27

4.3 在跳出的視窗連點 3 次 Next 後，點選 Finish

4.4 點選 xc7z020_1 再點選  ，如 Figure 28，即可看到擷取到的波形於 Figure 29

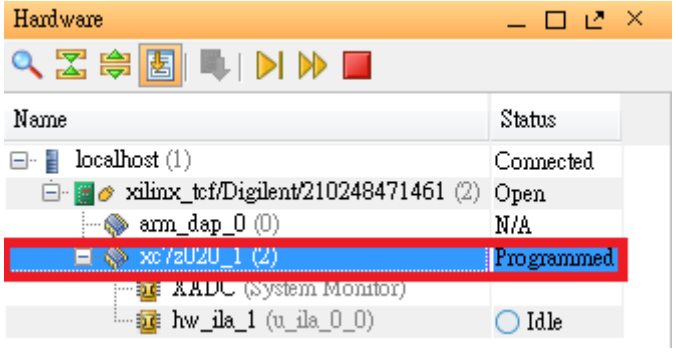

Figure 28

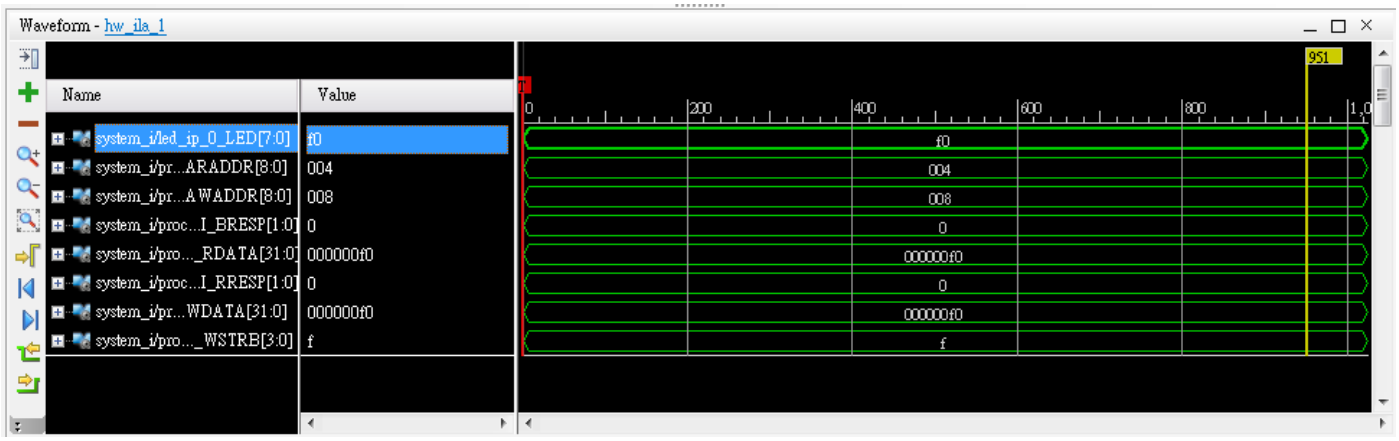

Figure 29

5 經由重複 3.13 和 4.4 即可同時進行 Software 和 Hardware Debugging