處理器設計與實作

CPU LAB

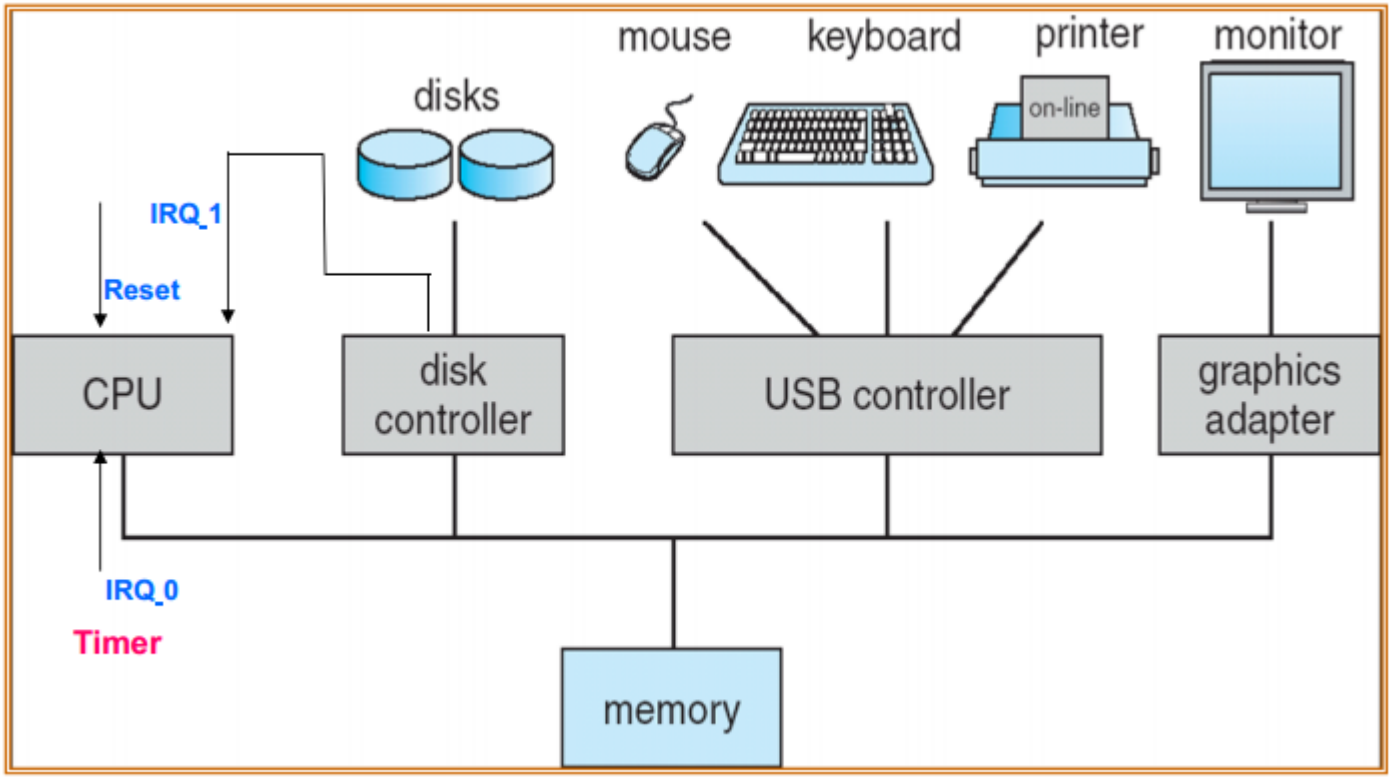
for Computer Organization

LAB 12: Interrupt Controller & Interrupt Service Routines on EASY Platform

實驗目的

1. 簡介ARM Interrupt Service Routine
2. 學習使用系統上的Interrupt Controller
3. 將平台透過Vivado合成，並在FPGA板驗證Interrupt Service Routine

Interrupts by External Signals



ARM中斷流程

- ARM微處理器有7種運行模式:
  - **User mode**(usr)：ARM處理器正常的程序執行狀態
  - **System mode**(sys)：運行具有特權的操作系統任務
  - **Supervisor mode**(svc)：操作系統使用的保護模式
  - **Abort mode**(abt)：當抓取的資料或指令錯誤時進入該模式
  - **Interrupt mode**(irq)：用於通用的中斷處理
  - **Fast Interrupt mode**(frq)：用於快速資料傳輸或通道處理
  - **Undefined mode**(und)：遇到未定義的指令執行時進入該模式

- ARM的外部硬體中斷訊號由IRQ(Interrupt Request)及FIQ(Fast Interrupt Request)兩支訊號腳來控制
  - ➢ 當IRQ或FIQ被觸發時，則ARM CPU將會強制的進入IRQ或FIQ模式 (if interrupt enabled)

ARM interrupt vector table

- ARM interrupt vector table(0x00000000~0x0000001F)

address	Vector jump to
0x00000000	Reset Interrupt Vector
0x00000004	Undefined Interrupt Vector
0x00000008	Software Interrupt Vector：Trap
0x0000000c	Prefetch Abort Interrupt Vector
0x00000010	Data Abort Interrupt Vector
0x00000014	reserved
0x00000018	Interrupt Request Vector
0x0000001c	Fast Interrupt Request Vector

ARM中斷流程

- 當IRQ引發後:

  (1)跳到 PC = 0x00000018的位址 (normal address)

  (2)At 0x18, usually a jump-like instruction to the ISR.

Exception types	Mode	Normal address	High vector address
Reset	Supervisor	0x00000000	0xFFFF0000
Undefined instructions	Undefined	0x00000004	0xFFFF0004
Software interrupt (SWI)	Supervisor	0x00000008	0xFFFF0008
Instruction prefetch abort	Abort	0x0000000C	0xFFFF000C
Data access abort	Abort	0x00000010	0xFFFF0010
IRQ (Interrupt)	IRQ	0x00000018	0xFFFF0018
FIQ (Fast interrupt)	FIQ	0x0000001C	0xFFFF001C

系統上的Interrupts

- 在一般的電腦系統裡，當device需要系統服務時，有二種方法：

1. Polling :

   最簡單的方式讓I/O device與CPU溝通，I/O device只要將information放進status register，CPU會周期性的檢查status register來得知需要服務的device

   如果polling 太頻繁會很浪費CPU 的時間

2. Interrupt-driven I/O ：

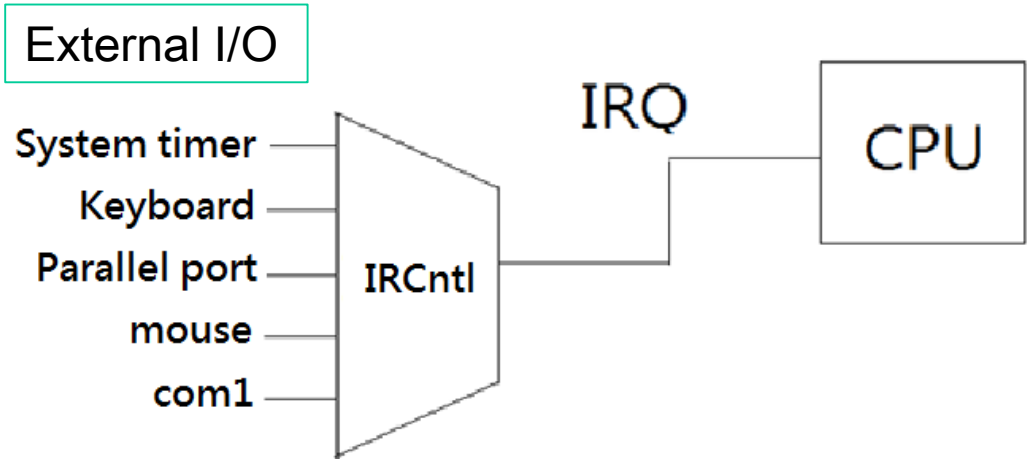
   當Device需要服務時就發出IRQ，當系統收到這個IRQ訊號時才去服務它，這樣可大大減小系統的負擔。

Interrupt controller

- IRQ是由Interrupt controller所處理的，Interrupt controller是連接Device和CPU的重要橋梁，一個Device產生中斷後，需經過中斷控制器的轉發，訊號才能到達CPU。

✓ 主要提供以下功能：

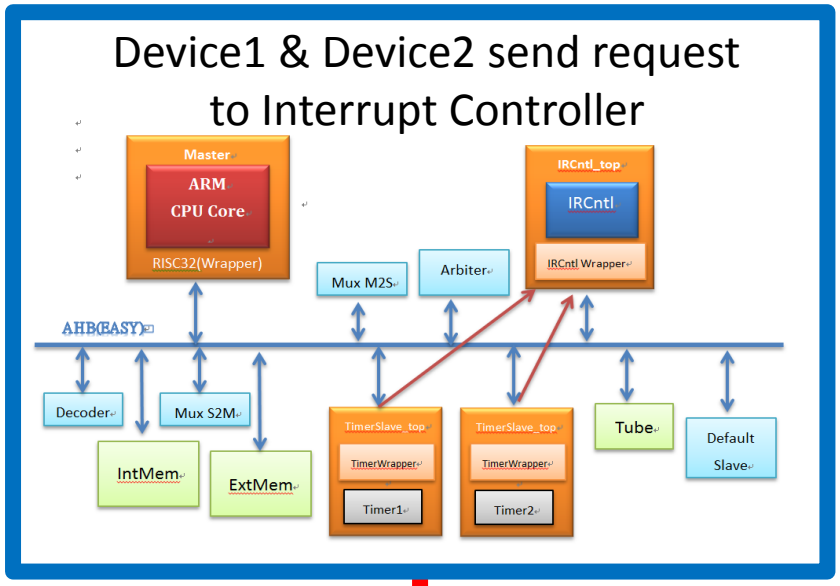
1. Handle multiple interrupts
2. 支援Asynchronous Interrupt
3. 確定引起 interrupt 的裝置
4. 提供 multilevel interrupt的機制 (Priority)
5. 決定每個 interrupt 所對應的 handler (處理程式)

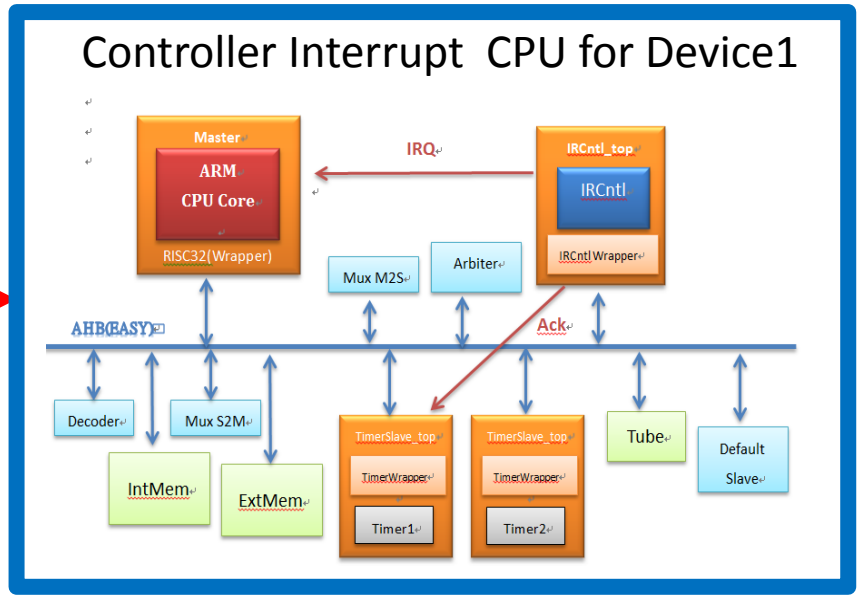


Hardware Interrupt 基本的流程 /8

Step	說明
Device 發出中斷到 Interrupt Controller	Interrupt Controller接收Device中斷 Controller會知道是那個Device發出的中斷及此 Device中斷的優先權。
Controller 發出IRQ到 CPU Controller Ack回Device	Interrupt Controller依據所有發出中斷的Device 的優先權決定誰可以中斷CPU，代替該Device 發出IRQ給CPU，並回應該Device IRQ Ack，表 示Device的中斷已經被傳達。
CPU被Interrupt後，跳到 ISR去後去執行該Device 的Interrupt Handler	CPU在ISR中會檢視Interrupt Controller內部紀錄 是那個Device所發出的IRQ，並處理該Device的 Interrupt Handler。

Hardware Interrupt 基本的流程 /8

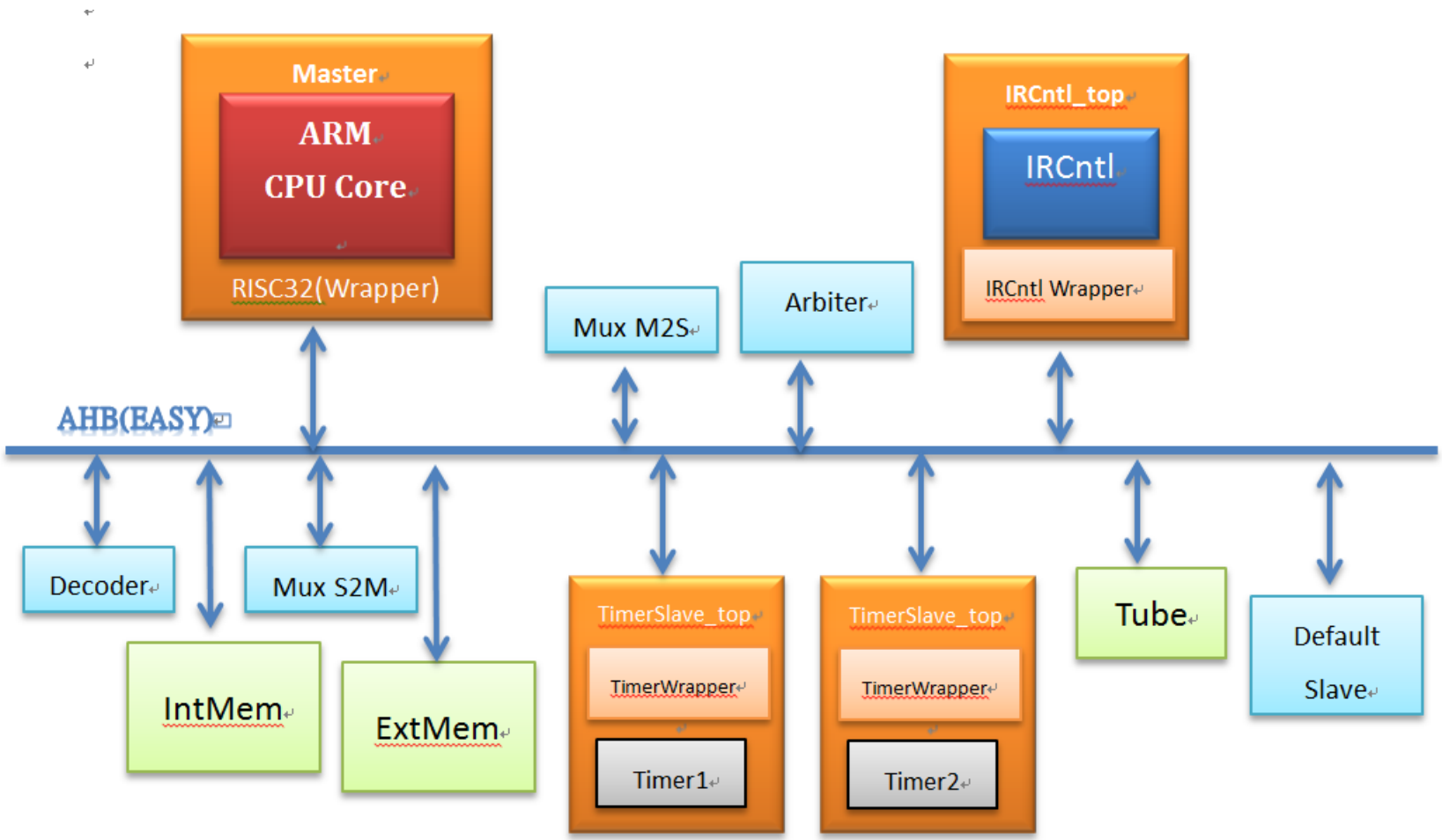




實驗系統架構

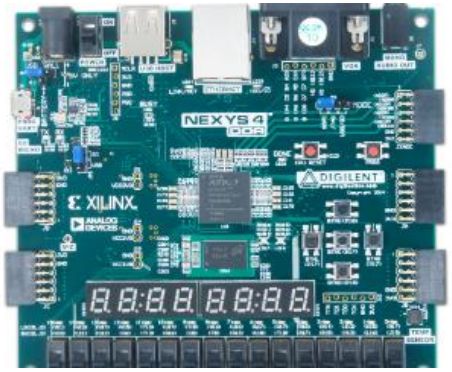
Example AMBA SYstem (EASY)

EASY平台(ARM processor)



實驗環境

- Tool used :
  1. ARM DS-5
     - 利用DS-5的ARM Cross Compiler將C轉成arm assembly和machine code
  2. Mentor Modelsim
     - 看平台上的wave來驗證CPU和AHB的行為
  3. Vivado 2014.4
     - 將驗證過後的平台透過Vivado合成成bit檔
  4. Nexys 4 DDR board
     - 利用此驗證版將平台燒錄至FPGA中並使用板子上的I/O驗證行為



實作(一)

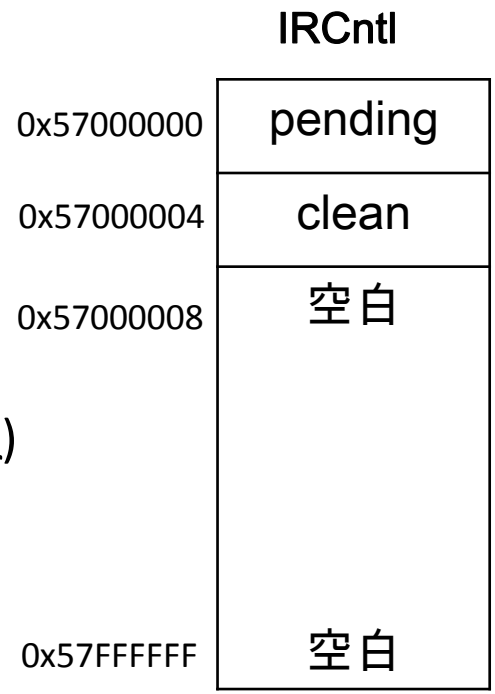
Interrupt Controller

實作(一)

- 請同學在EASY上新增一個Interrupt I/O，讓Interrupt Controller 可以去收到各種Device的Interrupt，並發出IRQ去中斷CPU

1. 我們的Interrupt Controller的運作機制簡介:

右圖是IRCntl的Memory Mapping位置，它有兩個32-bit的register存放pending和clean的值在以右圖對應的位置 。



pending: 當有IO發出中斷時，會在pending這裡紀錄目前是將那個IO發出的Interrupt(IRQ)送到CPU。

clean : CPU收到IRQ之後，去看pending來處理該IO中斷，CPU處理完後寫和pending相同的值到clean代表處理完畢，並且會把pending清空。

實作(一)

2. 當 I/O device發出的interrupt 成功被IRCntl送出IRQ之後，IRCntl會給該device訊號Ack=1，告知該device已經正在處理它的事情，可以把interrupt訊號拉下來了，當處理完畢後，會給device Ack = 0 。

```
always@(posedge (nCLK))
  begin
    if(~nIRQ)
    AckTemp = pending[15:0] ;
    else
    AckTemp = 16'h0000;
  end

assign HIRQAck = AckTemp ;
```

IRCntl 有16 bits input腳位(HIRQSource)，可以接16個Interrupt device的interrupt。

而 pending register也設為16 bits(對應到那16個 input 腳)，所以當HIRQSource有很多bits 為1(同時有很多不同的interrupt來源)時，pending register只會有1個bits會等於1。

而IRCntl Ack output(HIRQAck)也是16 bits的排線接回16個device，所以左圖就表示讓目前被pending的device收到Ack =1。

3. 目前平台上有Timer1和Timer2的會倒數一定的時間後發出timer_irt到IRCntl中去仲裁看哪個Timer被pending。

實作(一)

4. 請在EASY平台新增Timer的IRQ 和ACK線

```
//------------------------------
// Interrupt Signals
//------------------------------

wire          Timer1_ACK;
wire          Timer2_ACK;

wire          Timer1_IRQ;
wire          Timer2_IRQ;
```

5. 並將Timer1和Timer2的訊號接到IRCntl的HIRQSource和HIRQAck腳位

注意：

因為IRCntl是以Fixed priority去仲裁(HIRQSource[15] > HIRQSource [14] > HIRQSource[13].....)

請將Timer1接到 HIRQSource[15]、Timer2接到HIRQSource[14]。
HIRQAck 一樣接到對應位置。

Hint:
{ , ,{12'h000},{2'b00} }

```
TimersSlave_top  Timer1(
      //AHB Bus Input
      .HCLK        (HCLK),
      .HRESET_n    (HRESETn),
      .HADDRS      (HADDR),
      .HTRANSS     (HTRANS),
      .HWRITES     (HWRITE),
      .HSIZES      (HSIZE),
      .HWDATAS      (HWDATA),
      .HSELS       (HSEL_Timer1),
      .HREADYS     (HREADY),
      //AHB Bus Output
      .HRDATAS     (HRDATA_Timer1),
      .HREADYOUTS  (HREADY_Timer1),
      .HRESPS      (HRESP_Timer1),
      //From IRCntl to IP
      .ACK     (   ),
      //From IP to IRCntl
      .IRQ         (   )
      );
```

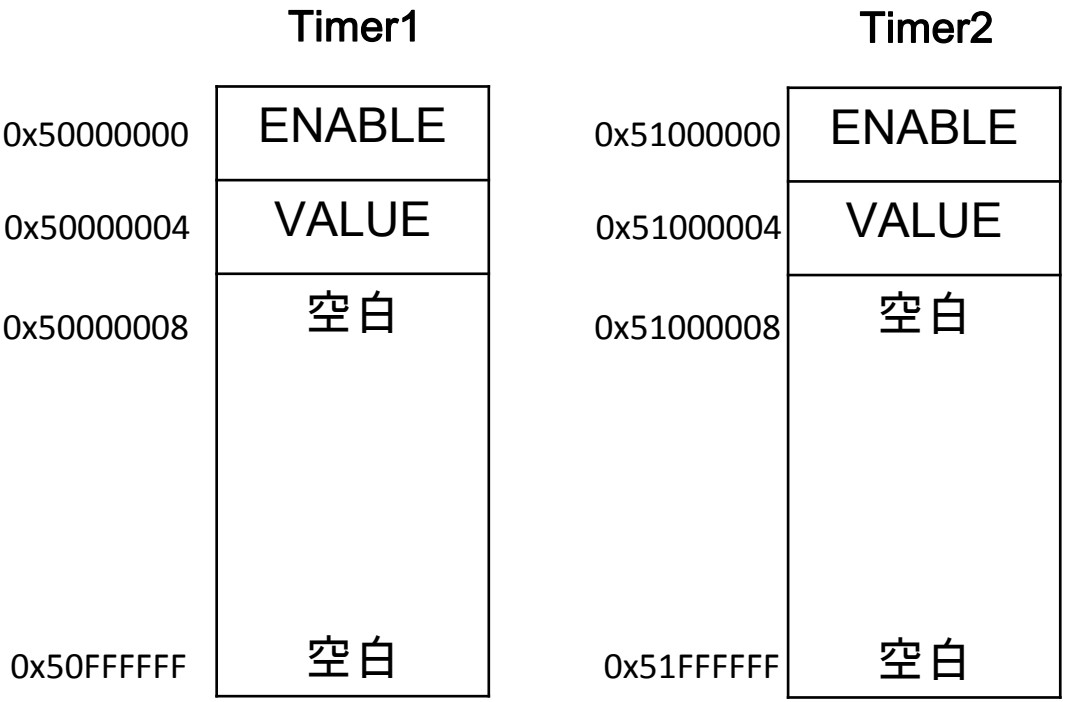
```
IRCntlSlave_top   IRCntl(
  // Input signal from System

      //AHB Bus Input
      .HCLK        (HCLK),
      .HRESET_n    (HRESETn),
      .HADDRS      (HADDR),
      .HTRANSS     (HTRANS),
      .HWRITES     (HWRITE),
      .HSIZES      (HSIZE),
      .HWDATAS     (HWDATA),
      .HSELS       (HSEL_IRCntl),
      .HREADYS     (HREADY),
      //AHB Bus Output
      .HRDATAS     (HRDATA_IRCntl),
      .HREADYOUTS  (HREADY_IRCntl),
      .HRESPS      (HRESP_IRCntl),

      //From Device to IP
      .HIRQSource (   ),
      //From IP to Device
      .HIRQAck     (   ),

      //From IP to Core
      .nIRQ        (   )
      );
```

實作(一)

6. 接好之後在Lab3_1->testcode->C->main_function.c

設定Timer1和Timer2的ENABLE和VALUE的值， mapping位置如下

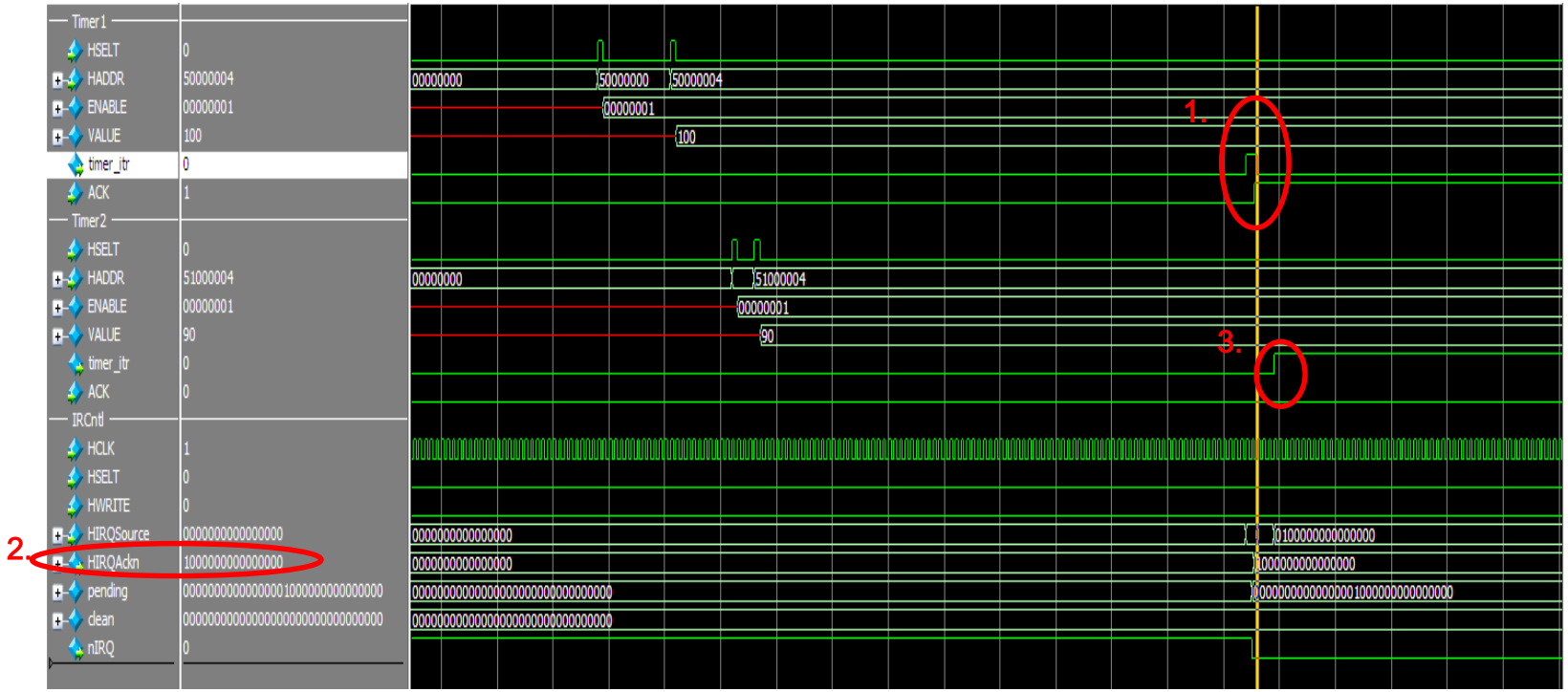
```
volatile int*  Timer1 ;
volatile int*  Timer2 ;

Timer1 =  (int*) 0x50000000;
Timer1[0] = 1;
Timer1[1] = 100 ;

Timer2 =  (int*) 0x51000000;
Timer2[0] = 1;
Timer2[1] = 90 ;
```



ㄇㄛㄊㄜ:記得編譯出testcode. txt 並複製到”xxxx/Lab3_1/testcode”

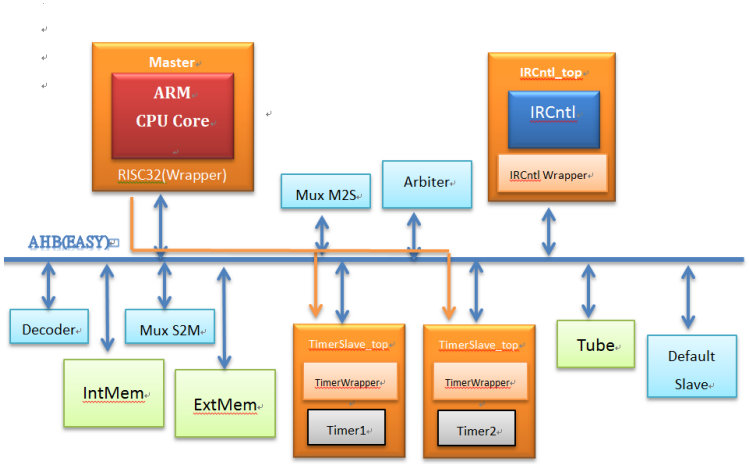
實作(一)

7. 之後跑波型可看到Timer1一發出timer_irt就寫到pending，ACK被拉起來後，Timer1的timer_irt降下來，而Timer2的timer_irt會一直拉著等待Timer1的IRQ做完才會換處理它。

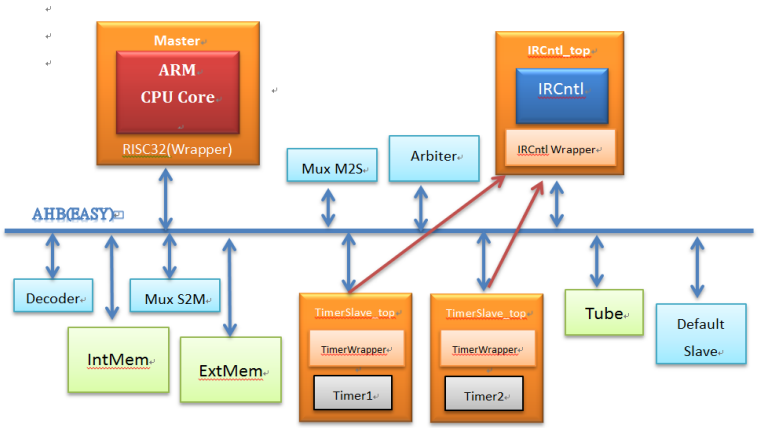


實作(一)

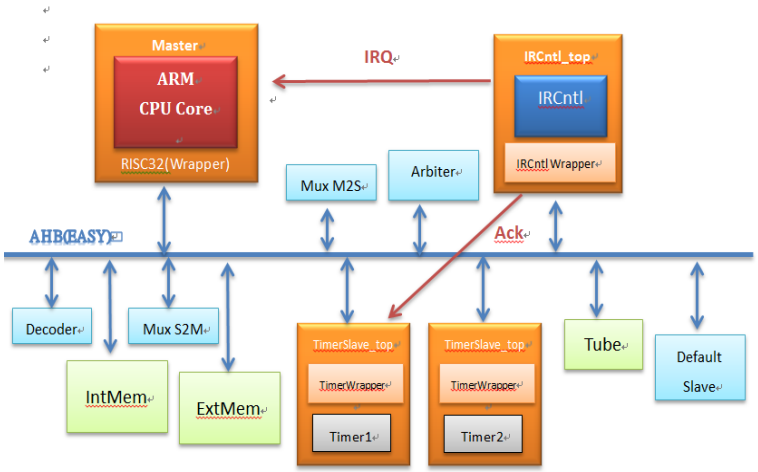
1. CPU enable Timer1、Timer2



2. Timer1 & Timer2 request to IRCntl



3. IRCntl Interrupt CPU for Timer1



實作(二)

Interrupt Service Routine

實作(二)

- 請同學新增一個來自FPGA板上的Switch Interrupt，並練習寫一個簡單的Interrupt Service Routine(ISR function)

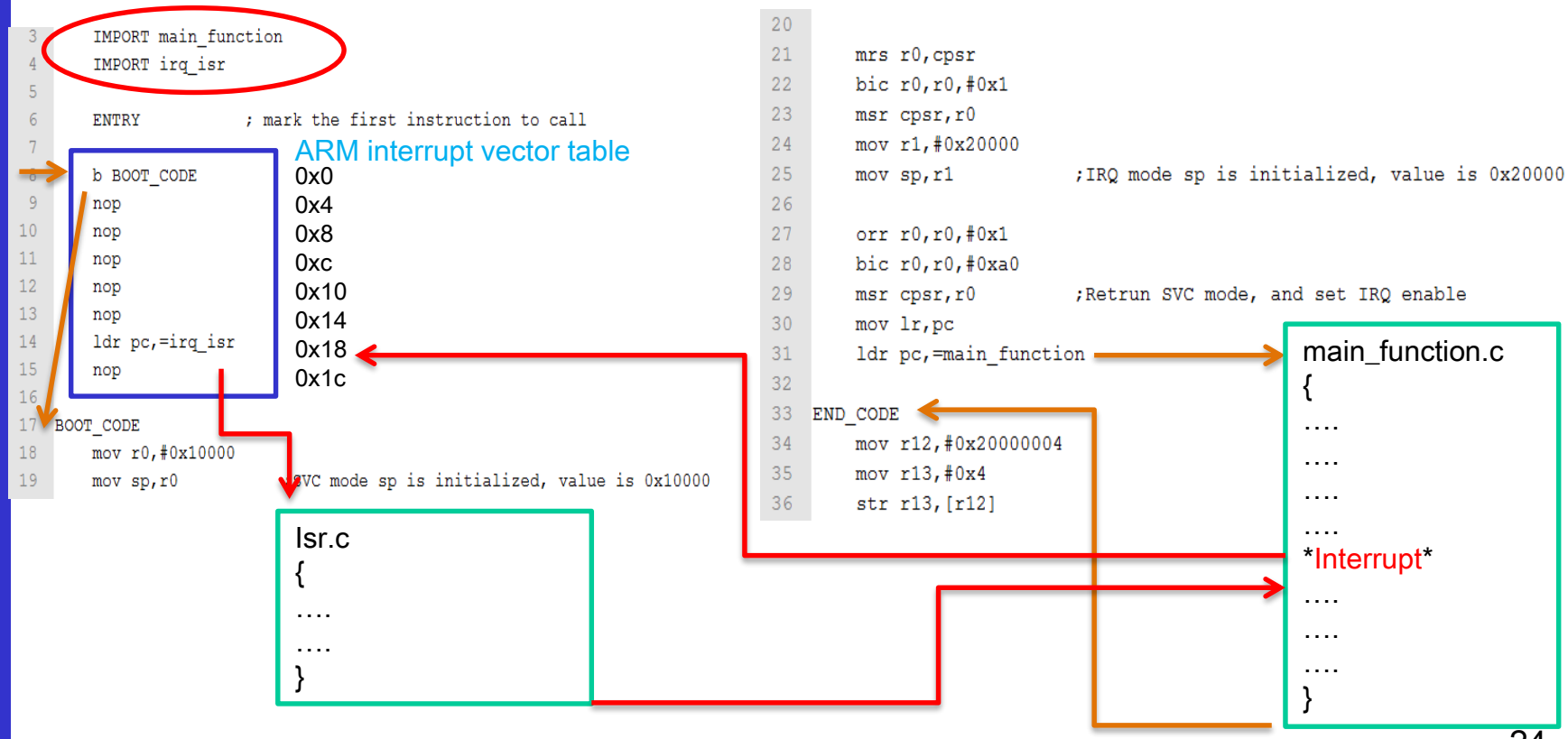
1. 首先請在EASY.v新增 input SW1_IRQ
   output LED1_ACK
2. 並把這兩條線接到IRCntl的HIRQSource和HIRQAck的top bit

```
module EASY (
    HCLK,
    HRESETn,
    SW1_IRQ,
    LED1_ACK
);

  input         HCLK;
  input         HRESETn;
  input         SW1_IRQ;
  output        LED1_ACK;
 // wire                nFIQ;
  wire              nIRQ;
```

```
IRCntlSlave_top   IRCntl(
  // Input signal from System

    //AHB Bus Input
    .HCLK        (HCLK),
    .HRESET_n    (HRESETn),
    .HADDRS      (HADDR),
    .HTRANSS     (HTRANS),
    .HWRITES     (HWRITE),
    .HSIZES      (HSIZE),
    .HWDATAS     (HWDATA),
    .HSELS       (HSEL_IRCntl),
    .HREADYS     (HREADY),
    //AHB Bus Output
    .HRDATAS     (HRDATA_IRCntl),
    .HREADYOUTS  (HREADY_IRCntl),
    .HRESPS      (HRESP_IRCntl),

    //From Device to IRCntl
    .HIRQSource ( {      ,Timer1_IRQ,Timer2_IRQ,{12'h000},{1'b0} } ),
    //From IRCntl to Device
    .HIRQAck     ({      ,Timer1_ACK,Timer2_ACK}),

    //From IRCntl to Core
    .nIRQ        (nIRQ)
    );
```

實作(二)

3.請同學打開Lab3_2->testcode->init.s，這是testcode.txt的系統程式，透過init.s把同學之前寫過的main_function.c和isr.c link起來。



實作(二)

目前的Interrupt controller共有接三個interrupt source:

SW_IRQ、Timer1_IRQ、Timer2_IRQ

4. 打開Lab3_2->testcode-> C -> isr.c 完成以空白的部分。

```
void __irq irq_isr(void)
{
  volatile int *tube =  (int*) 0x20000000;
  volatile int *IRCntl =  (int*) 0x57000000;

  tube[0] = 'I' ;
  tube[0] = 'R' ;
  tube[0] = 'Q' ;
  tube[0] = '\n' ;


   if(IRCntl[0]==0x8000)         //若是SW發出的intterrupt，請讓Tube印出 "*LED"
        {

        }
   else if(IRCntl[0]==0x4000)    //若是Timer1發出的intterrupt，請讓Tube印出"Ter1"
        {

        }
   else if(IRCntl[0]==0x2000)    //若是Timer2發出的intterrupt，請讓Tube印出"Ter2"
        {

        }


  //請讓clean被寫入pending的值，再對pending寫入0x0
  IRCntl[1] =
  IRCntl[0] =


}
```

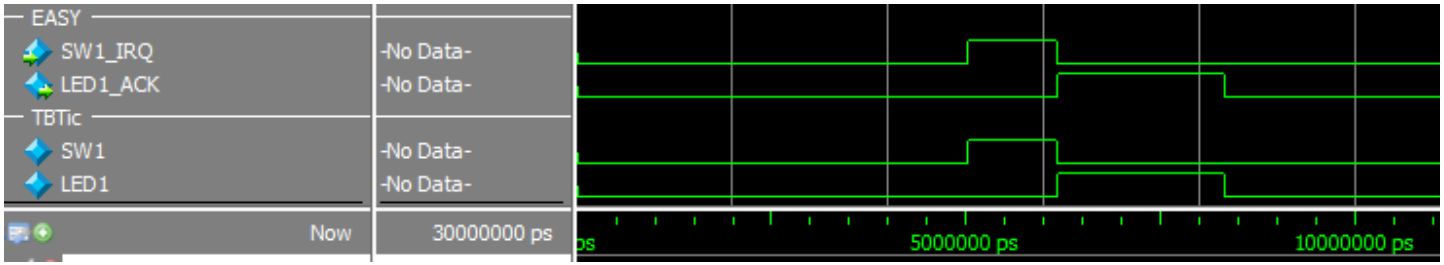
IRCntl

0x57000000	pending
0x57000004	clean
0x57000008	空白
0x57FFFFFF	空白

實作(二)

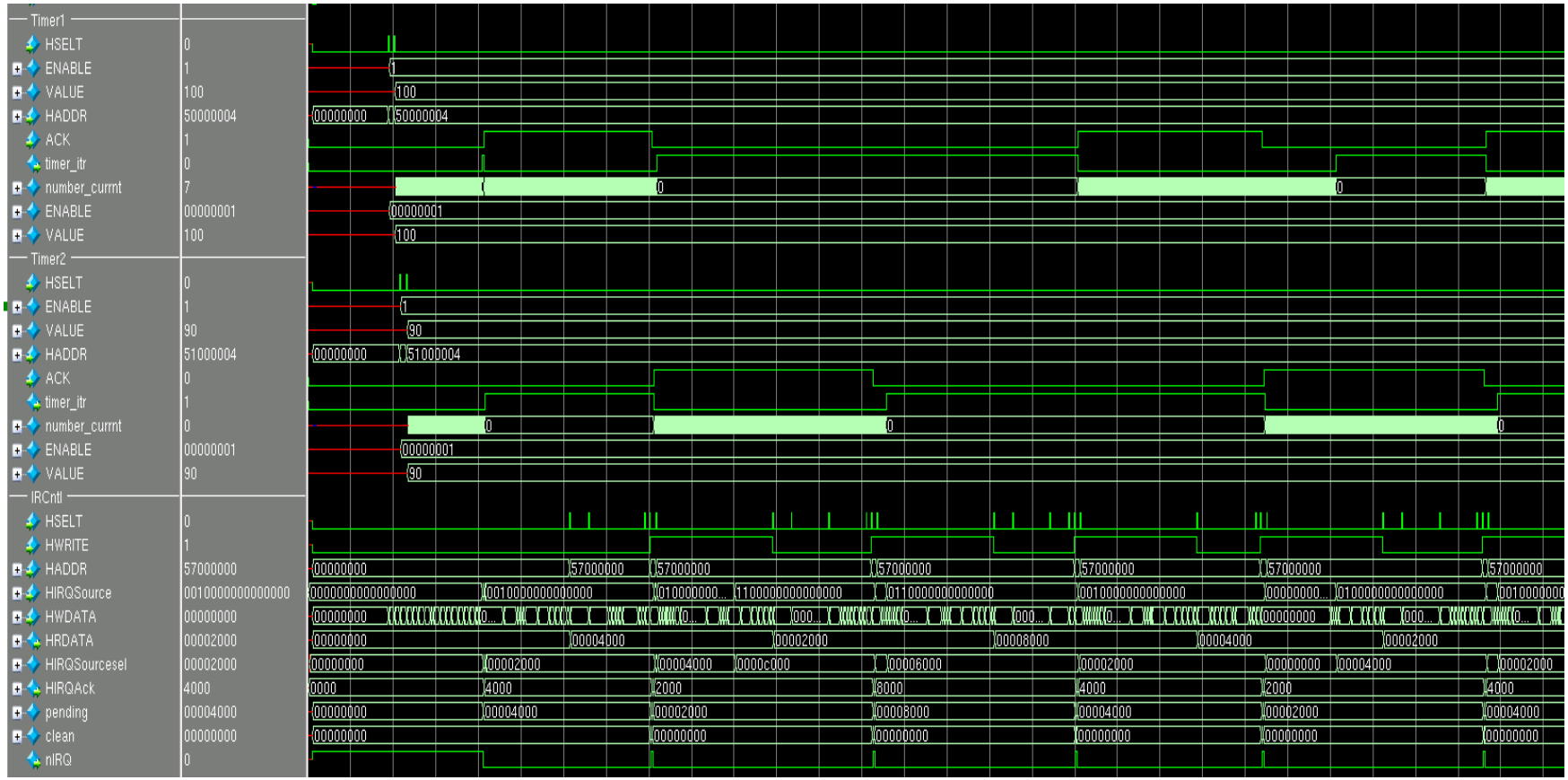
5. 在EASY新增的input(SW1_IRQ)要用Modelsim模擬，需讓EASY的testbench(TBTic)在某個時間點給SW_IRQ=1到EASY平台，我們這邊給同學的TBTic讓SW1_IRQ input在5us時等於1，成功的話就會看到下圖的結果。

```
VSIM 77> run
### Loading internal memory, Based Addr = 0x00000000, Length = 0x04000000 ###
### Load internal memory Complete ###
### Loading External memory, Based Addr = 0x40000000, Length = 0x00008000 ###
### Load internal memory Complete ###
# TUBE: Hello    EASY!
# TUBE: IRQ
# TUBE: Ter1
# TUBE: IRQ
# TUBE: Ter2
# TUBE: IRQ
# TUBE: IRQ
# TUBE: *LED
# TUBE: IRQ
# TUBE: Ter2
# TUBE: IRQ
# TUBE: Ter1
# TUBE: IRQ
# TUBE: Ter2
# TUBE: IRQ
# TUBE: Ter1
# TUBE: IRQ
# TUBE: Ter2
# TUBE: IRQ
```



實作(二)

6. 請同學在View->wave->file->Load->打開lab3-2.do的波形檔，觀察Timer1、Timer2和IRCntl之間的關係。

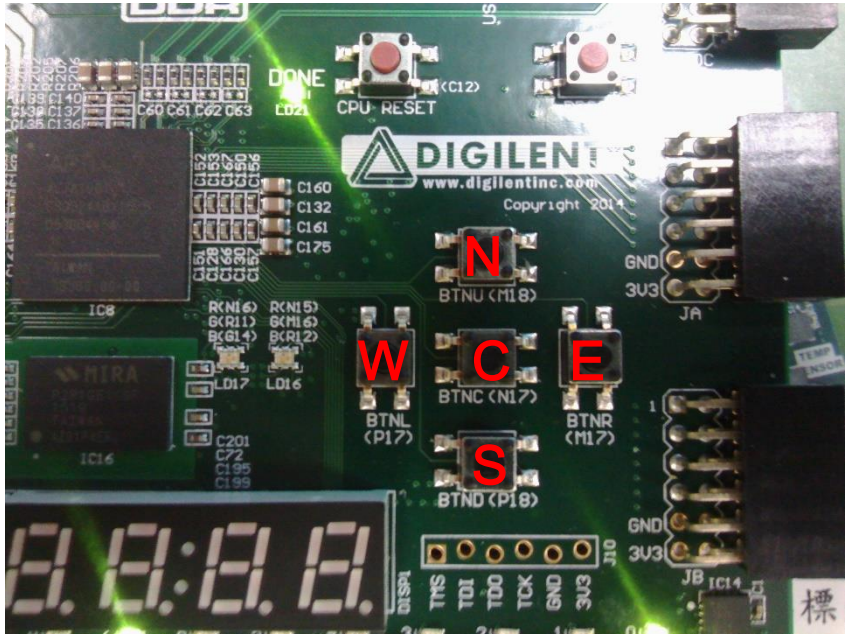


實作(三)－不計分

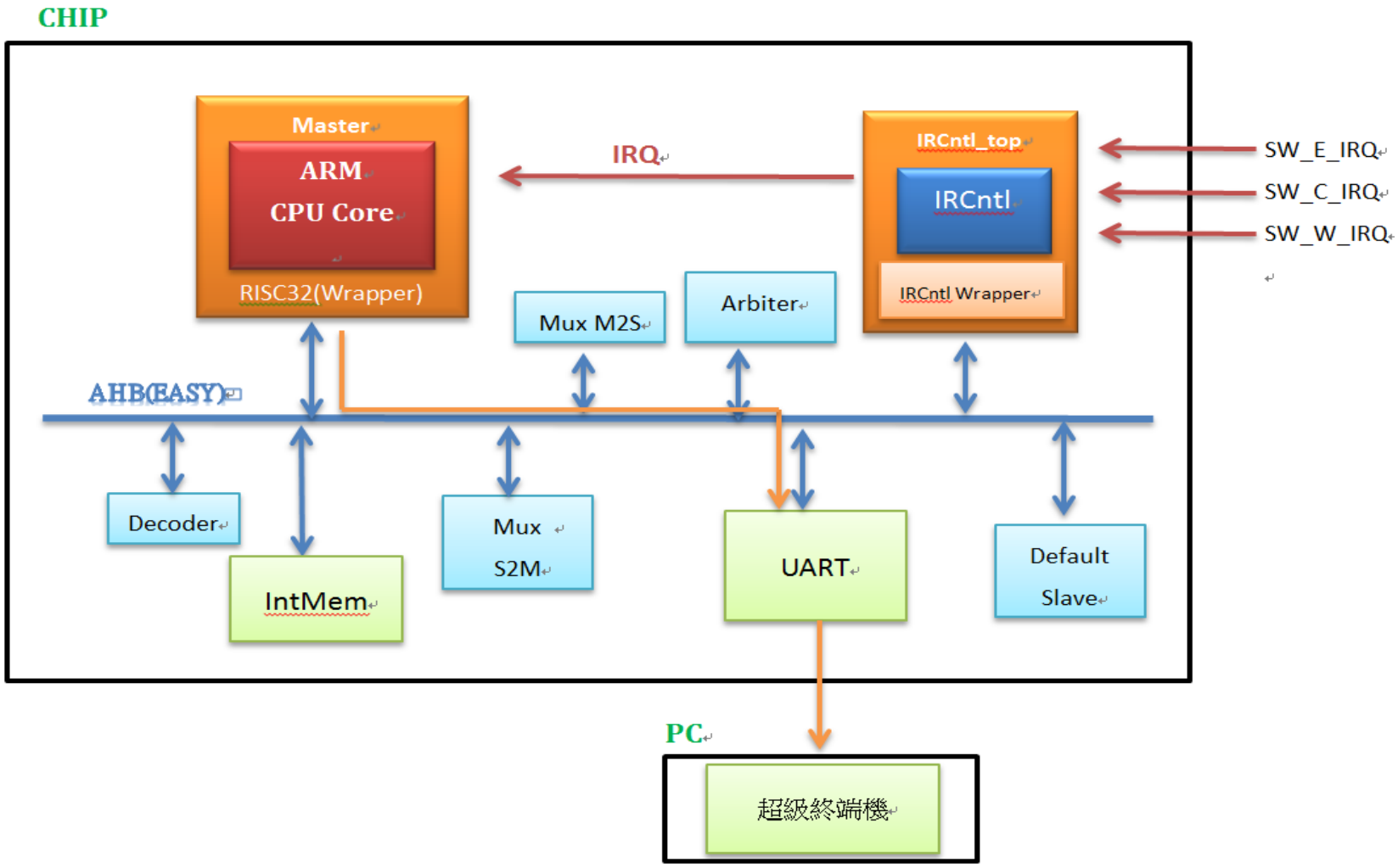
FPGA Verification

實作(三)—不計分

- 請同學依照實作(二)的方式，為EASY平台增加FPGA板上3個Switch 的Interrupt，並為他們寫簡單的ISR，然後利用Vivado合成電路並燒到FPGA板子驗證。
- 讓CPU可以透過UART回傳中斷訊號到電腦的超級終端機中，顯示是那個Switch發出的中斷。



實作(三)－不計分


CHIP
Master
ARM
CPU Core
RISC32(Wrapper)
IRQ
IRCntl_top
IRCntl
IRCntl Wrapper
SW_E_IRQ
SW_C_IRQ
SW_W_IRQ
Mux M2S
Arbiter
AHB(EASY)
Decoder
IntMem
Mux
S2M
UART
Default
Slave
PC
超級終端機

實作(三)－不計分

1. 開啟桌面的Vivado 2014.4

   File -> Open Project

   開啟SYSLab3 -> vivado -> lab10.xpr

2. 為EASY.v新增Interrupt input 到IRCntl

3. 請修改CrossCompiler-> main_function

   將轉好的testcode.txt利用2coe.exe

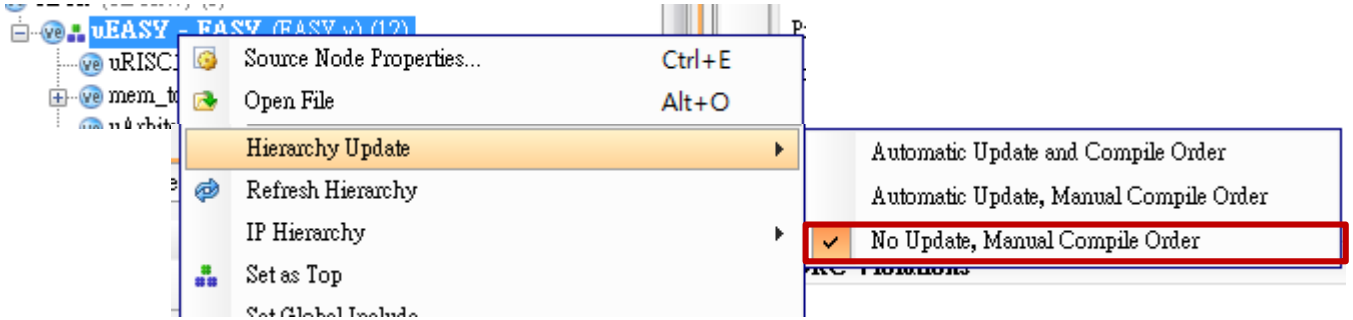
   再轉成lab.coe

```
module EASY (
    HCLK,
    HRESETn,
    uart_rst,
    lock,
    tx,
    rx,
    alarm,
    LED_7,

    SW_C_IRQ,
    SW_E_IRQ,
    SW_W_IRQ,
    LED_C_ACK,
    LED_E_ACK,
    LED_W_ACK
);

  input          HCLK;
  input          HRESETn;
  input      uart_rst ;
  input          rx;
  output         lock;
  output         tx;
  output         LED_7;
  output         alarm;

  input      SW_C_IRQ;
  input      SW_E_IRQ;
  input      SW_W_IRQ;

  output     LED_C_ACK;
  output     LED_E_ACK;
  output     LED_W_ACK;
```

```
   IRCntlSlave_top IRCntl(
// Input signal from System

      //AHB Bus Input
      .HCLK      (clk55MHz),
      .HRESET_n  (HRESETn),
      .HADDRS        (HADDR),
      .HTRANSS   (HTRANS),
      .HWRITES   (HWRITE),
      .HSIZES        (HSIZE),
      .HWDATAS   (HWDATA),
      .HSELS     (HSEL_IRCntl),
      .HREADYS   (HREADY),
      //AHB Bus Output
      .HRDATAS   (HRDATA_IRCntl),
      .HREADYOUTS    (HREADY_IRCntl),
      .HRESPS        (HRESP_IRCntl),

      //From Device to IP
     .HIRQSource ( {SW_C_IRQ,SW_E_IRQ,SW_W_IRQ,13'b0} ),
      //From IP to Device
     .HIRQAck ({LED_C_ACK,LED_E_ACK,LED_W_ACK}),

     //From IP to Core
     .nIRQ     (nIRQ)
      );
```

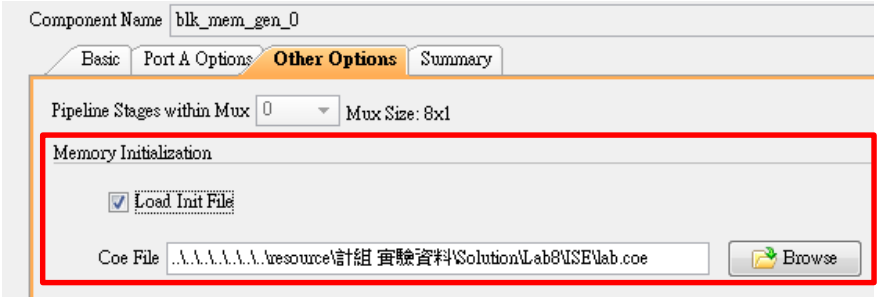
實作(三)－不計分

4. 將HierarchyUpdate設置為“NO Update, Manual Compile Order”以便接下來的編譯步驟能順利進行。



5. 如同之前lab，將步驟三產生的lab.coe載入記憶體中。

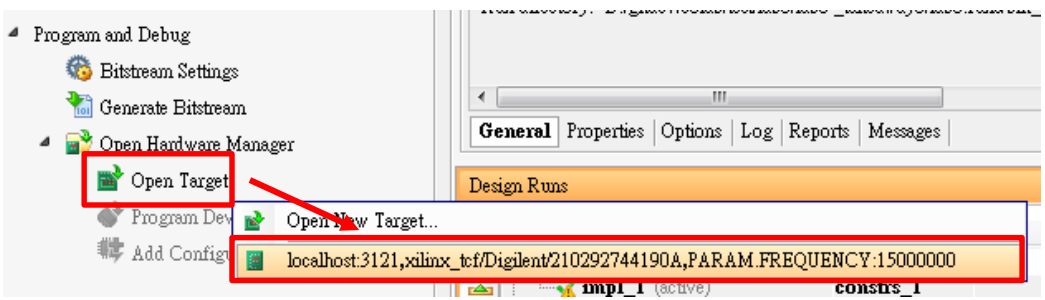
點開Hierarchy中 mem_top -> u_block_mem，在Other Options分頁下載入lab.coe，完成後按下Generate產生新的記憶體。

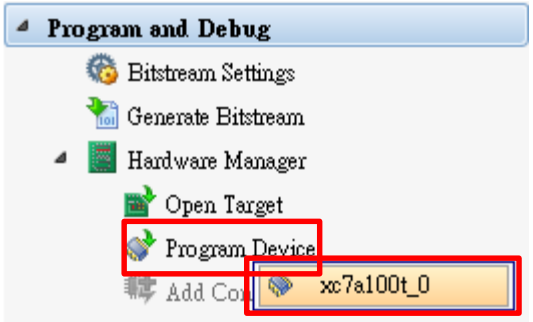


實作(三)－不計分

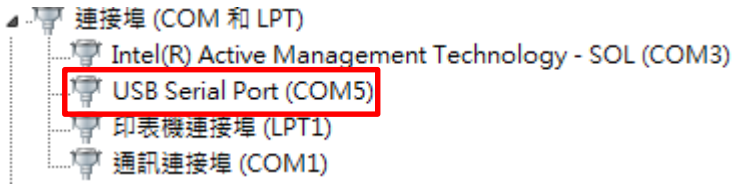
6. 按下Program and Dubug下的Generate Bitstream。

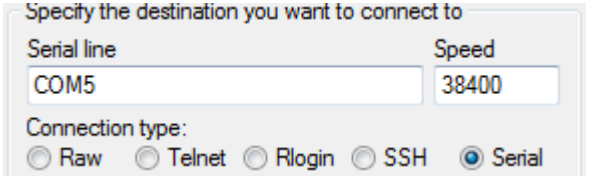
7.完成後Open Target選取板子，Program Device將資料燒入FPGA。
產生的bit檔放置在./vivado/lab10.runs/impl_1





8. 如同之前lab，利用終端機pietty與板子建立連線。
先找出USB的名稱，再開啟Pietty的puTTY模式進行連線
選取 Serial -> Serial line (Ex: COM5) & Speed (38400) -> Open





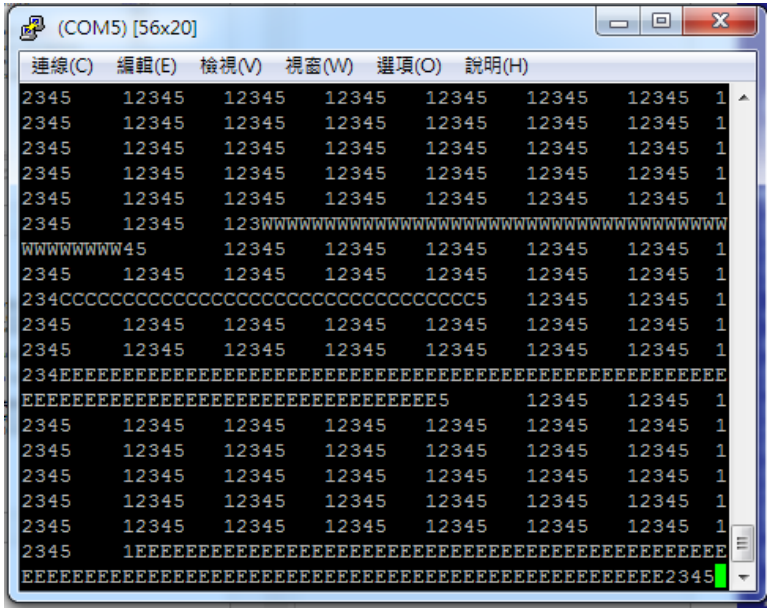
實作(三)－不計分

9. 在這個main_function.c的內容是CPU對UART印完”Hello EASY! 後，會在一個無限回圈內不斷印出”12345\t12345….”如下圖所示

```
void main_function ()
{
 volatile int *UART =  (int*) 0x20000000;
 int i;

  UART[0] = '\n' ;
  UART[0] = 'H' ; UART[0] = 'e' ;
  UART[0] = 'l' ; UART[0] = 'l' ;
  UART[0] = 'o' ; UART[0] = '\t' ;
  UART[0] = 'E' ; UART[0] = 'A' ;
  UART[0] = 'S' ; UART[0] = 'Y' ;
  UART[0] = '!' ; UART[0] = '\n' ;

  while (1){

      UART[0] = '1' ; for(i=0;i<20000;i++){}
      UART[0] = '2' ; for(i=0;i<20000;i++){}
      UART[0] = '3' ; for(i=0;i<20000;i++){}
      UART[0] = '4' ; for(i=0;i<20000;i++){}
      UART[0] = '5' ; for(i=0;i<20000;i++){}
      UART[0] = '\t'; for(i=0;i<20000;i++){}
  }
}
```

實作(三)－不計分

10.此時若對SW_E、SW_C、SW_W按鈕按下去，可以馬上在超級終端機中看到CPU跳到ISR中，對UART寫出是對應的E、C、W字元，並且會回到main_function中的無限迴圈繼續跑。(跳到ISR一次只會印出一個字元，會印出這麼多個'E'是因為我們手動按那個按鈕的瞬間已經被sample到Switch Interrupt 好幾次了!)



實驗結報

- 結報格式(每組一份)
  - 封面
  - 實驗內容(程式碼註解、結果截圖)
  - 實驗心得
- 繳交位置
  - ftp : 140.116.164.235
  - 帳號/密碼 : co2016

  DeadLine:禮拜天(1/13)晚上12:00
- TA Contact Information:
  - 助教信箱 : franksaifranksai@gmail.com
  - Rm 92617
  - Office hour : (Tues)1300~1500、(Wed)1300~1400