

處理器設計與實作

編撰者

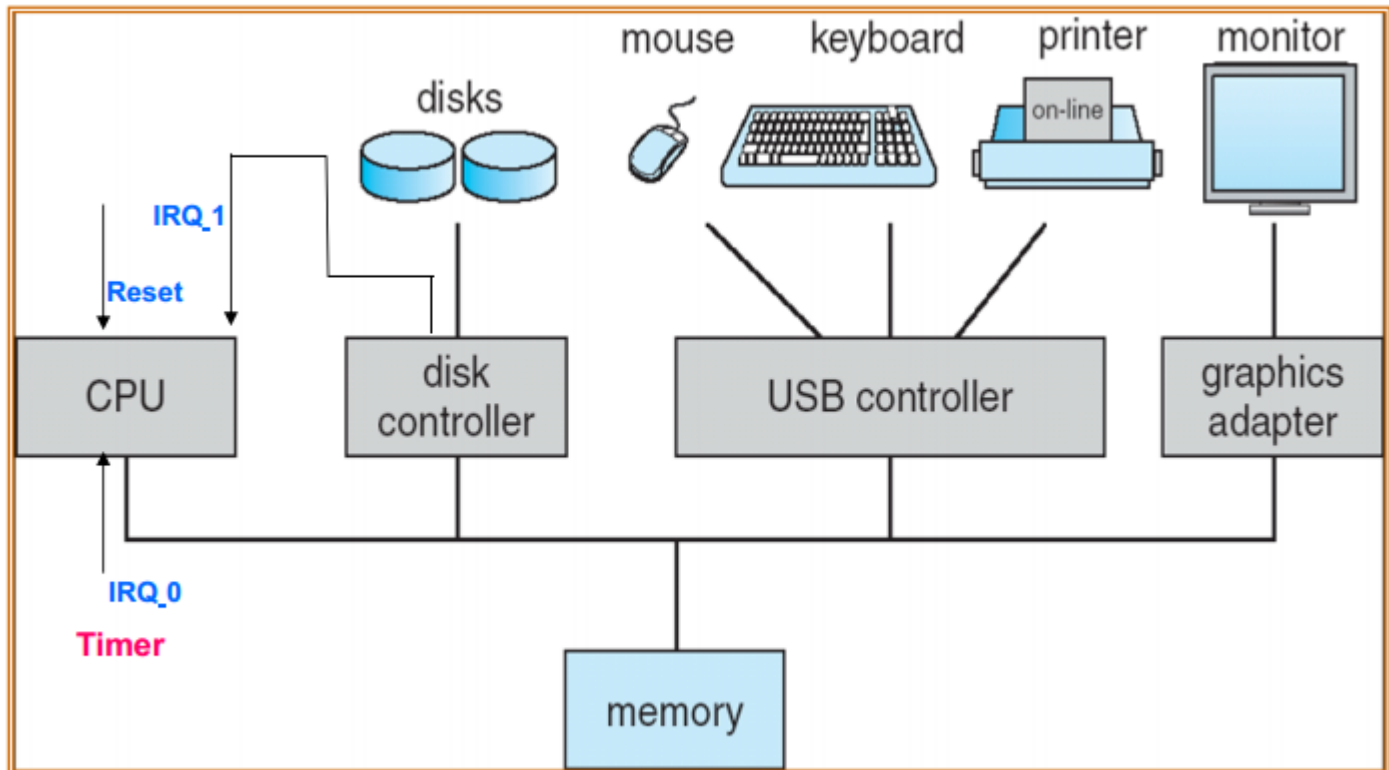
成大電通所計算機架構與系統研究室CASLAB

LAB 10: 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

Interrupts by External Signals



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。

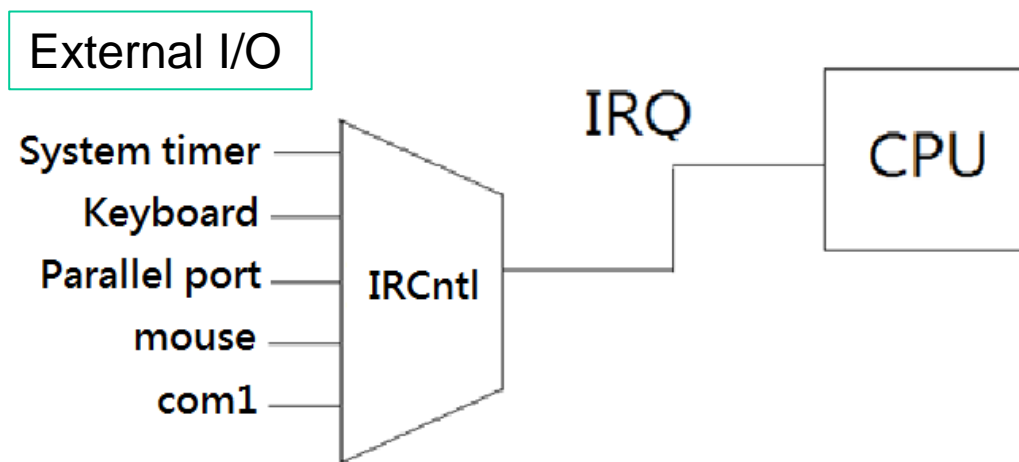
ARM interrupt vector table

⊕ 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

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)

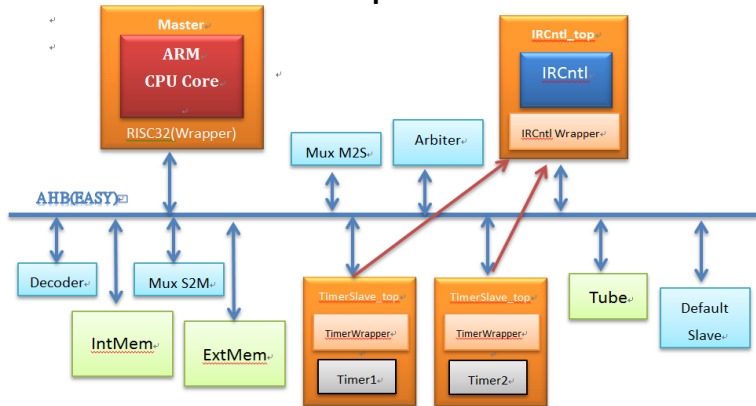


Hardware Interrupt 基本的流程 /A

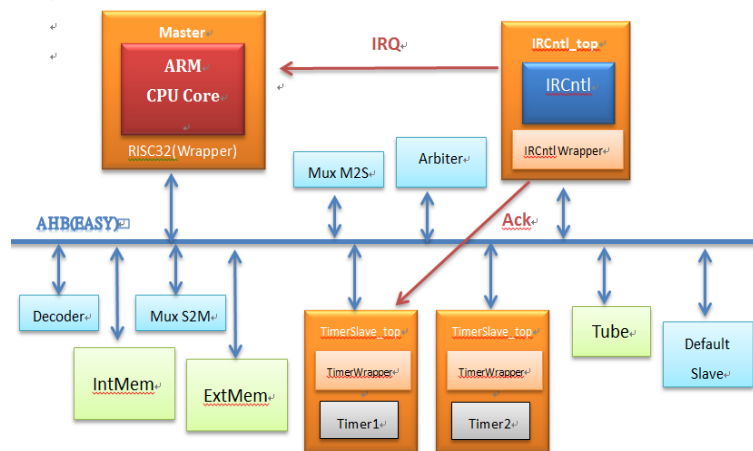
Step	說明
Device 發出中斷到 Interrupt Controller	Interrupt Controller接收Device中斷 Controller會知道是那個Device發出的中斷及此Device中斷的優先權。
Controller 發出IRQ到 CPU Controller Ack回Device	Interrupt Controller依據所有發出中斷的Device的優先權決定誰可以中斷CPU，代替該Device發出IRQ給CPU，並回應該Device IRQ Ack，表示Device的中斷已經被傳達。
CPU被Interrupt後，跳到ISR去後去執行該Device的Interrupt 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 /B

Device1 & Device2 send request to Interrupt Controller



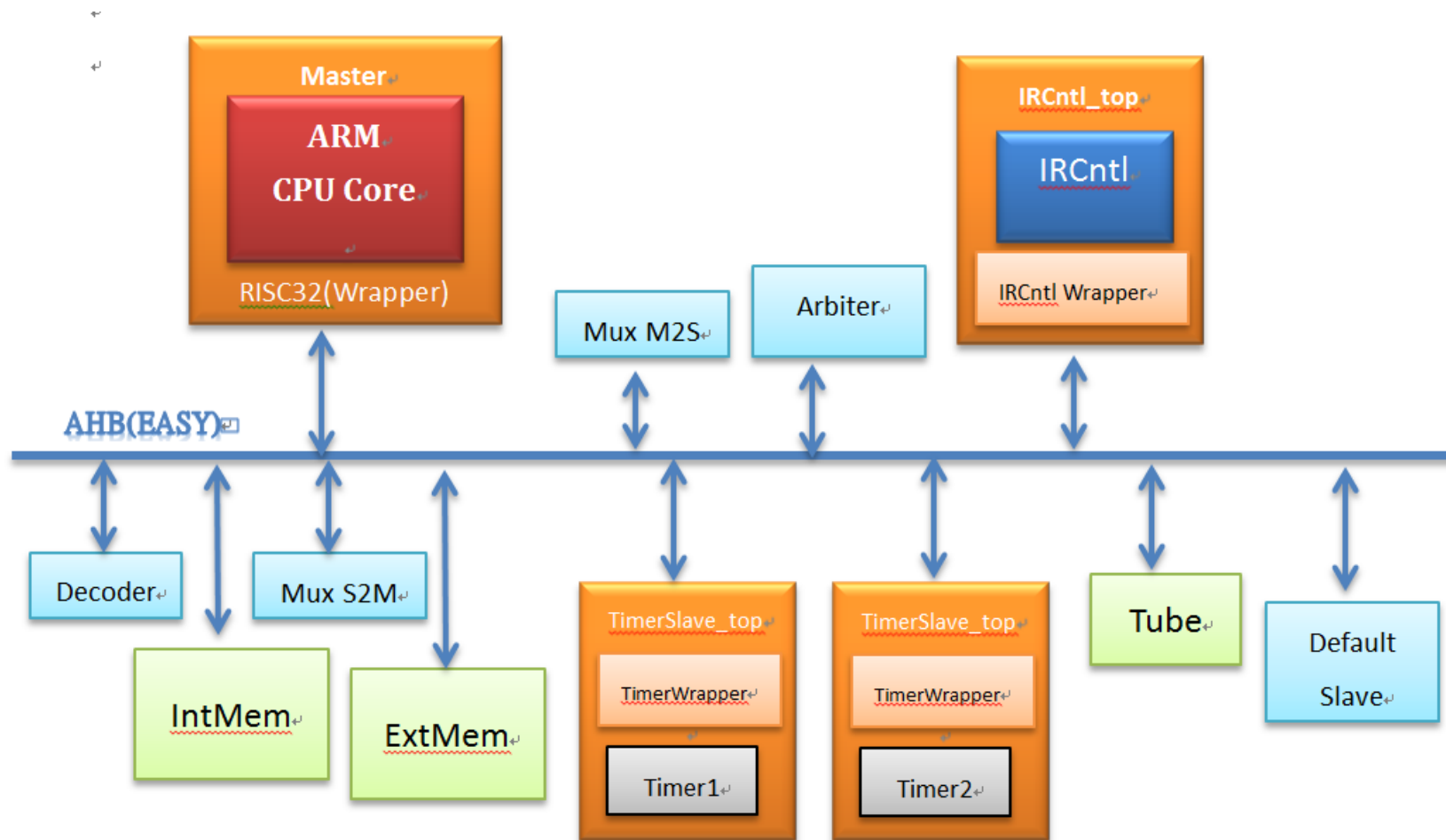
Controller Interrupt CPU for Device1



實驗系統架構

Example **A**MBA **S**ystem (EASY)

EASY平台 (ARM processor)



實驗環境

⊕ Tool used :

1. ARM GNU Toolchain overview

- 利用GNU 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

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。

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

實作(一)

2. 當 I/O device發出的interrupt 成功被IRCntI送出IRQ之後，IRCntI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 ;
```

IRCntI 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IRCntI 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IRCntI中去仲裁看哪個Timer被pending。

實作(一)

IMPORNTENT

4. EASY平台已經新增Timer的IRQ 和ACK線

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

wire      Timer1_ACK;
wire      Timer2_ACK;

wire      Timer1_IRQ;
wire      Timer2_IRQ;
```

5. 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),
    .HTRANS    (HTRANS),
    .HWRITES   (HWRITE),
    .HSIZES    (HSIZE),
    .HWDATAS   (HWDATA),
    .HSELS     (HSEL_Timer1),
    .HREADY    (HREADY),
    //AHB Bus Output
    .HRDATAS   (HRDATA_Timer1),
    .HREADYOUT (HREADY_Timer1),
    .HRESP     (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),
    .HTRANS    (HTRANS),
    .HWRITES   (HWRITE),
    .HSIZES    (HSIZE),
    .HWDATAS   (HWDATA),
    .HSELS     (HSEL_IRCntl),
    .HREADY    (HREADY),
    //AHB Bus Output
    .HRDATAS   (HRDATA_IRCntl),
    .HREADYOUT (HREADY_IRCntl),
    .HRESP     (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

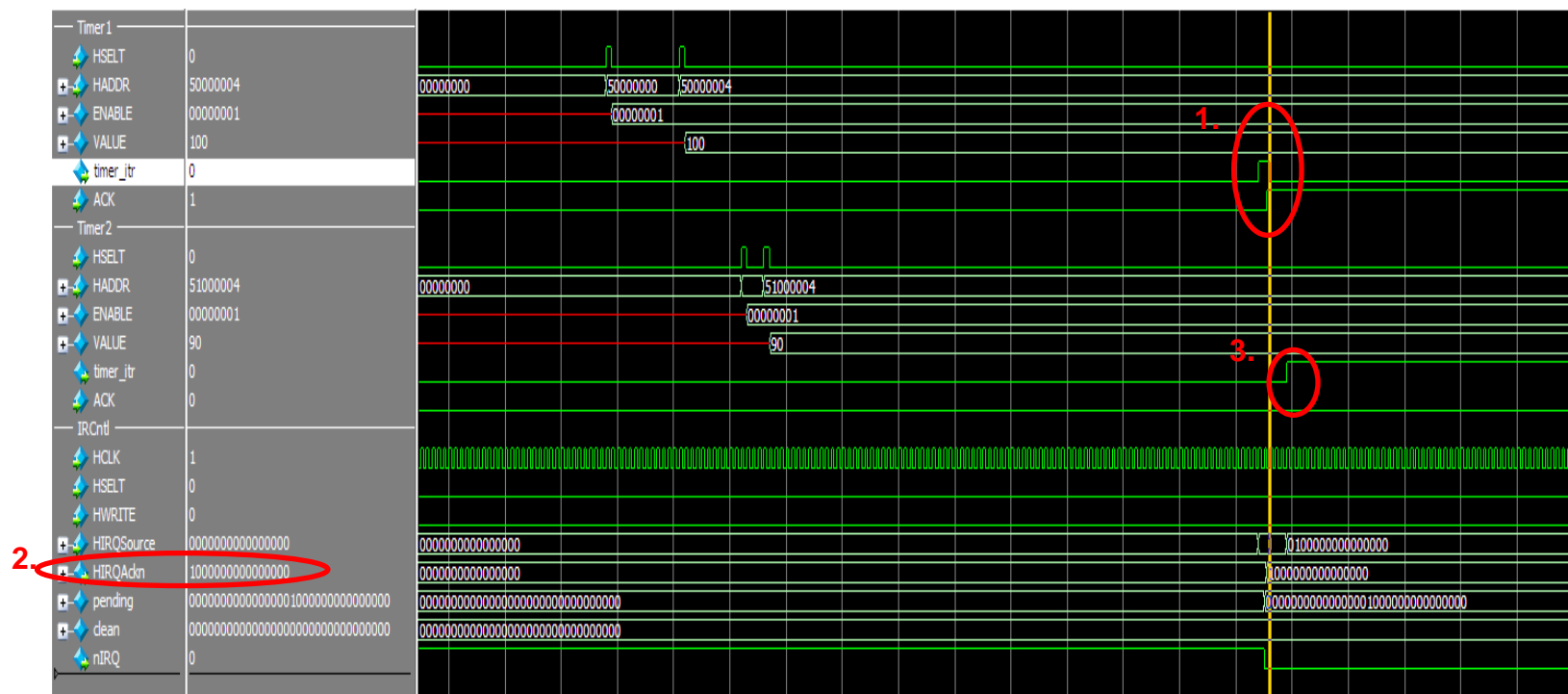
設定Timer1和Timer2的ENABLE和VALUE的值，mapping位置如下

		Timer1		Timer2
<pre>volatile int* Timer1; volatile int* Timer2; Timer1 = (int*) 0x50000000; Timer1[0] = 1; Timer1[1] = 90; Timer2 = (int*) 0x51000000; Timer2[0] = 1; Timer2[1] = 100;</pre>	0x50000000	ENABLE	0x51000000	ENABLE
	0x50000004	VALUE	0x51000004	VALUE
	0x50000008	空白	0x51000008	空白
	0x50FFFFFF	空白	0x51FFFFFF	空白

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

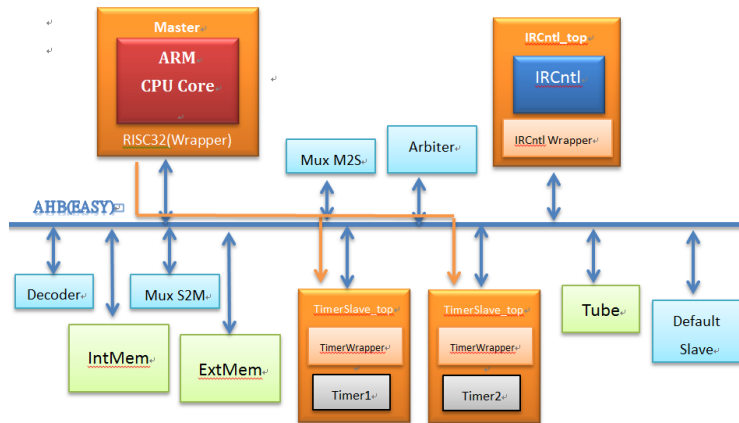
實作(一)

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。

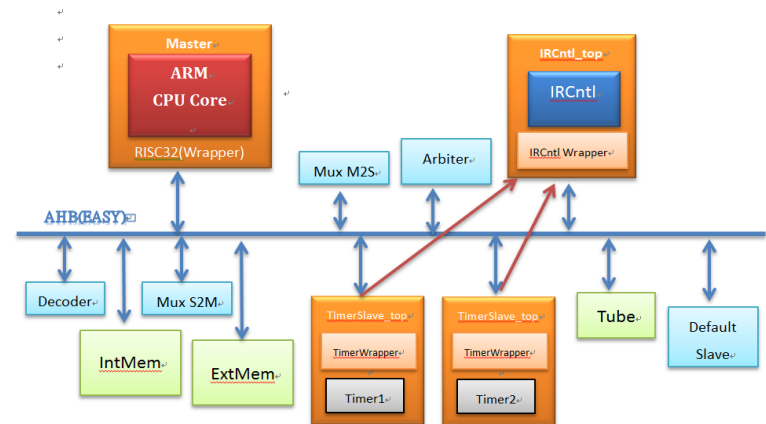


實作(一)

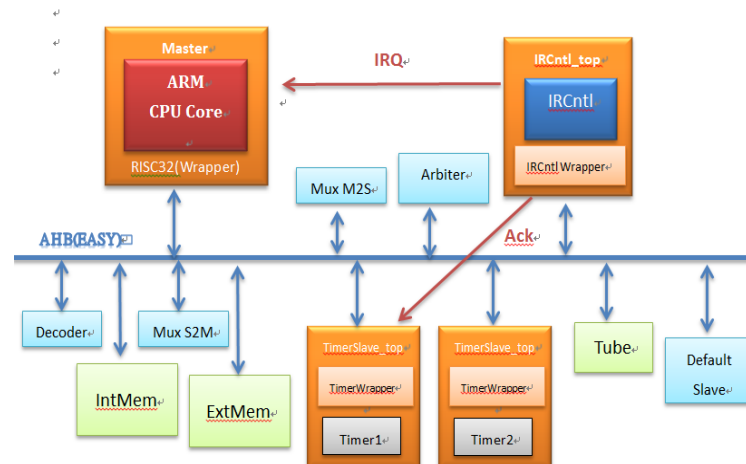
1. CPU enable Timer1、Timer2



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. EASY.v已經新增
 input SW1_IRQ
 output LED1_ACK
2. 把這兩條線接到IRCntl的HIRQSource和
 HIRQAck的top bit

```

IRCntlSlave_top  IRCntl(
    // Input signal from System

    //AHB Bus Input
    .HCLK      (HCLK),
    .HRESET_n  (HRESETn),
    .HADDRS    (HADDR),
    .HTRANS    (HTRANS),
    .HWRITES   (HWRITE),
    .HSIZES    (HSIZE),
    .HWDATAS   (HWDATA),
    .HSELS     (HSEL_IRCntl),
    .HREADY    (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)
);

```

```

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

input  HCLK;
input  HRESETn;
input  SW1_IRQ;
output LED1_ACK;

// wire nFIQ;
wire nIRQ;

```

實作(二)

3. 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interrupt，請讓Tube印出 "**LED"
    {
        .
        .
        .
    }
    else if(IRCntl[0]==0x4000)     //若是Timer1發出的interrupt，請讓Tube印出"Ter1"
    {
        .
        .
        .
    }
    else if(IRCntl[0]==0x2000)     //若是Timer2發出的interrupt，請讓Tube印出"Ter2"
    {
        .
        .
        .
    }

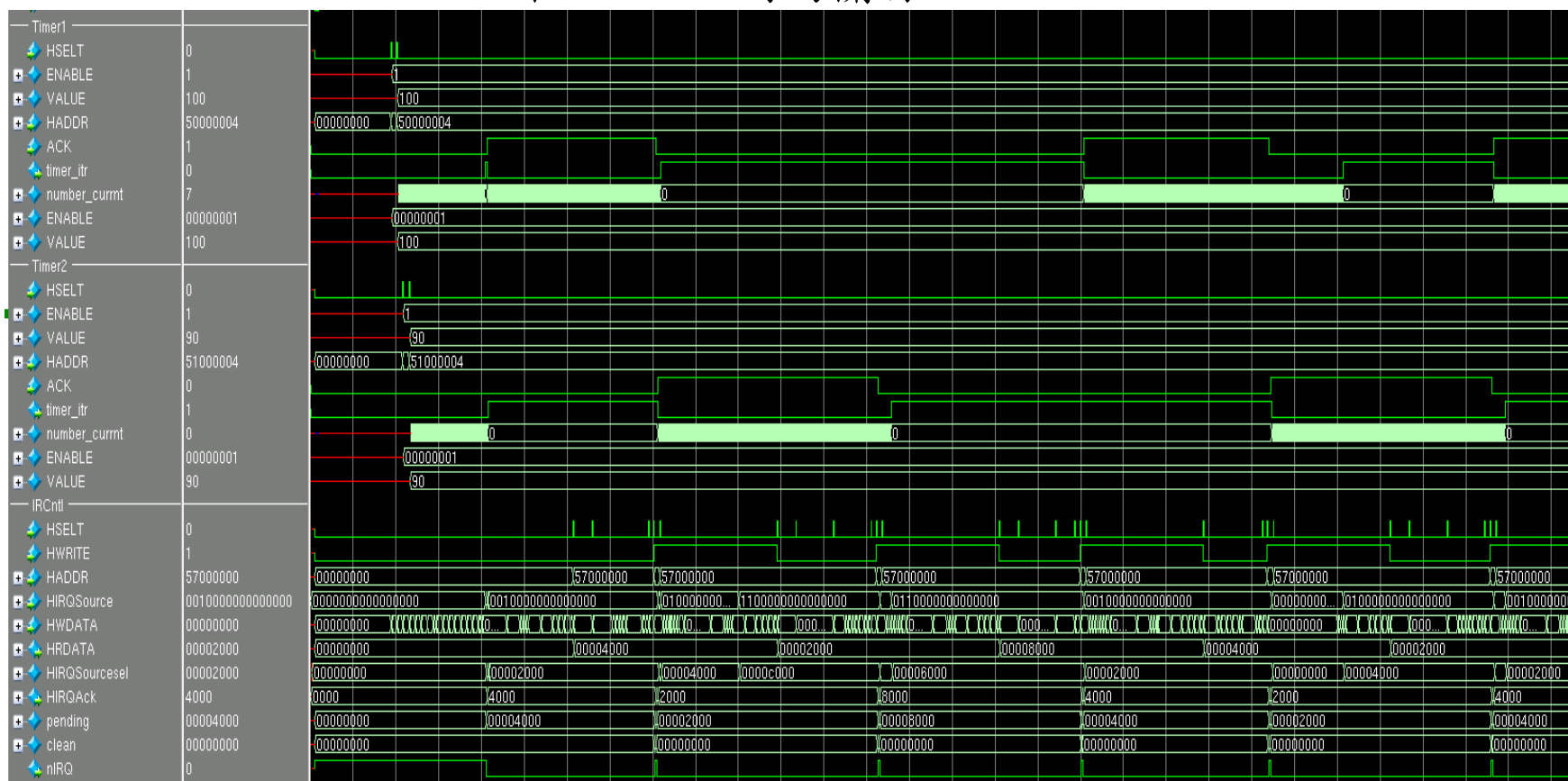
    //請讓clean被寫入pending的值，再對pending寫入0x0
    IRCntl[1] =
    IRCntl[0] =
}

```

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

實作(二)

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 : 140.116.164.225
- 帳號/密碼 : ca_lab / Carch2020

DeadLine: 禮拜天(12/31)晚上12:00

● TA Contact Information:

- 助教信箱 : anita19961013@gmail.com
- Rm 92617