影像處理之軟硬體協同

負 責 人：劉益彤、吳東穎

開始日期: 2023/11/09

報告日期: 2023/12/21

Git連結

需求

功能:

- 輸入車道灰階影像(720*480, 從SD卡讀取)
- ORB演算出稀疏光流
- 透過UDP傳送至host(PC)
- 在PC上用網頁輸出影像處理的結果(畫光流方向)

效能:

- 處理速度
  - clock: 100MHz
  - FPS: 40~50幀
- 呈現速度
  - FPS : 40~50幀
- 網路傳輸速度
  - 100Mbps(待測)

環境:

- Vivado 2020.2
- python3.8.10
- flask/Django
- zedboard(xc7z020clg484-1)

介面:

- UDP/IP
- AXI-stream

需求

驗收:

驗功能:

1. ILA查AXI介面(DMA⇔ORB)
2. 直接開SDK看結果(PL端中斷、當前幀資料流)
3. 用網頁看影像結果(RGB444)
4. 用網頁看影像結果(RGB444、匹配座標、光流方向)

驗效能:

1. SDK查DMA搬運速度
2. 以軟體(opencv)為groundtruth，跟硬體加速做比較

預期結果:

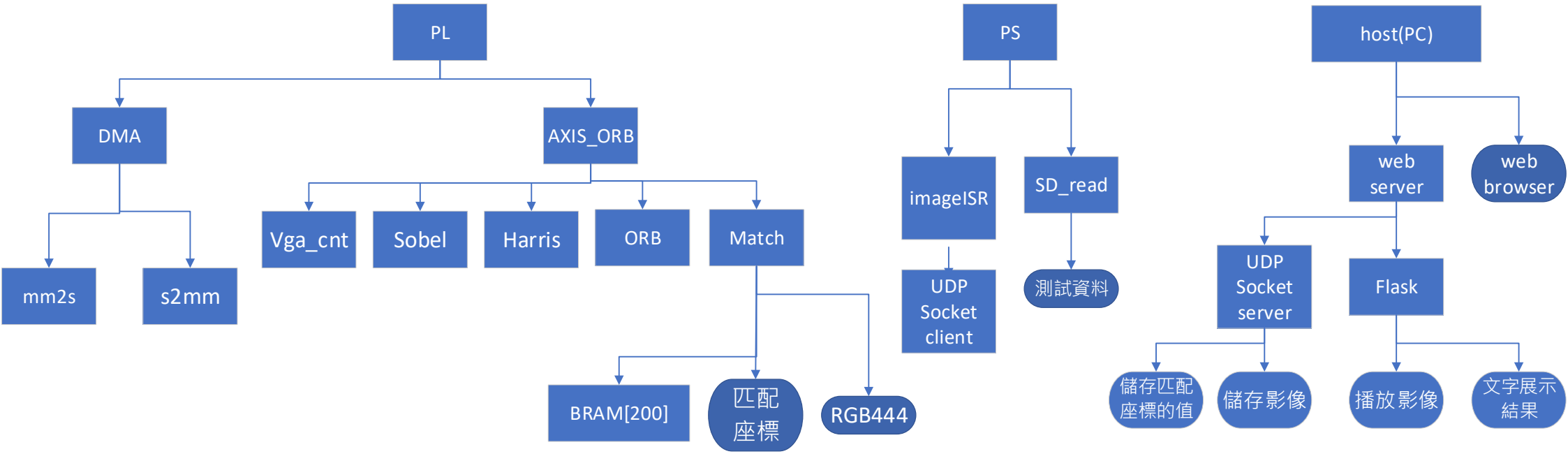
驗功能:

1. AXI-stream介面交握正確
2. PL端中斷後，當前幀行列數正確
3. 硬體處理後的灰階影像、匹配座標(RGB444)
4. 以單張灰階影像u8 [720*480] 測試Zynq ←socket=====→PC

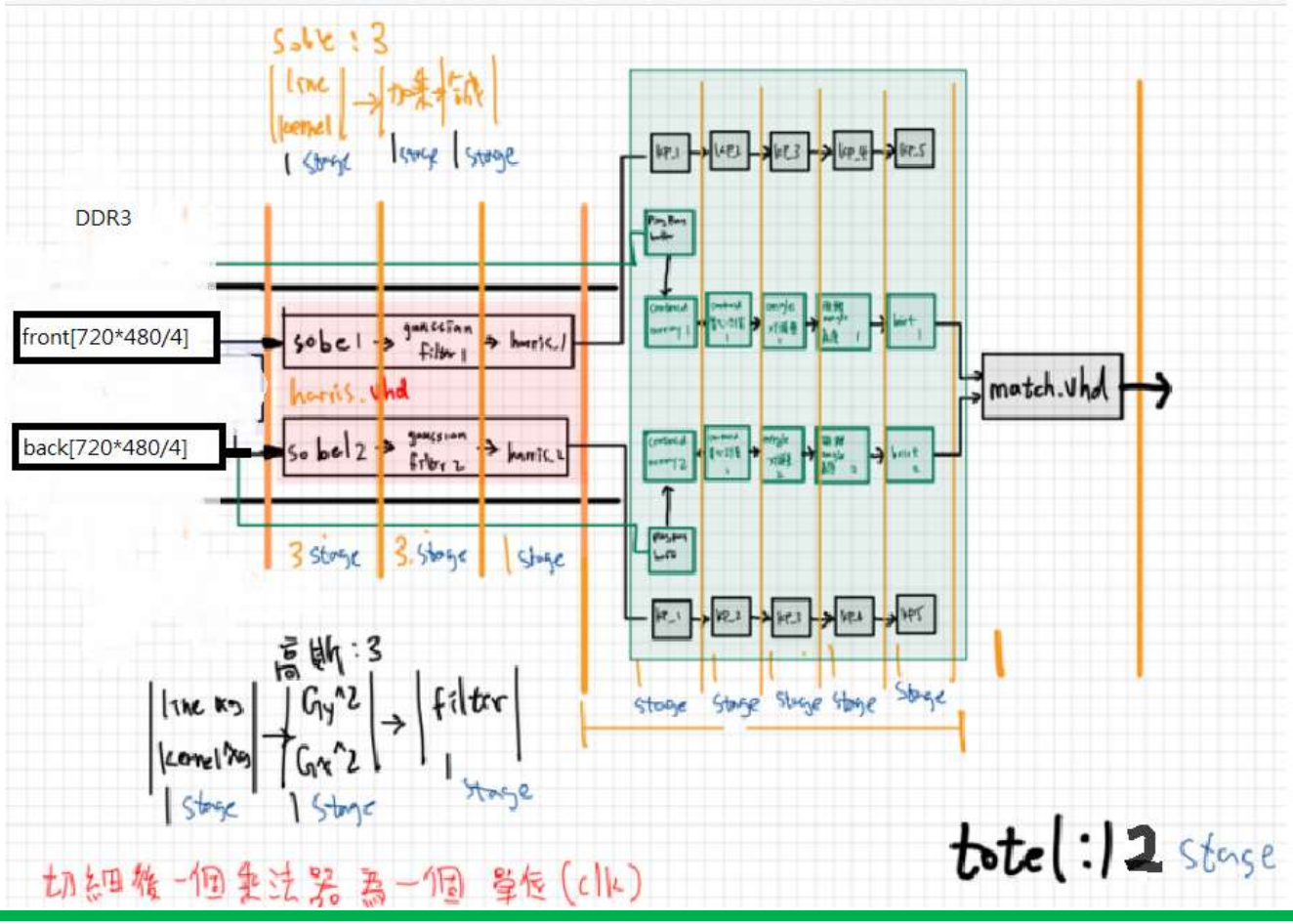
驗效能:

1. 以xil_printf()在Terminal寫出XXMB/s
2. 軟體跟硬體畫面同框比較

Breakdown


PL
DMA
AXIS_ORB
mm2s
s2mm
Vga_cnt
Sobel
Harris
ORB
Match
BRAM[200]
匹配座標
RGB444
PS
imageISR
SD_read
UDP Socket client
測試資料
host(PC)
web server
web browser
UDP Socket server
Flask
儲存匹配座標的值
儲存影像
播放影像
文字展示結果

分析-PL端打包模組



圖源、GSlab_影像組

說明- DMA內部暫存器

為了解DMA_busy、DMA預設reg值關係

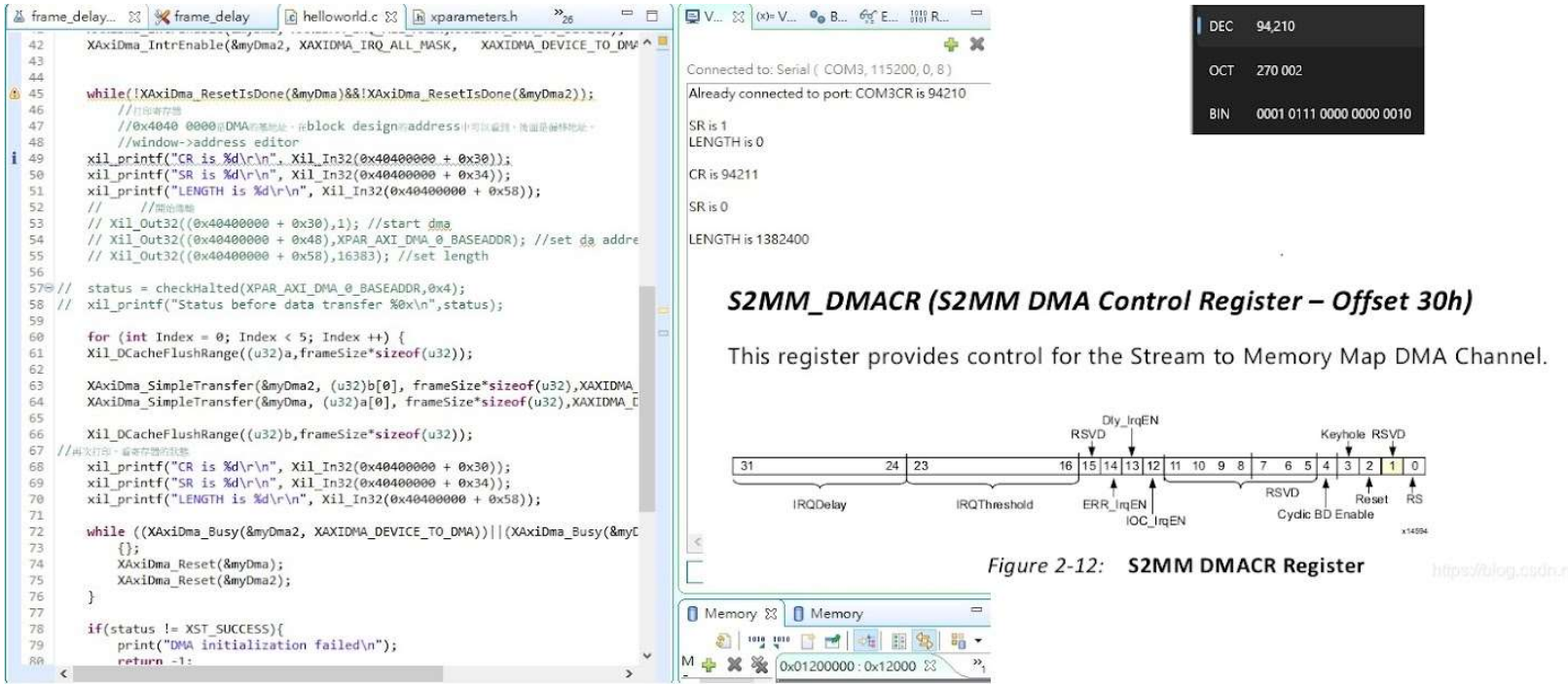
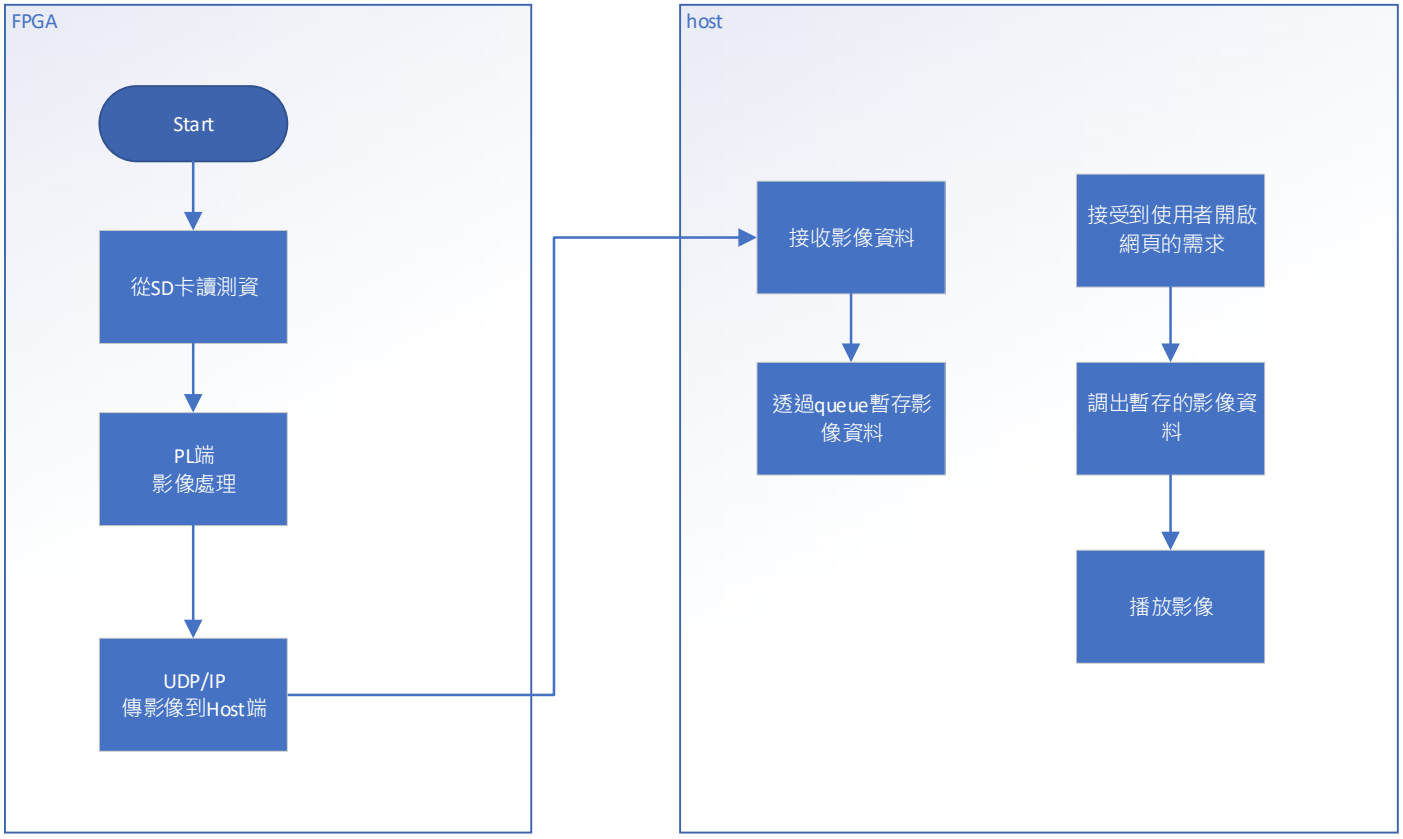


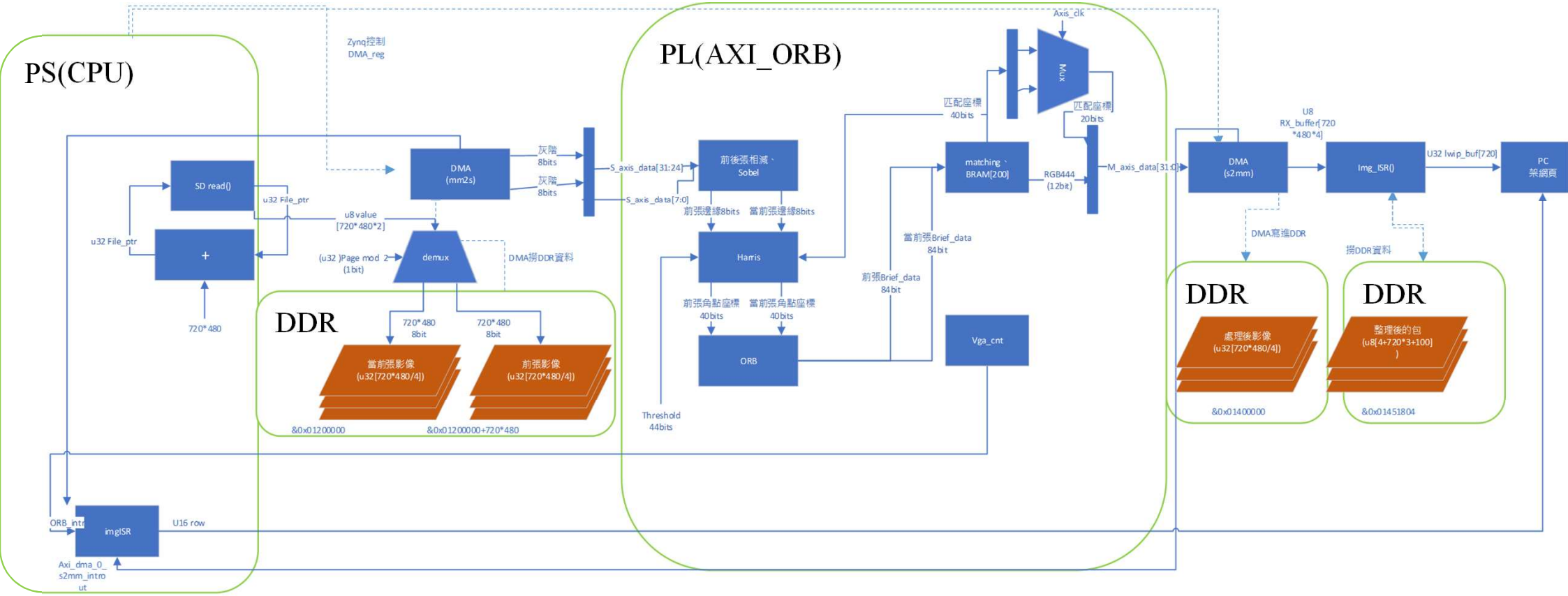
圖 、以單次burst測試

Xilinx _手冊:https://docs.xilinx.com/r/en-US/pg021_axi_dma/Memory-Map-to-Stream-Register-Detail

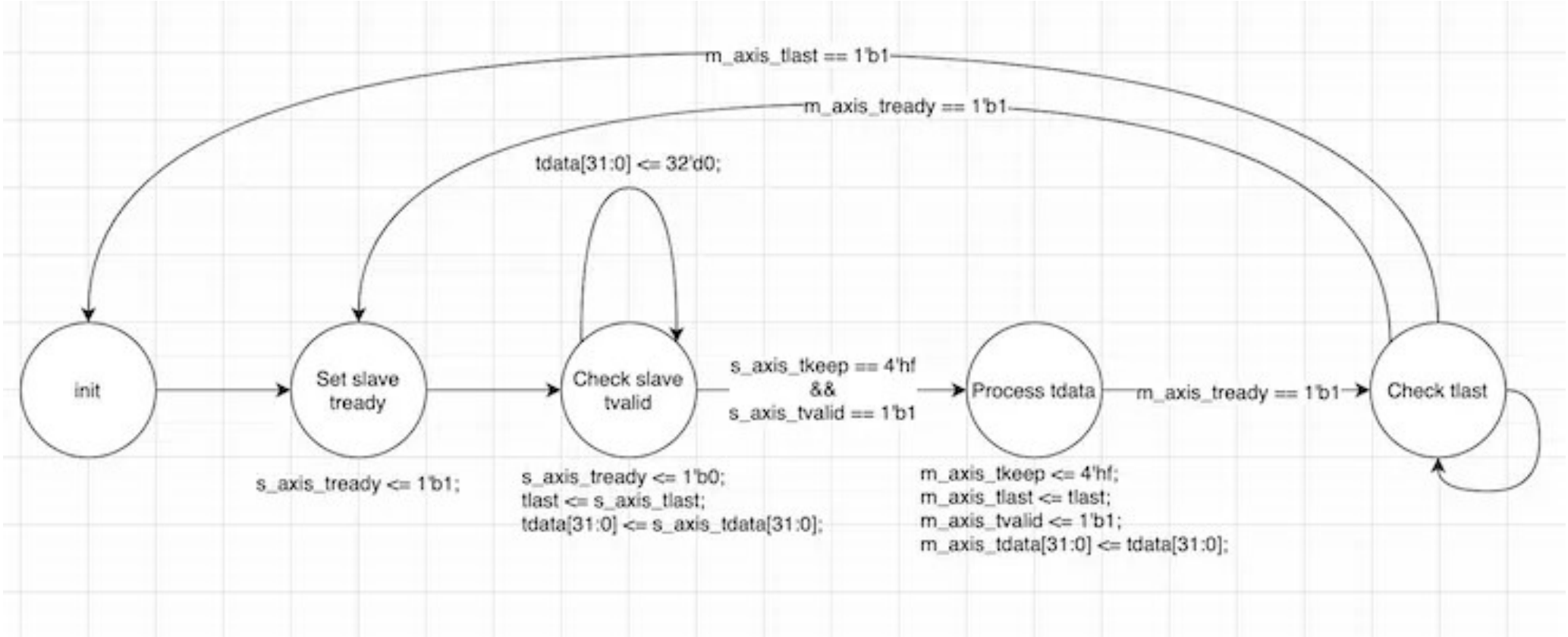
設計-流程圖(ver1)


FPGA
Start
從SD卡讀測資
PL端
影像處理
UDP/IP
傳影像到Host端
host
接收影像資料
透過queue暫存影像資料
接受到使用者開啟網頁的需求
調出暫存的影像資料
播放影像

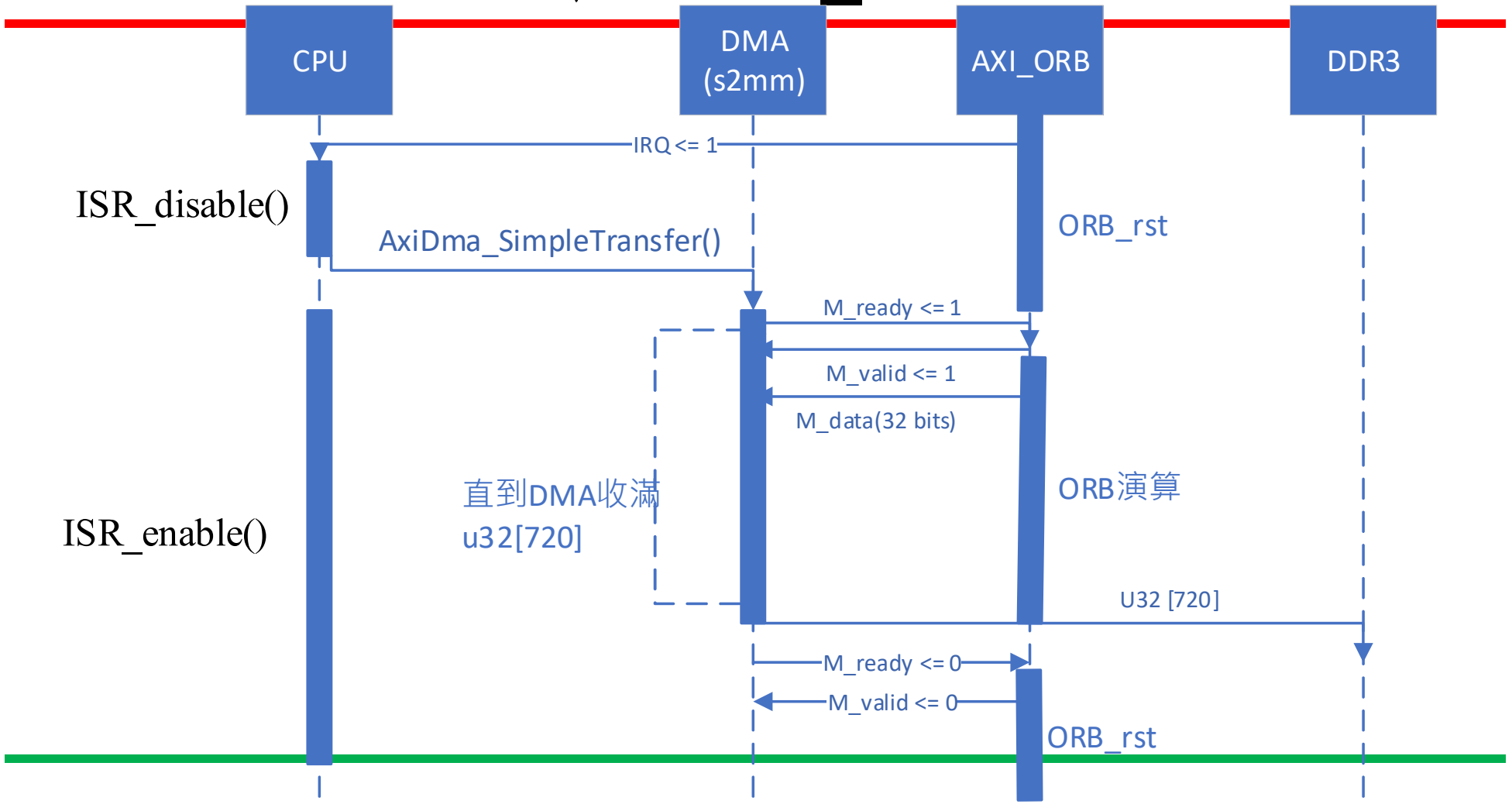
設計-硬體架構圖(ver2. 未架系統)



設計-FSM_axis

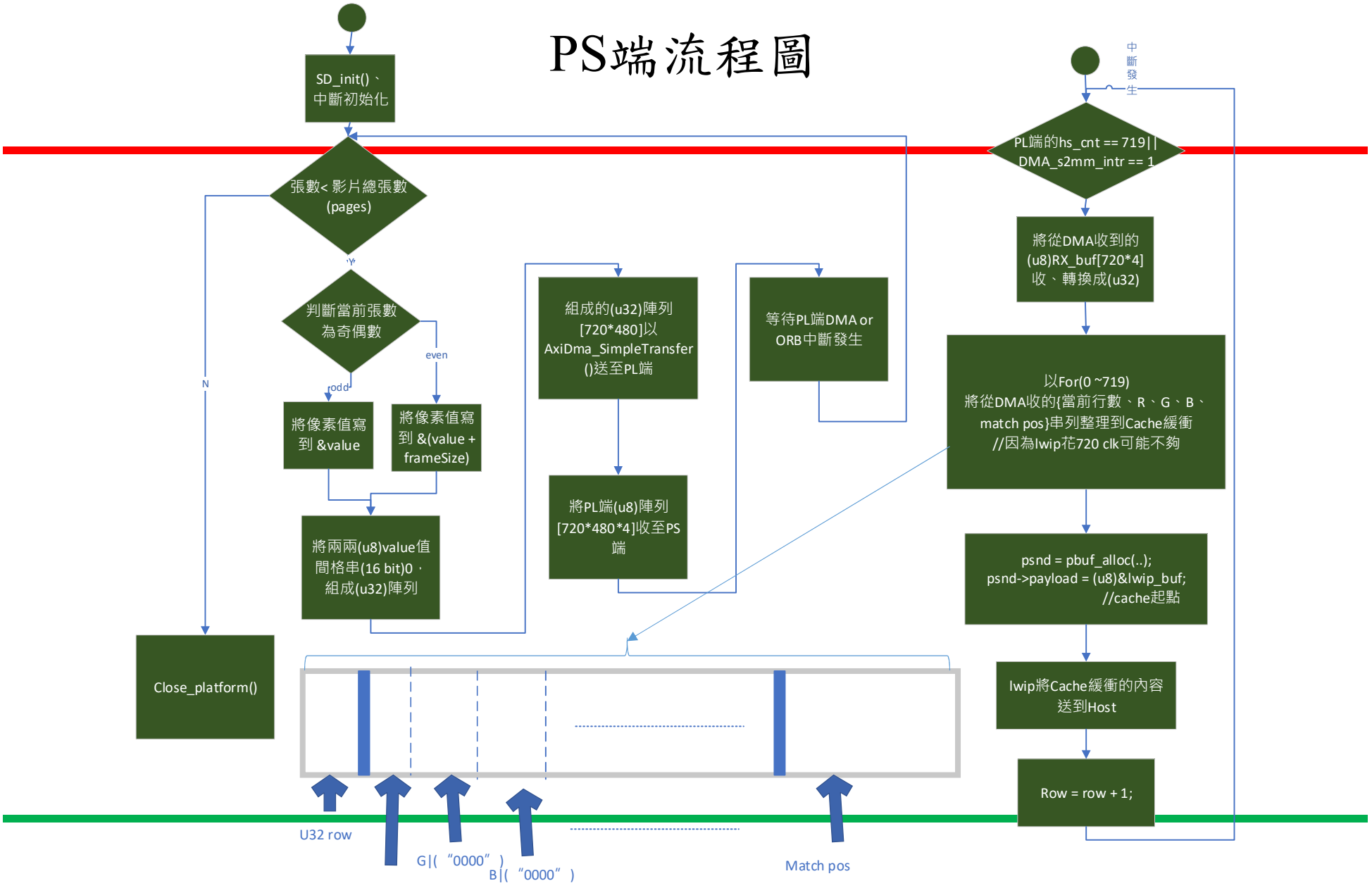

m_axis_tlast == 1'b1
m_axis_tready == 1'b1
tdata[31:0] <= 32'd0;
init
Set slave tready
Check slave tvalid
s_axis_tkeep == 4'hf
&&
s_axis_tvalid == 1'b1
Process tdata
m_axis_tready == 1'b1
Check tlast
s_axis_tready <= 1'b1;
s_axis_tready <= 1'b0;
tlast <= s_axis_tlast;
tdata[31:0] <= s_axis_tdata[31:0];
m_axis_tkeep <= 4'hf;
m_axis_tlast <= tlast;
m_axis_tvalid <= 1'b1;
m_axis_tdata[31:0] <= tdata[31:0];

設計-DMA_s2mm

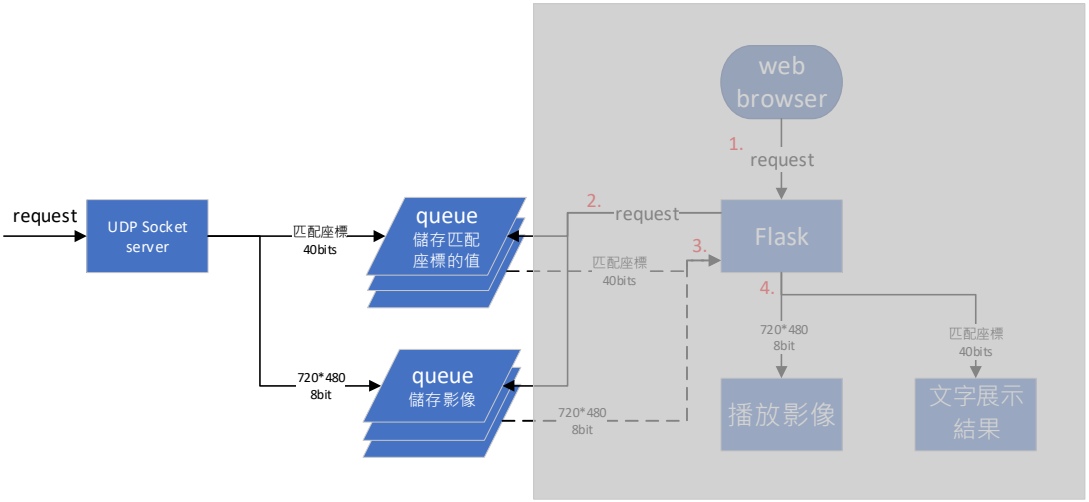


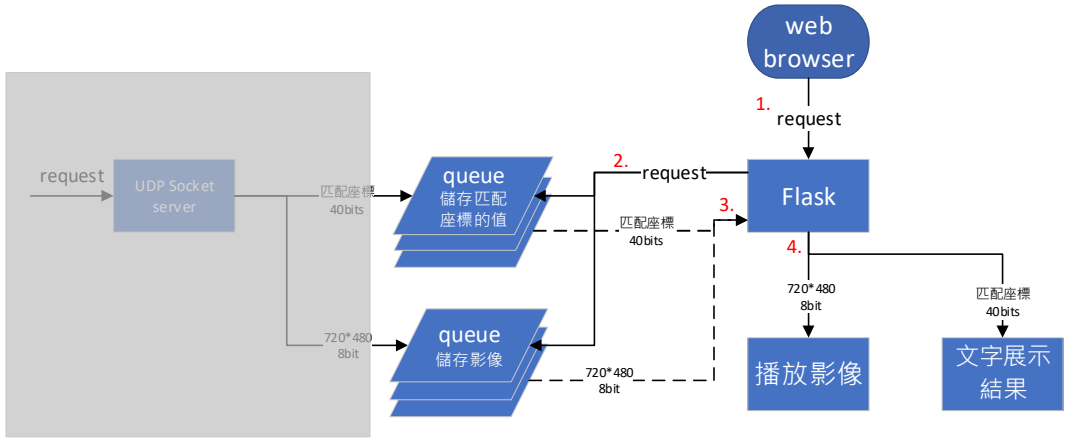
MSC圖 、ORB wr 2 DMA

PS端流程圖

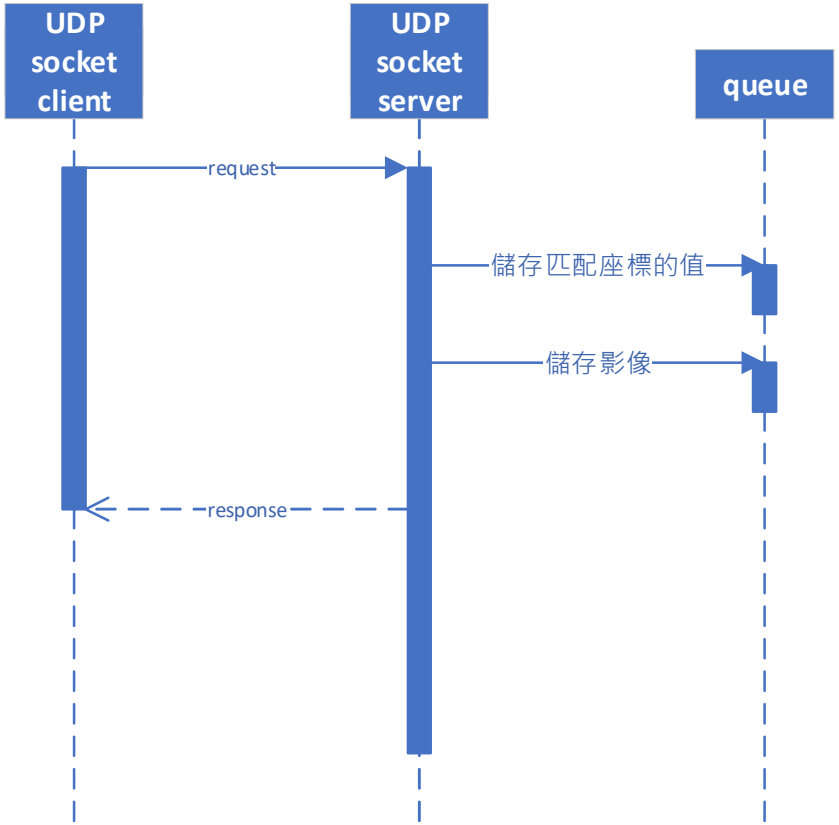

SD_init()、
中斷初始化
張數< 影片總張數
(pages)
N
判斷當前張數
為奇偶數
odd
even
將像素值寫
到 &value
將像素值寫
到 &(value +
frameSize)
將兩兩(u8)value值
間格串(16 bit)0，
組成(u32)陣列
Close_platform()
組成的(u32)陣列
[720*480]以
AxiDma_SimpleTransfer
()送至PL端
將PL端(u8)陣列
[720*480*4]收至PS
端
等待PL端DMA or
ORB中斷發生
中
斷
發
生
PL端的hs_cnt == 719||
DMA_s2mm_intr == 1
將從DMA收到的
(u8)RX_buf[720*4]
收、轉換成(u32)
以For(0 ~719)
將從DMA收的{當前行數、R、G、B、
match pos}串列整理到Cache緩衝
//因為lwip花720 clk可能不夠
psnd = pbuf_alloc(..);
psnd->payload = (u8)&lwip_buf;
//cache起點
lwip將Cache緩衝的內容
送到Host
Row = row + 1;
U32 row
G|(“0000”)
B|(“0000”)
Match pos

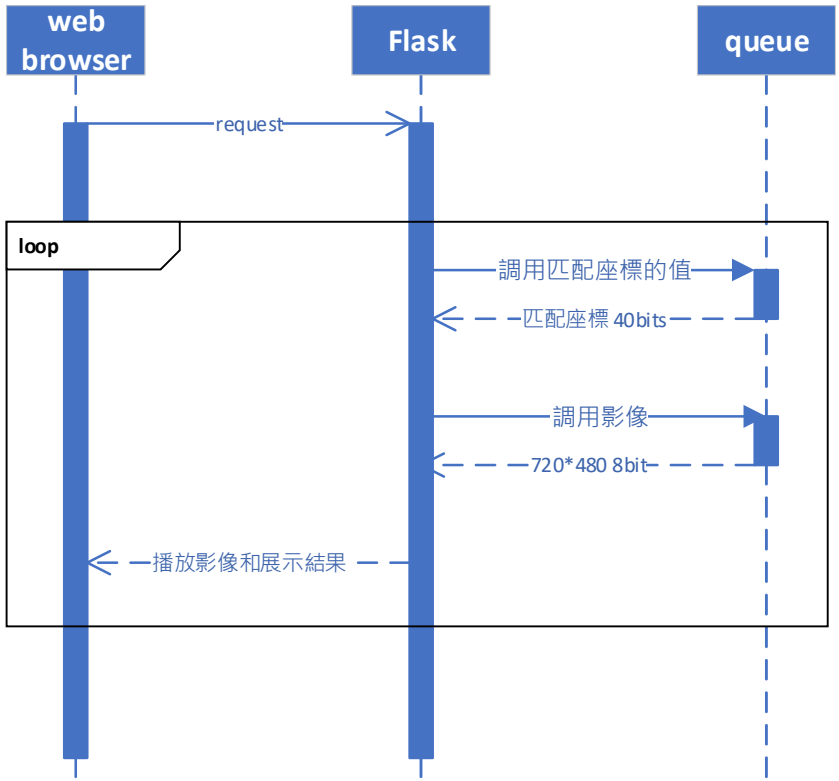
設計- PC端 流程圖


web browser
1.
request
request
UDP Socket server
匹配座標 40bits
queue 儲存匹配座標的值
2.
request
Flask
匹配座標 40bits
3.
4.
720*480 8bit
匹配座標 40bits
720*480 8bit
queue 儲存影像
720*480 8bit
播放影像
文字展示結果


request
UDP Socket server
匹配座標 40bits
queue 儲存匹配座標的值
web browser
1.
request
2.
request
Flask
匹配座標 40bits
3.
4.
720*480 8bit
匹配座標 40bits
720*480 8bit
queue 儲存影像
720*480 8bit
播放影像
匹配座標 40bits
文字展示結果

設計-PC端 MSC


UDP socket client
UDP socket server
queue
request
儲存匹配座標的值
儲存影像
response

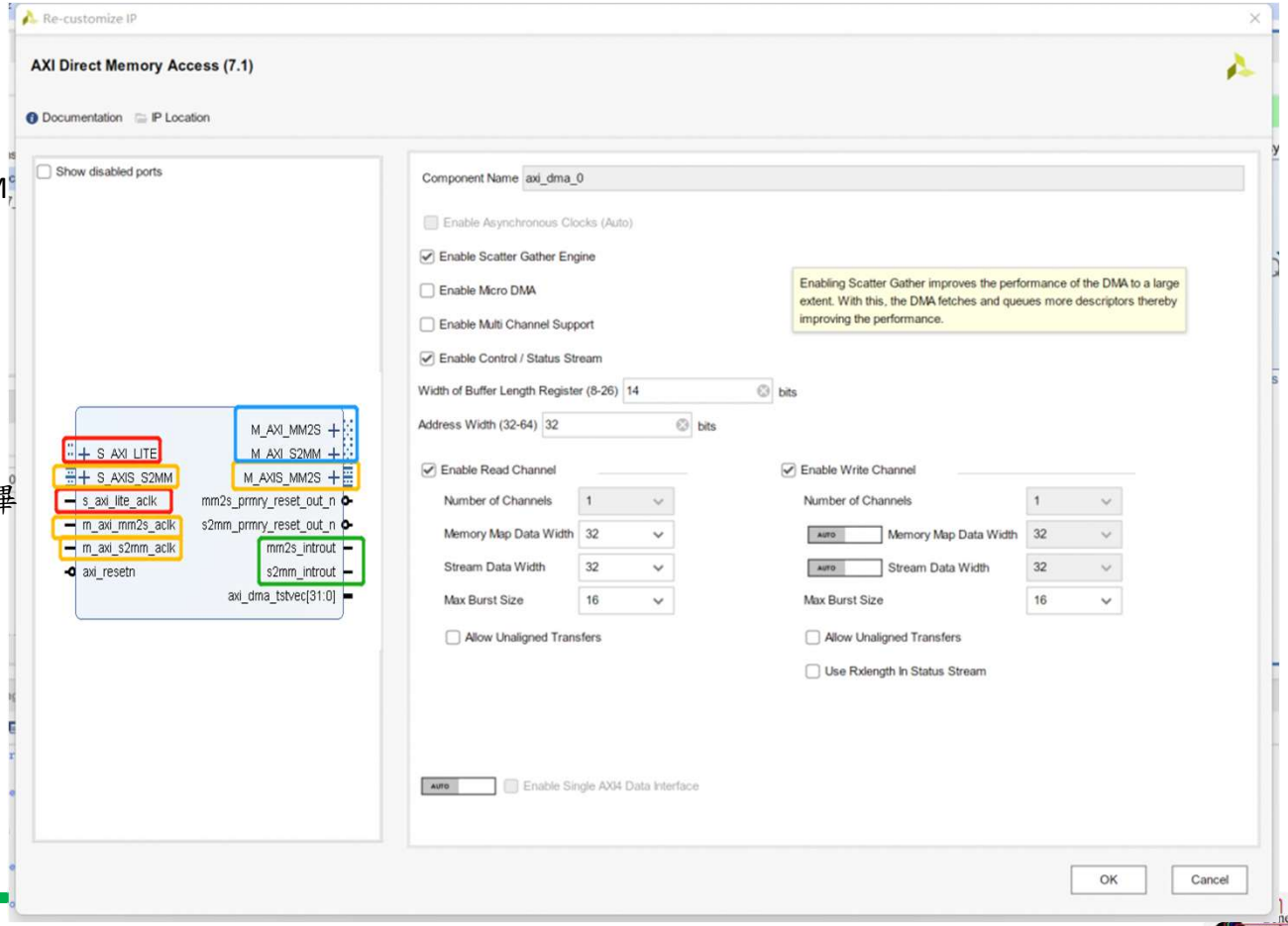

web browser
Flask
queue
request
loop
調用匹配座標的值
匹配座標 40bits
調用影像
720*480 8bit
播放影像和展示結果

API-PL端(AXIS_ORB)

Name	AXIS_ORB
Inputs	input axi_Mclk, input axi_reset_n, input wire [32-1:0] s_axis_data, input wire s_axis_valid, input wire [3:0] s_axis_keep, input wire m_axis_ready,
Outputs	output s_axis_ready, output wire [32-1:0] m_axis_data, output m_axis_valid, output ORB_intr,
Parameters	DMA_rst [9:0] vga_vs_cnt [9:0] vga_hs_cnt
Methods	s_axis⇔DMA_1收DDR影像u32 TX_buffer[720*480] m_axis⇔DMA_0發影像到DDR u32 RX_buffer [720] ORB_intr⇔IRQ_F2P[61]

API-PL端(DMA)

- 紅色部分（IP控制埠）
  - S_AXI_LITE ⇔Zynq的Master AXI GP）aclk是時鐘
- 黃色部分（AXI-Stream協定連接埠）
  - S_AXIS_S2MM AXI-Stream Slave端口
    - 取得AXI-Stream資料流並透過M_AXI_S2MM進行Memory Map(寫入記憶體)
  - M_AXIS_MM2S AXI-Stream Master端口
    - 取得從M_AXI_MM2S得到的資料並轉換成AXI-Stream協定進行傳送
  - 剩下兩個aclk是各自的時鐘
- 綠色部分（中斷）
  - mm2s_introut mm2s中斷
    - 代表指定的長度已經全部作為AXIS發送完畢
  - s2mm_introut s2mm中斷
    - 代表AXIS的資料已經全部映射到內存



API- SD_read(Zynq)

Name	SD_Transfer_read()
Inputs	char *FileName, u32 DestinationAddress, u32 ByteLength
Outputs	FRESULT rc;
Parameters	#define imageSize 345600 //720*480 FIL fil; FRESULT rc; UINT br;

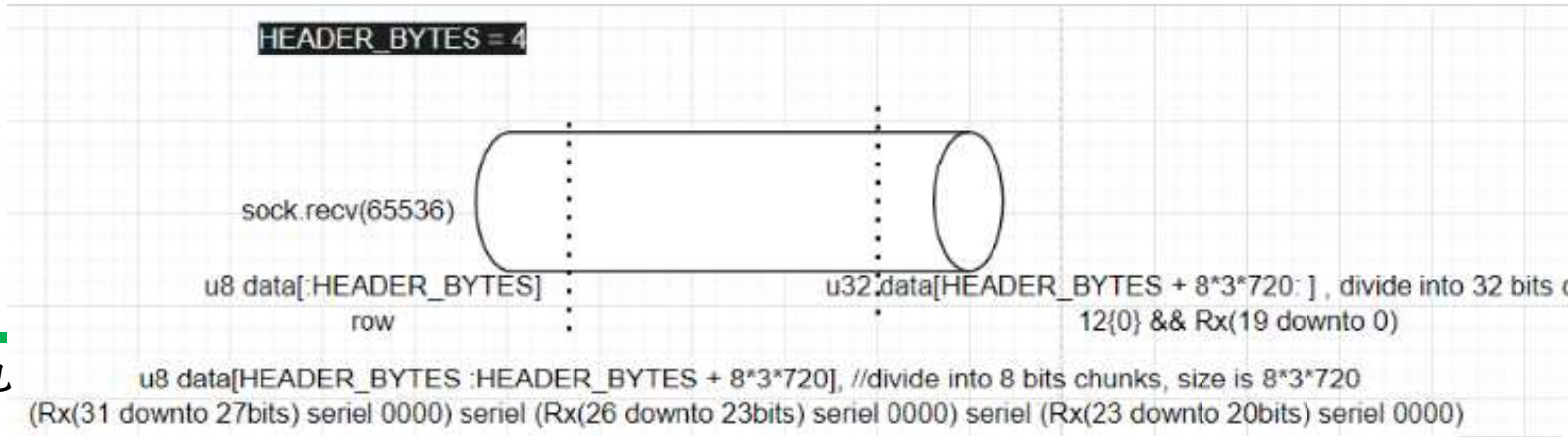
Methods	f_open(&fil, FileName, FA_READ);
	
	// Move the file pointer
	f_lseek(&fil, file_pointer);
	
	// Read data from the file
	f_read(&fil, (void *) DestinationAddress, ByteLength, &br);
	
	// Close the file
	f_close(&fil);

API-Zynq_ISR

lame	imageProcISR( )
nputs	XAxiDma *CallBackRef, // (u32*)&RX_buf_ptr U32* row
Outputs	bool Rx_done
arameters	#define RX_length 720

Methods	udp_sendto(&send_pcb, psnd, &RemoteAddr, RemotePort); //psnd->payload = &RX_buf; //RX_buf[4+720+200]={
	RGB, match pos} //預估單行的match<2

(u8)RX_buffer切分、PC端收包



API-PC端

Name	Get /image/<frame>
Inputs	frame: (int)取某一幀的圖片
Outputs	img: (2-D array 720*480)
Parameters	None
Methods	前端網頁向後端要特定幀的圖片，每次回傳一幀 Browser: http://127.0.0.1:8100/image/0 Return: 

API-PC端

Name	Get /result/<frame>
Inputs	frame: (int)取某一幀的結果
Outputs	result: (string)影像處理結果
Parameters	None
Methods	前端網頁向後端要特定幀的結果，每次回傳該幀的結果 Browser: http://127.0.0.1:8100/result/0 Return: Test/test/test Test2/test2/test2

API-PC端

Name	udp_socket_server(ip, port)
Inputs	data: (string) fpga處理後的圖片或結果
Outputs	response: (bool) True/False
Parameters	ip: (string)監聽的ip位址，預設"0.0.0.0" port: (int)監聽的port，預設8001
Methods	開啟一個UDP socket server，預設監聽所有IP的請求，收到請求後判斷是圖片還是結果

```
try:
    udp_thread = threading.Thread(
        target=udp_listener, args=("0.0.0.0", 8001))
    udp_thread.daemon = True
    udp_thread.start()

    app.run(debug=True, host="0.0.0.0", port=8000)
except Exception as e:
    print(e)
```

```
let02 ❯❯ python .\app.py
Start UDP listener: 0.0.0.0:8001
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:8000
 * Running on http://192.168.200.80:8000
Press CTRL+C to quit
 * Restarting with stat
Start UDP listener: 0.0.0.0:8001
 * Debugger is active!
 * Debugger PIN: 141-676-121
```

API-PC端

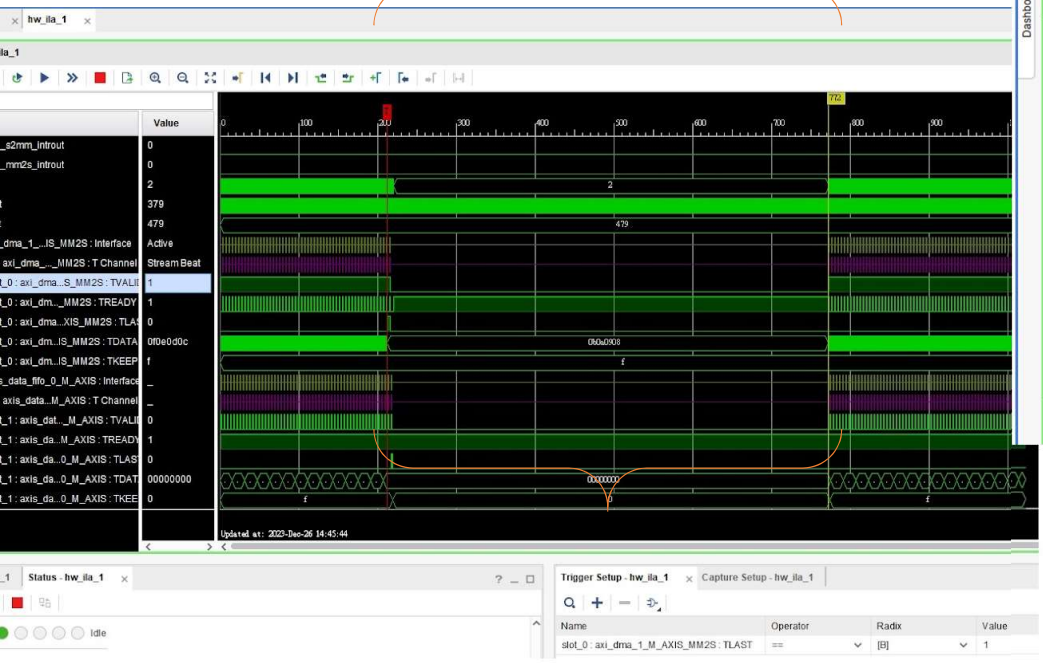
Name	handle_image(data)
Inputs	data: (string) fpga處理後的圖片
Outputs	result: (bool) True/False
Parameters	None
Methods	將影像處理過的圖片存到對應的queue，並回傳成功與否 Python: image = cv2.imread(“./test/test.png”) result = handle_image(image) print(result) Return: True

API-PC端

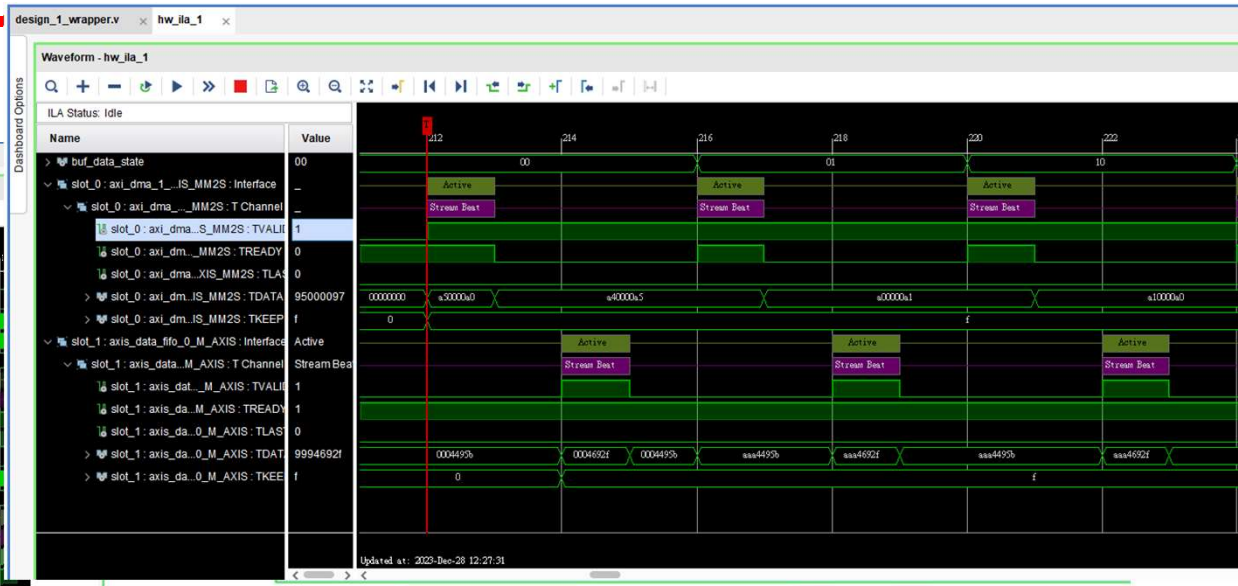
Name	handle_result(data)
Inputs	data: (string) fpga處理後的結果
Outputs	result: (bool) True/False
Parameters	None
Methods	將影像處理過的結果存到對應的queue，並回傳成功與否 Python: path = “./test/test.txt” with open(path, 'r') as file: content = file.read() result = handle_result(content) print(result) Return: True

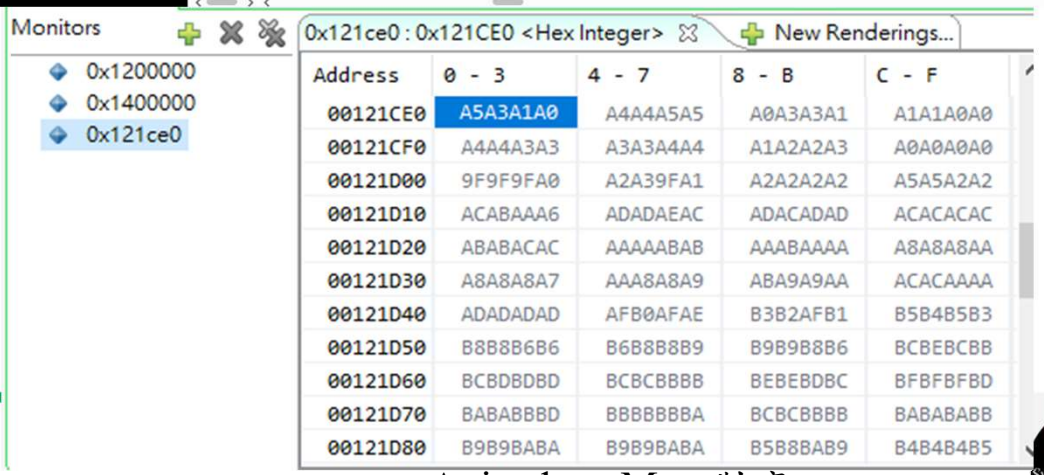
驗證- (PL端)axis介面

XDMA_busy約560 clock(5.6us)



AXI-stream介面交握正確
(多次burst)





Axis data. Mem對應

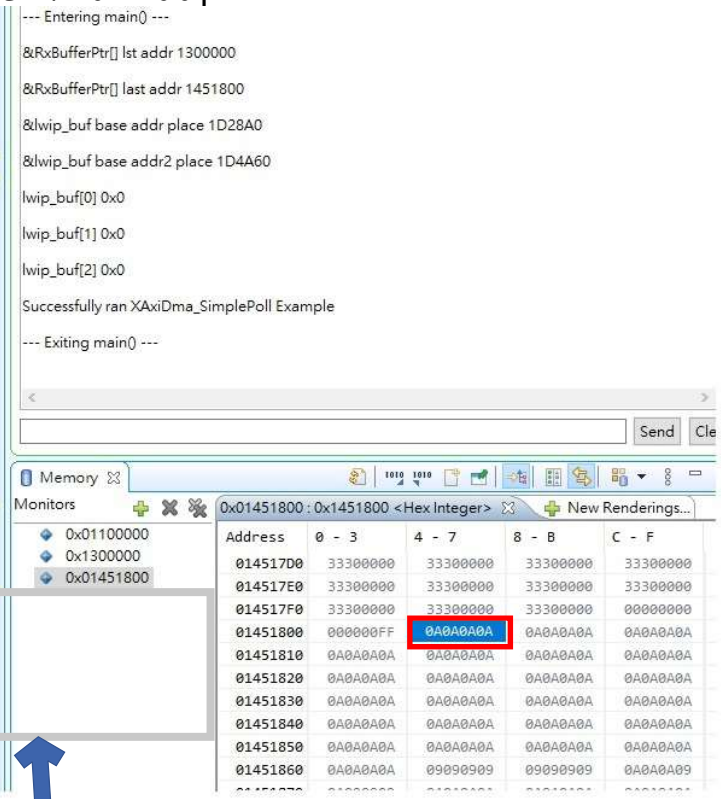
驗證- Zynq_ISR

以xil_print()驗DMA_RX、整理:

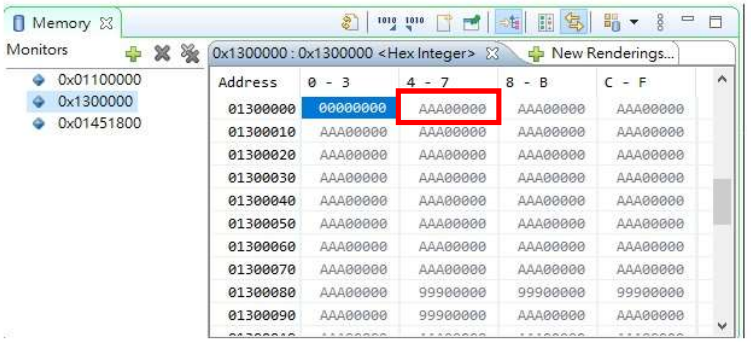
1- 將(u8) RX_buf_ptr[720*4]切分整理，前加(u32 row)

2-整理完住進lwip_buf[4+3*720+100]

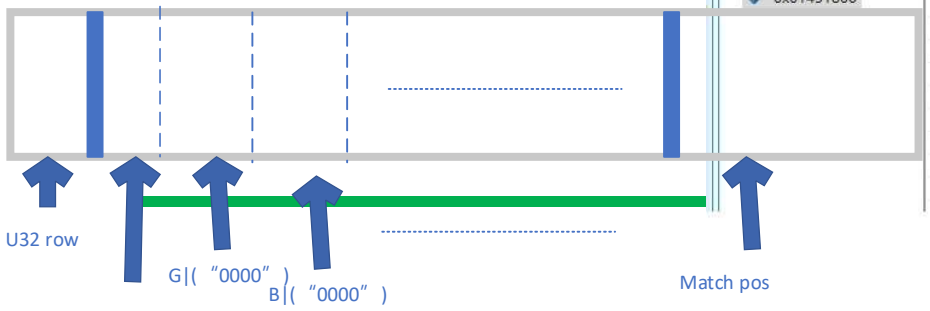
註:兩者都在DDR



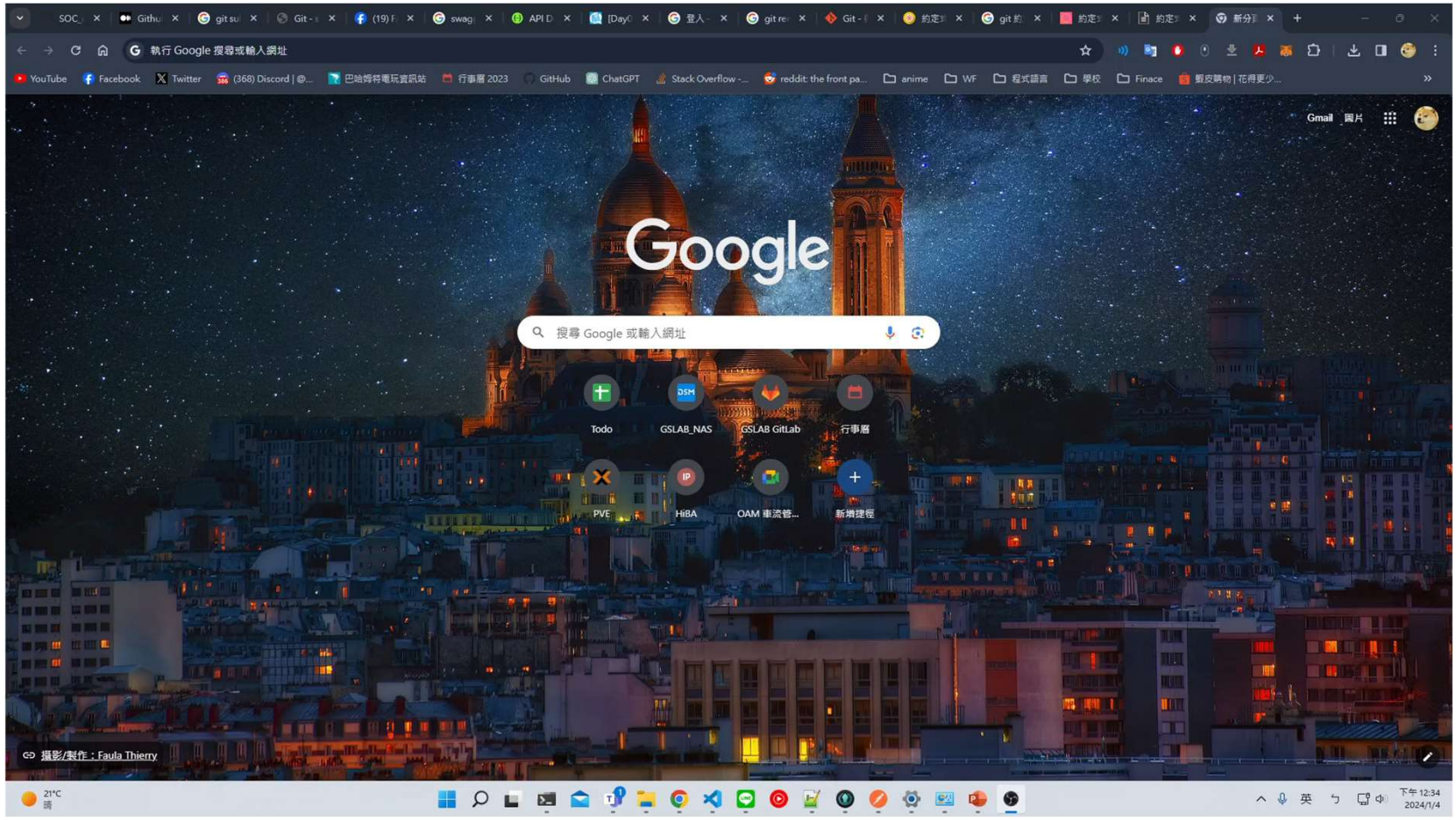
&lwip_buf[0]



&RX_buf_ptr[0]



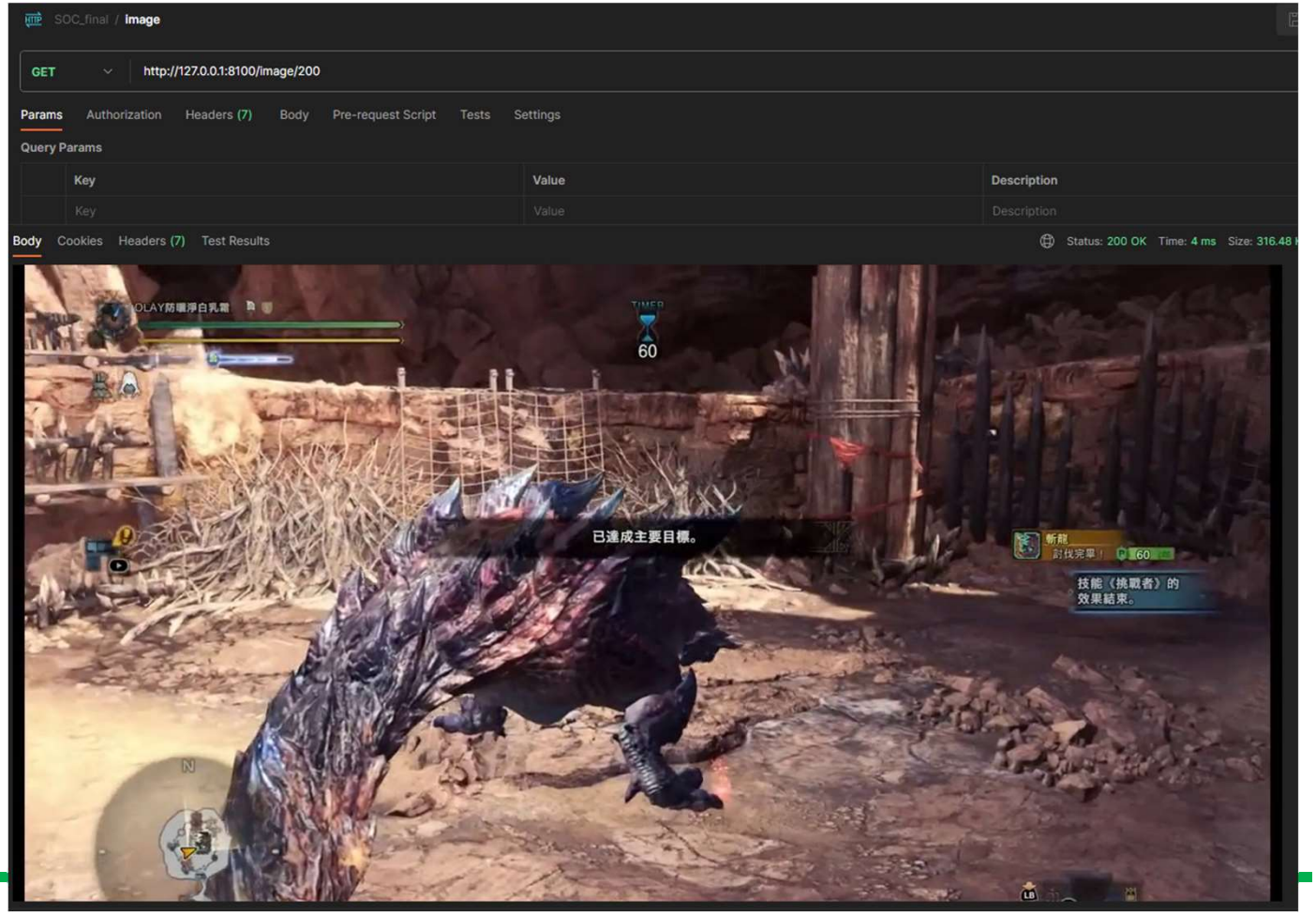
API-PC端 網頁撥放影片



API-PC端 Get /image/<frame>

預先放入1000幀做測試
Get /image/200

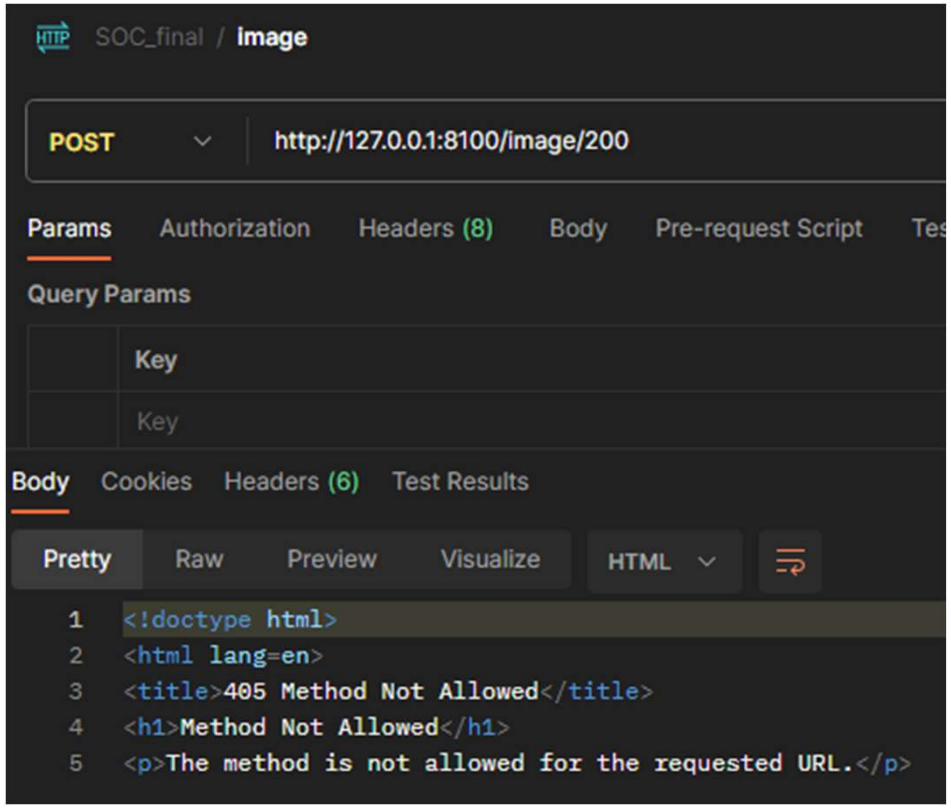
正常返回第200幀的圖



API-PC端 Get /image/<frame>

預先放入1000幀做測試
Post /image/200

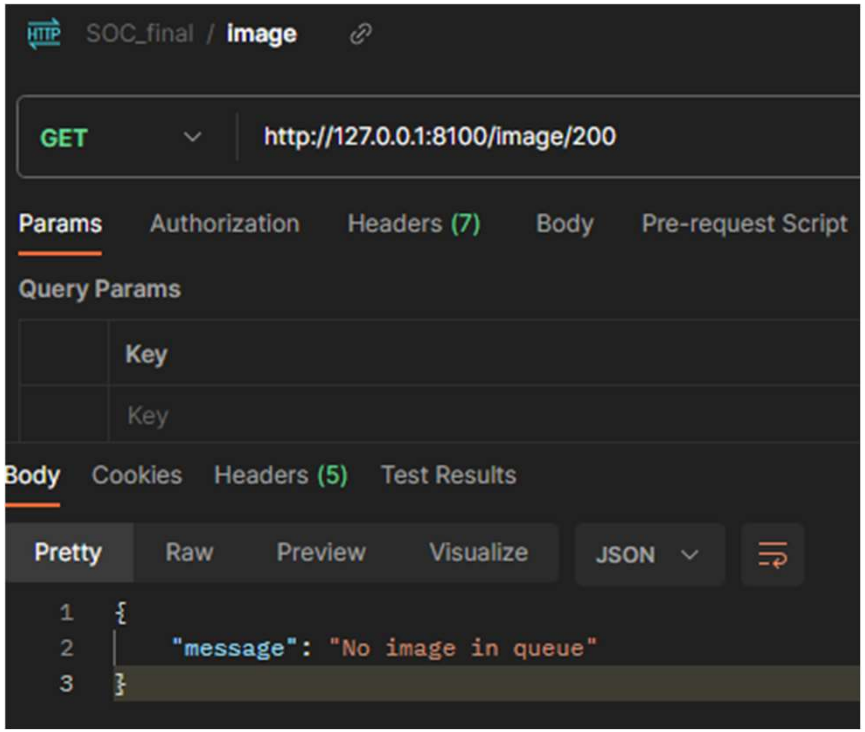
回傳方法錯誤



API-PC端 Get /image/<frame>

預先放入1000幀做測試
Get /image/200

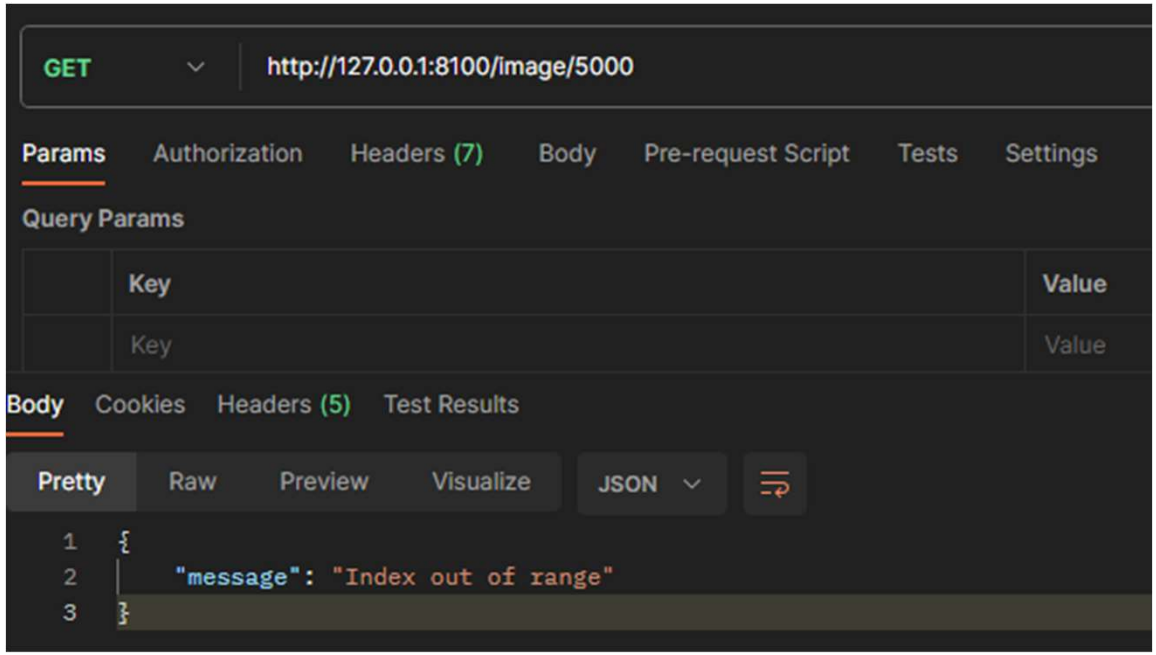
queue裡沒有資料時，不會回傳圖片



API-PC端 Get /image/<frame>

預先放入1000幀做測試
Get /image/5000

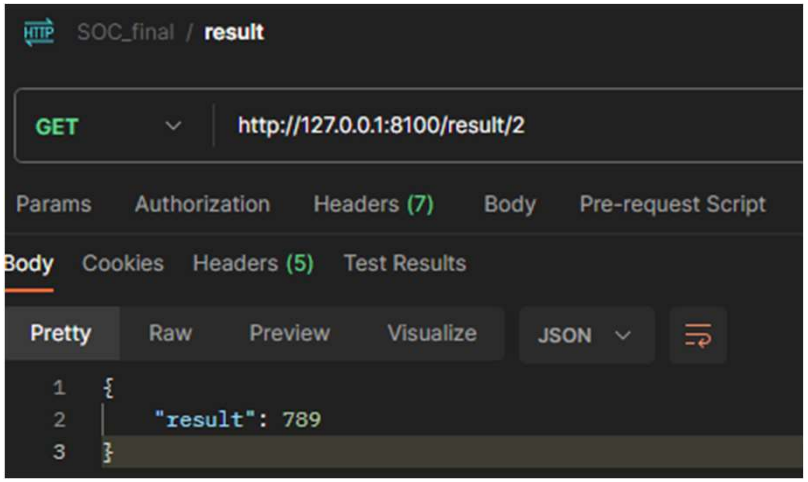
超過queue的大小



API-PC端 /result/<frame>

預先放入3筆測試資料
Get /result/2

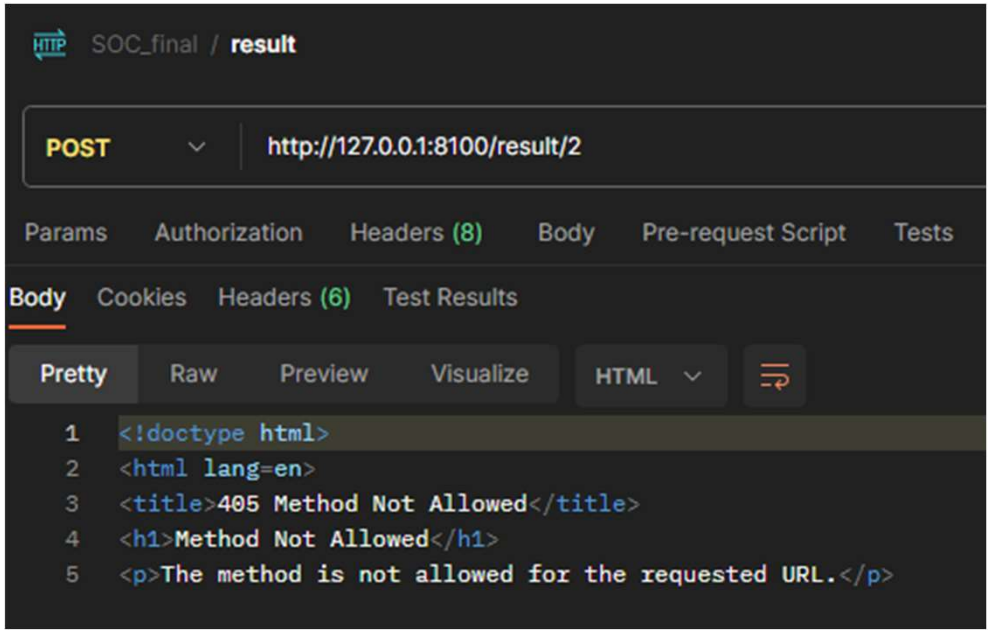
正常回傳該幀json



API-PC端

預先放入3筆測試資料
Post /result/2

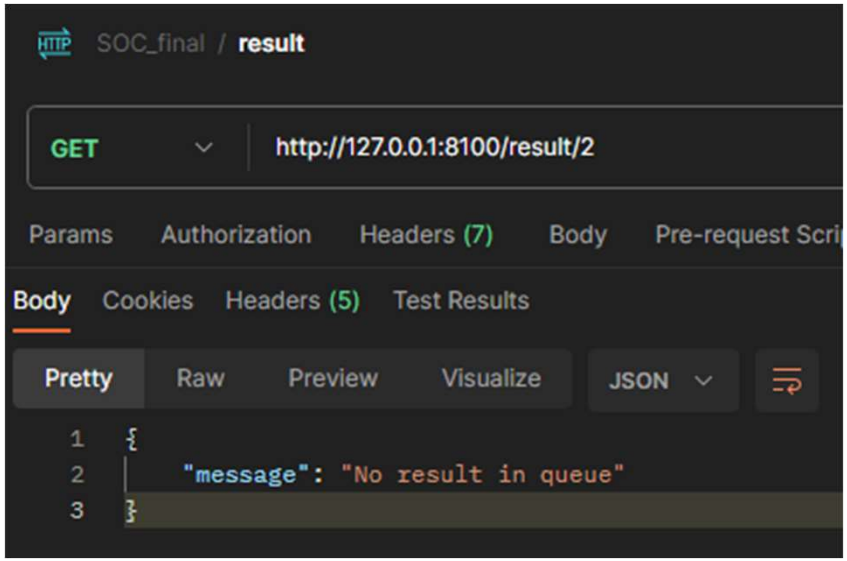
回傳方法錯誤



API-PC端

預先放入3筆測試資料
Get /result/2

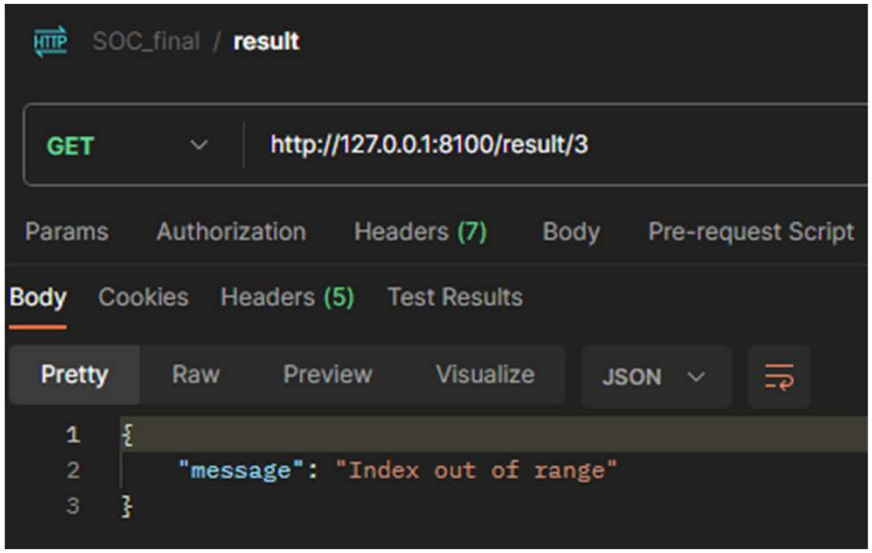
queue裡沒有資料時，不會回傳結果



API-PC端

預先放入3筆測試資料
Get /result/3

超過queue的大小



API-PC端