HDL HW3 報告

B023040050 顏義洋

第一部分：撰寫組合語言將RGB轉為YUV

檔案：YUV.v

R\*0.299：

LDR r0, data\_memory[0]

r3 = r0 >> 2

r4 = r0 >> 5

r5 = r0 >> 6

r6 = r0 >> 8

ADD r3,r3,r4

ADD r5,r5,r6

ADD r3,r3,r5

STR r3,data\_memory[12]

G\*0.589：

LDR r0, data\_memory[4]

r3 = r0 >> 1

r4 = r0 >> 4

r5 = r0 >> 6

r6 = r0 >> 7

ADD r3,r3,r4

ADD r5,r5,r6

ADD r3,r3,r5

STR r3,data\_memory[16]

B\*0.114：

LDR r0, data\_memory[8]

r3 = r0 >> 4

r4 = r0 >> 5

r5 = r0 >> 6

r6 = r0 >> 7

ADD r3,r3,r4

ADD r5,r5,r6

ADD r3,r3,r5

STR r3,data\_memory[20]

Y=R+G+B：

LDR r3, data\_memory[12]

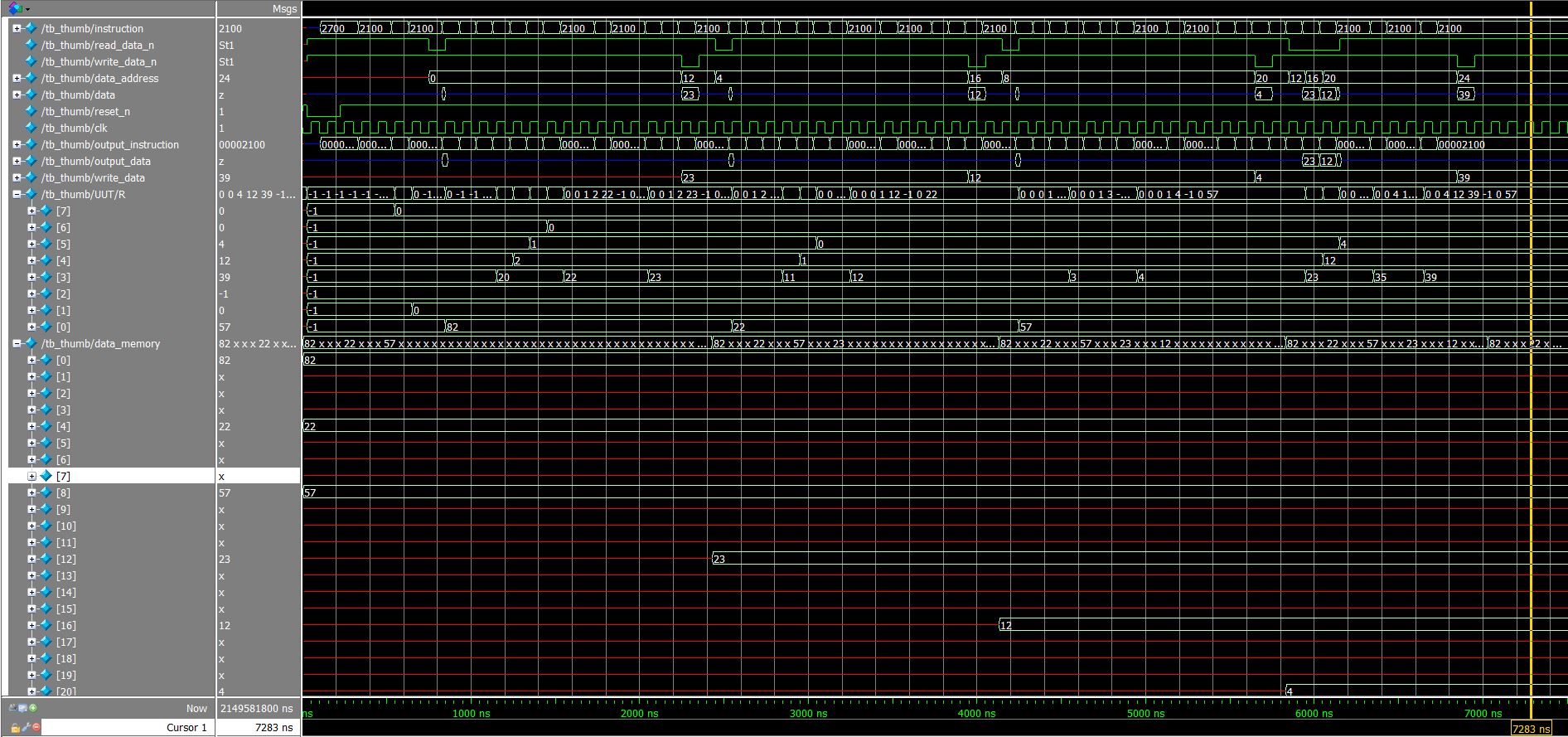
LDR r4, data\_memory[16]

LDR r5, data\_memory[20]

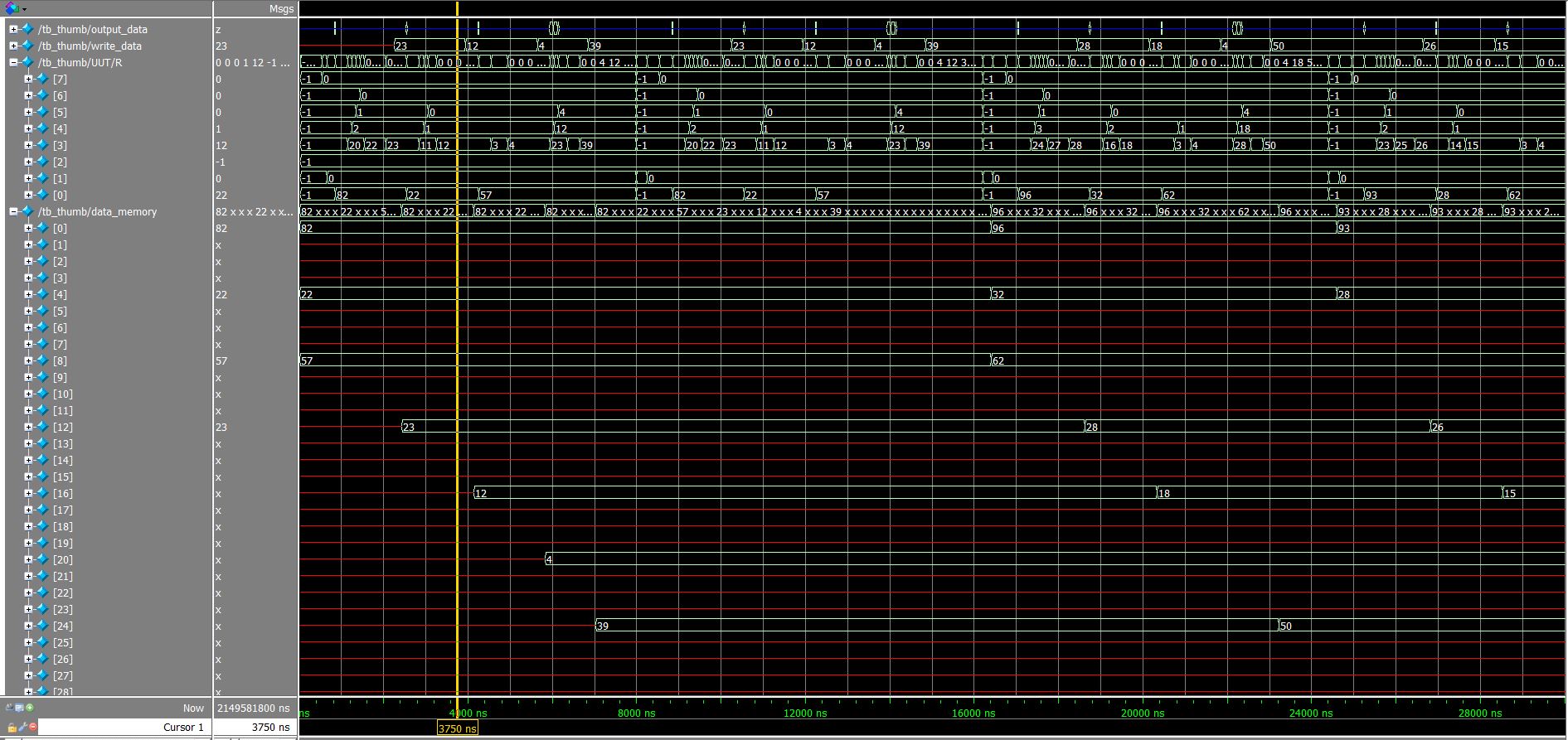
ADD r3,r3,r4

ADD r3,r3,r5

STR r3,data\_memory[24]



▲圖一



▲圖二

運算流程：

(1)把R load到r0，再對r0作數個shift分別存到不同reg，最後把總合存到r3

(2)把R\*0.299的近似值store到data\_memory[12]

(3)G\*0.589、B\*0.114的方法同(1)(2)，結果分別store到data\_memory[16]，data\_memory[20]

(4)計算Y = R\*0.299 + G\*0.589 + B\*0.114 存到data\_memory[24]

Testbench流程：

1. 開啟input的BMP，讀取header取得bmp的長跟寬還有檔案大小和pixel data的起始點。一次讀300個byte，一次放3個byte：把R、G、B分別塞到data\_memory[0]，data\_memory[4] data\_memory[8]
2. 等運算完再reset，此時Y值已經在data\_memory[24]
3. 一次寫3個byte的Y值進圖片檔
4. LOOP以上(2)~(4)的步驟直到讀檔結束

以上流程可以參考圖一和圖二。

第二部分：撰寫組合語言執行sobel運算

檔案：sobel.v

LDR r0, data\_memory[0]

LDR r1, data\_memory[4]

LDR r2, data\_memory[8]

LDR r3, data\_memory[12]

LDR r4, data\_memory[16]

LDR r5, data\_memory[20]

SUB r2,r3,r2;

SUB r0,r1,r0;

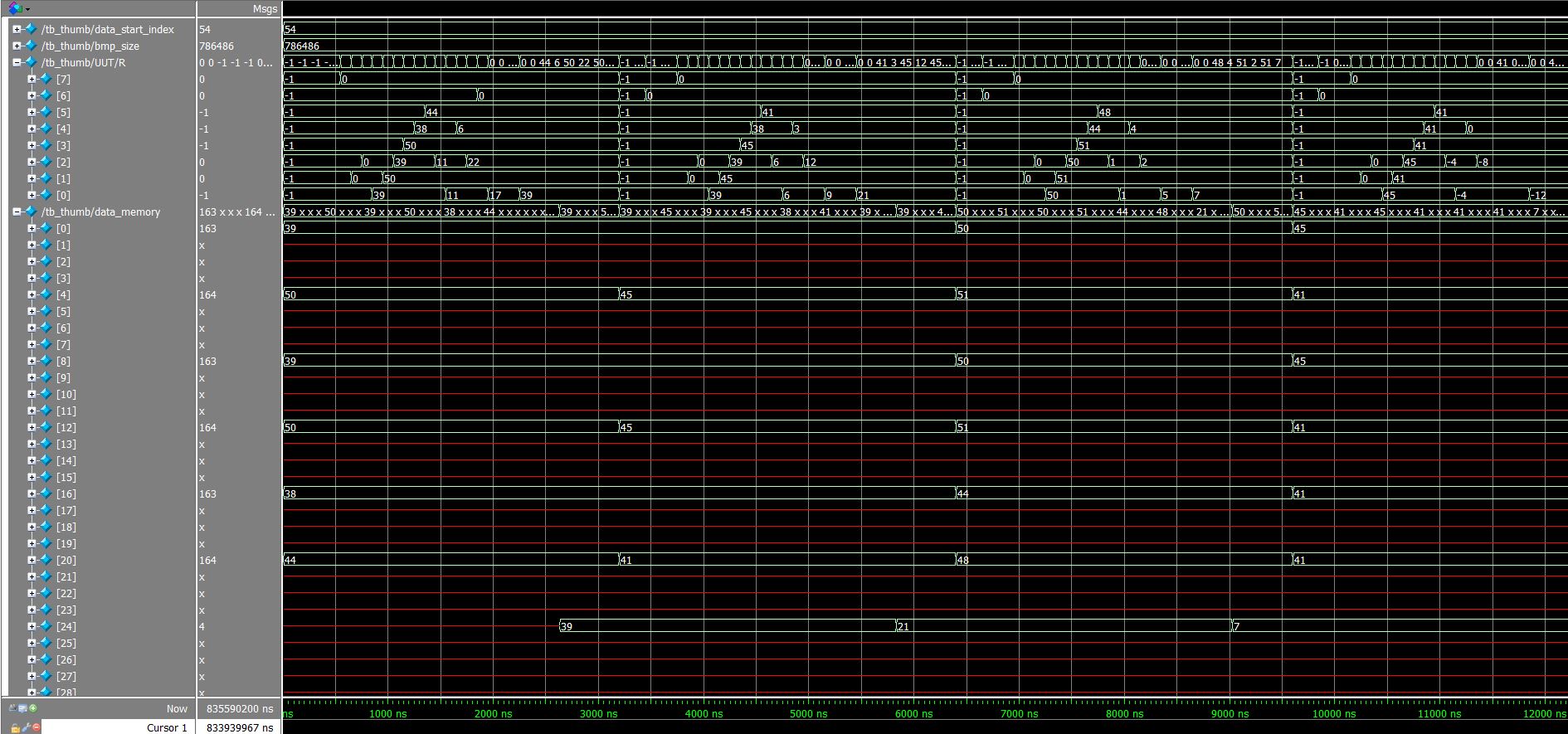
SUB r4,r5,r4;

LSL r2,#2

ADD r0,r0,r4

ADD r0,r0,r2

STR r0,data\_memory[24]



運算流程：



1. 將sobel所需要的Y值分別從data\_memory[0,4,8,12,16,20]load進r0~r5內
2. 利用加減和shift運算，算出所需要的梯度值
3. 把算出來的梯度值放入data\_memory[24]

Testbench流程：

1. 開啟input的BMP，讀取header取得bmp的長跟寬還有檔案大小和pixel data的起始點。
2. 開個很大個陣列把所有的BMP pixel的data先讀進來
3. 最外圍的那一圈pixel不做sobel運算，寫檔時直接寫黑色
4. 把相對應的pixel放進data\_memory
5. 運算完後在reset，此時sobel算出來的梯度已經在data\_memory[24]
6. 因為在thumb內沒有類似arm conditional execution的指令，所以不方便在thumb內就直接算出要寫白色還是黑色的pixel進圖片
7. 將出來的梯度取絕對值
8. 選定64為標準，梯度大於64寫白色(255,255,255)進圖片，反之填入黑色(0,0,0,)
9. 用兩層LOOP執行以上(3)~(8)的步驟直到所有的pixel都算完

第三部分：

刪除thumb.v裡面在此份作業sobel運算用不到的指令，我將ADD、SUB、LDR、STR、MOV系列之外的程式碼刪除，減少約300行左右

再用同一份testbench驗證所得到的結果與原本相同

比較原始與優化的thumb processor所合成出來的timing和area

Area比較：

原始thumb.v(檔案thumb\_area.txt)

Number of ports: 117

Number of nets: 6154

Number of cells: 5398

Number of combinational cells: 4612

Number of sequential cells: 768

Number of macros/black boxes: 0

Number of buf/inv: 375

Number of references: 62

Combinational area: 93778.330076

Buf/Inv area: 3119.616081

Noncombinational area: 20509.286325

Macro/Black Box area: 0.000000

Net Interconnect area: 556216.691343

Total cell area: 114287.616402

Total area: 670504.307744

刪減後modified\_thumb(檔案modified\_thumb\_area.txt)

Number of ports: 117

Number of nets: 3841

Number of cells: 3394

Number of combinational cells: 2665

Number of sequential cells: 719

Number of macros/black boxes: 0

Number of buf/inv: 191

Number of references: 48

Combinational area: 46167.552201

Buf/Inv area: 1184.256031

Noncombinational area: 19264.204739

Macro/Black Box area: 0.000000

Net Interconnect area: 372709.291742

Total cell area: 65431.756940

Total area: 438141.048681

Timing Path比較：

原始thumb.v(檔案thumb\_timing.txt)

------------------------------------------------

thumb tc8000 saed90nm\_typ

Point Incr Path

-----------------------------------------------------------

write\_data\_n\_reg/CLK (DFFASX1) 0.00 0.00 r

write\_data\_n\_reg/QN (DFFASX1) 0.24 0.24 f

data\_tri[15]/INOUT2 (BSLEX1) 0.28 0.52 f

data[15] (inout) 0.00 0.52 f

data arrival time 0.52

-----------------------------------------------------------

(Path is unconstrained)

刪減後modified\_thumb(檔案modified\_thumb\_timing.txt)

------------------------------------------------

thumb tc8000 saed90nm\_typ

Point Incr Path

-----------------------------------------------------------

DR\_reg[31]/CLK (DFFX1) 0.00 0.00 r

DR\_reg[31]/Q (DFFX1) 0.22 0.22 f

data\_tri[31]/INOUT2 (BSLEX1) 0.78 1.00 f

data[31] (inout) 0.00 1.00 f

data arrival time 1.00

-----------------------------------------------------------

(Path is unconstrained)

第四部分：

利用DV合成之thumb proceesor的gatelevel檔案用相同testbench進行驗證，這部分沒有成功，原因可能是因為timing delay給的不夠正確，也有可能是作者原本提供的testbench沒有考量到硬體的delay層面。

第五部分：

利用FPGA驗證，成功合成出gatelevel檔案，但是用同一個testbech進行驗證也是錯誤。

寫作業遇到的所有問題or心得：

1. 一開始誤會題目的意思，以為所有的步驟包含開檔寫檔都要用組合語言寫，經由詢問助教後才知道只有運算的部分，運算部份的組語對我們並沒有很困難。
2. 雖然在計算機組織學過hazard的觀念，不過在寫組語丟進thumb跑的時候並沒有想到要考慮hazard的問題，所以有遇到data hazard的問題，還有遇到structural hazard的問題，不能有連續兩行做LDR/STR，解決方式都是插入不重要的指令在中間來避免hazard。
3. 合成出來的thumb processor用相同的testbench怎麼改都跑不出結果，通常在第一個指令要WB的那一個clock正緣，thumb內部的regfile就變成紅線，原因不明。嘗試過將Clock PERIOD1, READ\_DELAY, WRITE\_DELAY, STABLE\_TIME的延遲時間都加長到10倍，結果仍是錯誤。