

team01

Final project report

用FPGA實作指紋辨識

B09901026何式功

B09901030藍照淇

B09901035陳亮瑜

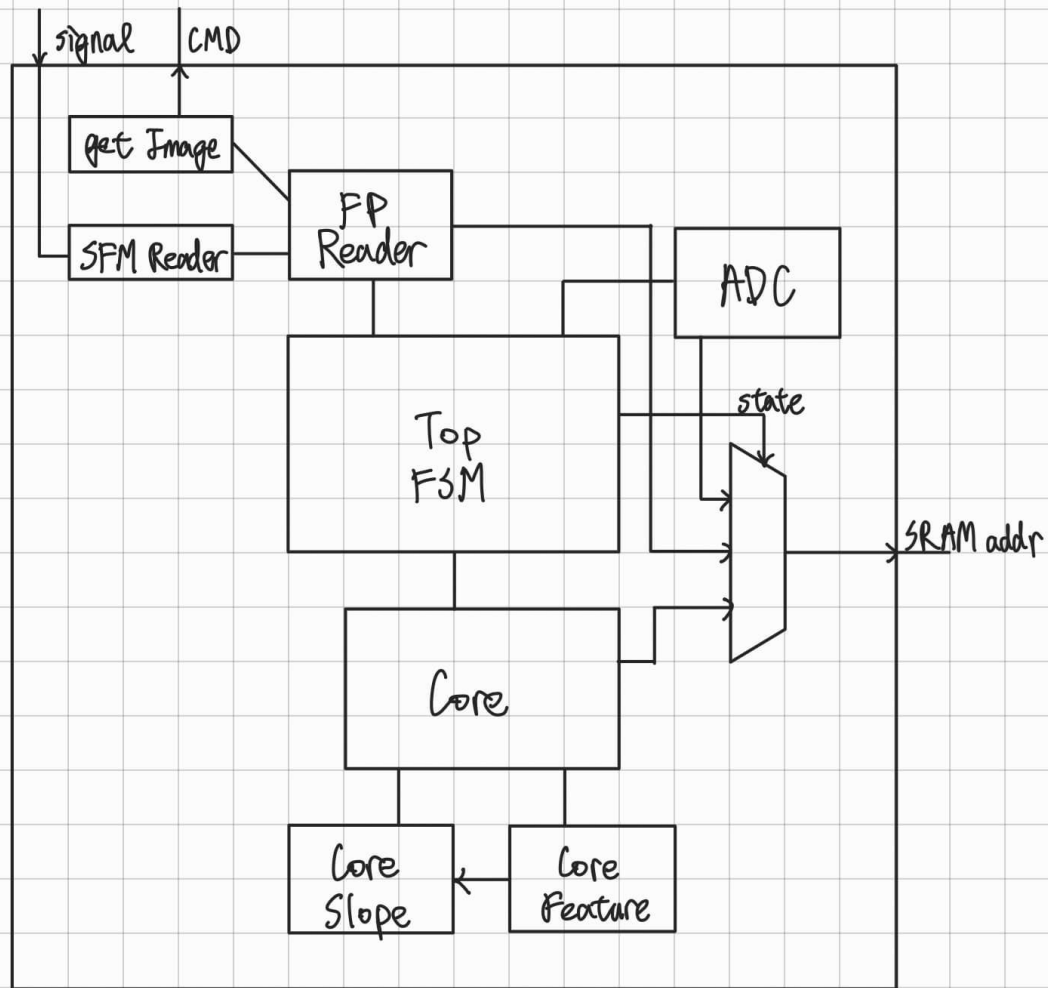


Intro

簡介



sfm-V1.7



Block Diagram



Algorithm

演算法

Slope algorithm 斜率比較原理

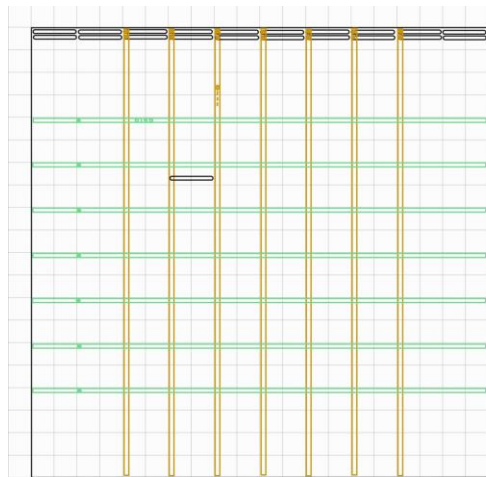
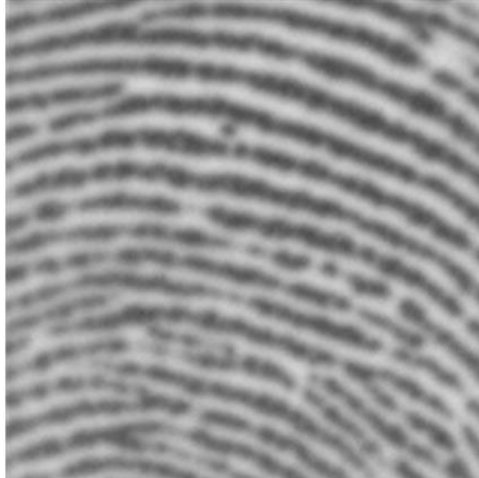
透過比對指紋的**斜率**判斷是否為同一個指紋

在每張指紋上選擇了7組水平線、鉛直線，分別位

於第32/48/64/80/96/112/128個row/col。

將上面XOR=1的次數分別加總，則斜率為

鉛直線上的(ver_cnt) / 水平線上的(hor_cnt)



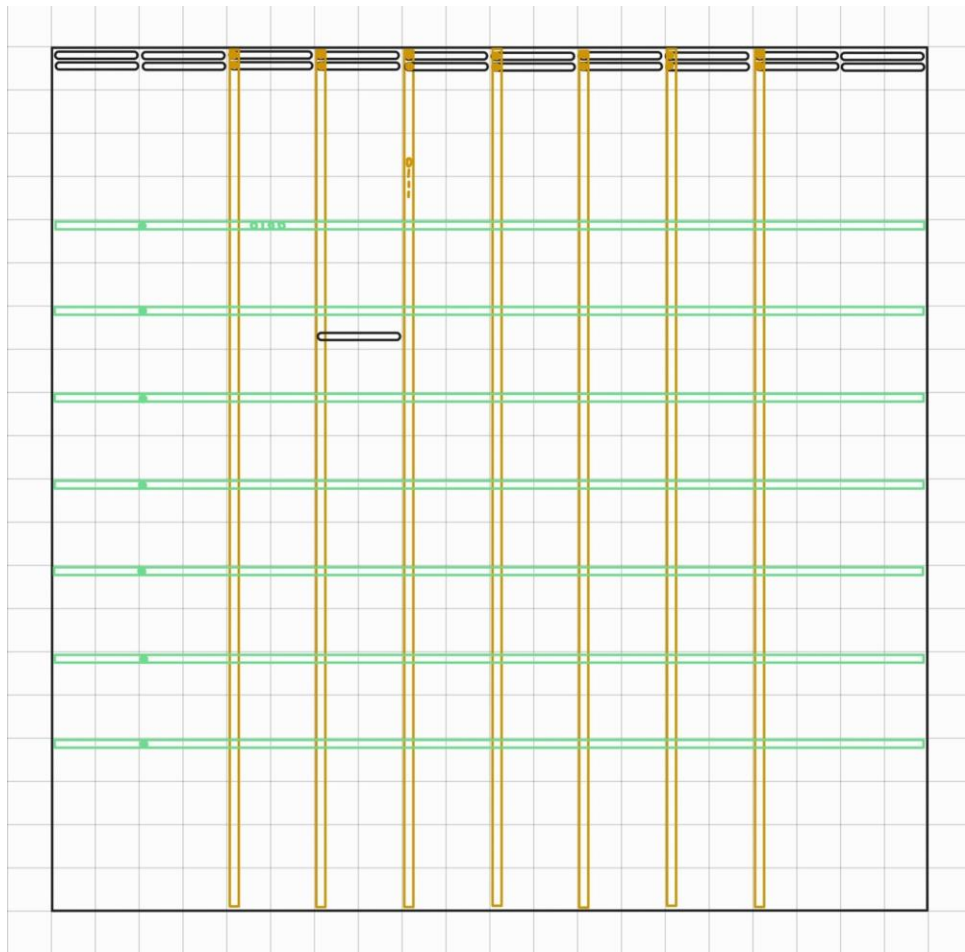
另外，以14個bit分別記錄這14條線上，
前一個

pixel的值，以便跟當前的pixel比較。

每當讀入新的16bit(一個SRAM address的資料)，確認這些pixel在指紋中的位置後，計為[0:7] hor_val, ver_val

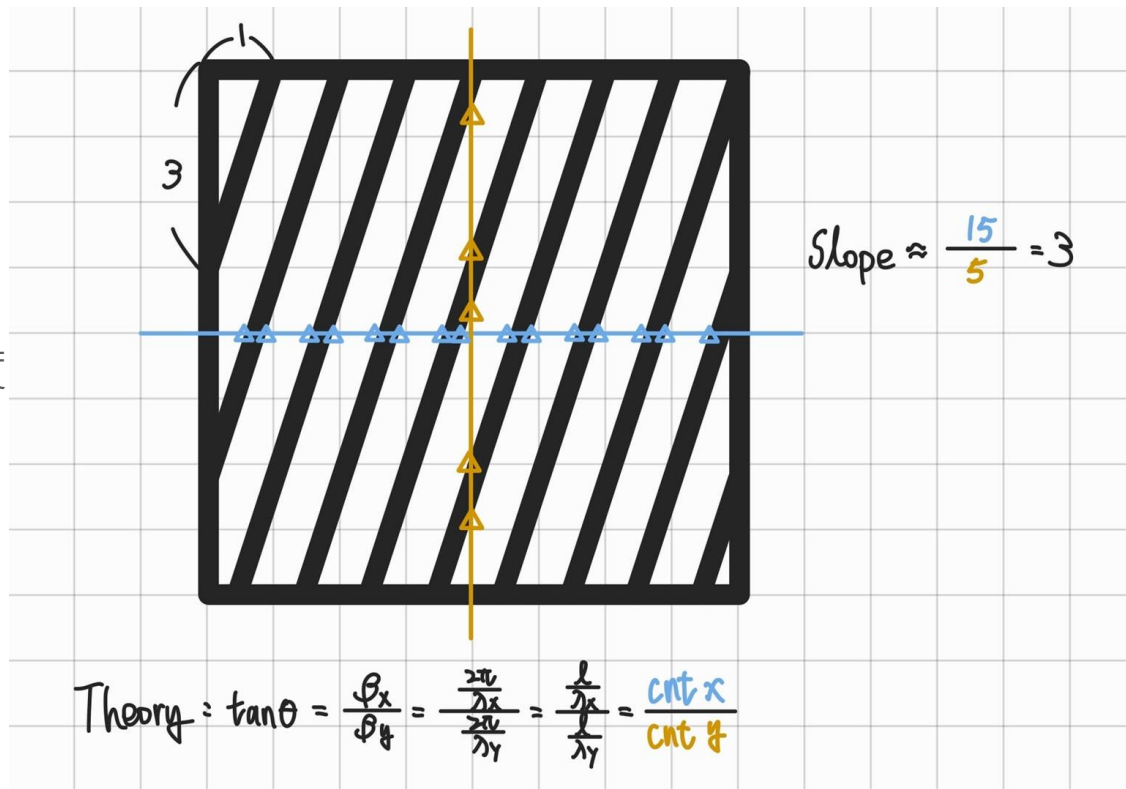
比較(若有)對應到的hor_val and/or ver_val，更新hor_cnt and/or ver_cnt。

然後覆寫hor_val and/or ver_val



利用 $\tan(\theta) = \frac{\text{apparent wavelength}}{\text{fixed length}}$ 相除

apparent wavelength 反比固定長度
中，顏色變換的次數




變換：


xor(pre-request:threshold)

正負號：

左上到右下的#xor > 右上到左下的
#xor

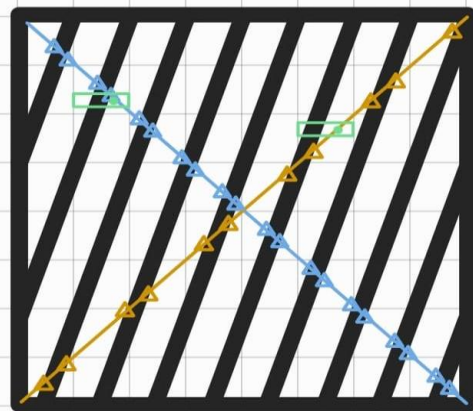
let col = 0~9 (each col = 16 lines) , 左上 (0,0)
row = 0~159

 : $r = C \times 16 + k \rightarrow k = r - C \times 16$

 : $159 - r = C \times 16 + k \rightarrow k = 159 - r - C \times 16$

if $k \in [0, 15]$, the address corresponds to
(r, c) has data we need at `i_data[k]`

Slope Sign



$\triangle = 20$

$\triangle = 11$

{ $\triangle > \triangle$: slope 為正
 $\triangle < \triangle$: slope 為負

Feat algorithm特徵比較原理

透過比對指紋的**特徵**判斷是否為同一個指紋

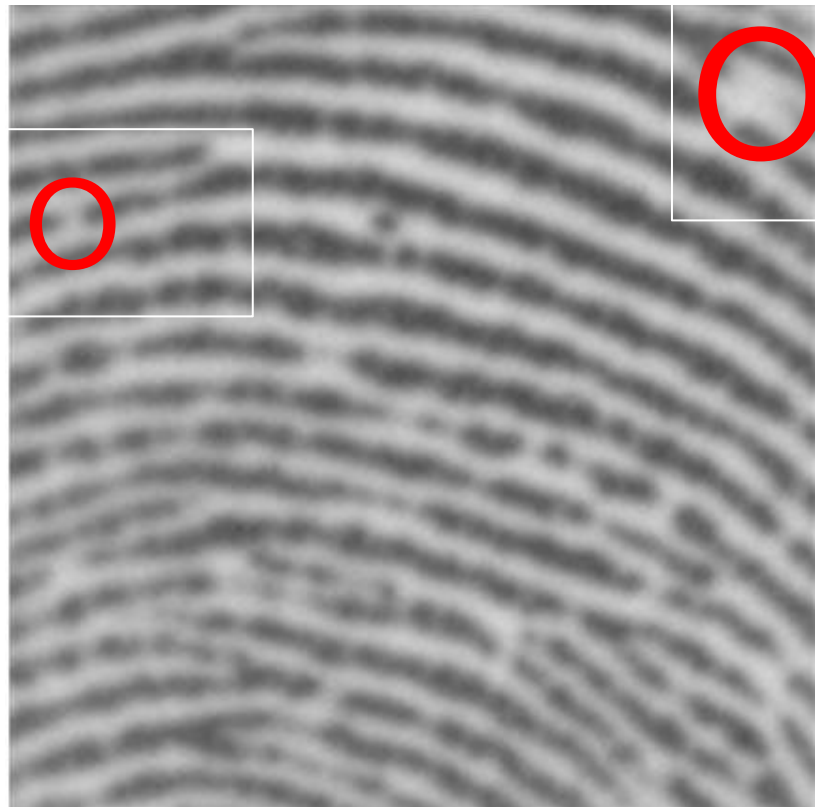
1. 截取斷點：

若該點附近25宮格有超過11個黑點

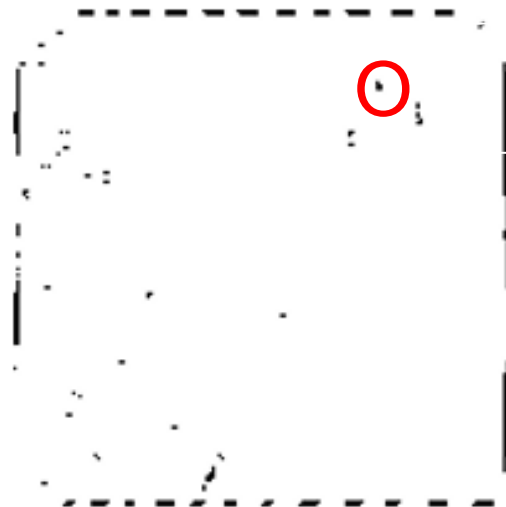
判斷該點非斷點。

原因：斷點落在一條黑線的末端，

周圍約僅1/4的點是黑點。



用MATLAB擷取到的斷點



2.pooling：將特徵圖由 $160*160$ 降為 $30*30$

捨棄特徵圖邊界5格(邊界可能會出現誤判)後，每25宮格轉為一格。

判斷條件：若該25格中，超過一個特徵點則設為1，否則為0。

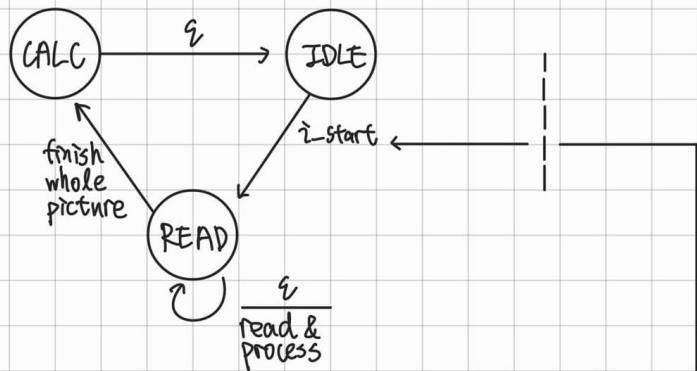
原因是，原圖的一條黑線上，可能出現一個凸點，可能被判斷為特徵點。

但是這類的點會落單，因此pooling可以消除他們，保留成群的特徵點。

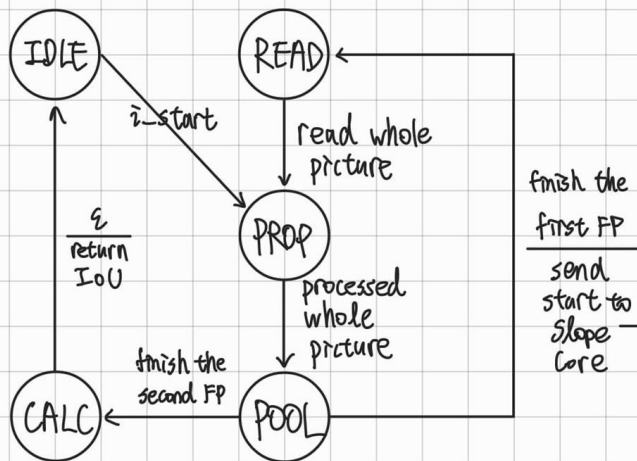
另外，pooling可以減少稍微減少按壓時未對齊的狀況，增加判斷彈性。

3.IoU：Intersection over union。判斷相似度時，我們採用交集除以聯集這個常見的標準，用以加重懲罰相異的部分。

Slope FSM

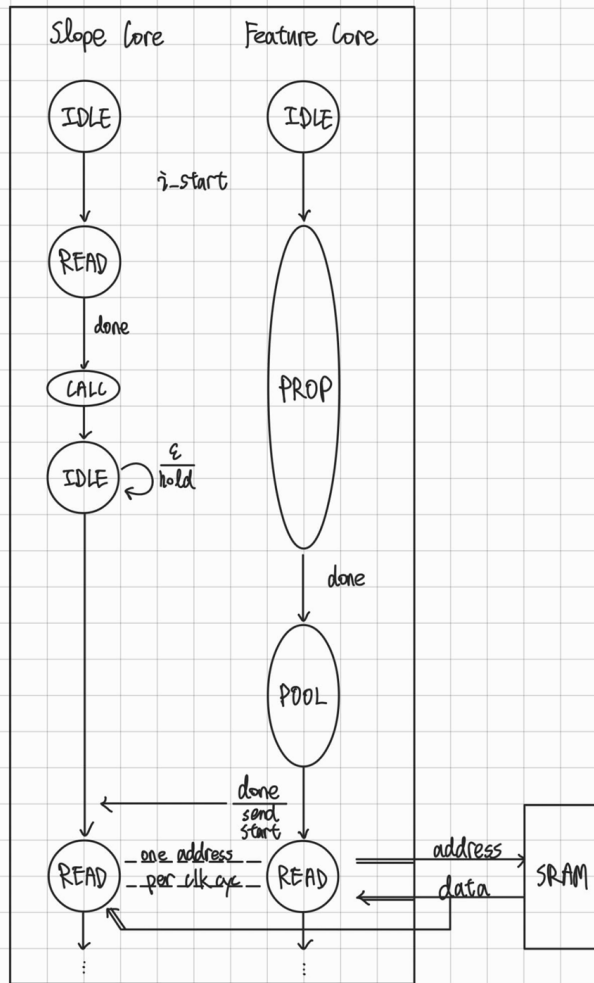


Feature FSM



Core設計

State Alignment





Design

設計細節

指紋資料用 160×160 的array
存在SRAM

Fingerprint Array Framework 160×160 matrix



Submodule基本功能介紹

1. FP_reader

FPGA通過UART protocol傳送讀取圖片的指令給指紋模組，模組指紋回傳讀到的指紋

1. ADconvert

指紋模組回傳的圖片，每個pixel以0~255表示影像強度(類比)，這個submodule的功能就是將類比資料轉換為數位

3. Core

執行指紋辨識的演算法，分成特徵比較法和斜率比較法

FP_reader.sv設計流程

step 1 在能夠傳輸application layer data之前，需要讓兩端的Baud Rate相同，因此採用Qsys生成115200的clk接在FPGA中的FP_Reader module。

step 2 為確保bit layer protocol相同，我們使用示波器讀取指紋讀取機的信號，並比對說明書上的指令。

FP_reader.sv設計流程

step 3 依照讀取機制定的application layer protocol，決定所需的指令集，包含initialization以及get fingerprint image等基本指令。

step 4 製作UART溝通模組，以byte為單位讀取、寫出指令。
製作控制模組，計算已接收的byte數並解析訊息，從中解析出指紋原圖。同時寫入SRAM，以減少暫存邏輯單元的消耗

實作FP_reader.sv時碰到的問題1

遇到的問題：

板子的**baud rate**不精確(和指紋模組無法完美對應)。

可能的原因：

DE2_115板子所提供的3.3V並非精確、穩定數值，接上

模組之後，電壓稍微改變，導致模組中自帶的震盪器無法
恰好提供115200的**baud rate**。

修正方式：

FP_Reader改為使用不同**baud rate**。經測試，116800為
最佳解，可使傳輸過程錯誤率最低。

實作FP_reader.sv時碰到的問題2

主要問題：

每個人每次按壓感應器的力度都不相同，因此整張圖片往往偏暗或偏亮。如果簡單的以亮度128為分界，會出現整張指紋全黑全白的狀況

解法：

亮度的分界線由整體亮度的平均決定。

衍生問題：

靠近邊角的地方，可能因為沒有被手指覆蓋，出現全白的狀況。該區平均就過高，導致稍黑的點就被判斷為黑色。

修正方法：

切塊只在中間進行，周邊的區塊則使用，離該區最近的切塊所算出的平均值為門

ADconvert.sv功能：比較器

ADconverter不需要額外分配記憶體邏輯，從SRAM讀出資料的同時，就會立即判斷該pixel所對應到的門檻值為何，並將16bit(2 pixel)轉成2bit輸出。

Core.sv設計：矩陣操作

1. 斜率模組因為只需看與前一個Xor，所以從SRAM讀值的同時可以同時完成計算，立刻將用過的input視為don't care，因此整個模組 $O(1)$ 的記憶體空間，並且讀完整張指紋的瞬間，計算就完成了。
2. 特徵模組因為步驟1.同時需要讀取原圖，寫入特徵圖，因此需要每張指紋至少需要兩個 $160*160$ (完整大小)的register。但是為避免fpga的邏輯單元不足，我把兩個指紋分開成兩次處理，當第一個圖完成pooling後(大小 $30*30$)，將他原圖、特徵圖的register拿來操作第二張圖的前兩步驟。省下兩個 $160*160$ 的空間。

Core.sv設計：矩陣操作

3. Parallel process and state alignment between two cores.

為了讓執行速度加倍，我讓兩個core可以並行處理，但問題是SRAM同時只支援一個address的讀寫，因此兩個core必須洽好在同一個clock讀取同一個SRAM位置。作法：由於特徵模組的執行時間較長，所以可以用它完成第一個指紋的時間，當作開始讀取下一輪SRAM的基準。於是我把它的state外接到控制兩個core的wrapper，讓wrapper藉此控制斜率模組的開始信號，經過進一步的對齊，兩個模組就會在同一個clk,要求同一個SRAM信號。



DEMO

成果展示





Reflection

心路歷程

一開始選上數位電路實驗時，我們都很好奇SystemVerilog是怎麼用程式語言描述數位電路，當我們開始研究always_comb和always_ff這兩個語法時，才驚奇的發現他們就是在描述我們大一交電學過的combinational logic和sequential logic，lab1算是小試身手，透過在always_comb中實作的state machine我們可以製作多種功能，接著是有趣的lab2，透過rs232介面我們實現電腦端和FPGA端的溝通，並且第一次感受到透過FPGA進行解碼的運算，確實可以比只用軟體例如C++或python處理得更快。在lab2的bonus中，我們有嘗試過使用藍芽代替Rs232的傳訊功能，雖然最後沒有完成，但是透過示波器親自去確認訊號波形的過程對我們後來的期末專題也有很大幫助。lab3我們接觸到了I2C 這個新的protocol，以及如何使用FPGA上的SRAM存取和讀取資料。

最後的lab4讓我們真正地體驗到了硬體運算速度的優勢，透過取用FPGA上更多的運算資源，很多之前在軟體中必須依序處理的資料我們可以平行處理，讓時間複雜度下降了一個等級。Smith-Waterman序列比對的演算法也是很實用的知識。期末專題我們選擇做指紋辨識，在實作透過Uart傳送指令給指紋模組並讀取圖片的過程中，我們運用到了之前研究I2C和藍芽得到的知識。研究指紋比對演算法的過程中，我們也參考了Smith-Waterman的一些想法。之前對SRAM的研究也讓我們在存取指紋時少了不少障礙。過去的努力往往會在未來不經意的時刻收成。我們組中也有組員因為這次課程，對數位電路燃起熱忱。也希望還能有用數位電路做專題的機會，就算沒有機會，也希望以後人生的每一段時光都能如這學期一般充實。

層級架構

DE2115.sv



Top.sv



Reader.sv

ADconvert.sv

Core.sv



getImage.sv

FP_Reader.sv



Core_Feat.sv

Core_Slope.sv

Table of Contents

- Flow Summary
- Flow Settings
- Flow Non-Default Global Settings
- Flow Elapsed Time
- Flow OS Summary
- Flow Log
- > Analysis & Synthesis
 - ▼ Fitter
 - Summary
 - Settings
 - Parallel Compilation
 - I/O Assignment Warnings
 - Ignored Assignments
 - > Incremental Compilation Section
 - Pin-Out File
 - > Resource Section
 - > I/O Rules Section
 - Device Options
 - Operating Settings and Conditions
 - > Estimated Delay Added for Hold Timing
 - Messages
 - Suppressed Messages
 - Flow Messages
 - Flow Suppressed Messages
 - > Assembler
 - > TimeQuest Timing Analyzer

Fitter Summary

Fitter Status	Successful - Thu Jun 09 15:11:25 2022
Quartus II 64-Bit Version	15.0.0 Build 145 04/22/2015 SJ Full Version
Revision Name	fingerprint
Top-level Entity Name	DE2_115
Family	Cyclone IV E
Device	EP4CE115F29C7
Timing Models	Final
Total logic elements	63,343 / 114,480 (55 %)
Total combinational functions	63,294 / 114,480 (55 %)
Dedicated logic registers	26,080 / 114,480 (23 %)
Total registers	26080
Total pins	517 / 529 (98 %)
Total virtual pins	0
Total memory bits	0 / 3,981,312 (0 %)
Embedded Multiplier 9-bit elements	4 / 532 (< 1 %)
Total PLLs	1 / 4 (25 %)



Multicorner Timing Analysis Summary

	Clock	Setup	Hold	Recovery	Removal	Minimum Pulse Width
1	▼ Worst-case Slack	-59.416	-0.001	8559.140	0.486	9.400
1	CLOCK2_50	N/A	N/A	N/A	N/A	16.000
2	CLOCK3_50	N/A	N/A	N/A	N/A	16.000
3	CLOCK_50	N/A	N/A	N/A	N/A	9.400
4	clk1 altpll_0 sd1 pll7 clk[1]	-2.752	-0.001	8559.140	0.486	4280.539
5	clk1 altpll_0 sd1 pll7 clk[2]	-59.416	0.180	N/A	N/A	41.362
2	▼ Design-wide TNS	-749.703	-0.001	0.0	0.0	0.0
1	CLOCK2_50	N/A	N/A	N/A	N/A	0.000
2	CLOCK3_50	N/A	N/A	N/A	N/A	0.000
3	CLOCK_50	N/A	N/A	N/A	N/A	0.000
4	clk1 altpll_0 sd1 pll7 clk[1]	-333.744	-0.001	0.000	0.000	0.000
5	clk1 altpll_0 sd1 pll7 clk[2]	-415.959	0.000	N/A	N/A	0.000