數位邏輯設計-Final Project 鄭凱文 104062223

電梯功能介紹：

一共四層樓，一開始停在1F，門為關閉，有人按了{D4,U3,D3,U2,D2,U1}，電梯會移動該樓層，打開門，讓使用者按下{F4, F3, F2, F1}決定目的地。當一次有多個人按按鈕時，有4種condition(K1~K4)決定先往上/往下、開門/關門，再利用這些condition判斷電梯的運行狀態(S0~S14，共15種state)。

Primary Input:

set\_goal [3:0] : 由{F4, F3, F2, F1}等input組成

set\_call [5:0]: 由{D4,U3,D3,U2,D2,U1}等input組成

Primary Output:

get\_goal[3:0]: 由{F4\_led, F3\_led, F2\_led, F1\_led}等output組成，代表外部按鈕的LED燈

get\_call [5:0]: 由{D4\_led, U3\_led, D3\_led, U2\_led, D2\_led, U1\_led}等output組成，代表內部按鈕的LED燈。

各功能模組：

Module 1 indoor\_call: 控制裡面的按鈕呼叫

<當進入到此module，代表有人按了裡面的按鈕>

Input:

set\_goal [3:0] : 由{F4, F3, F2, F1}等input組成，當收到這些輸入且非cur\_Floor，可記錄此目的地

rst: 歸零

ok: 確認抵達cur\_Floor且停止，其值為1，會將set\_goal的cur\_Floor值清除

cur\_Floor: 傳入目前樓層

Output:

get\_goal[3:0]: 由{F4\_led, F3\_led, F2\_led, F1\_led}等output組成，輸出到達此樓層後仍未完成的目的地(亮著即代表還沒達成要求)。

Module 2 outdoor\_call: 控制電梯外面的按鈕呼叫

<當進入到此module，代表有人按了外面的按鈕>

Input:

set\_call [5:0]: 由{D4,U3,D3,U2,D2,U1}等input組成，當收到這些輸入且非cur\_Floor，可記錄此呼叫

rst: 歸零

ok: 確認抵達cur\_Floor且停止，其值為1，會將set\_goal的cur\_Floor值清除

cur\_Floor: 傳入目前樓層

Output:

get\_call [5:0]: 由{D4\_led, U3\_led, D3\_led, U2\_led, D2\_led, U1\_led}等output組成，輸出到達此樓層後仍未完成的呼叫(亮著代表還沒處理外面上下樓的要求)

Module 3 cur\_floor: 輸出、更改現在的樓層

<接受訊號來決定下一個樓層往上/下>

Input:

up\_Floor, down\_Floor: 決定要向上or下

Output:

Cur\_Floor: 目前電梯所在樓層

<以下兩個為上下/停止與否的決定輸出>

Module 4 up\_down: 決定電梯上下，輸出目前停or向上or向下

Input:

Up\_d: 1→向上or歸零 ; 0→向下

Output:

out\_state\_dir: 輸出電梯上下狀態

Module 5 stop\_run: 決定電梯停止or運作，輸出目前運作與否

Input:

Stop\_r: 0→運作中; 1→停止(hold)

Output:

out\_state\_run: 輸出電梯停止狀態

Module 6 Counter: 計算動作改變間的週期

<此間接產生的ok決定其他input要求是否已經被執行>

Input:

ok\_count=1: 表示倒數開始 ; =0: 表未倒數，將count=預設值

Output:

Consignal=1: 表示倒數結束的訊號，control就會執行下個state

Module 7 condition judge: 電梯決定往上抑或往下的條件判斷

<共有四種可能性，決定下一個cycle該往上or下>

Input:

sel\_con: 結合其他三個input，做出四種state的條件選擇

K1: cur\_Floor之上是不是還有人呼叫 or 未達成目的地

K2: cur\_Floor之下是不是還有人呼叫 or 未達成目的地

K3: cur\_Floor為目的地之一 or 該層外有向上的要求

K4: cur\_Floor為目的地之一 or 該層外有向下的要求

get\_call, get\_goal, cur\_Floor: 來自其他module的訊號

Output:

result: 將sel\_con計算出的結果送給controller

Module 8 controller: 電梯總流程控制

<接受、控制電梯的動作與流程進行>

input:

clk: clock。

reset: 歸零

consignal: 來自counter的訊號

result\_Con:來自conditon的訊號

output:

rst,sel\_Con,inc\_Floor,red\_Floor,ce,up\_d,stop\_r, ce\_Coun朝其他module發出指令

裡面會定義一個state [3:0]，有s0~s14共15種state，藉由讀取來自condition judge的K1~K4，來變更電梯所在樓層與運行方向，分別為：

S0: 初始化，位於1F，直接進入S1等待。

S1: 等待， dir = 00，判斷K1，符合進入S3電梯向上，不符合進入S2。

S2: 判斷K2，符合進入S9電梯向下，不符合回到S1繼續等待指令。

S3: dir = 10 (往上)，直接進入S4進行counter倒數。

S4: 切換樓層的counter開始倒數，若倒數完進入S5，否則待在S4。

S5: cur\_Floor + 1後，直接進入S6。

S6: 判斷K3，符合進入S8停在該樓層(dir = hold)，不符合進入S7。

S7: 判斷K1，符合進入S4繼續執行，不符合同樣進入S8停止。

S8: door\_open = 1(開門)，進行開關門的counter倒數，倒數完回到S1待命。

S9: dir = 01(往下)，直接進入S10倒數。

S10: 切換樓層的counter開始倒數，若倒數完進入S11，否則待在S10。

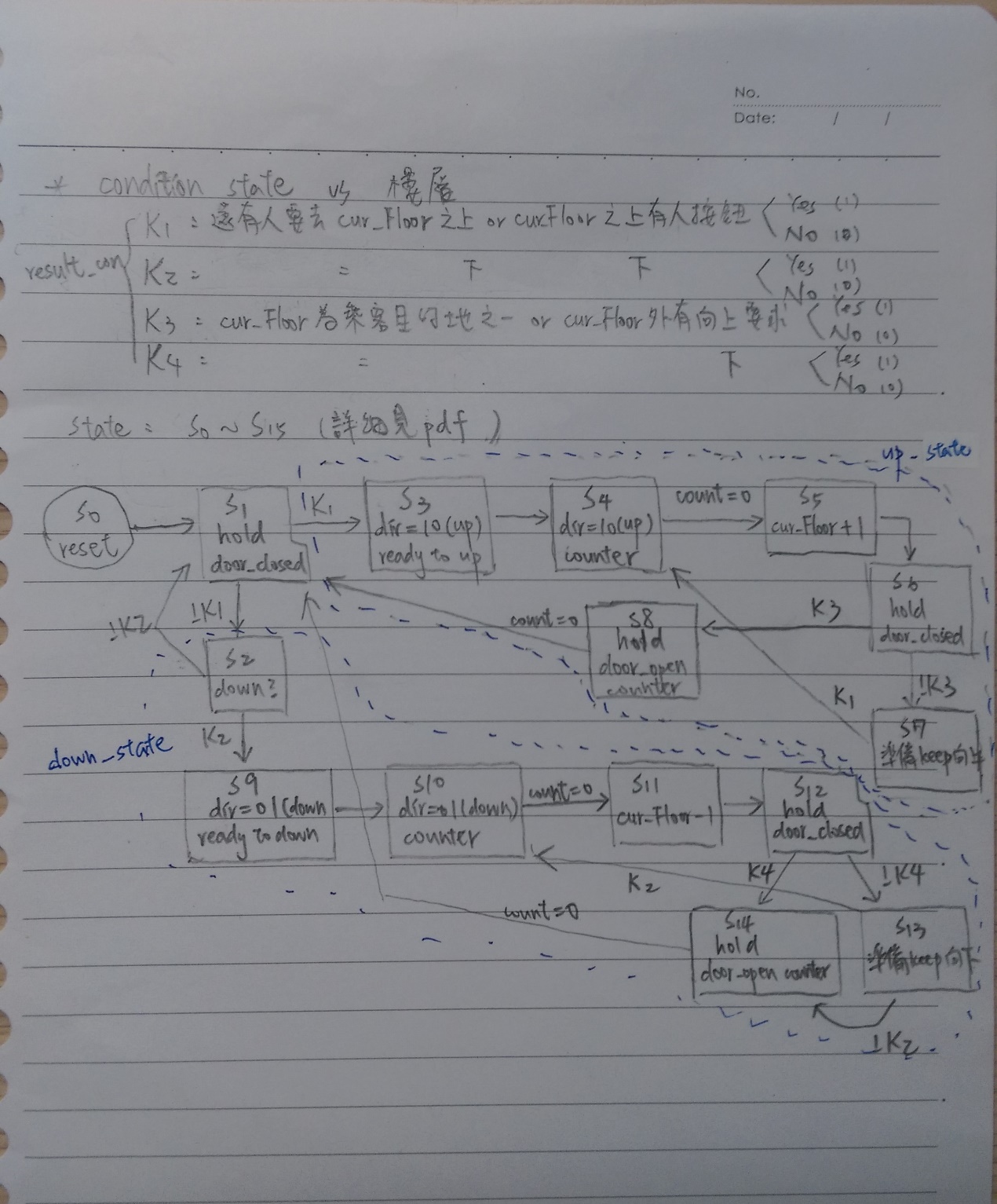
S11: cur\_Floor – 1後，直接進入S12。

S12: 判斷K4，符合進入S14停止，不符合進入S13。

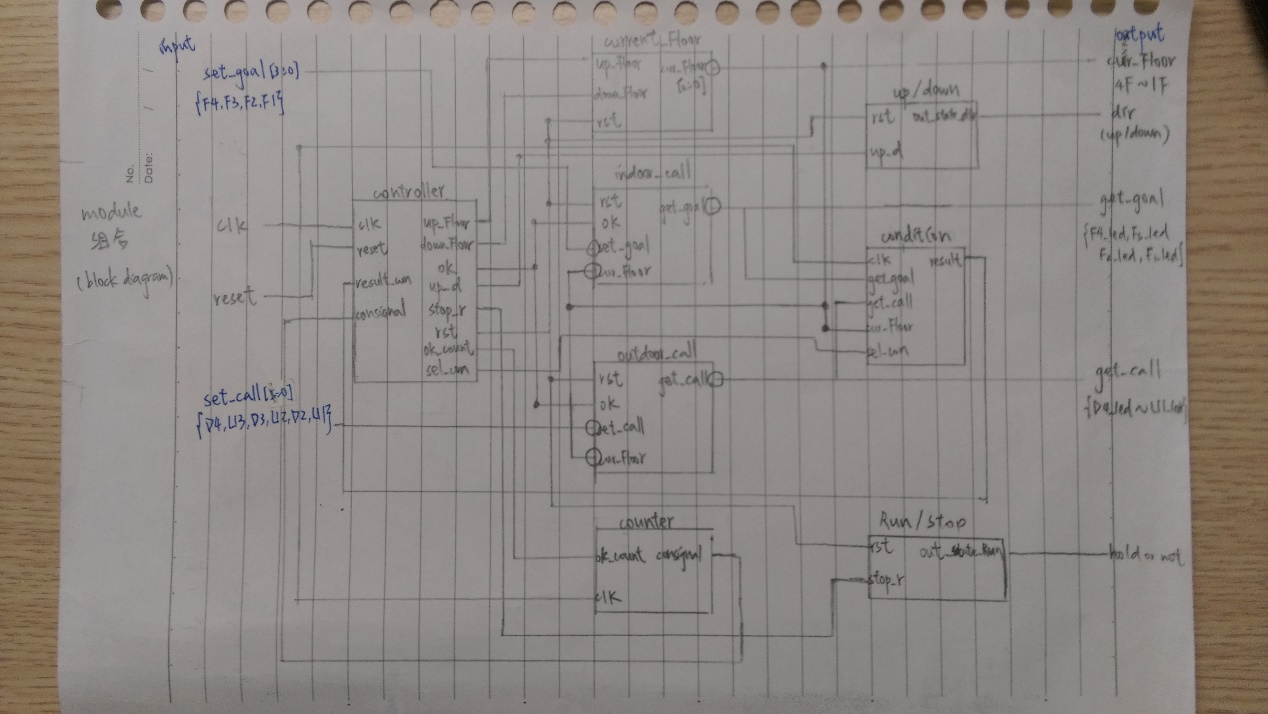
S13: 判斷K2，符合進入S10繼續執行，不符合同樣進入S14停止。

S14: door\_open = 1(開門)，進行開關門的counter倒數，倒數完回到S1待命。

整個module組合的block diagram:



K1~K4 & S0~S14的state diagram:



問題：

1. 無法順利的將所有模組順利整合
2. Testbench設計不完整。因此無法順利執行。