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，有人按了外面的{D4,U3,D3,U2,D2,U1}任一按鈕後，電梯會移動該樓層，打開門，再讓使用者按下{F4, F3, F2, F1}決定目的地。

按下按鈕時，對應的LED燈會亮，即{U1\_led, U2\_led, D2\_led, U3\_led, D3\_led, D4\_led, F1\_led, F2\_led, F3\_led, F4\_led}對應的為1；到達該樓層後，與該樓層有關的LED就會暗掉，即變為0。

Block diagram的設計與結構(配合之後的圖)：

Primary Input:

F4, F3, F2, F1: 電梯內按鈕(目的地選擇)

D4,U3,D3,U2,D2,U1: 電梯外按鈕(要求呼叫)

Primary Output:

F1\_led, F2\_led, F3\_led, F4\_led: 電梯內按鈕的LED亮暗

U1\_led, U2\_led, D2\_led, U3\_led, D3\_led, D4\_led: 電梯外按鈕的LED亮暗

Floor: 目前所在樓層

door\_open(opened): 1: 開門 ; 0: 關門

Dir: 運行方向 or HOLD

State:

S\_F1, S\_F2, S\_F3, S\_F4: 代表1~4樓

MOVE: 電梯移動中

OPEN: 電梯開門中

State diagram的切換條件(配合之後的圖)：

State\_F1: 只能HOLD or 向上

If (U1 or F1==1) 即第一層為目的地or有向上請求 →next state = OPEN

Else if (U2\_led or D2\_led or F2\_led or U3\_led or D3\_led or F3\_led or D4\_led or F4\_led)

即有往樓上為目的地的請求or上方樓層有請求的話

→next state=MOVE

→Dir = up

State\_F4同State\_F1，指示改為下方有要求時，Dir = down

State\_F2: 可向上也可向下

If (U2 or F2==1) →next state = OPEN

Else if (U1\_led or F1\_led or U3\_led or D3\_led or F3\_led orD4\_led or F4\_led)

必須依目前正往上or往下，來決定先執行哪項呼叫。

If (Dir = up) →先執行上方樓層要求，再執行下方樓層

Else if (Dir = down) →剛好相反，先執行下方再執行上方。

State\_F3與State\_F2雷同。

電梯運行規則：

單一呼叫時的State切換：

假使目前位於第i層，而第i層有任何按鈕要求(無論電梯內或電梯外)時：

→將第i層的相關按鈕全部歸零

→Opened=1，Dir = HOLD

執行完成後，關門，在該地點等待下一次呼叫。

多重呼叫時的State切換：

Ex: 電梯一開始在1F往上到第3層，電梯內同時有人按F2&F4。

因為Dir=up，而第四層比第二層高，因此應優先處理完第四層後再去F2。

→Dir=up，抵達4F，Opened=1，Dir=HOLD，將4F有關的任何按鈕歸零

→將Dir=down，去處理F2的目的地請求

執行完成後，關門，在該地點等待下一次呼叫。

程式碼呈現：

請見elevator.v內，所加註的註解。

模擬時的執行指令：

ncverilog test.v elevator.v +access+r (testbench1)

ncverilog test2.v elevator.v +access+r (testbench2)

Testbench:

1. 三樓有人按向下請求，電梯上升到三樓後打開。乘客按下F2，電梯往下到F2後停止。
2. 二樓有人按向上請求，進去後同時按了F3F4，會先執行F3後再執行F4。

無法解決的問題：

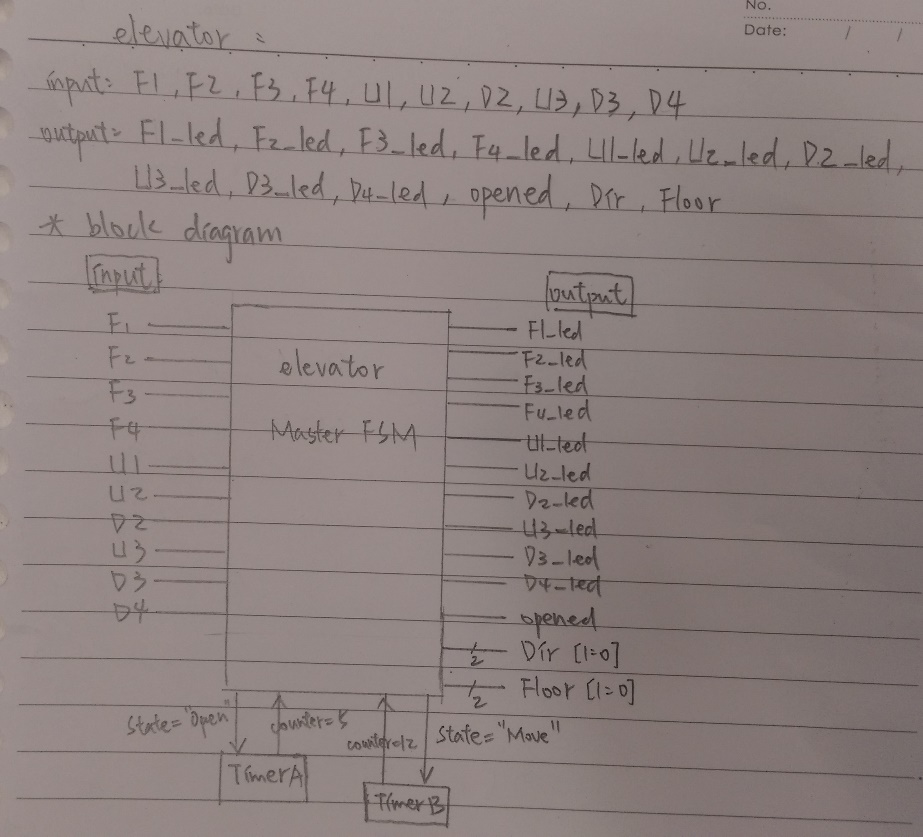
1. 對testbench的操作不慎熟悉，導致沒辦法完美的達成想要的input/output。
2. 對now state / next state/ previous state配合clk取值的調控不好，有時會讓next state讀取到錯誤的值，造成判斷錯誤。
3. Timer因為第二點的原因無法執行，開關門與切換樓層變成只有1 cycle。

計畫中的解決方法：

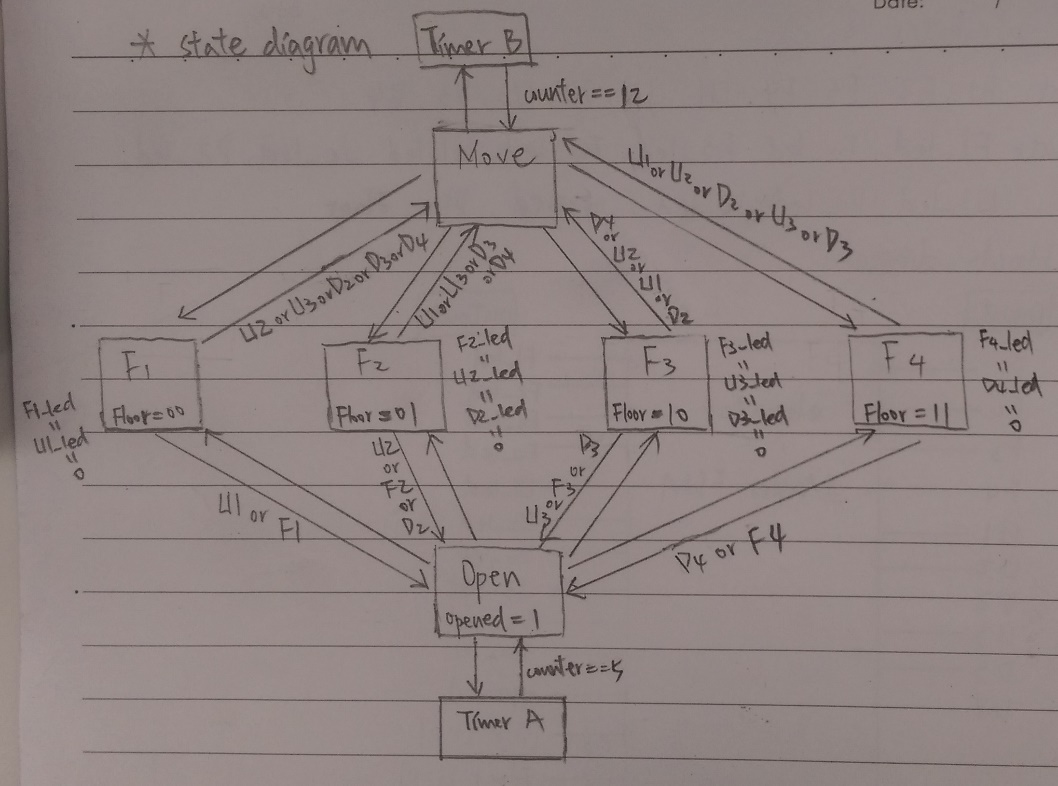
1. 藉由反覆測試與翻書，來抓到想要的測資內容與結果。
2. 本身state machine時一開始的設計錯誤，為避免吃到錯誤/不存在的值，應該避免同時取值 (previous state=state ,next state=state)的方法，和設置default來讀取出錯而不存在的state。
3. 若第二點能能成功改良，相信第三點便迎刃而解。

整體電梯設計：

Block diagram:

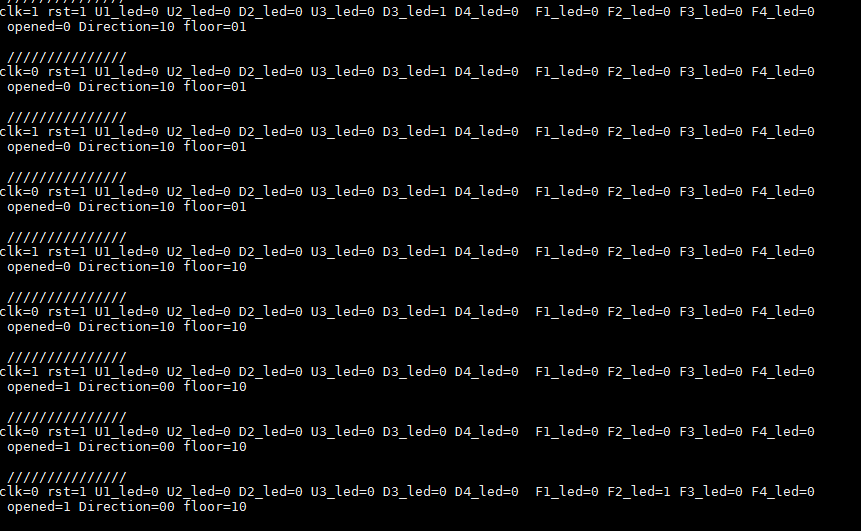


State diagram:

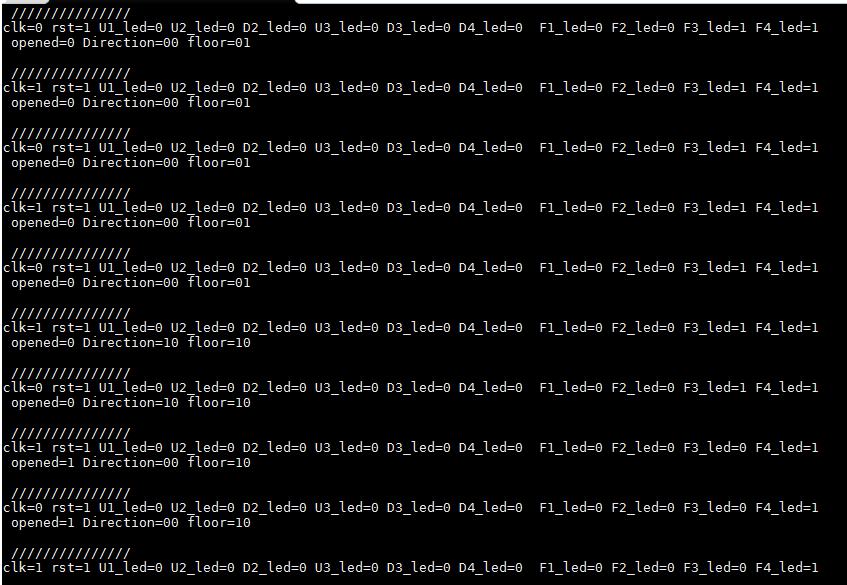


Simulation:

Testbench1:

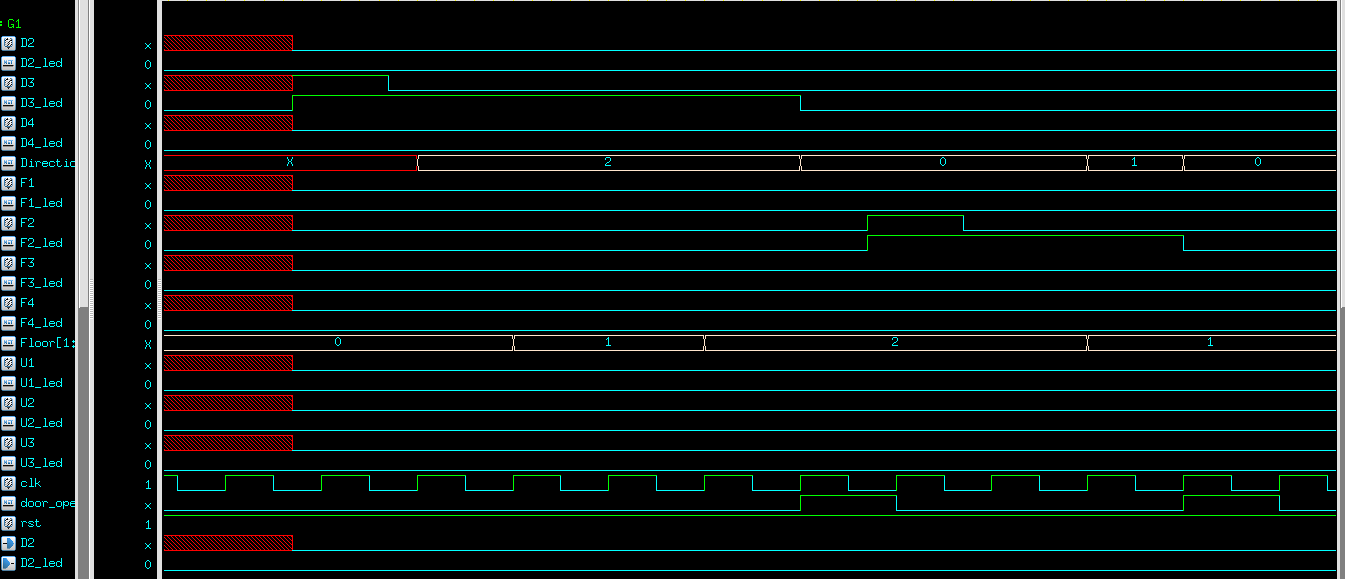


Testbench2:



波形圖:

Testbench1:



Testbench2:

