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。

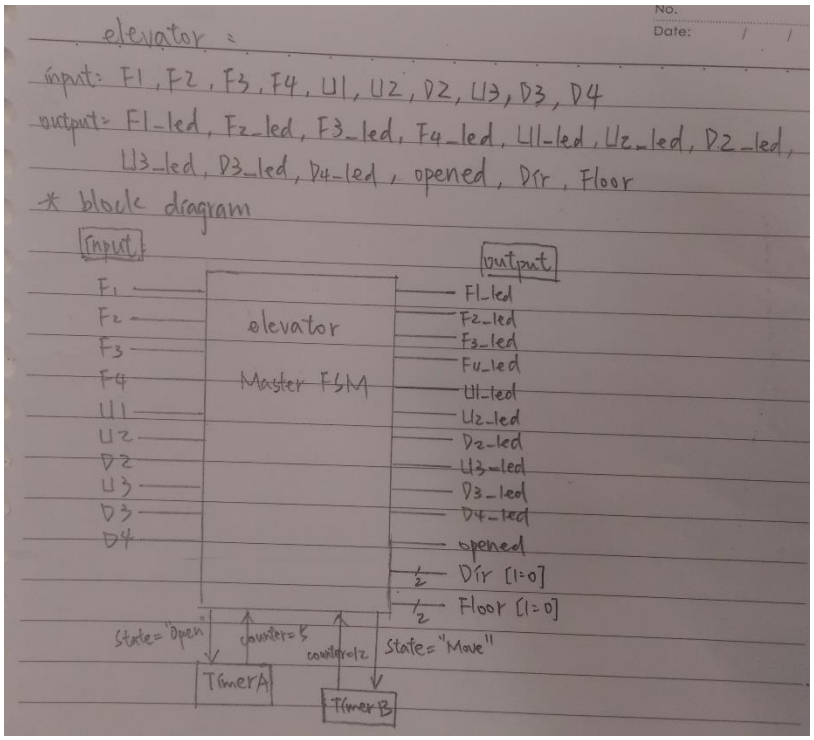
計畫中的解決方法：
1. 藉由反覆測試與翻書，來抓到想要的測資內容與結果。
2. 本身 state machine 時一開始的設計錯誤，為避免吃到錯誤/不存在的值，應該避免同時取值 (previous state=state ,next state=state)的方法，和設置 default 來讀

取出錯而不存在的 state。

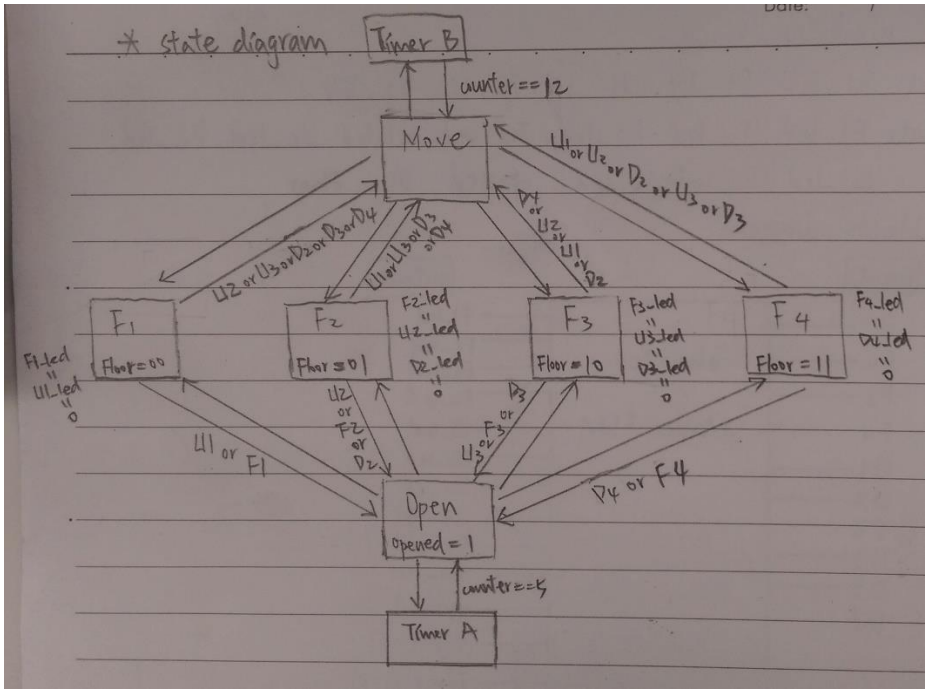
3. 若第二點能能成功改良，相信第三點便迎刃而解。

整體電梯設計：

Block diagram:



State diagram:



Simulation:

Testbench1:

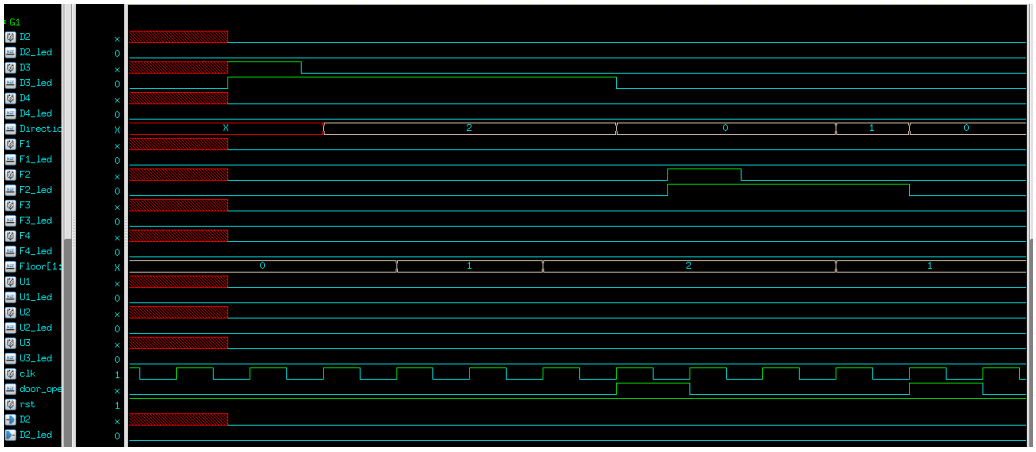
```
clk=1 rst=1 U1_led=0 U2_led=0 D2_led=0 U3_led=0 D3_led=1 D4_led=0  F1_led=0 F2_led=0 F3_led=0 F4_led=0
 opened=0 Direction=10 floor=01

 ///////////////
clk=0 rst=1 U1_led=0 U2_led=0 D2_led=0 U3_led=0 D3_led=1 D4_led=0  F1_led=0 F2_led=0 F3_led=0 F4_led=0
 opened=0 Direction=10 floor=01

 ///////////////
clk=1 rst=1 U1_led=0 U2_led=0 D2_led=0 U3_led=0 D3_led=1 D4_led=0  F1_led=0 F2_led=0 F3_led=0 F4_led=0
 opened=0 Direction=10 floor=01

 ///////////////
clk=0 rst=1 U1_led=0 U2_led=0 D2_led=0 U3_led=0 D3_led=1 D4_led=0  F1_led=0 F2_led=0 F3_led=0 F4_led=0
 opened=0 Direction=10 floor=01

 ///////////////
clk=1 rst=1 U1_led=0 U2_led=0 D2_led=0 U3_led=0 D3_led=1 D4_led=0  F1_led=0 F2_led=0 F3_led=0 F4_led=0
 opened=0 Direction=10 floor=10

 ///////////////
clk=0 rst=1 U1_led=0 U2_led=0 D2_led=0 U3_led=0 D3_led=1 D4_led=0  F1_led=0 F2_led=0 F3_led=0 F4_led=0
 opened=0 Direction=10 floor=10

 ///////////////
clk=1 rst=1 U1_led=0 U2_led=0 D2_led=0 U3_led=0 D3_led=0 D4_led=0  F1_led=0 F2_led=0 F3_led=0 F4_led=0
 opened=1 Direction=00 floor=10

 ///////////////
clk=0 rst=1 U1_led=0 U2_led=0 D2_led=0 U3_led=0 D3_led=0 D4_led=0  F1_led=0 F2_led=0 F3_led=0 F4_led=0
 opened=1 Direction=00 floor=10

 ///////////////
clk=0 rst=1 U1_led=0 U2_led=0 D2_led=0 U3_led=0 D3_led=0 D4_led=0  F1_led=0 F2_led=1 F3_led=0 F4_led=0
 opened=1 Direction=00 floor=10
```

Testbench2:

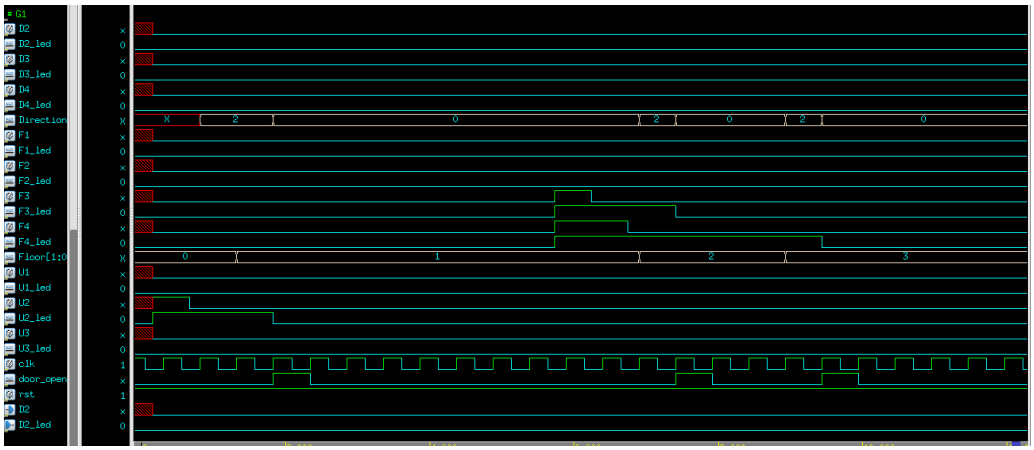
```
 ///////////////
clk=0 rst=1 U1_led=0 U2_led=0 D2_led=0 U3_led=0 D3_led=0 D4_led=0  F1_led=0 F2_led=0 F3_led=1 F4_led=1
 opened=0 Direction=00 floor=01

 ///////////////
clk=1 rst=1 U1_led=0 U2_led=0 D2_led=0 U3_led=0 D3_led=0 D4_led=0  F1_led=0 F2_led=0 F3_led=1 F4_led=1
 opened=0 Direction=00 floor=01

 ///////////////
clk=0 rst=1 U1_led=0 U2_led=0 D2_led=0 U3_led=0 D3_led=0 D4_led=0  F1_led=0 F2_led=0 F3_led=1 F4_led=1
 opened=0 Direction=00 floor=01

 ///////////////
clk=1 rst=1 U1_led=0 U2_led=0 D2_led=0 U3_led=0 D3_led=0 D4_led=0  F1_led=0 F2_led=0 F3_led=1 F4_led=1
 opened=0 Direction=00 floor=01

 ///////////////
clk=0 rst=1 U1_led=0 U2_led=0 D2_led=0 U3_led=0 D3_led=0 D4_led=0  F1_led=0 F2_led=0 F3_led=1 F4_led=1
 opened=0 Direction=00 floor=01

 ///////////////
clk=1 rst=1 U1_led=0 U2_led=0 D2_led=0 U3_led=0 D3_led=0 D4_led=0  F1_led=0 F2_led=0 F3_led=1 F4_led=1
 opened=0 Direction=10 floor=10

 ///////////////
clk=0 rst=1 U1_led=0 U2_led=0 D2_led=0 U3_led=0 D3_led=0 D4_led=0  F1_led=0 F2_led=0 F3_led=1 F4_led=1
 opened=0 Direction=10 floor=10

 ///////////////
clk=1 rst=1 U1_led=0 U2_led=0 D2_led=0 U3_led=0 D3_led=0 D4_led=0  F1_led=0 F2_led=0 F3_led=0 F4_led=1
 opened=1 Direction=00 floor=10

 ///////////////
clk=0 rst=1 U1_led=0 U2_led=0 D2_led=0 U3_led=0 D3_led=0 D4_led=0  F1_led=0 F2_led=0 F3_led=0 F4_led=1
 opened=1 Direction=00 floor=10

 ///////////////
clk=1 rst=1 U1_led=0 U2_led=0 D2_led=0 U3_led=0 D3_led=0 D4_led=0  F1_led=0 F2_led=0 F3_led=0 F4_led=1
```

波形圖:

Testbench1:


G1
D2
D2_led
D3
D3_led
D4
D4_led
Directio
F1
F1_led
F2
F2_led
F3
F3_led
F4
F4_led
Floor[1:
U1
U1_led
U2
U2_led
U3
U3_led
clk
door_ope
rst
D2
D2_led

Testbench2:


G1
D2
D2_led
D3
D3_led
D4
D4_led
Direction
F1
F1_led
F2
F2_led
F3
F3_led
F4
F4_led
Floor[1:0
U1
U1_led
U2
U2_led
U3
U3_led
clk
door_open
rst
D2
D2_led