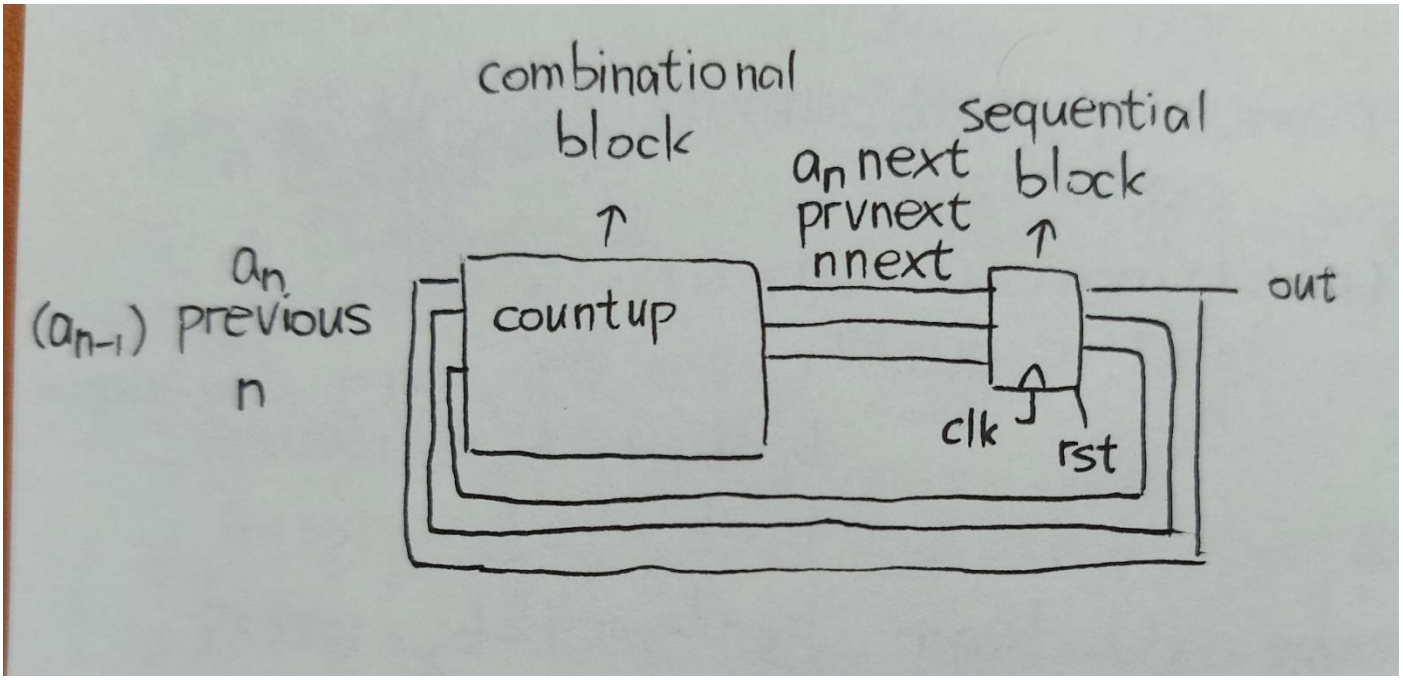
Lab 2

學號:109062318 姓名: 簡弘哲

1. 實作過程

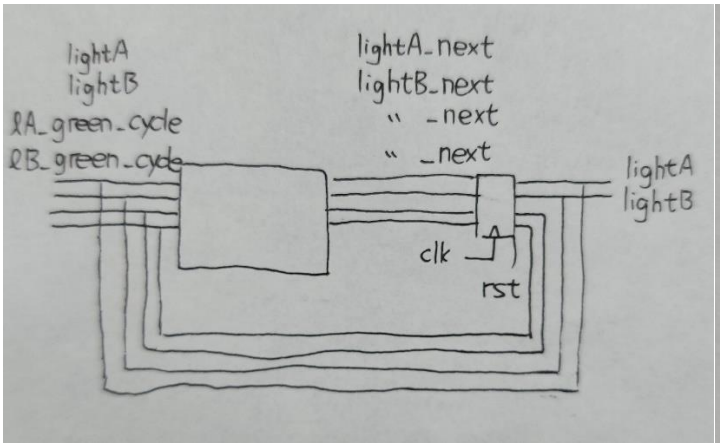
Lab2-1:

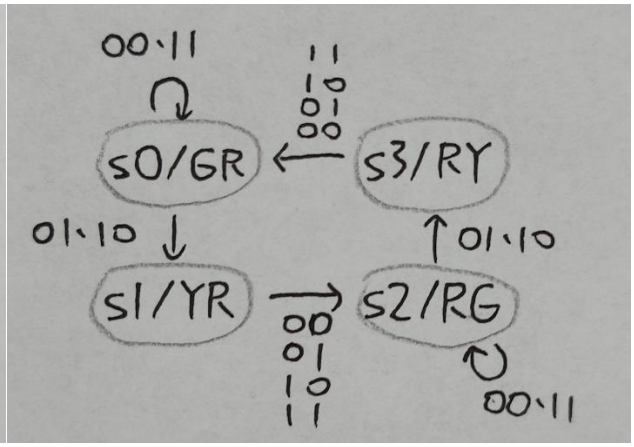


(此圖最左邊那 3 條接上 combinational block 的線應向左延伸以代表 input)

如圖，flip-flop 的部分只專注於 memorizing state，至於 combinational block 負責計算下一個數字，會需要用到前一個數字 a_{n-1} 以及 n 去推出 a_n，以及有一個 countup 變數記錄目前是往上數還是往下數，我把 n+=1 的動作放在 combinational block 中處理，也就是 n_next=n+1，以及判斷 edge case 的部分(數到 0 或 63 時的 n 跟 countup 該怎麼變化)。

Lab2-2:





Lab2-2 也是分為 combinational block 跟 sequential block，前者負責計算下一個紅綠燈的情況，需要知道當前紅綠燈的狀態以及綠燈所維持的 cycle 數；後者專注於 memorizing state。

至於右圖的 state diagram 我將四個 state 取名為 s0-s3(原本要取顏色名，但為了方便辨識才改用 s0-s3)，在 code 中我的 state 就直接用顏色名，沒有事先定義 s0-s3 的 parameter。

2. 學到的東西與遇到的困難

Lab2-1:

一開始可以寫出 countdown 的樣子，雖然它遇到了 overflow 的問題而且產生出的數列也不是題目所要(我一開始是 0->63->61->57->49->33->1)，但這是一個好的開始。

後來不小心把 code 改爛，導致數列直接變公差為-1 的等差數列，不過這沒有持續多久我就把 code 改成原來的了。

之後卡了一陣子，一直不知為何數列不是我所期待的，直到我注意到了在 countup 中某個 if 的寫法可能有 bug，就是 $a_{n-1} – n > 0$(原本的寫法)，而這似乎在 boundary case 的時候($a_{n-1} = 0, n = 1$)會導致 $a_{n-1} – n = 63 > 0$。後來改成 $a_{n-1} > n$ 就解決問題了。

Countup 的雛型基本上完成，但要數到 63 的時候，63 反而不會出現就直接開始往下減了，後來發現這是因為一開始我在一個單獨的 always block 裡檢查 edge number 1,63 的時候，我直接把 countup 反轉，導致它不顯示 63 就直接開始往下減。之後思考了好一陣子到底要怎麼把 63 維持一個 clock cycle，所以我去觀察了一下 wave form 以後，發現我應該可以用 n 的值來判斷它是否達到 edge，果不其然把原本的 if(an==0 || an==63)改成 if(an==0 || n==58)以後一切就完美了(因為從 6->63 的時候 n 就從 57 變 58，這時已達到 edge case) ，但是之後遇到在 rst 的時候我的 0 會出現兩個 cycle，一開始我以為是要對 flipflop 的值做修改，後來仔細想想我可以在判斷 edge case 的時候再判斷 rst=1 or 0，如果 rst=1 那 n 就從 1 開始，反之從 0 開始(也就是一般的數上數下)。。

Lab2-2:

第一次測試的時候遇到 error2，發現我沒有考慮(carA,carB)=(1, 1) or (0, 0)的情況，再加一個 else 讓燈繼續維持就解決了。

一開始我沒有用 flip-flop 去記住綠燈維持了幾個 cycle(code 中的 lightX_green_cycle)，且為了避免在 combinational block 中出現 a=a+1 之類的 statement，所以我新增了 2 個變數(lightX_green_cylce_next, X=A, B)並把它加入 flip-flop，error4 就解決了。

最後一個 error5 讓我頭痛許久，因為我錯的地方剛好跟 testbench 的差一個 clock cycle，不管我把 code 做怎樣的小修改(把某一行移到另一個看似可行的地方)都不見起色，我也回去檢查了一下我的 state diagram，看起來是沒錯的，所以讓我開始在想是不是因為 mealy 跟 moore 的 timing difference 所造成的，但當我試著把 moore 轉成 mealy 的時候感覺又有點奇怪，明明 moore 就差一點點就成功了，應該不是 mealy 的問題。

隔天看 waveform 的時候，心中突然有一些疑問，那就是我有沒有記得在(carA,carB)=(1, 1) or (0, 0)的情況下也要處理綠燈的維持 clycle，果然發現該地方我沒有 maintain light_green_cycle，把該加的 code 加上去以後就一切正常 pass test 了，之後的 testbench 有做 revision 導致原本的 code 會錯，不過將原本 code 的 green light cycle 也做個小修改就 ok 了。

3. 想對老師或助教說的話

我覺得我從 verilog series 07,08 中收穫非常多，影片講解清晰易懂，在看完 FSM modeling 的 coding guidelines 後，我突然知道大一下學期的邏輯設計課中的某一個 verilog FSM 作業為什麼要那樣設計了，不然在當時不知道為何那樣設計導致我不懂 code 的邏輯，讓我掙扎了許久。但經過了這次的練習以後，對 verilog 越來越熟悉，也漸漸變得不再排斥 verilog

希望下次出完 spec 之後盡量不要小修改題目，這樣先寫完的人要再回去修改+測試會有一點小麻煩。