|  |  |
| --- | --- |
| **Lab 2** | |
| 學號:109062318 | 姓名: 簡弘哲 |

1. 實作過程

Lab2-1:

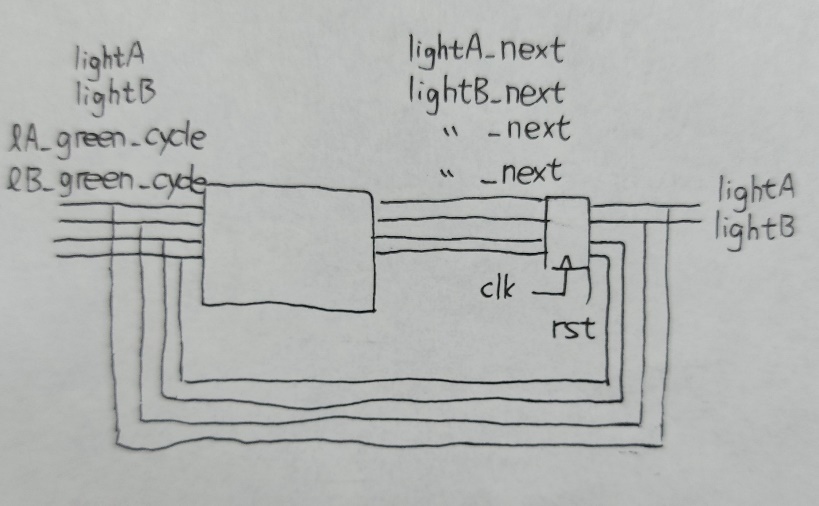
一張含有 文字, 白板 的圖片

自動產生的描述

(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an-1以及n去推出an，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:

一張含有 文字, 白板 的圖片

自動產生的描述

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。

1. 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an-1 – n > 0(原本的寫法)，而這似乎在boundary case的時候(an-1 = 0, n = 1)會導致an-1 – n = 63 > 0。後來改成an-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。

1. 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。