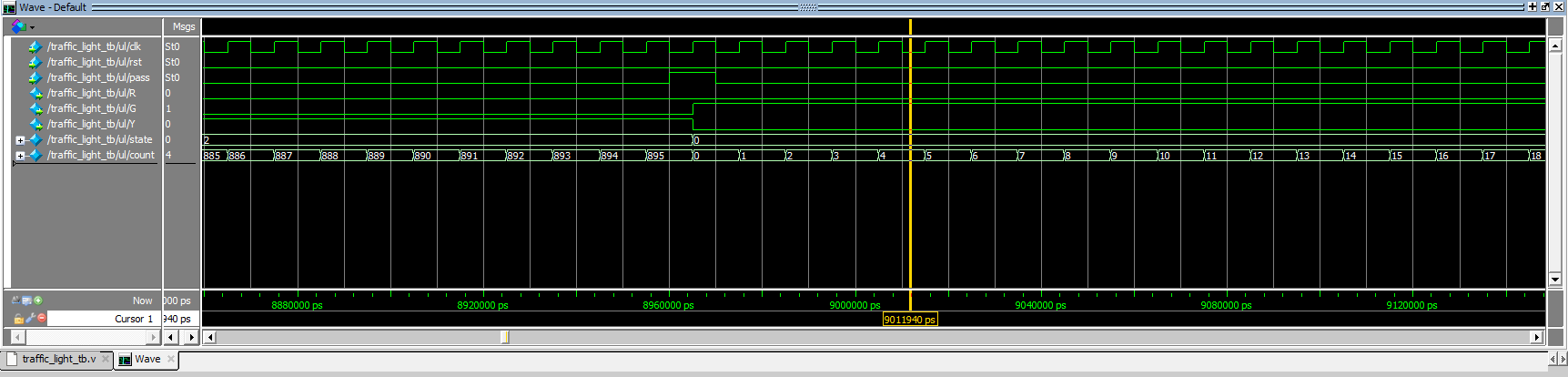
**Computer Organization 2022**

**HOMEWORK 2**

系級: 資訊113 學號: F94081076 姓名: 郭立晨

**實驗結果圖:**

(波形圖及模擬完成截圖)



**一張含有 文字 的圖片

自動產生的描述**

**程式運作流程:**

(簡單說明波形變化的意義)

第一個always中使用非同步正緣reset, 當reset為1時，將state與count(紀錄cycle數)設為0以及pass\_state(紀錄是否有重設到初始綠燈狀態)設為0。在reset為0時開始運作，使用nonblocking更新cycle的數量並使用case來判斷當前的cycle的數量，並決定要進入哪一個狀態。

當狀態改變時，會觸發第二個always，並根據當前的狀態利用case決定RGY的值各是多少，如果剛好在最後一個狀態要回到初始綠燈狀態時重設count值。若是pass升起，則在第一個always當中會判斷count數是否仍在初始綠燈狀態範圍中，如果在初始綠燈狀態的範圍則不做任何改變，反之則會將state以及count初始化為最一開始的綠燈以及0 cycles，也將pass\_state設為1，pass\_state為1代表有執行重設的動作，必須要將當前初始化的狀態保持住，直到pass結束。當pass一結束，其負緣訊號會觸發第二個always，將pass\_state設為0，使下次正緣進來時第一個always能繼續計數cycle數量，並正常運作下去。。

**心得**

(請寫下完成本次作業的心得、學到哪些東西、困難點的部分。)

本次作業主要學習到如何在時脈變化下根據前一個狀態決定出相對應的下一個狀態。最困難的部份我覺得是要應付隨時有可能升起的pass訊號，必須要根據當前狀態來決定要有甚麼動作。在這些操作之中，哪一些要使用nonblocking，哪一些要用blocking的寫法讓我蠻困擾的，現在仍然很難第一時間就馬上有想法，有時候一不注意有可能就會導致不可預期的錯誤發生。Pass狀態下如果有重設回到初始綠燈狀態，必須將重設的狀態保持住，這個部份我一開始沒有發現，導致我的結果都會提早幾個cycle改變，最後才發現是上述的問題，因此設了一個pass\_state來記住現在是否是需要保持初始狀態的狀態。加了pass\_state之後就全部過關了!

整體來說我覺得這次作業比第一次的簡單，也花相對少的時間解題，面對錯誤也能越來越快能找到問題出在哪，就能把時間花在解決問題，而不是一直尋找問題，那會很浪費時間。我覺得這樣的練習剛剛好，既能熟悉verilog的基本語法也能熟悉各種題型，並能更流暢的轉換寫硬體描述語言與一般軟體的思維。期望之後的練習可以讓目前所學都應用上，並加深加廣，建立起對於硬體描述語言更深一層的技能。