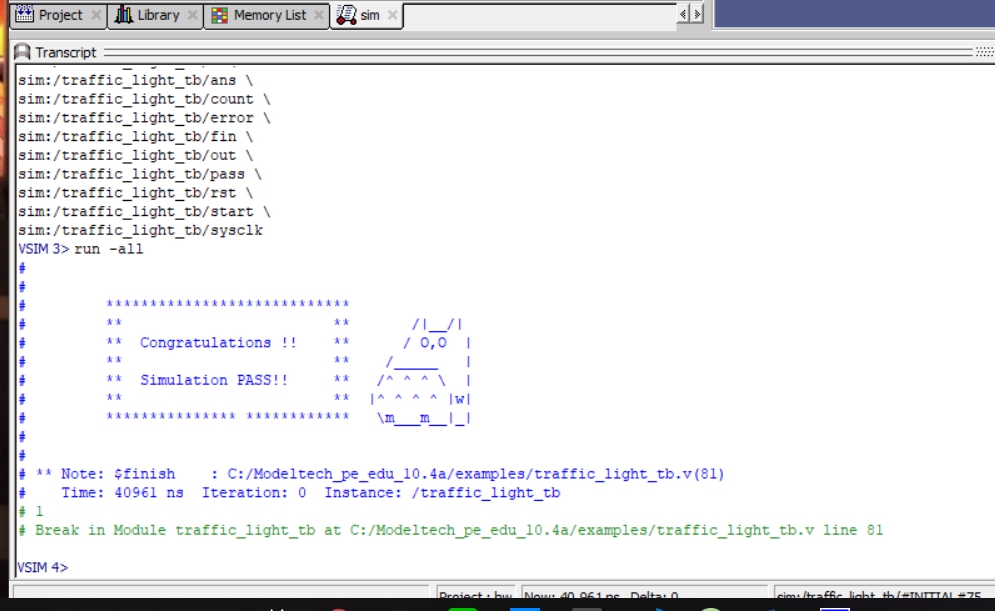
**Computer Organization 2019**

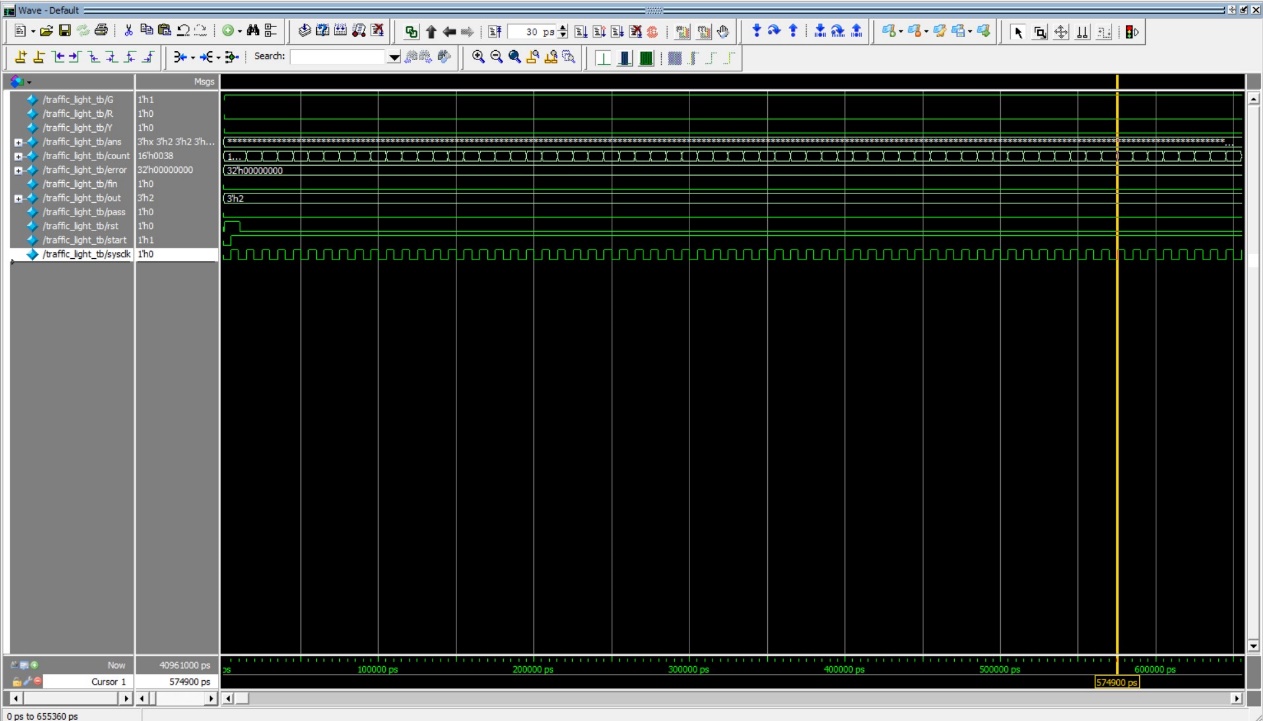
**HOMEWORK 1**

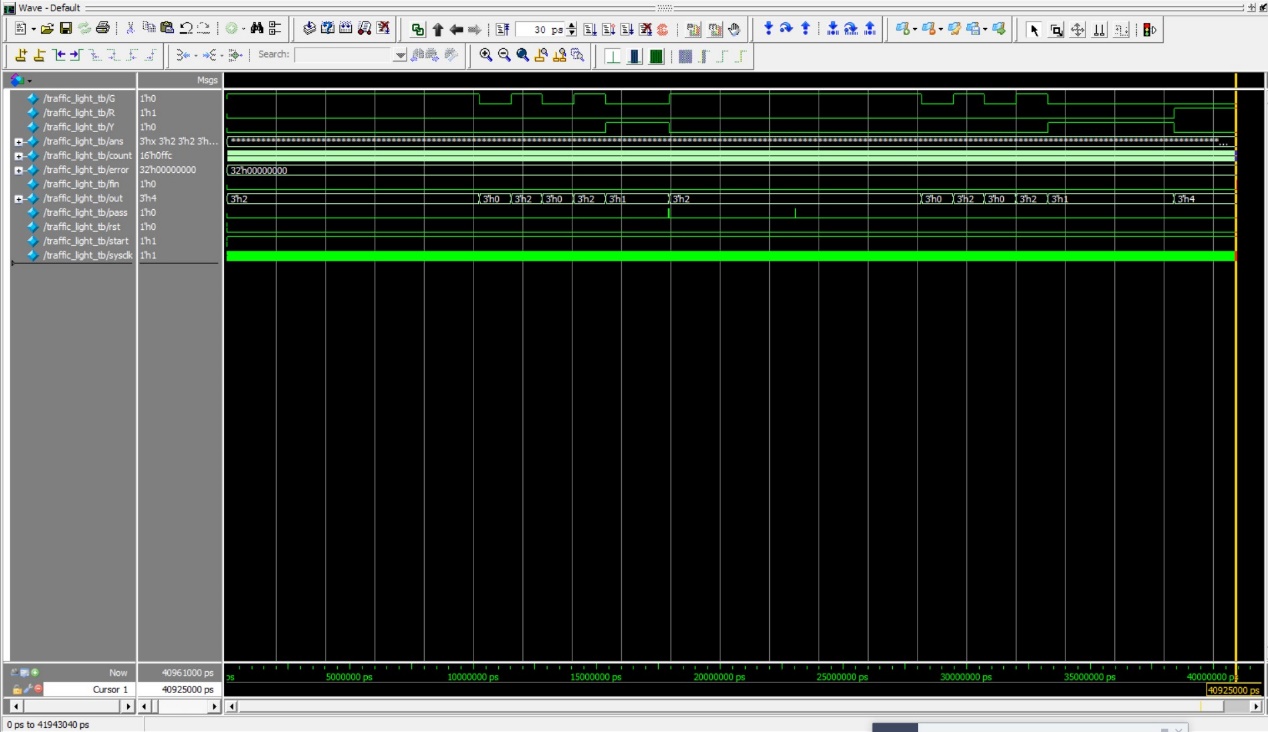
系級: 工資110 學號:H34064090 姓名: 游哲瑋

**實驗結果圖:**

(波形圖及模擬完成截圖)

****

****

**程式運作流程:**

(簡單說明波形變化的意義)

最上面三條波形G R Y輪流的變化代表了燈號的變化

每次波形的觸發都代表燈號的更換

而每次pass一觸發 就又會從G開始重新循環

至於out則是理所當然地跟著R G Y一起變化(不如說R G Y就是out)

clk則是從頭持續到尾的循環計時 因為正緣觸發的關係只在正邊緣時才執行

而每次執行都是在判斷要跑到哪一個狀態 因此使用大量的if else來同時判斷當前的狀態是甚麼而只判斷狀態是不夠的 因為有多個跑128秒的綠燈也有多個跑128秒的無燈號 因此還需要判斷此時是在跑哪一個時段的燈故還需要一個計時器 count來判斷

另外還有reset需要令整個狀態回到初始狀態

**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。)

本次作業看似簡單但其實暗藏了許多危險，

首先是4個狀態 我一開始就想到要使用4個大小的reg去做，

但後來發現若使用=去賦予狀態，會出錯。

後來查了一下counter的使用方法，發現<= 和 = 之間的差別才改為使用<=

接下來是初始狀態的部分，原先我認為count初始狀態是從1開始數，但後來跑了幾千個錯誤出來…煩惱了幾天後發現該不會是從0開始數吧…，一改就只剩下幾個錯誤了…簡直傻了。

另外就是在debug過程中曾經想說if else有點太多，不如改為switch寫過 但發現不能用 原因應該在於條件是多重的(兩個)所以不太能使用。

在本次作業其實複習了蠻多verilog基礎概念，身為外系的我也因此必須將基礎補起來才能寫出這份作業，但好險只用到基礎，不燃應該是要交上一堆錯誤的程式屍體了。