Computer-Aided Design for VLSI Design

Homework3（Student ID: 61047046S｜Name: 林宣佑）

1. Provide a simple explanation of your code.

這次的作業定義了一個名為 HW3 的 entity 作為主程式，另外定義了 Counter 以及 Controller 兩個 entity 作為 component，並引入它們的功能來實現紅綠燈的控制。

HW3：

在 entity HW3 中定義了它的實體。它具有一個 input：clk（時脈），三個 output：glight（綠燈）、ylight（黃燈）和 rlight（紅燈）。clk 用於驅動整個系統的時脈，而 glight、ylight 和 rlight 則用於輸出控制紅綠燈的狀態。

在 architecture Behavioral of HW3 中，先是定義了 Counter 及 Controller 互相溝通用的三個 signal：reset、enable、complete，以及用來定義計數器上限的 count。接下來引入了 Counter 和 Controller 兩個 component。

接下來定義 Counter_inst 和 Controller_inst 來實例化 Counter 和 Controller 兩個 component，並使用 port map 將兩個 component 與主程式相互連接，由此將它們的功能集成到主程式中，並且可以透過 clk、reset、enable 和 complete 訊號進行相對應的操作和控制。

Counter 負責進行計數操作。它接收 clk 訊號作為時脈輸入，reset 訊號

用於重置計數器，enable 訊號用於判斷計數器是否啟用，計數器的計數值可由主程式設定上限。

Controller 則負責控制紅綠燈的狀態，它接收 clk 訊號作為時脈輸入，reset 和 enable 訊號由 Controller 生成並輸出給 Counter，complete 訊號則由 Counter 輸出並作為輸入給 Controller，另外，glight、ylight、rlight 訊號是用於輸出控制紅綠燈的狀態。

主程式中的 reset、enable 和 complete 訊號在 Counter 和 Controller 之間作用為傳遞和控制。當 HW3 收到 clk 訊號時，它會同時將 clk 訊號傳遞給 Counter 和 Controller，並從 Controller 接收到 glight、ylight 和 rlight 訊號。根據 Counter 和 Controller 的操作和狀態轉換，HW3 會相應地設定紅綠燈的狀態並更新下一個狀態。

總結來說，HW3 作為紅綠燈控制器的主程式，它將 Counter 和 Controller 作為組件引入並使用它們的功能來實現紅綠燈的控制。Counter 負責計數操作，Controller 負責控制紅綠燈的狀態。通過主程式中的訊號傳遞和控制，紅綠燈的狀態會根據 Counter 和 Controller 的操作進行切換和更新，整個專案也是一個有限狀態機。

Counter：

Counter 在這個紅綠燈控制系統中負責計數操作，在 entity Counter 中定義了它的實體，它具有三個 input：clk（時脈）、reset（重置）、enable（啟用），兩個 output：complete（完成）和 count（計數）。

- clk 是時脈訊號，用於驅動計數器的計數操作。
- reset 是重置訊號，當 reset 為高電位（'1'）時，計數器將被重置為初始值。
- enable 是啟用訊號，當 enable 為高電位（'1'）時，計數器開始計數。
- complete 是完成訊號，當計數器的計數值達到最大值時，complete 將被設置為高電位（'1'），表示計數操作已完成。
- count 是計數訊號，它是一個 buffer integer 型態，用於輸出計數器的計數值。

在 architecture Behavioral of Counter 中，我們定義了一個 process，用於控制計數器的操作。在這個過程中，我們偵測 clk 訊號的變化。

- 當 clk 訊號的 rising edge 觸發時，首先檢查 reset 訊號是否為高電位（'1'），若是則我們將計數器的輸出值重置為全零，即 count <= 0、complete <=（'0'），若否則接著
- 檢查 enable 訊號是否為高電位（'1'），若是則我們將計數器的值加 1，即 count <= count + 1，並檢查計數器的值是否已經達到最大值，即 count = MAX_COUNT。如果是，則將 complete 訊號設置為高電位（'1'），表示計數操作已經完成。
- 如果計數器的值尚未達到最大值，則將 complete 訊號設置為低電位（'0'）。
- 最後，我們將計數器的值轉換為 buffer integer 型態，並將其賦值給 count 訊號，以便在 Counter 中輸出計數值。

總結來說，Counter 組件通過使用 clk 訊號作為時脈輸入，reset 訊號用於重置計數器，enable 訊號用於啟用計數器的計數操作，並通過 complete 和 count 訊號輸出計數器的狀態。它實現了計數器的功能，並在計數值達到最大值時發出完成訊號。

Controller：

Controller 在這個紅綠燈控制系統中負責控制紅綠燈的狀態，在 entity Controller 中定義了它的實體，它具有兩個 input：clk（時脈）、complete（完成），五個 output：reset（重置）、enable（啟用）、glight（綠燈）、ylight（黃燈）和 rlight（紅燈）。

- clk 是時脈訊號，用於驅動 Controller 的操作。
- reset 是重置訊號，由 Controller 組件生成並輸出，用於重置 Counter 組件。
- enable 是啟用訊號，由 Controller 組件生成並輸出，用於啟用 Counter 組件的計數操作。
- complete 是完成訊號，由 Counter 組件輸出並作為輸入給 Controller 組件，表示計數操作是否已完成。
- glight、ylight 和 rlight 分別是綠燈、黃燈和紅燈的輸出訊號。

在 architecture Behavioral of Controller 中，我們定義了一個狀態類型 State，其中包含六個狀態：green1、green2、red1、red2、yellow1 和 yellow2，初始狀態為 green1。

在主要的 process 中，我們檢測時脈訊號的 rising edge。如果是上升訊號則將 current_state 更新為 next_state，即進入下一個狀態。

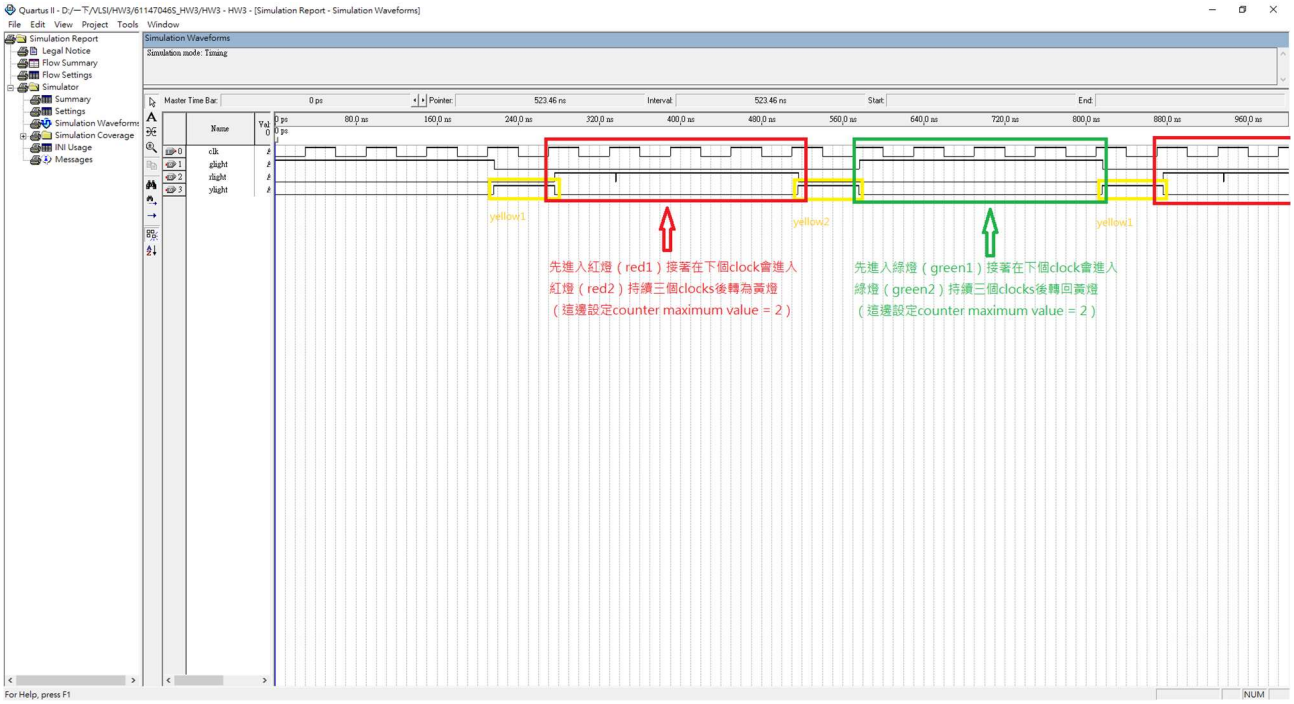
在第二個 process 中，我們根據 current_state 的值進行分支判斷，並對輸出訊號進行設置。根據不同的狀態，我們設置對應的綠燈、黃燈和紅燈訊號的值，同時設置 reset、enable 訊號的值以跟 Counter 進行溝通。

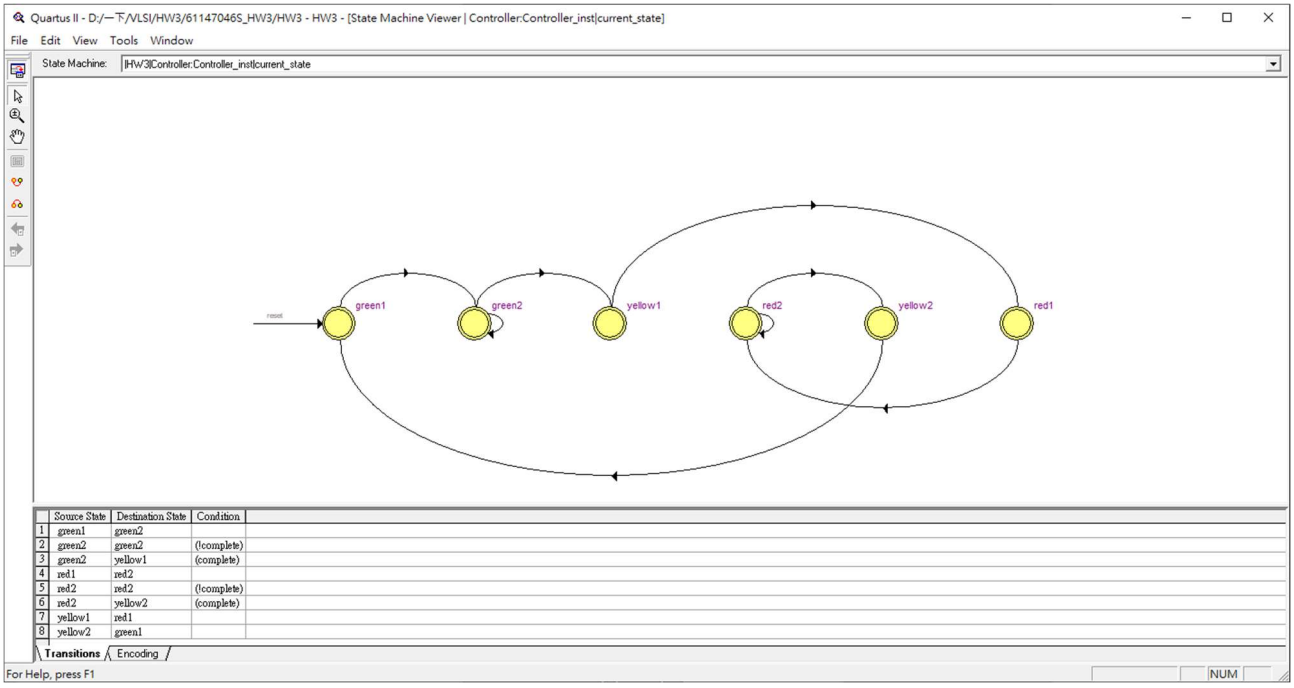
在每個狀態的最後，我們使用 next_state 變數指定下一個狀態。這樣在下一個時脈週期中，current_state 會被更新為 next_state，從而使控制系統

進入下一個狀態。

總結來說，Controller 組件根據紅綠燈系統的狀態進行控制，根據不同的狀態設置對應的輸出訊號，同時控制計數器的啟用。它根據時脈訊號和計數器的完成訊號進行狀態轉換，實現了紅綠燈系統的控制邏輯。

2. Waveform diagram here (Simulation Results)





3. Reflections and discussions

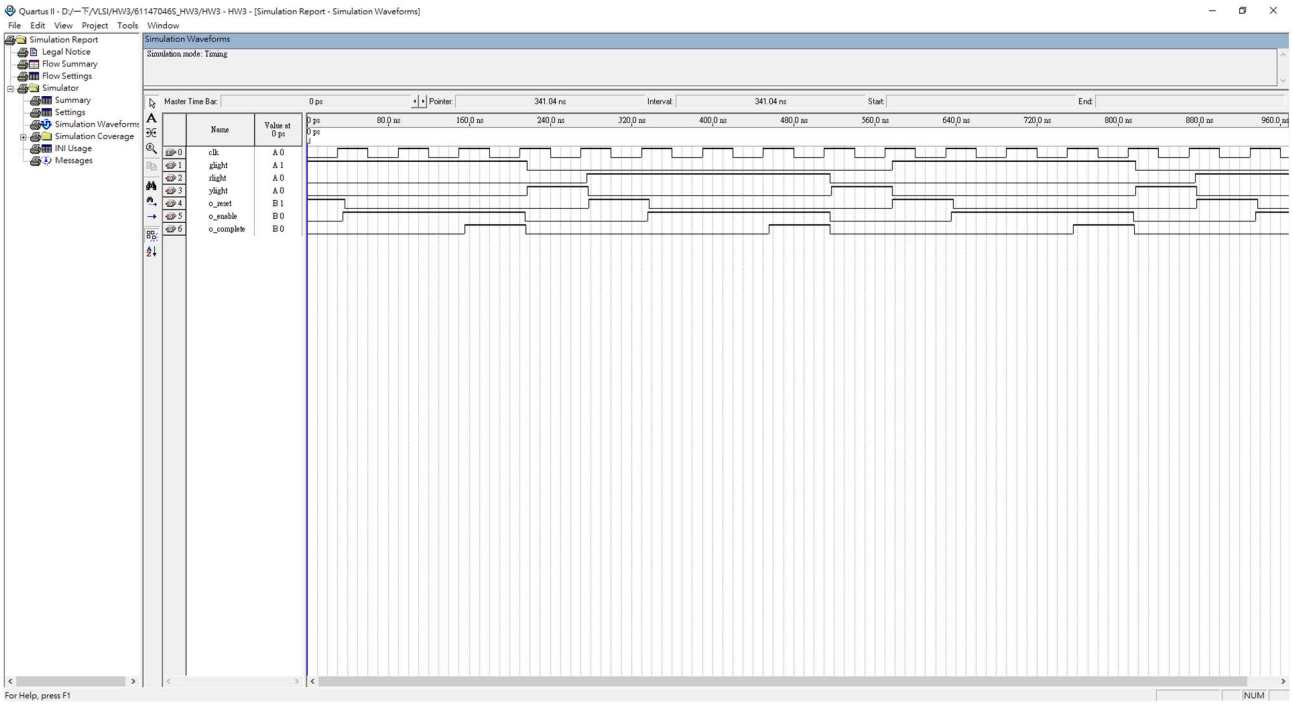
這個紅綠燈控制電路的設計過程中，我們延伸了上次的 component 用法。透過將系統切分為主程式及 Counter 和 Controller 兩個組件，實現了模組化和可重用的程式碼。

在 Counter 組件中，我們使用了一個計數器來實現紅綠燈燈號的切換。透過時脈信號的驅動，計數器可以在每個時脈週期上升一次，並在達到最大值時產生完成信號。這個計數器的設計簡單而直觀，可以應用於各種計數操作。

在 Controller 組件中，我們使用了狀態機的設計方法來控制紅綠燈的狀態轉換。透過定義不同的狀態和對應的輸出信號，我們可以在每個狀態中設定紅綠燈的亮滅以及計數器的啟用。

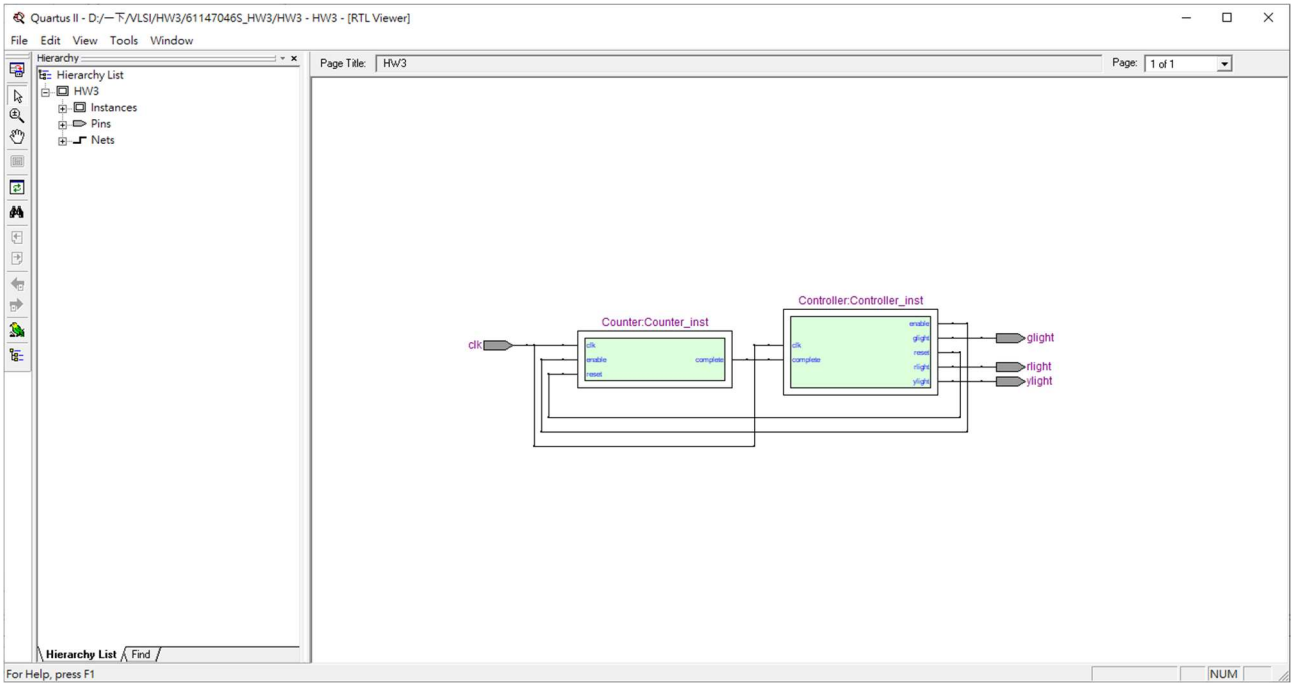
這次的作業使用狀態機的設計方法來實現複雜的控制邏輯。狀態機的設計使得控制系統的狀態轉換清晰可讀，並提供了靈活性和可擴展性。透過定義不同的狀態和對應的輸出控制，我們可以有效的控制紅綠燈系統的運作，輕鬆調整狀態的邏輯、順序。

另外有趣的是在此次實驗中一開始是先套用學姊的範例輸入 counter maximum value = 2，但顯示出來的波形圖數量卻不一樣，於是自己另外加上三個 output，觀察 reset、enable、complete 訊號之關聯性。



以上這張波型圖雖然跟學姊的範例波型圖一樣，但學姊設定的 counter maximum value 為 2，而我則是設定為 1。根據波型圖可觀察到進入 green1/red1 時發送 reset 訊號會消耗一個 clock，根據我撰寫 Counter 的程式邏輯 count: buffer integer range 0 to MAX_COUNT，因為 0 to 1 所以在 green2/red2 時會等待兩個 clocks，最後等待 complete 訊號也會消耗一個 clock，故 1+2+1=4 clocks。

附圖為產生出的 RTL 電路圖：



可觀察到 Counter 與 Controller 的 enable、reset、complete 訊號相互連接，且由 Controller 輸出綠、紅、黃三個燈號的訊號。