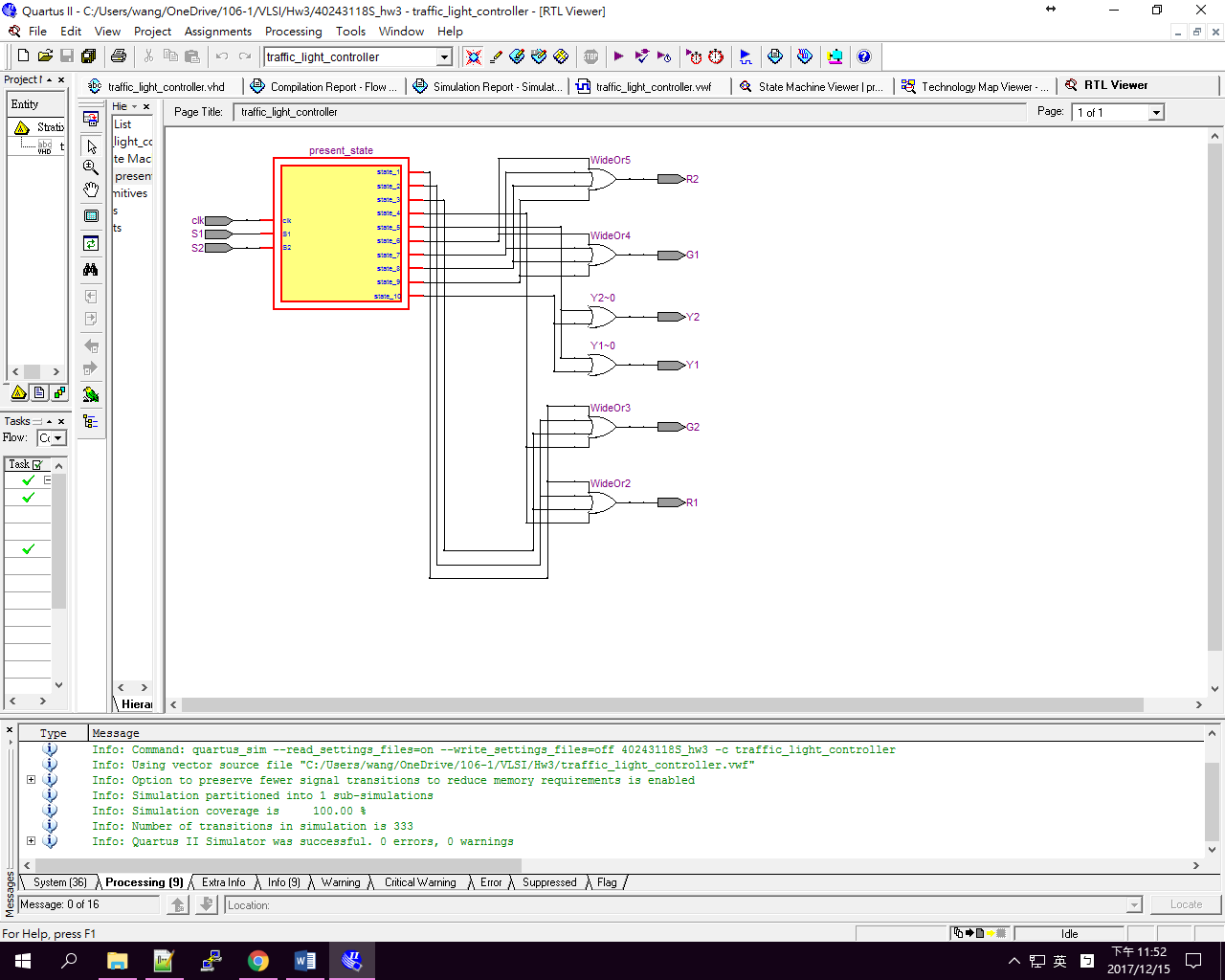
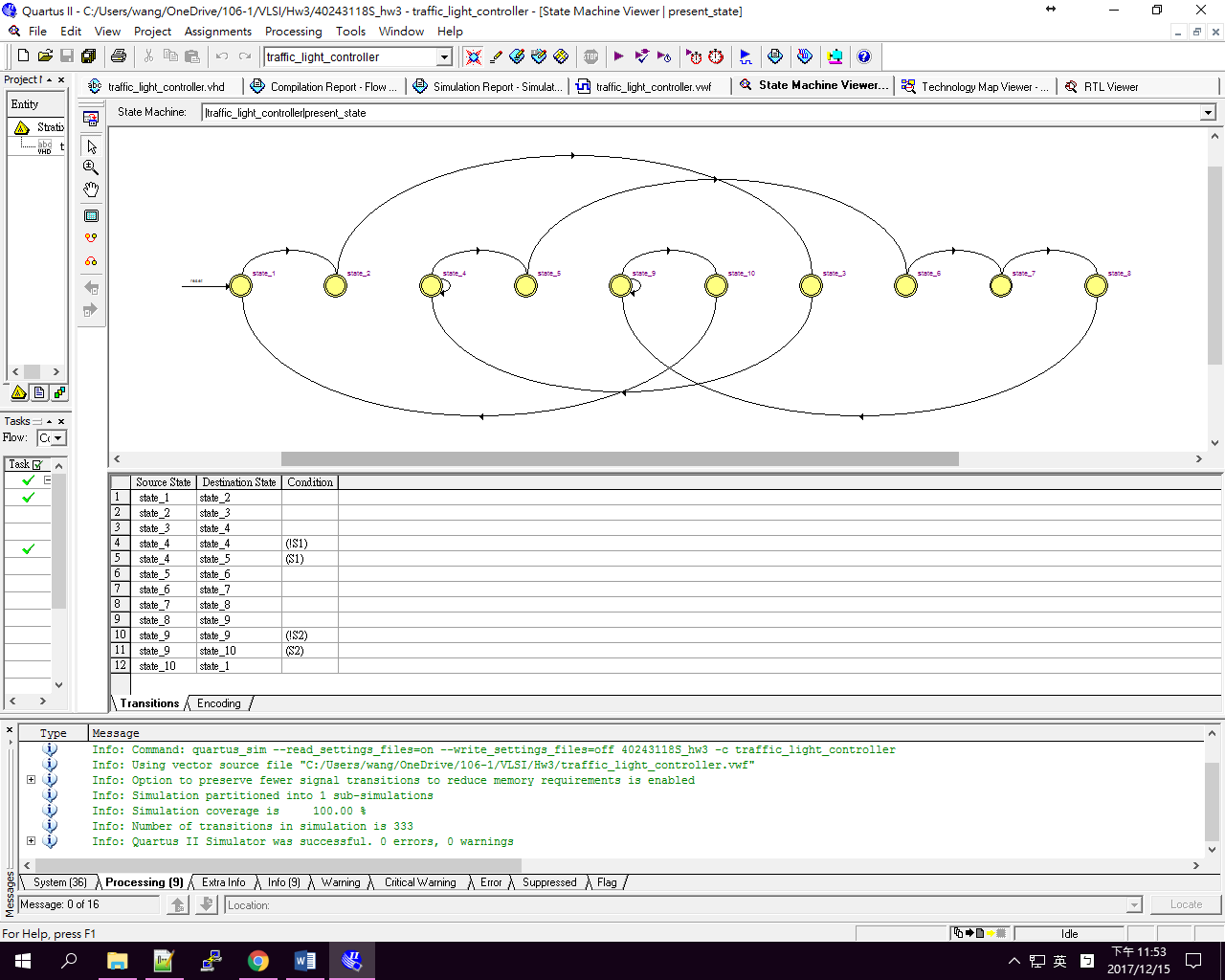
Hw 3 40243118S 王擎天

1. 設計電路說明(20%):

此線路圖是一個簡單的3個input和6個output組成,中間由一個finite state machine組成,它會在每一clock cycle更新內部的數值,並由input的兩個訊號S1,S2來調控更新的狀態,並由狀態來決定六個output的值.





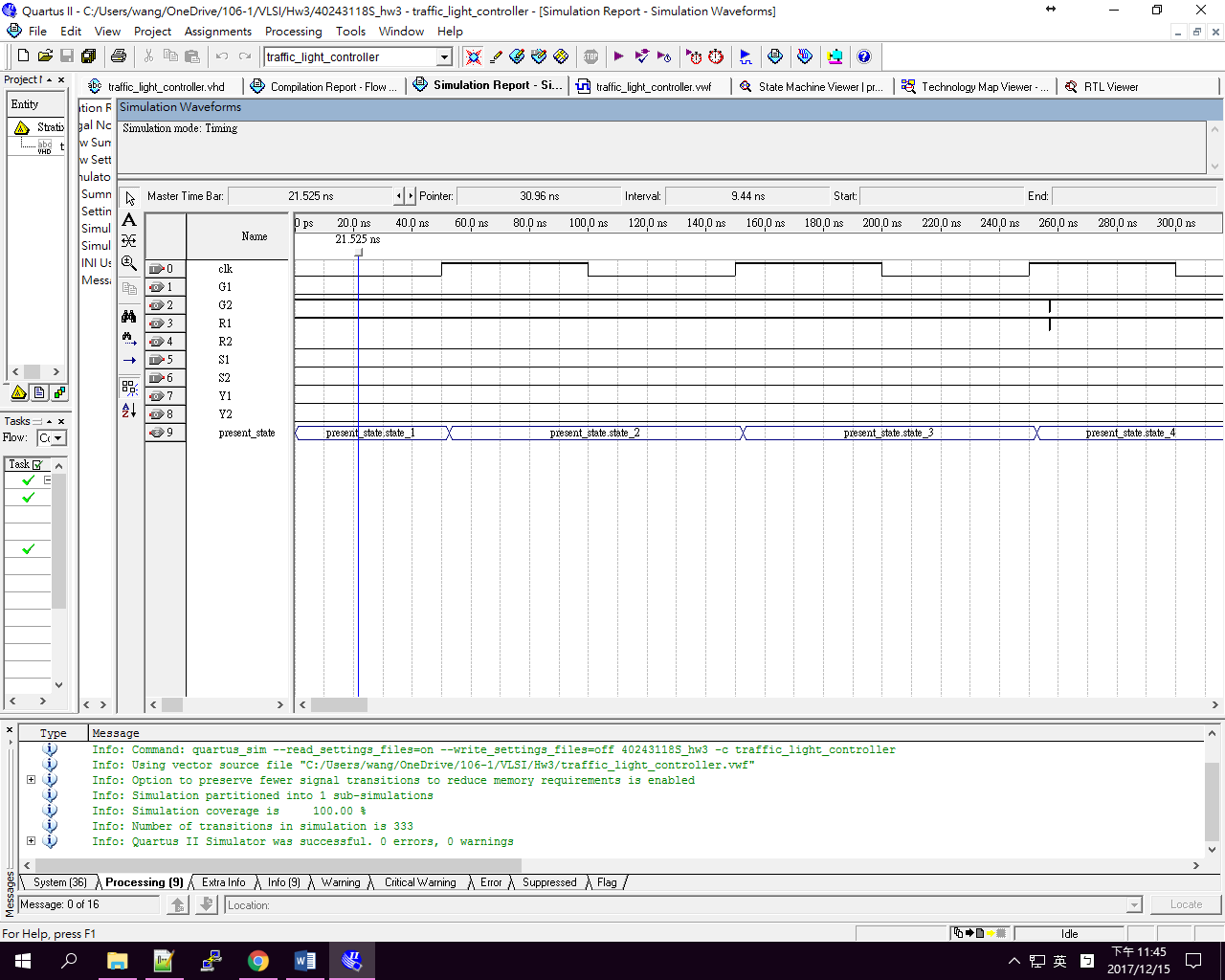
2. 程式碼說明(30%):

此程式由先宣告status\_type,並設定10種state組成,又創造兩個signal,一個存現在的狀態(present\_state),一個存將來的狀態(next\_state).每次clock rising edge發生時,就會更新六個output的狀態(R1,Y1,G1,R2,Y2,G2),並且用next\_state存下一步的數值,再用它跟改present\_state.其中,S1,S2可以用來調控state machine更新到下一步的狀態.

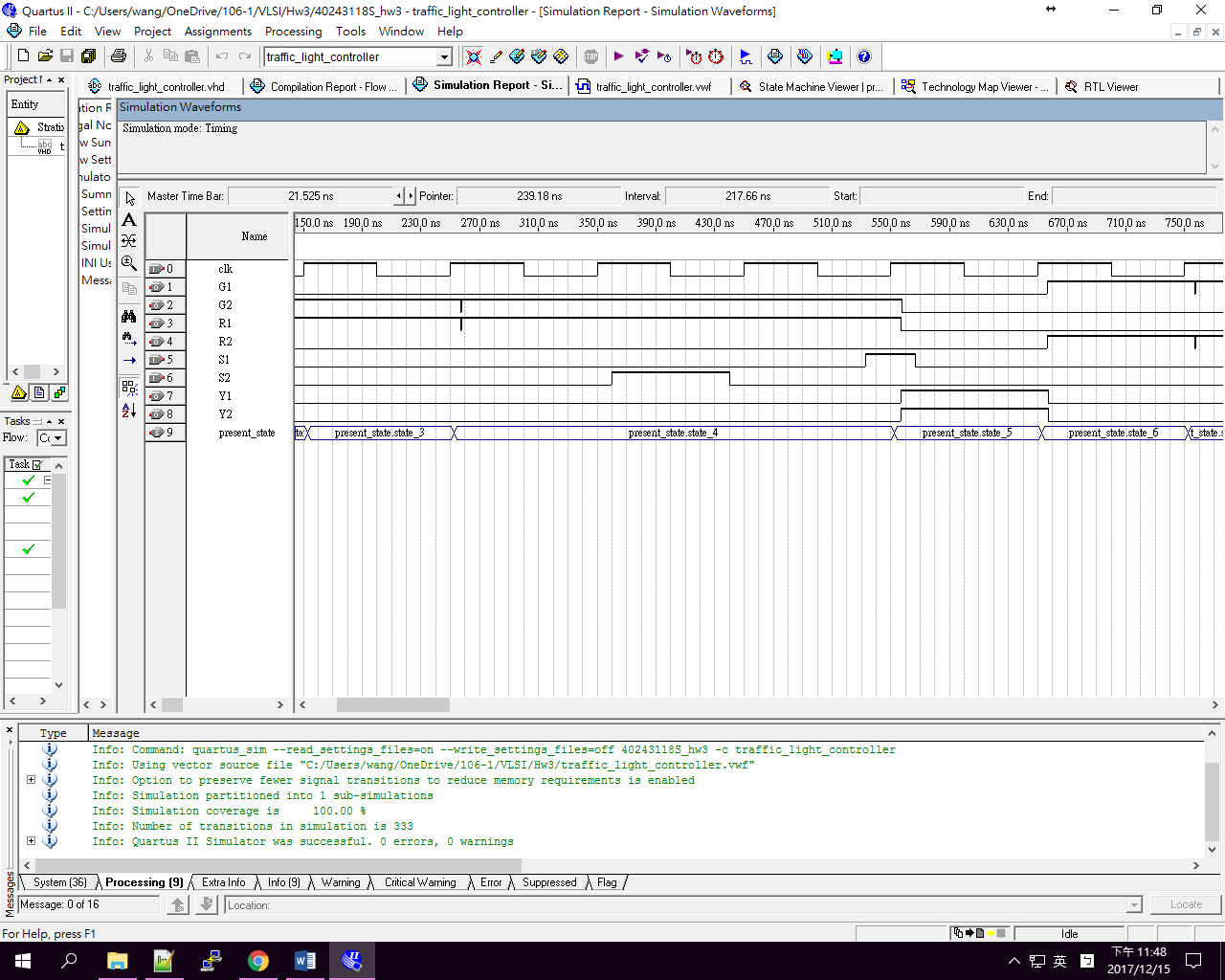
|  |
| --- |
| entity traffic\_light\_controller is  port(  S1,S2,clk:in bit;  R1,Y1,G1,R2,Y2,G2:out bit  );  end entity;  architecture arch\_traffic\_light\_controller of traffic\_light\_controller is  type state\_status is (state\_1,state\_2,state\_3,state\_4,state\_5,state\_6,state\_7,state\_8,state\_9,state\_10);  signal present\_state,next\_state: state\_status;  begin  process(present\_state,S1,S2)  begin  R1<='0';  G1<='0';  Y1<='0';  G2<='0';  R2<='0';  Y2<='0';  case present\_state is  when state\_1=>  next\_state<=state\_2;  R1<='1';  G2<='1';  when state\_2=>  next\_state<=state\_3;  R1<='1';  G2<='1';  when state\_3=>  next\_state<=state\_4;  R1<='1';  G2<='1';  when state\_4=>  if (S1='1') then  next\_state<=state\_5;  else  next\_state<=state\_4;  end if;  R1<='1';  G2<='1';  when state\_5=>  next\_state<=state\_6;  Y1<='1';  Y2<='1';  when state\_6=>  next\_state<=state\_7;  G1<='1';  R2<='1';  when state\_7=>  next\_state<=state\_8;  G1<='1';  R2<='1';  when state\_8=>  next\_state<=state\_9;  G1<='1';  R2<='1';  when state\_9=>  if(S2='1') then  next\_state<=state\_10;  else  next\_state<=state\_9;  end if;  G1<='1';  R2<='1';  when state\_10=>  next\_state<=state\_1;  Y1<='1';  Y2<='1';  end case;  end process;  process(clk)  begin  if( clk'EVENT and clk = '1') then  present\_state <=next\_state;  end if;  end process;  end architecture; end if;  G1<='1';  R2<='1';  when state\_10=>  next\_state<=state\_1;  Y1<='1';  Y2<='1';  end case;  end process;  process(clk,reset)  begin  if(reset='1') then  present\_state<=state\_1;  elsif( clk'EVENT and clk = '1') then  present\_state <=next\_state;  end if;  end process;  end architecture; |

3. 模擬結果說明(30%):

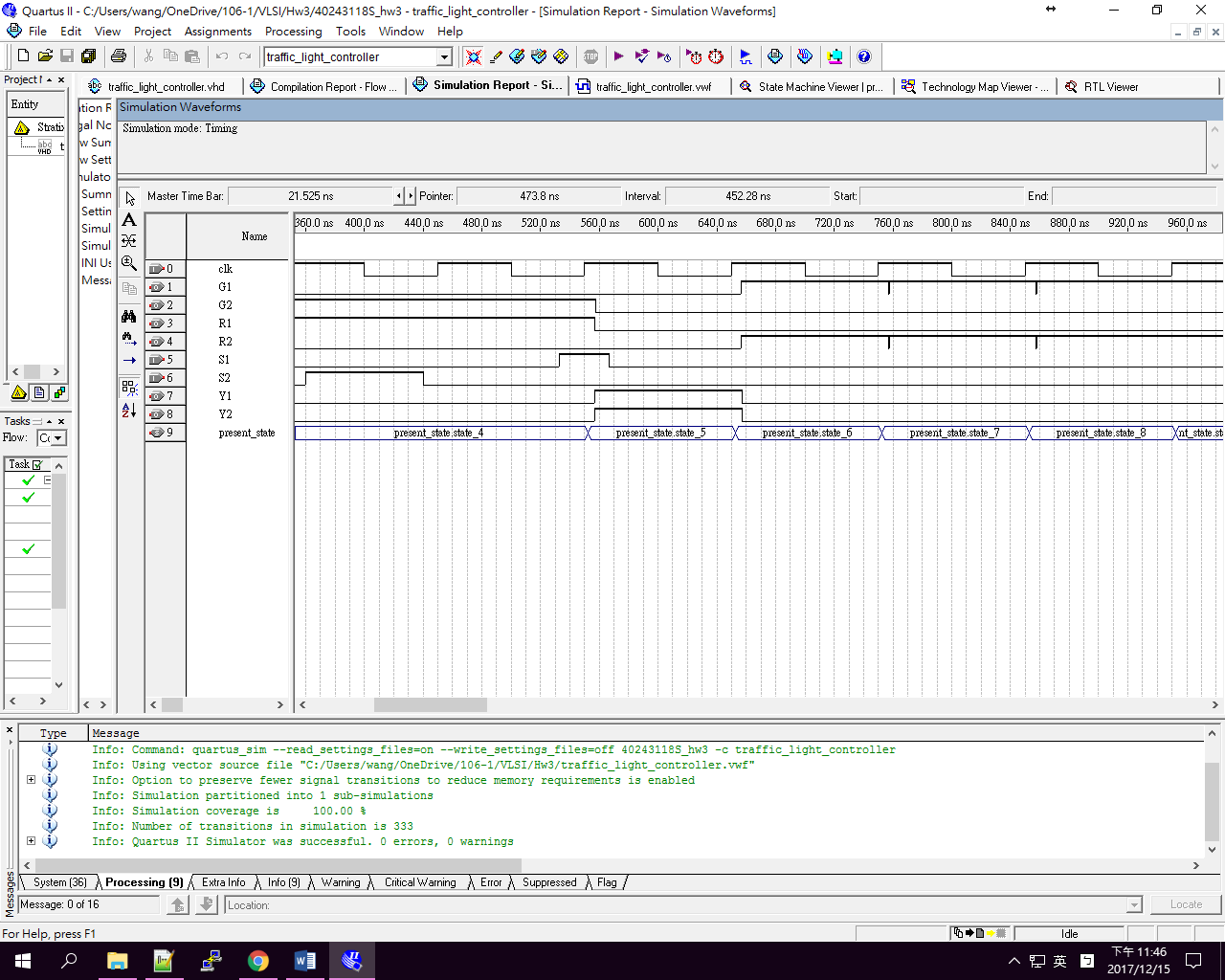
此圖呈現state從state 1開始隨著clock cycle跟新狀態的現象



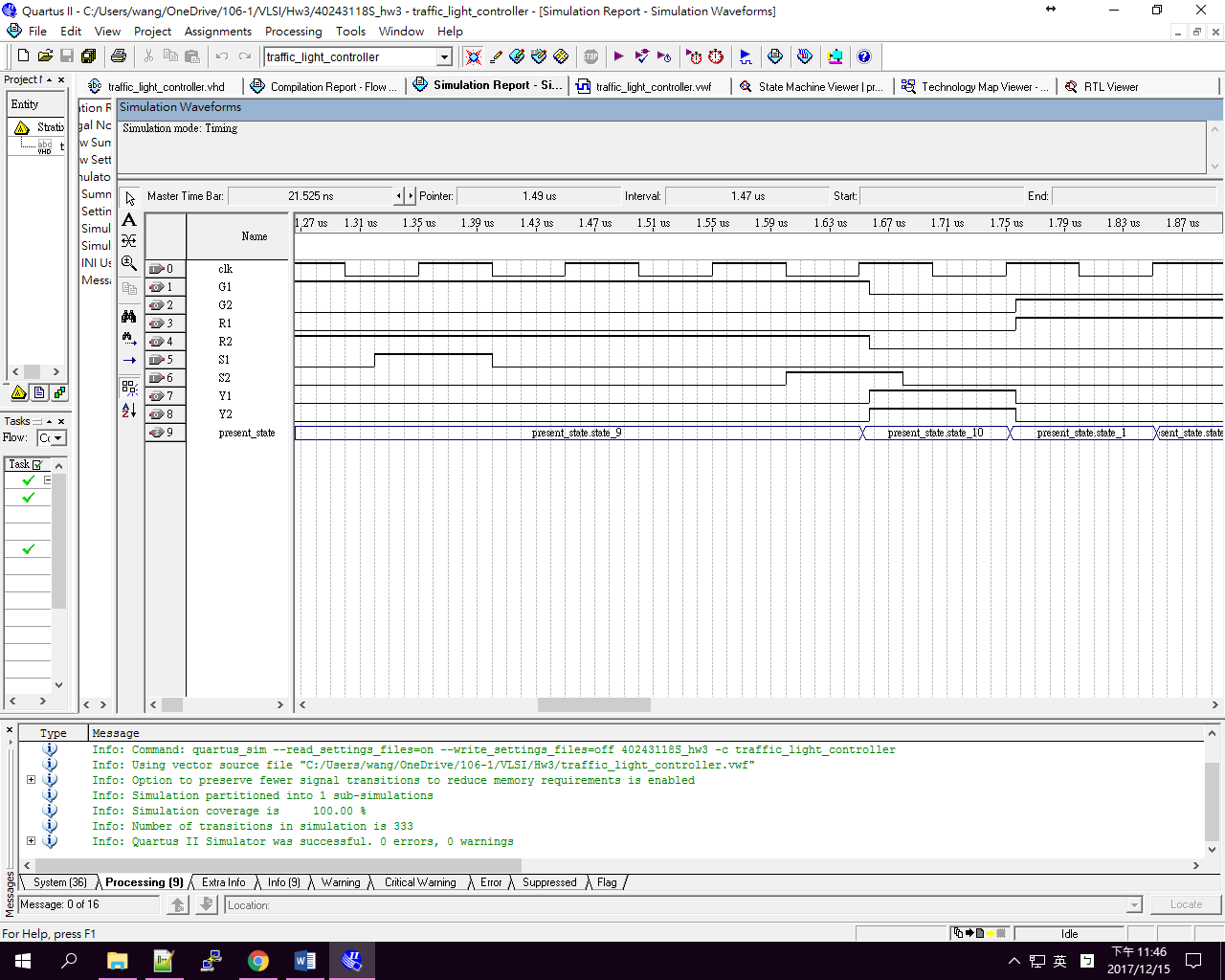
此圖呈現state 4只有在S1為一的狀態下才會更新到state 5,否則停留在state 4



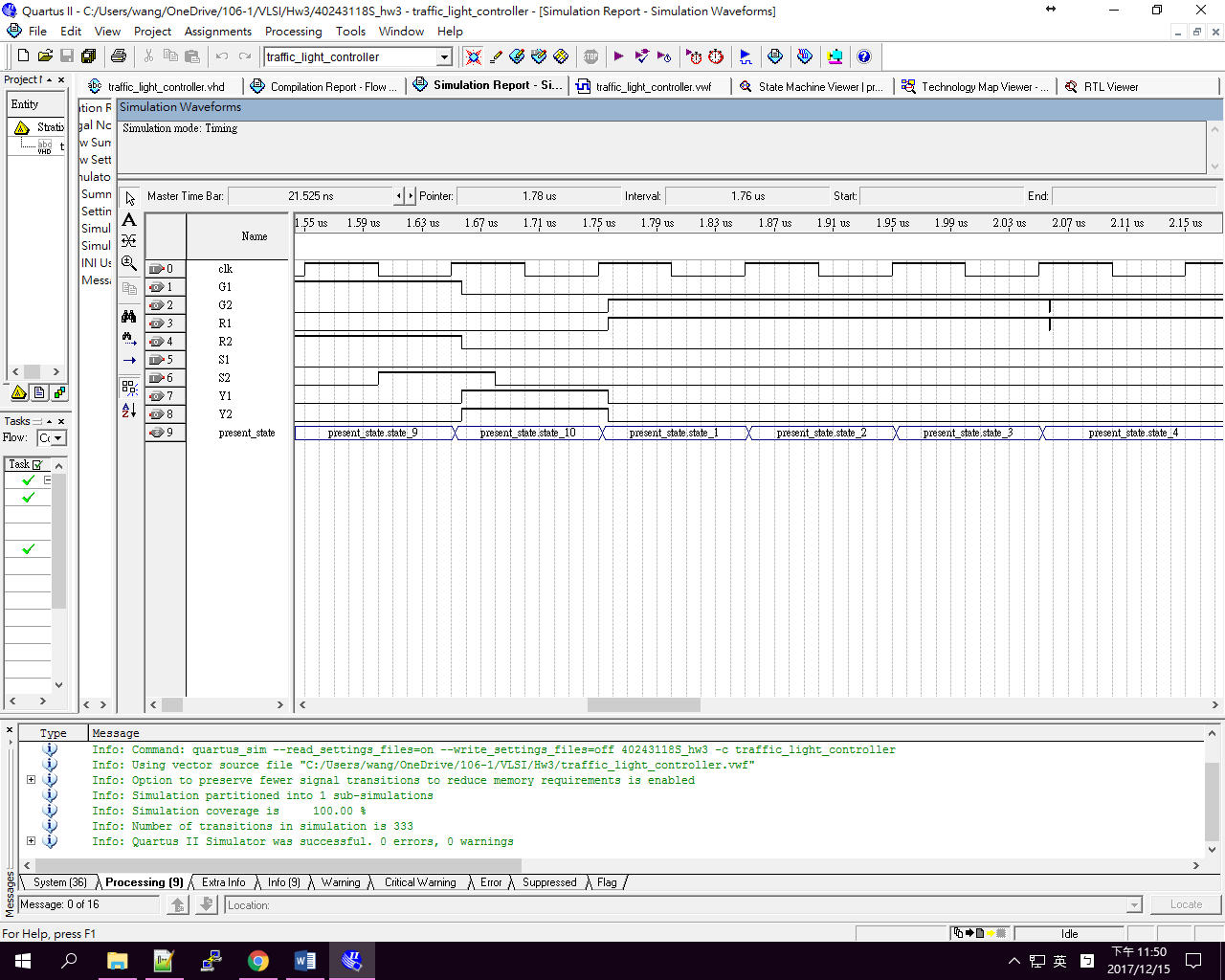
此圖呈現state隨著clock cycle跟新狀態的現象



此圖呈現state 9只有在S2為一的狀態下才會更新到state 10,否則停留在state 9



此圖呈現state 10後回到state 1,並開始一個新循環



4. 結論及心得(20%):

透過上圖的state machine viewer和模擬圖和模擬圖的所呈現每個state的狀態得知,此程式有正常運作和編寫.透過這次的練習我可以使用VHDL所提供的便利性產生出state machine的線路,並透過quartus看出state machine運作的機制.