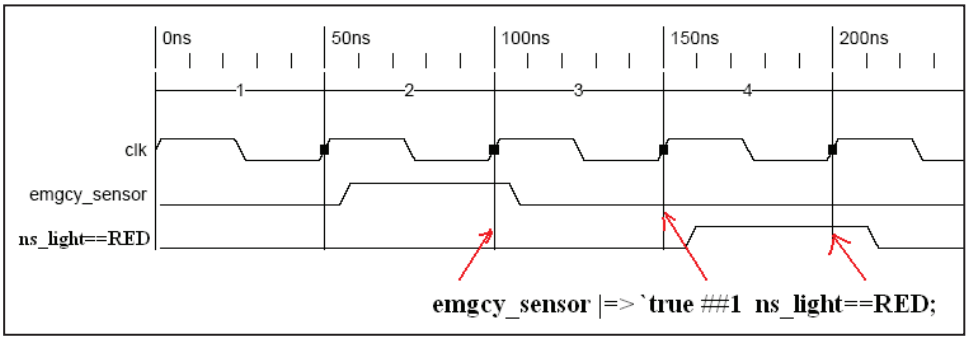
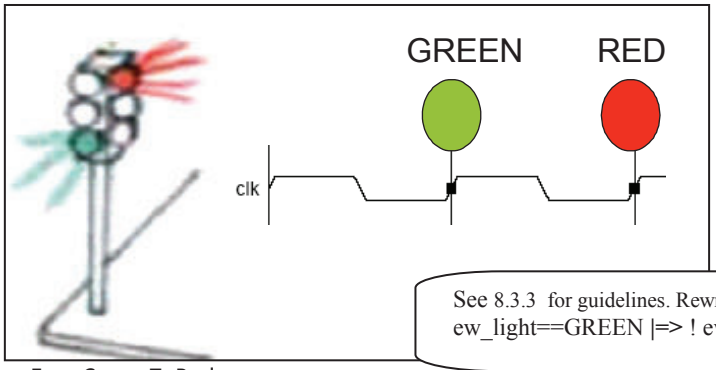
```
`define true 1
`ifndef MULTIPLE_FILE_COMPILE
  typedef enum {OFF, RED, YELLOW, GREEN, PRE_GREEN} lights_t;   `endif
module tlight_props  ( // ch7/7.3/tlight_props2.sv
   input  lights_t ns_light,  // North/South light status, Main road
   input  lights_t ew_light,  // East/West light status
   input          ew_sensor, // East/West sensor for new car
   input          emgcy_sensor, // emergency sensor
   input          reset_n,   // synchronous reset
   input          clk,       // master clock
   input [1:0]  ns_green_timer
   );
  parameter FAIL   = 1'b0;

  // ************************************************
  // Safety property
  property Never_NS_EW_ALL_GREEN;
      disable iff (!reset_n)
     not (ns_light==GREEN && ew_light==GREEN);
  endproperty : Never_NS_EW_ALL_GREEN
  Never_NS_EW_ALL_GREEN_1 : assert property(@ (posedge clk) Never_NS_EW_ALL_GREEN);
  // ************************************************
  // State of lights at reset
  property nsLightAtReset;
     // disable iff (!reset_n)  // <-- this causes the assertion to always be vacuous
     reset_n==1'b0 |=> ns_light==OFF;
  endproperty : nsLightAtReset
  nsLightAtReset_1 : assert property(@ (posedge clk) nsLightAtReset);
  //
  property ewLightAtReset;
     // disable iff (!reset_n) // <-- this causes the assertion to always be vacuous
    reset_n==1'b0 |=> ew_light==OFF;  // RED???
   endproperty : ewLightAtReset
  ewLightAtReset_1 : assert property(@ (posedge clk) ewLightAtReset);
  // ************************************************
 // State of lights during emergency
  // Lights switch from GREEN to YELLOW to RED
  property NsLightsWhenEmergency;
     disable iff (!reset_n)
    emgcy_sensor |=> `true[*2] ##1  ns_light==RED;
   endproperty : NsLightsWhenEmergency
  NsLightsWhenEmergency_1 : assert property(@ (posedge clk) NsLightsWhenEmergency);
```

The following is preferred (see 8.3.3)
`(ns_light==GREEN |-> !ew_light==GREEN)`




```
  property EwLightsWhenEmergency;
     disable iff (!reset_n)
     emgcy_sensor |=> `true[*2] ##1 ew_light==RED;
  endproperty : EwLightsWhenEmergency
  EwLightsWhenEmergency_1 : assert property(@ (posedge clk) EwLightsWhenEmergency);
```

```
// Safety, GREEN to RED is illegal.  Need YELLOW
  property NsNeverFromGreenToRed;
     disable iff (!reset_n)
     not(ns_light==GREEN ##1 ns_light==RED);
  endproperty : NsNeverFromGreenToRed
  NsNeverFromGreenToRed_1 : assert property(@ (posedge clk) NsNeverFromGreenToRed);
```

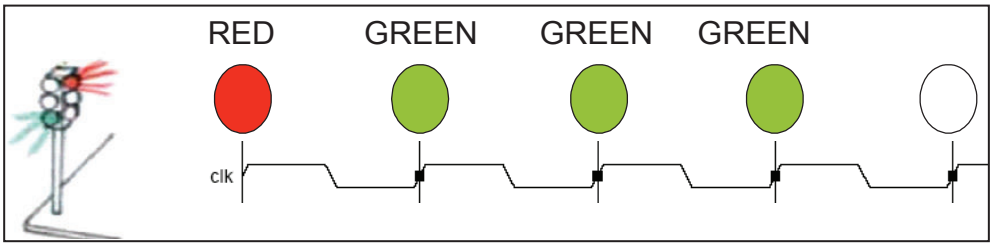



See 8.3.3 for guidelines. Rewrite as
ew_light==GREEN |=> ! ew_light==RED

```
property EwNeverFromGreenToRed;
    disable iff (!reset_n)
     not(ew_light==GREEN ##1 ew_light==RED);
endproperty : EwNeverFromGreenToRed
 EwNeverFromGreenToRed_1 : assert property(@ (posedge clk) EwNeverFromGreenToRed);
// ***********************************************
 // The NorthSouth light is the main street light.
// If ns is green and no emergency or ew sensor, then next cycle is also GREEN
property NsGreenNext;
   (ns_light==GREEN) && ($past(emgcy_sensor)==1'b0 && reset_n==1'b1)
                |=> ns_light==GREEN;
endproperty : NsGreenNext
NsGreenNext_1:  assert property (@ (posedge clk) NsGreenNext);
```

See 8.3.3 for guidelines. Rewrite as
(ew_light==GREEN |-> !ns_light==YELLOW)
and
(ns_light==GREEN |-> !ew_light==YELLOW)

```
  // GREEN-YELLOW at the same time
 property NeverGreenYellow;
    not ((ew_light==GREEN && ns_light==YELLOW) ||
         (ns_light==GREEN && ew_light==YELLOW));
 endproperty : NeverGreenYellow
 NeverGreenYellow_1: assert property (@ (posedge clk) NeverGreenYellow);
 // ***************************************************
// The NorthSouth light is the main street light.
// It must remain GREEN for ns_green_timer == 3 before it can switch.
// Timer ns_green_timer will count to 3, and remain at 3 until light changes.
  property NsGreenForMin3Ccyles;
   @ (posedge clk) disable iff (!reset_n || emgcy_sensor)
     $rose(ns_light==GREEN) && !$past(emgcy_sensor) |=>
           ns_light==GREEN[*2]; // abort emgcy_sensor);
  endproperty : NsGreenForMin3Ccyles
 NsGreenForMin3Ccyles_1 : assert property (NsGreenForMin3Ccyles);
```




```
// ***********************************************
```

```
  // East-West North-South Lights with East-West sensor
  // If ew_sensor is activated (new car), then light will switch for the ew_light
  // when minimum time for ns_light is satisfied.  ew_green_timer will count to 3,
  // at which time, the ns_green_timer will regain control of GREEN.
    property EwNewSensorActivation;
     @ (posedge clk) disable iff (!reset_n || emgcy_sensor)
      ( (ew_sensor==1'b1) && $rose(ns_green_timer==2'b11 )) &&
                           !$past(emgcy_sensor) && ns_light!= RED
                                 |=>   ns_light==YELLOW ##1 ew_light==GREEN;
    endproperty : EwNewSensorActivation
    EwNewSensorActivation_1 : assert property (EwNewSensorActivation);
// End of new properties 09/10/09
 endmodule : tlight_props

bind trafficlight tlight_props tlight_props1 (.*);
```

### 7.3.3 Verification

The above model was verified with *OneSpin 360 MV*, and it revealed several failures in the design, as shown in Figure 7.4.3-1.[61] In that figure, "fail (9)" means that the tool detected a violation of the property starting 9 cycles after reset.

As an example of debugging a failing property, Figure 7.4.3-2 shows the debugging view for the first property "*Never_NS_EW_ALL_GREEN*":

> The left part of the debugging window shows an interactive view of the property, with the failing parts highlighted in red (see *ch6/tlight/1_some_fail.png* file for a color view of a larger image). The waveform shows that indeed in cycle 0, both the EW and the NS lights are green[62]. Further, it indicates that some steps earlier, the emergency sensor, and the EW sensor were activated. To explore this situation, the time-point "-2" has been selected (indicated by the yellow vertical bar). The active source code annotation in the upper right corner shows the critical part of the DUV, with the active source lines marked in red: the root cause is the conditional transition from RED to the PRE_GREEN state, the condition
>
> ```
> if (ns_green_timer==3'b11 && ew_sensor==1'b1)
> ```
>
> being satisfied although in fact the NS light is not green, but being switched to green in the same step.

The sequence of events leading to this situation is fairly complex, and would have required extensive simulation with pseudo-random patterns to arrive at the failed situation. The bug, together with the other bugs detected by the formal tool, led to a thorough redesign of the controller, as discussed in the next section.

---

[61] *OneSpin*'s *360 MV* is a family of formal verification tools ranging from fully automatic RTL checks for large designs all the way to *OneSpin*'s patented gap-free verification. http://www.onespin-solutions.com/

[62] The cycles are numbered such that the property always starts at cycle 0, while the reset cycle is at some negative number, not shown in the figure.