Embedded System Design
TDDI08

Traffic Light Controller

Yi-Jie Chow

1. System Diagram

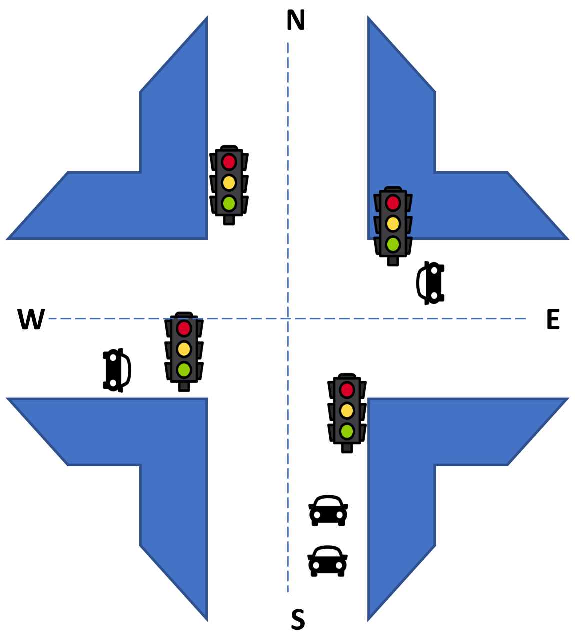


Figure1. Traffic scenario

Figure 1 shows the scenario of traffic with four kinds of direction, NS, SN, WE, EW. And there are four traffic sensors in front of each direction respectively to control the traffic light.

1.1. Counter

Generate random input traffic for NS, SN, WE, EW direction. 1 presents there is a car coming in that direction and 0 presents there is no car coming in that direction.

1.2. TrafficSensor

The traffic sensor will sense the input from the counter, and do the calculation to control the traffic light. The main goal is to make all cars wait in the minimum time and do not hit themselves.

The idea of the system is to calculate the number of cars in each direction, for example, in figure 1 there is 0 car in front of NS and 2 cars in SN, in contrast, there is 1 car in WE and 1 car in EW. Thus, the traffic sensor will let the traffic light in WE and EW work first because there are more cars are waiting in the horizontal direction.

If the number of cars is identical in both horizontal and vertical direction, the sensor will let the vertical one go first.

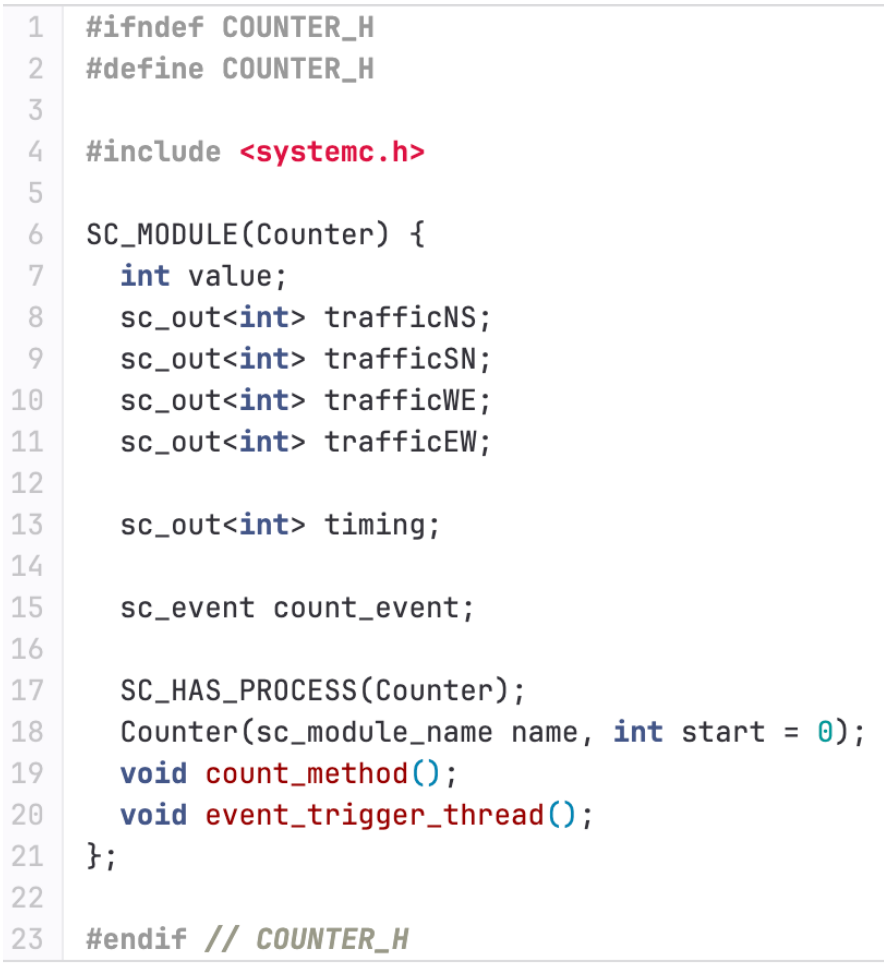
Furthermore, to prevent the car always come in the same direction, there is a timeout mechanism. If the traffic light keep light in the same direction for longer than 5 time slots, even though there is still a car coming in that direction, the sensor will force the light red and let the cars in other direction go first.

1.3. Monitor

Monitor.cc will write the input traffic and result traffic light every time unit to the output.txt file.

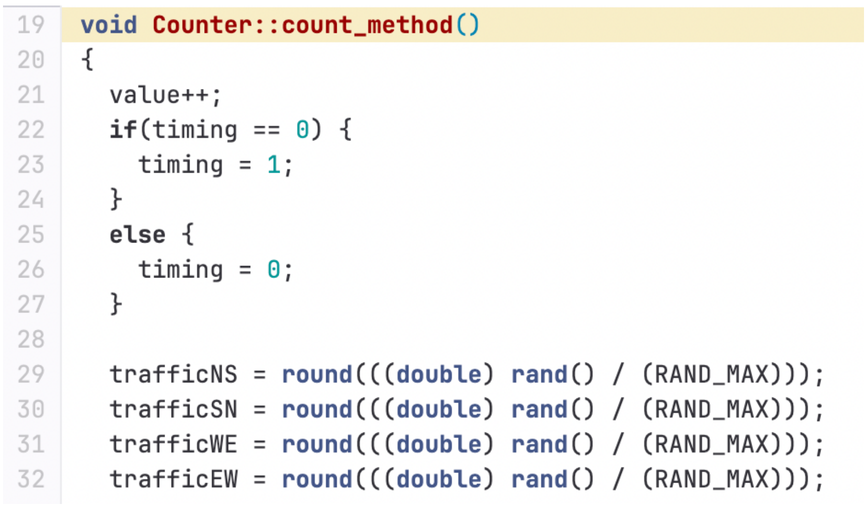
2. Control Logic

2.1. Counter.cc



```
#ifndef COUNTER_H
#define COUNTER_H

#include <systemc.h>

SC_MODULE(Counter) {
  int value;
  sc_out<int> trafficNS;
  sc_out<int> trafficSN;
  sc_out<int> trafficWE;
  sc_out<int> trafficEW;

  sc_out<int> timing;

  sc_event count_event;

  SC_HAS_PROCESS(Counter);
  Counter(sc_module_name name, int start = 0);
  void count_method();
  void event_trigger_thread();
};

#endif // COUNTER_H
```

Figure2. counter.h



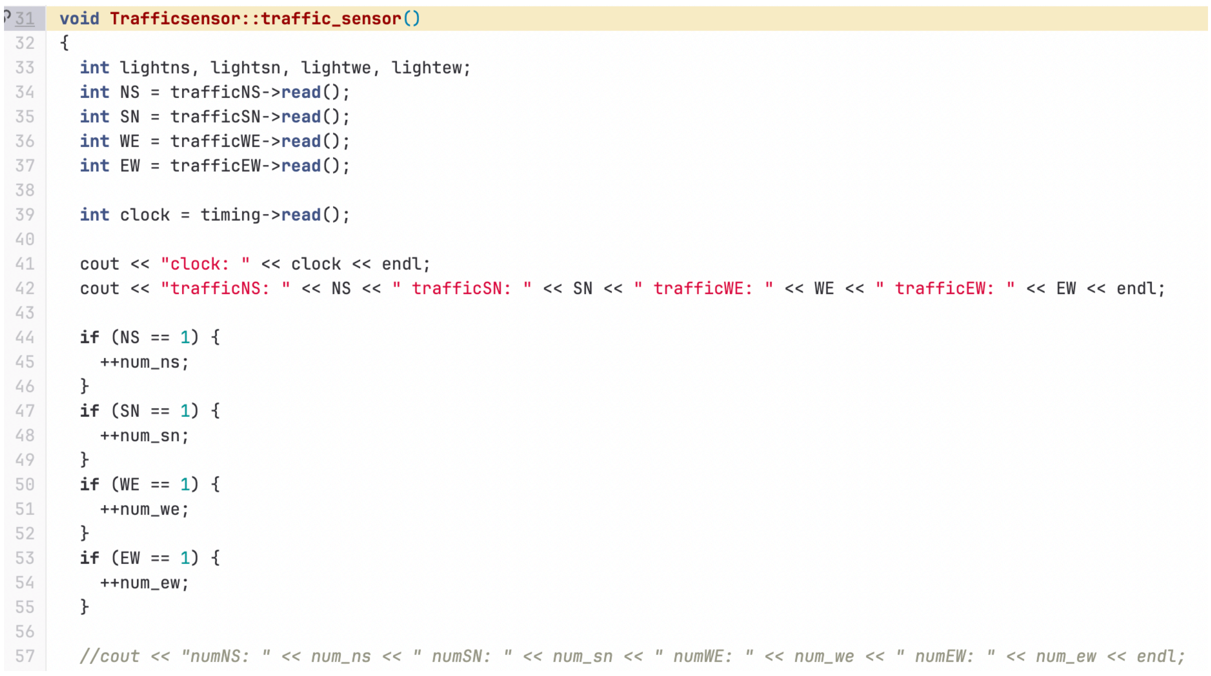
```
void Counter::count_method()
{
  value++;
  if(timing == 0) {
    timing = 1;
  }
  else {
    timing = 0;
  }

  trafficNS = round(((double) rand() / (RAND_MAX)));
  trafficSN = round(((double) rand() / (RAND_MAX)));
  trafficWE = round(((double) rand() / (RAND_MAX)));
  trafficEW = round(((double) rand() / (RAND_MAX)));
```

Figure3. counter.cc

Variable timing will be change between 0 or 1, to record the clock and let the trafficsensor.cc to be sensitive the clock prevents the random input traffic may be the same between the adjacent time unit.

And trafficNS, trafficSN, trafficWE, trafficEW is the random input between 0 or 1, and then create the channel and link the ports to the traffic sensor.

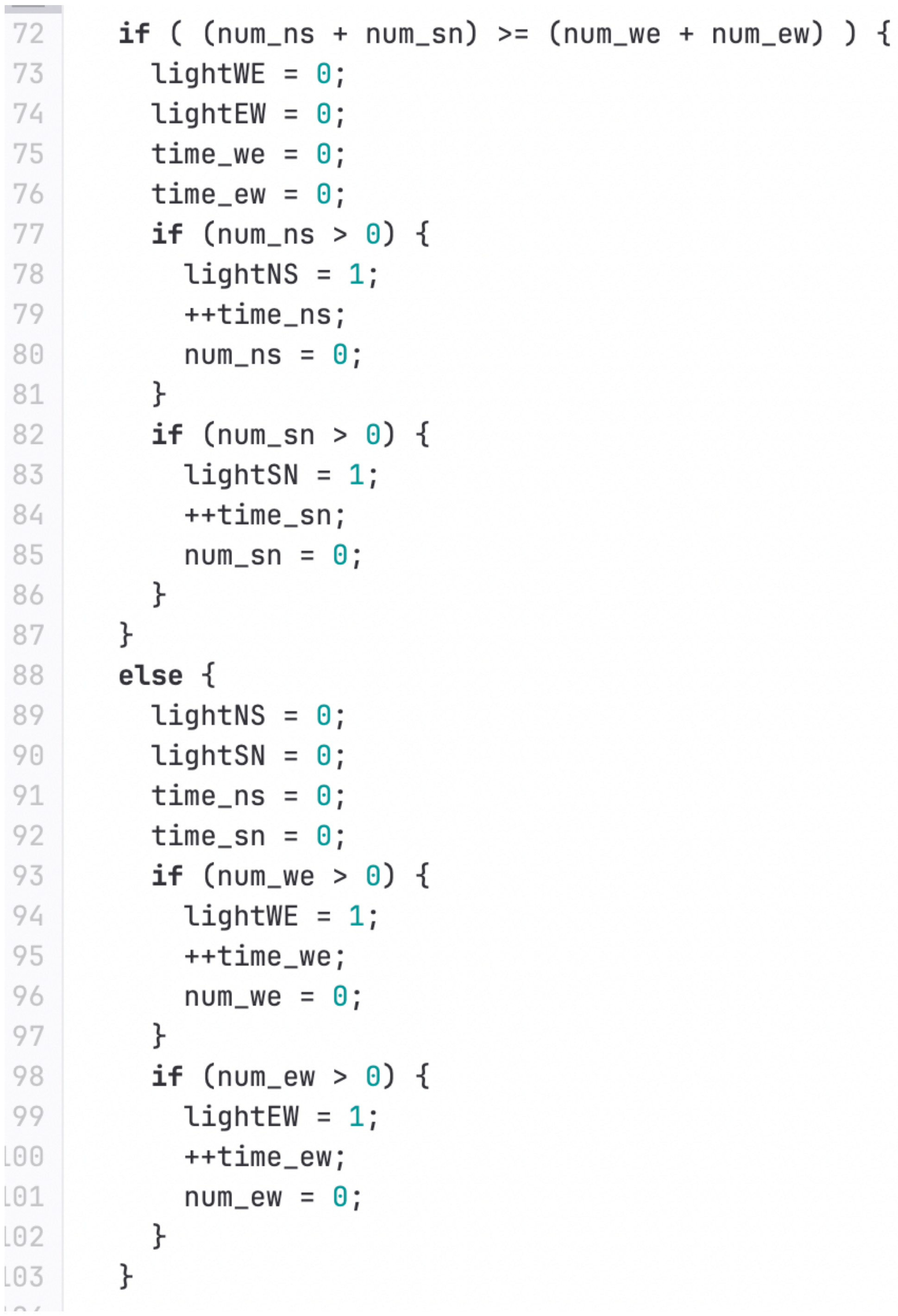
2.2. TrafficSensor.cc



```
void Trafficsensor::traffic_sensor()
{
  int lightns, lightsn, lightwe, lightew;
  int NS = trafficNS->read();
  int SN = trafficSN->read();
  int WE = trafficWE->read();
  int EW = trafficEW->read();

  int clock = timing->read();

  cout << "clock: " << clock << endl;
  cout << "trafficNS: " << NS << " trafficSN: " << SN << " trafficWE: " << WE << " trafficEW: " << EW << endl;

  if (NS == 1) {
    ++num_ns;
  }
  if (SN == 1) {
    ++num_sn;
  }
  if (WE == 1) {
    ++num_we;
  }
  if (EW == 1) {
    ++num_ew;
  }

  //cout << "numNS: " << num_ns << " numSN: " << num_sn << " numWE: " << num_we << " numEW: " << num_ew << endl;
```

Figure4. Trafficsensor.cc1

Traffic sensor will read the input values from counter.cc and if there is a car coming, the number of car will increase 1 in the corresponding direction.

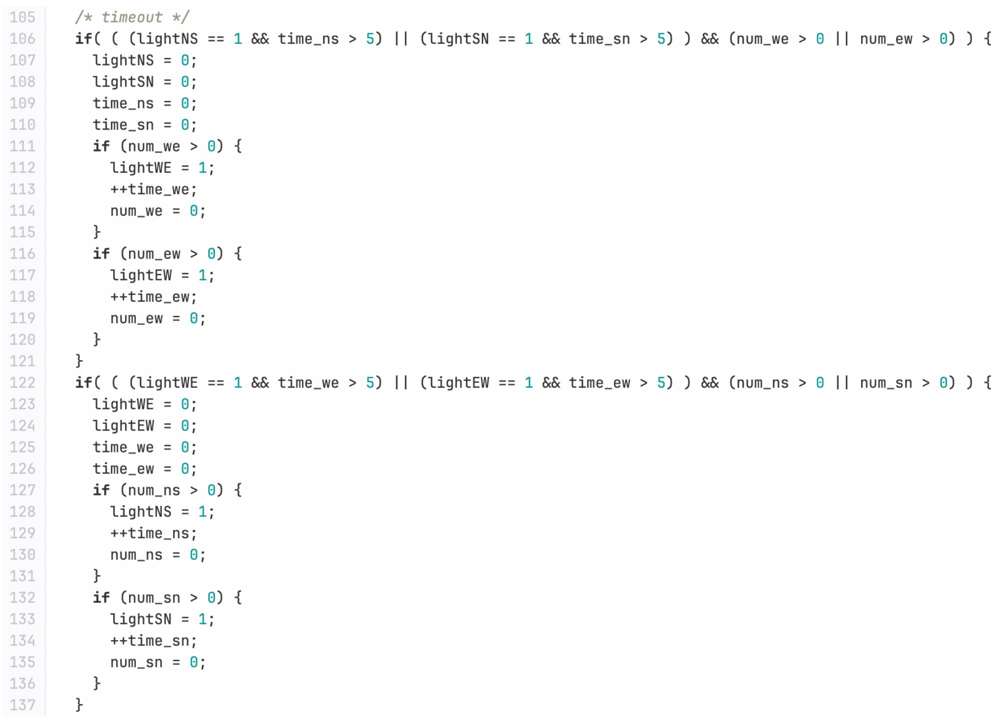


```
if ( (num_ns + num_sn) >= (num_we + num_ew) ) {
  lightWE = 0;
  lightEW = 0;
  time_we = 0;
  time_ew = 0;
  if (num_ns > 0) {
    lightNS = 1;
    ++time_ns;
    num_ns = 0;
  }
  if (num_sn > 0) {
    lightSN = 1;
    ++time_sn;
    num_sn = 0;
  }
}
else {
  lightNS = 0;
  lightSN = 0;
  time_ns = 0;
  time_sn = 0;
  if (num_we > 0) {
    lightWE = 1;
    ++time_we;
    num_we = 0;
  }
  if (num_ew > 0) {
    lightEW = 1;
    ++time_ew;
    num_ew = 0;
  }
}
```

Figure5. Trafficsensor.cc2

If the number of cars in vertical direction are more or equal than the horizontal direction, the light in NS or SN will be triggered independently and clear the number of cars in the corresponding direction. (Same as the horizontal direction.)

Moreover, in order to implement the timeout mechanism, there are variable time_ns, time_sn, time_we, time_ew to record the continuous time of the corresponding green light.

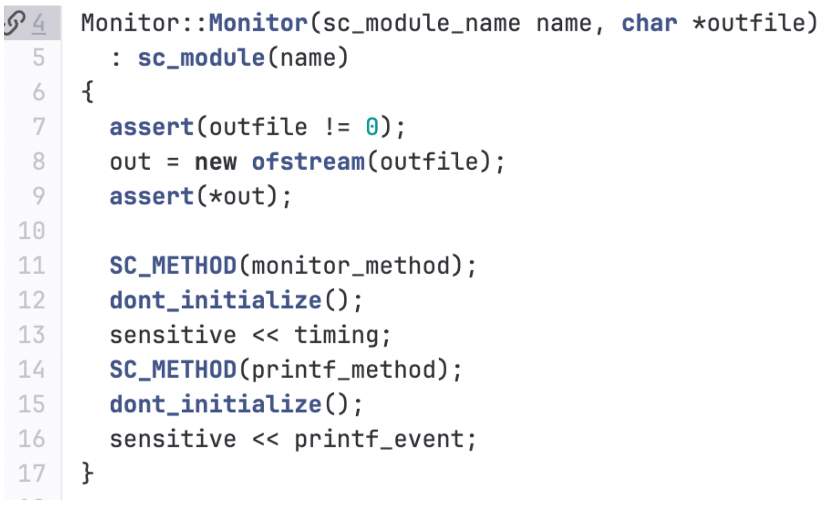


```
/* timeout */
if( ( (lightNS == 1 && time_ns > 5) || (lightSN == 1 && time_sn > 5) ) && (num_we > 0 || num_ew > 0) ) {
  lightNS = 0;
  lightSN = 0;
  time_ns = 0;
  time_sn = 0;
  if (num_we > 0) {
    lightWE = 1;
    ++time_we;
    num_we = 0;
  }
  if (num_ew > 0) {
    lightEW = 1;
    ++time_ew;
    num_ew = 0;
  }
}
if( ( (lightWE == 1 && time_we > 5) || (lightEW == 1 && time_ew > 5) ) && (num_ns > 0 || num_sn > 0) ) {
  lightWE = 0;
  lightEW = 0;
  time_we = 0;
  time_ew = 0;
  if (num_ns > 0) {
    lightNS = 1;
    ++time_ns;
    num_ns = 0;
  }
  if (num_sn > 0) {
    lightSN = 1;
    ++time_sn;
    num_sn = 0;
  }
}
```

Figure6. Trafficsensor.cc3

The timeout mechanism is to prevent the light keep be green more than 5 timestamp. The idea is if there is a car coming but the time of corresponding direction has more over than 5 timestamp, and there is a car waiting in front of different direction, then the sensor will force the light to be red instead of green, and let the traffic light of different direction to be green.

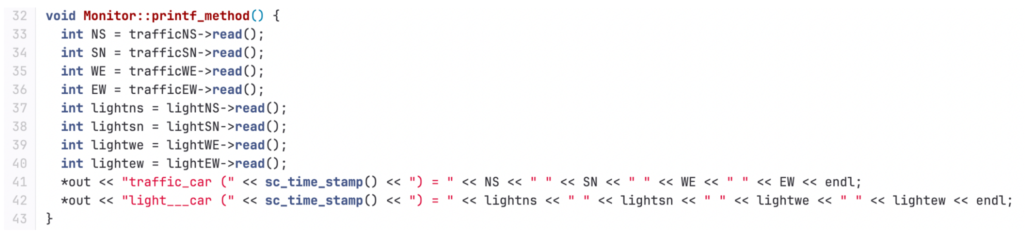
2.3. Monitor.cc



```
Monitor::Monitor(sc_module_name name, char *outfile)
  : sc_module(name)
{
  assert(outfile != 0);
  out = new ofstream(outfile);
  assert(*out);

  SC_METHOD(monitor_method);
  dont_initialize();
  sensitive << timing;
  SC_METHOD(printf_method);
  dont_initialize();
  sensitive << printf_event;
}
```

Figure7. Monitor.cc1

The monitor.cc has a module and sensitive to the timing (clock) to prevent the same input.

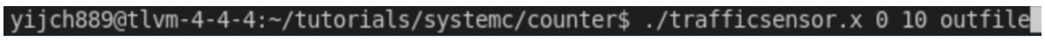


```cpp
void Monitor::printf_method() {
  int NS = trafficNS->read();
  int SN = trafficSN->read();
  int WE = trafficWE->read();
  int EW = trafficEW->read();
  int lightns = lightNS->read();
  int lightsn = lightSN->read();
  int lightwe = lightWE->read();
  int lightew = lightEW->read();
  *out << "traffic_car (" << sc_time_stamp() << ") = " << NS << " " << SN << " " << WE << " " << EW << endl;
  *out << "light___car (" << sc_time_stamp() << ") = " << lightns << " " << lightsn << " " << lightwe << " " << lightew << endl;
}
```

Figure8. Monitor.cc2

Read the input value from counter.cc and result from trafficsendor.cc then write into the ouput.txt file.

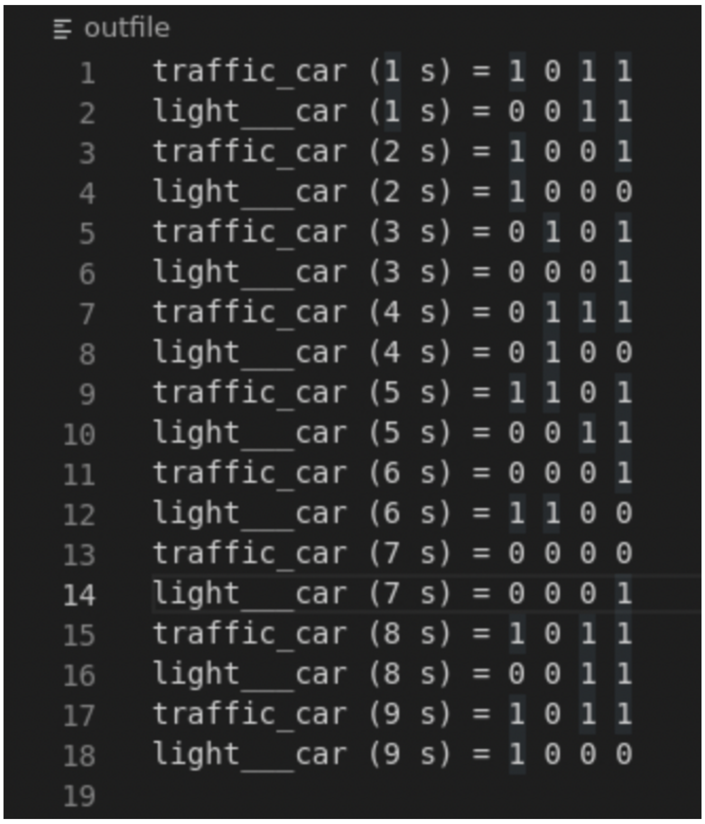
3. Result



```
yijch889@tlvm-4-4-4:~/tutorials/systemc/counter$ ./trafficsensor.x 0 10 outfile
```

Figure 9. simulation command

The first argument is the initial value of the time unit, the second argument is the simulation time, the third argument is the output file name you want to write in.



```
outfile
traffic_car (1 s) = 1 0 1 1
light___car (1 s) = 0 0 1 1
traffic_car (2 s) = 1 0 0 1
light___car (2 s) = 1 0 0 0
traffic_car (3 s) = 0 1 0 1
light___car (3 s) = 0 0 0 1
traffic_car (4 s) = 0 1 1 1
light___car (4 s) = 0 1 0 0
traffic_car (5 s) = 1 1 0 1
light___car (5 s) = 0 0 1 1
traffic_car (6 s) = 0 0 0 1
light___car (6 s) = 1 1 0 0
traffic_car (7 s) = 0 0 0 0
light___car (7 s) = 0 0 0 1
traffic_car (8 s) = 1 0 1 1
light___car (8 s) = 0 0 1 1
traffic_car (9 s) = 1 0 1 1
light___car (9 s) = 1 0 0 0
```

Figure 10. simulation result

The simulation result is shown above in figure 10, traffic_car presents if there is a car coming in the corresponding direction. For example, in the first time unit there is 1 car coming in NS, 0 car is coming in SN, 1 car is coming in WE, 1 car is coming in EW. Since the total cars in horizontal direction, i.e., WE and EW direction, is more than the vertical direction, i.e,. NS and SN direction; thus, the sensor will trigger the light in WE and EW light first. As the light_car shown in 1s, the light will be triggered as 0 0 1 1. For the second time unit, there is a car coming in NS, 0 car is coming in SN, 0 car is coming in WE, 1 car is coming in EW; moreover, there is still a car waiting in previous time unit in NS, so the total number in time unit 2 now is 2 0 0 1. Since the number of cars in the vertical direction is more than horizontal direction, the sensor will trigger the lights turn red in both WE and EW direction and trigger the light turns green in NS, but there is no car in SN so it will remain red. (works independently)