HACETTEPE UNIVERSITY
Computer Engineering Department
BBM233 Logic Design Laboratory
Fall 2019

Martian Underground Road Intersection Controller

Mehmet Giray Nacakcı

21989009

Section: 2

Instructors: Selma Dilek, Dr. Cemil Zalluhoğlu

Submission Date: 8.01.2020

BACKGROUND

As technology advanced, and concerns about humanity's future on Earth increased, arguments favoring space colonization gained momentum. Both private and public Space agencies and organizations made serious commitments to researching the viability of long-term colonization efforts, and to taking steps toward a permanent human presence on Mars, which is considered as the next giant leap for humankind.

After a long period of research and development, the human race is finally ready to launch its first batch of Mars settlers in 2020. Since colonization requires the establishment of permanent habitats that have the potential for self-expansion and self-sustenance, the astronaut selection program was very rigorous. Nevertheless, due to your extreme competence and skillfulness in digital logic design, you, the students of Hacettepe Computer Engineering Department, were unanimously selected to head to Mars as the Primary Logic Design Team in charge of all computer hardware and electronics projects that will be implemented on Mars.

The mission is to design and implement a Martian Underground Road Intersection Control System which will ensure safe and efficient transport of people and vehicles around our first settlement on Mars. A road network will be built underground due to the safety reasons; primarily for protection against radiation, reduced air pressure and inhospitable atmosphere.

SPECIFICATIONS

A Martian Underground Road Intersection Controller is to be designed, and implement it in Verilog.

System Specifications:

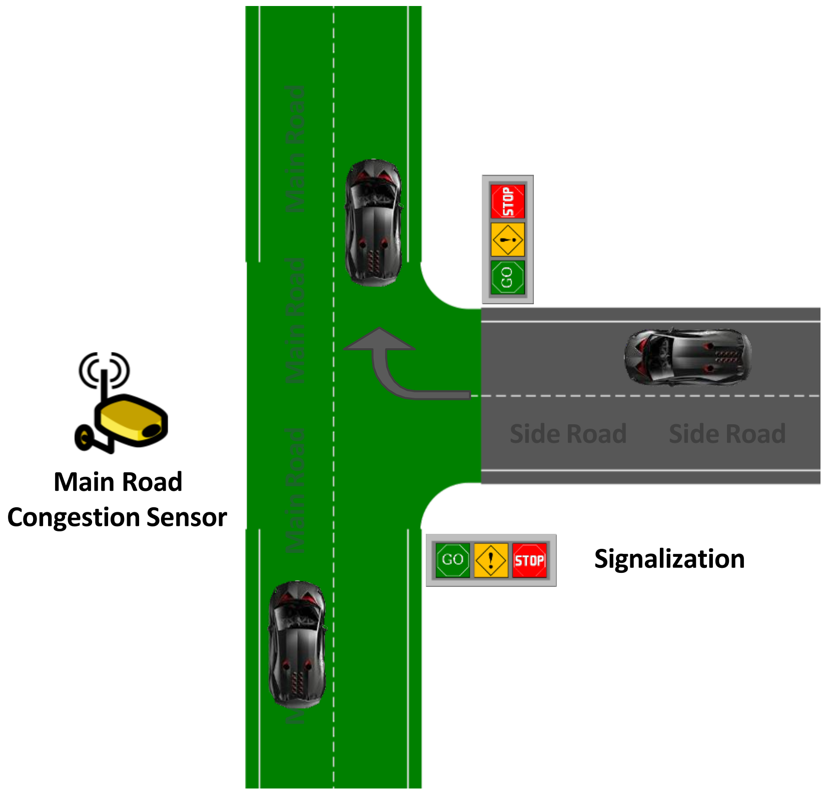
The first Martian underground road network will consist of two types of roads:

- **Main Roads** (**green** roads whose traffic gets high priority), and
- **Side Roads** (**grey** roads with lower priority traffic).

An intersection of a main road with a side road will therefore need to be controlled carefully with appropriate signalization. The tasks of the Martian Underground Road Intersection Controller will be the following:

- Ensure efficient and safe flow of vehicles in all directions,
- Take the road priorities into consideration, so that the main roads get higher throughput,
- Monitor and handle possible traffic congestion of the main roads.

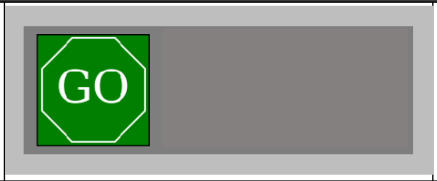
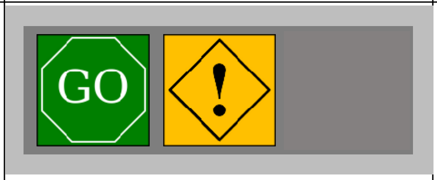
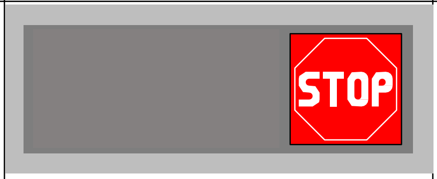
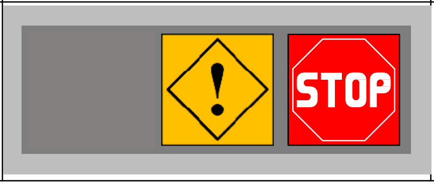
An intersection will look as follows:



The main road will have congestion sensors which will monitor the traffic flow and signal the controller in case of a traffic jam.

State Specifications for Signalization

Road Signalization on any road can be in one of the following states:

State	Signalization	Meaning
GO		Vehicles are free to drive forward.
GO-ATTENTION		Get ready to stop in less than 3 seconds.
STOP		All vehicles must stop.
STOP-ATTENTION		Get ready to start driving forward in less than 3 seconds.

At an intersection, there will be two signalization signboards, one for the main road, and another for the side road. The controller must control both signboards at the same time.

Signalization Rules:

- While one road is in the GO state, the other road can only be in the STOP state.
- After the GO state, a road must go through the GO-ATTENTION state before switching to the STOP state.
- Similarly, after the STOP state, a road must go through STOP-ATTENTION state before switching to the GO state.
- While one road is in the GO-ATTENTION state, the other road can only be in the STOP-ATTENTION state.

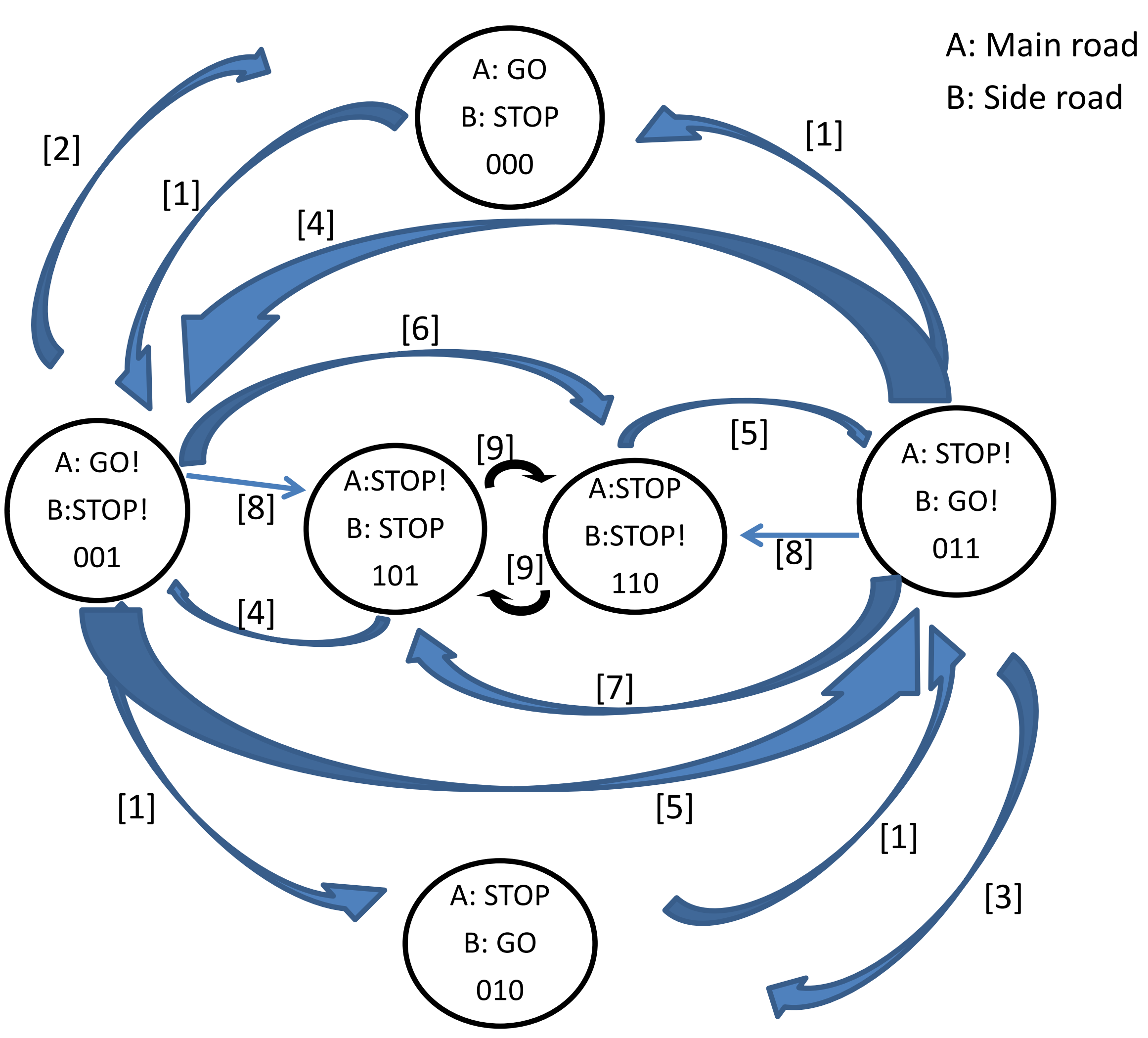
Timing Specifications for Signalization

Case 1 – No Congestion: The main road signalization stays in GO state for **exactly 20 seconds**. The side road signalization stays in GO state **for exactly 10 seconds**. GO-ATTENTION and STOP-ATTENTION states last **for 3 seconds** on all roads.

Case 2 – Congestion is Detected: The main road signalization stays in GO state for **at least 40 seconds and until the congestion sensor stops detecting traffic jam**. The side road signalization stays in GO state for **at most 10 seconds**, **but may be interrupted earlier in case congestion on the main road is detected**. GO-ATTENTION and STOP-ATTENTION states last **for 3 seconds** on all roads.

DESIGN

As the first step of designing an efficient trafic control flowing system, all possible states and transitions (logical or not) (according to signalization rules) are shown below. Logical reasonings and flaws are annotated.



[1]: These paths are efficient.

[2]: A loses acceleration; confusing to all drivers.

[3]: B loses acceleration; confusing to all drivers.

[4]: illogical since A should be prior.

[5]: illogical because B is already stopping. Confuses B, since they just came from the Earth afterall.

[6]: B is wasting A’s time. B is not even going.

[7]: A could have been better going (waste of time).

[8]: Confusing: conflicting with drivers’ predictions.

[9]: Waste of time.

State Diagram

After illogical transitions and states were removed, and timing specifications are taken into consideration, the solution obtained is below ; with binary encoded states.

A: Main road

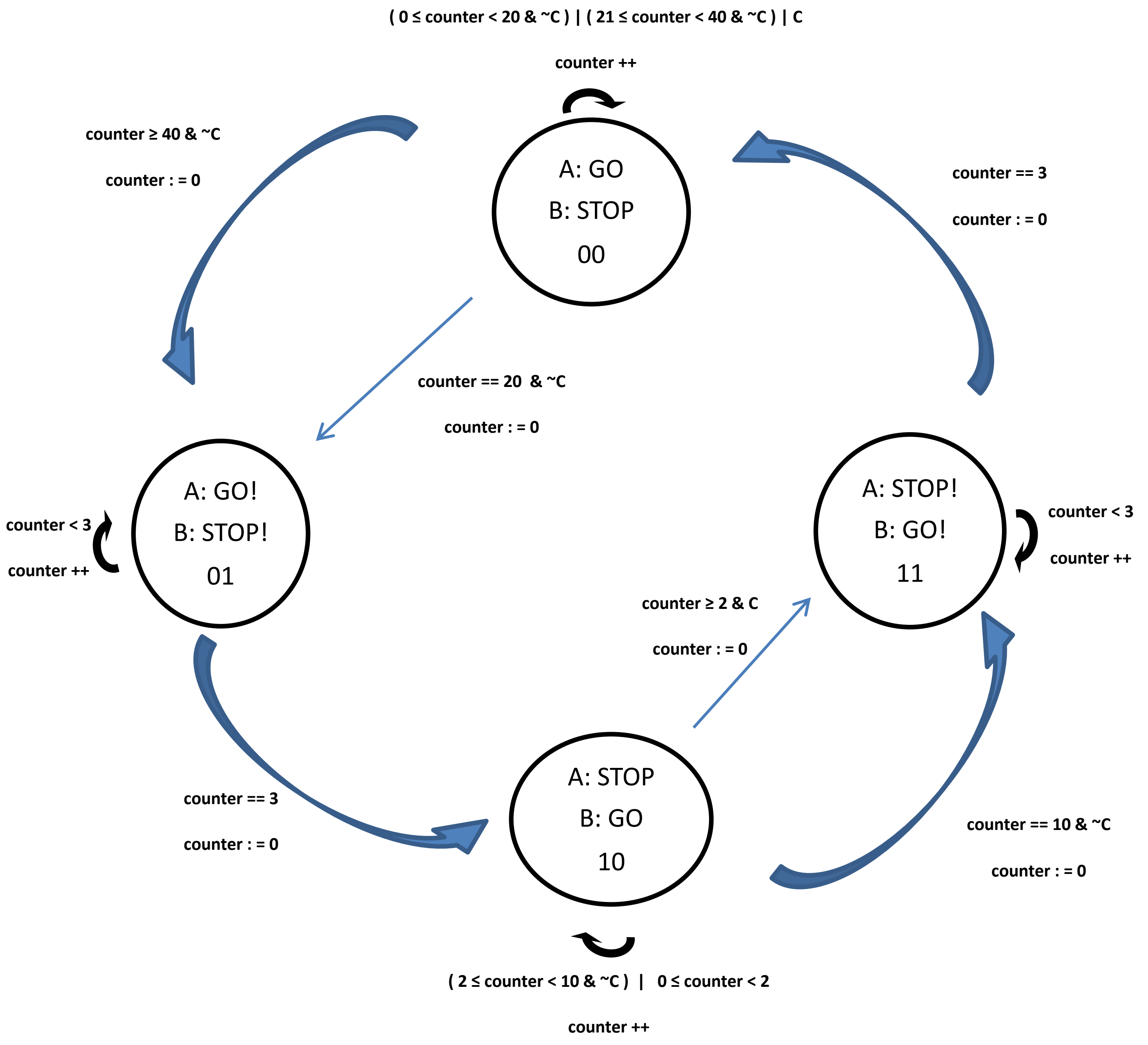
Counter is a variable, neither an input nor an output.

B: Side road

Unit of counter is seconds.

C : Congestion (Input)

State Table for The Underground Road Controller as a Time Embedded Moore FSM:



State Table For The Underground Road Controller As A Time Embedded Moore FSM

Present State (Binary Encoding)		Transition Condition	Transition Action	Next State		Output (Encoding: Being 1 lights up an individual sign.)					
						GO	!	STOP	GO	!	STOP
F	G			F(t+1)	G(t+1)	XA	YA	ZA	XB	YB	ZB
0	0	(counter ≥ 40 OR counter = 20) AND (no congestion detected)	Assign counter to zero	0	1	1	0	0	0	0	1
0	1	Counter = 3	Assign counter to zero	1	0	1	1	0	0	1	1
1	0	(counter ≥ 2 AND congestion detected) OR (counter = 10 AND no congestion detected)	Assign counter to zero	1	1	0	0	1	1	0	0
1	1	Counter = 3	Assign counter to zero	0	0	0	1	1	1	1	0
0	0	(Congestion detected) OR no congestion detected while counter is in the interval [0,20)U[21,40)	Increment counter by 1	0	0	1	0	0	0	0	1
0	1	0 ≤ counter < 3	Increment counter by 1	0	1	1	1	0	0	1	1
1	0	0 ≤ counter < 2 OR (2 ≤ counter < 10 AND no congestion detected)	Increment counter by 1	1	0	0	0	1	1	0	0
1	1	0 ≤ counter < 3	Increment counter by 1	1	1	0	1	1	1	1	0

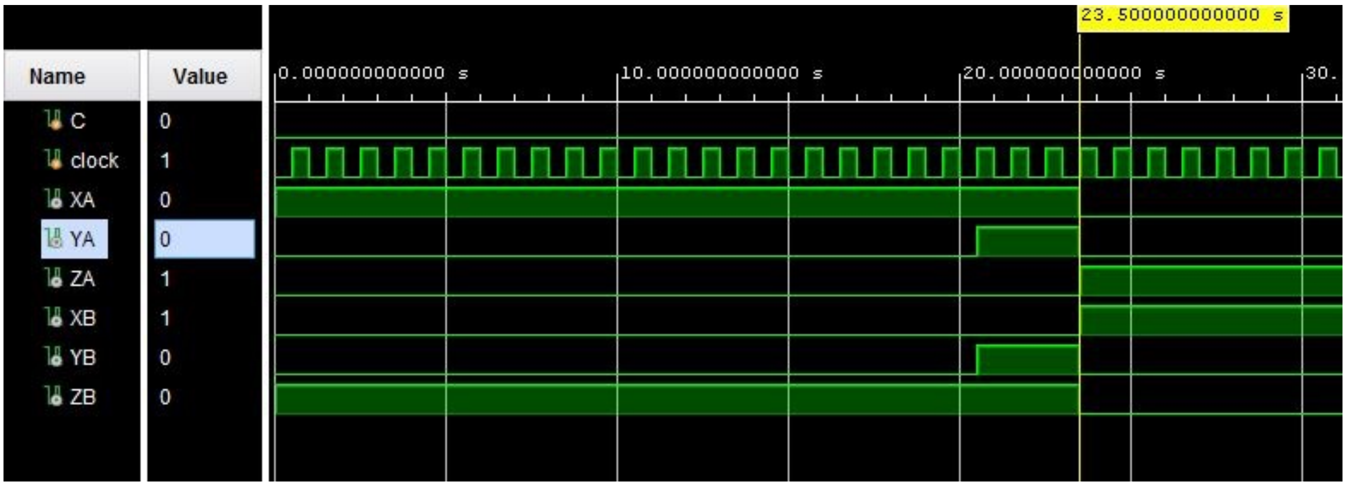
Verilog Code With Behavioral Design Approach

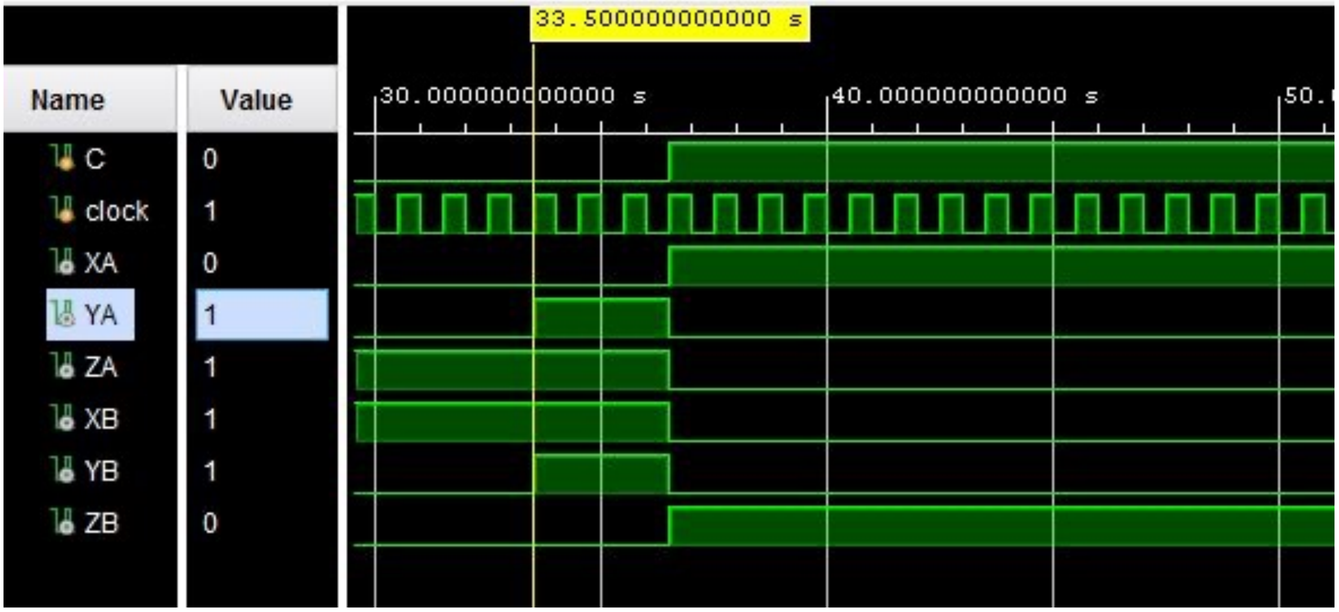
```verilog
`timescale 1ms / 1ms
                // scale is 1 milisecond because " 1s / 1s" (second) owerflows somehow and turns itself to picosecond.

module controller(C, clock, XA, YA, ZA, XB, YB, ZB );
input C, clock;
output XA, YA, ZA, XB, YB, ZB;

reg[1:0] state;
reg [8:0] counter;
parameter s0=2'b00, s1=2'b01, s2=2'b10, s3=2'b11;

initial begin
   state=s0;
   counter=0;
   end

/* "case" statement in always@(posedge clock) block with non-blocking assignments are used because counter should be updated
only once a second, rather than a combinational always@(*) block with blocking assignments.  */

always@ (posedge clock)
   begin
     case(state)
       s0: if((counter >= 40 || counter == 20)&& C==0)
            begin    state<=s1;
            counter<=1;   end           // not "0" because state changes already takes one second (1 count) in this non-blocking syntax
          else begin   state<=s0;
            counter<=counter+1;   end
       s1: if(counter==3)
            begin   state<=s2;
            counter<=1;   end
          else begin   state<=s1;
            counter<=counter+1;   end
       s2: if((counter>=2 && C==1)||(counter==10 &&C==0))
            begin   state<=s3;
            counter<=1;   end
          else begin   state<=s2;
            counter<=counter+1;   end
       s3: if(counter==3)
            begin   state<=s0;
            counter<=1;   end
          else begin   state<=s3;
            counter<=counter+1;   end

      endcase
   end

assign XA=~state[1];
assign YA= state[0];
assign ZA= state[1];
assign XB= state[1];
assign YB= state[0];
assign ZB=~state[1];

endmodule
```

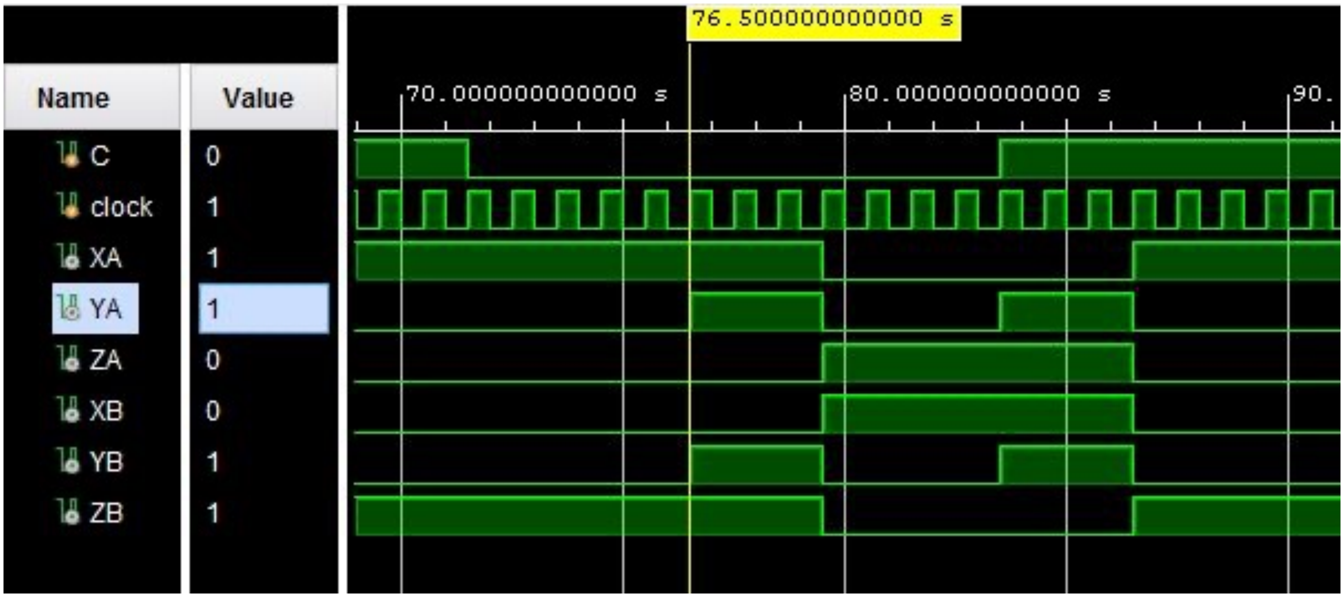
Testbench for
The Underground Road Controller

```
`timescale 1ms / 1ms
 // scale is 1 milisecond because " 1s / 1ms" owerflows somehow and turns itself to  picosecond.
 /* Precision here is 1ms; however, waveform shows the digits
                                                    with picosecond precision. Vivado has a contradiction. */

module controller_tb( );
 reg C, clock;
 wire XA, YA, ZA, XB, YB, ZB;

controller uut(C, clock, XA, YA, ZA, XB, YB, ZB );

initial begin
      clock=0;
      forever begin
         #500 clock=~clock; //period is 1 second, counter counts seconds
      end
    end


initial begin
       C=0;
       #36500;   /* No Congestion; 0.5 seconds from start to first rising clockedge,
       20 seconds   in s0,  3 seconds in s1, 10 seconds in s2, 3 seconds in s3    */

       C=1;
       #35000;   // Congestion; 35 seconds in s0

       C=0;
       #12000;   // No congestion; 5 seconds in s0 (rest of the 40), 3 seconds in s1, 4 seconds   in s2

       C=1;
       #63000;   // Prolonged Congestion; 3 seconds in s3, 60 seconds in s0

       C=0;
       #4000;   // No congestion: 3 seconds in s1, 1 second in s2, ... All transitions are tested.
       $finish;
       end
endmodule
```

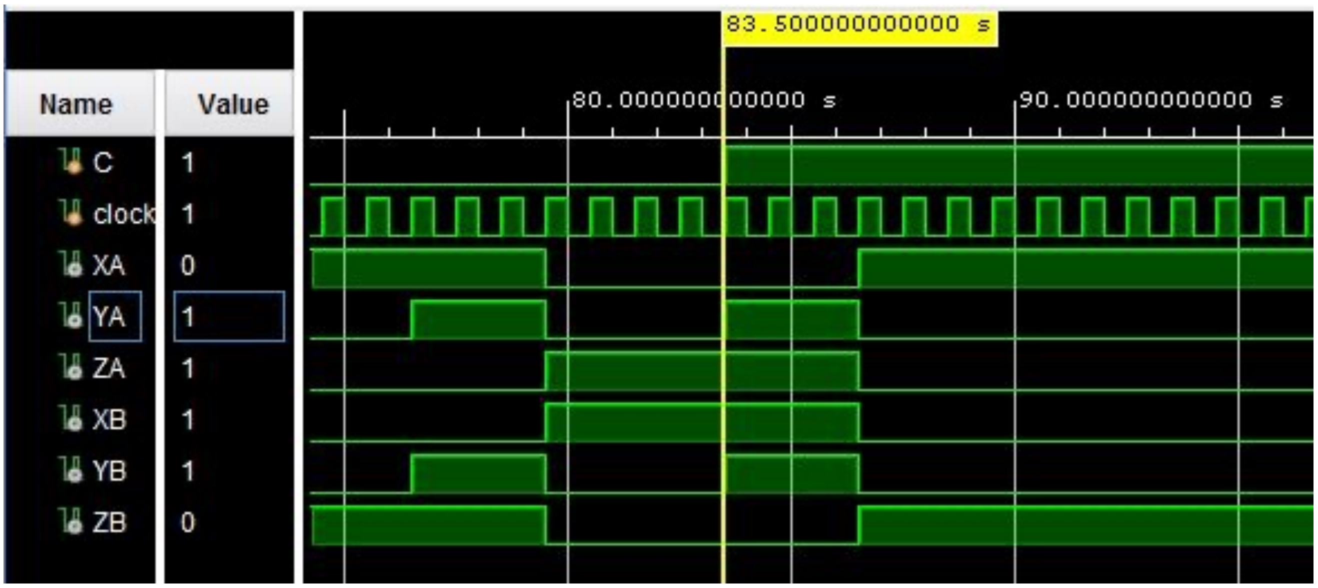
Simulation

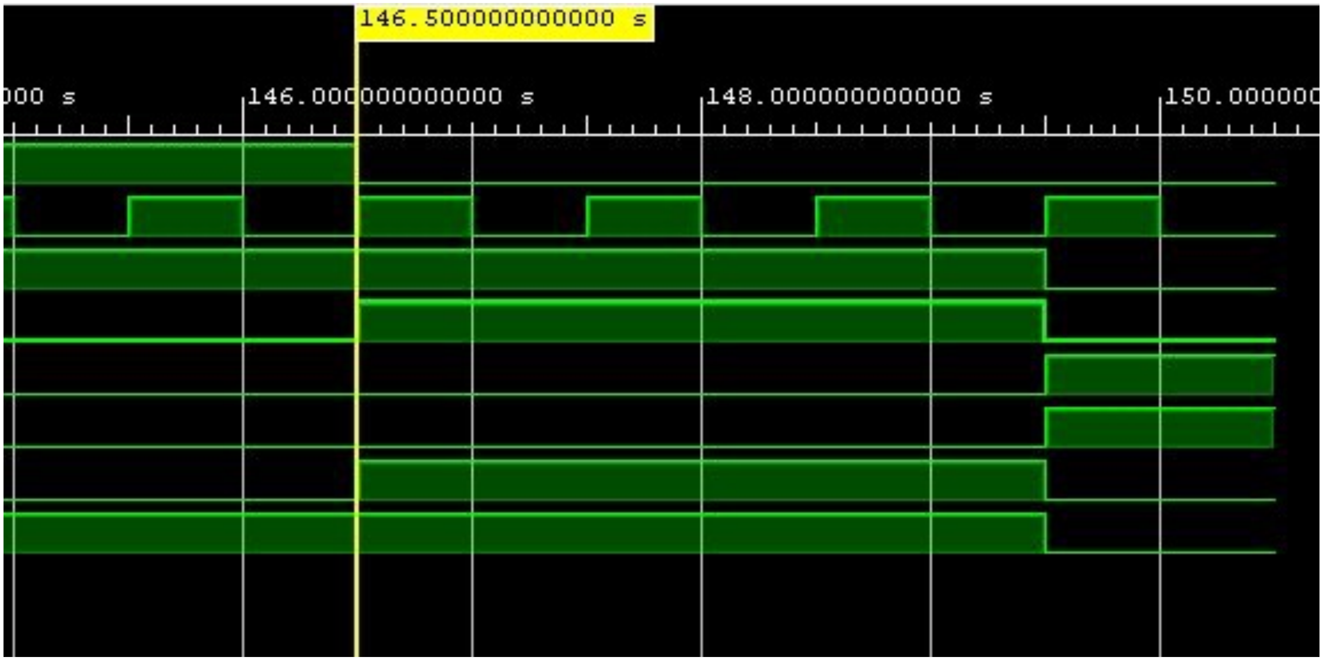

23.500000000000 s
Name
Value
0.000000000000 s
10.000000000000 s
20.000000000000 s
30.
C 0
clock 1
XA 0
YA 0
ZA 1
XB 1
YB 0
ZB 0


33.500000000000 s
Name
Value
30.000000000000 s
40.000000000000 s
50.
C 0
clock 1
XA 0
YA 1
ZA 1
XB 1
YB 1
ZB 0


76.500000000000 s
Name
Value
70.000000000000 s
80.000000000000 s
90.
C 0
clock 1
XA 1
YA 1
ZA 0
XB 0
YB 1
ZB 1

Simulation





Discussion

Simulation results are as expected. This Controller is what Martian Civilization needed for a safe and sound traffic flow.