EEE339 Digital System Design with HDL

Assignment 1: Traffic Light Control

Student Name: Yi.Wang1902
Student ID: 1929718
Submission Date: Nov. 11th 2022

Contents

1 Task Description

There are four sets of traffic light required in the cross shown in Figure 1. Light 1 and 3 in direction A are controlled by the same signal. Light 2 and 4 in direction B are controlled by the same signal. There are four lights in each light set: turn left, red, yellow, and green. By default, the countdown time of each light is: t1 = 17s for green light, t2 = 6s for yellow light, t3 = 10s for turn left light. When the green light counts down to 5, the light will flash every 1 second. And the yellow light will always be flashing when turned on.

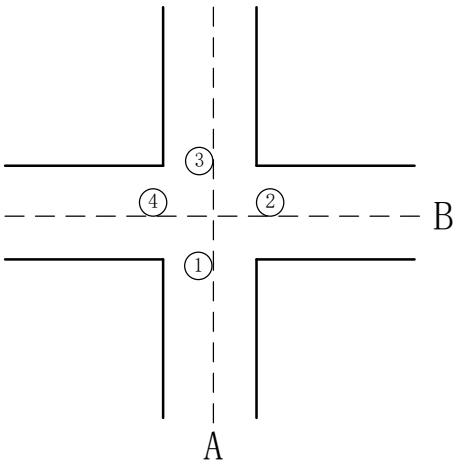


Figure 1: A Crossroad in this design

There are three modes in this system: manual, auto, and load. These modes are controlled by three toggle switches. The ‘load mode’ functions in parallel with the ‘auto mode’. That is, when the system is in ‘auto mode’, a number can be loaded at the same time.

1.1 Manual Mode

In manual mode, the traffic light can be manually controlled by the police. There are four states in this mode. When M2M1M0 is ‘000’, the light on path A is constant green and the light on path B is constant red. When M2M1M0 is ‘001’, the light on path A is constant turn left and the light on path B is constant red. When M2M1M0 is ‘010’, the light on path A is constant red and the light on path B is constant green. When M2M1M0 is ‘011’, the light on path A is constant red and the light on path B is constant turn left. In this mode, the number on seven segment display is always 00.

The corresponding input control signal and output LED lights are also shown in Table 1.

Control Mode				
Mode	M2	M1	M0	Description
Manul	0	0	0	Path A Green; Path B Red
	0	0	1	Path A Left; Path B Red
	0	1	0	Path A Red; Path B Green
	0	1	1	Path A Red; Path B Left
Auto	1	0	0	Tansists from S1 to S8
Load	1	0	1	Load t1: Green light time
	1	1	0	Load t2: Yellow light time
	1	1	1	Load t3: turn left light time

Table 1: Control signals and related modes

1.2 Auto Mode

In auto mode, the lights on road A and B are turned on and off automatically. The transition table is shown in Table 2.

	M2M1M0	State	A	B	time
Manul	000		Green	Red	~
	001		Left	Red	~
	010		Red	Green	~
	011		Red	Left	~
Auto	100	S1	Green	Red	t1
		S2	Yellow	Red	t2
		S3	Left	Red	t3
		S4	Yellow	Red	t2
		S5	Red	Green	t1
		S6	Red	Yellow	t2
		S7	Red	Left	t3
		S8	Red	Yellow	t2

Table 2: Transition table in manual and auto mode

In the states S1 to S4, the LEDs on path A goes from Green to Yellow to Left and finally to Yellow, and the LEDs on path B is always Red in these four states. In the states S5 to S8, the LEDs on path B goes from Green to Yellow to Left and finally to Yellow, and the LEDs on path A is always Red in these four states. The corresponding default time durations are: t1 = 17s, t2 = 6s, t3 = 10s. When the green light counts down to 5, the LED will flash every 1 second. And the yellow light will always be flashing when turned on. The seven-segment display will also show the countdown number in auto mode. For example, to count the green light time t1 = 17s, the last two digits of the seven-segment display counts from 16 to 00, and the number changes every 1 second.

1.3 Load Mode

In load mode, five bits binary number can be loaded into countdown time t1 or t2 or t3. And the binary number to be loaded is controlled by five toggle switches: SW7 to SW3. The related input control signal and the countdown time are shown in Table 1. When M2M1M0 is ‘101’, t1 could be changed. When M2M1M0 is ‘110’, t2 could be changed. When M2M1M0 is ‘111’, t3 could be changed. After the numbers are loaded, the traffic light will always function according to these

loaded times until the ‘preset’ switch is turned on.

1.4 Clock

In this system, a 27MHz internal clock signal is used. This signal will be transformed into a 1Hz signal by a module called ‘timeCounter’. The leftmost LEDR9 also flashes every one second to indicate the 1Hz clock.

2 User interface

The DE1 board and related functions of LEDs and switches are shown in Figure 2.

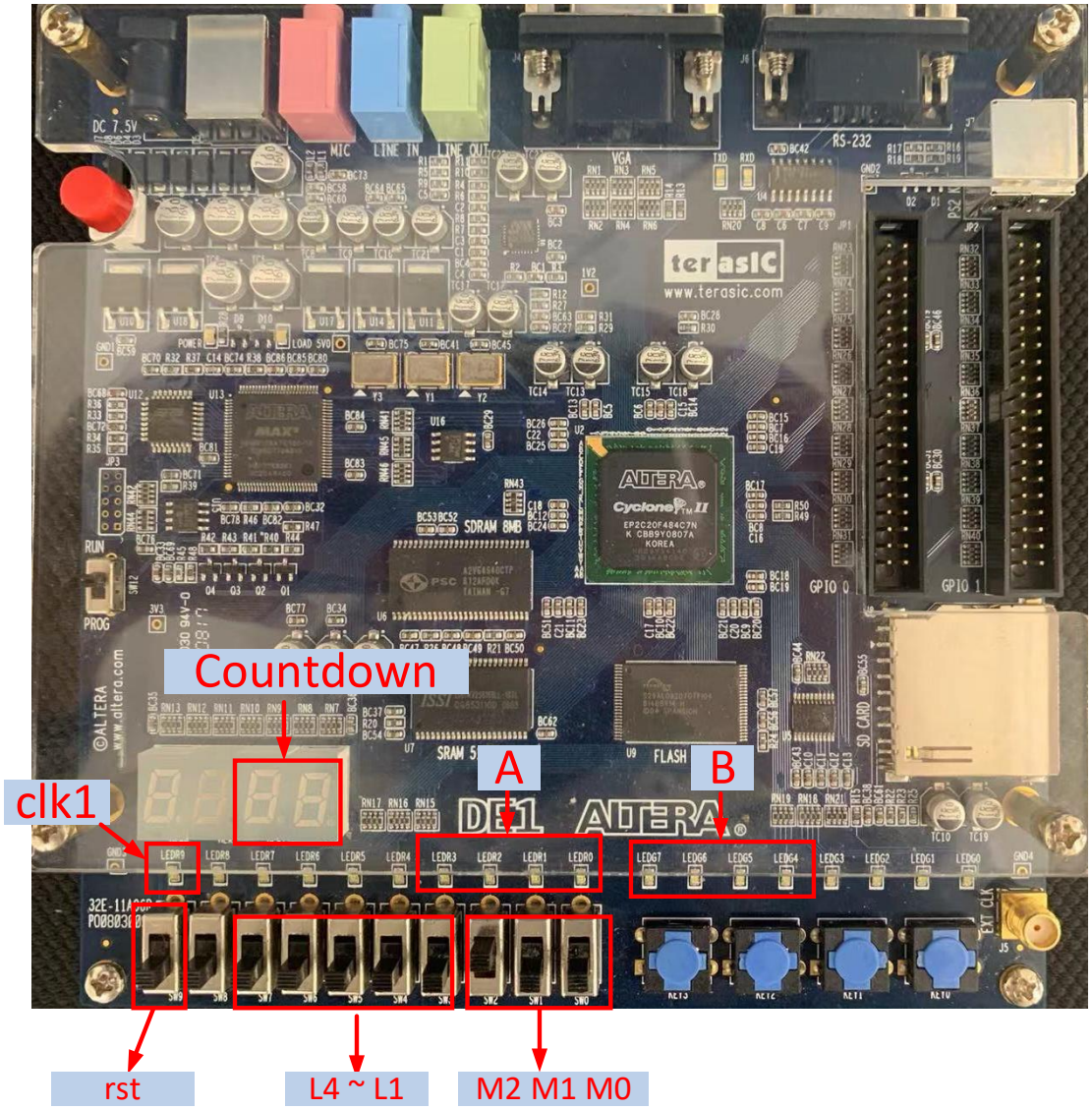


Figure 2: The DE1 board and related functions

2.1 Functions of toggle switched

Toggle Switch	name	function
SW[9]	rst	Reset the system to state S1 and the defalt time
SW[7]	L4	Input the binary represnetation of the loaded time
SW[6]	L3	
SW[5]	L2	
SW[4]	L1	
SW[3]	L0	
SW[2]	M2	Select Control Mode
SW[1]	M1	
SW[0]	M0	

SW[9] is the reset switch. When SW[9] is on, the state of auto mode returns to S1, and all the countdown time t1, t2, t3 returns to default.

SW[7] to SW[3] represents loaded binary number. For example, set SW[7] to SW[3] to '11001', and after M2M1M0 set the board to load mode, '25' will be loaded into the corresponding countdown time.

SW[2], SW[1] and SW[0] represents the input mode control signal M2M1M0. Through toggling these three switches, the system can transit between 'Manual Mode', 'Auto Mode', and 'Load Mode'.

2.2 Functions of LEDs

LEDs	name	function	
LEDR[3]	led[7]	A Left	Control the traffic light in direction A
LEDR[2]	led[6]	A Red	
LEDR[1]	led[4]	A Yellow	
LEDR[0]	led[5]	A Green	
LEDG[7]	led[3]	B Left	Control the traffic light in direction B
LEDG[6]	led[2]	B Red	
LEDG[5]	led[0]	B Yellow	
LEDG[4]	led[1]	B Green	
LEDR[9]	clk1	indicate the 1Hz clock	

LEDR[3] to LEDR[0] controls the traffic light in direction A; and LEDG[7] to LEDG[4] controls the traffic light in direction B.

LEDR[9] indicate the 1HZ clock. It flashes every 1 second.

2.3 Functions of 7Segment Display

7Segment Display	name	function	7Segment Display	name	function
HEX0[0]	Ones	Control the ones digit of the 7 Segment Display on path A	HEX2[0]	Ones_add	Control the ones digit of the 7 Segment Display on path B
HEX0[1]			HEX2[1]		
HEX0[1]			HEX2[1]		
HEX0[2]			HEX2[2]		
HEX0[3]			HEX2[3]		
HEX0[4]			HEX2[4]		
HEX0[5]			HEX2[5]		
HEX0[6]			HEX2[6]		
HEX1[0]	Tens	Control the tens digit of the 7 Segment Display on path A	HEX3[0]	Tens_add	Control the tens digit of the 7 Segment Display on path B
HEX1[1]			HEX3[1]		
HEX1[2]			HEX3[2]		
HEX1[3]			HEX3[3]		
HEX1[4]			HEX3[4]		
HEX1[5]			HEX3[5]		
HEX1[6]			HEX3[6]		

In auto mode, HEX0 and HEX1 can display the remaining countdown time in path A. HEX2 and HEX3 can display the remaining countdown time in path B. In manual mode, the 7Segment Display will all display 0.

HEX0 and HEX2 represents the tens digit of the countdown time. HEX1 and HEX3 represents the tens digit of the countdown time

3 ASM Chart

The ASM chart in auto mode is shown in Figure 3.

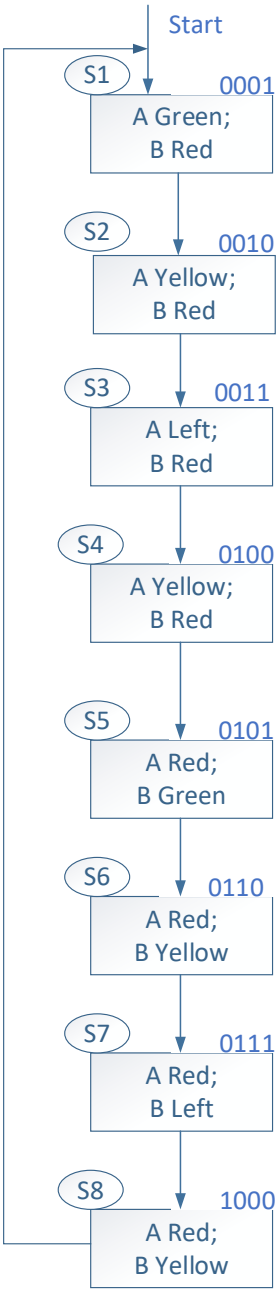
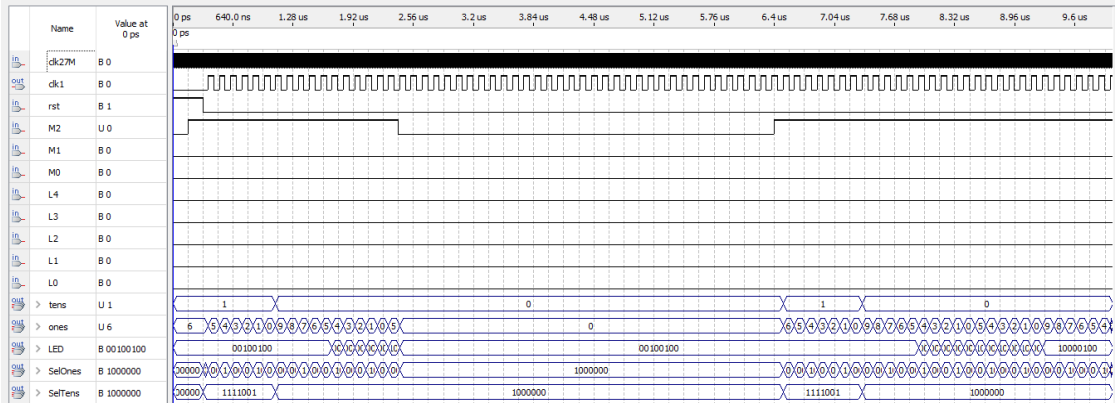
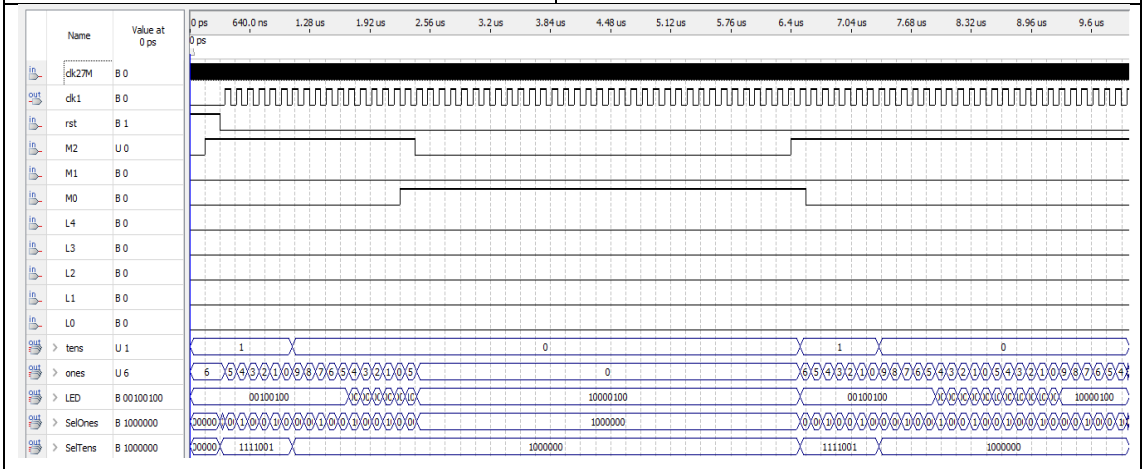


Figure 3: ASM chart in auto mode

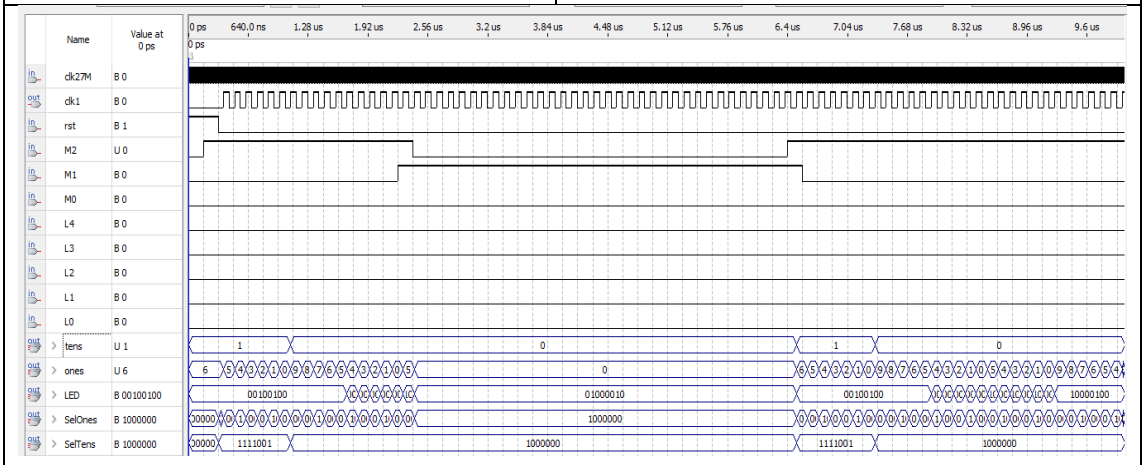
4 Simulation Result

To better simulate the traffic light system, the ‘timeCounter’ module and the input 27MHz clock signals are adjusted so that the period of clk1 is 120ns. ‘rst’ is the reset signal. ‘M2M1M0 are the mode control signals. ‘L4 L3 L2 L1 L0’ represents the number to be loaded into the system. ‘tens’ and ‘ones’ represent the corresponding digits of t1, t2, and t3. LED represents the eight traffic lights: A Left, A Red, A Green, A Yellow, B Left, B Red, B Green, B Yellow. ‘SelOnes’ controls the ones digit on the 7Segment Display, and ‘SelTens’ controls the tens digit on the 7Segment Display.

Mode: Manual	M2M1M0: 000
	
Description: When M2M1M0 turns to 000, the countdown ‘tens’ and ‘ones’ are both 0. LED is ‘00100100’, indicating A Green and B Red are on.	

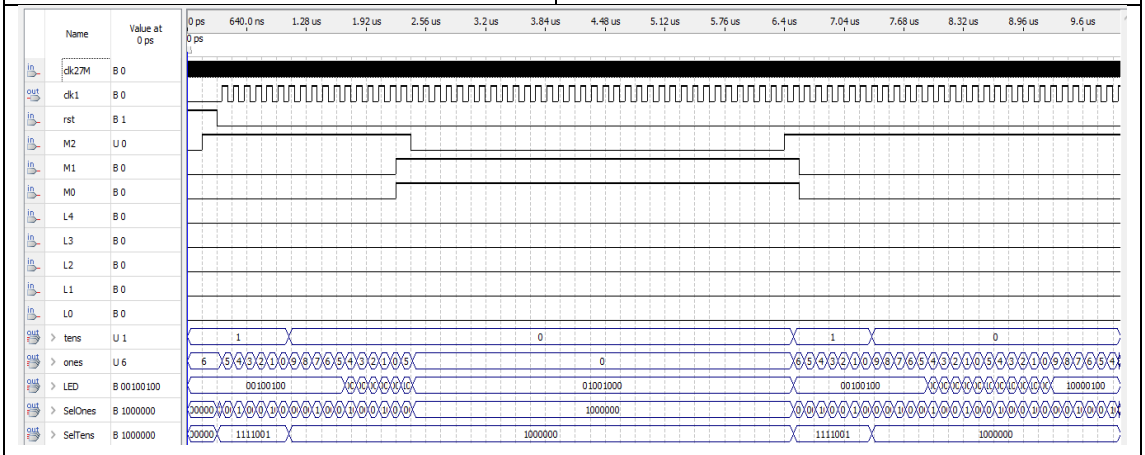
Mode: Manual	M2M1M0: 001
	
Description: When M2M1M0 turns to 001, the countdown ‘tens’ and ‘ones’ are both 0. LED is ‘10000100’, indicating A Left and B Red are on.	

Mode: Manual	M2M1M0: 010



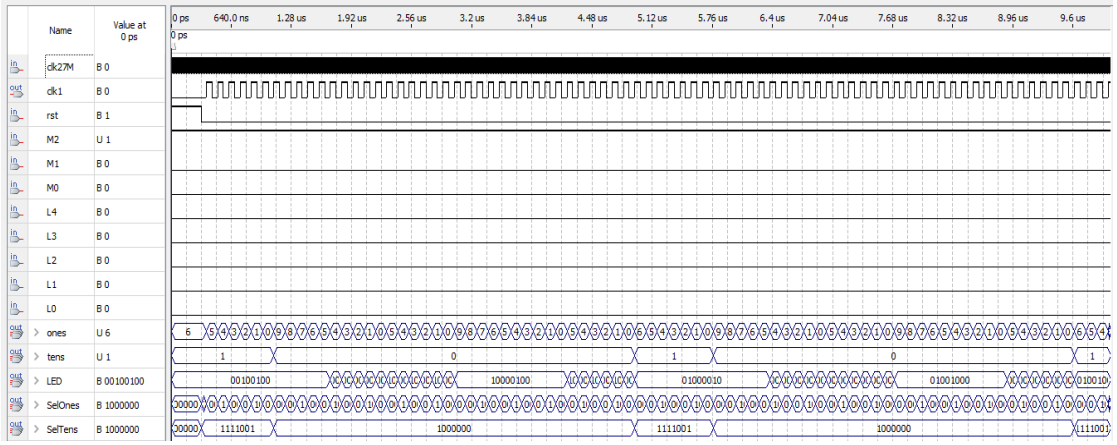
Description: When M2M1M0 turns to 010, the countdown ‘tens’ and ‘ones’ are both 0. LED is ‘01000010’, indicating A Red and B Green are on.

Mode: Manual	M2M1M0: 011



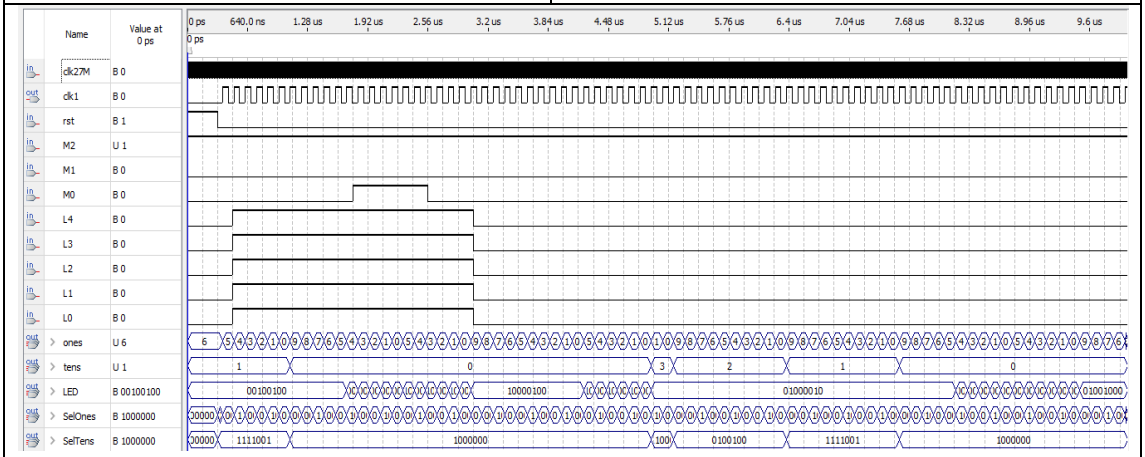
Description: When M2M1M0 turns to 000, the countdown ‘tens’ and ‘ones’ are both 0. LED is ‘01001000’, indicating A Green and B Left are on.

Mode: Auto	M2M1M0: 100



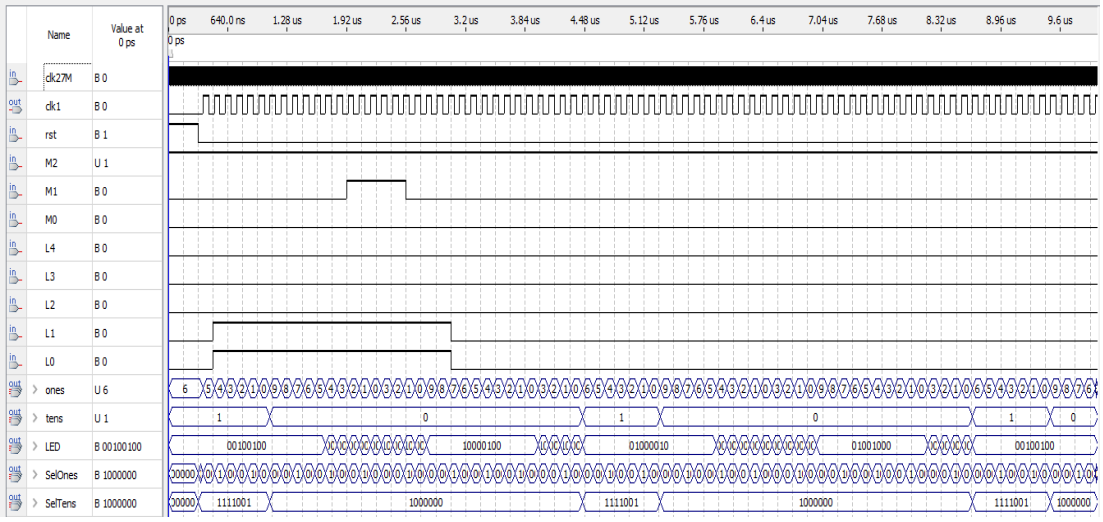
Description: The traffic light system transits from S1 to S8, and then back to S1. When the green light counts down to 5, the LED flashes every 1 second. And the yellow LED is always flashing when turned on.

Mode: Load t1	M2M1M0: 101



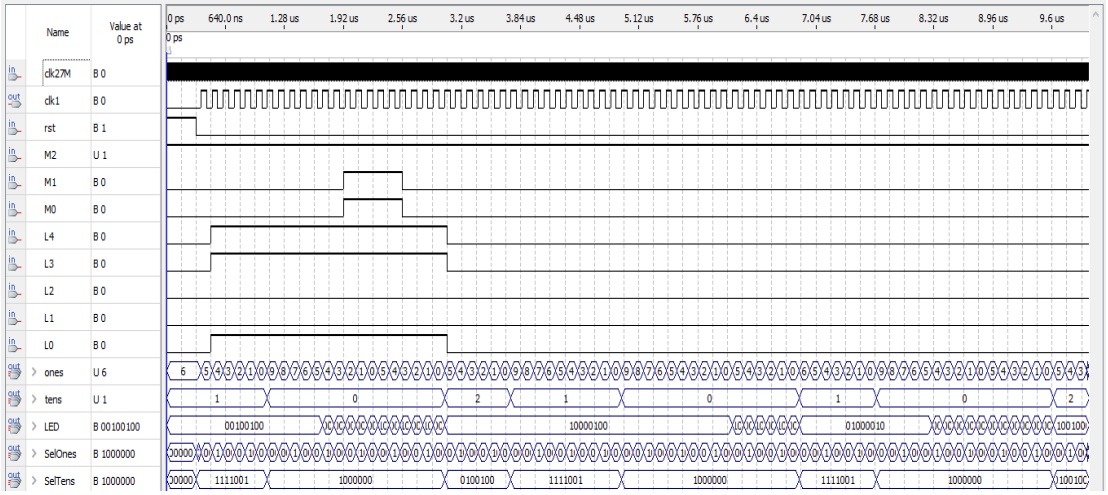
Description: Load 31 to t1. After M2M1M0 returns back to auto mode, the red-light time t1 starts the countdown from 31.

Mode: Load t2	M2M1M0: 110



Description: Load 3 to t1. After M2M1M0 returns back to auto mode, the yellow-light time t2 starts the countdown from 3.

Mode: Load t3	M2M1M0: 111



Description: Load 25 to t3. After M2M1M0 returns back to auto mode, the green-light time t3 starts the countdown from 25.

5 Diagram of the synthesized circuit

A diagram of the synthesized circuit is shown in Figure 4 and Figure 5.

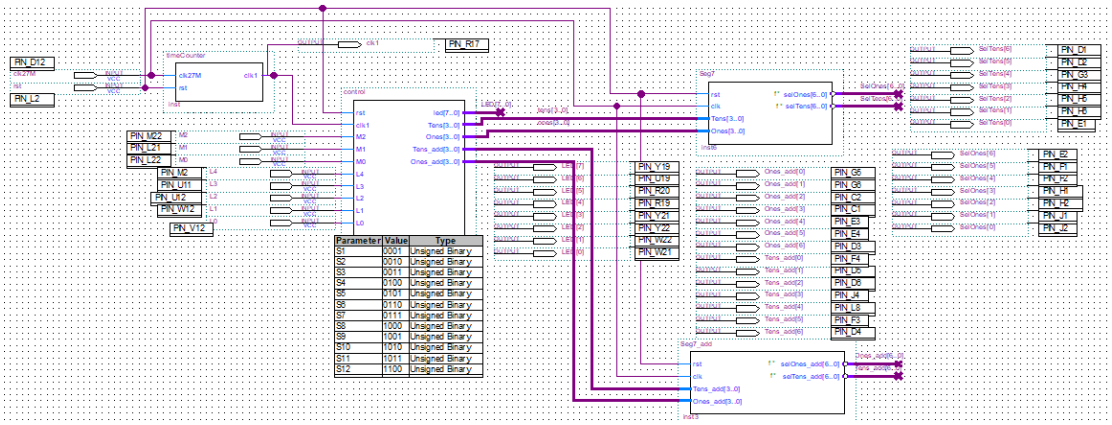


Figure 4: Complete synthesized circuit

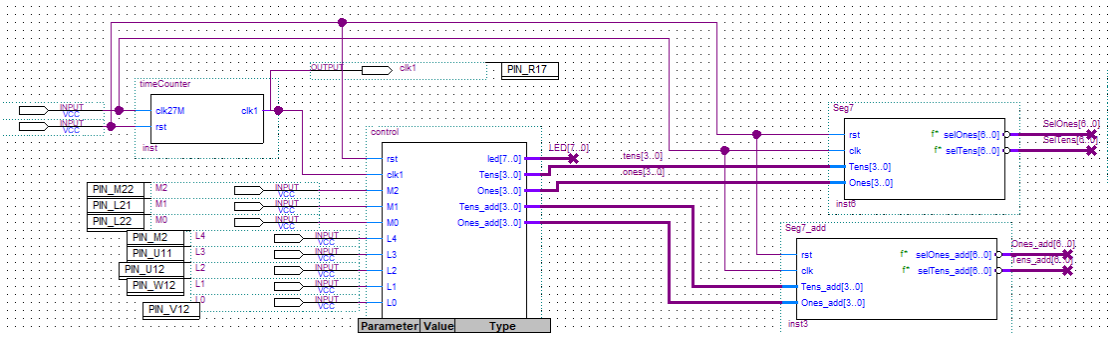


Figure 5: Enlarged view of the main modules

Module ‘timeCounter’ transform the input internal 27MHz clock signal into 1Hz clock signal.

Module ‘control’ is responsible for countdown operation. It can also process the mode control command and the LEDs. It has 10 inputs and 3 output buses. ‘rst’ and ‘clk1’ are reset and clock inputs. M2M1M0 are mode control signals. L4 to L0 are the input loaded numbers. ‘led[7..0]’ controls the 8 LED traffic lights. ‘Tens[3..0]’ and ‘Ones[3..0]’ controls the countdown numbers that should be displayed on 7Segment Display.

Module ‘Seg7’ control the HEX0 and HEX1 on the 7Segment Display. It can read the input 2 digits number from the ‘control’ module. And then display these 2 digits number on the 7Segment Display. Output ‘selOnes[6…0]’ controls the ones digit, and the output ‘selTens[6…0]’ controls the tens digit. Module ‘Seg7_add’ the HEX2 and HEX3 on the 7Segment Display, and it has the same function as the Module ‘Seg7’.

All four modules have an asynchronous reset input ‘rst’.

Appendix A: Related pin assignments

Toggle switches

Signal Name	FPGA Pin No.	Description
SW[0]	PIN_L22	Toggle Switch[0]
SW[1]	PIN_L21	Toggle Switch[1]
SW[2]	PIN_M22	Toggle Switch[2]
SW[3]	PIN_V12	Toggle Switch[3]
SW[4]	PIN_W12	Toggle Switch[4]
SW[5]	PIN_U12	Toggle Switch[5]
SW[6]	PIN_U11	Toggle Switch[6]
SW[7]	PIN_M2	Toggle Switch[7]
SW[8]	PIN_M1	Toggle Switch[8]
SW[9]	PIN_L2	Toggle Switch[9]

LEDs

Signal Name	FPGA Pin No.	Description
LEDR[0]	PIN_R20	LED Red[0]
LEDR[1]	PIN_R19	LED Red[1]
LEDR[2]	PIN_U19	LED Red[2]
LEDR[3]	PIN_Y19	LED Red[3]
LEDR[4]	PIN_T18	LED Red[4]
LEDR[5]	PIN_V19	LED Red[5]
LEDR[6]	PIN_Y18	LED Red[6]
LEDR[7]	PIN_U18	LED Red[7]
LEDR[8]	PIN_R18	LED Red[8]
LEDR[9]	PIN_R17	LED Red[9]
LEDG[0]	PIN_U22	LED Green[0]
LEDG[1]	PIN_U21	LED Green[1]
LEDG[2]	PIN_V22	LED Green[2]
LEDG[3]	PIN_V21	LED Green[3]
LEDG[4]	PIN_W22	LED Green[4]
LEDG[5]	PIN_W21	LED Green[5]
LEDG[6]	PIN_Y22	LED Green[6]
LEDG[7]	PIN_Y21	LED Green[7]

7Segment Display

Signal Name	FPGA Pin No.	Description
HEX0[0]	PIN_J2	Seven Segment Digit 0[0]
HEX0[1]	PIN_J1	Seven Segment Digit 0[1]
HEX0[2]	PIN_H2	Seven Segment Digit 0[2]
HEX0[3]	PIN_H1	Seven Segment Digit 0[3]
HEX0[4]	PIN_F2	Seven Segment Digit 0[4]
HEX0[5]	PIN_F1	Seven Segment Digit 0[5]
HEX0[6]	PIN_E2	Seven Segment Digit 0[6]
HEX1[0]	PIN_E1	Seven Segment Digit 1[0]
HEX1[1]	PIN_H6	Seven Segment Digit 1[1]
HEX1[2]	PIN_H5	Seven Segment Digit 1[2]
HEX1[3]	PIN_H4	Seven Segment Digit 1[3]
HEX1[4]	PIN_G3	Seven Segment Digit 1[4]
HEX1[5]	PIN_D2	Seven Segment Digit 1[5]
HEX1[6]	PIN_D1	Seven Segment Digit 1[6]
HEX2[0]	PIN_G5	Seven Segment Digit 2[0]
HEX2[1]	PIN_G6	Seven Segment Digit 2[1]
HEX2[2]	PIN_C2	Seven Segment Digit 2[2]
HEX2[3]	PIN_C1	Seven Segment Digit 2[3]
HEX2[4]	PIN_E3	Seven Segment Digit 2[4]
HEX2[5]	PIN_E4	Seven Segment Digit 2[5]
HEX2[6]	PIN_D3	Seven Segment Digit 2[6]
HEX3[0]	PIN_F4	Seven Segment Digit 3[0]
HEX3[1]	PIN_D5	Seven Segment Digit 3[1]
HEX3[2]	PIN_D6	Seven Segment Digit 3[2]
HEX3[3]	PIN_J4	Seven Segment Digit 3[3]
HEX3[4]	PIN_L8	Seven Segment Digit 3[4]
HEX3[5]	PIN_F3	Seven Segment Digit 3[5]
HEX3[6]	PIN_D4	Seven Segment Digit 3[6]

Clock input

Signal Name	FPGA Pin No.	Description
CLOCK_27	PIN_D12, PIN_E12	27 MHz clock input
CLOCK_50	PIN_L1	50 MHz clock input
CLOCK_24	PIN_A12, PIN_B12	24 MHz clock input from USB Blaster
EXT_CLOCK	PIN_M21	External (SMA) clock input

Appendix B: Verilog Code

```
module timeCounter (clk27M, rst, clk1);      //27M Hz clock frequency to 1Hz
      input       clk27M, rst;
      output      clk1;
      reg [24:0]j;
      reg clk1;
      always@ (posedge clk27M or posedge rst)
            if (rst)
                  begin
                  clk1 <= 0;
                  j <= 0;
                  end
            //else if (j == 13500000)
            else if (j == 5)
                  begin
                  j <= 0;
                  clk1 <= ~clk1;
                  end
            else
                  j <= j + 1;
endmodule

module control (rst, clk1,
                              M2, M1, M0,     //control mode
                              L4, L3, L2, L1, L0,       //load number
                              led,              //traffic light
                              Tens,             //ns seg tens
                              Ones,                   //ns seg ones
                              Tens_add,Ones_add
                              );
      input rst, clk1,  M2, M1, M0, L4, L3, L2, L1, L0;

      output [7:0] led;                 //"aL,aR,aG,aY; bL,bR,bG,bY"
      output [3:0] Tens, Ones;    //for 7 segment
      output [3:0] Tens_add, Ones_add;      //for 7 segment
      reg [7:0] led;
      reg [3:0] Tens, Ones;
      reg [3:0] Tens_add, Ones_add;
      wire [2:0]a;                        //control index
      wire [4:0]L;
      reg [3:0] state;
      parameter S1 = 4'b0001,
                        S2 = 4'b0010,
                        S3 = 4'b0011,
                        S4 = 4'b0100,
                        S5 = 4'b0101,
                        S6 = 4'b0110,
                        S7 = 4'b0111,
                        S8 = 4'b1000,
                        S9 = 4'b1001,
                        S10= 4'b1010,
                        S11= 4'b1011,
                        S12= 4'b1100;
      reg [3:0]t1_Tens = 4'b0001, t1_Ones = 4'b0110,          //Green light time
```

```
                t2_Tens = 4'b0000, t2_Ones = 4'b0101,          //Yellow light time
                t3_Tens = 4'b0000, t3_Ones = 4'b1001;          //turn left time
reg [3:0]t_Tens_add = 4'b0011, t_Ones_add = 4'b1000;
assign a = {M2, M1, M0};
assign L = {L4, L3, L2, L1, L0};

always @(posedge clk1 or posedge rst)
    if (rst)    //initialize
        begin
        t1_Tens = 4'b0001; t1_Ones = 4'b0110;          //Green light time
        t2_Tens = 4'b0000; t2_Ones = 4'b0101;          //Yellow light time
        t3_Tens = 4'b0000; t3_Ones = 4'b1001;
        end
    else if (a == 3'b111)  //load Turn Left time t3
        begin
        case (L)
        //5'b00000:begin t3_Tens = 0; t3_Ones = 0; end
        5'b00001:begin t3_Tens = 0; t3_Ones = 1; end
        5'b00010:begin t3_Tens = 0; t3_Ones = 2; end
        5'b00011:begin t3_Tens = 0; t3_Ones = 3; end
        5'b00100:begin t3_Tens = 0; t3_Ones = 4; end
        5'b00101:begin t3_Tens = 0; t3_Ones = 5; end
        5'b00110:begin t3_Tens = 0; t3_Ones = 6; end
        5'b00111:begin t3_Tens = 0; t3_Ones = 7; end
        5'b01000:begin t3_Tens = 0; t3_Ones = 8; end
        5'b01001:begin t3_Tens = 0; t3_Ones = 9; end
        5'b01010:begin t3_Tens = 1; t3_Ones = 0; end
        5'b01011:begin t3_Tens = 1; t3_Ones = 1; end
        5'b01100:begin t3_Tens = 1; t3_Ones = 2; end
        5'b01101:begin t3_Tens = 1; t3_Ones = 3; end
        5'b01110:begin t3_Tens = 1; t3_Ones = 4; end
        5'b01111:begin t3_Tens = 1; t3_Ones = 5; end

        5'b10000:begin t3_Tens = 1; t3_Ones = 6; end
        5'b10001:begin t3_Tens = 1; t3_Ones = 7; end
        5'b10010:begin t3_Tens = 1; t3_Ones = 8; end
        5'b10011:begin t3_Tens = 1; t3_Ones = 9; end
        5'b10100:begin t3_Tens = 2; t3_Ones = 0; end
        5'b10101:begin t3_Tens = 2; t3_Ones = 1; end
        5'b10110:begin t3_Tens = 2; t3_Ones = 2; end
        5'b10111:begin t3_Tens = 2; t3_Ones = 3; end
        5'b11000:begin t3_Tens = 2; t3_Ones = 4; end
        5'b11001:begin t3_Tens = 2; t3_Ones = 5; end
        5'b11010:begin t3_Tens = 2; t3_Ones = 6; end
        5'b11011:begin t3_Tens = 2; t3_Ones = 7; end
        5'b11100:begin t3_Tens = 2; t3_Ones = 8; end
        5'b11101:begin t3_Tens = 2; t3_Ones = 9; end
        5'b11110:begin t3_Tens = 3; t3_Ones = 0; end
        5'b11111:begin t3_Tens = 3; t3_Ones = 1; end
        default:begin t3_Tens = 0; t3_Ones = 9; end
        endcase
        end
    else if (a == 3'b110)  //load Yellow light time t2
        begin
        case (L)
        //5'b00000:begin t2_Tens = 0; t2_Ones = 0; end
        5'b00001:begin t2_Tens = 0; t2_Ones = 1; end
```

```
                5'b00010:begin t2_Tens = 0; t2_Ones = 2; end
                5'b00011:begin t2_Tens = 0; t2_Ones = 3; end
                5'b00100:begin t2_Tens = 0; t2_Ones = 4; end
                5'b00101:begin t2_Tens = 0; t2_Ones = 5; end
                5'b00110:begin t2_Tens = 0; t2_Ones = 6; end
                5'b00111:begin t2_Tens = 0; t2_Ones = 7; end
                5'b01000:begin t2_Tens = 0; t2_Ones = 8; end
                5'b01001:begin t2_Tens = 0; t2_Ones = 9; end
                5'b01010:begin t2_Tens = 1; t2_Ones = 0; end
                5'b01011:begin t2_Tens = 1; t2_Ones = 1; end
                5'b01100:begin t2_Tens = 1; t2_Ones = 2; end
                5'b01101:begin t2_Tens = 1; t2_Ones = 3; end
                5'b01110:begin t2_Tens = 1; t2_Ones = 4; end
                5'b01111:begin t2_Tens = 1; t2_Ones = 5; end

                5'b10000:begin t2_Tens = 1; t2_Ones = 6; end
                5'b10001:begin t2_Tens = 1; t2_Ones = 7; end
                5'b10010:begin t2_Tens = 1; t2_Ones = 8; end
                5'b10011:begin t2_Tens = 1; t2_Ones = 9; end
                5'b10100:begin t2_Tens = 2; t2_Ones = 0; end
                5'b10101:begin t2_Tens = 2; t2_Ones = 1; end
                5'b10110:begin t2_Tens = 2; t2_Ones = 2; end
                5'b10111:begin t2_Tens = 2; t2_Ones = 3; end
                5'b11000:begin t2_Tens = 2; t2_Ones = 4; end
                5'b11001:begin t2_Tens = 2; t2_Ones = 5; end
                5'b11010:begin t2_Tens = 2; t2_Ones = 6; end
                5'b11011:begin t2_Tens = 2; t2_Ones = 7; end
                5'b11100:begin t2_Tens = 2; t2_Ones = 8; end
                5'b11101:begin t2_Tens = 2; t2_Ones = 9; end
                5'b11110:begin t2_Tens = 3; t2_Ones = 0; end
                5'b11111:begin t2_Tens = 3; t2_Ones = 1; end
                default:begin t2_Tens = 0; t2_Ones = 5; end
                endcase
                end
        else if (a == 3'b101)  //load Green time t1
                begin
                case (L)
                //5'b00000:begin t1_Tens = 0; t1_Ones = 0; end
                5'b00001:begin t1_Tens = 0; t1_Ones = 1; end
                5'b00010:begin t1_Tens = 0; t1_Ones = 2; end
                5'b00011:begin t1_Tens = 0; t1_Ones = 3; end
                5'b00100:begin t1_Tens = 0; t1_Ones = 4; end
                5'b00101:begin t1_Tens = 0; t1_Ones = 5; end
                5'b00110:begin t1_Tens = 0; t1_Ones = 6; end
                5'b00111:begin t1_Tens = 0; t1_Ones = 7; end
                5'b01000:begin t1_Tens = 0; t1_Ones = 8; end
                5'b01001:begin t1_Tens = 0; t1_Ones = 9; end
                5'b01010:begin t1_Tens = 1; t1_Ones = 0; end
                5'b01011:begin t1_Tens = 1; t1_Ones = 1; end
                5'b01100:begin t1_Tens = 1; t1_Ones = 2; end
                5'b01101:begin t1_Tens = 1; t1_Ones = 3; end
                5'b01110:begin t1_Tens = 1; t1_Ones = 4; end
                5'b01111:begin t1_Tens = 1; t1_Ones = 5; end

                5'b10000:begin t1_Tens = 1; t1_Ones = 6; end
                5'b10001:begin t1_Tens = 1; t1_Ones = 7; end
                5'b10010:begin t1_Tens = 1; t1_Ones = 8; end
```

```
            5'b10011:begin t1_Tens = 1; t1_Ones = 9; end
            5'b10100:begin t1_Tens = 2; t1_Ones = 0; end
            5'b10101:begin t1_Tens = 2; t1_Ones = 1; end
            5'b10110:begin t1_Tens = 2; t1_Ones = 2; end
            5'b10111:begin t1_Tens = 2; t1_Ones = 3; end
            5'b11000:begin t1_Tens = 2; t1_Ones = 4; end
            5'b11001:begin t1_Tens = 2; t1_Ones = 5; end
            5'b11010:begin t1_Tens = 2; t1_Ones = 6; end
            5'b11011:begin t1_Tens = 2; t1_Ones = 7; end
            5'b11100:begin t1_Tens = 2; t1_Ones = 8; end
            5'b11101:begin t1_Tens = 2; t1_Ones = 9; end
            5'b11110:begin t1_Tens = 3; t1_Ones = 0; end
            5'b11111:begin t1_Tens = 3; t1_Ones = 1; end
            default:begin t1_Tens = 1; t1_Ones = 6; end
            endcase
            end

always@ (posedge clk1 or posedge rst)
if (rst)
      begin
      state <= S1;
      led <= 8'b00100100;        //aG bR
      Tens <= t1_Tens;      Ones <= t1_Ones;     //t=40s
      end
else
      begin
      casex (a)
      3'b000:                                    //manual 1
            begin
            state <= S9;
            led <= 8'b00100100; //aG bR
            Tens <= 4'd0;    Ones <= 4'd0;
            end
      3'b001:                                    //msnual 2
            begin
            state <= S10;
            led <= 8'b10000100; //aL bR
            Tens <= 4'd0;    Ones <= 4'd0;
            end
      3'b010:                                    //manual 3
            begin
            state <= S11;
            led <= 8'b01000010; //aR bG
            Tens <= 4'd0;    Ones <= 4'd0;
            end
      3'b011:                                    //manual 4
            begin
            state <= S12;
            led <= 8'b01001000; //aR bL
            Tens <= 4'd0;    Ones <= 4'd0;
            end
      3'b1xx:                                    //auto 5
            begin
                  case (state)
                  S1: if (Tens==0 && Ones==0)    //if both ones and tens are 0, switch to next state
                              begin
                              state <= S2;
```

```
                led <= 8'b00010100; //aY bR
                Tens <= t2_Tens;      Ones <= t2_Ones;
                end
            else if (Ones)    //ones are not 0
                begin
                Ones <= Ones - 1;
                if (Ones <= 5 && Tens == 0)
                    begin
                    led[5] <= ~led[5];    //A Green flash
                    end
                end
            else if (Tens)    //tens are not 0
                begin
                Ones <= 9;
                Tens <= Tens - 1;
                end
        S2: if (Tens==0 && Ones==0)   //if both ones and tens are 0, switch to next state
                begin
                state <= S3;
                led <= 8'b10000100; //aL bR
                Tens <= t3_Tens;      Ones <= t3_Ones;
                end
            else if (Ones)    //ones are not 0
                begin
                led[4] <= ~led[4];    //A Yellow flash
                led[5] <= 0;
                Ones <= Ones - 1;
                end
            else if (Tens)    //tens are not 0
                begin
                Ones <= 9;
                Tens <= Tens - 1;
                end
        S3: if (Tens==0 && Ones==0)   //if both ones and tens are 0, switch to next state
                begin
                state <= S4;
                led <= 8'b00010100; //aY bR 5s
                Tens <= t2_Tens;      Ones <= t2_Ones;
                end
            else if (Ones)    //ones are not 0
                begin
                Ones <= Ones - 1;
                end
            else if (Tens)    //tens are not 0
                begin
                Ones <= 9;
                Tens <= Tens - 1;
                end
        S4: if (Tens==0 && Ones==0)   //if both ones and tens are 0, switch to next state
                begin
                state <= S5;
                led <= 8'b01000010; //aR bG 40s
                Tens <= t1_Tens;      Ones <= t1_Ones;
                end
            else if (Ones)    //ones are not 0
                begin
                led[4] <= ~led[4];    //A Yellow flash
```

```
                led[7] <= 0;
                Ones <= Ones - 1;
                end
        else if (Tens)      //tens are not 0
                begin
                Ones <= 9;
                Tens <= Tens - 1;
                end
S5: if (Tens==0 && Ones==0)    //if both ones and tens are 0, switch to next state
                begin
                state <= S6;
                led <= 8'b01000001; //aR bY 5s
                Tens <= t2_Tens;       Ones <= t2_Ones;
                end
        else if (Ones)      //ones are not 0
                begin
                Ones <= Ones - 1;
                if (Ones <= 5 && Tens == 0)
                        begin
                        led[1] <= ~led[1];      //B Green flash
                        end
                end
        else if (Tens)      //tens are not 0
                begin
                Ones <= 9;
                Tens <= Tens - 1;
                end
S6: if (Tens==0 && Ones==0)    //if both ones and tens are 0, switch to next state
                begin
                state <= S7;
                led <= 8'b01001000; //aR bL 10s
                Tens <= t3_Tens;       Ones <= t3_Ones;
                end
        else if (Ones)      //ones are not 0
                begin
                led[0] <= ~led[0];      //B Yellow flash
                led[1] <= 0;
                Ones <= Ones - 1;
                end
        else if (Tens)      //tens are not 0
                begin
                Ones <= 9;
                Tens <= Tens - 1;
                end
S7: if (Tens==0 && Ones==0)    //if both ones and tens are 0, switch to next state
                begin
                state <= S8;
                led <= 8'b01000001; //aR bY 5s
                Tens <= t2_Tens;       Ones <= t2_Ones;
                end
        else if (Ones)      //ones are not 0
                begin
                Ones <= Ones - 1;
                end
        else if (Tens)      //tens are not 0
                begin
                Ones <= 9;
```

```
                        Tens <= Tens - 1;
                        end
                S8: if (Tens==0 && Ones==0)    //if both ones and tens are 0, switch to next state
                        begin
                        state <= S1;
                        led <= 8'b00100100; //aG bR  40s
                        Tens <= t1_Tens;       Ones <= t1_Ones;
                        end
                    else if (Ones)     //ones are not 0
                        begin
                        led[0] <= ~led[0];      //B Yellow flash
                        led[3] <= 0;
                        Ones <= Ones - 1;
                        end
                    else if (Tens)      //tens are not 0
                        begin
                        Ones <= 9;
                        Tens <= Tens - 1;
                        end
                default:                                        //entrance state S1
                        begin
                        state <= S1;
                        led <= 8'b00100100; //aG bR 40s
                        Tens <= t1_Tens;       Ones <= t1_Ones;
                        end
                endcase
                end
        endcase
        end

        always@ (posedge clk1 or posedge rst)
    if (rst)
        begin
        Tens_add <= t_Tens_add;  Ones_add <= t_Ones_add; //t=40s
        end
    else
        begin
        casex (a)
        3'b000:                                 //manual 1
            begin
            Tens_add <= 4'd0;    Ones_add <= 4'd0;
            end
        3'b001:                                 //msnual 2
            begin
            Tens_add <= 4'd0;    Ones_add <= 4'd0;
            end
        3'b010:                                 //manual 3
            begin
            Tens_add <= 4'd0;    Ones_add <= 4'd0;
            end
        3'b011:                                 //manual 4
            begin
            Tens_add <= 4'd0;    Ones_add <= 4'd0;
            end
        3'b1xx:                                 //auto 5
            begin
                    if (Tens_add==0 && Ones_add==0)  //if both ones and tens are 0, switch to
```

```
next state
                                    begin
                                    Tens_add <= t_Tens_add;   Ones_add <= t_Ones_add;
                                    end
                                else if (Ones_add)      //ones are not 0
                                    begin
                                    Ones_add <= Ones_add - 1;
                                    end
                                else if (Tens_add)      //tens are not 0
                                    begin
                                    Ones_add <= 9;
                                    Tens_add <= Tens_add - 1;
                                    end

                        end
                endcase
                end

endmodule


module Seg7 (rst, clk, Tens, Ones, selOnes, selTens);
        input rst, clk;             //reset, clock
        input [3:0]Tens, Ones;//traffic light count down
        output [6:0]selOnes;
        output [6:0]selTens;

        reg [6:0]selOnes;
        reg [6:0]selTens;
        reg [13:0]nonDisplay;

        always@(Ones or rst)
                begin
                if (rst)
                        selOnes = 7'b0111111;
                else
                        begin
                        case (Ones)
                                4'b0000:selOnes = 7'b0111111;
                                4'b0001:selOnes = 7'b0000110;
                                4'b0010:selOnes = 7'b1011011;
                                4'b0011:selOnes = 7'b1001111;
                                4'b0100:selOnes = 7'b1100110;
                                4'b0101:selOnes = 7'b1101101;
                                4'b0110:selOnes = 7'b1111101;
                                4'b0111:selOnes = 7'b0000111;
                                4'b1000:selOnes = 7'b1111111;
                                4'b1001:selOnes = 7'b1101111;
                                default:selOnes = 7'b0111111;
                        endcase
                        end
                end

        always@(Tens or rst)
                begin
                if (rst)
                        selTens = 7'b0111111;
```

```
        else
            begin
            case (Tens)
                4'b0000:selTens = 7'b0111111;
                4'b0001:selTens = 7'b0000110;
                4'b0010:selTens = 7'b1011011;
                4'b0011:selTens = 7'b1001111;
                4'b0100:selTens = 7'b1100110;
                4'b0101:selTens = 7'b1101101;
                4'b0110:selTens = 7'b1111101;
                4'b0111:selTens = 7'b0000111;
                4'b1000:selTens = 7'b1111111;
                4'b1001:selTens = 7'b1101111;
                default:selTens = 7'b0111111;
            endcase
            end
        end
endmodule

module Seg7_add (rst, clk, Tens_add, Ones_add, selOnes_add, selTens_add);
    input rst, clk;          //reset, clock
    input [3:0]Tens_add, Ones_add;  //traffic light count down
    output [6:0]selOnes_add;
    output [6:0]selTens_add;

    reg [6:0]selOnes_add;
    reg [6:0]selTens_add;

    always@(Ones_add or rst)
        begin
        if (rst)
            selOnes_add = 7'b0111111;
        else
            begin
            case (Ones_add)
                4'b0000:selOnes_add = 7'b0111111;
                4'b0001:selOnes_add = 7'b0000110;
                4'b0010:selOnes_add = 7'b1011011;
                4'b0011:selOnes_add = 7'b1001111;
                4'b0100:selOnes_add = 7'b1100110;
                4'b0101:selOnes_add = 7'b1101101;
                4'b0110:selOnes_add = 7'b1111101;
                4'b0111:selOnes_add = 7'b0000111;
                4'b1000:selOnes_add = 7'b1111111;
                4'b1001:selOnes_add = 7'b1101111;
                default:selOnes_add = 7'b0111111;
            endcase
            end
        end

    always@(Tens_add or rst)
        begin
        if (rst)
            selTens_add = 7'b0111111;
        else
            begin
            case (Tens_add)
```

```
                    4'b0000:selTens_add = 7'b0111111;
                    4'b0001:selTens_add = 7'b0000110;
                    4'b0010:selTens_add = 7'b1011011;
                    4'b0011:selTens_add = 7'b1001111;
                    4'b0100:selTens_add = 7'b1100110;
                    4'b0101:selTens_add = 7'b1101101;
                    4'b0110:selTens_add = 7'b1111101;
                    4'b0111:selTens_add = 7'b0000111;
                    4'b1000:selTens_add = 7'b1111111;
                    4'b1001:selTens_add = 7'b1101111;
                    default:selTens_add = 7'b0111111;
                endcase
                end
        end
endmodule
```