Traffic Light System

Chai Wei Chee, Goh Ken Onn, Goh Wei Shun and Hong Sing Min

***Abstract*— This experiment is to produce a systematic traffic light system that focuses on a main road and then a farm road. The traffic lights at the main road would take turns to run until there is or are car(s) been detected at the farm road. When it happened, the current main traffic lights would be allowed to run from green to yellow then to red then it would be switched to the traffic light of the farm road to run. A truth table was drawn to implement this system. State diagram and table were derived from it and then a combinational and sequential logic circuit diagram was drawn according to it. In this project, Quartus II Verilog coding was used using Altera Quartus II Tools software and the design was demonstrated using Altera FPGA DE2 board.**

I. Introduction

Given a cross junction, where one road is the main road while the other is a farm road as shown in Fig. 1. There are two motion sensors M1 and M2 at the farm road, each of them is situated before traffic lights SC and SD respectively. Each main traffic light SA and SB take turns to change colour. Only one traffic light changes to green for 12 seconds and then yellow for 2 seconds at a time. However, when sensor M1 or M2 detects the presence of car, after one main traffic light changes to red, traffic light SC or SD changes to green for 12 seconds and then yellow for 2 seconds depending on which sensor senses the car (SC changes colour when M1 senses car whereas SD changes colour when M2 senses car). If both sensors M1 and M2 sense the presence of car, SC is given the priority first to change colour, followed by SD.

The overall concept of the system was implemented into a truth table. And then from the truth table, we designed the combinational and sequential circuit design using functional block diagrams. From the block diagrams, we implemented the diagrams into Altera Quartus II Tools software using Verilog code. And then we programmed the code into Altera FPGA DE2 board to demonstrate the prototype.

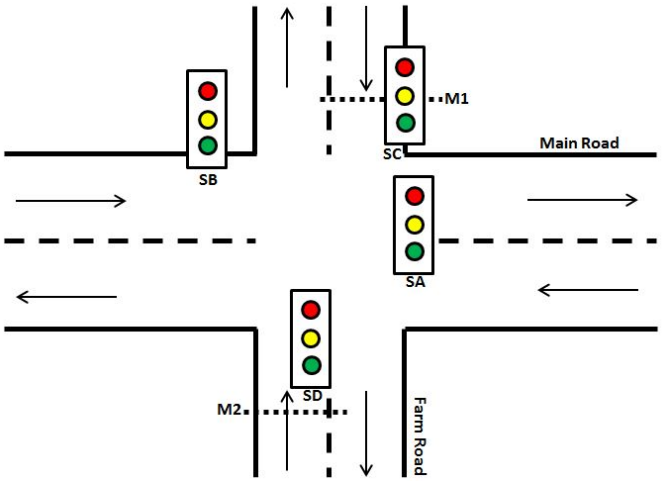

Fig. 1. The Traffic Light System

II. Procedure

1. Specifications of the traffic light system were identified.
2. Inputs and outputs of the traffic light system were all determined.
3. State transition diagram of the traffic light system was designed and drawn out for the ease of following design.
4. Block diagrams of data-path and control unit were designed based on the state transition diagram and specifications of the traffic light systems.
5. Verilog code of the traffic light system based on the above designs was created and simulated by using Quartus ll.
6. Functions, pins and clock frequency of the FPGA board provided were identified.
7. A frequency divider was designed to make the system's clock frequency to 1 second per cycle.
8. All inputs and outputs of the traffic light system were mapped to the pins of the FPGA board according to the pin panel of the FPGA board.
9. The Verilog code of the traffic light system was programmed into the FPGA board to run the system.
10. The traffic light system on the FPGA board was tested and checked to make sure it fulfill all the specifications of the design.

III. Data and Results

The overview of the system is as shown in TABLE I. Looking at the system as a whole, the top level module is named Traffic Light System. This top level module branches out into three different sub modules, which are the Datapath Unit, Control Unit and Frequency Divider.

TABLE I. THE OVERVIEW OF THE SYSTEM

Module Level	Module Name	Module Content
Top Level Module	Traffic Light System	Sub Level Modules
Sub Level Module	Datapath Unit	4-bits Register
		Increment
		Seven-segment Decoder
		Subtractor
		Detector of 1 second
		Detector of 12 seconds
	Control Unit	Combinational Logic Circuit
		3-bits D Flip-Flops
		12-bits D Flip-Flops
	Frequency Divider	

A. Top Level Module

Traffic Light System

```
1  module trafficlight(extclk,M,A,B,C,D,leds);
2  input [1:2]M;
3  input extclk;
4  output [2:0]A,B,C,D;
5  output [1:14]leds;
6  wire TL,TS,en,clr,clk;
7
8  freqdiv(extclk,clk);
9  datapath(en,clk,clr,TL,TS,leds);
10 control(clk,M,TL,TS,A,B,C,D,clr,en);
11 endmodule
```

Fig. 2. Verilog Code of Traffic Light System

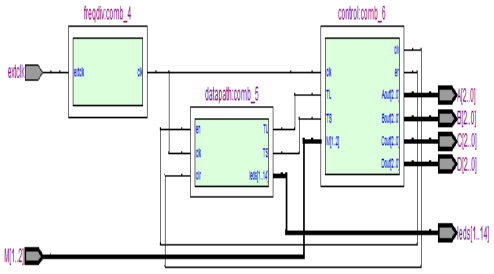

Fig. 3. Functional Block Diagram of Traffic Light System

Comment: The traffic light system has 3 main structures, which are Control Unit, Datapath Unit and Frequency Divider.

B. Sub Level Module

1) Datapath Unit

```
1  module datapath(en,clk,clr,TL,TS,leds);
2  input en,clk,clr;
3  output [1:14]leds;
4  output TL,TS;
5  wire [3:0]T,T1,bcd;
6
7  reg4(clk,clr,T1,T);
8  inc(T,T1);
9  sub(T,bcd);
10 seg7(en,bcd,leds);
11 det1s(T,TS);
12 det11s(T,TL);
13 endmodule
```

Fig. 4. Verilog Code of Datapath Unit

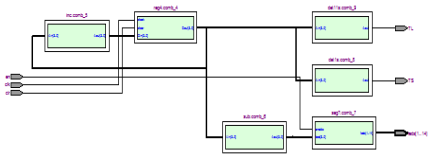

Fig. 5. Functional Block Diagram of Datapath

Comment: Datapath Unit is consists of 6 smaller blocks, which are 4-bits register, increment, subtractor, 7-segment display, 1-second detector, and 12-seconds detector.

- 4-bits Register

```
1  module reg4(clock,clear,Din,Dout);
2  input [3:0]Din;
3  input clock,clear;
4  output [3:0]Dout;
5  reg [3:0]Dout;
6
7  always@(negedge clock)
8      if(clear) Dout<=0;
9      else Dout<=Din;
10 endmodule
```

Fig. 6. Verilog Code of 4-bits Register

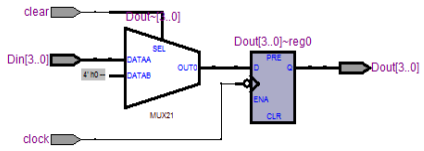

Fig. 7. Functional Block Diagram of 4-bits Register

Comment: 4 bit register acted as the timer when it connects to increment.

- Increment

```
1  module inc(Ain,Aout);
2  input [3:0]Ain;
3  output [3:0]Aout;
4
5  assign Aout=Ain+1;
6  endmodule
```

Fig. 8. Verilog Code of Increment

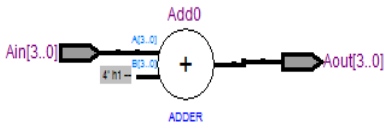

Fig. 9. Functional Block Diagram of Increment

Comment: Increment is used to increase the input by one.

- 7-segment Decoder

```
module seg7(enable,bcd,leds);
input [3:0]bcd;
input enable;
output [1:14]leds;          // bit 1-7 seg1 bit 8-14 seg2
reg [1:14]leds;

always@(enable or bcd)
    if(enable==1)
        case(bcd)
        0: leds=14'b00000010000001;
        1: leds=14'b00000011001111;
        2: leds=14'b00000010010010;
        3: leds=14'b00000010000110;
        4: leds=14'b00000011001100;
        5: leds=14'b00000010100100;
        6: leds=14'b00000010100000;
        7: leds=14'b00000010001111;
        8: leds=14'b00000010000000;
        9: leds=14'b00000010000100;
        10: leds=14'b10011110000001;
        11: leds=14'b10011111001111;
        12: leds=14'b10011110010010;
        default:leds=14'bxxxxxxxxxxxxxx;
        endcase
    else if(enable==0) leds=14'b11111101111110;
endmodule
```

Fig. 10. Verilog Code of 7-segment Decoder

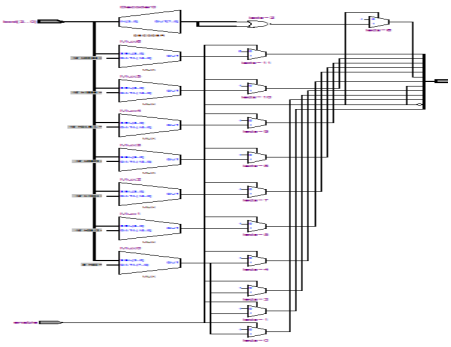

Fig. 11. Functional Block Diagram of 7-segment Decoder

Comment: 7 seven display used to show the output from the 4 bit register.

- Subtractor

```
module sub(Ain,Aout);
  input [3:0]Ain;
  output [3:0]Aout;

  assign Aout=12-Ain;
  endmodule
```

Fig. 12. Verilog Code of Subtractor

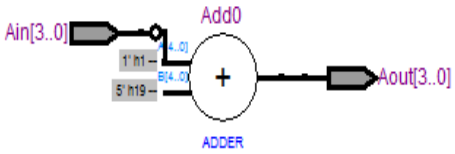

Fig. 13. Functional Block Diagram of Subtractor

Comment: subtractor used to count down the output from 4 bit register.

- 1-second detector

```
module det1s(Ain,Aout);
  input [3:0]Ain;
  output Aout;
  reg Aout;

  always@(Ain)
      if(Ain==4'b0001) Aout=0;
      else Aout=1;
endmodule
```

Fig. 14. Verilog Code of 1-second detector

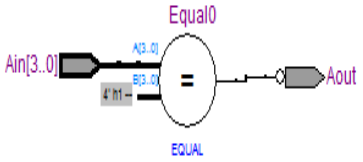

Fig. 15. Functional Block Diagram of 1-second detector

Comment: 1-second detector was used to ensure the time reach 1s only. When Ain is 1, Aout is 0.

- 12-seconds detector

```
module det11s(Ain,Aout);
  input [3:0]Ain;
  output Aout;
  reg Aout;

  always@(Ain)
      if(Ain==4'b1011) Aout=0;
      else Aout=1;
endmodule
```

Fig. 16. Verilog Code of 12-seconds detector

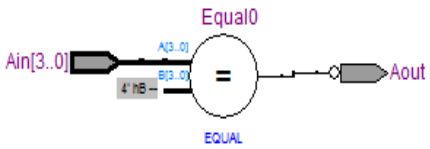

Fig. 17. Functional Block Diagram of 12-seconds detector

Comment: 12s detector use to ensure the time reach 12s only. When Ain is 11, Aout is 0.

2) Control Unit

```
module control(clk,M,TL,TS,Aout,Bout,Cout,Dout,clr,en);
input [1:2]M;
input TL,TS,clk;
output [2:0]Aout,Bout,Cout,Dout;
output en,clr;
wire [2:0]PS,NS,Ain,Bin,Cin,Din;

comlogic(M,TL,TS,PS,NS,Ain,Bin,Cin,Din,clr,en);
D12ff(clk,Ain,Bin,Cin,Din,Aout,Bout,Cout,Dout);
D3ff(clk,NS,PS);
endmodule
```

Fig. 18. Verilog Code of Control Unit

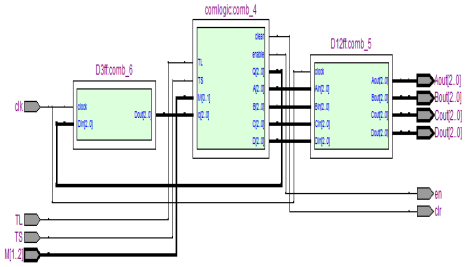

Fig. 19. Functional Block Diagram of Control Unit

Comment: Control Unit is consists of 3 component, which are 12-bits D flip-flop, 3-bits D flip-flop, and combinational logic circuit.

- Combinational Logic Circuit

```
module comlogic(M,TL,TS,q,Q,A,B,C,D,clear,enable);
input [0:1]M;
input TL,TS;
input [2:0]q;
output [2:0]Q,A,B,C,D;
output clear,enable;
reg [2:0]Q,A,B,C,D;
reg clear,enable;
parameter [2:0] s0=3'b000, s1=3'b001, s2=3'b010, s3=3'b011,
                s4=3'b100, s5=3'b101, s6=3'b110, s7=3'b111;

always@(M or TL or TS or q)
   case(q)
      s0: if(TL==1) begin Q=s0; A=3'b001; B=3'b100; C=3'b100; D=3'b100; clear=0; enable=1; end
          else if(TL==0) begin Q=s1; A=3'b010; B=3'b100; C=3'b100; D=3'b100; clear=1; enable=1; end
      s1: if(TS==1) begin Q=s1; A=3'b010; B=3'b100; C=3'b100; D=3'b100; clear=0; enable=0; end
          else if(TS==0) begin Q=s2; A=3'b100; B=3'b001; C=3'b100; D=3'b100; clear=1; enable=0; end
      s2: if(TL==1) begin Q=s2; A=3'b100; B=3'b001; C=3'b100; D=3'b100; clear=0; enable=1; end
          else if(TL==0) begin Q=s3; A=3'b100; B=3'b010; C=3'b100; D=3'b100; clear=1; enable=1; end
      s3: if(TS==1) begin Q=s3; A=3'b100; B=3'b010; C=3'b100; D=3'b100; clear=0; enable=0; end
          else if((M,TS)==3'b000) begin Q=s0; A=3'b001; B=3'b100; C=3'b100; D=3'b100; clear=1; enable=0; end
          else if(((M,TS)==3'b100)||((M,TS)==3'b110)) begin Q=s4; A=3'b100; B=3'b100; C=3'b001; D=3'b100; clear=1; enable=0; end
          else if((M,TS)==3'b010) begin Q=s6; A=3'b100; B=3'b100; C=3'b100; D=3'b001; clear=1; enable=0; end
      s4: if(TL==1) begin Q=s4; A=3'b100; B=3'b100; C=3'b001; D=3'b100; clear=0; enable=1; end
          else if(TL==0) begin Q=s5; A=3'b100; B=3'b100; C=3'b010; D=3'b100; clear=1; enable=1; end
      s5: if(TS==1) begin Q=s5; A=3'b100; B=3'b100; C=3'b010; D=3'b100; clear=0; enable=0; end
          else if(((M,TS)==3'b000)||((M,TS)==3'b100)) begin Q=s0; A=3'b001; B=3'b100; C=3'b100; D=3'b100; clear=1; enable=0; end
          else if(((M,TS)==3'b010)||((M,TS)==3'b110)) begin Q=s6; A=3'b100; B=3'b100; C=3'b100; D=3'b001; clear=1; enable=0; end
      s6: if(TL==1) begin Q=s6; A=3'b100; B=3'b100; C=3'b100; D=3'b001; clear=0; enable=1; end
          else if(TL==0) begin Q=s7; A=3'b100; B=3'b100; C=3'b100; D=3'b010; clear=1; enable=1; end
      s7: if(TS==1) begin Q=s7; A=3'b100; B=3'b100; C=3'b100; D=3'b010; clear=0; enable=0; end
          else if(TS==0) begin Q=s0; A=3'b001; B=3'b100; C=3'b100; D=3'b100; clear=1; enable=0; end
   endcase
endmodule
```

Fig. 20. Verilog Code of Combinational Logic Circuit

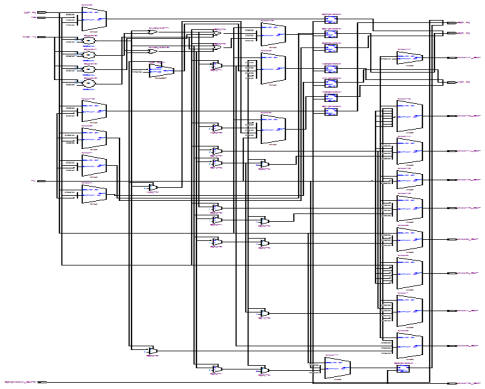

Fig. 21. Functional Block Diagram of Combinational Logic Circuit

Comment: Combinational Logic Circuit performs the logic operation of traffic light system.

- 3-bits D Flip-Flop

```
module D3ff(clock,Din,Dout);
input [2:0]Din;
input clock;
output [2:0]Dout;
reg [2:0]Dout;

always@(negedge clock)
    Dout=Din;
endmodule
```

Fig. 22. Verilog Code of 3-bits D Flip-Flop

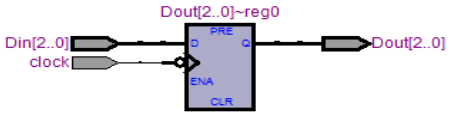

Fig. 23. Functional Block Diagram of 3-bits D Flip-Flop

Comment: 3 bit D flip-flop acts as memory to store the data from the output of combinational logic circuit and sensors (M1 and M2) during 1 clock cycle.

- 12-bits D Flip-Flop

```
module D12ff(clock,Ain,Bin,Cin,Din,Aout,Bout,Cout,Dout);
 input [2:0]Ain,Bin,Cin,Din;
 input clock;
 output [2:0]Aout,Bout,Cout,Dout;
 reg [2:0]Aout,Bout,Cout,Dout;

 always@(negedge clock)
    begin
    Aout<=Ain;
    Bout<=Bin;
    Cout<=Cin;
    Dout<=Din;
    end
endmodule
```

Fig. 24. Verilog Code of 12-bits D Flip-Flop

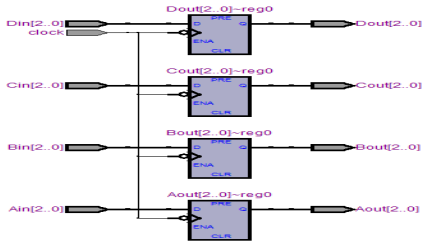

Fig. 25. Functional Block Diagram of 12-bits D Flip-Flop

Comment: 12 bit D flip-flop acts as memory to store the data of the output of combinational logic circuit during 1 clock cycle.

3) Frequency Divider

```
module freqdiv(extclk,clk);
input extclk;
output clk;
reg clk;
reg [31:0]count;

always@(negedge extclk)
    if(count==27000000) begin count<=0; clk<=1; end
    else if(count!=27000000) begin count<=count+1; clk<=0; end
endmodule
```

Fig. 26. Verilog Code of Frequency Divider

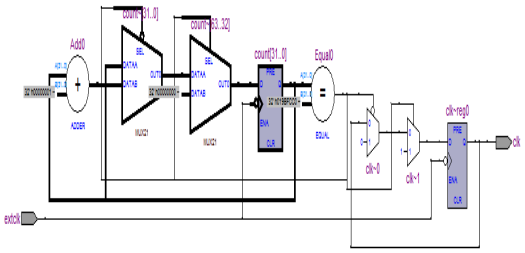

Fig. 27. Functional Block Diagram of Frequency Divider

Comment: frequency divider used to decrease the frequency of the traffic light system in order to achieve 1 second at FPGA board.

C. State Diagram

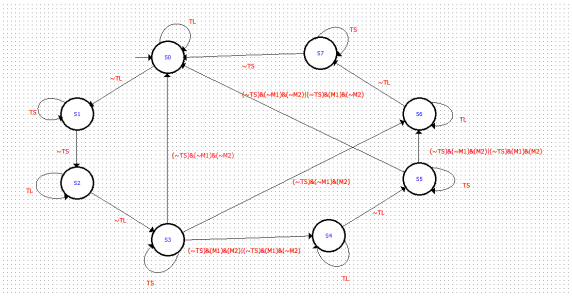

Fig. 28. State Diagram of Traffic Light System

D. Results

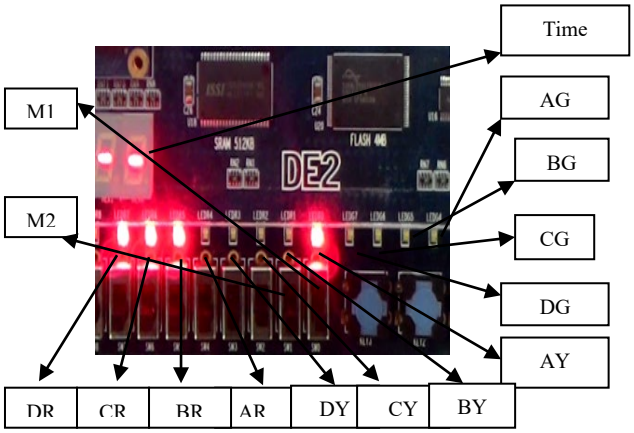

Fig. 29. AG to DG represent Green lights. AY to DY represent Yellow lights. AR to DR represent Red lights. Time represents the countdown timer at 7-segment display. M1 and M2 denote the sensors.

Fig. 30. When TL = 1, A is green, the rest is red.

Fig. 31. When TL =0, A is yellow, the rest is red.

Fig. 32. When TS = 1, A is yellow, the rest is red.

Fig. 33. When TS =0, B is green, the rest is red.

Fig. 34. When TL =1, B is green, the rest is red.

Fig. 35. When TL = 0, B is yellow, the rest is red.

Fig. 36. When TS =1, B is yellow, the rest is red.

Fig. 37. When TS = 0, M1 = 1, C is green, the rest is red.

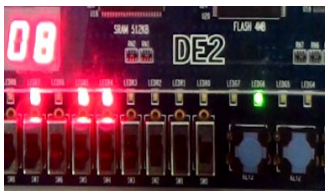

Fig. 38. When TL = 1, M1 = 1, C is green, the rest is red.

Fig. 39. When TL = 0, C is yellow, the rest is red.

Fig. 40. When TS = 1, C is yellow, the rest is red.

Fig. 41. When TS =0, M1 = M2 = 1, C is green, the rest is red.

Fig. 42. When TL = 1, M1 = M2 = 1, C is green, the rest is red.

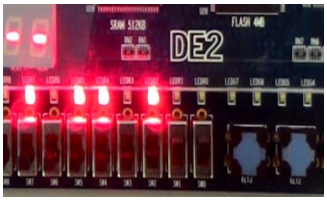

Fig. 43. When TL = 0, C is yellow, the rest is red.

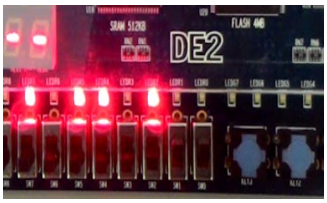

Fig. 44. When TS = 1, C is yellow, the rest is red.

Fig. 45. When TS = 0, M2 = 1, D is green, the rest is red.

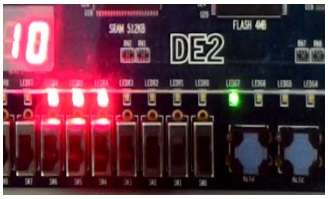

Fig. 46. When TL = 1, M2 = 1, D is green, the rest is red.

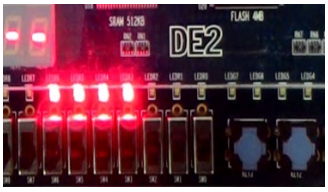

Fig. 47. When TL = 0, D is yellow, the rest is red.

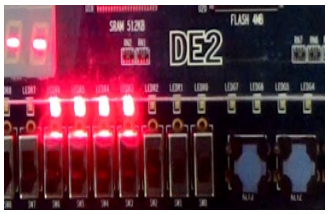

Fig. 48. When TS = 1, D is yellow, the rest is red.

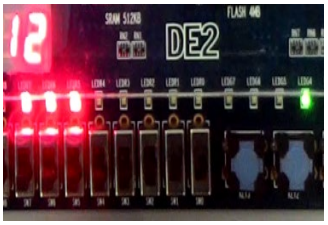

Fig. 49. When TS = 0, M1 = M2 = 0, A is green, the rest is red.

IV. Discussion

The whole traffic light circuit consists of three main parts; frequency divider, datapath unit and control unit.

Since the available clock frequencies are only 27MHz and 50MHz, there is a need to reduce the clock frequency to the suitable frequency of the circuit design. For this circuit, the suitable frequency is 1Hz. This is because the counter is made up of a 4-bit register with increment every clock cycle. Hence, to replicate 1 clock cycle of 1 second, the frequency must be set to 1Hz. To implement this, a frequency divider module is added to take the higher clock frequency and turn it down to a lower frequency.

For the datapath unit, the purpose is to send long timer signal, TL and short timer signal, TS to the control unit and to display the timer on 2 BCD to 7-segment displays.

The timer signals are determined by a detector which are 11s detector module and 1s detector module for long timer and short timer respectively. The timer is generated by a 4-bit register with increment every 1 clock cycle. Since the time, T starts from 0, if we want 12 seconds to activate the long timer signal, TL, then T must equal to 11. As for short timer, TS, the duration is 2 seconds, which means T equal to 1. Once any of the detectors detects the corresponding value of T, its output will be set to low (0).

As for the display of timer, it should be a countdown sequence. However, since the timer generates in count up sequence, we need to invert the sequence. To do this, a 12-T module is added to the datapath circuit. Example, when T=0, the output is 12. The output is then sent to BCD to 7 segment display to show the time.

For the control unit, its controls the outputs based on combinational logic with few inputs which are the sensors at junction, M1 and M2, TL, TS and the present state. The control unit is programmed based on the state transition diagram. The output of the combinational logic which controls the state (Q2, Q1, Q0) is sent to the D flip flop to determine the next state. The other output involves the traffic lights of each junction, the enable and clear signal. The enable signal is to display the timer when it is at green light and clear is to reset the timer when it is needed.

The modules in the traffic light system are designed by using Verilog. Since the complexity of the circuit is quite high, it is easier to design by Verilog instead of using schematic diagram. The datapath and control unit are programmed using structural modelling while the sub modules are programmed using behavioural modelling. Another advantage of using Verilog it is easier to debug since each sub module is constructed separately.

V. Conclusion

The traffic light system was prototyped and demonstrated successfully by using Altera FPGA DE2 board. The traffic light system was constructed by HDL coding and was verified its function successfully by using Altera Quartus II Tools software. Throughout the experiment, it was been found that Verilog is better than schematic-drawing in terms of complexity in creating each module.

References

[1] Mohamed Khalil Hani, *RTL Design of Digital Systems with Verilog*, UTM, 2013.

[2] S. Brown and Z. Vranesic, *Fundamentals of Digital Logic with Verilog Design*. 2nd ed. Singapore: McGraw-Hill, 2005.

[3] M. J. S. Smith, *Application-Specific Integrated Circuits*. Addison Wesley, 1997.

[4] Randy H. Katz and Gaetano Borriello, *Contemporary Logic Design*. 2nd ed. Upper Saddle River, NJ: Pearson Education, Inc., 2005.

[5] John F. Wakerly, *Digital Design: Principles and Practices*. Fourth Ed Pearson Prentice Hall 2006.

[6] D. D. Givone, *Digital Principles and Design*, McGraw-Hill, 200