SAN JOSÉ STATE UNIVERSITY
CharlesW.DavidsonCollege of Engineering
DEPARTMENT OF ELECTRICAL ENGINEERING

EE271 – Advanced Digital System Design and Synthesis

Fall 2017 FPGA Project Report

Topic: Design and Implementation of traffic signal controller

Name	Deekshith Krishnegowda Rajesh Krishna Mundrpaddy	SJSU ID	012417080 012184328
Email	deekshithkrishnegowda@yahoo.com rajeshkrishna093@gmail.com	Phone	(669)281-9059 (669)281-9064

Executive Summary
Designed, synthesized and checked proper functioning on spartan 6 series FPGA of two traffic signals separated by few meters. Synchronised the two signals such that the second signal works after 3 seconds and in the window of the first signal. Frequency divider logic was written to bring the frequency down from 100MHz to 1 Hz.

This is actual traffic signal found near 280N and 10th & 11th street in San Jose.

I. GeneralProject Information

Table I.1: List of EDA Tools Used

EDA Tool Name	Company	You Used it for
Xilinx XST	Xilinx	Synthesis
Xilinx Isim	Xilinx	Simulation & test

Table I.2: Device and package details

Device name	Used with (EDA tool name)
Spartan 6 xc6slx9-2csg324	Xilinx XST

Table I.3: List of Verilog Modules (both design and test modules)

Module Name	Ports	Short Description
Traffic_contoller	clock,reset, pedestrain_north_input, pedestrain_south_input, pedestrain_east_input, pedestrain_west_input, pedestrain_north_one_input, pedestrain_south_one_input, pedestrain_east_one_input, pedestrain_west_one_input, red_north, yellow_north, green_north, red_west, yellow_west, green_west, red_north_one, yellow_north_one, green_north_one, red_east_one, yellow_east_one, green_east_one, pedestrain_north, pedestrain_south, pedestrain_east, pedestrain_west, pedestrain_north_one, pedestrain_south_one, pedestrain_east_one, pedestrain_west_one	Finite state machine which controls the flow of output signals.

Traffic_contoller_tb	clock,reset, pedestrain_north_input, pedestrain_south_input, pedestrain_east_input, pedestrain_west_input, pedestrain_north_one_input, pedestrain_south_one_input, pedestrain_east_one_input, pedestrain_west_one_input, red_north, yellow_north, green_north, red_west, yellow_west, green_west, red_north_one, yellow_north_one, green_north_one, red_east_one, yellow_east_one, green_east_one, pedestrain_north, pedestrain_south, pedestrain_east, pedestrain_west, pedestrain_north_one, pedestrain_south_one, pedestrain_east_one, pedestrain_west_one	Test bench to the FSM module

II. Implementation Overview

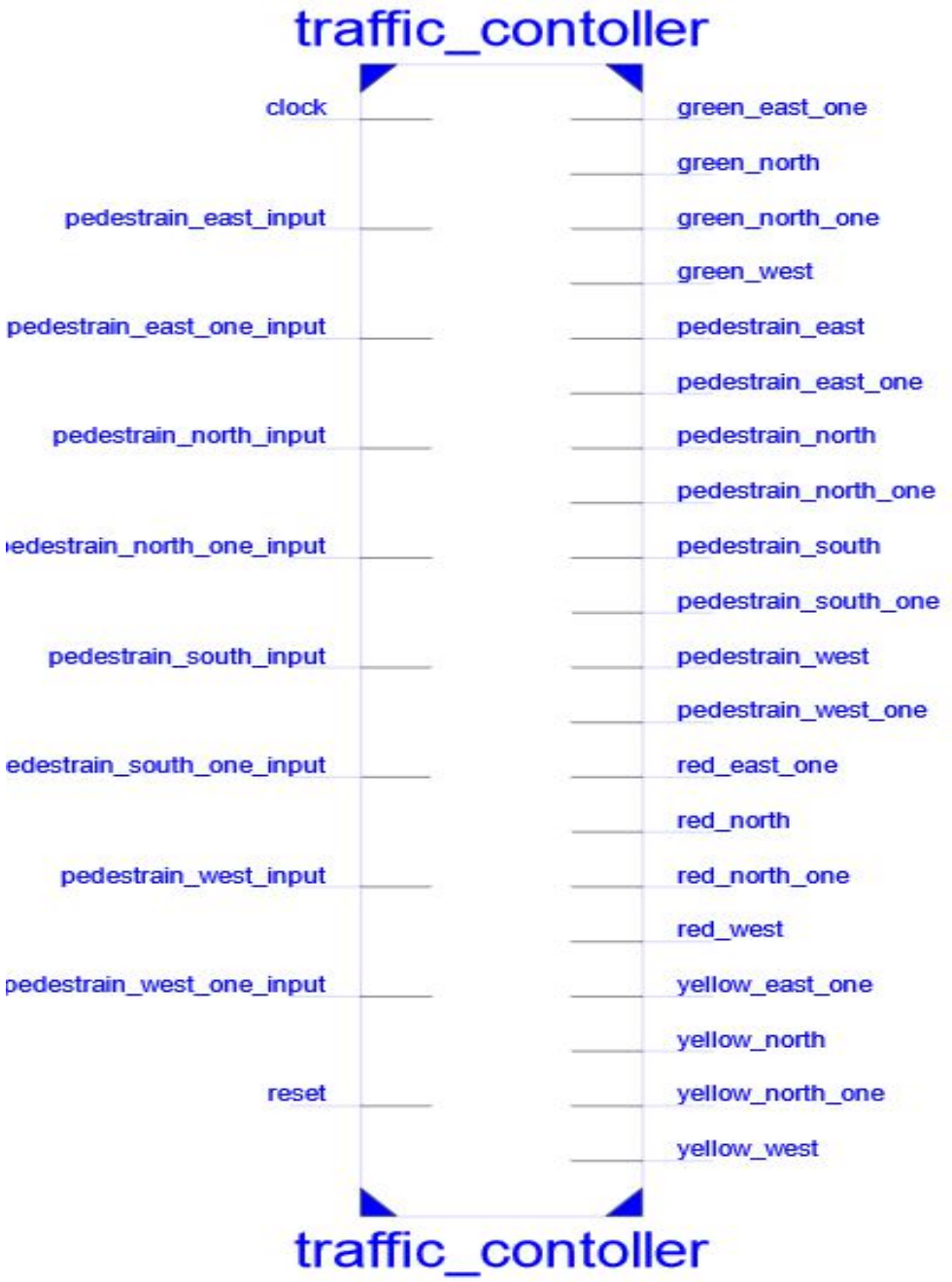
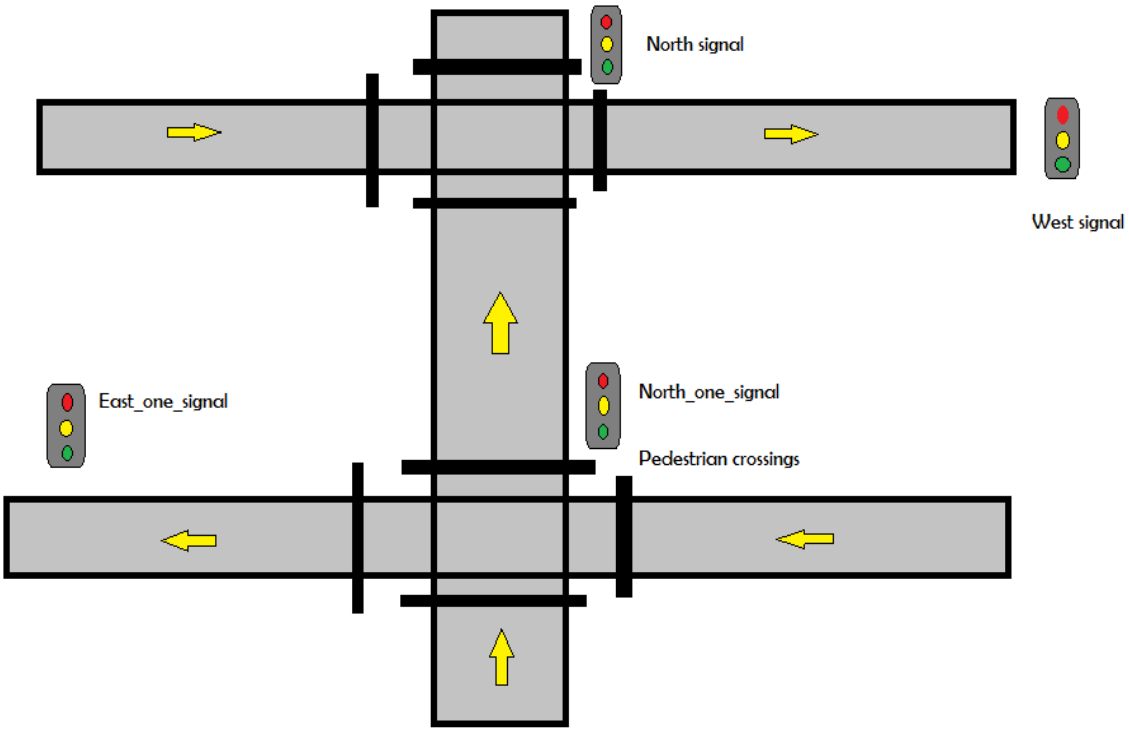


Figure II.1: Block Diagram of top



The north_one_signal acts in the window of north_signal. After 3 seconds of green in north_signal, the north_one_signal starts to work so that the traffic in the middle section is cleared.

Important things to be noticed here ,

- Green in north_signal goes first, then after 3 seconds green on north_one_signal appears
- Red in north_one_signal goes high and after few seconds later red in north_signal goes high
- Pedestrian crossing works when corresponding red signal is on and when pedestrian input is available.
- The east_one_signal and west_signal works simultaneously.

III. RTL-Level(Pre-synthesis) Simulations/Tests

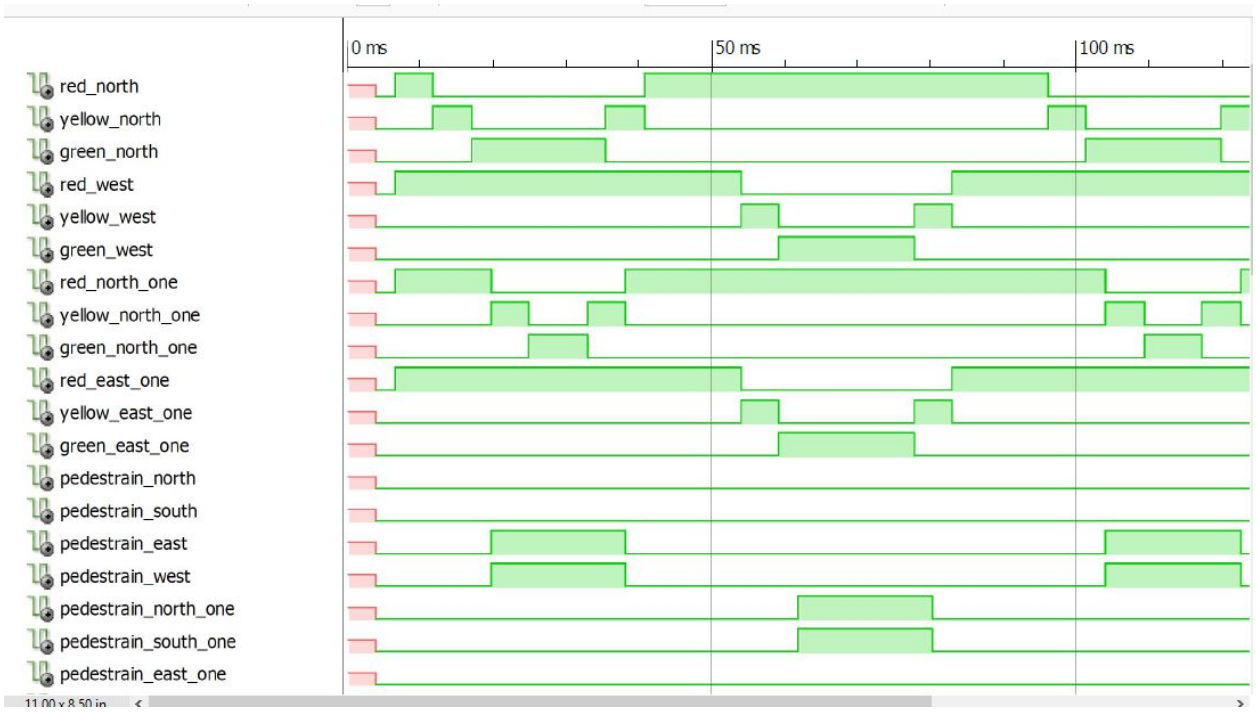


Figure III.1a:RTL simulation waveform that shows small part of simulation

TESTBENCH SNIPPET

```
initial
begin
//    $dumpfile("traffic.vcd");
//    $dumpvars;
      @(negedge clock) reset=1;
      @(negedge clock) reset=0;
      {pedestrain_north_input,
pedestrain_south_input,pedestrain_east_input,pedestrain_west_input,
             pedestrain_north_one_input,
pedestrain_south_one_input,pedestrain_east_one_input,pedestrain_west_one_inpu
t}={$random}%256;

      #140;
      $finish;
end
```

IV. Conclusion

The working of four traffic signals as described above was successfully tested on Numato lab's Spartan 6 FPGA at1Hz clock frequency .

Appendix A
Completed Verilog Source Codes

```
module traffic_contoller(input clock,reset,pedestrain_north_input,
pedestrain_south_input,pedestrain_east_input,pedestrain_west_input,
                                    pedestrain_north_one_input,
pedestrain_south_one_input,pedestrain_east_one_input,pedestrain_west_one_inpu
t,
                    output reg red_north,yellow_north,green_north,
                            red_west,yellow_west,green_west,

                            red_north_one,yellow_north_one,green_north_one,
                            red_east_one,yellow_east_one,green_east_one,

                    output reg
pedestrain_north,pedestrain_south,pedestrain_east,pedestrain_west,

pedestrain_north_one,pedestrain_south_one,pedestrain_east_one,pedestrain_west
_one);

parameter state_north=1'b0,
        state_west=1'b1;

reg ns,ps; //next_state and present_state
reg [3:0] count_north,count_west;
reg [2:0] count_north_one;/*count_east_one,count_west_one*/
reg temp;
reg [35:0] count;
reg clock_1hz;

always@(posedge clock)
begin
      if(reset)
      begin
            count<=0;clock_1hz<=0;
      end
      else
      begin
            if(count==(10**8))begin
                  count<=0;
                  clock_1hz<=!clock_1hz;
                        end
            else
                  count<=count+1;
      end
end
////////////////////////////////////////////////////////////////////counter_logi
c/////////////////////////////////////////
always@(posedge clock_1hz)
begin
      if(reset)
            begin
            count_north<=0;count_west<=0;
            count_north_one<=0;
            end
```

```
      else if (count_north==15||count_west==15)
            begin
            count_north<=0;count_west<=0;
            end
/*    else if(count_north_one==7)
            begin
            count_north_one<=0;count_east_one<=0;count_west_one<=0;
            end*/
      else
            begin
            case (ps)
            state_north:      begin
                              count_north<=count_north+1'b1;
                              if(count_north>=5 && count_north<=12)

count_north_one<=count_north_one+1'b1;
                              else
                                          count_north_one<=0;


                        end
            state_west: begin
                              count_west<=count_west+1'b1;
                        /*    if(count_west>=5)
count_west_one<=count_west_one+1'b1;
                              else          count_west_one<=0;*/

                        end

            endcase
            end
end


////////////////////////////////////////////////////////present_state///////
//////////////////////////////////
always@(posedge clock_1hz)
begin
      if(reset)
            ps<=state_north;
      else
      begin
            ps<=ns;
      end
end



////////////////////////////////////////////////////////////////next_state/////
//////////////////////////////////
always@(*)
begin
      case(ps)
      state_north:if(count_north==15) ns=state_west;
                  else ns=state_north;
      state_west: if(count_west==15) ns=state_north;
```

```
                else ns=state_west;
        default: ns=state_north;
        endcase
end



////////////////////////////////////////////////////////////////////output_logic
//////////////////////////////////////
always@(posedge clock_1hz)
begin
        if(reset)
              begin
                          red_north<=0;yellow_north<=0;green_north<=0;
                          red_west<=0;yellow_west<=0;green_west<=0;


        red_north_one<=0;yellow_north_one<=0;green_north_one<=0;
                          red_east_one<=0;yellow_east_one<=0;green_east_one<=0;


        pedestrain_north<=0;pedestrain_south<=0;pedestrain_east<=0;pedestrain_w
est<=0;

        pedestrain_north_one<=0;pedestrain_south_one<=0;pedestrain_east_one<=0;
pedestrain_west_one<=0;
              end
        else
        begin
        case(ps)
              state_north:
              begin
                          temp<=1'bz;
                    red_north<=1;red_west<=1;
                    yellow_north<=0;yellow_west<=0;
                    green_north<=0;green_west<=0;

                    red_north_one<=1;yellow_north_one<=0;green_north_one<=0;
                    red_east_one<=1;yellow_east_one<=0;green_east_one<=0;


                           if(count_north>=0&&count_north<=2)
                                begin
                                red_north<=1;yellow_north<=0;green_north<=0;

        red_north_one<=1;yellow_north_one<=0;green_north_one<=0;
                                end
                           else
                                temp<=1'bz;
                           if(count_north>=2&&count_north<=4)
                                begin
                                red_north<=0;yellow_north<=1;green_north<=0;
                                end
                           else
                                temp<=1'bz;
                          if(count_north>=4 && count_north<=11)
                                begin
```

```
                              green_north<=1;yellow_north<=0;red_north<=0;
                              end
                              else
                                    temp<=1'bz;
                        if(count_north>=11 && count_north<=13)
                              begin
                              green_north<=0;yellow_north<=1;red_north<=0;
                              end
                        else
                              temp<=1'bz;
                        if(count_north>=13 && count_north<=15)
                              begin
                              yellow_north<=0;red_north<=1;green_north<=0;
                              red_north_one<=1;
                              end
                        else
                              temp<=1'bz;

                        if(red_west &&
(pedestrain_west_input||pedestrain_east_input) && green_north ) //pedestrain
crossing
                              begin
                              pedestrain_west<=1;
                              pedestrain_east<=1;
                                end
                        else
                              begin
                              pedestrain_west<=0;
                              pedestrain_east<=0;
                              end



                        if(red_east_one &&
(pedestrain_west_one_input||pedestrain_east_one_input) && green_north_one )
//pedestrain crossing
                              begin
                              pedestrain_west_one<=1;
                              pedestrain_east_one<=1;
                                end
                        else
                              begin
                              pedestrain_west_one<=0;
                              pedestrain_east_one<=0;
                              end



                        if(count_north>=5 && count_north<=12)
                        begin
                              if(count_north_one>=0 && count_north_one<2)
                                    begin

      red_north_one<=0;yellow_north_one<=1;green_north_one<=0;
                                    end
                              else
                                      temp<=1'bz;
```

```
                        if(count_north_one>=2 && count_north_one<5)
                              begin

red_north_one<=0;yellow_north_one<=0;green_north_one<=1;
                              end
                        else
                              temp<=1'bz;

                        if(count_north_one>=5 && count_north_one<7)
                              begin

red_north_one<=0;yellow_north_one<=1;green_north_one<=0;
                              end
                        else
                              temp<=1'bz;
                  end
                  else
                        temp<=1'bz;



      end

      state_west:
      begin
       red_north<=1;red_west<=1;
       yellow_north<=0;yellow_west<=0;
       green_north<=0;green_west<=0;

       red_north_one<=1;yellow_north_one<=0;green_north_one<=0;
       red_east_one<=1;yellow_east_one<=0;green_east_one<=0;

                   if(count_west>=0&&count_west<=2)
                        begin
                        red_west<=1;yellow_west<=0;green_west<=0;

red_east_one<=1;yellow_east_one<=0;green_east_one<=0;
                        end
                   else
                        temp<=1'bz;
                   if(count_west>=2&&count_west<=4)
                        begin
                        red_west<=0;yellow_west<=1;green_west<=0;

red_east_one<=0;yellow_east_one<=1;green_east_one<=0;
                        end
                   else
                        temp<=1'bz;
                  if(count_west>=4 && count_west<=11)
                        begin
                        green_west<=1;yellow_west<=0;red_west<=0;

red_east_one<=0;yellow_east_one<=0;green_east_one<=1;
                        end
                        else
                              temp<=1'bz;
```

```
                        if(count_west>=11 && count_west<=13)
                              begin
                              green_west<=0;yellow_west<=1;red_west<=0;

      red_east_one<=0;yellow_east_one<=1;green_east_one<=0;
                              end
                        else
                              temp<=1'bz;
                        if(count_west>=13 && count_west<=15)
                              begin
                              yellow_west<=0;red_west<=1;green_west<=0;

      red_east_one<=1;yellow_east_one<=0;green_east_one<=0;
                              end
                        else
                              temp<=1'bz;

                        if(red_north &&
(pedestrain_north_input||pedestrain_south_input) && green_west)
      //pedestrain crossing
                              begin
                              pedestrain_north<=1;
                              pedestrain_south<=1;
                              end
                        else
                              begin
                              pedestrain_north<=0;
                              pedestrain_south<=0;
                              end

                  if(red_north_one &&
(pedestrain_north_one_input||pedestrain_south_one_input) && green_east_one)
      //pedestrain crossing
                              begin
                              pedestrain_north_one<=1;
                              pedestrain_south_one<=1;
                              end
                        else
                              begin
                              pedestrain_north_one<=0;
                              pedestrain_south_one<=0;
                              end



            end

      endcase
      end
end

endmodule
```

Appendix B
Reports EDA Tools

1.Synthesis Report

```
=========================================================================
*                      Synthesis Options Summary                        *
=========================================================================
---- Source Parameters
Input File Name                    : "traffic_contoller.prj"
Ignore Synthesis Constraint File   : NO

---- Target Parameters
Output File Name                   : "traffic_contoller"
Output Format                      : NGC
Target Device                      : xc6slx9-2-csg324

---- Source Options
Top Module Name                    : traffic_contoller
Automatic FSM Extraction           : YES
FSM Encoding Algorithm             : Auto
Safe Implementation                : No
FSM Style                          : LUT
RAM Extraction                     : Yes
RAM Style                          : Auto
ROM Extraction                     : Yes
Shift Register Extraction          : YES
ROM Style                          : Auto
Resource Sharing                   : YES
Asynchronous To Synchronous        : NO
Shift Register Minimum Size        : 2
Use DSP Block                      : Auto
Automatic Register Balancing       : No

---- Target Options
LUT Combining                      : Auto
Reduce Control Sets                : Auto
Add IO Buffers                     : YES
Global Maximum Fanout              : 100000
Add Generic Clock Buffer(BUFG)     : 16
Register Duplication               : YES
Optimize Instantiated Primitives   : NO
Use Clock Enable                   : Auto
Use Synchronous Set                : Auto
Use Synchronous Reset              : Auto
Pack IO Registers into IOBs        : Auto
Equivalent register Removal        : YES

---- General Options
Optimization Goal                  : Speed
Optimization Effort                : 1
Power Reduction                    : NO
Keep Hierarchy                     : No
```

```
Netlist Hierarchy                   : As_Optimized
RTL Output                          : Yes
Global Optimization                 : AllClockNets
Read Cores                          : YES
Write Timing Constraints            : NO
Cross Clock Analysis                : NO
Hierarchy Separator                 : /
Bus Delimiter                       : <>
Case Specifier                      : Maintain
Slice Utilization Ratio             : 100
BRAM Utilization Ratio              : 100
DSP48 Utilization Ratio             : 100
Auto BRAM Packing                   : NO
Slice Utilization Ratio Delta       : 5

=========================================================================


=========================================================================
*                            HDL Parsing                                *
=========================================================================
Analyzing Verilog file "C:\Users\Pc\Desktop\traffic controller FPGA\traffic
controller FPGA\FSM.v" into library work
Parsing module <traffic_contoller>.

=========================================================================
*                             HDL Elaboration                           *
=========================================================================

Elaborating module <traffic_contoller>.
WARNING:HDLCompiler:1127 - "C:\Users\Pc\Desktop\traffic controller
FPGA\traffic controller FPGA\FSM.v" Line 107: Assignment to temp ignored,
since the identifier is never used

=========================================================================
*                             HDL Synthesis                             *
=========================================================================

Synthesizing Unit <traffic_contoller>.
    Related source file is "C:\Users\Pc\Desktop\traffic controller
FPGA\traffic controller FPGA\FSM.v".
        state_north = 1'b0
        state_west = 1'b1
    Register <green_east_one> equivalent to <green_west> has been removed
    Register <yellow_east_one> equivalent to <yellow_west> has been removed
    Register <red_east_one> equivalent to <red_west> has been removed
    Found 1-bit register for signal <clock_1hz>.
    Found 4-bit register for signal <count_north>.
    Found 4-bit register for signal <count_west>.
    Found 3-bit register for signal <count_north_one>.
    Found 1-bit register for signal <ps>.
    Found 1-bit register for signal <red_north>.
    Found 1-bit register for signal <yellow_north>.
    Found 1-bit register for signal <green_north>.
    Found 1-bit register for signal <red_west>.
    Found 1-bit register for signal <yellow_west>.
    Found 1-bit register for signal <green_west>.
```

```
    Found 1-bit register for signal <red_north_one>.
    Found 1-bit register for signal <yellow_north_one>.
    Found 1-bit register for signal <green_north_one>.
    Found 1-bit register for signal <pedestrain_north>.
    Found 1-bit register for signal <pedestrain_south>.
    Found 1-bit register for signal <pedestrain_east>.
    Found 1-bit register for signal <pedestrain_west>.
    Found 1-bit register for signal <pedestrain_north_one>.
    Found 1-bit register for signal <pedestrain_south_one>.
    Found 1-bit register for signal <pedestrain_east_one>.
    Found 1-bit register for signal <pedestrain_west_one>.
    Found 36-bit register for signal <count>.
    Found 36-bit adder for signal <count[35]_GND_1_o_add_2_OUT> created at
line 35.
    Found 4-bit adder for signal <count_north[3]_GND_1_o_add_9_OUT> created
at line 59.
    Found 3-bit adder for signal <count_north_one[2]_GND_1_o_add_12_OUT>
created at line 61.
    Found 4-bit adder for signal <count_west[3]_GND_1_o_add_14_OUT> created
at line 68.
    Found 4-bit comparator lessequal for signal <n0032> created at line 141
    Found 4-bit comparator lessequal for signal <n0034> created at line 141
    Found 4-bit comparator lessequal for signal <n0038> created at line 147
    Found 4-bit comparator lessequal for signal <n0040> created at line 147
    Found 4-bit comparator lessequal for signal <n0045> created at line 153
    Found 4-bit comparator lessequal for signal <n0047> created at line 153
    Found 4-bit comparator greater for signal <n0053> created at line 159
    Found 4-bit comparator lessequal for signal <n0064> created at line 193
    Found 4-bit comparator lessequal for signal <n0066> created at line 193
    Found 3-bit comparator greater for signal
<count_north_one[2]_GND_1_o_LessThan_42_o> created at line 195
    Found 3-bit comparator greater for signal
<count_north_one[2]_PWR_1_o_LessThan_44_o> created at line 202
    Found 3-bit comparator greater for signal
<count_north_one[2]_PWR_1_o_LessThan_46_o> created at line 209
    Found 4-bit comparator lessequal for signal <n0087> created at line 239
    Found 4-bit comparator lessequal for signal <n0089> created at line 239
    Found 4-bit comparator lessequal for signal <n0093> created at line 246
    Found 4-bit comparator lessequal for signal <n0095> created at line 246
    Found 4-bit comparator lessequal for signal <n0100> created at line 253
    Found 4-bit comparator lessequal for signal <n0102> created at line 253
    Found 4-bit comparator greater for signal <n0108> created at line 260
    Summary:
        inferred   4 Adder/Subtractor(s).
        inferred  66 D-type flip-flop(s).
        inferred  19 Comparator(s).
        inferred  25 Multiplexer(s).
Unit <traffic_contoller> synthesized.

=========================================================================
HDL Synthesis Report

Macro Statistics
# Adders/Subtractors                                   : 4
 3-bit adder                                           : 1
 36-bit adder                                          : 1
 4-bit adder                                           : 2
```

```
# Registers                                             : 23
 1-bit register                                        : 19
 3-bit register                                        : 1
 36-bit register                                       : 1
 4-bit register                                        : 2
# Comparators                                           : 19
 3-bit comparator greater                              : 3
 4-bit comparator greater                              : 2
 4-bit comparator lessequal                            : 14
# Multiplexers                                          : 25
 1-bit 2-to-1 multiplexer                              : 25

=========================================================================

=========================================================================
*                      Advanced HDL Synthesis                           *
=========================================================================


Synthesizing (advanced) Unit <traffic_contoller>.
The following registers are absorbed into counter <count>: 1 register on
signal <count>.
The following registers are absorbed into counter <count_north>: 1 register
on signal <count_north>.
The following registers are absorbed into counter <count_west>: 1 register on
signal <count_west>.
The following registers are absorbed into counter <count_north_one>: 1
register on signal <count_north_one>.
Unit <traffic_contoller> synthesized (advanced).

=========================================================================
Advanced HDL Synthesis Report

Macro Statistics
# Counters                                             : 4
 3-bit up counter                                      : 1
 36-bit up counter                                     : 1
 4-bit up counter                                      : 2
# Registers                                            : 19
 Flip-Flops                                            : 19
# Comparators                                          : 19
 3-bit comparator greater                              : 3
 4-bit comparator greater                              : 2
 4-bit comparator lessequal                            : 14
# Multiplexers                                         : 25
 1-bit 2-to-1 multiplexer                              : 25
```

2.Timing report

```
==========================================================================
Timing Report

NOTE: THESE TIMING NUMBERS ARE ONLY A SYNTHESIS ESTIMATE.
      FOR ACCURATE TIMING INFORMATION PLEASE REFER TO THE TRACE REPORT
      GENERATED AFTER PLACE-and-ROUTE.

Clock Information:
------------------
-----------------------------------+------------------------+-------+
Clock Signal                       | Clock buffer(FF name)  | Load  |
-----------------------------------+------------------------+-------+
clock_1hz                          | BUFG                   | 25    |
clock                              | BUFGP                  | 28    |
-----------------------------------+------------------------+-------+

Asynchronous Control Signals Information:
----------------------------------------
No asynchronous control signals found in this design

Timing Summary:
---------------
Speed Grade: -2

   Minimum period: 4.733ns (Maximum Frequency: 211.282MHz)
   Minimum input arrival time before clock: 5.107ns
   Maximum output required time after clock: 4.240ns
   Maximum combinational path delay: No path found

Timing Details:
---------------
All values displayed in nanoseconds (ns)

==========================================================================
Timing constraint: Default period analysis for Clock 'clock_1hz'
  Clock period: 4.733ns (frequency: 211.282MHz)
  Total number of paths / destination ports: 227 / 53
--------------------------------------------------------------------------
Delay:               4.733ns (Levels of Logic = 2)
  Source:            ps (FF)
  Destination:       count_north_one_1 (FF)
  Source Clock:      clock_1hz rising
  Destination Clock: clock_1hz rising

  Data Path: ps to count_north_one_1
                                Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     FDR:C->Q             18   0.525   1.690  ps (ps)
     LUT6:I0->O            1   0.254   0.790  Mcount_count_north_one_val_F
(N8)
     LUT3:I1->O            3   0.250   0.765  Mcount_count_north_one_val1
(Mcount_count_north_one_val)
     FDRE:R                    0.459          count_north_one_0
    ----------------------------------------
```

```
    Total                      4.733ns (1.488ns logic, 3.245ns route)
                                       (31.4% logic, 68.6% route)

=========================================================================
Timing constraint: Default period analysis for Clock 'clock'
  Clock period: 4.711ns (frequency: 212.269MHz)
  Total number of paths / destination ports: 1135 / 28
-------------------------------------------------------------------------
Delay:               4.711ns (Levels of Logic = 3)
  Source:            count_7 (FF)
  Destination:       count_0 (FF)
  Source Clock:      clock rising
  Destination Clock: clock rising

  Data Path: count_7 to count_0
                              Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     FD:C->Q               2   0.525   1.181  count_7 (count_7)
     LUT6:I0->O            3   0.254   1.042
count[35]_GND_1_o_equal_2_o<35>4 (count[35]_GND_1_o_equal_2_o<35>3)
     LUT6:I2->O           14   0.254   1.127  Mcount_count_val361
(Mcount_count_val)
     LUT2:I1->O            1   0.254   0.000  count_0_rstpot (count_0_rstpot)
     FD:D                      0.074          count_0
    ----------------------------------------
    Total                      4.711ns (1.361ns logic, 3.350ns route)
                                       (28.9% logic, 71.1% route)

=========================================================================
Timing constraint: Default OFFSET IN BEFORE for Clock 'clock_1hz'
  Total number of paths / destination ports: 36 / 29
-------------------------------------------------------------------------
Offset:              5.107ns (Levels of Logic = 3)
  Source:            reset (PAD)
  Destination:       count_north_one_1 (FF)
  Destination Clock: clock_1hz rising

  Data Path: reset to count_north_one_1
                              Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     IBUF:I->O            19   1.328   1.261  reset_IBUF (reset_IBUF)
     LUT6:I5->O            1   0.254   0.790  Mcount_count_north_one_val_F
(N8)
     LUT3:I1->O            3   0.250   0.765  Mcount_count_north_one_val1
(Mcount_count_north_one_val)
     FDRE:R                    0.459          count_north_one_0
    ----------------------------------------
    Total                      5.107ns (2.291ns logic, 2.816ns route)
                                       (44.9% logic, 55.1% route)

=========================================================================
Timing constraint: Default OFFSET IN BEFORE for Clock 'clock'
  Total number of paths / destination ports: 28 / 28
-------------------------------------------------------------------------
Offset:              4.753ns (Levels of Logic = 3)
```

```
  Source:            reset (PAD)
  Destination:       count_0 (FF)
  Destination Clock: clock rising

  Data Path: reset to count_0
                                Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     IBUF:I->O            19   1.328   1.716  reset_IBUF (reset_IBUF)
     LUT6:I0->O           14   0.254   1.127  Mcount_count_val361
(Mcount_count_val)
     LUT2:I1->O            1   0.254   0.000  count_0_rstpot (count_0_rstpot)
     FD:D                      0.074          count_0
    ----------------------------------------
    Total                      4.753ns (1.910ns logic, 2.843ns route)
                                       (40.2% logic, 59.8% route)

==========================================================================
Timing constraint: Default OFFSET OUT AFTER for Clock 'clock_1hz'
  Total number of paths / destination ports: 20 / 20
--------------------------------------------------------------------------
Offset:              4.240ns (Levels of Logic = 1)
  Source:            red_west (FF)
  Destination:       red_west (PAD)
  Source Clock:      clock_1hz rising

  Data Path: red_west to red_west
                                Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     FDR:C->Q             4   0.525   0.803  red_west (red_east_one_OBUF)
     OBUF:I->O                 2.912          red_west_OBUF (red_west)
    ----------------------------------------
    Total                      4.240ns (3.437ns logic, 0.803ns route)
                                       (81.1% logic, 18.9% route)

==========================================================================

Cross Clock Domains Report:
--------------------------

Clock to Setup on destination clock clock
---------------+---------+---------+---------+---------+
               | Src:Rise| Src:Fall| Src:Rise| Src:Fall|
Source Clock   |Dest:Rise|Dest:Rise|Dest:Fall|Dest:Fall|
---------------+---------+---------+---------+---------+
clock          |    4.711|         |         |         |
---------------+---------+---------+---------+---------+

Clock to Setup on destination clock clock_1hz
---------------+---------+---------+---------+---------+
               | Src:Rise| Src:Fall| Src:Rise| Src:Fall|
Source Clock   |Dest:Rise|Dest:Rise|Dest:Fall|Dest:Fall|
---------------+---------+---------+---------+---------+
clock_1hz      |    4.733|         |         |         |
---------------+---------+---------+---------+---------+
```

3. Device Utilization summary

Device Utilization Summary				[-]
Slice Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Registers	53	11,440	1%	
Number used as Flip Flops	53			
Number used as Latches	0			
Number used as Latch-thrus	0			
Number used as AND/OR logics	0			
Number of Slice LUTs	88	5,720	1%	
Number used as logic	87	5,720	1%	
Number using O6 output only	53			
Number using O5 output only	25			
Number using O5 and O6	9			
Number used as ROM	0			
Number used as Memory	0	1,440	0%	
Number used exclusively as route-thrus	1			
Number with same-slice register load	0			
Number with same-slice carry load	1			
Number with other load	0			
Number of occupied Slices	31	1,430	2%	
Number of MUXCYs used	28	2,860	1%	
Number of LUT Flip Flop pairs used	88			
Number with an unused Flip Flop	41	88	46%	
Number with an unused LUT	0	88	0%	
Number of fully used LUT-FF pairs	47	88	53%	
Number of unique control sets	9			
Number of slice register sites lost to control set restrictions	43	11,440	1%	
Number of bonded IOBs	30	200	15%	
Number of LOCed IOBs	8	30	26%	
Number of RAMB16BWERs	0	32	0%	

Number of RAMB8BWERs	0	64	0%	
Number of BUFIO2/BUFIO2_2CLKs	0	32	0%	
Number of BUFIO2FB/BUFIO2FB_2CLKs	0	32	0%	
Number of BUFG/BUFGMUXs	2	16	12%	
Number used as BUFGs	2			
Number used as BUFGMUX	0			
Number of DCM/DCM_CLKGENs	0	4	0%	
Number of ILOGIC2/ISERDES2s	0	200	0%	
Number of IODELAY2/IODRP2/IODRP2_MCBs	0	200	0%	
Number of OLOGIC2/OSERDES2s	0	200	0%	
Number of BSCANs	0	4	0%	
Number of BUFHs	0	128	0%	
Number of BUFPLLs	0	8	0%	
Number of BUFPLL_MCBs	0	4	0%	
Number of DSP48A1s	0	16	0%	
Number of ICAPs	0	1	0%	
Number of MCBs	0	2	0%	
Number of PCILOGICSEs	0	2	0%	
Number of PLL_ADVs	0	2	0%	
Number of PMVs	0	1	0%	
Number of STARTUPs	0	1	0%	
Number of SUSPEND_SYNCs	0	1	0%	
Average Fanout of Non-Clock Nets	2.60			