**Raed Hassan**

**102262488**

**Traffic Intersection Report**

**June 9, 2020**

**Project description**

The goal of this project is to design a control circuit for a T\_intersection by hardware description language (VHDL). The East-West road on the top of the T is a priority road and other road and pedestrian are forbidden until any buttons pushed.

**Description of control circuit performance**

The control circuit has three inputs, including two pedestrian buttons and a car detection sensor. The east-west road is always allowed to pass until one of the entrances turns on. When a circuit input is turned on, the lights turn on and off properly.

1-If a Car is detected, then the traffic lights will cycle to Green for the side road for an interval.

2-If a North-South Pedestrian is detected, then the lights will cycle so that the appropriate Walk light is displayed for a short interval.

3-If an East-West Pedestrian is detected, then the appropriate walk light will be displayed for a short interval. In each of the above cases, it is assumed that the east-west road is always green.

To implement this circuit, we used a top module that has button inputs and a car sensor and reset and clock. The output consist of two signals that sets lights.   
To prevent using latches we use 3 signal in a clock sensitive process structure then synthesis tool locate flip-flop instead of latches in the circuit. There are 2 main component for timer and traffic state machine. This components added to design as separately modules. *Counter\_top* and *Traffic\_state\_machine*. *Amber\_time*, *Car\_time* and *ped\_time*. We set this to 10 for simulation but in the real world they are longer and different. It has Mealy-style outputs and asynchrony reset. Another module *Traffic\_state\_machine*. This module has input and output *below.*Inputs includes reset, clock, *car\_time, ped\_time, amber\_time, CarNS, PedEW, and PedNS* .Outputs includes *LightsEW*, *LightsNS,* and a led to debug. This module has a process called *ped\_button\_memory* to save buttons. And a process for debugging. It shows present state encoding in *leds*.

The main process called *comb\_traffic* and contain moor state machine. Sensitive list of process consist of state, CarNS*, PedEW\_temp, pedNS\_temp, car\_time, ped\_time, amber\_time* signals.

**Simulation output waveform**



**State Machine Diagram**



**Block diagram of top-level structure**





**VHDL sources : top module**

**library** IEEE**;**

**use** IEEE**.**STD\_LOGIC\_1164**.all;**

**use** IEEE**.**STD\_LOGIC\_ARITH**.all;**

**use** IEEE**.**STD\_LOGIC\_UNSIGNED**.all;**

**entity** Traffic **is**

**port**

**(**

--asycnron rest

Reset **:** **in** STD\_LOGIC**;**

Clock **:** **in** STD\_LOGIC**;**

-- 2 signals for debuge

debugLED **:** **out** std\_logic\_vector**(**3 **downto** 0**);**

LEDs **:** **out** std\_logic\_vector**(**2 **downto** 0**);**

-- Car and pedestrian buttons

CarNS **:** **in** STD\_LOGIC**;** -- Car on NS road

PedEW **:** **in** STD\_LOGIC**;** -- Pedestrian moving EW (crossing NS road)

PedNS **:** **in** STD\_LOGIC**;** -- Pedestrian moving NS (crossing EW road)

-- Light control

LightsEW **:** **out** STD\_LOGIC\_VECTOR **(**1 **downto** 0**);** -- controls EW lights

LightsNS **:** **out** STD\_LOGIC\_VECTOR **(**1 **downto** 0**)** -- controls NS lights

**);**

**end** **entity;**

**architecture** Behavioral **of** Traffic **is**

**signal** clear **:** std\_logic**;**

-- signals that will be connected between state machine and counter modules

-- When the counter reaches the target value, it becomes signal 1.

**signal** car\_time **:** std\_logic**;**

**signal** ped\_time **:** std\_logic**;**

**signal** amber\_time **:** std\_logic**;**

-- signals for inputs after the flip flop is used

**signal** CarNS\_sync **:** STD\_LOGIC**;**

**signal** PedEW\_sync **:** STD\_LOGIC**;**

**signal** PedNS\_sync **:** STD\_LOGIC**;**

**begin**

--Observe the timer status

debugLED**(**3**)** **<=** amber\_time**;**

debugLED**(**2**)** **<=** car\_time**;**

debugLED**(**1**)** **<=** ped\_time**;**

synchroniser**:**

**process** **(**reset**,** clock**)**

**begin**

--reset everything to start at 0 initially

**if** **(**reset **=** '1'**)** **then**

PedEW\_sync **<=** '0'**;**

PedNS\_sync **<=** '0'**;**

CarNS\_sync **<=** '0'**;**

**elsif** **rising\_edge(**clock**)** **then**

-- each rising edge of the clock needs to make sure that all inputs are synchronised together so we are assigning whenever an input was held or pressed to a flip flop that saves it and updates it every rising edge of the clock

-- PedEW is the original input while PedEW\_sync is the signal pedEW being assigned to a flip flop for sync

PedEW\_sync **<=** PedEW**;**

PedNS\_sync **<=** PedNS**;**

CarNS\_sync **<=** CarNS**;**

**end** **if;**

**end** **process** synchroniser**;**

-- Show reset status on FPGA LED

debugLed**(**0**)** **<=** Reset**;**

--instance counter moudole

theCounter**:**

**entity** work**.**counter\_top

**port** **map(**

reset **=>** reset**,**

clock **=>** clock**,**

clear **=>** clear**,**

car\_time **=>** car\_time**,**

ped\_time **=>** ped\_time**,**

amber\_time **=>** amber\_time**);**

State\_machine**:**

**entity** work**.**Traffic\_state\_machine

**port** **map(**

Reset **=>** Reset**,** -- assigning ports from state machine to its equivalent in the top\_level module as these will be part of both the counter and state machine

Clock **=>** Clock**,**

clear **=>** clear**,**

car\_time **=>** car\_time**,**

ped\_time **=>** ped\_time**,**

amber\_time **=>** amber\_time**,**

CarNS **=>** CarNS\_sync**,**

PedEW **=>** PedEW\_sync**,**

PedNS **=>** PedNS\_sync**,**

-- Light control

LightsEW **=>** LightsEW**,**

LightsNS **=>** LightsNS**,**

LEDs **=>** LEDs**);**

**end** **architecture;**

**VHDL sources : counter**

**library** IEEE**;**

**use** IEEE**.**STD\_LOGIC\_1164**.ALL;**

-- counter

**entity** counter\_top **is**

**port(**

reset **:** **in** std\_logic**;**

clock **:** **in** std\_logic**;**

clear **:** **in** std\_logic**;**

car\_time **:** **out** std\_logic**;**

ped\_time **:** **out** std\_logic**;**

amber\_time **:** **out** std\_logic**);**

**end** **entity;**

**architecture** beh **of** counter\_top **is**

**signal** count **:** natural **range** 0 **to** 100**;** --

**constant** car\_time\_const **:** natural **:=** 10**;**

**constant** ped\_time\_const **:** natural **:=** 10**;**

**constant** amber\_time\_const **:** natural **:=** 10**;**

**begin**

**process(**reset**,** clock**)**

**begin**

**if** **(**reset **=** '1'**)** **then**

count **<=** 0**;**

**elsif** **rising\_edge(**clock**)** **then** -- if clear is detected due to traffic light time finishing then we also clear the counter (reseting it)

**if** **(**clear **=** '1'**)** **then**

count **<=** 0**;**

**else**

count **<=** count **+** 1**;** -- if the time limit is not reached we will count up untill we reach and clear the time

**end** **if;**

**end** **if;**

**end** **process;**

Car\_time **<=** '1' **when** **(**count **=** car\_time\_const**)** **else** '0'**;** -- continue counting the counter up until counter = car\_time\_const

Ped\_time **<=** '1' **when** **(**count **=** ped\_time\_const**)** **else** '0'**;** -- continue counting the counter up until counter = ped\_time\_const

Amber\_time **<=** '1' **when** **(**count **=** amber\_time\_const**)** **else** '0'**;** -- continue counting the counter up until counter = amber\_time\_const

**end** **architecture;**

**VHDL sources : Traffic\_state\_machine**

**library** IEEE**;**

**use** IEEE**.**STD\_LOGIC\_1164**.all;**

**use** IEEE**.**STD\_LOGIC\_ARITH**.all;**

**use** IEEE**.**STD\_LOGIC\_UNSIGNED**.all;**

**entity** Traffic\_state\_machine **is**

**port**

**(**

Reset **:** **in** STD\_LOGIC**;**

Clock **:** **in** STD\_LOGIC**;**

Clear **:** **out** STD\_LOGIC**;**

car\_time **:** **in** std\_logic**;**

ped\_time **:** **in** std\_logic**;**

amber\_time **:** **in** std\_logic**;**

-- Car and pedestrian buttons

CarNS **:** **in** STD\_LOGIC**;** -- Car on NS road

PedEW **:** **in** STD\_LOGIC**;** -- Pedestrian moving EW (crossing NS road)

PedNS **:** **in** STD\_LOGIC**;** -- Pedestrian moving NS (crossing EW road)

-- Light control

LightsEW **:** **out** STD\_LOGIC\_VECTOR **(**1 **downto** 0**);** -- controls EW lights

LightsNS **:** **out** STD\_LOGIC\_VECTOR **(**1 **downto** 0**);** -- controls NS lights

--leds for debuge

LEDs **:** **out** std\_logic\_vector**(**2 **downto** 0**)**

**);**

**end** Traffic\_state\_machine**;**

**architecture** Behavioral **of** Traffic\_state\_machine **is**

-- Encoding for lights

**constant** RED **:** std\_logic\_vector**(**1 **downto** 0**)** **:=** "00"**;**

**constant** AMBER **:** std\_logic\_vector**(**1 **downto** 0**)** **:=** "01"**;**

**constant** GREEN **:** std\_logic\_vector**(**1 **downto** 0**)** **:=** "10"**;**

**constant** WALK **:** std\_logic\_vector**(**1 **downto** 0**)** **:=** "11"**;**

**type** state\_type **is** **(**EW\_Green**,** EW\_Amber**,** NS\_Green**,** NS\_Amber**,** NS\_walk**,** EW\_Walk**);**

**signal** state**,** next\_state **:** state\_type**;**

**signal** pedEW\_temp **:** STD\_LOGIC**;** -- signals that will each connect a flip flop to the state machine, recording whether the ped utton is saved as pressed or not pressed

**signal** pedNS\_temp **:** STD\_LOGIC**;**

**signal** Clear\_ped\_NS **:** STD\_LOGIC**;** -- 2 other signals directly going from the state machine to the flip flops to record whether we need to reset

**signal** Clear\_ped\_EW **:** STD\_LOGIC**;**

**begin**

--LEDs(2) <= pedEW\_temp;

--LEDs(1) <= pedNS\_temp;

--LEDs(0) <= CarNS;

sync **:**

**process** **(**Reset**,** clock**)**

**begin**

**if** **(**Reset **=** '1'**)** **then**

state **<=** EW\_Green**;**

**elsif** **rising\_edge(**clock**)** **then**

state **<=** next\_state**;**

**end** **if;**

**end** **process;**

ped\_button\_memory **:**

**process** **(**Reset**,** clock**)**

**begin**

**if** **(**Reset **=** '1'**)** **then** --REMEMBER PEDNS AND PED EW ARE THE ORIGINAL INPUTS, THEY WILL NOT BE DIRECTLY WRITTEN IN THE STATE MACHINE

pedNS\_temp **<=** '0'**;**

pedEW\_temp **<=** '0'**;**

**elsif** **rising\_edge(**clock**)** **then**

**if** **(**PedNS **=** '1'**)** **then** -- when the ped button is pressed we save it as a '1' and assigning it to pedns\_temp until the sstate is changed due to pedestrian light stopping its light

pedNS\_temp **<=** '1'**;**

**elsif** **(**Clear\_ped\_NS **=** '1'**)** **then** -- when we need to clear the ped button press we will go back to the initial pedNS\_temp not being pressed

pedNS\_temp **<=** '0'**;**

**end** **if;**

**if** **(**PedEW **=** '1'**)** **then**

pedEW\_temp **<=** '1'**;**

**elsif** **(**Clear\_ped\_EW **=** '1'**)** **then**

PedEW\_temp **<=** '0'**;**

**end** **if;**

**end** **if;**

**end** **process;**

--Monitor the status of states

--debug

**process** **(**next\_state**)**

**begin**

**case** next\_state **is**

**when** EW\_Green **=>**

leds **<=** "000"**;**

**when** EW\_Walk **=>**

leds **<=** "001"**;**

**when** EW\_Amber **=>**

leds **<=** "010"**;**

**when** NS\_Green **=>**

leds **<=** "011"**;**

**when** NS\_walk **=>**

leds **<=** "100"**;**

**when** NS\_amber **=>**

leds **<=** "101"**;**

**when** **others** **=>**

leds **<=** "111"**;**

**end** **case;**

**end** **process;**

--main procees

comb\_traffic **:**

**process** **(**state**,** CarNS**,** PedEW\_temp**,** pedNS\_temp**,** car\_time**,** ped\_time**,** amber\_time**)**

**begin**

lightsNS **<=** Red**;** -- default traffic light state

LightsEW **<=** Red**;**

-- default time counter start

clear **<=** '0'**;**

Clear\_ped\_NS **<=** '0'**;**

clear\_ped\_EW **<=** '0'**;**

next\_state **<=** state**;**

**case** state **is**

**when** EW\_Green **=>**

LightsEW **<=** GREEN**;**

**if** **(**pedNS\_temp **=** '1' or CarNS **=** '1'**)** **then**

next\_state **<=** EW\_Amber**;**

**elsif** **(**PedEW\_temp **=** '1'**)** **then**

next\_state **<=** EW\_Walk**;**

**end** **if;**

**when** EW\_Walk **=>**

lightsEW **<=** Walk**;**

clear\_ped\_EW **<=** '1'**;** -- clearing the pedestrian button press

**if** **(**ped\_time **=** '1'**)** **then**

clear **<=** '1'**;** ---------- clear counter then go to next state

next\_state **<=** EW\_Green**;**

**end** **if;**

**when** EW\_Amber **=>**

LightsEW **<=** Amber**;**

**if** **(**amber\_time **=** '1'**)** **then** --- amber time reached so reset counter and move to next state depending on whether the ped has pressed the button or not

clear **<=** '1'**;**

**if** **(**pedNS\_temp **=** '1'**)** **then**

next\_state **<=** NS\_Walk**;**

**else**

next\_state **<=** NS\_Green**;**

**end** **if;**

**end** **if;**

**when** NS\_Green **=>**

LightsNS **<=** Green**;**

**if** **(**car\_time **=** '1'**)** **then**

clear **<=** '1'**;**-- reset counter

next\_state **<=** NS\_Amber**;**

**end** **if;**

**when** NS\_walk **=>**

LightsNS **<=** Walk**;**

Clear\_ped\_NS **<=** '1'**;** -- clear

**if** **(**ped\_time **=** '1'**)** **then**

clear **<=** '1'**;**-- reset counter

next\_state **<=** EW\_Green**;**

**end** **if;**

**when** NS\_Amber **=>**

LightsNS **<=** Amber**;**

**if** **(**amber\_time **=** '1'**)** **then**

clear **<=** '1'**;**

next\_state **<=** EW\_Green**;**

**end** **if;**

**end** **case;**

**end** **process;**

**end** **architecture;**

**Synthesis report**

Here are the important parts of the synthesis report and some explanations.

FMS optimizing with one-hot encoding. All signals saved in flip-flop and no latch insert in design. Maximum clock frequency: 194MHz and no gated clock exists is design. The important part is the color red.

=========================================================================  
\*                       Advanced HDL Synthesis                          \*  
=========================================================================  
  
Analyzing FSM for best encoding.  
Optimizing FSM on signal with **one-hot encoding.**  
----------------------  
 State    | Encoding  
----------------------  
 ew\_green | 000001  
 ew\_amber | 000010  
 ns\_green | 010000  
 ns\_amber | 100000  
 ns\_walk  | 001000  
 ew\_walk  | 000100  
----------------------  
  
=========================================================================  
Advanced HDL Synthesis Report  
  
Macro Statistics  
# FSMs                                                 : 1  
# Counters                                             : 1  
 7-bit up counter                                      : 1  
# Registers                                            : 5  
 **Flip-Flops                                            : 5**  
  
=========================================================================  
  
=========================================================================  
\*                         Low Level Synthesis                           \*  
=========================================================================  
  
Optimizing unit  ...  
  
Optimizing unit  ...  
  
Mapping all equations...  
Building and optimizing final netlist ...  
Found area constraint ratio of 100 (+ 5) on block Traffic, actual ratio is 2.  
  
Final Macro Processing ...  
  
=========================================================================  
Final Register Report  
  
**Macro Statistics  
# Registers                                            : 18  
 Flip-Flops                                            : 18**=========================================================================  
  
=========================================================================  
\*                           Partition Report                            \*  
=========================================================================  
  
Partition Implementation Status  
-------------------------------  
  
  No Partitions were found in this design.  
  
-------------------------------  
  
=========================================================================  
\*                            Final Report                               \*  
=========================================================================  
Final Results  
RTL Top Level Output File Name     : Traffic.ngr  
Top Level Output File Name         : Traffic  
Output Format                      : NGC  
Optimization Goal                  : Speed  
Keep Hierarchy                     : No  
  
Design Statistics  
# IOs                              : 16  
  
Cell Usage :  
# BELS                             : 48  
#      LUT2                        : 10  
#      LUT2\_L                      : 1  
#      LUT3                        : 4  
#      LUT3\_D                      : 2  
#      LUT3\_L                      : 2  
#      LUT4                        : 18  
#      LUT4\_D                      : 1  
#      LUT4\_L                      : 3  
#      MUXF5                       : 6  
#      VCC                         : 1  
# FlipFlops/Latches                : 18  
#      FDC                         : 12  
#      FDCE                        : 5  
#      FDP                         : 1  
# Clock Buffers                    : 1  
#      BUFGP                       : 1  
# IO Buffers                       : 15  
#      IBUF                        : 4  
#      OBUF                        : 11  
=========================================================================  
  
Device utilization summary:  
---------------------------  
  
Selected Device : 3s50pq208-5   
  
 Number of Slices:                       22  out of    768     2%    
 Number of Slice Flip Flops:             18  out of   1536     1%    
 Number of 4 input LUTs:                 41  out of   1536     2%    
 Number of IOs:                          16  
 Number of bonded IOBs:                  16  out of    124    12%    
 Number of GCLKs:                         1  out of      8    12%    
  
---------------------------  
Partition Resource Summary:  
---------------------------  
  
  No Partitions were found in this design.  
  
---------------------------  
  
  
=========================================================================  
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                              | BUFGP                  | 18    |  
-----------------------------------+------------------------+-------+  
  
Asynchronous Control Signals Information:  
----------------------------------------  
-----------------------------------+------------------------+-------+  
Control Signal                     | Buffer(FF name)        | Load  |  
-----------------------------------+------------------------+-------+  
Reset                              | IBUF                   | 18    |  
-----------------------------------+------------------------+-------+  
  
Timing Summary:  
---------------  
Speed Grade: -5  
  
   Minimum period: 5.141ns (Maximum Frequency: 194.532MHz)  
   Minimum input arrival time before clock: 1.572ns  
   Maximum output required time after clock: 11.979ns  
   Maximum combinational path delay: 6.856ns  
  
Timing Detail:  
--------------  
All values displayed in nanoseconds (ns)  
  
=========================================================================  
Timing constraint: Default period analysis for Clock 'Clock'  
  **Clock period: 5.141ns (frequency: 194.532MHz)**  
  Total number of paths / destination ports: 202 / 20  
-------------------------------------------------------------------------  
Delay:               5.141ns (Levels of Logic = 4)  
  Source:            theCount/count\_2 (FF)  
  Destination:       theCount/count\_6 (FF)  
  Source Clock:      Clock rising  
  Destination Clock: Clock rising  
  
  Data Path: theCount/count\_2 to theCount/count\_6  
                                Gate     Net  
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)  
    ----------------------------------------  ------------  
     FDC:C->Q              5   0.626   1.078  theCount/count\_2 (theCount/count\_2)  
     LUT3\_D:I0->O          5   0.479   0.806  theCount/Mcount\_count\_eqn\_321 (N01)  
     LUT4:I3->O            1   0.479   0.704  State\_machine/Clear\_SW10\_F (N46)  
     LUT4:I3->O            1   0.479   0.000  theCount/Mcount\_count\_eqn\_6\_G (N55)  
     MUXF5:I1->O           1   0.314   0.000  theCount/Mcount\_count\_eqn\_6 (theCount/Mcount\_count\_eqn\_6)  
     FDC:D                     0.176          theCount/count\_6  
    ----------------------------------------  
    Total                      5.141ns (2.553ns logic, 2.588ns route)  
                                       (49.7% logic, 50.3% route)  
  
=========================================================================  
Timing constraint: Default OFFSET IN BEFORE for Clock 'Clock'  
  Total number of paths / destination ports: 3 / 3  
-------------------------------------------------------------------------  
Offset:              1.572ns (Levels of Logic = 1)  
  Source:            PedEW (PAD)  
  Destination:       PedEW\_sync (FF)  
  Destination Clock: Clock rising  
  
  Data Path: PedEW to PedEW\_sync  
                                Gate     Net  
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)  
    ----------------------------------------  ------------  
     IBUF:I->O             1   0.715   0.681  PedEW\_IBUF (PedEW\_IBUF)  
     FDC:D                     0.176          PedEW\_sync  
    ----------------------------------------  
    Total                      1.572ns (0.891ns logic, 0.681ns route)  
                                       (56.7% logic, 43.3% route)  
  
  
=========================================================================  
Timing constraint: Default path analysis  
  Total number of paths / destination ports: 1 / 1  
-------------------------------------------------------------------------  
Delay:               6.856ns (Levels of Logic = 2)  
  Source:            Reset (PAD)  
  Destination:       debugLED<0> (PAD)  
  
  Data Path: Reset to debugLED<0>  
                                Gate     Net  
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)  
    ----------------------------------------  ------------  
     IBUF:I->O            19   0.715   1.233  Reset\_IBUF (debugLED\_0\_OBUF)  
     OBUF:I->O                 4.909          debugLED\_0\_OBUF (debugLED<0>)  
    ----------------------------------------  
    Total                      6.856ns (5.624ns logic, 1.233ns route)  
                                       (82.0% logic, 18.0% route)  
  
=========================================================================  
  
  
Total REAL time to Xst completion: 6.00 secs  
Total CPU  time to Xst completion: 5.67 secs  
   
-->   
  
Total memory usage is 256944 kilobytes  
  
Number of errors   :    0 (   0 filtered)  
Number of warnings :    0 (   0 filtered)  
Number of infos    :    0 (   0 filtered)