

Princess Nourah Bint Abdulrahman University

College of Computer and Information Sciences Computer Sciences Department

Course CS 207T

Computer Architecture

LAB PROJECT

|  |  |  |  |
| --- | --- | --- | --- |
| Section: | 4C2 | Group: | 4 |

|  |
| --- |
| No. |
| 1 |
| 2 |
| 3 |
| 4 |
| 5 |

Second Semester, 2022

|  |  |  |  |
| --- | --- | --- | --- |
| Table of Contents | | | |
| No. | Contents | | No. page |
| 1. | Cover Page | | 1 |
| 2. | Table of Contents | | 2 |
| 3. | The Project Aim | | 3 |
| 4. | The Problem Solution | | 3 |
|  | 4.1 | Inputs and Outputs | 3 |
|  | 4.2 | Solution code | 3 |
|  | 4.3 | Testbenches code | 11 |
| **5**. | Snapshots of the solution Design | | 12 |
|  | 5.1 | The code | 12 |
|  | 5.2 | The diagram | 21 |
|  | 5.3 | The simulation trace | 21 |
| **6**. | References | | 22 |

1. The Project Aim:

Our project aims to design a four-way traffic light controller for a four ways intersection. We will be using Verilog hardware description language to implement it.

1. The Problem Solution:

There are three colors for the traffic lights green, yellow, and lastly red, and they turn on in that order. However, in a multiple way traffic lights or an intersection only one way will be green, and the rest will be red. When changing from green to red, it will first change to yellow to let drivers know to slow down preparing for the red light. The traffic light order will be north, south, east, and lastly west.

Each light has a specific number of clock cycles. The duration of clock cycles of the different signal lights will be eight for the green light, and four for the yellow light. In other words, the light will change from green to yellow after eight clock cycles. Then, after four clock cycles, it will change from yellow to red.

* 1. Inputs and Outputs:

In our project design there are two input signals, which are the clock signal and the rest signal. Also, we have four output signals that are constructed of 3 bits, and they are the north light, the east light, the south light, and the west light.

The reset signal will start the traffic lights at the north light signal. The output values will appear after the reset signal’s value is zero, which is low.

The three-bit output signals will represent the green light by ‘001’, the yellow light by ‘010’, and the red light by ‘100’.

* 1. Solution code:

// there are 2 inputs signal - clock & reset - , and 3 bit 4 output signals - North\_signal,South\_signal,East\_signal,West\_signal-

module Traffic\_signal (North\_signal,South\_signal,East\_signal,West\_signal,clock,reset);

// The outputs:

output reg [2:0] North\_signal,South\_signal,East\_signal,West\_signal;

// The inputs:

input clock,reset;

//-----------------------

reg [2:0] state;

reg [2:0] count;

//-----------------------

// green & yellow (North):

parameter [2:0] Ngreen=3'b000;

parameter [2:0] Nyellow=3'b001;

// green & yellow (South):

parameter [2:0] Sgreen=3'b010;

parameter [2:0] Syellow=3'b011;

// green & yellow (East):

parameter [2:0] Egreen=3'b100;

parameter [2:0] Eyellow=3'b101;

// green & yellow (West):

parameter [2:0] Wgreen=3'b110;//west\_green

parameter [2:0] Wyellow=3'b111;//west\_yellow

//-----------------------

always @ (posedge clock, posedge reset)

begin

if (reset==1)

begin

#1 state=Ngreen;

count=3'b000;

end

else

begin

case (state)

// the North case:

// the -green-: ------ 1 ------

Ngreen :

begin

if (count==3'b111)

begin

count=3'b000;

state=Nyellow;

end

else

begin

count=count+3'b001;

state=Ngreen;

end

end

// end for the green

// the -yellow-: ------ 2 ------

Nyellow :

begin

if(count==3'b011)

begin

count=3'b000;

state=Sgreen;

end

else

begin

count=count+3'b001;

state=Nyellow;

end

end

// the end for the yellow

//-----------------------------------------

// the south case:

// the -green-: ------ 3 ------

Sgreen :

begin

if (count==3'b111)

begin

count=3'b000;

state=Syellow;

end

else

begin

count=count+3'b001;

state=Sgreen;

end

end

// end for the green

// the -yellow-: ------ 4 ------

Syellow :

begin

if(count==3'b011)

begin

count=3'b000;

state=Egreen;

end

else

begin

count=count+3'b001;

state=Syellow;

end

end

// the end for the yellow

//----------------------------------------

// the east case:

// the -green-: ------ 5 ------

Egreen :

begin

if (count==3'b111)

begin

count=3'b000;

state=Eyellow;

end

else

begin

count=count+3'b001;

state=Egreen;

end

end

// end for the green

// the -yellow-: ------ 6 ------

Eyellow :

begin

if(count==3'b011)

begin

count=3'b000;

state=Wgreen;

end

else

begin

count=count+3'b001;

state=Eyellow;

end

end

// the end for the yellow

//----------------------------------------

// the west case:

// the -green-: ------ 7 ------

Wgreen :

begin

if (count==3'b111)

begin

count=3'b000;

state=Wyellow;

end

else

begin

count=count+3'b001;

state=Wgreen;

end

end

// end for the green

// the -yellow-: ------ 8 ------

Wyellow :

begin

if(count==3'b011)

begin

count=3'b000;

state=Ngreen;

end

else

begin

count=count+3'b001;

state=Wyellow;

end

end

// the end for the yellow

//----------------------------------------

endcase

end

end

//-----------------------------------------

//-----------------------------------------

// the north:

always @(state)

begin

case (state)

Ngreen:

begin

North\_signal = 3'b001;

South\_signal = 3'b100;

East\_signal= 3'b100;

West\_signal=3'b100;

end // end for north green

Nyellow:

begin

North\_signal = 3'b010;

South\_signal = 3'b100;

East\_signal= 3'b100;

West\_signal=3'b100;

end // end for north yellow

//-----------------------------

// the south:

Sgreen:

begin

North\_signal = 3'b100;

South\_signal = 3'b001;

East\_signal = 3'b100;

West\_signal = 3'b100;

end //end south green

Syellow:

begin

North\_signal = 3'b100;

South\_signal = 3'b010;

East\_signal= 3'b100;

West\_signal=3'b100;

end // end for south yellow

//----------------------------

// the East:

Egreen:

begin

North\_signal = 3'b100;

South\_signal = 3'b100;

East\_signal = 3'b001;

West\_signal = 3'b100;

end //end east green

Eyellow:

begin

North\_signal = 3'b100;

South\_signal = 3'b100;

East\_signal= 3'b010;

West\_signal=3'b100;

end // end for east yellow

//----------------------------

// the West:

Wgreen:

begin

North\_signal = 3'b100;

South\_signal = 3'b100;

East\_signal = 3'b001;

West\_signal = 3'b100;

end //end West green

Wyellow:

begin

North\_signal = 3'b100;

South\_signal = 3'b100;

East\_signal= 3'b010;

West\_signal=3'b100;

end // end for west yellow

//----------------------------

endcase // end the cases

end // end the state

endmodule// end the program

* 1. Testbenches code:

Verilog testbenches are a verilog code which are used to simulate and produce inputs to the digital design, and it ensures that the outputs are correct and meet the timing without the need for any physical hardware.

`timescale 1ns/1ps

module Traffic\_signal\_test;

// making a branch teat to insert values initial the run time.

// we put wire for the outputs and reg for the input when the program be under the test:

wire [2:0] North\_signal,South\_signal,East\_signal,West\_signal;

reg clock,reset;

//the clock will be 1 in the set initial, and change its value every 4 time:

initial

begin

clock= 1'b0;

forever #4 clock=~clock;

end

//the reset will be 1 in the set initial, and will be zero after 3 time:

// the program will stop after 400.

initial

begin

reset =1'b1;

#4

reset=1'b0;

#400

$stop;

end

// call the original - main- module:

Traffic\_signal at (North\_signal,South\_signal,East\_signal,West\_signal,clock,reset);

endmodule

//---------------------------------------------------------------------------

//---------------------------------------------------------------------------

1. Snapshots of the solution Design:
   1. The Code:

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

* 1. صورة تحتوي على نص, لقطة شاشة, داخلي, كمبيوتر

     تم إنشاء الوصف تلقائياًThe Diagram:
  2. The simulation trace:

Graphical user interface, application, table

Description automatically generated

1. References:

* FPGA Tutorial. 2020. *How to Write a Basic Verilog Testbench - FPGA Tutorial*. [online] Available at: <https://fpgatutorial.com/how-to-write-a-basic-verilog-testbench/> .
* Verilog Hardware Description Language (Verilog HDL) Edited by Chu Yu

# The Verilog® Hardware Description Language 5th Edition by [Donald E. Thomas](https://www.amazon.com/s/ref=dp_byline_sr_book_1?ie=UTF8&field-author=Donald+E.+Thomas&text=Donald+E.+Thomas&sort=relevancerank&search-alias=books)

* *Verilog Tutorial Index - Tutorials for beginners in Verilog*. Nandland.com. (2022). Available at: https://www.nandland.com/verilog/tutorials/index.html.
* Arar, D. (2019). Getting Started with the Verilog Hardware Description Language. Available at: https://www.allaboutcircuits.com/technical-articles/getting-started-with-the-verilog-hardware-description-language/.