Traffic Light Controller

CS4362 - Hardware Description Languages

Practical Assignment

180725C – Croos I A

TABLE OF CONTENTS

FIGURES

TABLES

Task

Task is to develop a traffic light controller system to a intersection where a side street is crossing a main street. Both streets have usual traffic lights for vehicles and pedestrians.

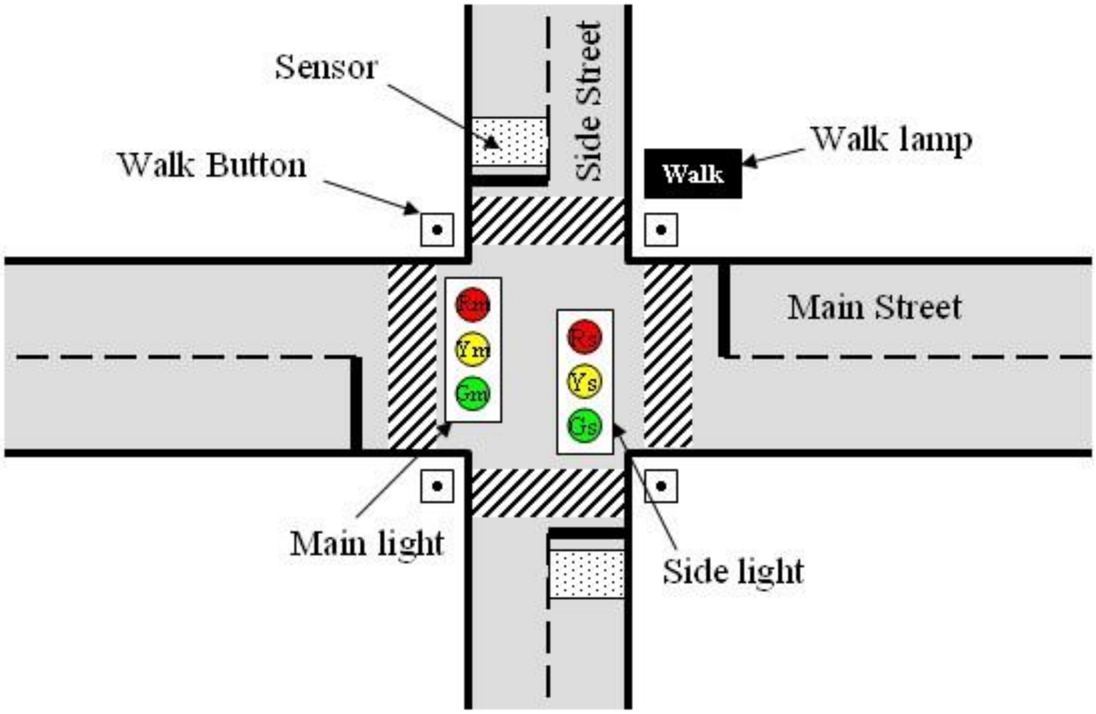

Figure 1 Diagram for intersection with corresponding lights and sensors

All the walk request buttons are attached to the controller using a wired OR. There are two sensors on the side street to detect vehicles that are passing over them. (Assumption: Sensor remains high constantly when several cars pass over it).

The Traffic light controller is timed based on three parameters (in seconds) t_{BASE}, t_{EXT}, t_{YEL}. These parameters can be changed using operation. Default values are as follows.

Interval Name	Symbol	Parameter Number	Default Time (sec)	Time Value
Base Interval	t_{BASE}	00	6	0110
Extended Interval	t_{EXT}	01	3	0011
Yellow Interval	t_{YEL}	10	2	0010

Table 1 Default timing parameters

The normal operating sequence of this intersection begins with the Main Street having a green light for tBASE seconds, followed by a yellow light for tYEL seconds, and then a red light. During this time, the Side Street traffic light turns green for tBASE seconds, with a yellow light for tYEL seconds. When one street's light is green or yellow, the other street's light is always red. This cycle is repeated continuously, unless one of the two possible deviations occur.

The first deviation is when a pedestrian presses the walk button, which triggers the internal Walk Register and the controller will interrupt its normal sequence. The traffic lights will turn red and the walk signal will turn on. After tEXT seconds, the walk signal will turn off and the Side Street will turn green. The internal Walk Register will be cleared at the end of this cycle.

The second deviation is the traffic sensor, which can detect the traffic on the street. If the sensor detects high traffic at the end of the first tBASE seconds of the Main Street green light, it will extend the green light for an additional tEXT seconds. Similarly, if the sensor detects high traffic at the end of the Side Street green light, it will also extend the green light for an additional tEXT seconds.

Design

State chart diagram

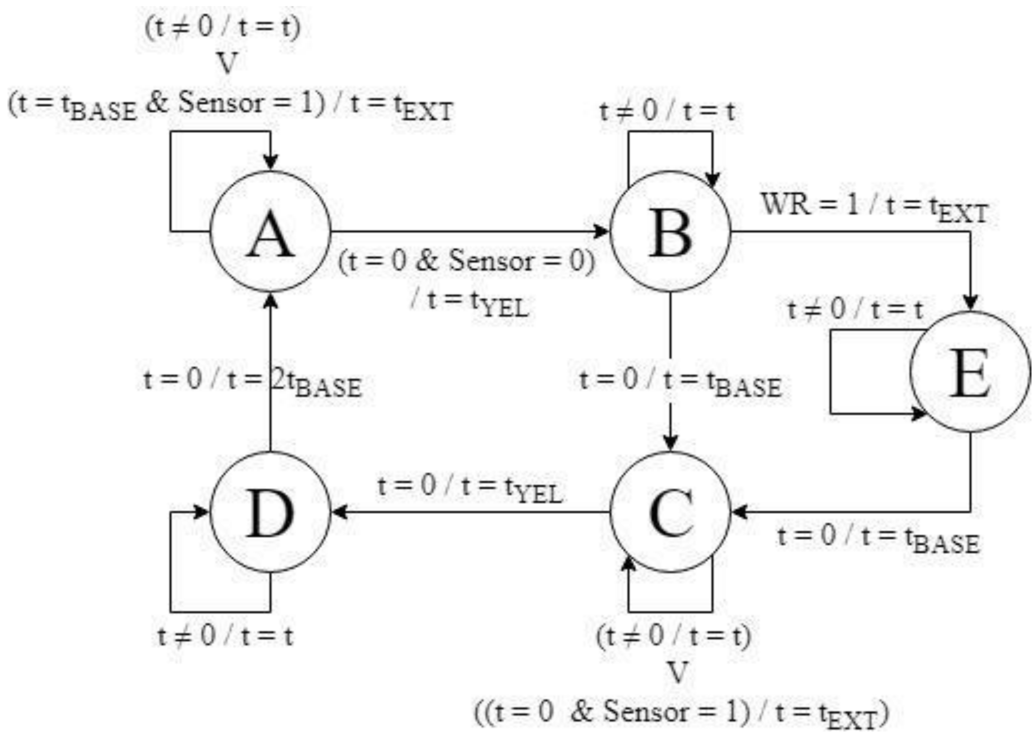

Figure 2 Finite state machine

For the convenience of the design for the state chart time is used as the parameter for the state chart. In the implementation this is replaced with expired from the timer module.

Top Module (TrafficControllerMain)

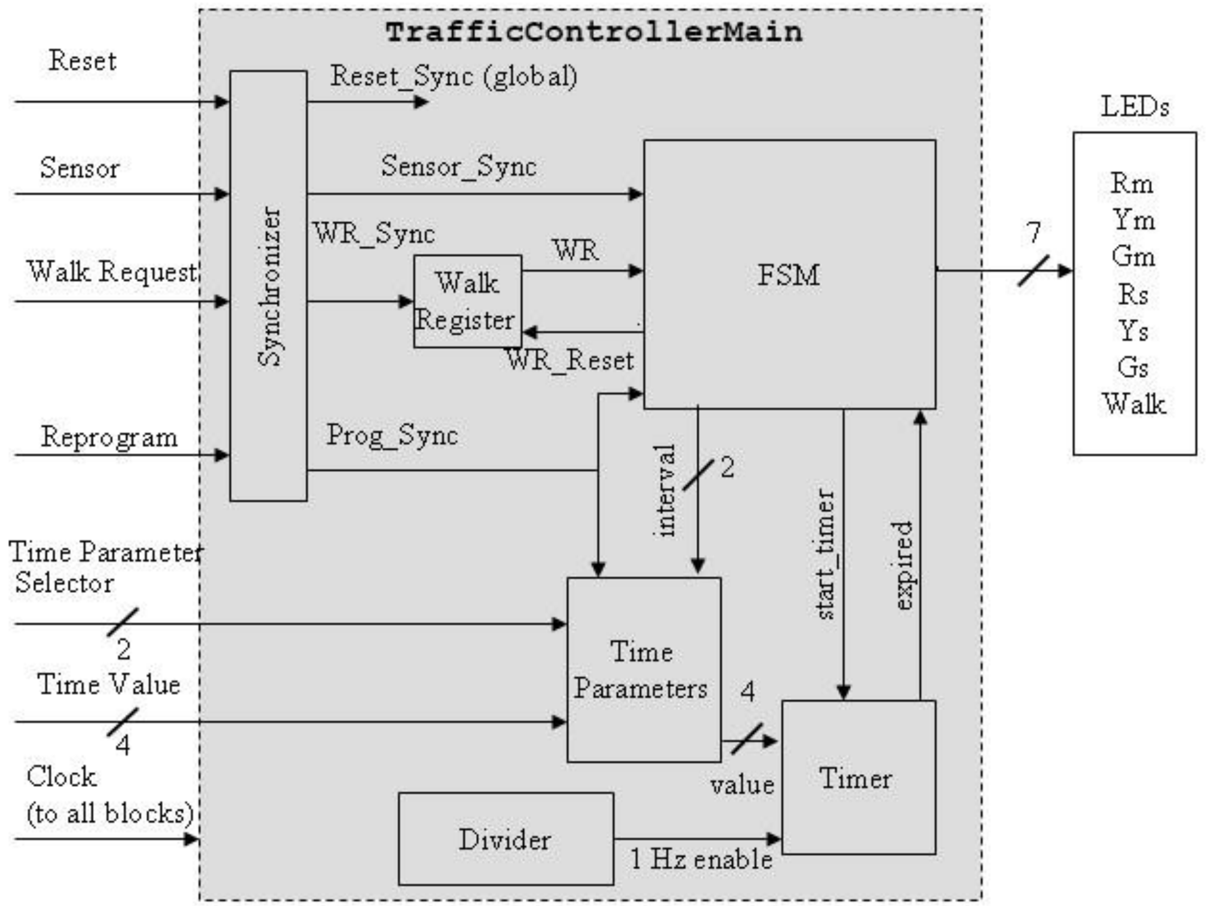

Figure 3 Top module block diagram

Variables are named as per the above diagram. 7bit vector array is used as follows for the lights. LED sequence: [Red_{main}, $Yellow_{main}$, $Green_{main}$, Red_{side}, $Yellow_{side}$, $Green_{side}$, Walk]

Integrated system Simulation

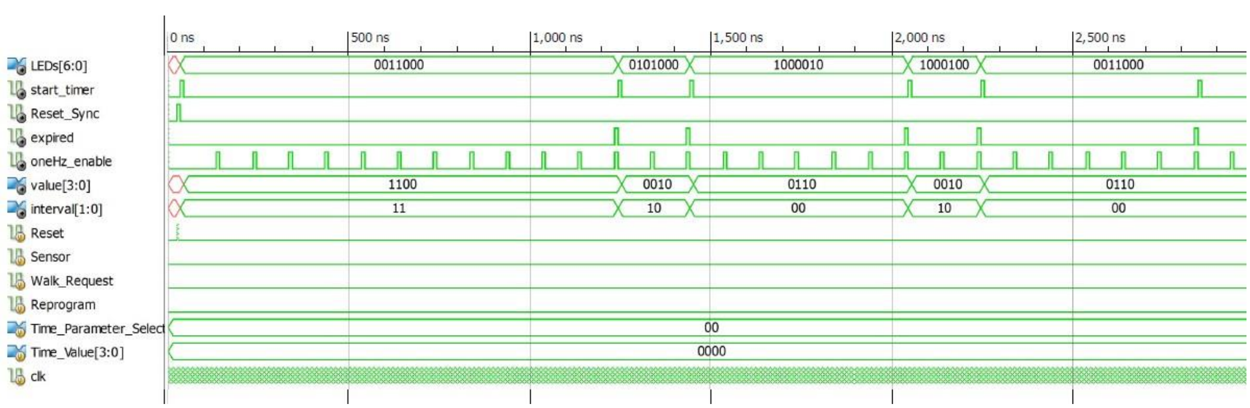

Figure 4 Integrated system simulation with internal wires (Normal routine) 10MHz timer

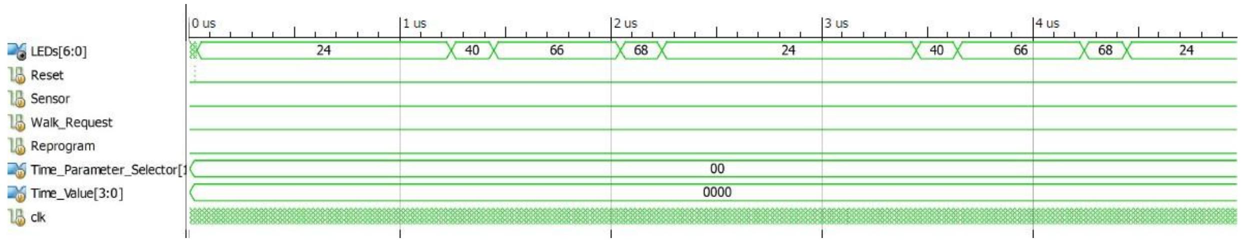

Figure 5 Full system simulation (Normal routine | timer at 10Mhz)

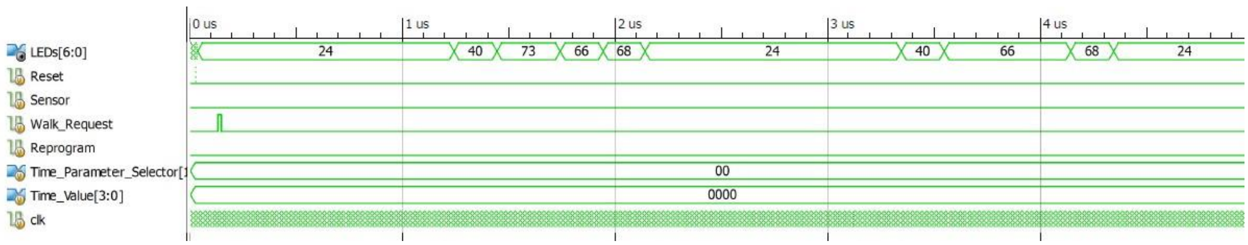

Figure 6 Full system simulation (Walk request | timer at 10Mhz)

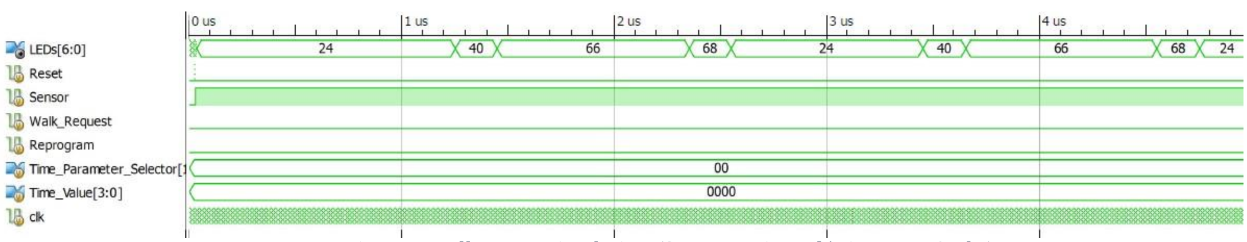

Figure 7 Full system simulation (Sensor activated | timer at 10Mhz)

States represented by decimal values

Decimal value	State	Binary value
24	A	0011000
40	B	0101000
66	C	1000010
68	D	1000100
73	E	1001001

Table 2 States represented by Decimals

Synchronizer

Module description

The purpose of the synchronizer is to ensure that the inputs are synchronized to the system clock. So, all input signals pass through the synchronizer before going to other blocks.

Module overview

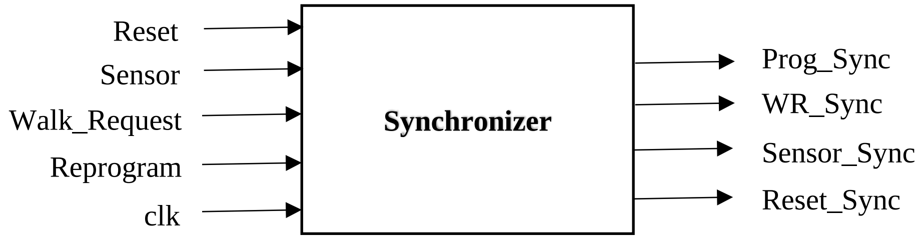

Figure 8 Synchronizer block diagram

Test bench output

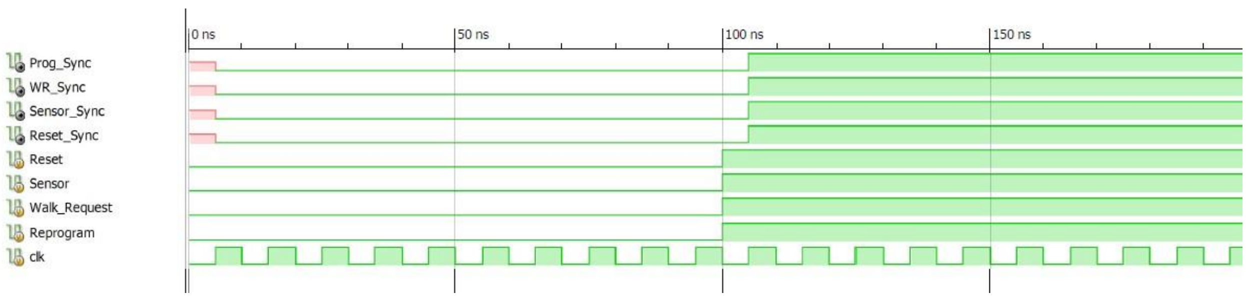

Figure 9 Synchronizer simulation

Walk Register

Module description

The Walk Register allows pedestrians to set a walk request at any time. There is a signal controlled by the finite state machine that will be also able to reset the register at the end of the actual walk cycle.

Module overview

Figure 10 Walk Register block diagram

Test bench output

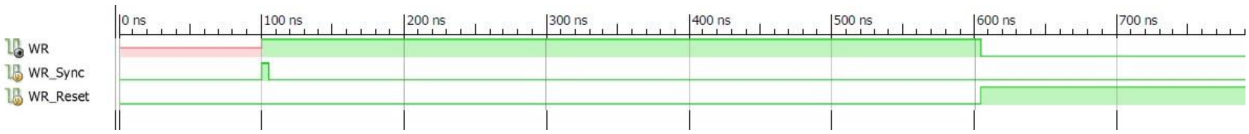

Figure 11 Walk register simulation

Time Parameters

Module description

The Time Parameters module on an FPGA is a small memory that stores three different time values, tBASE, tEXT, and tYEL. These values can be accessed and modified by the FSM and Timer blocks on the FPGA. For the user, the three time values can be modified using the Time_Parameter_Selector, Time_Value, and Reprogram inputs, which are all 4 bits and selected using a 2-bit address. On reset, the three time values are set to 6, 3, and 2 seconds, but the user can change them at any time. When a time value is reprogrammed, the FSM is reset to its starting state.

Module overview

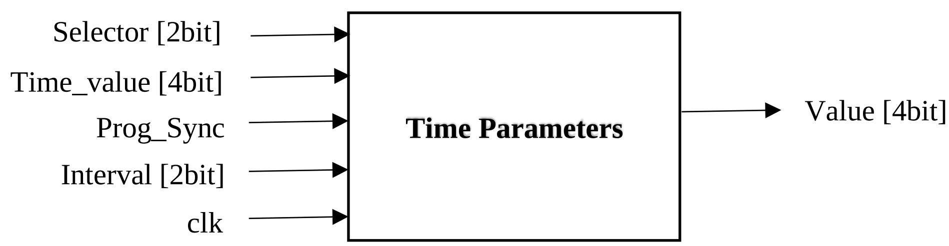

Figure 12 TimeParameters block diagram

Interval value is assigned for t_{BASE}, t_{EXT} and t_{YEL} as follows.

Interval	Symbol	Default Time (sec)
00	t_{BASE}	6
01	t_{EXT}	3
10	t_{YEL}	2
11	$2*t_{BASE}$	$2*t_{BASE}$

Table 3 Interval values

The time values are reset to their default values when the selector input is set to 00. Otherwise, the values on the time_value input are assigned to tBASE, tEXT, and tYEL, respectively. Since the module stores these values in a register, the selected values will be saved until the selector input is set to 00, at which point the values will be reset.

Test bench output

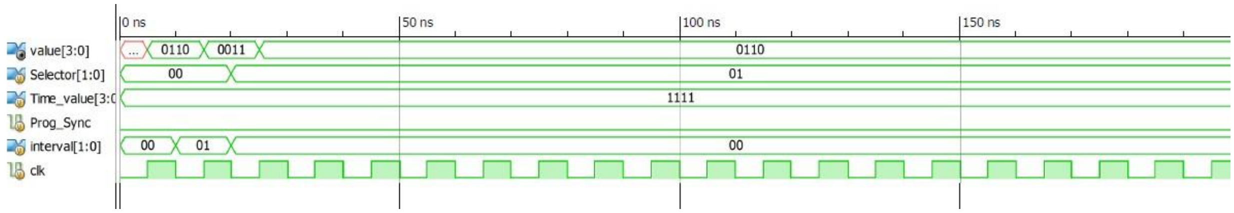

Figure 13 Time parameter simulation

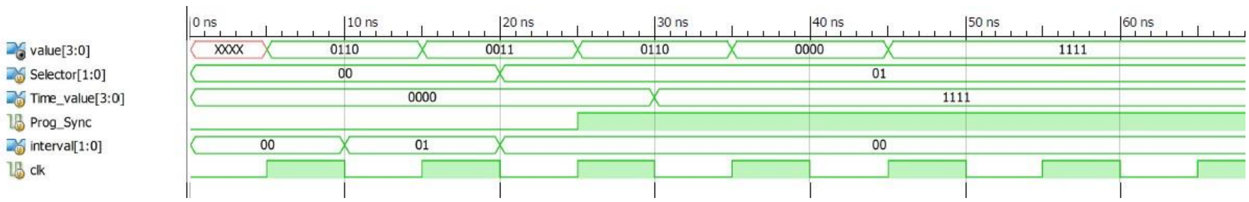

Figure 14 Changing t_{BASE} to 15 seconds

Divider

Module description

A divider module is needed for the timer to accurately measure the duration of each traffic light state. It takes in the clock signal as input and generates a 1 Hz enable signal, which is sent to the timer. This enable signal is a pulse that is high for one clock cycle every second, which allows the timer to properly measure the duration of each state.

Module overview

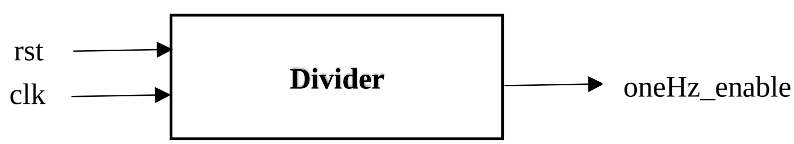

Figure 15 Divider block diagram

Test bench output

1Hz enable requires 10^6 clock cycles at 100MHz. Since it is output cannot be visualized clearly 10MHz pulses were generated instead of 1Hz pulses.

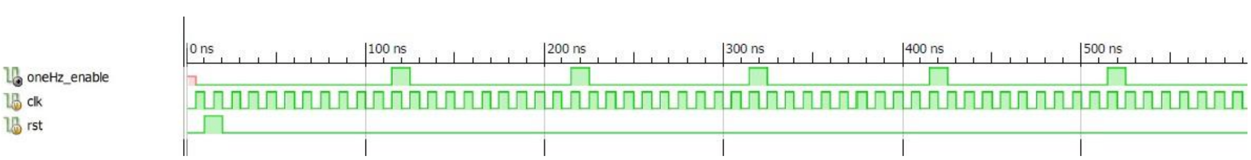

Figure 16 Divider at 10MHz

Timer

Module description

The timer is responsible for measuring the duration of each traffic light state by using the start_timer, 1Hz enable, and Time Parameter value inputs. It counts for the appropriate duration, and when it has finished counting, the expired signal goes high for one clock cycle to signal to the FSM that it should transition to the next state.

Module overview

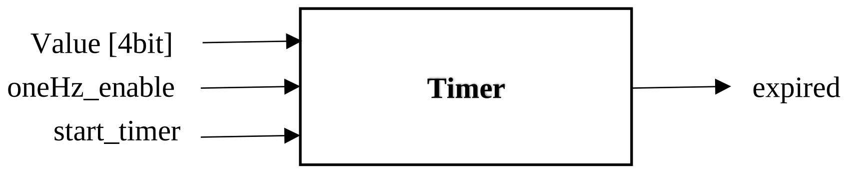

Figure 17 Timer block diagram

Since the value change is behind the start_timer by a one clock cycle this is solved by assigning value to the time_left after a one clock cycle. This way it does not change the time values.

Test bench output

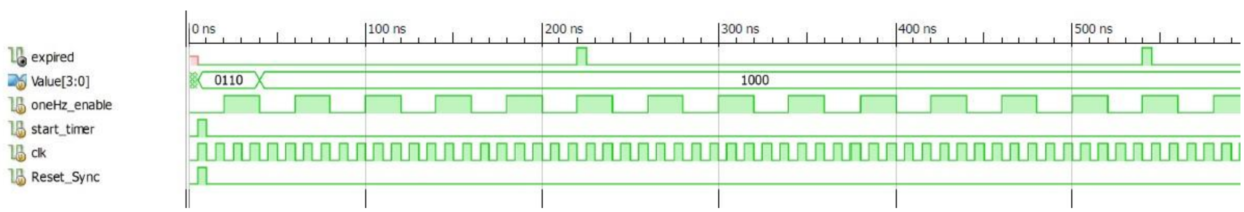

Figure 18 Timer simulation output

Finite State Machine

Module description

The Finite State Machine (FSM) is responsible for controlling the sequencing of the traffic lights. It determines when to transition from one state to another, based on inputs from the Walk Register, sensor signals, and the expired signal from the timer. It effectively manages the changes in state of the traffic light according to the inputs available to it.

Module overview

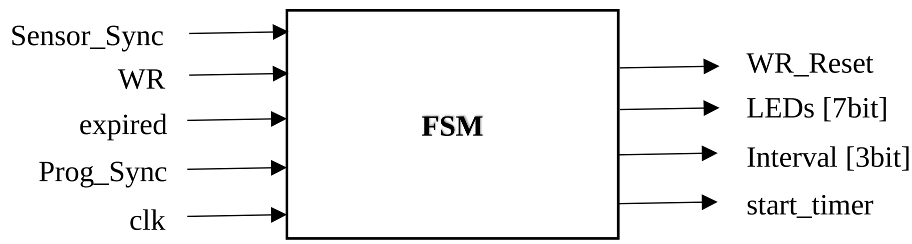

Figure 19 FSM block diagram

Test bench output

Normal routine

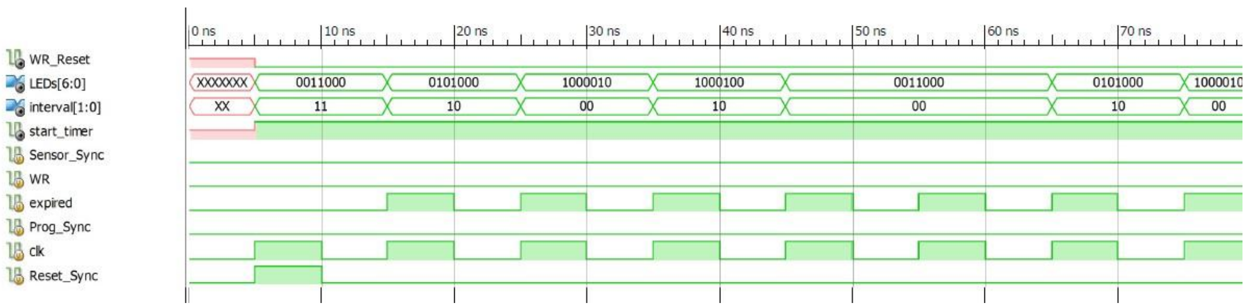

Figure 20 Normal operation

Walk request

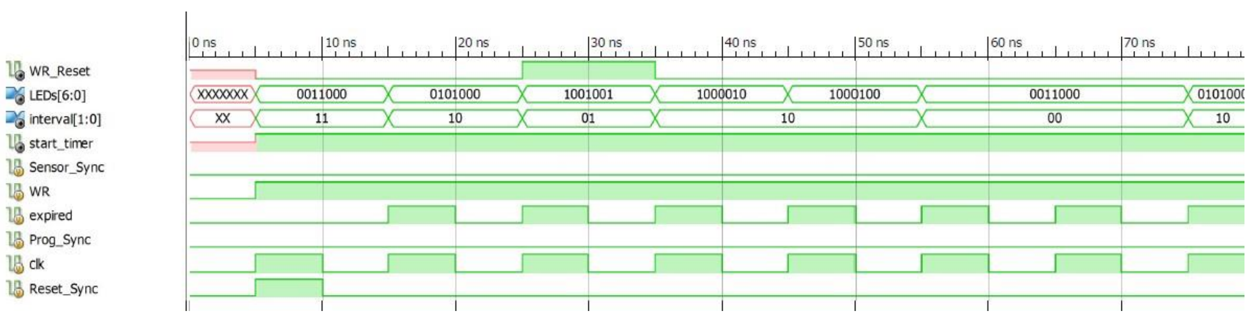

Figure 21 Walk request by a pedestrian

Vehicle sensor

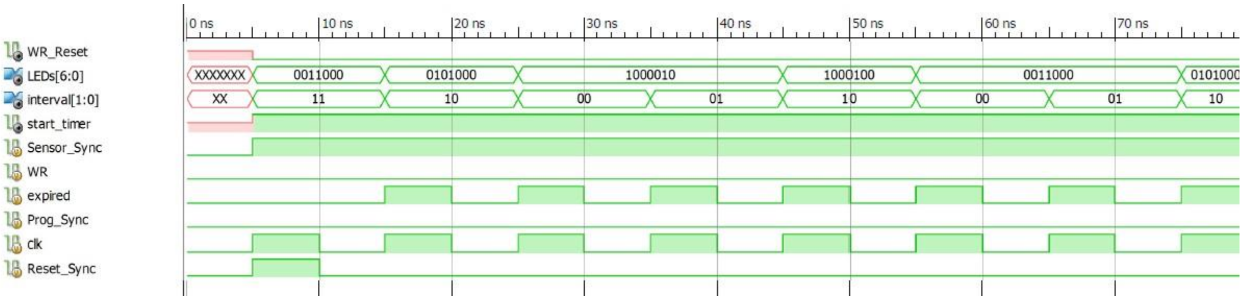

Figure 22 Vehicle sensor activated

At the start of the system, FSM directly enter the State A for a time of 2* t_{BASE} without considering about any other sensor inputs. This occurs when the system is reset.

LEDs values represent the Following states of the system

Bits of the LEDs represent lights in following manner

LED sequence: [Red_{main}, $Yellow_{main}$, $Green_{main}$, Red_{side}, $Yellow_{side}$, $Green_{side}$, Walk]

State	Representation	Meaning
A	0011000	Main green
B	0101000	Main yellow
C	1000010	Side green
D	1000100	Side yellow
E	1001001	walk

Interval values represent following timer values

Representation	Timer value
00	t_{BASE}
01	t_{EXT}
10	t_{YEL}
11	2*t_{BASE}