ATTENDANCE REGISTER

DSD Project Report

Submitted in partial fulfilment of the requirements for the degree of

BACHELOR OF TECHNOLOGY in

ELECTRONICS AND COMMUNICATION ENGINEERING

By

NISHCHAY PALLAV (221EC233)

&

YASH (221EC263)

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING,

NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA,

SURATHKAL, MANGALORE -575025

November, 2023

ATTENDANCE REGISTER

The Objective of the project is to develop a digital attendance checker which updates the students about their current attendance status. The project consists the Verilog design which can either be used as a software model or fabricated into an actual circuit along with physical circuital model. The core objective is to keep a record of an individual's attendance and the total working days by which the system compares the attendance of an individual with a predetermined threshold and display him safe or unsafe message accordingly.

As in institutions attendance is also an integral part of assessment nowadays. It allows educators to identify students who may be struggling or falling behind, enabling timely intervention and support.

Even in corporations, schools and other functional bodies attendance plays a vital role in keeping record of participation, proper functioning of infrastructure, Evaluation of programs, Promotions, Safety and security etc. The mandatory actions can also be taken within time by keeping a good and manageable record of attendees.

The project consists flip-flop based Counters, Shift Registers, 4 Bit Adders, 4 Bit Comparators and a 7-Segment Display.

All the components work together to count and perform suitable operations on the recorded attendance. The current model is designed for maximum attendance of up to 24 sessions. The safe/unsafe status is given to students such that if there attendance is above 75 percent of the total working days then there are considered safe otherwise unsafe.

PROBLEM STATEMENT:

The Attendance register is a device with everything related to his attendance criterion that a student *needs*. This includes the current attendance of the student, the total number of days the classes have been going on, the numbers of leaves the student can take or the number of days the student has to attend the class, if the attendance percentage of the student is safe, i.e., above 75%, and it even shows him if his attendance is short thereby giving him a FA grade.

COUNTING THE ATTENDANCE:

To keep the record of attendance the present days and total working days are being counted simultaneously with the use of two different up-down counters.

UP COUNTERS:

The Basic structure of up counters is made of sequentially arranged flip-flops with their output signals performing as input clock signals for the next arranged flip-flop. This means that each flip-flop will toggle on every clock pulse, but only if the previous flip-flop is in the "high" or “1” state. The above definition is mostly used for Synchronous Counters.

COMPARISON WITH THE THRESHOLD:

To compare with the threshold, we need to perform some mathematical operations with the help of Shift Registers and 4-Bit Adders. After That the comparison will take place with the help of 4-Bit Comparators.

Shift Register:

It is a digital sequential circuit consisting flip-flops arranged in a sequence each of which stores one bit. They can shift data bit by bit and have vast uses. They can also be used to perform some specific mathematical operations on stored binary data.

4-Bit Adders:

4-Bit Adders are used to add two 4-bit binary numbers. They are used as basic elements in many digital circuits. The 4-Bit Adders consists four full adders which adds two binary digits to and produces sum and carry. They are easy to design and can be managed according to the size of numbers to be added.

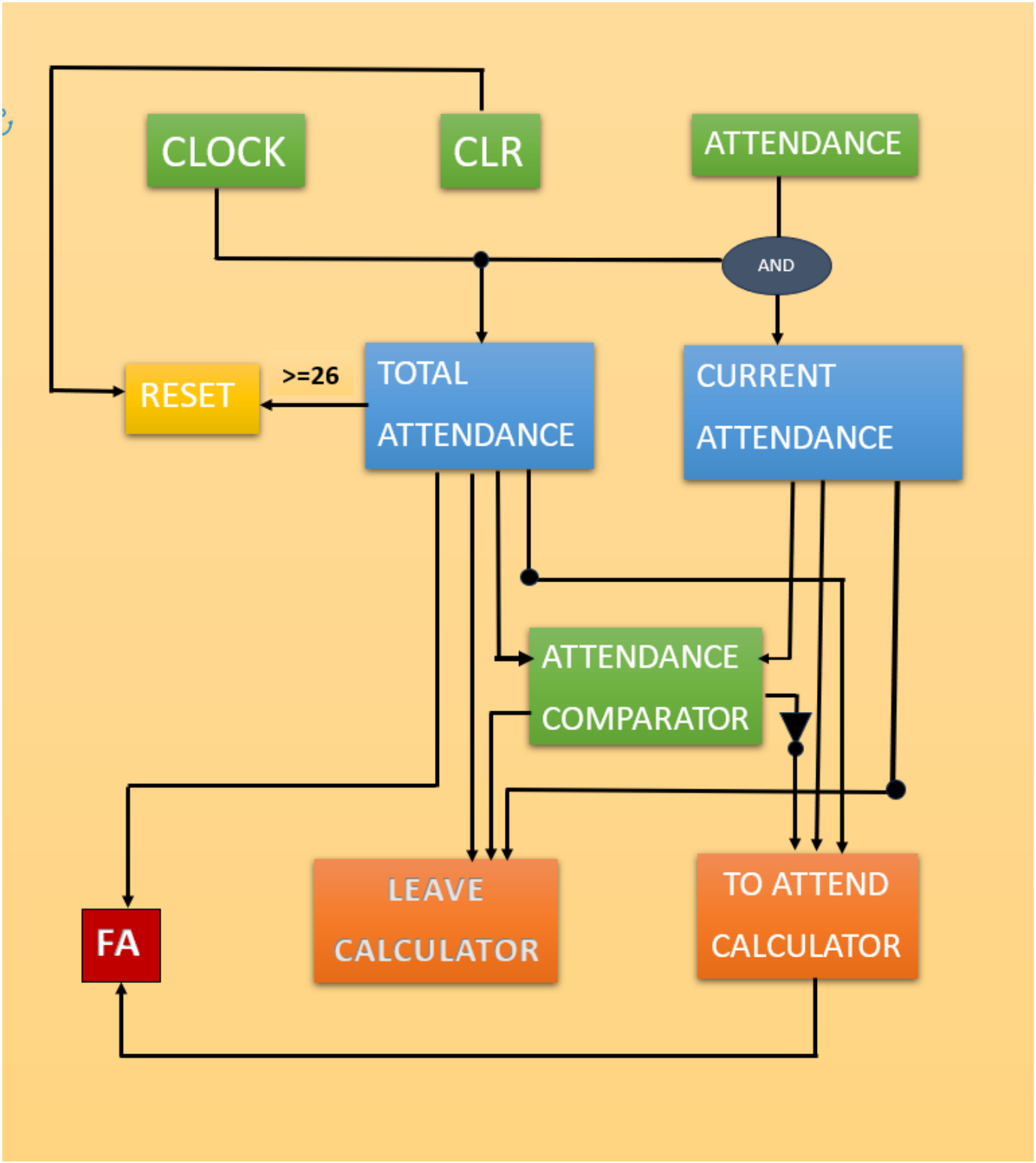
4-Bit Comparators:

4-Bit Comparators are used to compare two 4-bit binary numbers. They consist four 1 bit comparators which compares two binary digits to check the cases of greater than, less than or equal to. The 1-bit comparators are connected in a parallel fashion, so that the corresponding bits of the two input numbers are compared simultaneously.

DESIGN:

Flowchart:

Given below is a flowchart that describes the working of the device in brief, it includes various component from counters to attendance percentage calculator, **the inputs used are clk** (clock, where each cycle represents a day), **clr** (clear, this represents the start of a new batch i.e., it resets the machine) and **attendance** (this represents that student has attended the class), **the outputs that our machine provides are total_attendance** (the number of days the classes have been going on, this maxes out at 25), **current_attendance** (the number of days the student has attended the classes, this is always <= total_attendance), **is_safe** (this shows the student whether his attendance >= 75%, if true this gives a high output), **FA** (this output is high when the student has a shortage of attendance which can't be salvaged in the remaining days of class), **leaves** (this shows the number of days the student can take a leave while keeping his attendance safe), **to_attend** (this shows the number of days the student will have to attend classes to keep his attendance safe).


CLOCK
CLR
ATTENDANCE
AND
RESET
>=26
TOTAL ATTENDANCE
CURRENT ATTENDANCE
ATTENDANCE COMPARATOR
FA
LEAVE CALCULATOR
TO ATTEND CALCULATOR

Verilog Modelling:

Given below is the MODULE code of the attendance register:

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// TEAMATES:NISHCHAY PALLAV       &       YASH
// ROLL NUMBERS:221EC233          &       221EC263
//
// Create Date: 13.11.2023 18:27:00
// Module Name: attendance_register
// Project Name: ATTENDANCE REGISTER
// Tool Versions: 1.0
// Description: A USER FRIENDLY ATTENDANCE REGISTER FOR STUDENTS
//
//
// Revision 0.01 - File Created
// Additional Comments: LAB PROJECT FOR 3 SEMESTER DSD LAB EC204
//////////////////////////////////////////////////////////////////////////////////
module attendance_register (
  input clk,
  input clr,
  input attendance,
  output reg is_safe,
  output reg FA,
  output reg [3:0]leaves,
  output reg [6:0]to_attend,
  output reg [6:0]total_attandance,
  output reg [6:0]current_attandance
);


  reg fa=0;
  reg [6:0] count1 = 1;
  reg [6:0] count2 = 1;
  reg [6:0] temp_leaves = 0;
```

```
//COUNTERS USED FOR BOTH TOTAL_ATTENDANCE AND CURRENT_ATTENDANCE
 always @(posedge clk)begin
  total_attandance = total_attandance + 1;
  if(attendance)
    current_attandance = current_attandance + 1;
 end

 //RESET
 always @(*)begin
  if(total_attandance >= 26 || clr==1 ) begin
    total_attandance = 1; current_attandance = 1 ; count1 = 1 ; count2 = 1 ; leaves = 0 ;
    to_attend = 0 ; is_safe = 0 ; FA = 0 ; temp_leaves = 0 ;
  end
 end

//COMPARATOR FOR ATTENDANCE SAFETY AND FA CALCULATOR
 always @(posedge clk) begin
  if((total_attandance + to_attend) >= 26 )
    fa = 1;
  FA=fa;
  count2 = current_attandance << 2;
  count1 = (total_attandance << 1) + total_attandance;
  if(count2 >= count1)
    is_safe = 1;
  else
    is_safe = 0;
 end

//TOTAL NUMBER OF LEAVES CALCULATOR
 always @(posedge clk)begin
  if(is_safe)begin
    to_attend = 0;
    if((current_attandance<<2) >= ((total_attandance<<1) + total_attandance))begin
```

```
        temp_leaves = (current_attandance<<2) - ((total_attandance<<1) + total_attandance);
        leaves = temp_leaves/3;
      end
    else
      leaves = 0;
    end

 //TOTAL NUMBER OF DAYS TO ATTEND CLASSES TO BE SAFE CALCULATOR
    else begin
      leaves = 0;
      if(((total_attandance<<1) + total_attandance) >= (current_attandance<<2))
        to_attend = ((total_attandance<<1) + total_attandance) - (current_attandance<<2);
      else
        to_attend=0;
    end
  end

endmodule
```

Given below is the TESTBENCH for the given module:

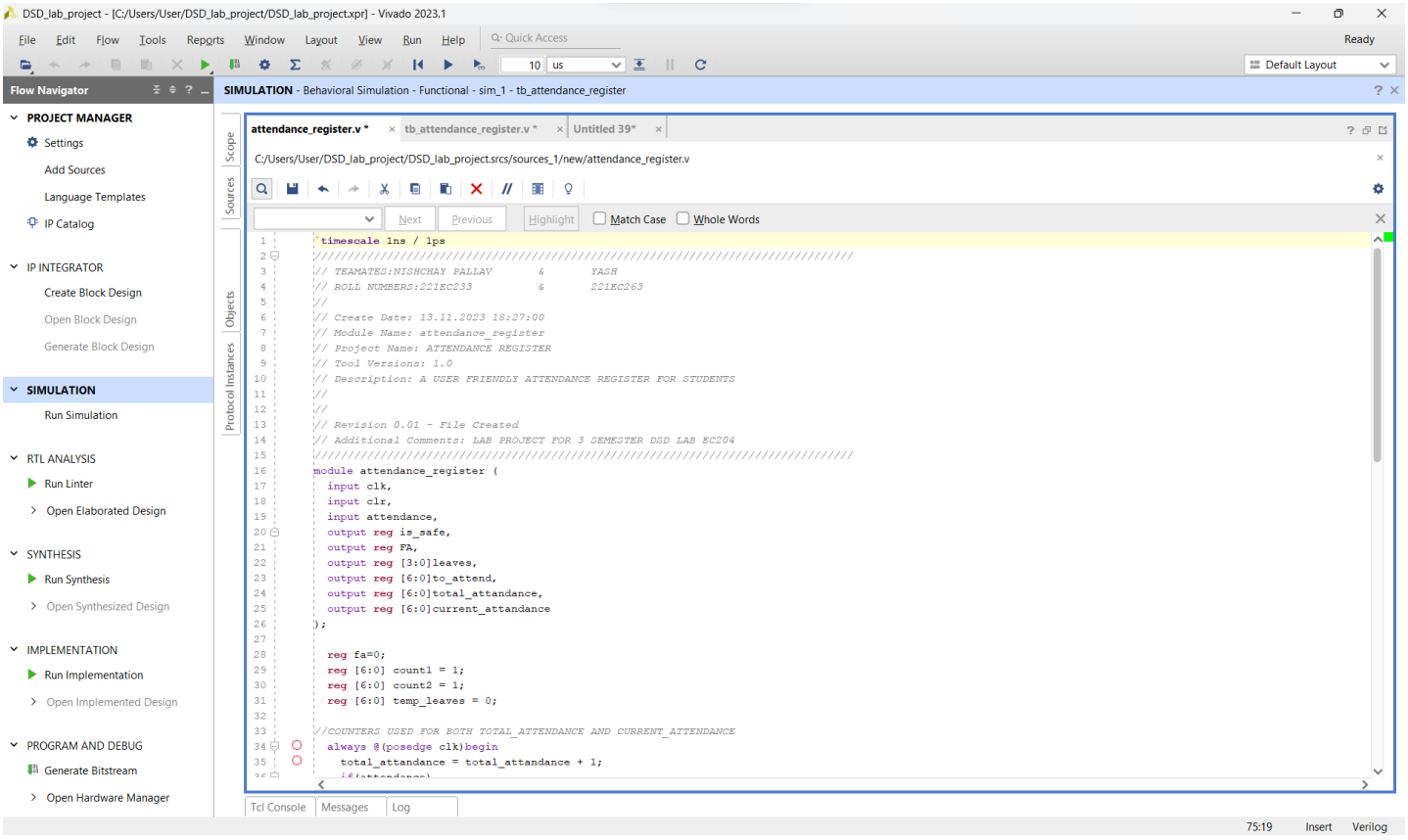
```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// TEAMATES:NISHCHAY PALLAV       &       YASH
// ROLL NUMBERS:221EC233          &       221EC263
//
// Create Date: 13.11.2023 18:27:00
// Module Name: attendance_register
// Project Name: ATTENDANCE REGISTER
// Tool Versions: 1.0
// Description: A USER FRIENDLY ATTENDANCE REGISTER FOR STUDENTS
//
//
// Revision 0.01 - File Created
// Additional Comments: LAB PROJECT FOR 3 SEMESTER DSD LAB EC204
//////////////////////////////////////////////////////////////////////////////////
```

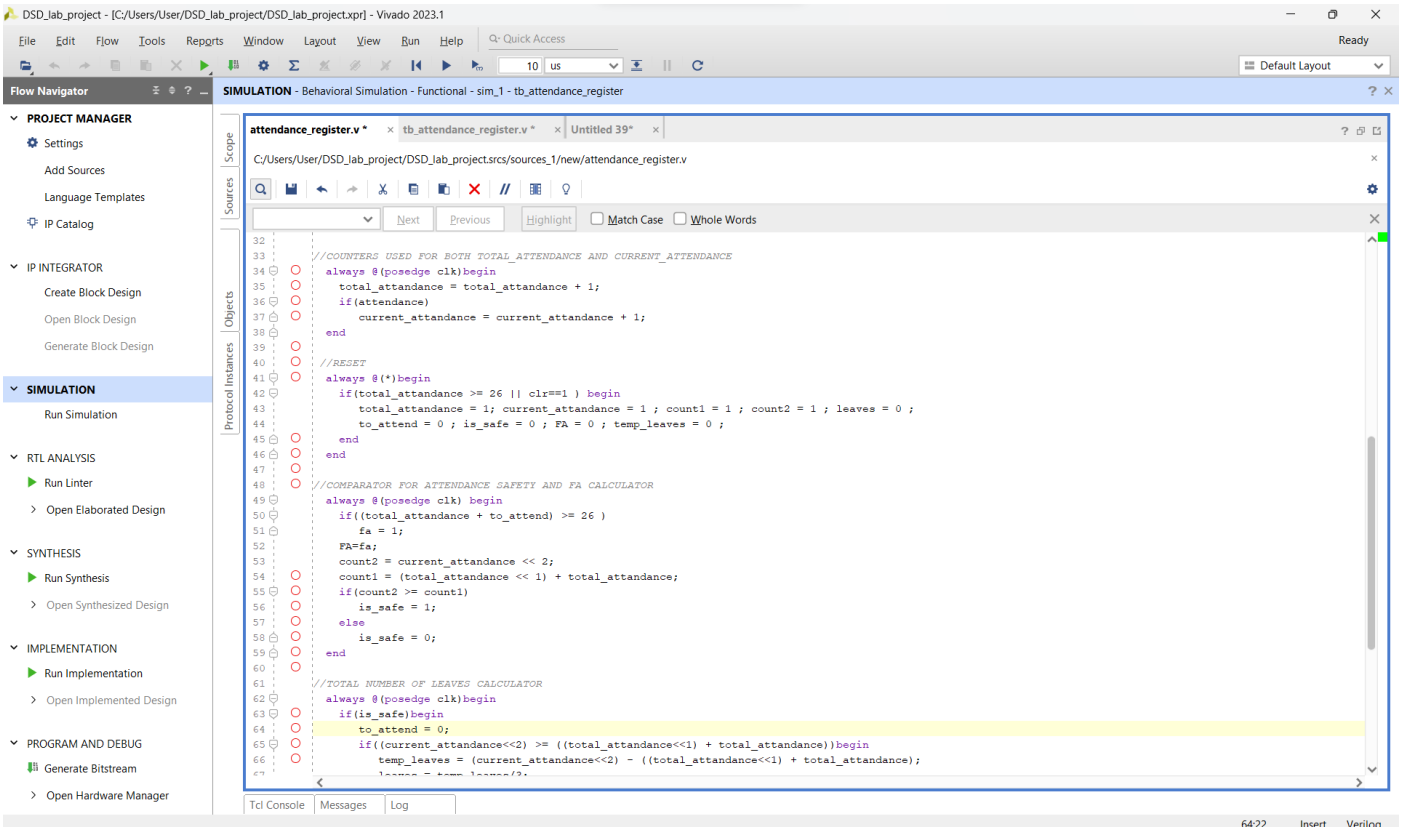
```
module tb_attendance_register;
 // Inputs
 reg clk;
 reg attendance;
 reg clr;

 // Outputs
 wire is_safe;
 wire FA;
 wire [3:0]leaves;
 wire [6:0]to_attend;
 wire [6:0]total_attandance;
 wire [6:0]current_attandance;

 // Attendance register instance
 attendance_register uut_attendance_checker(
  clk,
  clr,
  attendance,
  is_safe,
  FA,
  leaves,
  to_attend,
  total_attandance,
  current_attandance
 );

  initial begin
     clk <= 1'b0;
     attendance=1;clr <= 1'b1;#1
     attendance<=1;clr <= 1'b0;#11
     attendance<=0;clr <= 1'b0;#7
```

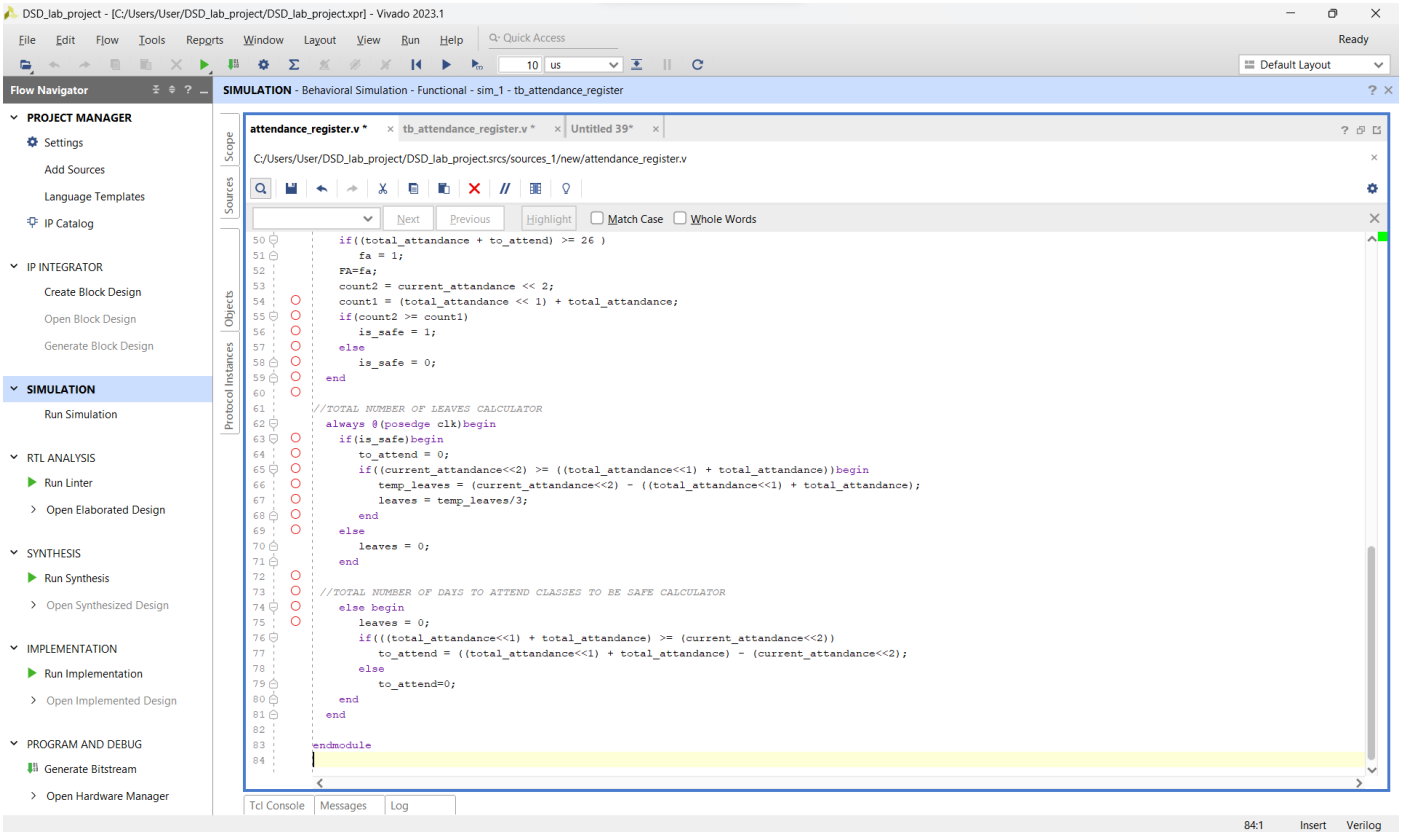
```
    attendance<=1;clr <= 1'b0;#6
    attendance=0;
  end

  // Clock generator
  always #0.5 clk <= ~clk;

endmodule
```

Given below are the snapshots of the above code:

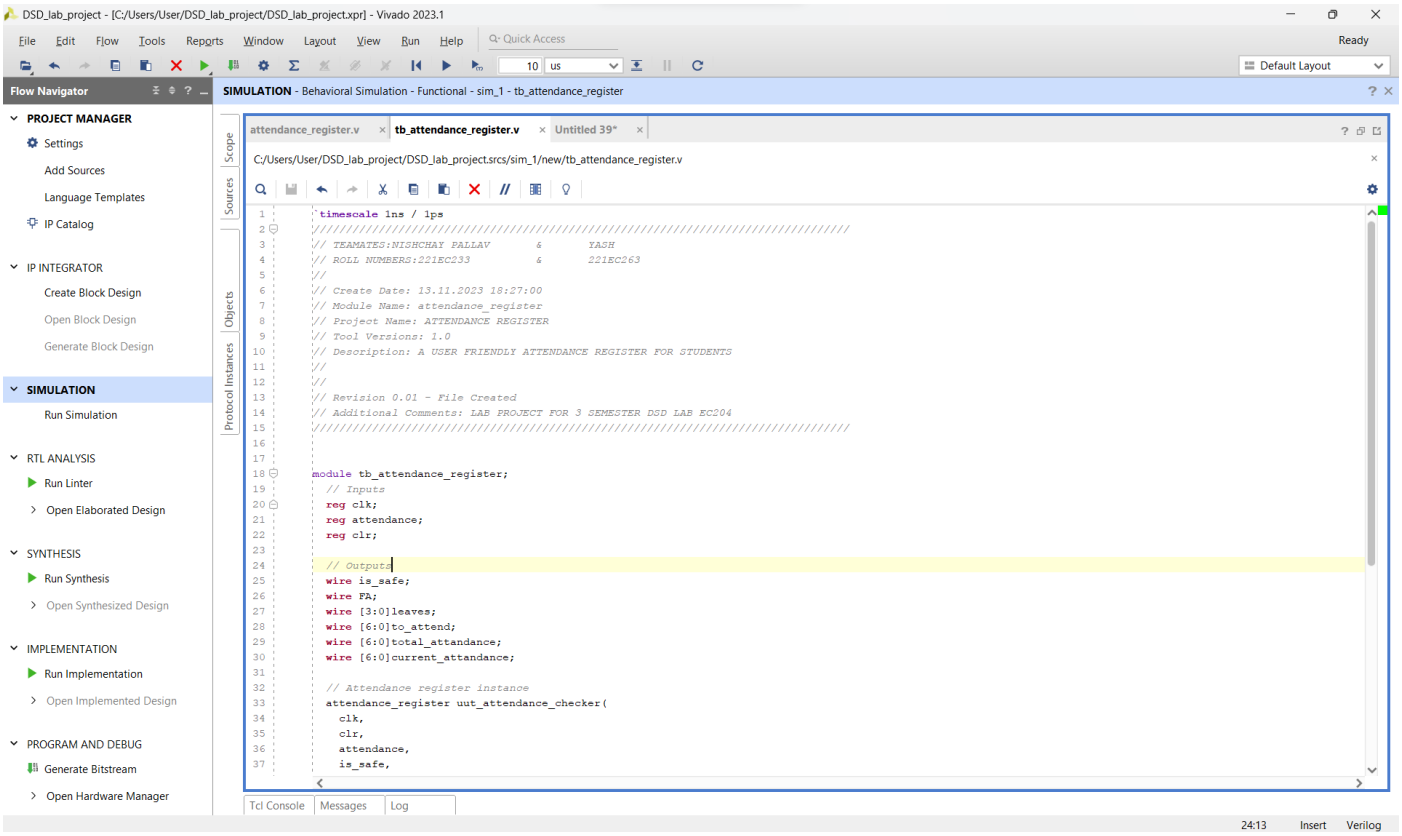
THE MODULES:



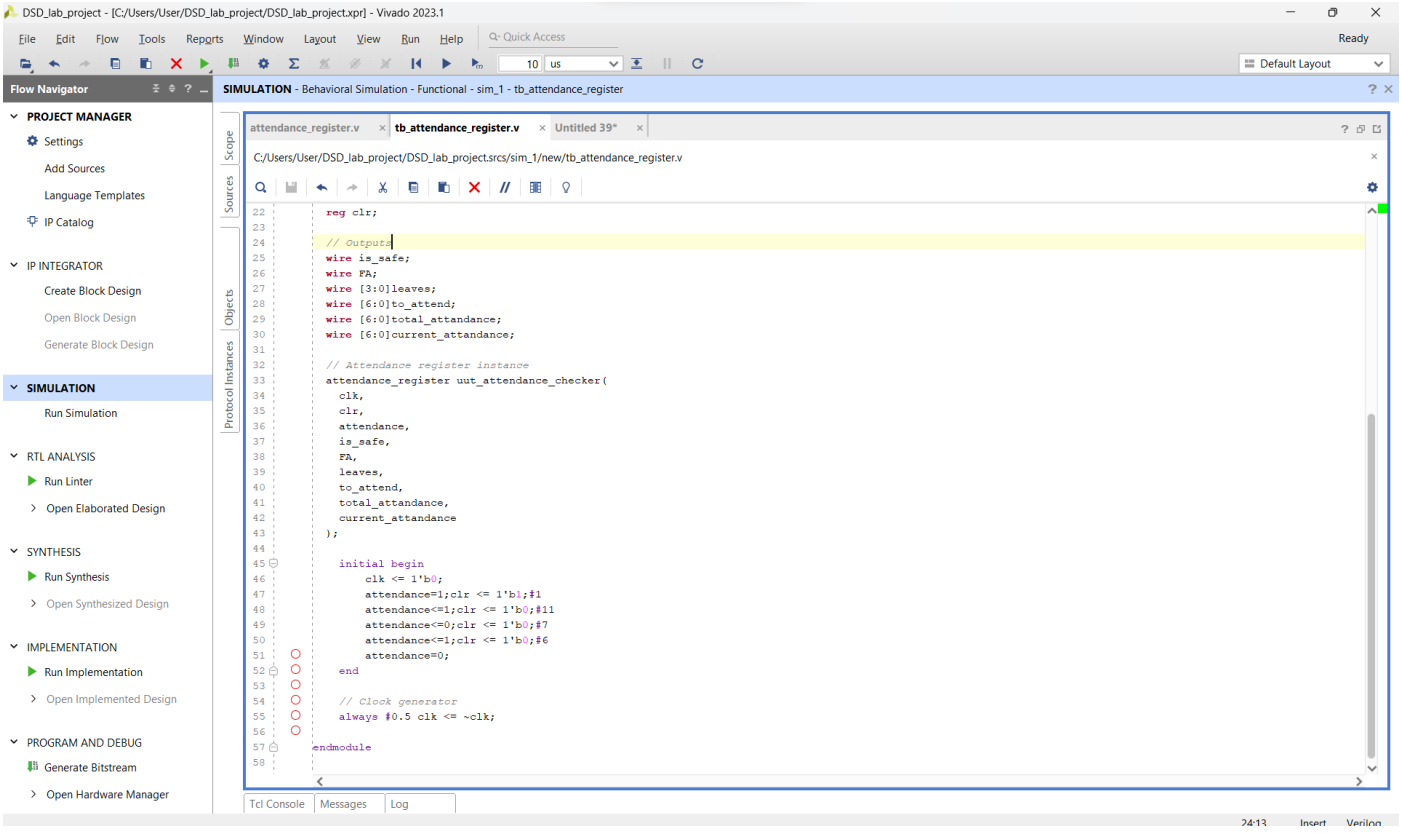

DSD_lab_project - [C:/Users/User/DSD_lab_project/DSD_lab_project.xpr] - Vivado 2023.1
File Edit Flow Tools Reports Window Layout View Run Help
Quick Access
Ready
Default Layout
Flow Navigator
PROJECT MANAGER
Settings
Add Sources
Language Templates
IP Catalog
IP INTEGRATOR
Create Block Design
Open Block Design
Generate Block Design
SIMULATION
Run Simulation
RTL ANALYSIS
Run Linter
Open Elaborated Design
SYNTHESIS
Run Synthesis
Open Synthesized Design
IMPLEMENTATION
Run Implementation
Open Implemented Design
PROGRAM AND DEBUG
Generate Bitstream
Open Hardware Manager
SIMULATION - Behavioral Simulation - Functional - sim_1 - tb_attendance_register
attendance_register.v *
tb_attendance_register.v *
Untitled 39*
C:/Users/User/DSD_lab_project/DSD_lab_project.srcs/sources_1/new/attendance_register.v
Next
Previous
Highlight
Match Case
Whole Words
//COUNTERS USED FOR BOTH TOTAL_ATTENDANCE AND CURRENT_ATTENDANCE
always @(posedge clk)begin
total_attandance = total_attandance + 1;
if(attendance)
current_attandance = current_attandance + 1;
end
//RESET
always @(*)begin
if(total_attandance >= 26 || clr==1) begin
total_attandance = 1; current_attandance = 1 ; count1 = 1 ; count2 = 1 ; leaves = 0 ;
to_attend = 0 ; is_safe = 0 ; FA = 0 ; temp_leaves = 0 ;
end
end
//COMPARATOR FOR ATTENDANCE SAFETY AND FA CALCULATOR
always @(posedge clk) begin
if((total_attandance + to_attend) >= 26)
fa = 1;
FA=fa;
count2 = current_attandance << 2;
count1 = (total_attandance << 1) + total_attandance;
if(count2 >= count1)
is_safe = 1;
else
is_safe = 0;
end
//TOTAL NUMBER OF LEAVES CALCULATOR
always @(posedge clk)begin
if(is_safe)begin
to_attend = 0;
if((current_attandance<<2) >= ((total_attandance<<1) + total_attandance))begin
temp_leaves = (current_attandance<<2) - ((total_attandance<<1) + total_attandance);
Tcl Console
Messages
Log


DSD_lab_project - [C:/Users/User/DSD_lab_project/DSD_lab_project.xpr] - Vivado 2023.1
File Edit Flow Tools Reports Window Layout View Run Help
Quick Access
Ready
Default Layout
Flow Navigator
PROJECT MANAGER
Settings
Add Sources
Language Templates
IP Catalog
IP INTEGRATOR
Create Block Design
Open Block Design
Generate Block Design
SIMULATION
Run Simulation
RTL ANALYSIS
Run Linter
Open Elaborated Design
SYNTHESIS
Run Synthesis
Open Synthesized Design
IMPLEMENTATION
Run Implementation
Open Implemented Design
PROGRAM AND DEBUG
Generate Bitstream
Open Hardware Manager
SIMULATION - Behavioral Simulation - Functional - sim_1 - tb_attendance_register
attendance_register.v *
tb_attendance_register.v *
Untitled 39*
C:/Users/User/DSD_lab_project/DSD_lab_project.srcs/sources_1/new/attendance_register.v
Next
Previous
Highlight
Match Case
Whole Words
if((total_attandance + to_attend) >= 26)
fa = 1;
FA=fa;
count2 = current_attandance << 2;
count1 = (total_attandance << 1) + total_attandance;
if(count2 >= count1)
is_safe = 1;
else
is_safe = 0;
end
//TOTAL NUMBER OF LEAVES CALCULATOR
always @(posedge clk)begin
if(is_safe)begin
to_attend = 0;
if((current_attandance<<2) >= ((total_attandance<<1) + total_attandance))begin
temp_leaves = (current_attandance<<2) - ((total_attandance<<1) + total_attandance);
leaves = temp_leaves/3;
end
else
leaves = 0;
end
//TOTAL NUMBER OF DAYS TO ATTEND CLASSES TO BE SAFE CALCULATOR
else begin
leaves = 0;
if(((total_attandance<<1) + total_attandance) >= (current_attandance<<2))
to_attend = ((total_attandance<<1) + total_attandance) - (current_attandance<<2);
else
to_attend=0;
end
end
endmodule
Tcl Console
Messages
Log
84:1
Insert
Verilog

THE TESTBENCH:

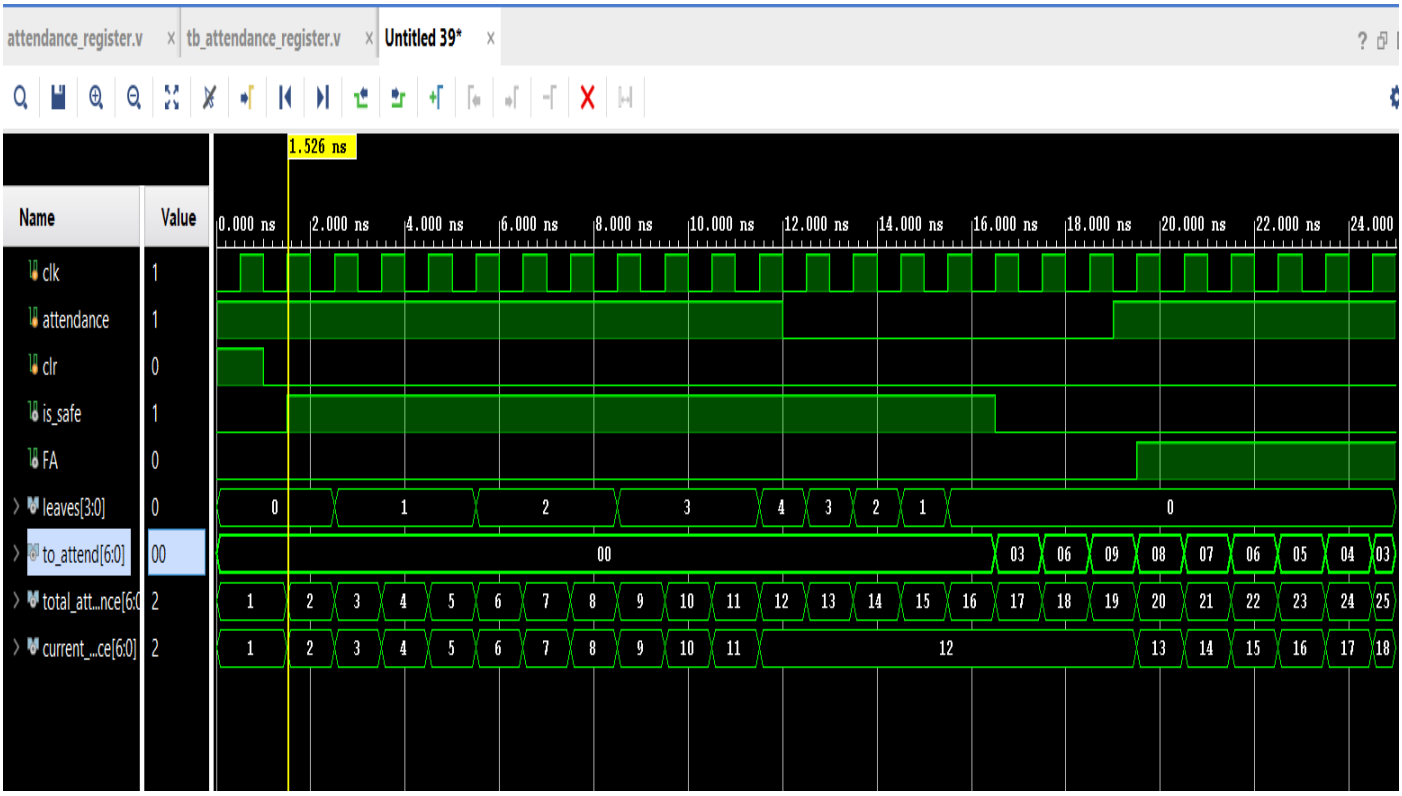


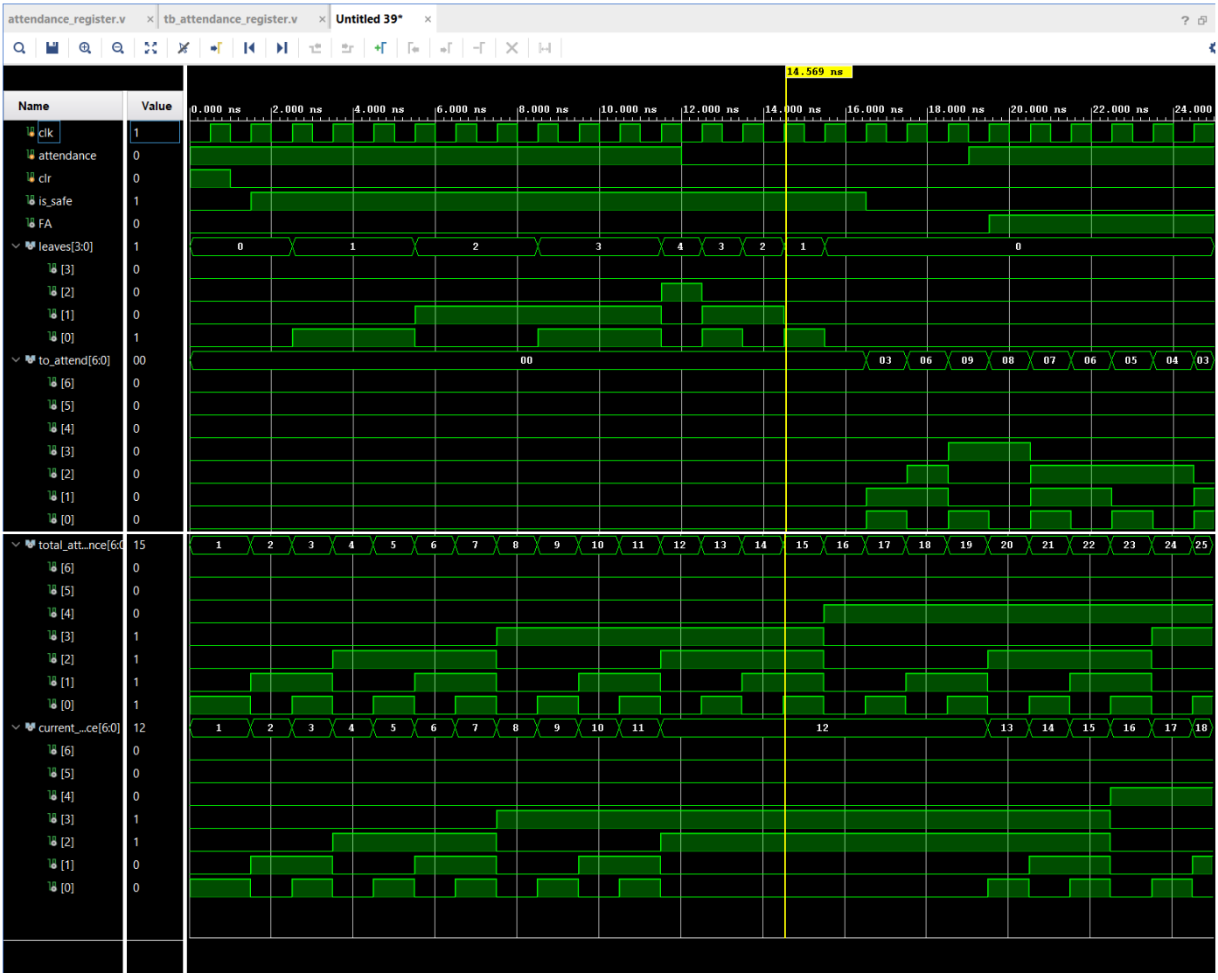

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// TEAMATES:NISHCHAY PALLAV        &        YASH
// ROLL NUMBERS:221EC233           &        221EC263
//
// Create Date: 13.11.2023 18:27:00
// Module Name: attendance_register
// Project Name: ATTENDANCE REGISTER
// Tool Versions: 1.0
// Description: A USER FRIENDLY ATTENDANCE REGISTER FOR STUDENTS
//
//
// Revision 0.01 - File Created
// Additional Comments: LAB PROJECT FOR 3 SEMESTER DSD LAB EC204
//////////////////////////////////////////////////////////////////////////////////


module tb_attendance_register;
  // Inputs
  reg clk;
  reg attendance;
  reg clr;

  // Outputs
  wire is_safe;
  wire FA;
  wire [3:0]leaves;
  wire [6:0]to_attend;
  wire [6:0]total_attandance;
  wire [6:0]current_attandance;

  // Attendance register instance
  attendance_register uut_attendance_checker(
    clk,
    clr,
    attendance,
    is_safe,
```




```
  reg clr;

  // Outputs
  wire is_safe;
  wire FA;
  wire [3:0]leaves;
  wire [6:0]to_attend;
  wire [6:0]total_attandance;
  wire [6:0]current_attandance;

  // Attendance register instance
  attendance_register uut_attendance_checker(
    clk,
    clr,
    attendance,
    is_safe,
    FA,
    leaves,
    to_attend,
    total_attandance,
    current_attandance
  );

    initial begin
        clk <= 1'b0;
        attendance=1;clr <= 1'b1;#1
        attendance<=1;clr <= 1'b0;#11
        attendance<=0;clr <= 1'b0;#7
        attendance<=1;clr <= 1'b0;#6
        attendance=0;
    end

    // Clock generator
    always #0.5 clk <= ~clk;

endmodule
```

SIMULATION:


attendance_register.v
tb_attendance_register.v
Untitled 39*
1.526 ns
Name
Value
clk
attendance
clr
is_safe
FA
leaves[3:0]
to_attend[6:0]
total_att...nce[6:0]
current_...ce[6:0]


attendance_register.v
tb_attendance_register.v
Untitled 39*
14.569 ns
Name
Value
clk
attendance
clr
is_safe
FA
leaves[3:0]
to_attend[6:0]
total_att...nce[6:0]
current_...ce[6:0]

WORKING:

1. The code uses the clock pulses to describe a passage of a day, this clock is used as input to the total_attendance counter: whenever a day passes i.e., with each cycle of the clock pulse the counter increase its value by 1, the value of the total attendance is capped at 25 days.

2. After which the code is reset using a defined reset statement that signifies the start of new batch, this is done using if statement: total_attendance>=26, the batch can also be reset using an input called clear.

3. The current_attendance counter works by increasing its count if attendance input is high i.e., the student is present and a day passes i.e., a clock pulse.

4. The percentage criterion of 75% is calculated and compared using the idea of multiplying the total_attendance by 3 and current attendance by 4 derived from the equation: (current_attendance/total_attendance) >= 3/4 (75%). This is done by shifting the bit value of current_attendance twice in left direction: (current_attendance<<2), and bit value of total_attendance once in left direction and adding itself to the shifted value: (total_attendance<<1 + total_attendance).

5. The value of current_attendance<<2 is assigned to **count1** and (total_attendance<<1 + total_attendance) is assigned to **count2** and the is_safe criterion is decided by the statement: count2 >= count1.

6. Using the same method for multiplying by 3 and 4 we can calculate the values for leaves that is found from the equation: (current_attendance/ (total_attendance + leaves)) = 3/4. This solves to a final equation:
   leaves = (4*current_attendance – 3*total_attendance)/3.

7. Similarly, we find the value of to_attend using the equation:
   (current_attendance + to_attend)/ (total_attendance + to_attend) = 3/4. This solves to a final equation:
   to_attend = 3*total _attendance – 4*current_attendance.

8. To save us from the cases where the value of leaves and to_attend are negative we use if statements with suitable statements.

9. Finally the FA criterion is decided when the remaining number of days the classes are left are less than the numbers of days you have to attend by the relation: (total_attendance + to_attend) >= 26.

SUMMARY:

Project Objective

- Develop a digital attendance checker for students.
- Utilizes Verilog design for software or circuit implementation.
- Records individual attendance and compares it with a predetermined threshold.
- Aims to provide timely intervention and support in educational settings.

Components Used (If hardware implemented):

- Flip-flop based Counters.
- Shift Registers.
- 4 Bit Adders.
- 4 Bit Comparators.
- 7-Segment Display.
- Basic gates

Functionality:

- Records attendance using up-down counters and flip-flops.
- Utilizes Shift Registers, 4-Bit Adders, and 4-Bit Comparators for mathematical operations.
- Designed for a maximum of 25 sessions.
- Students labelled safe if attendance is above 75%, otherwise unsafe.

Problem Statement:

- Device includes current attendance, total days, leave information, and safety status.
- Addresses attendance concerns in educational institutions and functional bodies.
- Enables timely actions and maintains manageable attendance records.

Counting Attendance:

- Uses up counters with sequentially arranged flip-flops.
- Involves comparison with a threshold using Shift Registers, 4-Bit Adders, and 4-Bit Comparators.

Design:

- Flowchart guides the working of the device.
- Inputs include clk (clock), clr (clear), and attendance.
- Outputs include total_attendance, current_attendance, is_safe, FA, leaves, and to_attend.

Working:

1. Clock pulses represent a day, increasing total_attendance.
2. Reset occurs after 25 days or through a clear input.
3. current_attendance increases on student attendance.
4. Safety status is determined by comparing attendance percentages.
5. Leaves and to_attend calculated to maintain safety status.
6. FA status determined based on remaining days and attendance requirements.

Equations Used:

- (current_attendance/total_attendance) >= 3/4
- (current_attendance/ (total_attendance + leaves)) = 3/4
- (current_attendance + to_attend)/ (total_attendance + to_attend) = 3/4

Final Criteria:

- FA status is determined when remaining days are insufficient to meet attendance requirements.

CONCLUSION:

In conclusion, the developed digital attendance checker offers a robust solution for tracking and managing student attendance. By integrating various components such as counters, shift registers, and 4-bit comparators, the system provides a reliable mechanism to assess individual attendance against a predetermined threshold. The project not only addresses the immediate need for accurate attendance records in educational institutions but also highlights its potential applicability in diverse functional bodies. The inclusion of features like safety status, leave calculations, and future attendance requirements enhances its utility. Overall, this project serves as an effective tool for promoting timely intervention, facilitating proper program evaluation, and ensuring the seamless functioning of educational and corporate environments.

REFERENCES:

- Verilog reference Appendix A.
- Digital system Design Lab record book.
- Google for basic understanding.
- IRIS attendance interface for design idea.