**Design:-**

module traffic\_light\_controller (

input wire clk,

input wire rst,

output reg [3:0] north,

output reg [3:0] east,

output reg [3:0] south,

output reg [3:0] west );

parameter GREEN\_TIME = 8;

parameter YELLOW\_TIME = 4;

reg [3:0] state\_north;

reg [3:0] state\_east;

reg [3:0] state\_south;

reg [3:0] state\_west;

reg [3:0] counter;

reg [3:0] timer;

localparam GREEN = 2'b00;

localparam YELLOW = 2'b01;

localparam RED = 2'b10;

initial begin

state\_north = GREEN;

state\_east = RED;

state\_south = RED;

state\_west = RED;

counter = 4'b0000;

timer = GREEN\_TIME;

end

always @(posedge clk or posedge rst) begin

if (rst) begin

state\_north <= GREEN;

state\_east <= RED;

state\_south <= RED;

state\_west <= RED;

counter <= 4'b0000;

timer <= GREEN\_TIME;

end else begin

state\_north = RED;

state\_east = RED;

state\_south = RED;

state\_west = RED;

case(counter)

4'b0000: begin

if (timer == 0) begin

timer <= YELLOW\_TIME;

state\_north <= YELLOW;

counter <= counter + 1;

end else begin

timer <= timer - 1;

end

end

4'b0001: begin

if (timer == 0) begin

timer <= GREEN\_TIME;

state\_east <= GREEN;

counter <= counter + 1;

end else begin

timer <= timer - 1;

end

end

4'b0010: begin

if (timer == 0) begin

timer <= YELLOW\_TIME;

state\_south <= YELLOW;

counter <= counter + 1;

end else begin

timer <= timer - 1;

end

end

4'b0011: begin

if (timer == 0) begin

timer <= GREEN\_TIME;

counter <= 4'b0000;

end else begin

timer <= timer - 1;

end

end

default: counter <= 4'b0000;

endcase

end

end

assign north = state\_north;

assign east = state\_east;

assign south = state\_south;

assign west = state\_west;

endmodule

**TESTBENCH :-**

// Code your testbench here

// or browse Examples

module testbench();

parameter CLK\_PERIOD = 10;

reg clk;

reg rst;

wire [3:0] north;

wire [3:0] east;

wire [3:0] south;

wire [3:0] west;

traffic\_light\_controller traffic\_light\_inst (

.clk(clk),

.rst(rst),

.north(north),

.east(east),

.south(south),

.west(west));

always begin

clk = 0;

#((CLK\_PERIOD / 2));

clk = 1;

#((CLK\_PERIOD / 2));

end

initial begin

rst = 1;

#50;

rst = 0;

#50;

#500;

$finish;

end

always @(posedge clk) begin

$display("Time = %0t ns: North = %b, East = %b, South = %b, West = %b", $time, north, east, south, west);

$dumpfile("testbench.vcd");

$dumpvars(1,testbench);

end

endmodule

**WAVEFORM:-**



**Design:-**

module vending\_machine(

output reg can\_despatch,

input [1:0] coin,

input clk, rst

);

reg [1:0] state, next\_state;

parameter s0 = 2'd0, s5 = 2'd1, s10 = 2'd2, s15 = 2'd3;

always @(posedge clk) begin

if (rst)

state <= s0;

else

state <= next\_state;

end

always @(state, coin) begin

case (state)

s0: begin

if (coin == 2'b01)

next\_state = s5;

else if (coin == 2'b10)

next\_state = s10;

else

next\_state = s0;

end

s5: begin

if (coin == 2'b01)

next\_state = s10;

else if (coin == 2'b10)

next\_state = s15;

else

next\_state = s5;

end

s10: begin

if (coin == 2'b01)

next\_state = s15;

else if (coin == 2'b10)

next\_state = s15; // No residual coin

else

next\_state = s10;

end

s15: begin

next\_state = s0; // After can is dispensed, reset to s0

end

default: next\_state = s0;

endcase

end

assign can\_despatch = (state == s15) ? 1 : 0;

endmodule

**Testbench :-**

module tb;

reg clk, rst;

wire can\_despatch;

reg [1:0] coin\_stack [0:5], coin;

integer i;

vending\_machine uut (

.can\_despatch(can\_despatch),

.clk(clk),

.coin(coin),

.rst(rst)

);

initial begin

$monitor("i=%0d, Time=%0t, coin=%b, can\_despatch=%b", i, $time, coin, can\_despatch);

$dumpfile("vending\_machine.vcd");

$dumpvars(1, tb);

clk = 0;

rst = 1;

i = 0;

#50 rst = 0;

coin\_stack[0] = 2'b00; // no rupee

coin\_stack[1] = 2'b01; // 5 rupee

coin\_stack[2] = 2'b10; // 10 rupee

coin\_stack[3] = 2'b01; // 5 rupee

coin\_stack[4] = 2'b10; // 10 rupee

coin\_stack[5] = 2'b00; // 0 rupee

end

always #10 clk = ~clk;

always @(posedge clk) begin

coin = coin\_stack[i];

i = i + 1;

if (i == 6) $finish;

end

endmodule

**Waveform :-**

