BASAVESHWARA ENGINEERING COLLEGE (AUTONOMOUS), BAGALKOT



DEPARTMENT OF

ELECTRONICS AND COMMUNICATION

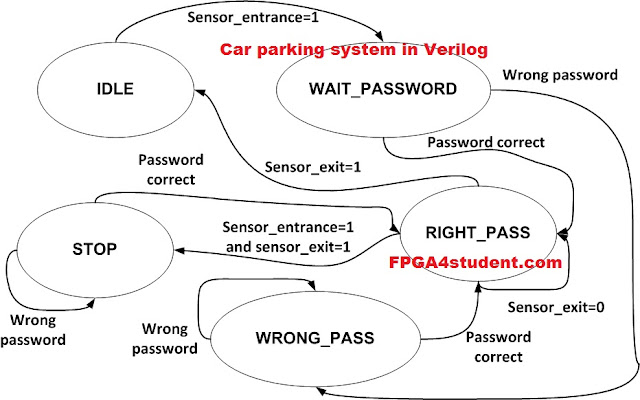
VERILOG PROJECT ON  
“CAR PARKING SYSTEM”

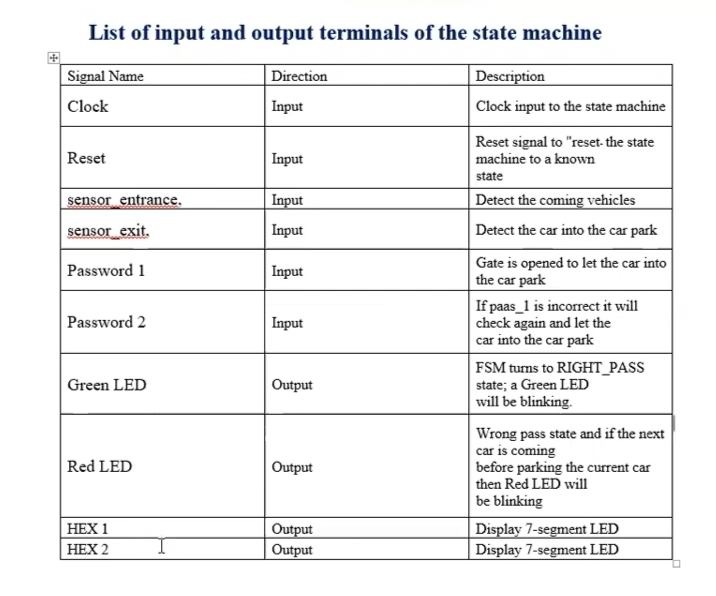
GUIDED BY:  
PROF: M.C. ARALIMARAD  
  
BY: KARUNA KANDAGALL 2BA20EC033  
 POOJA S BAGALI 2BA20EC050

INTRODUCTION…

* This simple project is to implement a car parking system in verilog . The [verilog code](https://www.fpga4student.com/p/verilog-project.html) for the car parking system is fully presented.
* In the entrance of the [parking system](https://en.wikipedia.org/wiki/Car_parking_system), there is a sensor which is activated to detect a vehicle coming. Once the sensor is triggered, a password is requested to open the gate. If the entered password is correct, the gate would open to let the vehicle get in. Otherwise, the gate is still locked. If the current car is getting in the car park being detected by the exit sensor and another the car comes, the door will be locked and requires the coming car to enter passwords.

BLOCK DIAGRAM:





CODE IS AS FOLLOWS…

* // fpga4student.com FPGA projects, Verilog projects, VHDL projects
* // Verilog project: Verilog code for car parking system
* `timescale 1ns / 1ps
* module parking\_system( input clk,reset\_n, input sensor\_entrance, sensor\_exit, input [1:0] password\_1, password\_2, output wire GREEN\_LED,RED\_LED, output reg [6:0] HEX\_1, HEX\_2 );
* parameter IDLE = 3'b000, WAIT\_PASSWORD = 3'b001, WRONG\_PASS = 3'b010, RIGHT\_PASS = 3'b011,STOP = 3'b100;
* // Moore FSM : output just depends on the current state
* reg[2:0] current\_state, next\_state;
* reg[31:0] counter\_wait;
* reg red\_tmp,green\_tmp;
* // Next state
* always @(posedge clk or negedge reset\_n)
* begin
* if(~reset\_n)
* current\_state = IDLE;
* else
* current\_state = next\_state;
* end
* // counter\_wait
* always @(posedge clk or negedge reset\_n)
* begin
* if(~reset\_n)
* counter\_wait <= 0;
* else if(current\_state==WAIT\_PASSWORD)
* counter\_wait <= counter\_wait + 1;
* else
* counter\_wait <= 0;
* end
* // change state
* always @(\*)
* begin
* case(current\_state)
* IDLE: begin
* if(sensor\_entrance == 1)
* next\_state = WAIT\_PASSWORD;
* else
* next\_state = IDLE;
* end
* WAIT\_PASSWORD: begin
* if(counter\_wait <= 3)
* next\_state = WAIT\_PASSWORD;
* else
* begin
* if((password\_1==2'b01)&&(password\_2==2'b10))
* next\_state = RIGHT\_PASS;
* else
* next\_state = WRONG\_PASS;
* end
* end
* WRONG\_PASS: begin
* if((password\_1==2'b01)&&(password\_2==2'b10))
* next\_state = RIGHT\_PASS;
* else
* next\_state = WRONG\_PASS;
* end
* RIGHT\_PASS: begin
* if(sensor\_entrance==1 && sensor\_exit == 1)
* next\_state = STOP;
* else if(sensor\_exit == 1)
* next\_state = IDLE;
* else
* next\_state = RIGHT\_PASS;
* end
* STOP: begin
* if((password\_1==2'b01)&&(password\_2==2'b10))
* next\_state = RIGHT\_PASS;
* else
* next\_state = STOP;
* end
* default: next\_state = IDLE;
* endcase
* end
* // LEDs and output, change the period of blinking LEDs here
* always @(posedge clk) begin
* case(current\_state)
* IDLE: begin
* green\_tmp = 1'b0;
* red\_tmp = 1'b0;
* HEX\_1 = 7'b1111111; // off
* HEX\_2 = 7'b1111111; // off
* end
* WAIT\_PASSWORD: begin
* green\_tmp = 1'b0;
* red\_tmp = 1'b1;
* HEX\_1 = 7'b000\_0110; // E
* HEX\_2 = 7'b010\_1011; // n

End

* WRONG\_PASS: begin
* green\_tmp = 1'b0;
* red\_tmp = ~red\_tmp;
* HEX\_1 = 7'b000\_0110; // E
* HEX\_2 = 7'b000\_0110; // E
* end
* RIGHT\_PASS: begin
* green\_tmp = ~green\_tmp;
* red\_tmp = 1'b0;
* HEX\_1 = 7'b000\_0010; // 6
* HEX\_2 = 7'b100\_0000; // 0
* end
* STOP: begin
* green\_tmp = 1'b0;
* red\_tmp = ~red\_tmp;
* HEX\_1 = 7'b001\_0010; // 5
* HEX\_2 = 7'b000\_1100; // P
* end
* endcase
* end
* assign RED\_LED = red\_tmp ;
* assign GREEN\_LED = green\_tmp;
* endmodule

**//Testbench Verilog code for car parking system:**

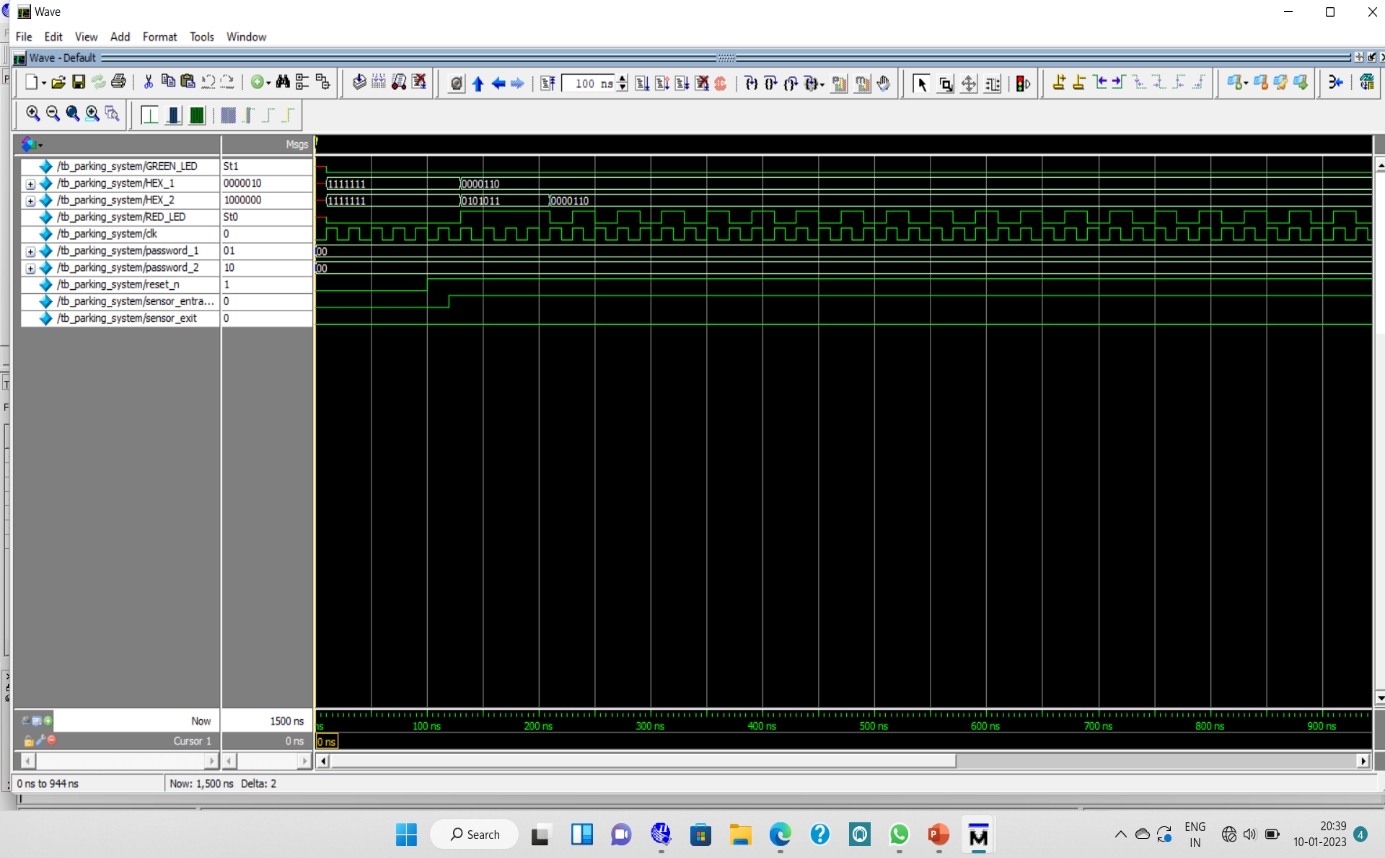
* `timescale 1ns / 1ps
* // fpga4student.com FPGA projects, Verilog projects, VHDL projects
* // Verilog project: Verilog code for car parking system
* module tb\_parking\_system;
* // Inputs
* reg clk;
* reg reset\_n;
* reg sensor\_entrance;
* reg sensor\_exit;
* reg [1:0] password\_1;
* reg [1:0] password\_2;
* // Outputs
* wire GREEN\_LED;
* wire RED\_LED;
* wire [6:0] HEX\_1;
* wire [6:0] HEX\_2;
* // fpga4student.com FPGA projects, Verilog projects, VHDL projects
* // Instantiate the Unit Under Test (UUT)
* parking\_system uut (
* .clk(clk),
* .reset\_n(reset\_n),
* .sensor\_entrance(sensor\_entrance),
* .sensor\_exit(sensor\_exit),
* .password\_1(password\_1),
* .password\_2(password\_2),
* .GREEN\_LED(GREEN\_LED),
* .RED\_LED(RED\_LED),
* .HEX\_1(HEX\_1),
* .HEX\_2(HEX\_2)
* );
* initial begin
* clk = 0;
* forever #10 clk = ~clk;
* end
* initial begin
* // Initialize Inputs
* reset\_n = 0;
* sensor\_entrance = 0;
* sensor\_exit = 0;
* password\_1 = 0;

password\_2 = 0;

* // Wait 100 ns for global reset to finish
* #100;
* reset\_n = 1;
* #20;
* sensor\_entrance = 1;
* #1000;
* sensor\_entrance = 0;
* password\_1 = 1;
* password\_2 = 2;
* #2000;
* sensor\_exit =1;
* // Add stimulus here
* // fpga4student.com FPGA projects, Verilog projects, VHDL projects
* end

endmodule

SIMULATED RESULT:



CONCLUSION…

This design aimed to solve the issue of car parking system by proposing a simulation-based approach with the feature of identifying the availability for multiple slots. The system caters to a wide range of issues. Device utilization was found to be much lower in comparison to the existing models. It isa safe and cost-effective personalized parking solution that focuses on the fundamental principle of saving time while parking. Because the majority of the operation is hardware-based, maintaining the network of incoming and outgoing cars is also relatively simple. The concept also promotes quick parking.