

طراحی سیستم های دیجیتال

پروژه

امیر محمد جزایری

۴۰۱۱۰۵۷۸۶

## الف)

ابتدا باید **module** را طراحی کنیم که طراحی آن به شکل زیر است:

```
module parking_management_system (  
    input wire clk,  
    input wire reset,  
    input wire car_entered,  
    input wire car_exited,  
    input wire is_uni_car_entered,  
    input wire is_uni_car_exited,  
    output reg [9:0] uni_parked_car, // Number of university cars currently  
parked  
    output reg [9:0] parked_car, // Number of non-university cars currently  
parked  
    output reg [9:0] uni_vacated_space, // Number of vacated spaces reserved for  
university cars  
    output reg [9:0] vacated_space, // Number of vacated spaces for non-  
university cars  
    output reg uni_is_vacated_space, // indicates if there is a vacated space for  
university cars  
    output reg is_vacated_space // Indicates if there is a vacated space for  
non-university cars  
);  
  
    parameter MAX_PARKING_SPACE = 700; // Total maximum parking spaces available  
    parameter MAX_UNI_SPACE = 500; // Maximum parking spaces reserved for  
university cars  
    parameter CLK_FREQ = 100_000_000; // Frequency of the clock signal (in Hz)  
    parameter NON_UNI_BASE_SPACE = 200; // Initial base space reserved for non-  
university cars  
  
    // these regs are used to adjust the non_uni_space  
    reg [31:0] elapsed_time_cycles;  
    reg [9:0] non_uni_space;  
    reg [3:0] time_threshold;  
  
    always @(posedge clk or posedge reset) begin  
        if (reset) begin  
            elapsed_time_cycles <= 0;  
            time_threshold <= 0;  
            uni_parked_car <= 0;  
            parked_car <= 0;  
            uni_vacated_space <= MAX_UNI_SPACE;  
            vacated_space <= NON_UNI_BASE_SPACE;  
        end  
    end
```

```

        non_uni_space <= NON_UNI_BASE_SPACE;
        uni_is_vacated_space <= 1;
        is_vacated_space <= 1;
    end else begin
        elapsed_time_cycles <= elapsed_time_cycles + 1;

        // Update time threshold based on elapsed time
        if (elapsed_time_cycles >= CLK_FREQ * 300 * 60) begin
            time_threshold <= 4;
        end else if (elapsed_time_cycles >= CLK_FREQ * 240 * 60) begin
            time_threshold <= 3;
        end else if (elapsed_time_cycles >= CLK_FREQ * 180 * 60) begin
            time_threshold <= 2;
        end else if (elapsed_time_cycles >= CLK_FREQ * 120 * 60) begin
            time_threshold <= 1;
        end else begin
            time_threshold <= 0;
        end

        // Adjust non-uni space allocation based on time threshold
        if (time_threshold == 0) begin
            non_uni_space <= NON_UNI_BASE_SPACE;
        end else if (time_threshold == 1) begin
            non_uni_space <= 250;
        end else if (time_threshold == 2) begin
            non_uni_space <= 300;
        end else if (time_threshold == 3) begin
            non_uni_space <= 350;
        end else if (time_threshold == 4) begin
            non_uni_space <= MAX_UNI_SPACE;
        end

        // Handle university car entered
        if (car_entered && is_uni_car_entered) begin
            if (uni_parked_car < MAX_UNI_SPACE && uni_parked_car + parked_car
< MAX_PARKING_SPACE) begin
                uni_parked_car <= uni_parked_car + 1;
                uni_vacated_space <= uni_vacated_space - 1;
            end
            // Directly implement the logic of update_space_availability
            if (uni_parked_car < MAX_UNI_SPACE && uni_parked_car + parked_car
< MAX_PARKING_SPACE) begin
                uni_is_vacated_space <= 1;
            end else begin
                uni_is_vacated_space <= 0;
            end
        end
    end
end

```

```

        end
        if (parked_car < non_uni_space && uni_parked_car + parked_car <
MAX_PARKING_SPACE) begin
            is_vacated_space <= 1;
        end else begin
            is_vacated_space <= 0;
        end
        // Handle university car exited
    end else if (car_exited && is_uni_car_exited) begin
        if (uni_parked_car > 0) begin
            uni_parked_car <= uni_parked_car - 1;
            uni_vacated_space <= uni_vacated_space + 1;
            uni_is_vacated_space <= 1;
        end
        // Handle non-university car entered
    end else if (car_entered && !is_uni_car_entered) begin
        if (parked_car + uni_parked_car < MAX_PARKING_SPACE && parked_car
< non_uni_space) begin
            parked_car <= parked_car + 1;
            vacated_space <= vacated_space - 1;
        end
        // Directly implement the logic of update_space_availability
        if (uni_parked_car < MAX_UNI_SPACE && uni_parked_car + parked_car
< MAX_PARKING_SPACE) begin
            uni_is_vacated_space <= 1;
        end else begin
            uni_is_vacated_space <= 0;
        end
        if (parked_car < non_uni_space && uni_parked_car + parked_car <
MAX_PARKING_SPACE) begin
            is_vacated_space <= 1;
        end else begin
            is_vacated_space <= 0;
        end
        // Handle non-university car exited
    end else if (car_exited && !is_uni_car_exited) begin
        if (parked_car > 0) begin
            parked_car <= parked_car - 1;
            vacated_space <= vacated_space + 1;
            is_vacated_space <= 1;
        end
    end
end
end
end
endmodule

```

حالا باید با استفاده از تست بنچ، ماژول طراحی شده را تست کنیم.

من تست بنچ زیر را برای همین موضوع طراحی کرده ام:

```
module tb_parking_management_system;

    // Inputs
    reg clk;
    reg reset;
    reg car_entered;
    reg car_exited;
    reg is_uni_car_entered;
    reg is_uni_car_exited;

    // Outputs
    wire [9:0] uni_parked_car;
    wire [9:0] parked_car;
    wire [9:0] uni_vacated_space;
    wire [9:0] vacated_space;
    wire uni_is_vacated_space;
    wire is_vacated_space;

    // Clock period definition
    parameter CLK_PERIOD = 10; // 10 ns

    // Instantiate the DUT
    parking_management_system dut (
        .clk(clk),
        .reset(reset),
        .car_entered(car_entered),
        .car_exited(car_exited),
        .is_uni_car_entered(is_uni_car_entered),
        .is_uni_car_exited(is_uni_car_exited),
        .uni_parked_car(uni_parked_car),
        .parked_car(parked_car),
        .uni_vacated_space(uni_vacated_space),
        .vacated_space(vacated_space),
        .uni_is_vacated_space(uni_is_vacated_space),
        .is_vacated_space(is_vacated_space)
    );

    // Clock generation
    always #CLK_PERIOD clk = ~clk;
```

```
// Initial conditions and test scenario
initial begin
    // Initialize inputs
    clk = 0;
    reset = 1;
    car_entered = 0;
    car_exited = 0;
    is_uni_car_entered = 0;
    is_uni_car_exited = 0;

    // Wait for some time after reset
    #100;

    // Release reset
    reset = 0;

    // Test scenario 1: University car enters
    $display("Action 1: University car enters");
    is_uni_car_entered = 1;
    car_entered = 1;
    #20;
    car_entered = 0;
    is_uni_car_entered = 0;
    #100;

    // Test scenario 2: Non-university car enters
    $display("Action 2: Non-university car enters");
    is_uni_car_entered = 0;
    car_entered = 1;
    #20;
    car_entered = 0;
    is_uni_car_entered = 0;
    #100;

    // Test scenario 3: University car exits
    $display("Action 3: University car exits");
    is_uni_car_exited = 1;
    car_exited = 1;
    #20;
    car_exited = 0;
    is_uni_car_exited = 0;
    #100;

    // Test scenario 4: Non-university car exits
```

```

$display("Action 4: Non-university car exits");
is_uni_car_exited = 0;
car_exited = 1;
#20;
car_exited = 0;
is_uni_car_exited = 0;
#100;

// Add more test scenarios as needed

// Finish simulation
#1000;
$finish;
end

endmodule

```

همانطور که قابل مشاهده است ۴ حالت مختلف برای پارکینگ پیش بینی و در این تست پنج طراحی شده است که نتیجه آن به شکل زیر می شود.

```

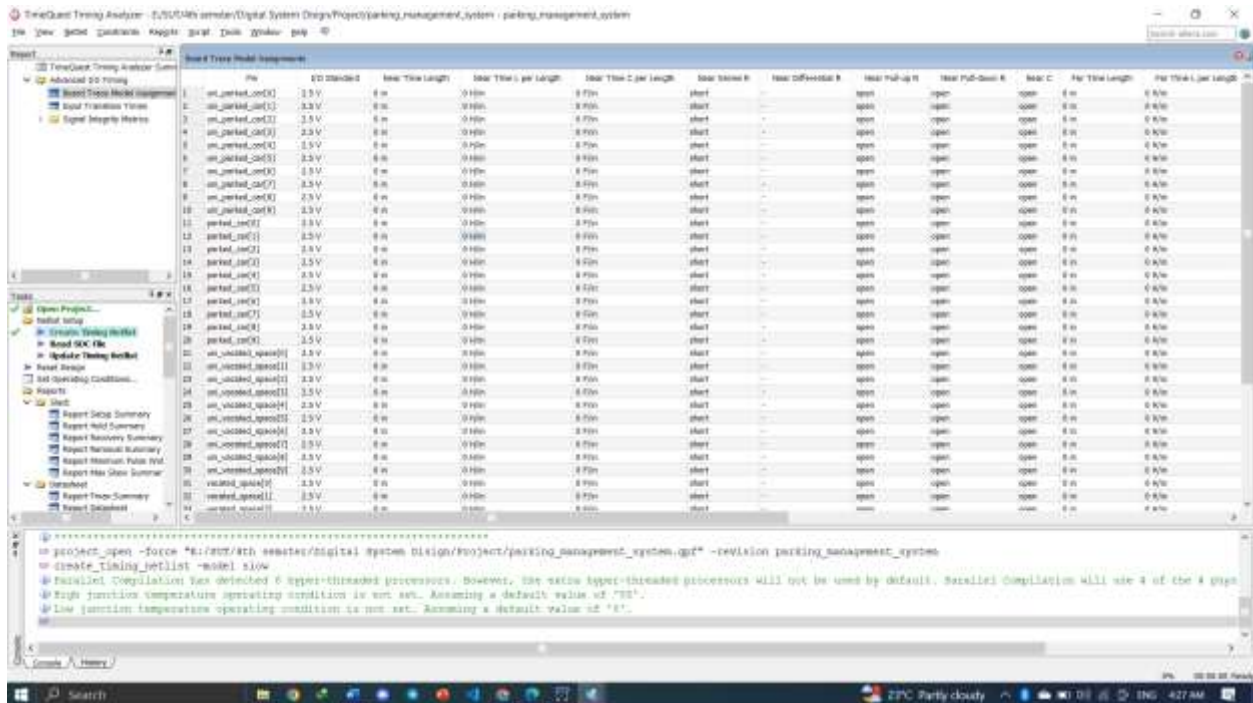
VSIM9> run -all
# Action 1: University car enters
# Action 2: Non-university car enters
# Action 3: University car exits
# Action 4: Non-university car exits
# ** Note: $finish      : E:/SUT/4th semster/Digital System Design/Project/TB.v(99)
#   Time: 1580 ns  Iteration: 0  Instance: /tb_parking_management_system

```

(ب)

برای سنتز کردن ابتدا فایل وریلاگ مازول درست شده رو در کوارتوس باز می کنیم و سپس در لیست دیوایس ها، **Cydone IV GX** را انتخاب می کنیم و سپس کامپایل می کنیم.

بعد از آن باید ابزار **Time Quest Analyzer** را باز کنیم و در آن یک **Timing Netlist** بسازیم و پس از ساخته شدن به نتیجه زیر میرسیم:



پس از این مرحله، باید Read STC File و بعد از آن، Update Timing Netlist را انجام دهیم و سپس در بین گزارش ها Report Fmax Summary را باز می کنیم که به فرکانس ماکس ایجاد شده را خروجی می دهد:

Fmax Summary				
	Fmax	Restricted Fmax	Clock Name	Note
1	202.27 MHz	202.27 MHz	clk	