طراحی سیستم های دیجیتال

پروژه

امیرمحمد جزایری

401105786

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;

            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

                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

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*

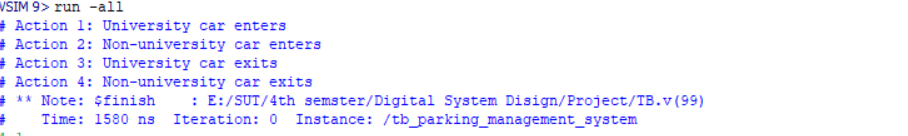
        #1000;

        $finish;

    end

endmodule

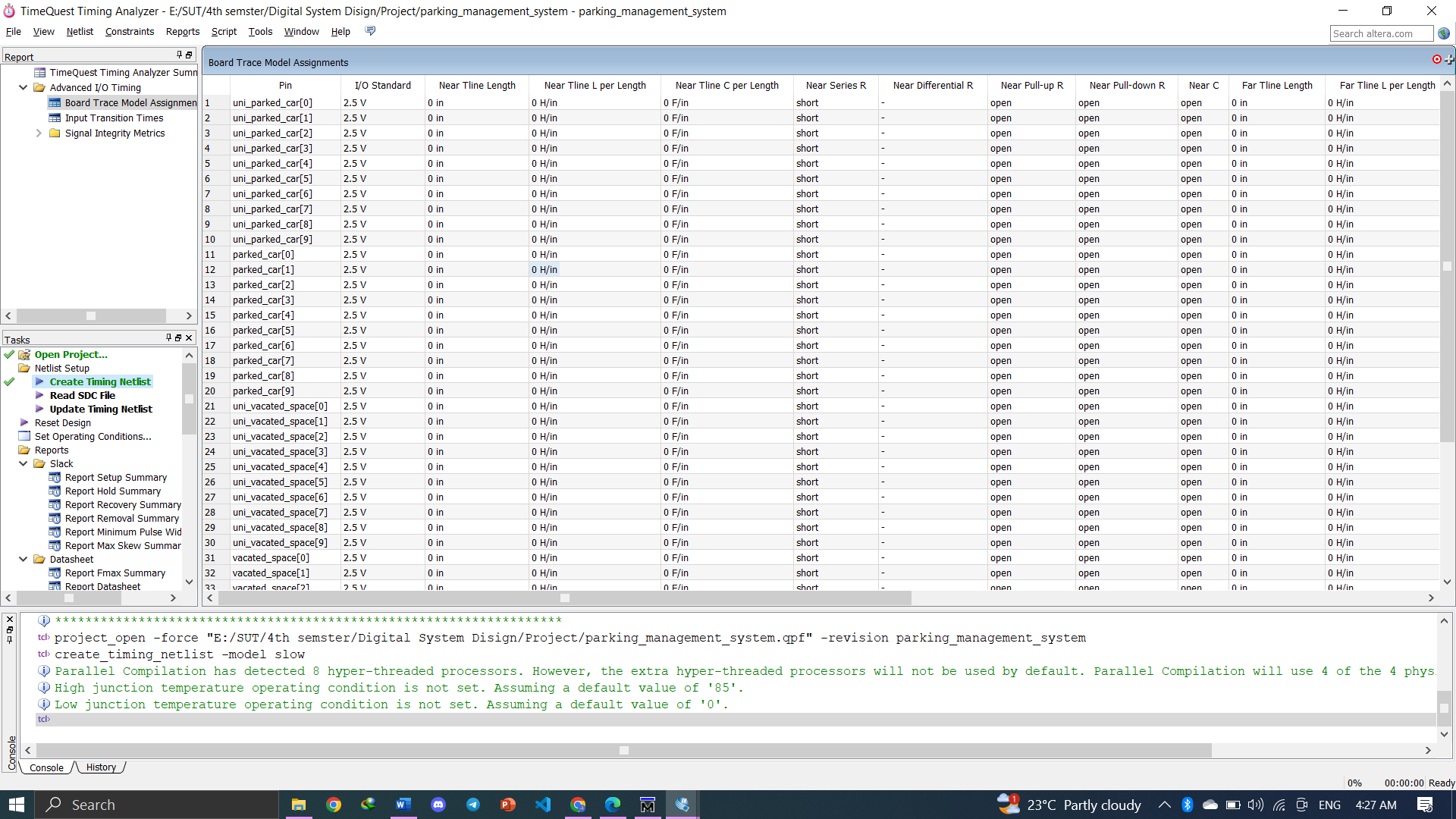
**همانطور که قابل مشاهده است 4 حالت مختلف برای پارکینگ پیش بینی و در این تست بنچ طراحی شده است که نتیجه آن به شکل زیر می شود.**

****

ب)

**برای سنتز کردن ابتدا فایل وریلاگ ماژول 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 ار 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:

