مداری برای مدیریت پارکینگ دانشگاه طراحی می‌کنیم که امکانات زیر را داشته باشد:

* اولویت فضاي پارکینگ با اساتید و کارمندان دانشگاه است و این ظرفیت حداکثر ۵۰۰ خودرو است.
* فضاي کل پارکینگ ۷۰۰ خودرو است و از ساعت ۸ تا ۱۳ تنها ۲۰۰ ظرفیت خالی براي ورود آزاد است.
* از ساعت ۱۳ تا ۱۶ به ازاي هر ساعت، ظرفیت ورود آزاد ۵۰ خودرو افزایش می‌یابد و در ساعت ۱۶، ظرفیت ورود آزاد به ۵۰۰ خودرو می‌رسد.

ماژولی به زبان وریلاگ برای این مدار طراحی می‌کنیم. این ماژول دارای ورودی‌ها و خروجی‌های زیر است:

|  |  |
| --- | --- |
| ورودی‌ها | |
| ورود یک خودرو | car\_entered |
| آیا خودرو وارد شده متعلق به دانشگاه است؟ | is\_uni\_car\_entered |
| خروج یک خودرو | car\_exited |
| آیا خودرو خارج شده متعلق به دانشگاه است؟ | is\_uni\_car\_exited |
| خروجی‌ها | |
| ساعت کنونی | hour |
| تعداد خودروهایی متعلق به دانشگاه که در پارکینگ پارك شده اند. | uni\_parked\_cars |
| تعداد خودروهاي پارك شده در پارکینگ مربوط به ظرفیت آزاد | parked\_cars |
| تعداد فضاي خالی متعلق به دانشگاه | uni\_vacated\_space |
| تعداد فضاهاي خالی مربوط به ظرفیت آزاد | vacated\_space |
| آیا فضاي خالی براي دانشگاه موجود است؟ | uni\_is\_vacated\_space |
| آیا فضاي خالی براي ظرفیت آزاد موجود است؟ | is\_vacated\_space |

الف)

اولین بلوک always که با کلاک کار می‌کند، برای مدیریت ورود و خروج ماشین‌ها و محاسبه‌ی تعداد ماشین‌های پارک‌شده‌ی مربوط به دانشگاه و آزاد است. مکانیزم ریست در این بلوک وجود دارد که با سیگنال ریست یا رسیدن ساعت به ۲۴ کار می‌کند و ساعت را روی ۸ (آغاز کار پارکینگ) تنظیم می‌کند. در غیر این صورت، با گذشتن هر ده کلاک، یک ساعت می‌گذرد و در هر کلاک، یک ماشین می‌تواند وارد یا خارج شود. با توجه به ورودی is\_uni\_car\_entered یا is\_uni\_car\_exited تشخیص می‌دهیم که ظرفیت پر کدام دسته را تغییر دهیم. در بلوک always دوم، با توجه به زمان، ظرفیت کل دانشگاه و آزاد تعیین می‌شود. در نهایت، در بلوک آخر، ظرفیت خالی هر دو دسته از ماشین‌ها محاسبه می‌شود. در طراحی این ماژول، دقت شده که هیچ متغیری در بیش از یک بلوک، مقداردهی نشود.

module parking (

    input clk, reset, car\_entered, is\_uni\_car\_entered, car\_exited, is\_uni\_car\_exited,

    output reg [9:0] uni\_parked\_cars, parked\_cars, uni\_vacated\_space, vacated\_space,

    output reg uni\_is\_vacated\_space, is\_vacated\_space,

    output reg [4:0] hour

    );

    reg [9:0] uni\_capacity, capacity;

    reg [7:0] clocks;

    always @(posedge clk or negedge reset) begin

        if (!reset || (hour == 5'd24)) begin

            uni\_parked\_cars <= 0;

            parked\_cars <= 0;

            clocks <= 0;

            hour <= 5'd8;

        end

        else begin

            clocks <= clocks + 8'b1;

            if ((clocks % 8'd10) == 0)

                hour <= hour + 5'b1;

            if (car\_entered) begin

                if (is\_uni\_car\_entered && (uni\_parked\_cars < uni\_capacity))

                    uni\_parked\_cars <= uni\_parked\_cars + 10'b1;

                else if (!is\_uni\_car\_entered && (parked\_cars < capacity))

                    parked\_cars <= parked\_cars + 10'b1;

            end

            if (car\_exited) begin

                if (is\_uni\_car\_exited && (uni\_parked\_cars > 0))

                    uni\_parked\_cars <= uni\_parked\_cars - 10'b1;

                else if (!is\_uni\_car\_exited && (parked\_cars > 0))

                    parked\_cars <= parked\_cars - 10'b1;

            end

        end

    end

    always @(\*) begin

        if (hour < 5'd13) begin

            uni\_capacity = 10'd500;

            capacity = 10'd200;

        end

        else if (hour < 5'd14) begin

            uni\_capacity = 10'd450;

            capacity = 10'd250;

        end

        else if (hour < 5'd15) begin

            uni\_capacity = 10'd400;

            capacity = 10'd300;

        end

        else if (hour < 5'd16) begin

            uni\_capacity = 10'd350;

            capacity = 10'd350;

        end

        else begin

            uni\_capacity = 10'd200;

            capacity = 10'd500;

        end

    end

    always @(\*) begin

        uni\_vacated\_space = uni\_capacity - uni\_parked\_cars;

        vacated\_space = capacity - parked\_cars;

        uni\_is\_vacated\_space = (uni\_vacated\_space > 0);

        is\_vacated\_space = (vacated\_space > 0);

    end

endmodule

حال، به برنامه‌ی آزمون می‌پردازیم. این برنامه، در هر ساعت از ۸ تا ۱۳، پنج ماشین دانشگاه و یک ماشین آزاد را وارد، و یک ماشین دانشگاه را خارج می‌کند. سپس به ساعت ۱۴ می‌پرد و هشت ماشین آزاد را در این ساعت وارد می‌کند. پس از آن، به ساعت ۱۶ می‌پرد و تا ساعت ۲۳، سه ماشین دانشگاه و شش ماشین آزاد را خارج می‌کند.

module parking\_tb;

    reg clk, reset, car\_entered, is\_uni\_car\_entered, car\_exited, is\_uni\_car\_exited;

    wire [9:0] uni\_parked\_cars, parked\_cars, uni\_vacated\_space, vacated\_space;

    wire uni\_is\_vacated\_space, is\_vacated\_space;

    wire [4:0] hour;

    parking uut (

        clk, reset, car\_entered, is\_uni\_car\_entered, car\_exited, is\_uni\_car\_exited,

        uni\_parked\_cars, parked\_cars, uni\_vacated\_space, vacated\_space,

        uni\_is\_vacated\_space, is\_vacated\_space, hour

    );

    integer i;

    initial begin

        clk = 0;

        reset = 0;

        car\_entered = 0;

        is\_uni\_car\_entered = 0;

        car\_exited = 0;

        is\_uni\_car\_exited = 0;

        #10 reset = 1;

        #5;

        // 8-13 o'clock

        for (i = 0; i < 5; i = i + 1) begin

            #10 car\_entered = 1; is\_uni\_car\_entered = 1; // Faculty car enters

            #50 car\_entered = 1; is\_uni\_car\_entered = 0; // Public car enters

            #10 car\_entered = 0; car\_exited = 1; is\_uni\_car\_exited = 1; // Faculty car exits

            #10 car\_exited = 0;

        end

        #100; // Skip to 14 o'clock

        #10 car\_entered = 1; is\_uni\_car\_entered = 0; // Public car enters

        #90 car\_entered = 0;

        #100; // Skip to 16 o'clock

        for (i = 0; i < 7; i = i + 1) begin

            #10 car\_exited = 1; is\_uni\_car\_exited = 1; // Faculty car exits

            #30 car\_exited = 1; is\_uni\_car\_exited = 0; // Public car exits

            #60 car\_exited = 0;

        end

        // Now: 23 o'clock

        #110; // Skip to 24 o'clock

        #10; // Back to 8 o'clock

        $stop;

    end

    always #5 clk = ~clk;

    initial begin

        $monitor (

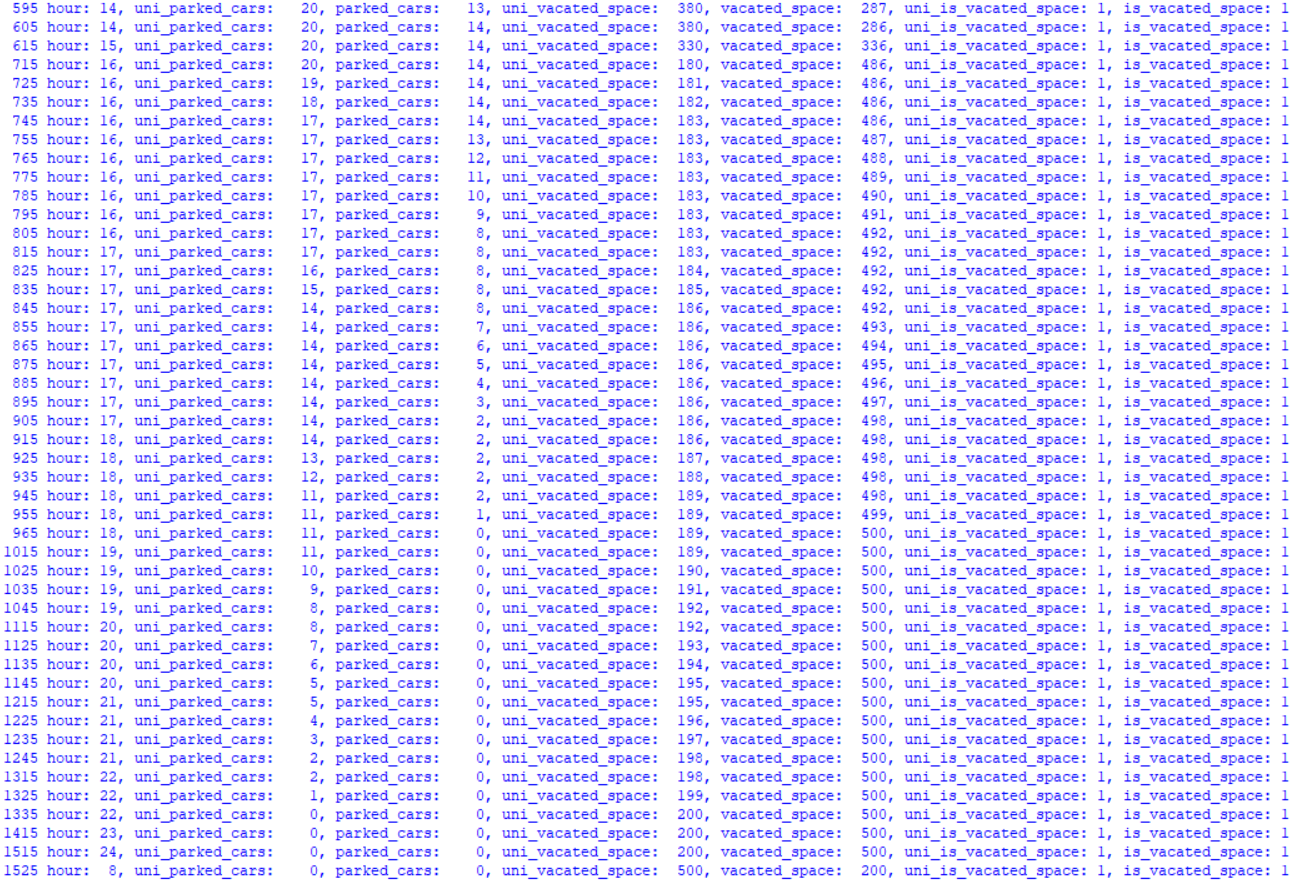
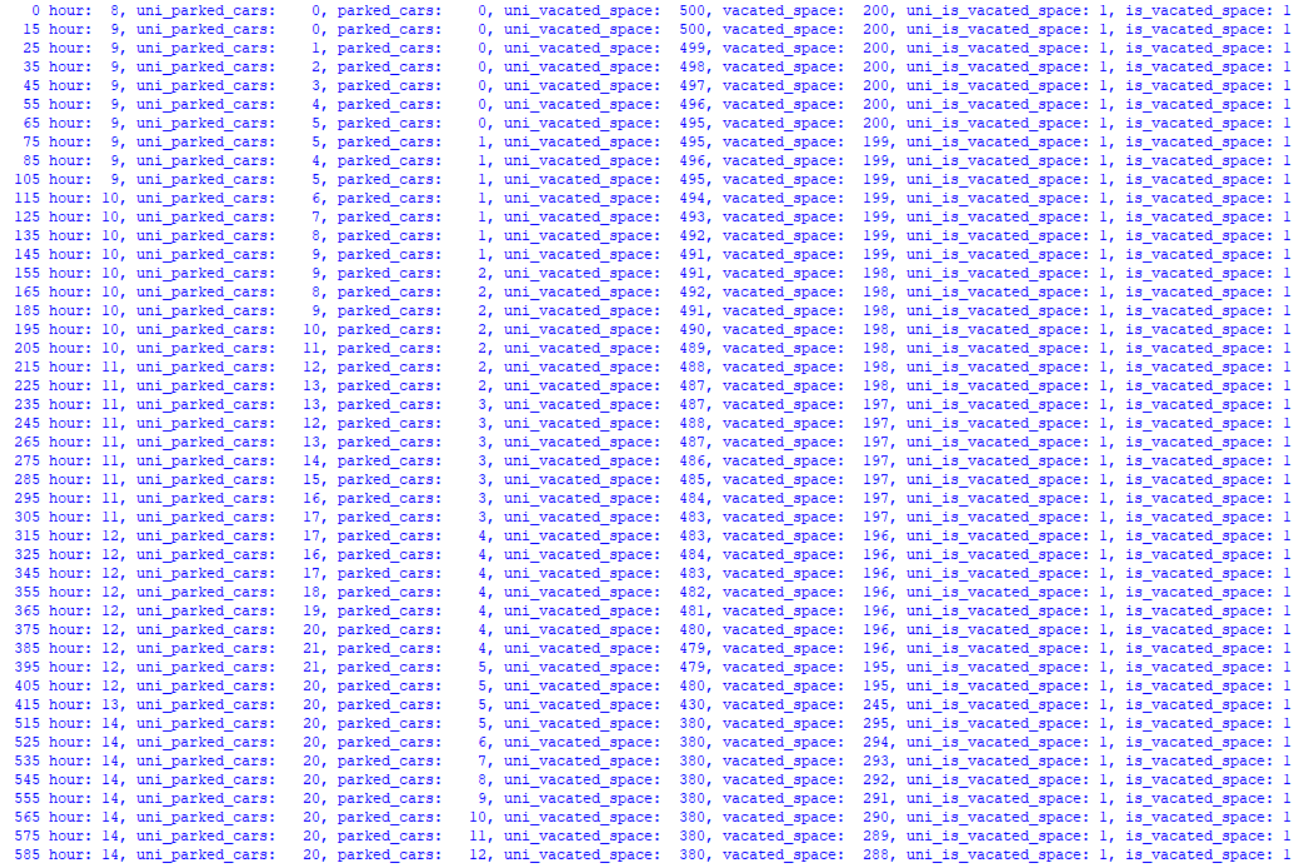
            $time, " hour: %d, uni\_parked\_cars: %d, parked\_cars: %d, uni\_vacated\_space: %d, vacated\_space: %d, uni\_is\_vacated\_space: %d, is\_vacated\_space: %d",

            hour, uni\_parked\_cars, parked\_cars, uni\_vacated\_space, vacated\_space, uni\_is\_vacated\_space, is\_vacated\_space

        );

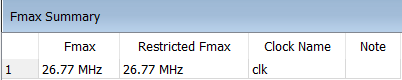
    end

endmodule



ب)

برای سنتز، فایل وریلاگ را در برنامه‌ی کوارتوس کامپایل می‌کنیم و با ابزار TimeQuest Timing Analyzer، سقف نرح کلاک را حدود ۲۶ مگاهرتز به‌دست می‌آوریم.



علت این نرخ پایین، می‌تواند وجود عملیات پیچیده در هر چرخه کلاک باشد؛ مانند بررسی شرط با استفاده از باقی‌مانده گرفتن، جمع و تفریق و به‌روزرسانی مقادیر رجیسترها در تنها یک چرخه کلاک.