

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

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

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

ورودی‌ها	
car_entered	ورود یک خودرو
is_uni_car_entered	آیا خودرو وارد شده متعلق به دانشگاه است؟
car_exited	خروج یک خودرو
is_uni_car_exited	آیا خودرو خارج شده متعلق به دانشگاه است؟
خروجی‌ها	
hour	ساعت کنونی
uni_parked_cars	تعداد خودروهایی متعلق به دانشگاه که در پارکینگ پارک شده‌اند.
parked_cars	تعداد خودروهای پارک شده در پارکینگ مربوط به ظرفیت آزاد
uni_vacated_space	تعداد فضای خالی متعلق به دانشگاه
vacated_space	تعداد فضاهای خالی مربوط به ظرفیت آزاد
uni_is_vacated_space	آیا فضای خالی برای دانشگاه موجود است؟
is_vacated_space	آیا فضای خالی برای ظرفیت آزاد موجود است؟

(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
```

```

        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

```

[illegible]

(ب)

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

Fmax Summary				
	Fmax	Restricted Fmax	Clock Name	Note
1	26.77 MHz	26.77 MHz	clk	

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.