پروژه امتیازی درس طراحی سیستم‌های دیجیتال

دانشگاه صنعتی شریف، نیمسال دوم ۰۲-۰۳

– استاد : دکتر امین فصحتی –

– دانشجو : رادین شاه دائی –

– شماره دانشجویی: ۴۰۱۱۰۶۰۹۶ –

– شرح پروژه –

در این پروژه، با استفاده از زبان Verilog VHDL، یک سیستم مدیریت پارکینگ طراحی شده‌است. فضای پارکینگ دانشگاه به گونه‌ای طراحی شده است که اولویت در تخصیص فضا به اساتید و کارمندان دانشگاه داده شود. این تصمیم بر اساس نیاز و اهمیت دسترسی راحت‌تر اساتید و کارمندان به دانشگاه اتخاذ شده است. بر اساس آمار، حداکثر 500 خودرو می‌توانند در این بخش از پارکینگ جا بگیرند. این بخش از پارکینگ مخصوص افرادی است که به صورت مداوم و روزانه در دانشگاه حضور دارند و باید به سرعت و بدون مشکل به محل کار خود دسترسی داشته باشند. با توجه به اینکه فضای کل پارکینگ دانشگاه ظرفیت پذیرش 700 خودرو را دارد، مدیریت این فضا باید به نحوی باشد که هم نیازهای کارکنان و اساتید دانشگاه و هم نیازهای دانشجویان و سایر بازدیدکنندگان را برآورده کند. در ساعات ابتدایی روز، یعنی از ساعت 8 تا 13، تنها 200 ظرفیت خالی برای ورود آزاد وجود دارد. این موضوع به این دلیل است که در این ساعات معمولاً بیشترین حجم تردد مربوط به اساتید و کارمندان است و باید فضایی مناسب برای آنها فراهم باشد. یکی از ویژگی‌های مهم این سیستم مدیریت پارکینگ، تغییرات ظرفیت ورود آزاد در طول روز است. از ساعت 13 تا 16، به ازای هر ساعت، ظرفیت ورود آزاد 50 خودرو افزایش می‌یابد. این به این معناست که تا ساعت 16، ظرفیت ورود آزاد به 500 خودرو می‌رسد. این تغییرات به دلیل کاهش نیاز اساتید و کارمندان به فضای پارکینگ در ساعات بعد از ظهر و افزایش تردد دانشجویان و بازدیدکنندگان دیگر در این ساعات در نظر گرفته شده است.

– ابزارهای استفاده شده –

ابزار iverilog:

Icarus Verilog یکی از ابزارهای متن‌باز برای شبیه‌سازی کدهای Verilog است. از این ابزار برای تست و شبیه‌سازی کدهای نوشته شده در Verilog استفاده شده است. این ابزار به ما این امکان را می‌دهد تا بدون نیاز به ابزارهای تجاری گران‌قیمت، به راحتی کدهای خود را شبیه‌سازی کرده و صحت عملکرد آن‌ها را بررسی کنیم. با استفاده ازIverilog ، شبیه‌سازی مدارهای نوشته شده در Verilog انجام شده و نتایج خروجی برای بررسی و تحلیل عملکرد مدار مورد استفاده قرار گرفته است.

ابزار Quartus Prime:

Quartus Prime برای طراحی و پیاده‌سازی مدارهای FPGA استفاده می‌شود. این ابزار امکانات گسترده‌ای برای طراحی، شبیه‌سازی و پیاده‌سازی مدارهای دیجیتال ارائه می‌دهد. در این پروژه، از Quartus Prime برای سنتز و پیاده‌سازی نهایی مدار طراحی شده بر روی تراشه‌های FPGA استفاده شده است. Quartus Prime به ما امکان می‌دهد تا کدهای Verilog نوشته شده را به سخت‌افزار واقعی تبدیل کنیم و عملکرد آن‌ها را بر روی بردهای FPGA تست کنیم.

- شرح کد وریلاگ -

بخش پارامترها و ورودی / خروجی‌ها

```
module parking #(
    // Define the parameters for initial and final university parking spaces,
    // total parking spaces, and increment value.
    parameter init_uni_space = 500,
    parameter final_uni_space = 200,
    parameter total_space = 700,
    parameter increment = 50
)(
    // Define the inputs for car entered/exited events and whether it is a university
car.
    input car_entered,
    input is_uni_car_entered,
    input car_exited,
    input is_uni_car_exited,
    input [5:0] hour,  // Define the input for the current hour.

    // Define the outputs for the count of parked cars and available spaces.
    output reg [9:0] uni_parked_car = 0,
    output reg [9:0] parked_car = 0,
    output reg [9:0] uni_vacated_space = init_uni_space,
    output reg [9:0] vacated_space = total_space - init_uni_space,
    output reg uni_is_vacated_space = 1,
    output reg is_vacated_space = 1
);

reg [9:0] uni_total_space;  // Internal register to hold the total university space at
a given hour.
```

در این بخش، پارامترها و ورودی/خروجی‌های ماژول تعریف می‌شوند. پارامترها شامل چهار مقدار مهم هستند که به صورت متغیرهای قابل تنظیم در ماژول تعریف شده‌اند. پارامتر init_uni_space ظرفیت اولیه پارکینگ دانشگاهی را مشخص می‌کند که مقدار پیش‌فرض آن 500 است، به این معنا که در ابتدای روز، 500 فضای پارکینگ به دانشگاه اختصاص داده شده است. پارامتر final_uni_space ظرفیت نهایی پارکینگ دانشگاهی را نشان می‌دهد که بعد از ساعت 16 به این مقدار کاهش می‌یابد و مقدار پیش‌فرض آن 200 است. پارامتر total_space کل ظرفیت پارکینگ را نشان می‌دهد که مقدار پیش‌فرض آن 700 است، یعنی پارکینگ به طور کلی ظرفیت پذیرش 700 خودرو را دارد. در نهایت، پارامتر increment مقدار افزایشی است که برای تغییر ظرفیت پارکینگ دانشگاهی در ساعات 13 تا 16 به کار می‌رود و مقدار پیش‌فرض آن 50 است، یعنی هر ساعت از 13 تا 16، ظرفیت پارکینگ دانشگاهی 50 فضای پارک کاهش می‌یابد.

ورودی‌ها شامل سیگنال‌های مختلفی هستند که وضعیت و عملکرد ماژول را کنترل می‌کنند. ورودی car_entered یک سیگنال است که نشان می‌دهد یک خودرو وارد پارکینگ شده است و هر بار که این سیگنال فعال می‌شود، تعداد خودروهای پارک شده افزایش می‌یابد. ورودی is_uni_car_entered نشان می‌دهد که آیا خودروی وارد شده متعلق به دانشگاه است یا خیر. ورودی car_exited سیگنالی است که نشان می‌دهد یک خودرو از پارکینگ خارج شده است و هر بار که این سیگنال فعال می‌شود، تعداد خودروهای پارک شده کاهش می‌یابد. ورودی is_uni_car_exited نشان می‌دهد که آیا خودروی خارج شده متعلق به دانشگاه است یا خیر. ورودی hour نیز ساعت جاری را به صورت یک مقدار 6 بیتی وارد ماژول می‌کند که بر اساس آن، ظرفیت پارکینگ تنظیم می‌شود. خروجی‌ها شامل تعداد خودروهای پارک شده، فضای خالی و وضعیت فضای خالی برای خودروهای دانشگاهی و غیر دانشگاهی هستند.

بخش تعیین ظرفیت پارکینگ دانشگاهی بر اساس ساعت

```
always @(*) begin
    case (hour)
        8, 9, 10, 11, 12: uni_total_space = init_uni_space;
        13: uni_total_space = init_uni_space - increment;
        14: uni_total_space = init_uni_space - 2 * increment;
        15: uni_total_space = init_uni_space - 3 * increment;
        default: uni_total_space = final_uni_space;
    endcase
```

در این بخش، ظرفیت کل پارکینگ دانشگاهی بر اساس ساعت تعیین می‌شود. اگر ساعت بین 8 تا 13 باشد، ظرفیت پارکینگ دانشگاهی برابر با مقدار اولیه یعنی init_uni_space است. این به این معناست که در این بازه زمانی، تعداد فضای پارکینگی که به دانشگاه اختصاص داده شده است ثابت و برابر با ظرفیت اولیه در نظر گرفته شده است. این رویکرد به دلیل نیاز بیشتر اساتید و کارکنان به فضای پارکینگ در ساعات ابتدایی روز اتخاذ شده است.

بین ساعت 13 تا 16، ظرفیت پارکینگ دانشگاهی به ازای هر ساعت کاهش می‌یابد. این کاهش بر اساس پارامتر increment محاسبه می‌شود، به طوری که هر ساعت ظرفیت پارکینگ 50 فضای پارک کاهش می‌یابد. پس از ساعت 16، ظرفیت پارکینگ به مقدار نهایی یعنی final_uni_space کاهش می‌یابد. این تغییرات به دلیل کاهش نیاز اساتید و کارکنان به فضای پارکینگ در ساعات بعد از ظهر و افزایش تردد دانشجویان و بازدیدکنندگان دیگر در این ساعات در نظر گرفته شده است. این تنظیمات زمانی بهینه‌ترین استفاده از فضای پارکینگ را در طول روز فراهم می‌کند.

بخش تنظیم تعداد خودروهای پارک شده و فضای خالی

```
    // Adjust parked cars if the university parked cars exceed the total university
space.
    if (uni_parked_car >= uni_total_space) begin
        parked_car = parked_car + (uni_parked_car - uni_total_space);
        uni_parked_car = uni_total_space;
        vacated_space = total_space - (uni_parked_car + parked_car);
        uni_vacated_space = 0;
    end

    // Update the vacated space status flags.
    uni_is_vacated_space = (uni_parked_car < uni_total_space);
    is_vacated_space = (parked_car < total_space - uni_total_space);
end
```

در این بخش، اگر تعداد خودروهای پارک شده دانشگاهی از ظرفیت کل پارکینگ دانشگاهی بیشتر باشد، خودروهای اضافی به بخش عمومی پارکینگ منتقل می‌شوند. به عبارت دیگر، اگر تعداد خودروهای پارک شده دانشگاهی برابر یا بیشتر از ظرفیت تعیین شده برای آن‌ها باشد، خودروهای اضافی به فضاهای پارک عمومی هدایت می‌شوند. این مکانیزم به ما اطمینان می‌دهد که حتی در صورت پر شدن فضای دانشگاهی، خودروها هنوز هم می‌توانند در پارکینگ عمومی جای پارک پیدا کنند.

همچنین در این بخش، فضای خالی به‌روزرسانی می‌شود و وضعیت فضای خالی بررسی می‌شود. با توجه به تغییرات در تعداد خودروهای پارک شده، مقدار فضای خالی برای هر دسته به‌روزرسانی می‌شود. اگر ظرفیت پارکینگ دانشگاهی به طور کامل پر شده باشد، سیگنال uni_is_vacated_space به صفر تغییر می‌کند تا نشان دهد که هیچ فضای خالی برای خودروهای دانشگاهی موجود نیست. به همین ترتیب، اگر پارکینگ عمومی نیز به طور کامل پر شود، سیگنال is_vacated_space به صفر تغییر می‌کند تا نشان دهد که هیچ فضای خالی برای خودروهای عمومی وجود ندارد. این به‌روزرسانی‌ها به سیستم کمک می‌کنند تا همیشه وضعیت جاری فضای پارکینگ را در دسترس داشته باشد و تصمیمات مناسب را برای مدیریت ترافیک و پارکینگ اتخاذ کند.

بخش مدیریت ورود خودروها

```
// Handle car entered and exited events.
always @(posedge car_entered or posedge car_exited) begin
    if (hour >= 8) begin
        if (car_entered) begin
            handle_car_entered(is_uni_car_entered);
        end
```

```
// Task to handle a car entering the parking lot.
task handle_car_entered(input is_uni_car);
    if (is_uni_car) begin
        if (uni_is_vacated_space) begin
            uni_parked_car = uni_parked_car + 1;
            uni_vacated_space = uni_vacated_space - 1;
        end else if (is_vacated_space) begin
            parked_car = parked_car + 1;
            vacated_space = vacated_space - 1;
        end
    end else begin
        if (is_vacated_space) begin
            parked_car = parked_car + 1;
            vacated_space = vacated_space - 1;
        end
    end
endtask
```

بله، در این بخش، مدیریت ورود خودروها انجام می‌شود. اگر خودروی ورودی دانشگاهی باشد و ساعت بین 8 تا 16 باشد، ابتدا بررسی می‌شود که آیا فضای خالی دانشگاهی موجود است یا نه. در صورت وجود فضای خالی دانشگاهی، خودرو در فضای دانشگاهی پارک می‌شود و تعداد فضای خالی دانشگاهی کاهش می‌یابد. این فرآیند اطمینان می‌دهد که اساتید و کارکنان دانشگاه بتوانند به راحتی فضای پارک خود را پیدا کنند و تردد آن‌ها با مشکل مواجه نشود.

در غیر این صورت، یعنی اگر فضای خالی دانشگاهی موجود نباشد، خودروی دانشگاهی در فضای عمومی پارک می‌شود. برای خودروهای غیر دانشگاهی نیز مشابه این روند بررسی می‌شود. اگر فضای خالی عمومی وجود داشته باشد، خودرو در فضای عمومی پارک می‌شود و تعداد فضای خالی عمومی کاهش می‌یابد. این مدیریت دقیق ورود خودروها کمک می‌کند تا استفاده بهینه‌ای از تمامی فضاهای پارکینگ صورت گیرد و از بروز مشکلات احتمالی جلوگیری شود.

بخش مدیریت خروج خودروها

```
        if (car_exited) begin
            handle_car_exited(is_uni_car_exited);
        end

        // Update the vacated space status flags.
        uni_is_vacated_space = (uni_parked_car < uni_total_space);
        is_vacated_space = (parked_car < total_space - uni_total_space);
    end
end
```

```
// Task to handle a car exiting the parking lot.
task handle_car_exited(input is_uni_car);
    if (is_uni_car && (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 else if (!is_uni_car && (parked_car > 0)) begin
        parked_car = parked_car - 1;
        vacated_space = vacated_space + 1;
        is_vacated_space = 1;
    end
endtask
```

در این بخش، مدیریت خروج خودروها انجام می‌شود. اگر خودروی خروجی دانشگاهی باشد و تعداد خودروهای پارک شده دانشگاهی بیشتر از 0 باشد، تعداد خودروهای پارک شده دانشگاهی کاهش یافته و فضای خالی دانشگاهی افزایش می‌یابد. این فرآیند کمک می‌کند تا فضای پارکینگ دانشگاهی برای استفاده‌های بعدی آماده باشد و اطلاعات مربوط به تعداد خودروهای پارک شده به‌روز باقی بماند.

برای خودروهای غیر دانشگاهی نیز مشابه این روند بررسی می‌شود. اگر خودروی خروجی غیر دانشگاهی باشد و تعداد خودروهای پارک شده در فضای عمومی بیشتر از 0 باشد، تعداد خودروهای پارک شده عمومی کاهش یافته و فضای خالی عمومی افزایش می‌یابد. در نهایت، وضعیت فضای خالی برای هر دو نوع پارکینگ (دانشگاهی و عمومی) به‌روزرسانی می‌شود تا سیستم مدیریت پارکینگ بتواند به‌طور دقیق و به‌موقع به تغییرات پاسخ دهد و ترافیک را به بهترین نحو ممکن مدیریت کند.

– شرح کد تست‌بنچ –

در طراحی فایل بستر آزمون، از task استفاده شده‌است، بدین ترتیب که برای ۳ عملیات ورود به پارکینگ، خروج از پارکینگ و تغییر زمان task تعریف شده‌است که با انجام این عملیات‌ها پشت سر هم، می‌توانیم ماژول طراحی شده را به طور کامل تست کنیم.

```
// Task to handle car entering
task test_enter_car(input is_uni, input [25*8:1] op);
begin
    car_entered = 1;
    is_uni_car_entered = is_uni;
    operation = op;
    #5 car_entered = 0;
    #5;
end
endtask
```

در test_enter_car، عملیات ورود خودرو به پارکینگ مدیریت می‌شود. این وظیفه دو ورودی دارد is_uni: که نشان می‌دهد آیا خودروی ورودی متعلق به دانشگاه است یا خیر، و op که پیامی توضیحی برای عملیات فعلی است. این وظیفه سیگنال car_entered را فعال کرده و سپس بر اساس نوع خودرو، مقدار is_uni_car_entered را تنظیم می‌کند. پیام توضیحی نیز به operation اختصاص داده می‌شود. پس از 5 واحد زمانی، سیگنال car_entered غیر فعال می‌شود تا عملیات ورود خودرو کامل شود.

```
// Task to handle car exiting
task test_exit_car(input is_uni, input [25*8:1] op);
begin
    car_exited = 1;
    is_uni_car_exited = is_uni;
    operation = op;
    #5 car_exited = 0;
    #5;
end
endtask
```

در test_exit_car، عملیات خروج خودرو از پارکینگ مدیریت می‌شود. این وظیفه نیز دو ورودی دارد is_uni: که نشان می‌دهد آیا خودروی خروجی متعلق به دانشگاه است یا خیر، و op که پیامی توضیحی برای عملیات فعلی است. این وظیفه سیگنال car_exited را فعال کرده و سپس بر اساس نوع خودرو، مقدار is_uni_car_exited را تنظیم می‌کند. پیام توضیحی نیز به operation اختصاص داده می‌شود. پس از 5 واحد زمانی، سیگنال car_exited غیر فعال می‌شود تا عملیات خروج خودرو کامل شود.

```
// Task to handle change of time
task change_time(input [5:0] in_hour, input [25*8:1] op);
begin
    hour = in_hour;
    operation = op;
    #10;
end
endtask
```

در change_time، عملیات تغییر زمان مدیریت می‌شود. این وظیفه دو ورودی دارد: in_hour که ساعت جدید را تعیین می‌کند، و op که پیامی توضیحی برای عملیات فعلی است. این وظیفه مقدار hour را بر اساس ورودی in_hour تنظیم کرده و پیام توضیحی را به operation اختصاص می‌دهد. پس از 10 واحد زمانی، تغییر زمان انجام می‌شود و عملیات تکمیل می‌شود.

دقت کنید که برای تست این ماژول، از پارامترهای کوچک‌تری استفاده شده‌است تا بتوان به سادگی ماژول parking.v را تست کرد و از صحت آن مطمئن شد.

```
// Parameters for the testbench
parameter init_uni_space = 5;
parameter final_uni_space = 2;
parameter total_space = 10;
parameter increment = 1;
```

در ادامه، عملیات انجام شده در فایل بستر آزمون را مشاهده می‌کنید.

```
    change_time(9, "Time changed to 9");
    // Test entering and exiting cars in various scenarios
    test_enter_car(1, "Uni car entered");
    test_enter_car(1, "Uni car entered");
    test_enter_car(1, "Uni car entered");
    test_enter_car(0, "Non-Uni car entered");
    test_enter_car(0, "Non-Uni car entered");

    change_time(10, "Time changed to 10");
    test_exit_car(1, "Uni car exited");

    change_time(11, "Time changed to 11");
    test_exit_car(1, "Uni car exited");
    test_exit_car(0, "Non-Uni car exited");

    change_time(12, "Time changed to 12");
    test_enter_car(1, "Uni car entered");
    test_enter_car(1, "Uni car entered");
    test_enter_car(1, "Uni car entered");
```

```
    change_time(13, "Time changed to 13");
    test_enter_car(1, "Uni car entered");
    test_enter_car(1, "Uni car entered");
    test_enter_car(0, "Non-Uni car entered");

    change_time(14, "Time changed to 14");
    test_exit_car(1, "Uni car exited");
    test_exit_car(0, "Non-Uni car exited");
    test_exit_car(0, "Non-Uni car exited");

    change_time(17, "Time changed to 17");
    test_enter_car(0, "Non-Uni car entered");
    test_enter_car(0, "Non-Uni car entered");
    test_enter_car(0, "Non-Uni car entered");
    test_enter_car(0, "Non-Uni car entered");
    test_enter_car(0, "Non-Uni car entered");

    change_time(20, "Time changed to 20");
    test_exit_car(1, "Uni car exited");
    test_exit_car(0, "Non-Uni car exited");
    test_exit_car(0, "Non-Uni car exited");
    test_exit_car(0, "Non-Uni car exited");
    test_exit_car(0, "Non-Uni car exited");
```

برای کامپایل کردن و اجرای این کد، از ابزار iverilog استفاده شده‌است. با استفاده از این ابزار، می‌توان کد وریلاگ را در محیط ترمینال لپ‌تاپ‌های مختلف اجرا کرد و به سادگی می‌توان از صحت ماژول‌های طراحی شده مطمئن شد.

خروجی فایل بستر آزمون در صفحه بعد آمده‌است و بخش‌های مختلف آن تحلیل شده‌است.

- خروجی فایل تست‌بنچ -

Time	Hour	Uni Parked	Parked	Uni Vacated	Vacated	Uni Vacant	Vacant	Operation
0	0	0	0	5	5	1	1	Initial state
5	9	0	0	5	5	1	1	Time changed to 9
15	9	1	0	4	5	1	1	Uni car entered
25	9	2	0	3	5	1	1	Uni car entered
35	9	3	0	2	5	1	1	Uni car entered
45	9	3	1	2	4	1	1	Non-Uni car entered
55	9	3	2	2	3	1	1	Non-Uni car entered
65	10	3	2	2	3	1	1	Time changed to 10
75	10	2	2	3	3	1	1	Uni car exited
85	11	2	2	3	3	1	1	Time changed to 11
95	11	1	2	4	3	1	1	Uni car exited
105	11	1	1	4	4	1	1	Non-Uni car exited
115	12	1	1	4	4	1	1	Time changed to 12
125	12	2	1	3	4	1	1	Uni car entered
135	12	3	1	2	4	1	1	Uni car entered
145	12	4	1	1	4	1	1	Uni car entered
155	13	4	1	0	5	0	1	Time changed to 13
165	13	4	2	0	4	0	1	Uni car entered
175	13	4	3	0	3	0	1	Uni car entered
185	13	4	4	0	2	0	1	Non-Uni car entered
195	14	3	5	0	2	0	1	Time changed to 14
205	14	2	5	1	2	1	1	Uni car exited
215	14	2	4	1	3	1	1	Non-Uni car exited
225	14	2	3	1	4	1	1	Non-Uni car exited
235	17	2	3	0	5	0	1	Time changed to 17
245	17	2	4	0	4	0	1	Non-Uni car entered
255	17	2	5	0	3	0	1	Non-Uni car entered
265	17	2	6	0	2	0	1	Non-Uni car entered
275	17	2	7	0	1	0	1	Non-Uni car entered
285	17	2	8	0	0	0	0	Non-Uni car entered
295	20	2	8	0	0	0	0	Time changed to 20
305	20	1	8	1	0	1	0	Uni car exited
315	20	1	7	1	1	1	1	Non-Uni car exited
325	20	1	6	1	2	1	1	Non-Uni car exited
335	20	1	5	1	3	1	1	Non-Uni car exited
345	20	1	4	1	4	1	1	Non-Uni car exited

تغییر زمان به 9 : ظرفیت دانشگاهی به 5 تنظیم می‌شود.

ورود خودرو دانشگاهی : یک خودرو وارد می‌شود و ظرفیت خالی دانشگاهی به 4 کاهش می‌یابد.

ورود خودرو دانشگاهی : یک خودرو دیگر وارد می‌شود و ظرفیت خالی دانشگاهی به 3 کاهش می‌یابد.

ورود خودرو دانشگاهی : یک خودرو دیگر وارد می‌شود و ظرفیت خالی دانشگاهی به 2 کاهش می‌یابد.

ورود خودرو عمومی : یک خودرو وارد می‌شود و ظرفیت خالی عمومی به 4 کاهش می‌یابد.

ورود خودرو عمومی : یک خودرو دیگر وارد می‌شود و ظرفیت خالی عمومی به 3 کاهش می‌یابد.

تغییر زمان به 10 : تغییر ظرفیت ندارد.

خروج خودرو دانشگاهی : یک خودرو خارج می‌شود و ظرفیت خالی دانشگاهی به 3 افزایش می‌یابد.

تغییر زمان به 11 : تغییر ظرفیت ندارد.
خروج خودرو دانشگاهی : یک خودرو دیگر خارج می‌شود و ظرفیت خالی دانشگاهی به 4 افزایش می‌یابد.
خروج خودرو عمومی : یک خودرو خارج می‌شود و ظرفیت خالی عمومی به 4 افزایش می‌یابد.

تغییر زمان به 12 : تغییر ظرفیت ندارد.
ورود خودرو دانشگاهی : یک خودرو وارد می‌شود و ظرفیت خالی دانشگاهی به 3 کاهش می‌یابد.
ورود خودرو دانشگاهی : یک خودرو دیگر وارد می‌شود و ظرفیت خالی دانشگاهی به 2 کاهش می‌یابد.
ورود خودرو دانشگاهی : یک خودرو دیگر وارد می‌شود و ظرفیت خالی دانشگاهی به 1 کاهش می‌یابد.
ورود خودرو دانشگاهی : یک خودرو دیگر وارد می‌شود و ظرفیت خالی دانشگاهی به 0 کاهش می‌یابد.

تغییر زمان به 13 : ظرفیت دانشگاهی به 4 کاهش می‌یابد.
ورود خودرو دانشگاهی : یک خودرو وارد می‌شود و ظرفیت خالی دانشگاهی به 0 کاهش می‌یابد.

تغییر زمان به 14 : ظرفیت دانشگاهی به 3 کاهش می‌یابد.
خروج خودرو دانشگاهی : یک خودرو خارج می‌شود و ظرفیت خالی دانشگاهی به 1 افزایش می‌یابد.
ورود خودرو عمومی : یک خودرو وارد می‌شود و ظرفیت خالی عمومی به 0 کاهش می‌یابد.

تغییر زمان به 17 : ظرفیت دانشگاهی به 2 کاهش می‌یابد.
ورود خودرو عمومی : یک خودرو وارد می‌شود و ظرفیت خالی عمومی به 0 کاهش می‌یابد.
ورود خودرو عمومی : یک خودرو دیگر وارد می‌شود و ظرفیت خالی عمومی به 0 کاهش می‌یابد.

تغییر زمان به 20 : تغییر ظرفیت ندارد.
خروج خودرو دانشگاهی : یک خودرو خارج می‌شود و ظرفیت خالی دانشگاهی به 1 افزایش می‌یابد .

این خروجی نشان‌دهنده تغییرات دقیق وضعیت پارکینگ در زمان‌های مختلف است . ما می‌توانیم ببینیم که چگونه با تغییرات ساعت و عملیات ورود و خروج خودروها، وضعیت پارکینگ تغییر می‌کند . سیستم به درستی تغییرات ظرفیت پارکینگ دانشگاهی و عمومی را بر اساس ساعت و عملیات مدیریت می‌کند و به‌روزرسانی‌های مناسبی انجام می‌دهد . این نشان می‌دهد که ماژول طراحی شده عملکرد صحیحی دارد و می‌تواند به طور موثر پارکینگ را مدیریت کند .

- سنتز روی FPGA با استفاده از QUARTUS -

در این بخش، ماژول طراحی شده با استفاده از نرم‌افزار Quartus Prime بر روی دستگاه Cyclone II سنتز شده‌است.
پس از سنتز ماژول درQuartus ، از ابزار Timing Analyzer برای تحلیل زمانی مدار استفاده شده است. این ابزار امکانات گسترده‌ای برای بررسی و تحلیل پارامترهای زمانی مدارهای دیجیتال ارائه می‌دهد. با استفاده ازTiming Analyzer ، می‌توان اطلاعات دقیقی در مورد زمان‌بندی مدار، شامل زمان‌های setup وhold ، و همچنین فرکانس بیشینه (Fmax) که مدار می‌تواند با آن عمل کند، به دست آورد.

فرکانس بیشینه یا Fmax

فرکانس بیشینه یا Fmax نشان‌دهنده بالاترین فرکانسی است که مدار می‌تواند بدون خطا در آن عمل کند. این پارامتر بسیار مهم است زیرا عملکرد صحیح مدار در فرکانس‌های بالاتر را تضمین می‌کند. پس از سنتز ماژول در Quartus و استفاده ازTiming Analyzer ، مقادیر دقیق Fmax به دست آمد. این مقدار به طراحان کمک می‌کند تا اطمینان حاصل کنند که مدار در شرایط عملیاتی مورد نظر به درستی عمل می‌کند.

زمان‌های hold و setup

Setup Time: زمان مورد نیاز برای ثابت ماندن ورودی‌های یک فلیپ‌فلاپ یا رجیستر قبل از لبه ساعت. این زمان باید به اندازه کافی طولانی باشد تا داده‌ها به درستی در فلیپ‌فلاپ ثبت شوند.
Hold Time: زمان مورد نیاز برای ثابت ماندن ورودی‌های یک فلیپ‌فلاپ یا رجیستر بعد از لبه ساعت. این زمان تضمین می‌کند که داده‌ها به درستی در فلیپ‌فلاپ حفظ شوند.
با بررسی Timing Analyzer، مقادیر دقیق برای setup time و hold time به دست آمدند. این مقادیر به طراحان این امکان را می‌دهند تا از عملکرد صحیح و بدون خطای مدار در شرایط مختلف اطمینان حاصل کنند.

مقادیر زیر برای این پارامترها یافت شد:

Fmax Summary

	Fmax	Restricted Fmax	Clock Name
1	23.45 MHz	23.45 MHz	car_entered

Summary (Setup)

	Clock	Slack
1	car_entered	-20.818

Summary (Hold)

	Clock	Slack
1	car_entered	-6.425