به نام خدا

پروژه درس طراحی سیستم های دیجیتال

استاد: مهندس فصحتی

اعضای گروه:

ارشیا ایزدیاری 401105656

علیرضا میررکنی 401106617

دانشکده مهندسی کامپیوتر دانشگاه صنعتی شریف – تابستان 1403

گزارش پروژه

در این پروژه، هدف طراحی یک مدار برای مدیریت پارکینگ می باشد. این مدار باید موارد زیر را پشتیبانی کند:

1) اولویت فضای پارکینگ با اساتید و کارمندان دانشگاه است و این ظرفیت بر اساس آمار حداکثر 500 خودرو تعیین گردیده است.
2) با توجه به اینکه فضای کل پارکینگ 700 خودرو است، از ساعت 8 تا 13 فقط 200 ظرفیت خالی برای ورود آزاد مجاز است.
3) از ساعت 13 تا 16، به ازای هر ساعت ظرفیت ورود آزاد 50 خودرو افزایش می یابد و در ساعت 16 ظرفیت ورود آزاد به 500 خودرو می رسد (به عبارت دیگر ظرفیت آزاد در ساعت 13 برابر250 خودرو، در ساعت 14 برابر 300 خودرو، در ساعت 15 برابر 350 خودرو و در ساعت 16 برابر 500 خودرو خواهد بود).

در ادامه هر کدام از ورودی ها و خروجی های مدار را به تفصیل بررسی می نماییم.

نام ورودی	نوع ورودی	توضیحات
clk	wire تک بیتی	به منظور تعیین ساعت استفاده می شود.
car_entered	wire تک بیتی	ورود یک خودرو را نمایش می دهد.
is_uni_car_entered	wire تک بیتی	آیا خودرو وارد شده متعلق به دانشگاه است؟
car_exited	wire تک بیتی	خروج یک خودرو را نمایش می دهد.
is_uni_car_exited	wire تک بیتی	آیا خودرو خارج شده متعلق به دانشگاه است؟

نام خروجی	نوع خروجی	توضیحات
uni_parked_car	vector یازده بیتی علامت دار	تعداد خودرو های متعلق به دانشگاه که در پارکینگ پارک شده اند.
parked_car	vector یازده بیتی علامت دار	تعداد خودرو های متعلق به ظرفیت آزاد که در پارکینگ پارک شده اند.
uni_vacated_space	vector یازده بیتی علامت دار	تعداد فضا های خالی متعلق به دانشگاه
vacated_space	vector یازده بیتی علامت دار	تعداد فضا های خالی متعلق به ظرفیت آزاد
uni_is_vacated_space	wire تک بیتی	آیا فضای خالی برای دانشگاه موجود است؟

آیا فضای خالی برای ظرفیت آزاد موجود است؟	wire تک بیتی	is_vacated_space
آیا وضعیت پارکینگ، وضعیت معتبری است؟	wire تک بیتی	valid

دقت کنید که تمامی سیگنال هایی که فقط می توانند مقادیر 0 یا 1 را داشته باشند (هم در ورودی و هم در خروجی)، از نوع wire تک بیتی در نظر گرفته شده اند. خروجی های uni_parked_car، parked_car، uni_vacated_space و vacated_space را بردار (vector) هایی علامت دار به طول 11 در نظر گرفته ایم. علت این است که بیشینه مقدار هر کدام از این متغیر ها برابر 700 می باشد (که به 10 بیت برای نگهداری آن نیاز داریم) و از طرفی ممکن است مقدار برخی از آن ها منفی شود (به طور مثال اگر ساعت 7 باشد و 500 ماشین آزاد در پارکینگ پارک کرده باشند، در ساعت 8 که ظرفیت آزاد برابر 200 ماشین می شود مقدار متغیر vacated_space برابر 300- خواهد شد)، به همین خاطر تمامی این متغیر ها را علامت دار در نظر گرفتیم تا در انجام عملیات های حسابی مشکلی ایجاد نشود (دقت کنید که مقدار متغیر های uni_parked_car و parked_car هیچ گاه منفی نخواهد شد).

به منظور طراحی مدار، از تعدادی متغیر کمکی استفاده کردیم. در ادامه به بررسی این متغیر های کمکی می پردازیم:

توضیحات	نوع متغیر کمکی	نام متغیر کمکی
تعداد کل خودرو های مربوط به ظرفیت آزاد که وارد پارکینگ شده اند.	vector یازده بیتی علامت دار	number_of_entered_cars
تعداد کل خودرو های مربوط به دانشگاه که وارد پارکینگ شده اند.	vector یازده بیتی علامت دار	number_of_entered_uni_cars
تعداد کل خودرو های مربوط به ظرفیت آزاد که از پارکینگ خارج شده اند.	vector یازده بیتی علامت دار	number_of_exited_cars
تعداد کل خودرو های مربوط به دانشگاه که از پارکینگ خارج شده اند.	vector یازده بیتی علامت دار	number_of_exited_uni_cars
متغیر کمکی برای نگهداری تعداد خودرو های متعلق به دانشگاه که در پارکینگ پارک شده اند.	vector یازده بیتی علامت دار	uni_parked_car_tmp
متغیر کمکی برای نگهداری تعداد خودرو های ظرفیت آزاد که در پارکینگ پارک شده اند.	vector یازده بیتی علامت دار	parked_car_tmp
ظرفیت پارکینگ برای خودرو های آزاد	integer	capacity
تعداد دقیقه سپری شده از ساعت جاری	integer	minute
نشان می دهد در چه ساعتی از روز هستیم.	integer	hour

دقت کنید که علت تعریف برخی از این متغیر های کمکی، این است که اگر مقدار متغیری در چند بلاک always مورد تغییر قرار بگیرد، ابزار سنتز دیگر قادر به سنتز کردن کد وریلاگ نخواهد بود. به طور مثال، علت تغییر متغیر های

number_of_entered_cars و number_of_exited_cars به جای تغییر مستقیم parked_car در بلاک های always، همین موضوع می باشد.

در ادامه به بررسی توصیف مدار با استفاده از زبان وریلاگ می پردازیم.

کد زیر توصیف مدار را با استفاده از زبان وریلاگ نمایش می دهد. این ماژول در فایل PARKING.v در پوشه verilog قرار گرفته است.

```
module PARKING (
    input clk,
    input car_entered,
    input is_uni_car_entered,
    input car_exited,
    input is_uni_car_exited,
    output signed [10 : 0] uni_parked_car,
    output signed [10 : 0] parked_car,
    output signed [10 : 0] uni_vacated_space,
    output signed [10 : 0] vacated_space,
    output uni_is_vacated_space,
    output is_vacated_space,
    output valid
);

    reg signed [10 : 0] number_of_entered_cars = 0;
    reg signed [10 : 0] number_of_entered_uni_cars = 0;
    reg signed [10 : 0] number_of_exited_cars = 0;
    reg signed [10 : 0] number_of_exited_uni_cars = 0;
    reg signed [10 : 0] uni_parked_car_tmp;
    reg signed [10 : 0] parked_car_tmp;

    integer capacity = 0;
    integer minute = 0;
    integer hour = 0;

    assign uni_parked_car = uni_parked_car_tmp;
    assign parked_car = parked_car_tmp;
    assign uni_vacated_space = 700 - capacity - uni_parked_car_tmp;
    assign vacated_space = capacity - parked_car_tmp;
    assign uni_is_vacated_space = (700 > capacity + uni_parked_car_tmp);
    assign is_vacated_space = (capacity > parked_car_tmp);
    assign valid = (uni_vacated_space >=0 && vacated_space >= 0);

    always @(posedge clk) begin
        minute <= minute + 1;
```

```
        if (minute == 600) begin
            minute <= 0;
            hour <= hour + 1;
        end

        if (hour == 24) begin
            hour <= 0;
        end

        if (hour >= 8 && hour < 13) begin
            capacity <= 200;
        end
        else if (hour >= 13 && hour <= 15) begin
            capacity <= 200 + (hour - 12) * 50;
        end
        else begin
            capacity <= 500;
        end
    end

    always @(posedge car_entered) begin
        number_of_entered_uni_cars <= number_of_entered_uni_cars +
((is_uni_car_entered & uni_is_vacated_space) ? 1 : 0);
        number_of_entered_cars <= number_of_entered_cars + ((~is_uni_car_entered
& is_vacated_space) ? 1 : 0);
    end

    always @(posedge car_exited) begin
        number_of_exited_uni_cars <= number_of_exited_uni_cars +
((is_uni_car_exited & (uni_parked_car > 0)) ? 1 : 0);
        number_of_exited_cars <= number_of_exited_cars + ((~is_uni_car_exited &
(parked_car > 0)) ? 1 : 0);
    end

    always @(number_of_entered_uni_cars, number_of_entered_cars,
number_of_exited_uni_cars, number_of_exited_cars) begin
        uni_parked_car_tmp = number_of_entered_uni_cars -
number_of_exited_uni_cars;
        parked_car_tmp = number_of_entered_cars - number_of_exited_cars;
    end

endmodule
```

در ادامه و در صفحه بعد، به توضیح بخش های مختلف کد بالا می پردازیم.

بخش اول: تعیین ورودی ها، خروجی ها و تعریف متغیر های کمکی

```
module PARKING (
    input clk,
    input car_entered,
    input is_uni_car_entered,
    input car_exited,
    input is_uni_car_exited,
    output signed [10 : 0] uni_parked_car,
    output signed [10 : 0] parked_car,
    output signed [10 : 0] uni_vacated_space,
    output signed [10 : 0] vacated_space,
    output uni_is_vacated_space,
    output is_vacated_space,
    output valid
);

    reg signed [10 : 0] number_of_entered_cars = 0;
    reg signed [10 : 0] number_of_entered_uni_cars = 0;
    reg signed [10 : 0] number_of_exited_cars = 0;
    reg signed [10 : 0] number_of_exited_uni_cars = 0;
    reg signed [10 : 0] uni_parked_car_tmp;
    reg signed [10 : 0] parked_car_tmp;

    integer capacity = 0;
    integer minute = 0;
    integer hour = 0;
```

در ابتدای گزارش، توضیحات کاملی در ارتباط با متغیر ها ارائه گردید؛ به همین دلیل از توضیحات بیشتر برای این بخش صرف نظر می کنیم.

بخش دوم: تعیین مقدار خروجی ها

```
    assign uni_parked_car = uni_parked_car_tmp;
    assign parked_car = parked_car_tmp;
    assign uni_vacated_space = 700 - capacity - uni_parked_car_tmp;
    assign vacated_space = capacity - parked_car_tmp;
    assign uni_is_vacated_space = (700 > capacity + uni_parked_car_tmp);
    assign is_vacated_space = (capacity > parked_car_tmp);
    assign valid = (uni_vacated_space >=0 && vacated_space >= 0);
```

در این بخش مقدار متغیر های خروجی را با استفاده از متغیر های کمکی تعیین کرده ایم.

مقدار خروجی های uni_parked_car و parked_car را به ترتیب برابر مقدار متغیر های کمکی uni_parked_car_tmp و parked_car_tmp قرار می دهیم. مقدار خود این متغیر های کمکی را نیز در یک بلاک always که در ادامه آن را بررسی خواهیم نمود، با تغییر متغیر های کمکی number_of_entered_uni_cars، number_of_entered_cars، number_of_exited_uni_cars و number_of_exited_cars تعیین می کنیم.

مقدار متغیر های uni_vacated_space و vacated_space از روابط زیر محاسبه می شود:

$$\text{uni_vacated_space} = 700 - \text{free capacity} - \text{number of parked uni cars}$$

$$\text{vacated_space} = \text{free capacity} - \text{number of parked free cars}$$

مقدار متغیر های uni_is_vacated_space و is_vacated_space نیز از روابط زیر محاسبه می شوند:

$$\begin{aligned}&\text{uni_is_vacated_space} \Leftrightarrow \text{uni_vacated_space} > 0\\&\Leftrightarrow 700 - \text{free capacity} - \text{number of parked uni cars} > 0\\&\Leftrightarrow 700 > \text{free capacity} + \text{number of parked uni cars}\end{aligned}$$

$$\begin{aligned}&\text{is_vacated_space} \Leftrightarrow \text{vacated_space} > 0\\&\Leftrightarrow \text{free capacity} - \text{number of parked free cars} > 0\\&\Leftrightarrow \text{free capacity} > \text{number of parked free cars}\end{aligned}$$

مقدار متغیر کمکی valid نیز زمانی برابر 1 می شود که هر دوی مقادیر uni_vacated_space و vacated_space نامنفی باشند که به وضوح در عبارت assign مربوط به این متغیر، این نکته رعایت شده است.

بخش سوم: تغییر دقیقه، ساعت و تعیین ظرفیت آزاد پارکینگ با توجه به ساعت

```
always @(posedge clk) begin
        minute <= minute + 1;

        if (minute == 600) begin
            minute <= 0;
            hour <= hour + 1;
        end

        if (hour == 24) begin
            hour <= 0;
        end

        if (hour >= 8 && hour < 13) begin
            capacity <= 200;
        end
```

```
        else if (hour >= 13 && hour <= 15) begin
            capacity <= 200 + (hour - 12) * 50;
        end
        else begin
            capacity <= 500;
        end
    end
```

در این بخش، موارد زیر انجام می شوند:

- با هر لبه بالارونده کلاک مقدار متغیر minute یک واحد افزایش می یابد.
- در صورتی که مقدار متغیر minute برابر 600 شود، مقدار متغیر hour یک واحد افزایش می یابد (به عبارت دیگر هر ساعت را معادل 600 کلاک و هر دقیقه را معادل 10 کلاک در نظر گرفته ایم).
- در صورتی که مقدار متغیر hour برابر 24 شود، یعنی به انتهای روز رسیده ایم و مقدار این متغیر به 0 ریست می شود.
- با توجه به ساعتی که در آن هستیم، ظرفیت آزاد پارکینگ تعیین می شود (بین ساعات 8 تا 12، این ظرفیت برابر 200 خودرو می باشد. در ساعات 13، 14 و 15 این ظرفیت از رابطه $200 + (\text{hour} - 12) * 50$ محاسبه می گردد. در سایر ساعات این ظرفیت برابر 500 خودرو می باشد).

بخش چهارم: ورود ماشین به پارکینگ

```
always @(posedge car_entered) begin
        number_of_entered_uni_cars <= number_of_entered_uni_cars +
((is_uni_car_entered & uni_is_vacated_space) ? 1 : 0);
        number_of_entered_cars <= number_of_entered_cars + ((~is_uni_car_entered
& is_vacated_space) ? 1 : 0);
    end
```

در صورتی که ماشینی وارد پارکینگ شود (مقدار ورودی car_entered برابر 1 شود)، این بلاک فعال می شود.

اگر ماشین وارد شده از نوع دانشگاهی باشد (is_uni_car_entered برابر 1 باشد) و ظرفیت دانشگاهی در پارکینگ وجود داشته باشد (is_uni_vacated_space برابر 1 باشد)، مقدار متغیر number_of_entered_uni_cars یک واحد افزایش می یابد.

اگر ماشین وارد شده از نوع دانشگاهی نباشد (is_uni_car_entered برابر 0 باشد) و ظرفیت آزاد در پارکینگ وجود داشته باشد (is_vacated_space برابر 1 باشد)، مقدار متغیر number_of_entered_cars یک واحد افزایش می یابد.

بخش پنجم: خروج ماشین از پارکینگ

```
always @(posedge car_exited) begin
        number_of_exited_uni_cars <= number_of_exited_uni_cars +
((is_uni_car_exited & (uni_parked_car > 0)) ? 1 : 0);
        number_of_exited_cars <= number_of_exited_cars + ((~is_uni_car_exited &
(parked_car > 0)) ? 1 : 0);
    end
```

در صورتی که ماشینی از پارکینگ خارج شود (مقدار ورودی car_exited برابر 1 شود)، این بلاک فعال می شود.

اگر ماشین خارج شده از نوع دانشگاهی باشد (is_uni_car_exited برابر 1 باشد) و ماشینی از نوع دانشگاهی در پارکینگ وجود داشته باشد (uni_parked_car مثبت باشد)، مقدار متغیر number_of_exited_uni_cars یک واحد افزایش می یابد.

اگر ماشین خارج شده از نوع دانشگاهی نباشد (is_uni_car_exited برابر 0 باشد) و ماشینی از نوع ظرفیت آزاد در پارکینگ وجود داشته باشد (parked_car مثبت باشد)، مقدار متغیر number_of_exited_cars یک واحد افزایش می یابد.

بخش ششم: بروزرسانی تعداد ماشین های پارک شده (متغیر های کمکی)

```
always @(number_of_entered_uni_cars, number_of_entered_cars,
number_of_exited_uni_cars, number_of_exited_cars) begin
        uni_parked_car_tmp = number_of_entered_uni_cars -
number_of_exited_uni_cars;
        parked_car_tmp = number_of_entered_cars - number_of_exited_cars;
    end
```

هرگاه هر یک از سیگنال های number_of_entered_uni_cars، number_of_entered_cars، number_of_exited_uni_cars و number_of_exited_cars تغییر کنند، این بلاک اجرا می‌شود و تعداد ماشین های پارک شده در پارکینگ (هم ماشین های دانشگاهی و هم ماشین های آزاد) بروزرسانی خواهند شد.

دقت کنید که تعداد ماشین های دانشگاهی پارک شده در پارکینگ (uni_parked_car_tmp) از تفاضل تعداد کل ماشین های دانشگاهی وارد شده (number_of_entered_uni_cars) و تعداد کل ماشین های دانشگاهی خارج شده (number_of_exited_uni_cars) به دست می آید.

همچنین تعداد ماشین های ظرفیت آزاد پارک شده در پارکینگ (parked_car_tmp) از تفاضل تعداد کل ماشین های ظرفیت آزاد وارد شده (number_of_entered_cars) و تعداد کل ماشین های ظرفیت آزاد خارج شده (number_of_exited_cars) به دست می آید.

در ادامه ماژول تحریک نوشته شده به منظور بررسی صحت مدار را مشاهده می کنید. این ماژول در فایل TB.v در پوشه verilog قرار دارد.

در نهایت به سنتز کد نوشته شده روی یک FPGA پرداختیم. تمامی فایل های مربوط به این بخش، در پوشه synthesize قرار گرفته است.

برای سنتز از نرم افزار Quartus استفاده کردیم؛ به این شکل که یک پروژه (با نام SYNTHESIZE) تحت FPGA ای به نام Arria II GX ایجاد کردیم، سپس فایل PARKING.v را (که حاوی ماژول PARKING می باشد) به آن اضافه نمودیم (این فایل از نوع Verilog HDL File می باشد). در ادامه این فایل را compile کردیم و فرایند سنتز توسط خود این نرم افزار انجام گردید.

مدار طراحی شده توسط این نرم افزار، از منوی tools>netlist viewer قابل مشاهده می باشد. به علت بزرگ بودن این مدار ها و کم کیفیت بودن تصاویر، از قرار دادن تصویر این مدار ها در این گزارش صرف نظر می کنیم.

برای مشاهده بیشترین فرکانس از قسمت compilation report منوی TimeQuest Timing Analyzer را انتخاب می کنیم. در این منو، سه فایل به ازای مدل سازی های مختلف وجود دارد که در هر کدام بیشینه فرکانس برای سیگنال های clk، car_exited و car_entered نوشته شده است. در ادامه تصویر خروجی این فایل ها را آورده ایم.

Slow 900mV 100C Model Fmax Summary

	Fmax	Restricted Fmax	Clock Name	Note
1	102.25 MHz	102.25 MHz	clk	
2	184.2 MHz	184.2 MHz	car_entered	
3	344.47 MHz	260.01 MHz	car_exited	limit due to minimum period restriction (max I/O toggle rate)

Slow 900mV -40C Model Fmax Summary

	Fmax	Restricted Fmax	Clock Name	Note
1	108.44 MHz	108.44 MHz	clk	
2	192.64 MHz	192.64 MHz	car_entered	
3	356.25 MHz	260.01 MHz	car_exited	limit due to minimum period restriction (max I/O toggle rate)

با مشاهده تصاویر بالا، به این نتیجه می رسیم که مسیر بحرانی برای مدار طراحی شده، مسیری است که منجر به فرکانس 102.25 MHz (در مدل سازی مربوط به Slow 900mV 100C Model) شده است، چرا که این فرکانس بیشترین مقدار را در بین فرکانس های نمایش داده شده دارد. همچنین می توان دریافت که مسیری که ورودی car_entered در مدار طی می کند، نسبت به مسیری که ورودی car_exited طی می کند دارای تاخیر بیشتری است (چرا که فرکانس آن کمتر می باشد).

دقت کنید که اگر فرکانس داده شده به مدار از فرکانس های مشخص شده بیشتر شود، در عملکرد مدار اختلال ایجاد خواهد شد. بهینه ترین حالت این است که فرکانس هر کدام از ورودی ها برابر با بیشینه فرکانس ممکن تنظیم شود تا علاوه بر افزایش سرعت و کارایی مدار (به دلیل کاهش تاخیر های موجود در مدار بویژه تاخیر مسیر بحرانی)، در عملکرد آن اختلالی ایجاد نشود.

در ادامه به بررسی setup time و hold time و رابطه آن ها با فرکانس ورودی می پردازیم. این بررسی ها برای مدل سازی Slow 900mV 100C Model انجام شده اند.

درابتدا مقادیر setup time و hold time را مشاهده می کنید.

Slow 900mV 100C Model Setup Summary

	Clock	Slack	End Point TNS
1	clk	-8.780	-394.518
2	car_entered	-4.498	-81.788
3	car_exited	-1.919	-39.942

Slow 900mV 100C Model Hold Summary

	Clock	Slack	End Point TNS
1	car_entered	0.409	0.000
2	car_exited	0.415	0.000
3	clk	0.419	0.000

مقدار فرکانس ورودی ها را نیز مجددا در شکل زیر می آوریم.

Slow 900mV 100C Model Fmax Summary

	Fmax	Restricted Fmax	Clock Name	Note
1	102.25 MHz	102.25 MHz	clk	
2	184.2 MHz	184.2 MHz	car_entered	
3	344.47 MHz	260.01 MHz	car_exited	limit due to minimum period restriction (max I/O toggle rate)

حال دقت کنید که می توان نوشت (جمع ها بدون در نظر گرفتن علامت انجام شده اند):

$$\text{clk: Setup Time} + \text{Hold Time} = 8.780 + 0.409 = 9.189 \text{ ns}$$
$$\Longrightarrow \frac{1}{\text{Setup Time} + \text{Hold Time}} = \frac{1}{9.189} \times 10^9 = 108.82 \text{ MHz} > 102.25 \text{ Mhz}$$

$$\text{car_entered: Setup Time} + \text{Hold Time} = 4.498 + 0.415 = 4.913 \text{ ns}$$
$$\Longrightarrow \frac{1}{\text{Setup Time} + \text{Hold Time}} = \frac{1}{4.913} \times 10^9 = 203.54 \text{ MHz} > 184.20 \text{ Mhz}$$

$$\text{car_exited: : Setup Time} + \text{Hold Time} = 1.919 + 0.419 = 2.338 \text{ ns}$$
$$\Longrightarrow \frac{1}{\text{Setup Time} + \text{Hold Time}} = \frac{1}{2.338} \times 10^9 = 427.71 \text{ MHz} > 260.01 \text{ Mhz}$$

همانطور که مشاهده می شود، برای تمامی ورودی ها وارون مجموع setup time و hold time بیشتر از فرکانس به دست آمده می باشد.

همچنین برای ورودی clk (ورودی با کمترین فرکانس)، مجموع setup time و hold time بسیار نزدیک به فرکانس آن می باشد و این یعنی به آنچه که می خواستیم، رسیده ایم.