بسم الله الرحمن الرحیم

جبرانی پایانترم طراحی سیستم‌های دیجیتال

سوال ۸

نام و نام خانوادگی: امیررضا سعیدی

تاریخ: ۱۴۰۲/۰۴/۰۶

دانشگاه صنعتی شریف

الف) در این سوال باید یک برنامه برای مدیریت پارکینگ طراحی کنیم.
ورودی و خروجی‌های مدار در صورت سوال داده شده است.
یک localparam برای نگهداری ظرفیت پارکینگ تعریف می‌کنیم. همچنین دو متغیر، یکی برای ظرفیت کارمندان و دیگری برای ظرفیت آزاد قرار می‌دهیم. با توجه به اینکه مجموع این دو برابر ۷۰۰ است، یکی را تعیین می‌کنیم و دیگری از روی آن مقداردهی می‌شود. در اینجا ظرفیت آزاد را مشخص می‌کنیم. با تغییر ساعت این مقادیر تغییر می‌کنند.
در اینجا اینگونه فرض کردیم که اگر پارکینگ برای اساتید ظرفیت نداشت، نتوانند از پارکینگ آزاد اضافه کنند و فقط ماشین‌های آزاد می‌توانند از پارکینگ آزاد استفاده کنند. همچنین اگر ظرفیت یکی از بخش‌ها منفی شد (مثلا از ۵۰۰ به ۴۵۰ در ساعت ۱۳ برای اساتید در حالی که ظرفیت پر است) ماشین‌هایی که قبلا داخل پارکینگ بوده‌اند باقی بمانند اما ماشین جدیدی نتواند وارد این بخش شود.
خروجی‌های مدار نیز طبق خواسته‌های سوال مقداردهی شده‌اند.

```
module ParkingSystem (
    input car_entered,
    input is_uni_car_entered,
    input car_exited,
    input is_uni_car_exited,
    input [4:0] hour,
    output reg signed [9:0] uni_parked_car = 0,
    output reg signed [9:0] parked_car = 0,
    output signed [9:0] uni_vacated_space,
    output signed [9:0] vacated_space,
    output uni_is_vacated_space,
    output is_vacated_space,
    output parking_is_vacated_space
);

localparam PARKING_CAPACITY = 700;
wire signed [9:0] uni_space;
reg signed [9:0] free_space = 0;

assign is_vacated_space = vacated_space > 0 && parking_is_vacated_space;
assign parking_is_vacated_space = uni_vacated_space + vacated_space > 0;
assign uni_vacated_space = uni_space - uni_parked_car;
assign uni_space = PARKING_CAPACITY - free_space;
assign uni_is_vacated_space = uni_vacated_space > 0 && parking_is_vacated_space;
assign vacated_space = free_space - parked_car;

always @(hour) begin
    if (hour >= 8 && hour < 13)
        free_space = 200;
    else if (hour >= 13 && hour < 16)
        free_space = 200 + (hour - 12) * 50;
    else
        free_space = 500;
end
```

حال باید یک بلوک always بسازیم که به ورود و خروج ماشین حساسیت دارد.
در صورتی که پارکینگ برای کارمندان جا داشته باشد و درخواست برای ورود ماشین کارمند باشد، ماشین کارمند می‌تواند وارد شود. سایر شرط‌ها نیز به همین صورت بررسی می‌شوند.

```
always @(negedge car_entered, negedge car_exited) begin
    if (!car_entered) begin
        if (is_uni_car_entered) begin
            if (uni_is_vacated_space)
                uni_parked_car <= uni_parked_car + 1;
        end
        else begin
            if (is_vacated_space)
                parked_car <= parked_car + 1;
        end
    end

    else if (!car_exited) begin
        if (is_uni_car_exited) begin
            if (uni_parked_car > 0)
                uni_parked_car <= uni_parked_car - 1;
        end
        else begin
            if (parked_car > 0)
                parked_car <= parked_car - 1;
        end
    end
end
```

تست‌بنچ:
مقادیر اولیه برای هردو تست‌بنچ:

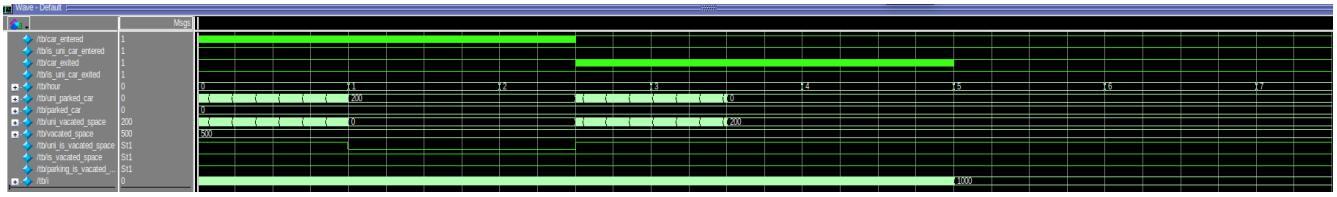
```
integer i = 0;
initial
    hour = 0;
always begin
    #400
    if (hour >= 23)
        hour = 0;
    else
        hour = hour + 1;
end
```

```
car_entered = 1;
is_uni_car_entered = 1;
car_exited = 1;
is_uni_car_exited = 1;
```

***هر ۴۰۰ واحد تاخیر ۱ ساعت به جلو می‌رویم.**

```
// tb1:
for (i = 0; i < 1000; i = i + 1) begin
    #1 car_entered <= !car_entered;
end

for (i = 0; i < 1000; i = i + 1) begin
    #1 car_exited <= !car_exited;
end
#1000 $stop;
```

۱) در این تست‌بنچ ۵۰۰ ماشین دانشگاه در ساعت ۰ درخواست ورود می‌دهند. از آنجا که ظرفیت در این ساعت فقط ۲۰۰ عدد است، به همین تعداد ماشین وارد پارکینگ شده و بقیه نمی‌توانند وارد شوند.
سپس ۵۰۰ ماشین درخواست خروج می‌دهند که به دلیل وجود تنها ۲۰۰ ماشین در پارکینگ، فقط این ۲۰۰ ماشین خارج می‌شوند.



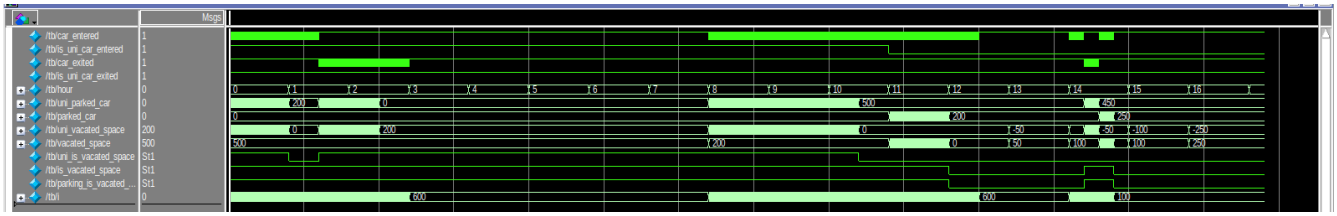
```
// tb2
for (i = 0; i < 600; i = i + 1) begin
    #1 car_entered <= !car_entered;
end


for (i = 0; i < 600; i = i + 1) begin
    #1 car_exited <= !car_exited;
end

#2000
for (i = 0; i < 1200; i = i + 1) begin
    #1 car_entered <= !car_entered;
end

is_uni_car_entered = 0;
for (i = 0; i < 600; i = i + 1) begin
    #1 car_entered <= !car_entered;
end

#600
for (i = 0; i < 100; i = i + 1) begin
    #1 car_entered <= !car_entered;
end

for (i = 0; i < 100; i = i + 1) begin
    #1 car_exited <= !car_exited;
end

for (i = 0; i < 100; i = i + 1) begin
    #1 car_entered <= !car_entered;
end

#1000 $stop();
```

۲) در این تست‌بنچ ابتدا ۳۰۰ ماشین دانشگاه درخواست ورود می‌دهند که فقط ۲۰۰ ماشین می‌توانند وارد شوند. سپس ۳۰۰ ماشین دانشگاه درخواست خروج می‌دهند که چون فقط ۲۰۰ ماشین در پارکینگ وجود دارد، فقط ۲۰۰ ماشین می‌توانند خارج شوند.

پس از آن ۵ ساعت صبر می‌کنیم تا ساعت ۸ شود.
در این ساعت ۶۰۰ ماشین دانشگاه درخواست ورود می‌دهند که فقط ۵۰۰ ماشین می‌توانند وارد شوند.
سپس ۳۰۰ ماشین آزاد درخواست ورود می‌دهند که فقط ۲۰۰ ماشین می‌توانند وارد شوند و ظرفیت پارکینگ پر می‌شود.

در ساعت ۱۳ ظرفیت آزاد ۵۰ تا بیشتر و دانشگاه ۵۰ تا کمتر می‌شود.
در این ساعت ۵۰ ماشین دانشگاه درخواست ورود می‌دهند. با وجود اینکه ظرفیت ماشین آزاد وجود دارد، اما ظرفیت پارکینگ پر است و امکان ورود ماشین وجود ندارد.
سپس ۵۰ ماشین دانشگاه درخواست خروج می‌دهند.
سپس ۵۰ ماشین آزاد درخواست ورود می‌دهند. حال از آنجا که ظرفیت کافی برای خودروهای آزاد وجود دارد، این ۵۰ ماشین وارد پارکینگ می‌شوند.

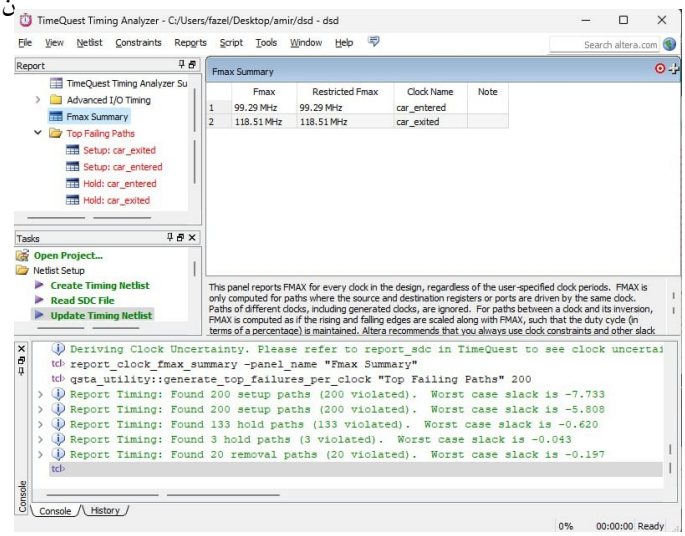
ن

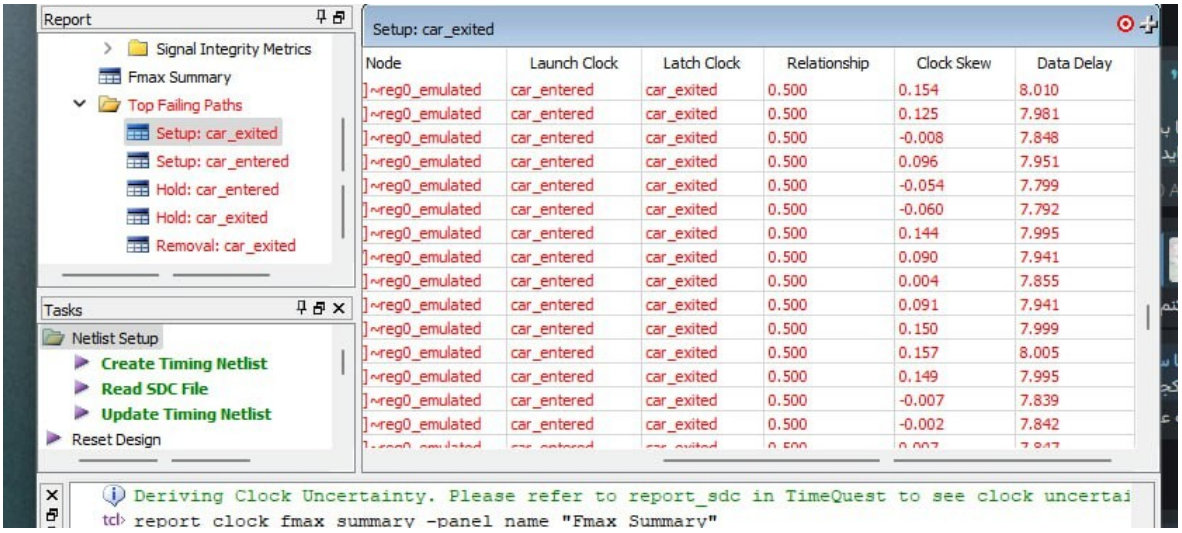


ب

ب) برای به دست آوردن فرکانس بیشینه مدار، از نرم‌افزار کوارتوس استفاده می‌کنیم. خروجی‌های این نرم‌افزار به شرح زیر است:

ن





تصویر اول فرکانس بیشینه مدار بر اساس دو سیگنال car_entered و car_exited نمایش می‌دهد.
تصویر بعدی نیز بیشینه تاخیر داده‌هاست. اگر این بیشینه تاخیر را معکوس کنیم، با تقریب به همان عدد فرکانس بیشینه می‌رسیم.

$$\frac{1}{8\,ns} = 125\,MHz \approx 118.5\,MHz$$