به نام خدا

Digital System Design

طراحی سیستم‌های دیجیتال

نام استاد : دکتر فصحتی

نام دانشجو : امیررضا اینانلو

شماره دانشجویی : 401105667

سؤال ۸:

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

۱) اولویت فضای پارکینگ با اساتید و کارمندان دانشگاه است و این ظرفیت بر اساس آمار حداکثر ۵۰۰ خودرو تعیین گردیده است.

۲) باتوجه‌به اینکه فضای کل پارکینگ ۷۰۰ خودرو است از ساعت ۸ تا ۱۳ فقط ۲۰۰ ظرفیت خالی برای ورود آزاد موجود است.

۳) 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.

الف) اگر در هنگام ورود/خروج خودرو یک سیگنال ورودی به مدار نوع آن را مشخص کند (دانشگاه/آزاد): با زبان وریلاگ مداری را توصیف کنید که دارای ورودی‌ها/خروجی‌های زیر باشد:

خروجی‌ها	
uni_parked_car	تعداد خودروهایی متعلق به دانشگاه که در پارکینگ پارک شده‌اند.

[1] Vector Processor

parked_care	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؟
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؟

درصورتی‌که نیاز به ورودی‌ها/خروجی‌های دیگری هم است آن را با ذکر دلیل به طراحی خود بیفزایید و جهت اطمینان از صحت عملکرد مدار، مدار خود را مورد آزمون قرار دهید (۵۰ نمره).

ب) مدار خود را برای یک **FPGA** به انتخاب خود سنتز کنید. از گزارش‌های سنتز، بیشترین فرکانس ممکن برای این مدار را با ذکر دلیل مشخص کنید (۱۰ نمره).

is_uni_car_exited;

همانطور که می بینید دو ورودی جدید نسبت به ورودی‌های سوال به مدارمان اضافه کرده ایم. ورودی اول یعنی start برای این است که مشخص کنیم برنامه از کی شروع به کار می کند. یعنی در واقعیت پارکینگ از چه زمانی باز می شود.

ورودی دوم یعنی clk برای این است که ساعت را افزایش دهد. یعنی یک متغیر از نوع integer به نام hour در برنامه تعریف کرده ایم. به ازای هر لبه ی بالا رونده ی clk مقدار hour یک واحد اضافه می شود. و مقدار ابتدایی hour = 8 است. زیرا پارکینگ از ساعت 8 شروع به کار می کند.

توضیح منطق برنامه :

همانطور که در کد برنامه که در ادامه هم قرار داده شده است منطق برنامه را توصیف می کنیم.

زمانی که posedge start اتفاق بیافتد، یعنی کلید start فشرده شود، مقادیر برنامه initial (مقدار دهی اولیه) می شوند.

که این مقادیر با توجه به صورت سوال به شرح زیر هستند:

```
uni_parked_car = 0;
            parked_care = 0;
            uni_vacated_space = 500;
            vacated_space = 200;
            uni_is_vacated_space = 1;
            is_vacated_space = 1;
```

در ادامه اگر لبه ی بالارونده ی کلاک ببینیم، یعنی یک ساعت گذشته است. به طور مثال اگر در ساعت 10 باشیم و لبه ی بالارونده ی کلاک را ببینیم، ساعت 11 می شود.

اکنون با توجه به ساعتی که در آن قرار داریم تصمیم می گیریم که چه مقادیری به متغیرهایمان بدهیم. اگر ساعت کمتر از 13(یعنی کمتر مساوی 12) بود، ظرفیت ها تغییری نمی کنند. اگر ساعت در بازه ی [13, 15] بود، طبق صورت سوال در هر ساعت 50 تا به ظرفیت پارکینگ خودروهای آزاد اضافه می شود. به طور مثال زمانی که ساعت از 12 به 13 تغییر می کند، ظرفیت آزاد از 200 نفر به 250 نفر تبدیل می شود. در نهایت در ساعت 15 ظرفیت آزاد برابر 350 خودرو می شود. در ادامه و در ساعت 16، ظرفیت خودروهای آزاد به 500 خودرو افزایش پیدا می کند. و پس از آن ساعت دیگر تغییری در ظرفیت خودروها ایجاد نمی شود.

کد حالتی که لبه ی بالارونده ی کلاک ببینیم به صورت زیر است:

```
if (clk) begin
            hour = hour + 1;
            if (hour < 13) begin
                // do nothing
            end else if (hour < 16) begin
                // increment capacity for non uni cars
                // if more than 450 uni cars were in parking
                // means we must exit some of them
                // to can place other non uni cars
                if (uni_parked_car > (500 - (hour - 12) * 50)) begin
                    uni_parked_car = 500 - (hour - 12) * 50;
                    uni_vacated_space = 0;
                    vacated_space = vacated_space + 50;
                    uni_is_vacated_space = 0;
                    is_vacated_space = 1;
                end else begin
                    uni_vacated_space = uni_vacated_space - 50;
                    vacated_space = vacated_space + 50;
                    is_vacated_space = 1;
                end
            end else if (hour == 16) begin
                // non uni car capacity reaches to 500
                if (uni_parked_car > 200) begin
```

```
                uni_parked_car = 200;
                uni_vacated_space = 0;
                vacated_space = vacated_space + 150;
                uni_is_vacated_space = 0;
                is_vacated_space = 1;
            end else begin
                uni_vacated_space = uni_vacated_space - 150;
                vacated_space = vacated_space + 150;
                is_vacated_space = 1;
            end
        end else begin
            // do nothing
        end
    end
```

با توجه به منطقی که پیاده سازی کرده ایم لبه ی بالا رونده بر لبه ی بالارونده ی car_entered و car_exited اولویت دارد. و طبیعتا این تنها راه پیاده سازی این برنامه نیست. می توان دیگری را هم در پیش گرفت.

در نهایت اگر posedge car_entered یا posedge car_exited ببینیم، به صورتی که در ادامه شرح می دهیم عمل می کنیم.

اگر posedge car_entered یا posedge car_exited ببینیم به این معنی است که ماشینی قصد دارد وارد پارکینگ شود یا از آن خارج شود. (در منطقی که پیاده سازی کرده ایم امکان ورود و خروج همزمان دو ماشین وجود دارد. یعنی همزمان یک ماشین می تواند وارد شود و ماشین دیگری خارج شود. مثل پارکینگ های واقعی.)

اگر car_entered برابر 1 بود، به نوع آن توجه می کنیم، یعنی اینکه آن خودرو متعلق به کارکنان دانشگاه است یا خودروی آزاد. و با توجه

به نوع خودرو و ظرفیت پارکینگ برای هر بخش تصمیم می گیریم که آن خودرو وارد پارکینگ بشود یا خیر.

اما برای car_exited چون پر بودن پارکینگ اهمیتی ندارد خودرو در هر صورت از آن خارج می شود و با توجه به نوع خودرو(آزاد یا متعلق به دانشگاه)، ظرفیت آن بخش یکی اضافه می شود.

کد این بخش به صورت زیر است:

```
else begin
        if (car_entered) begin
            if(is_uni_car_entered) begin
                if (uni_is_vacated_space) begin
                    uni_vacated_space =  uni_vacated_space - 1;
                    uni_parked_car = uni_parked_car + 1;
                    if (uni_vacated_space-1 < 0) begin
                        uni_is_vacated_space = 0;
                    end
                end else begin
                    uni_vacated_space = uni_vacated_space;
                    uni_parked_car = uni_parked_car;
                end
            end else begin
                if (is_vacated_space) begin
                    vacated_space = vacated_space - 1;
                    parked_care = parked_care + 1;
                    if (vacated_space-1 < 0) begin
                        is_vacated_space = 0;
                    end
                end else begin
                    vacated_space = vacated_space;
                    parked_care = parked_care;
                end
            end
        end
        if (car_exited) begin
            if (is_uni_car_exited) begin
                uni_vacated_space = uni_vacated_space + 1;
                uni_parked_car = uni_parked_car - 1;
                uni_is_vacated_space = 1;
            end else begin
                vacated_space = vacated_space + 1;
                parked_care = parked_care - 1;
                is_vacated_space = 1;
```

```
            end
        end else begin
            // do nothing
        end
        end
```

در نهایت کد کلی ماژول parking به صورت زیر است:

(توجه کنید که با توجه به مقادیری که قرار است متغیرها نگهداری کنند نوع آنها و تعداد بیت آن ها مشخص شده است. مثلا چون حداکثر ظرفیت در تمام ساعات برای خودروهای دانشگاه یا خودروهای آزاد 500 تا است، یک رجیستر 9 بیتی برای آن ها در نظر گرفتیم که بتوانند تمام مقادیر را نگهداری کنند.)

```
module parking(start, clk, car_entered, is_uni_car_entered, car_exited, is_uni_car_exited,
            uni_parked_car, parked_care, uni_vacated_space, vacated_space,
uni_is_vacated_space, is_vacated_space);

    // we can make our program synchronous
    // by adding clock signal to our architecture
    integer hour = 8;
    input start, clk, car_entered, is_uni_car_entered, car_exited, is_uni_car_exited;
    output reg [8:0] uni_parked_car;
    output reg [8:0] parked_care;
    output reg [8:0] uni_vacated_space;
    output reg [8:0] vacated_space;
    output reg uni_is_vacated_space;
    output reg is_vacated_space;

    always @(posedge start or posedge clk or posedge car_entered or posedge car_exited) begin
        // hour 8 means 8 till 9
        // means this range : 8 = [8, 9]
        // after pressign start button
        // logic starts to execute
        // parking starts at 8 AM
        if (start) begin
            uni_parked_car = 0;
            parked_care = 0;
            uni_vacated_space = 500;
            vacated_space = 200;
            uni_is_vacated_space = 1;
```

```
            is_vacated_space = 1;
        end
        if (clk) begin
            hour = hour + 1;
            if (hour < 13) begin
                // do nothing
            end else if (hour < 16) begin
                // increment capacity for non uni cars
                // if more than 450 uni cars were in parking
                // means we must exit some of them
                // to can place other non uni cars
                if (uni_parked_car > (500 - (hour - 12) * 50)) begin
                    uni_parked_car = 500 - (hour - 12) * 50;
                    uni_vacated_space = 0;
                    vacated_space = vacated_space + 50;
                    uni_is_vacated_space = 0;
                    is_vacated_space = 1;
                end else begin
                    uni_vacated_space = uni_vacated_space - 50;
                    vacated_space = vacated_space + 50;
                    is_vacated_space = 1;
                end
            end else if (hour == 16) begin
                // non uni car capacity reaches to 500
                if (uni_parked_car > 200) begin
                    uni_parked_car = 200;
                    uni_vacated_space = 0;
                    vacated_space = vacated_space + 150;
                    uni_is_vacated_space = 0;
                    is_vacated_space = 1;
                end else begin
                    uni_vacated_space = uni_vacated_space - 150;
                    vacated_space = vacated_space + 150;
                    is_vacated_space = 1;
                end
            end else begin
                // do nothing
            end
        end else begin
        if (car_entered) begin
            if(is_uni_car_entered) begin
                if (uni_is_vacated_space) begin
                    uni_vacated_space =  uni_vacated_space - 1;
                    uni_parked_car = uni_parked_car + 1;
                    if (uni_vacated_space-1 < 0) begin
                        uni_is_vacated_space = 0;
                    end
                end else begin
                    uni_vacated_space = uni_vacated_space;
                    uni_parked_car = uni_parked_car;
                end
            end else begin
```

```
                if (is_vacated_space) begin
                    vacated_space = vacated_space - 1;
                    parked_care = parked_care + 1;
                    if (vacated_space-1 < 0) begin
                        is_vacated_space = 0;
                    end
                end else begin
                    vacated_space = vacated_space;
                    parked_care = parked_care;
                end
            end
        end
        if (car_exited) begin
            if (is_uni_car_exited) begin
                uni_vacated_space = uni_vacated_space + 1;
                uni_parked_car = uni_parked_car - 1;
                uni_is_vacated_space = 1;
            end else begin
                vacated_space = vacated_space + 1;
                parked_care = parked_care - 1;
                is_vacated_space = 1;
            end
        end else begin
            // do nothing
        end
        end
    end
endmodule
```

در نهایت برای اطمینان از صحت عملکرد مدار طراحی شده، تست بنچ زیر را برای آن نوشتیم.

در این تست بنچ ابتدا ورودی ها و خروجی ها را تعریف می کنیم. و سپس یک نمونه (instance) از ماژول parking با ورودی ها و خروجی های متناسب می گیریم.

سپس سیگنال کلاک را به صورت زیر مقدار دهی می کنیم.

```
initial begin
        clk = 0;
        forever #50 begin
            clk = ~clk;
            #1;
            clk = ~clk;
        end
    end
```

علت اینکه در هر 50 واحد زمانی ابتدا کلاک را نات می کنیم و سپس بعد از یک واحد زمانی به حالت عادی بر می گردانیم این است که در طراحی ما کلاک اولویت بالاتری نسبت به car_entered و car_exited دارد. بنابراین اگر posedge car_entered یا posedge car_exited ببینیم، و کلاک مقدار 1 داشته باشد، به جای ورود یا خروج خودرو، ساعت یک واحد افزایش پیدا می کند.

سپس در بلاک initial بقیه ی ورودی ها را مقداردهی می کنیم. در ابتدا start را برابر 1 می کنیم و دیگر ورودی ها را برابر 0 می کنیم.(اگر این کار را نکنیم ورودی ها مقدار x خواهند داشت.)

سپس با یک وقفه ی زمانی start را برابر 0 می کنیم و برنامه ی ما از این نقطه آغاز می شود.

برای تست کردن تمامی حالت های ممکن از 4 حلقه استفاده کرده ایم. در حلقه ی اول خودروهای متعلق به اساتید یا کارمندان دانشگاه وارد پارکینگ می شوند. سپس در حلقه ی بعدی خودروهای آزاد وارد پارکینگ می شوند. و در دو حلقه ی بعدی به ترتیب خودروهای متعلق به استادان و خودروهای آزاد از پارکینگ خارج می شوند.

کد این قطعه از برنامه به صورت زیر است:

```
integer i, j, k, l;
    initial begin
        // initial
        car_entered = 0;
        is_uni_car_entered = 0;
        car_exited = 0;
        is_uni_car_exited = 0;
        // start the program
        start = 1;
        #30;
        start = 0;
        #10;

        for (i = 0; i < 5; i = i + 1) begin
            car_entered = 1;
            is_uni_car_entered = 1;
            #3;
            car_entered = 0;
            #3;
        end
        #15;

        for (j = 0; j < 5; j = j + 1) begin
            car_entered = 1;
            is_uni_car_entered = 0;
            #3;
            car_entered = 0;
            #3;
        end
        #15;

        for (k = 0; k < 3; k = k + 1) begin
            car_exited = 1;
            is_uni_car_exited = 1;
            #3;
```

```
            car_exited = 0;
            #3;
        end
        #15;

        for (l = 0; l < 3; l = l + 1) begin
            car_exited = 1;
            is_uni_car_exited = 0;
            #3;
            car_exited = 0;
            #3;
        end
        #15;
        $stop();
    end
```

در نهایت کد کامل تست بنچ به صورت زیر است.

کد تست بنچ:

```
module tb;
    reg start, clk, car_entered, is_uni_car_entered, car_exited, is_uni_car_exited;
    wire [8:0] uni_parked_car;
    wire [8:0] parked_care;
    wire [8:0] uni_vacated_space;
    wire [8:0] vacated_space;
    wire uni_is_vacated_space;
    wire is_vacated_space;

    parking parking_inst (
        .start(start),
        .clk(clk),
        .car_entered(car_entered),
        .is_uni_car_entered(is_uni_car_entered),
        .car_exited(car_exited),
        .is_uni_car_exited(is_uni_car_exited),
        .uni_parked_car(uni_parked_car),
        .parked_care(parked_care),
        .uni_vacated_space(uni_vacated_space),
        .vacated_space(vacated_space),
        .uni_is_vacated_space(uni_is_vacated_space),
        .is_vacated_space(is_vacated_space)
    );

    initial begin
        clk = 0;
        forever #50 begin
```

```
            clk = ~clk;
            #1;
            clk = ~clk;
        end
    end
    integer i, j, k, l;
    initial begin
        // initial
        car_entered = 0;
        is_uni_car_entered = 0;
        car_exited = 0;
        is_uni_car_exited = 0;
        // start the program
        start = 1;
        #30;
        start = 0;
        #10;

        for (i = 0; i < 5; i = i + 1) begin
            car_entered = 1;
            is_uni_car_entered = 1;
            #3;
            car_entered = 0;
            #3;
        end
        #15;

        for (j = 0; j < 5; j = j + 1) begin
            car_entered = 1;
            is_uni_car_entered = 0;
            #3;
            car_entered = 0;
            #3;
        end
        #15;

        for (k = 0; k < 3; k = k + 1) begin
            car_exited = 1;
            is_uni_car_exited = 1;
            #3;
            car_exited = 0;
            #3;
        end
        #15;

        for (l = 0; l < 3; l = l + 1) begin
            car_exited = 1;
            is_uni_car_exited = 0;
            #3;
            car_exited = 0;
            #3;
        end
```

```
        #15;
        $stop();
    end

    initial begin
        $monitor($time, ": (clk = %b, start = %b, car_entered = %b, is_uni_car_entered = %b,
car_exited = %b, is_uni_car_exited = %b)", clk, start, car_entered, is_uni_car_entered,
car_exited, is_uni_car_exited,
            "(uni_parked_car = %d, parked_car = %d, uni_vacated_space = %d, vacated_space =
%d, \n\t\t uni_is_vacated_space = %d, is_vacated_space = %d)\n", uni_parked_car, parked_care,
uni_vacated_space, vacated_space, uni_is_vacated_space, is_vacated_space);
    end
endmodule
```

خروجی این تست بنچ در ادامه قرار داده شده است.(به دلیل حجم زیاد خروجی، تصویر کوچک است، آن را زوم کنید. یا می توانید کد برنامه را که در همین پوشه قرار می گیرد اجرا کنید و خروجی آن را ببینید.)

همانطور که در خروجی صفحه ی بعد می بینید ابتدا مقدار uni_parked_car، یعنی تعداد خودروهای پارک شده متعلق به کارمندان دانشگاه از 0 به 5 افزایش پیدا می کند.(چون حلقه ی اول پنج بار اجرا می شود. با تغییر آن این مقدار هم به تبع تغییر می کند.)

سپس مقدار parked_care که همان خودروهای پارک شده ی آزاد هستند از 0 به 5 افزایش پیدا می کند.(باز هم با توجه به اینکه حلقه ی مرتبط با این بخش 5 بار اجرا می شود.

سپس مقدار uni_parked_car از 5 به 2 کاهش پیدا می کند. (چون این حلقه 3 بار اجرا می شود و 3 خودروی متعلق به کارمندان دانشگاه از پارکینگ خارج می شوند.)

در نهایت مقدار parked_care از 5 به 2 کاهش پیدا می کند. (زیرا این حلقه هم 3 بار اجرا می شود. و در هر بار اجرا شدن یک خودروی آزاد از پارکینگ خارج می شود.)

خروجی تست بنچ در صفحه ی بعد قرار دارد.

```
# Compile of parking.v was successful.
# Compile of tb.v was successful.
# 2 compiles, 0 failed with no errors.
VSIM 22> vsim -gui work.tb
# vsim -gui work.tb
# Loading work.tb
# Loading work.parking
VSIM 23> run -all
#                    0: (clk = 0, start = 1, car_entered = 0, is_uni_car_entered = 0, car_exited = 0, is_uni_car_exited = 0)(uni_parked_car =   0, parked_car =   0, uni_vacated_space = 500, vacated_space = 200,
#                uni_is_vacated_space = 1, is_vacated_space = 1)
#
#                   30: (clk = 0, start = 0, car_entered = 0, is_uni_car_entered = 0, car_exited = 0, is_uni_car_exited = 0)(uni_parked_car =   0, parked_car =   0, uni_vacated_space = 500, vacated_space = 200,
#                uni_is_vacated_space = 1, is_vacated_space = 1)
#
#                   40: (clk = 0, start = 0, car_entered = 1, is_uni_car_entered = 1, car_exited = 0, is_uni_car_exited = 0)(uni_parked_car =   1, parked_car =   0, uni_vacated_space = 499, vacated_space = 200,
#                uni_is_vacated_space = 1, is_vacated_space = 1)
#
#                   43: (clk = 0, start = 0, car_entered = 0, is_uni_car_entered = 1, car_exited = 0, is_uni_car_exited = 0)(uni_parked_car =   1, parked_car =   0, uni_vacated_space = 499, vacated_space = 200,
#                uni_is_vacated_space = 1, is_vacated_space = 1)
#
#                   46: (clk = 0, start = 0, car_entered = 1, is_uni_car_entered = 1, car_exited = 0, is_uni_car_exited = 0)(uni_parked_car =   2, parked_car =   0, uni_vacated_space = 498, vacated_space = 200,
#                uni_is_vacated_space = 1, is_vacated_space = 1)
#
#                   49: (clk = 0, start = 0, car_entered = 0, is_uni_car_entered = 1, car_exited = 0, is_uni_car_exited = 0)(uni_parked_car =   2, parked_car =   0, uni_vacated_space = 498, vacated_space = 200,
#                uni_is_vacated_space = 1, is_vacated_space = 1)
#
#                   50: (clk = 1, start = 0, car_entered = 0, is_uni_car_entered = 1, car_exited = 0, is_uni_car_exited = 0)(uni_parked_car =   2, parked_car =   0, uni_vacated_space = 498, vacated_space = 200,
#                uni_is_vacated_space = 1, is_vacated_space = 1)
#
#                   51: (clk = 0, start = 0, car_entered = 0, is_uni_car_entered = 1, car_exited = 0, is_uni_car_exited = 0)(uni_parked_car =   2, parked_car =   0, uni_vacated_space = 498, vacated_space = 200,
#                uni_is_vacated_space = 1, is_vacated_space = 1)
#
#                   52: (clk = 0, start = 0, car_entered = 1, is_uni_car_entered = 1, car_exited = 0, is_uni_car_exited = 0)(uni_parked_car =   3, parked_car =   0, uni_vacated_space = 497, vacated_space = 200,
#                uni_is_vacated_space = 1, is_vacated_space = 1)
#
#                   55: (clk = 0, start = 0, car_entered = 0, is_uni_car_entered = 1, car_exited = 0, is_uni_car_exited = 0)(uni_parked_car =   3, parked_car =   0, uni_vacated_space = 497, vacated_space = 200,
#                uni_is_vacated_space = 1, is_vacated_space = 1)
#
#                   58: (clk = 0, start = 0, car_entered = 1, is_uni_car_entered = 1, car_exited = 0, is_uni_car_exited = 0)(uni_parked_car =   4, parked_car =   0, uni_vacated_space = 496, vacated_space = 200,
#                uni_is_vacated_space = 1, is_vacated_space = 1)
#
#                   61: (clk = 0, start = 0, car_entered = 0, is_uni_car_entered = 1, car_exited = 0, is_uni_car_exited = 0)(uni_parked_car =   4, parked_car =   0, uni_vacated_space = 496, vacated_space = 200,
#                uni_is_vacated_space = 1, is_vacated_space = 1)
#
#                   64: (clk = 0, start = 0, car_entered = 1, is_uni_car_entered = 1, car_exited = 0, is_uni_car_exited = 0)(uni_parked_car =   5, parked_car =   0, uni_vacated_space = 495, vacated_space = 200,
#                uni_is_vacated_space = 1, is_vacated_space = 1)
#
#                   67: (clk = 0, start = 0, car_entered = 0, is_uni_car_entered = 1, car_exited = 0, is_uni_car_exited = 0)(uni_parked_car =   5, parked_car =   0, uni_vacated_space = 495, vacated_space = 200,
#                uni_is_vacated_space = 1, is_vacated_space = 1)
#
#                   85: (clk = 0, start = 0, car_entered = 1, is_uni_car_entered = 0, car_exited = 0, is_uni_car_exited = 0)(uni_parked_car =   5, parked_car =   1, uni_vacated_space = 495, vacated_space = 199,
#                uni_is_vacated_space = 1, is_vacated_space = 1)
#
#                   88: (clk = 0, start = 0, car_entered = 0, is_uni_car_entered = 0, car_exited = 0, is_uni_car_exited = 0)(uni_parked_car =   5, parked_car =   1, uni_vacated_space = 495, vacated_space = 199,
#                uni_is_vacated_space = 1, is_vacated_space = 1)
#
#                   91: (clk = 0, start = 0, car_entered = 1, is_uni_car_entered = 0, car_exited = 0, is_uni_car_exited = 0)(uni_parked_car =   5, parked_car =   2, uni_vacated_space = 495, vacated_space = 198,
#                uni_is_vacated_space = 1, is_vacated_space = 1)
#
#                   94: (clk = 0, start = 0, car_entered = 0, is_uni_car_entered = 0, car_exited = 0, is_uni_car_exited = 0)(uni_parked_car =   5, parked_car =   2, uni_vacated_space = 495, vacated_space = 198,
#                uni_is_vacated_space = 1, is_vacated_space = 1)
#
#                   97: (clk = 0, start = 0, car_entered = 1, is_uni_car_entered = 0, car_exited = 0, is_uni_car_exited = 0)(uni_parked_car =   5, parked_car =   3, uni_vacated_space = 495, vacated_space = 197,
#                uni_is_vacated_space = 1, is_vacated_space = 1)
#
#                  100: (clk = 0, start = 0, car_entered = 0, is_uni_car_entered = 0, car_exited = 0, is_uni_car_exited = 0)(uni_parked_car =   5, parked_car =   3, uni_vacated_space = 495, vacated_space = 197,
#                uni_is_vacated_space = 1, is_vacated_space = 1)
#
#                  101: (clk = 1, start = 0, car_entered = 0, is_uni_car_entered = 0, car_exited = 0, is_uni_car_exited = 0)(uni_parked_car =   5, parked_car =   3, uni_vacated_space = 495, vacated_space = 197,
#                uni_is_vacated_space = 1, is_vacated_space = 1)
#
#                  102: (clk = 0, start = 0, car_entered = 0, is_uni_car_entered = 0, car_exited = 0, is_uni_car_exited = 0)(uni_parked_car =   5, parked_car =   3, uni_vacated_space = 495, vacated_space = 197,
#                uni_is_vacated_space = 1, is_vacated_space = 1)
#
#                  103: (clk = 0, start = 0, car_entered = 1, is_uni_car_entered = 0, car_exited = 0, is_uni_car_exited = 0)(uni_parked_car =   5, parked_car =   4, uni_vacated_space = 495, vacated_space = 196,
#                uni is vacated space = 1, is vacated space = 1)
#                  103: (clk = 0, start = 0, car_entered = 1, is_uni_car_entered = 0, car_exited = 0, is_uni_car_exited = 0)(uni_parked_car =   5, parked_car =   4, uni_vacated_space = 495, vacated_space = 196,
#                uni_is_vacated_space = 1, is_vacated_space = 1)
#
#                  106: (clk = 0, start = 0, car_entered = 0, is_uni_car_entered = 0, car_exited = 0, is_uni_car_exited = 0)(uni_parked_car =   5, parked_car =   4, uni_vacated_space = 495, vacated_space = 196,
#                uni_is_vacated_space = 1, is_vacated_space = 1)
#
#                  109: (clk = 0, start = 0, car_entered = 1, is_uni_car_entered = 0, car_exited = 0, is_uni_car_exited = 0)(uni_parked_car =   5, parked_car =   5, uni_vacated_space = 495, vacated_space = 195,
#                uni_is_vacated_space = 1, is_vacated_space = 1)
#
#                  112: (clk = 0, start = 0, car_entered = 0, is_uni_car_entered = 0, car_exited = 0, is_uni_car_exited = 0)(uni_parked_car =   5, parked_car =   5, uni_vacated_space = 495, vacated_space = 195,
#                uni_is_vacated_space = 1, is_vacated_space = 1)
#
#                  130: (clk = 0, start = 0, car_entered = 0, is_uni_car_entered = 0, car_exited = 1, is_uni_car_exited = 1)(uni_parked_car =   4, parked_car =   5, uni_vacated_space = 496, vacated_space = 195,
#                uni_is_vacated_space = 1, is_vacated_space = 1)
#
#                  133: (clk = 0, start = 0, car_entered = 0, is_uni_car_entered = 0, car_exited = 0, is_uni_car_exited = 1)(uni_parked_car =   4, parked_car =   5, uni_vacated_space = 496, vacated_space = 195,
#                uni_is_vacated_space = 1, is_vacated_space = 1)
#
#                  136: (clk = 0, start = 0, car_entered = 0, is_uni_car_entered = 0, car_exited = 1, is_uni_car_exited = 1)(uni_parked_car =   3, parked_car =   5, uni_vacated_space = 497, vacated_space = 195,
#                uni_is_vacated_space = 1, is_vacated_space = 1)
#
#                  139: (clk = 0, start = 0, car_entered = 0, is_uni_car_entered = 0, car_exited = 0, is_uni_car_exited = 1)(uni_parked_car =   3, parked_car =   5, uni_vacated_space = 497, vacated_space = 195,
#                uni_is_vacated_space = 1, is_vacated_space = 1)
#
#                  142: (clk = 0, start = 0, car_entered = 0, is_uni_car_entered = 0, car_exited = 1, is_uni_car_exited = 1)(uni_parked_car =   2, parked_car =   5, uni_vacated_space = 498, vacated_space = 195,
#                uni_is_vacated_space = 1, is_vacated_space = 1)
#
#                  145: (clk = 0, start = 0, car_entered = 0, is_uni_car_entered = 0, car_exited = 0, is_uni_car_exited = 1)(uni_parked_car =   2, parked_car =   5, uni_vacated_space = 498, vacated_space = 195,
#                uni_is_vacated_space = 1, is_vacated_space = 1)
#
#                  152: (clk = 1, start = 0, car_entered = 0, is_uni_car_entered = 0, car_exited = 0, is_uni_car_exited = 1)(uni_parked_car =   2, parked_car =   5, uni_vacated_space = 498, vacated_space = 195,
#                uni_is_vacated_space = 1, is_vacated_space = 1)
#
#                  153: (clk = 0, start = 0, car_entered = 0, is_uni_car_entered = 0, car_exited = 0, is_uni_car_exited = 1)(uni_parked_car =   2, parked_car =   5, uni_vacated_space = 498, vacated_space = 195,
#                uni_is_vacated_space = 1, is_vacated_space = 1)
#
#                  163: (clk = 0, start = 0, car_entered = 0, is_uni_car_entered = 0, car_exited = 1, is_uni_car_exited = 0)(uni_parked_car =   2, parked_car =   4, uni_vacated_space = 498, vacated_space = 196,
#                uni is vacated space = 1, is vacated space = 1)
#                  166: (clk = 0, start = 0, car_entered = 0, is_uni_car_entered = 0, car_exited = 0, is_uni_car_exited = 0)(uni_parked_car =   2, parked_car =   4, uni_vacated_space = 498, vacated_space = 196,
#                uni_is_vacated_space = 1, is_vacated_space = 1)
#
#                  169: (clk = 0, start = 0, car_entered = 0, is_uni_car_entered = 0, car_exited = 1, is_uni_car_exited = 0)(uni_parked_car =   2, parked_car =   3, uni_vacated_space = 498, vacated_space = 197,
#                uni_is_vacated_space = 1, is_vacated_space = 1)
#
#                  172: (clk = 0, start = 0, car_entered = 0, is_uni_car_entered = 0, car_exited = 0, is_uni_car_exited = 0)(uni_parked_car =   2, parked_car =   3, uni_vacated_space = 498, vacated_space = 197,
#                uni_is_vacated_space = 1, is_vacated_space = 1)
#
#                  175: (clk = 0, start = 0, car_entered = 0, is_uni_car_entered = 0, car_exited = 1, is_uni_car_exited = 0)(uni_parked_car =   2, parked_car =   2, uni_vacated_space = 498, vacated_space = 198,
#                uni_is_vacated_space = 1, is_vacated_space = 1)
#
#                  178: (clk = 0, start = 0, car_entered = 0, is_uni_car_entered = 0, car_exited = 0, is_uni_car_exited = 0)(uni_parked_car =   2, parked_car =   2, uni_vacated_space = 498, vacated_space = 198,
#                uni_is_vacated_space = 1, is_vacated_space = 1)
#
# Break in Module tb at C:/Users/ASUS/OneDrive/Documents/Semester4/DSD/midterm_q8(myslef)/tb.v line 81
```

Figure 1 خروجی