امیرحسین شهیدی ۴۰۱۱۰۶۱۳۶

سوال ۸:

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

[1] Vector Processor

تعداد خودروهای پارک شده در پارکینگ مربوط به ظرفیت آزاد	parked_care
تعداد فضای خالی متعلق به دانشگاه	uni_vacated_space
تعداد فضاهای خالی مربوط به ظرفیت آزاد	vacated_space
آیا فضای خالی برای دانشگاه موجود است؟	uni_is_vacated_space
آیا فضای خالی برای ظرفیت آزاد موجود است؟	is_vacated_space
ورودی‌ها	
ورود یک خودرو	car_entered
آیا خودرو وارد شده متعلق به دانشگاه است؟	is_uni_car_entered
خروج یک خودرو	car_exited
آیا خودرو خارج شده متعلق به دانشگاه است؟	is_uni_car_ exited

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).

ابتدا ماژول اصلی پارکینگ که به شرح زیر است می نویسیم:

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

    reg [10:0] uni_capacity;
    reg [10:0] capacity;
    wire hour;
    integer i;
    reg [10:0] changed_capacity;

    initial begin
        i = 8;
        uni_capacity = 500;
        capacity = 200;
        uni_parked_car = 0;
        parked_car = 0;
        uni_vacated_space = 500;
        vacated_space = 200;
    end

   //localparam n = 5;
    //reg hour_reg;
    //integer hour_counter;
//
    //initial begin
    //    hour_reg = 0;
    //    hour_counter = 0;
    //end
    //always @(posedge clk) begin
    //    hour_counter = hour_counter + 1;
    //    if (hour_counter % n == 0)
    //        hour_reg = 1;
    //    else
    //        hour_reg = 0;
    //end
    //assign hour = hour_reg;

    ClockCounter #(5) clock_counter(clk, hour);
    always @(posedge hour or posedge clk) begin
        if (hour) begin
                i = i+1;
                 if (i%24 == 14 || i%24== 15) begin
```

```
                changed_capacity= (uni_vacated_space<50) ? uni_vacated_space : 50;
                uni_capacity=uni_capacity-changed_capacity;
                uni_vacated_space = uni_vacated_space-changed_capacity;
                capacity = capacity+changed_capacity;
                vacated_space = vacated_space+changed_capacity;
            end else if (i%24 == 16) begin
                uni_capacity = (uni_parked_car>200) ? uni_parked_car : 200;
                capacity = 700-uni_capacity;
                vacated_space = capacity-parked_car;
                uni_vacated_space = uni_capacity-uni_parked_car;
            end
        end else if (clk) begin
            uni_is_vacated_space =(uni_vacated_space != 0) ? 1 : 0;
            is_vacated_space = (vacated_space != 0) ? 1 : 0;

            if (car_entered && is_uni_car_entered && uni_is_vacated_space) begin
                uni_parked_car= uni_parked_car+1;
                uni_vacated_space = uni_vacated_space-1;
            end else if (car_entered && !is_uni_car_entered && is_vacated_space) begin
                parked_car = parked_car+1;
                vacated_space = vacated_space-1;
            end else if (car_exited && is_uni_car_exited && uni_vacated_space > 0) begin
                uni_parked_car = uni_parked_car-1;
                uni_vacated_space = uni_vacated_space+ 1;
            end else if (car_exited && !is_uni_car_exited && vacated_space > 0) begin
                parked_car = parked_car -1;
                vacated_space = vacated_space +1;
            end
        end
    end
endmodule
```

```
module ClockCounter #(parameter n = 5) (input clk, output reg hour);
    integer i;

    initial begin
        hour = 0;
        i = 0;
    end

    always @(posedge clk) begin
        i = i + 1;
        if (i % n == 0)
            hour = 1;
        else
            hour = 0;
    end
endmodule
```

ورودی ها را مطابق خواسته سوال به علاوه کلاک لازم طراحی می کنیم و حداکثر ۱۰ بیت برای خروجی در نظر می گیریم. برای آنکه بدانیم هر ساعت فرا رسید متغیر hour را داریم که توسط i که برای محاسبه ساعت در ماژول کلاک کانتر طراحی شده محاسبه می شود.

در کلاک کانتر مثلا هر ۵ کلاک را برای رندتر بودن یه ساعت در نظر می گیریم و همانطور که در ماژول اصلی از آن استفاده می کنیم بعد از هر ۵ کلاک ساعت را یک می کنیم. البته در ماژول اصلی هم می شد این فرایند را طراحی کرد که کامنت شده است.

ظرفیت های اولیه نیز مطابق خواسته سوال در ابتدا تعیین شده است.

همانطور که گفته شده در صورتی که در هر روز به ساعت ۱۴ و ۱۵ برسیم ظرفیت آزاد ۵۰ تا زیاد و دانشگاه ۵۰ تا کم میشود (علت % گذاشتن برای محاسبه در i هایی است که از یک روز گذر کرده اند.)

در ساعت ۱۶ نیز ظرفیت آزاد ۵۰۰ و دانشگاه ۲۰۰ می شود که اگر ظرفیت دانشگاه بیشتر باشد مقدار آن جابه جا می شود.

در قسمت بعد نیز صرفا شرایط ورود و خروج ماشین ها بررسی می شود که در نهایت در کد تست بنچ زیر نتیجه قابل مشاهده می باشد.

```
module Parking_test_bench();
    reg clk;
    wire [10:0] upc, pc, uvs, vs;
    wire uivs, ivs;
    reg ci, uci, ce, uce;
    reg i;
    initial begin
        clk = 0;
        forever #5 clk = ~clk;
    end
    Parking parking(ci, uci, ce, uce, clk, upc, pc, uvs, vs, uivs, ivs);

    initial begin
        ci = 0;
        uci = 0;
        ce = 0;
        uce = 0;
        #280 ci = 1;
        #60 uci = 1;
        #40 ci = 0; uci = 0; ce = 1; uce = 1;
        #30 ci = 1; uci = 1; ce = 0; uce = 0;
        #160 $finish();
    end

    initial begin
        $monitor("%4.t: ci = %b | uci = %b | ce = %b | uce = %b --> upc = %d | pc = %d | uivs = %b | ivs = %b uvs = %d | vs = %d",
        $time, ci, uci, ce, uce, upc, pc, uivs, ivs, uvs, vs,);
    end
endmodule
```

در تست بنچ نیز موارد به اختصار نوشته شده که به ترتیب خروجی به شرح:

Ci = car entered

Uci = uni car entered

Ce = car exited

Uce = uni car exited

Upc = uni parked car

Pc = parked car

Uivs = uni is vacated space

Ivs = is vacated space

Uvs = uni vacated space

Vs = vacated space

```
#  280: ci = 1 | uci = 0 | ce = 0 | uce = 0 --> upc =    0 | pc =    0 | uivs = 1 | ivs = 1 uvs =  500 | vs =  200
#  285: ci = 1 | uci = 0 | ce = 0 | uce = 0 --> upc =    0 | pc =    1 | uivs = 1 | ivs = 1 uvs =  500 | vs =  199
#  295: ci = 1 | uci = 0 | ce = 0 | uce = 0 --> upc =    0 | pc =    1 | uivs = 1 | ivs = 1 uvs =  450 | vs =  249
#  305: ci = 1 | uci = 0 | ce = 0 | uce = 0 --> upc =    0 | pc =    2 | uivs = 1 | ivs = 1 uvs =  450 | vs =  248
#  315: ci = 1 | uci = 0 | ce = 0 | uce = 0 --> upc =    0 | pc =    3 | uivs = 1 | ivs = 1 uvs =  450 | vs =  247
#  325: ci = 1 | uci = 0 | ce = 0 | uce = 0 --> upc =    0 | pc =    4 | uivs = 1 | ivs = 1 uvs =  450 | vs =  246
#  335: ci = 1 | uci = 0 | ce = 0 | uce = 0 --> upc =    0 | pc =    5 | uivs = 1 | ivs = 1 uvs =  450 | vs =  245
#  340: ci = 1 | uci = 1 | ce = 0 | uce = 0 --> upc =    0 | pc =    5 | uivs = 1 | ivs = 1 uvs =  450 | vs =  245
#  345: ci = 1 | uci = 1 | ce = 0 | uce = 0 --> upc =    0 | pc =    5 | uivs = 1 | ivs = 1 uvs =  400 | vs =  295
#  355: ci = 1 | uci = 1 | ce = 0 | uce = 0 --> upc =    1 | pc =    5 | uivs = 1 | ivs = 1 uvs =  399 | vs =  295
#  365: ci = 1 | uci = 1 | ce = 0 | uce = 0 --> upc =    2 | pc =    5 | uivs = 1 | ivs = 1 uvs =  398 | vs =  295
#  375: ci = 1 | uci = 1 | ce = 0 | uce = 0 --> upc =    3 | pc =    5 | uivs = 1 | ivs = 1 uvs =  397 | vs =  295
#  380: ci = 0 | uci = 0 | ce = 1 | uce = 1 --> upc =    3 | pc =    5 | uivs = 1 | ivs = 1 uvs =  397 | vs =  295
#  385: ci = 0 | uci = 0 | ce = 1 | uce = 1 --> upc =    2 | pc =    5 | uivs = 1 | ivs = 1 uvs =  398 | vs =  295
#  395: ci = 0 | uci = 0 | ce = 1 | uce = 1 --> upc =    2 | pc =    5 | uivs = 1 | ivs = 1 uvs =  198 | vs =  495
#  405: ci = 0 | uci = 0 | ce = 1 | uce = 1 --> upc =    1 | pc =    5 | uivs = 1 | ivs = 1 uvs =  199 | vs =  495
#  410: ci = 1 | uci = 1 | ce = 0 | uce = 0 --> upc =    1 | pc =    5 | uivs = 1 | ivs = 1 uvs =  199 | vs =  495
#  415: ci = 1 | uci = 1 | ce = 0 | uce = 0 --> upc =    2 | pc =    5 | uivs = 1 | ivs = 1 uvs =  198 | vs =  495
#  425: ci = 1 | uci = 1 | ce = 0 | uce = 0 --> upc =    3 | pc =    5 | uivs = 1 | ivs = 1 uvs =  197 | vs =  495
#  435: ci = 1 | uci = 1 | ce = 0 | uce = 0 --> upc =    4 | pc =    5 | uivs = 1 | ivs = 1 uvs =  196 | vs =  495
#  455: ci = 1 | uci = 1 | ce = 0 | uce = 0 --> upc =    5 | pc =    5 | uivs = 1 | ivs = 1 uvs =  195 | vs =  495
#  465: ci = 1 | uci = 1 | ce = 0 | uce = 0 --> upc =    6 | pc =    5 | uivs = 1 | ivs = 1 uvs =  194 | vs =  495
#  475: ci = 1 | uci = 1 | ce = 0 | uce = 0 --> upc =    7 | pc =    5 | uivs = 1 | ivs = 1 uvs =  193 | vs =  495
#  485: ci = 1 | uci = 1 | ce = 0 | uce = 0 --> upc =    8 | pc =    5 | uivs = 1 | ivs = 1 uvs =  192 | vs =  495
#  505: ci = 1 | uci = 1 | ce = 0 | uce = 0 --> upc =    9 | pc =    5 | uivs = 1 | ivs = 1 uvs =  191 | vs =  495
#  515: ci = 1 | uci = 1 | ce = 0 | uce = 0 --> upc =   10 | pc =    5 | uivs = 1 | ivs = 1 uvs =  190 | vs =  495
#  525: ci = 1 | uci = 1 | ce = 0 | uce = 0 --> upc =   11 | pc =    5 | uivs = 1 | ivs = 1 uvs =  189 | vs =  495
#  535: ci = 1 | uci = 1 | ce = 0 | uce = 0 --> upc =   12 | pc =    5 | uivs = 1 | ivs = 1 uvs =  188 | vs =  495
#  555: ci = 1 | uci = 1 | ce = 0 | uce = 0 --> upc =   13 | pc =    5 | uivs = 1 | ivs = 1 uvs =  187 | vs =  495
```

اکنون خروجی میگیریم از کد و مشاهده می شود خواسته ها برآورده می شود(البته مواردی مثل پرشدن و اختصاص پرفیت و تغییر آن در پی آن اینجا با توجه به ظرفیت زیاد پارکینگ که پر نشده ملاحظه نمی شود) مثلا:

در زمان ۲۹۵ همانطور که باید ۵۰ تا از ظرفیت کم میشود زیرا ساعت ۱۴ است.

ورود ماشین ها تا ۵۰ واحد بعد صورت می گیرد و مطابق خواسته ۵۰ تا از ظرفیت نیز جابه جا می شود.

در زمان ۳۹۵ نیز ظرفیت به حالت نرمال می رسد و ۲۰۰ تا به آن اضافه می شود.

ب) در کوارتوس با ساخت پروژه باید باید timequest timing analyzer را انتخاب کرده و create timing netlist را انجام دهیم.

و گزارش فرکانس ماکسیمم و دیلی ها را دریافت کنیم که به دلیل مشکل در کوارتوس قادر به ساخت آن نشدم اما نتیجه این بخش باید به صورتی باشد که فرکانس بیشینه همان کلاک برنامه شود و معکوس تاخیر ما برابر فرکانس شود که نتیجه درست باشد.