

به نام خدا



# Digital System Design

طراحی سیستم‌های دیجیتال

---

نام استاد : دکتر فصحتی

نام دانشجو : امیررضا اینانلو

شماره دانشجویی : 401105667

## سوال 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) باتوجه به 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 می یابد و در ساعت ۱۶ ظرفیت ورود آزاد به ۵۰۰ خودرو می رسد.

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

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

<sup>۴</sup> Vector Processor

۳

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؟

در صورتی که 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).

همانطور که می بینید دو ورودی جدید نسبت به ورودی های 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 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
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 طبیعتاً 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
end
```

```

    end
end else begin
    // do nothing
end
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;
        end
    end
end

```

```

        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
        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
        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
        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
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 پنج 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.)

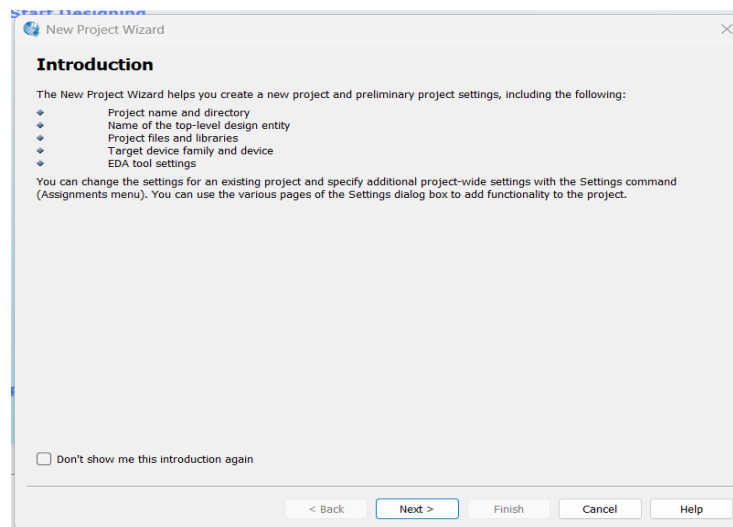
خروجی تست بنچ در صفحه ی بعد قرار دارد.

Figure 1 خروجی

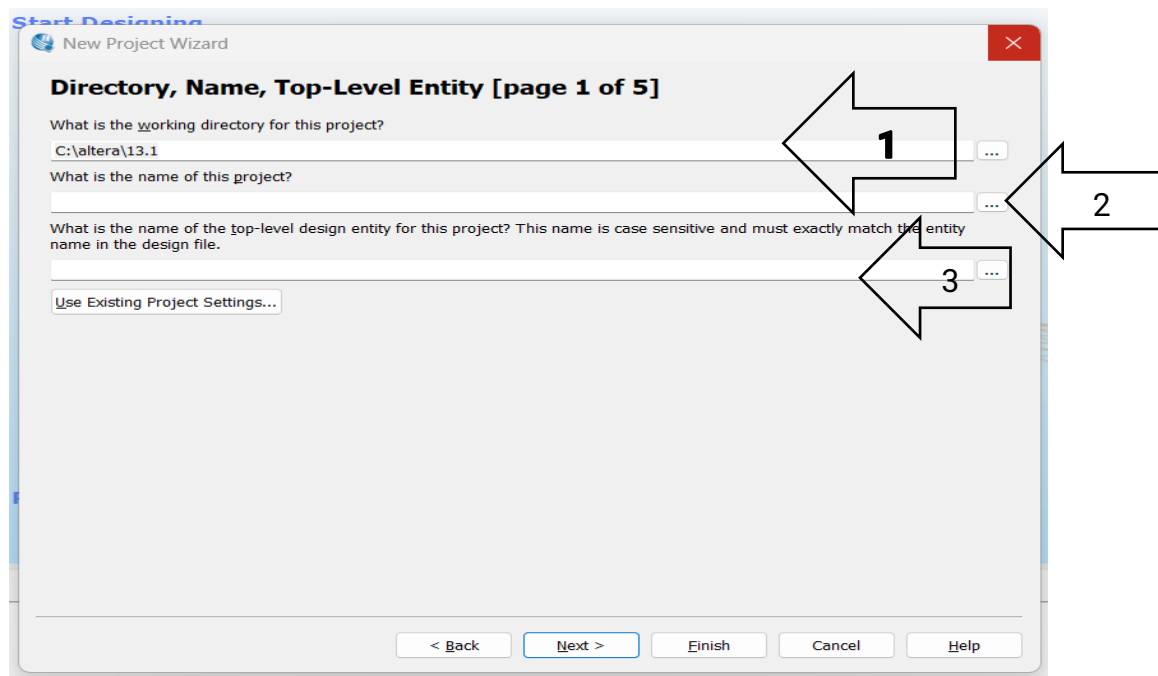


ب) در این قسمت از پروژه به صورت زیر عمل می‌کنیم.  
ابتدا یکی از نسخه‌های برنامه‌ی Quartus را نصب می‌کنیم.  
(نسخه‌ای که از آن استفاده کردم Quartus II 64\_bit بود).  
سپس مراحل زیر را به ترتیب انجام می‌دهیم.

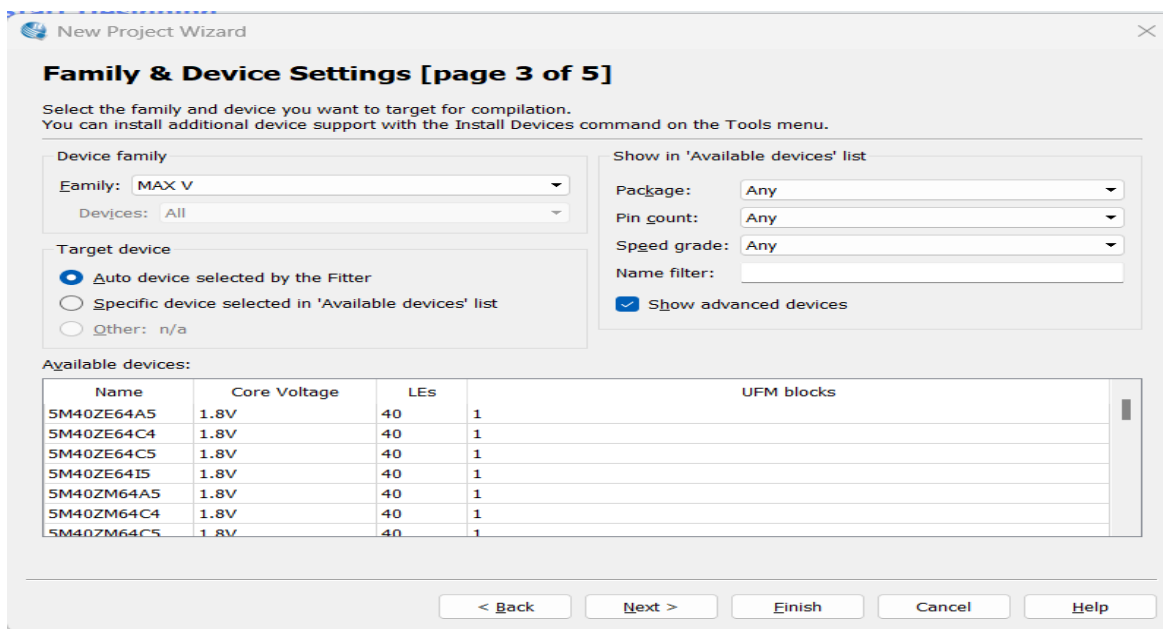
ابتدا برنامه‌ی Quartus را باز کرده و در نوار بالای صفحه از بخش File  
بر روی گزینه‌ی New Project Wizard کلیک می‌کنیم.  
پنجره زیر نمایش داده می‌شود.



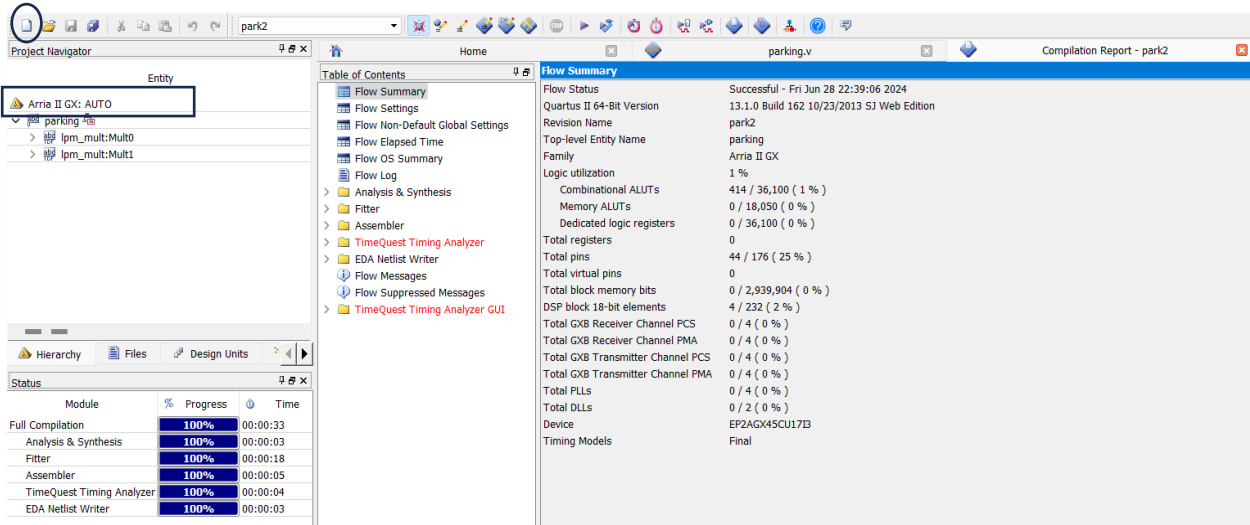
بر روی Next کلیک می‌کنیم. سپس در مرحله بعد فولدری را که می‌خواهیم پروژه در آن جا قرار بگیرد انتخاب می‌کنیم و یک اسم هم برای پروژه خود انتخاب می‌کنیم. (در تصویر زیر در بخش 1 فولدر مد نظر و در بخش‌های 2 و 3 نام دلخواهی که می‌خواهید به پروژه بدهید را قرار دهید).



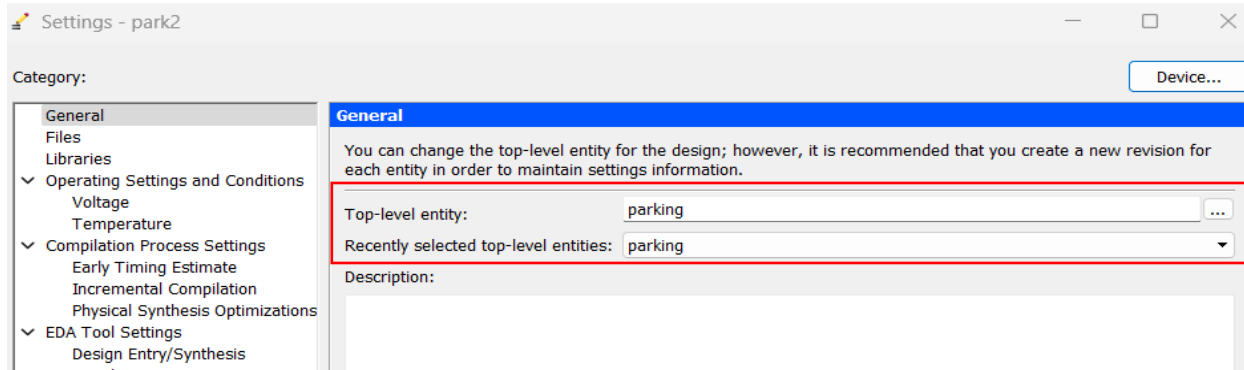
در مرحله بعد هم Next را بزنید. سپس در پنجره زیر Device family مد نظر را انتخاب کنید. (که ما از Arria II GX استفاده کردیم. این Devince را از قبل از سایت رسمی intel نصب کرده بودیم).



در مرحله بعد هم روی Next کلیک کنید. اکنون پروژه ما ساخته شده است. اکنون فایل برای ساخت فایل وریداگ بر روی بخش 1 تصویر زیر کلیک می کنیم و گزینه Verilog HDL File را انتخاب می کنیم.

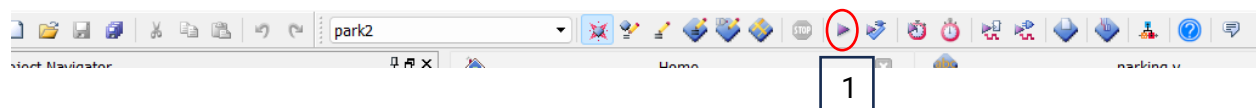


سپس کد وریداگی که در بخش الف نوشته بودیم را در فایل ایجاد شده کپی می کنیم و این فایل را ذخیره می کنیم. سپس بر روی بخش 2 در تصویر بالا راست کلیک می کنیم و روی settings کلیک می کنیم. در پنجره ای که باز می شود به تب General می رویم و برای بخش های نشان داده شده نام فایل وریداگ را به عنوان Top\_level\_entity قرار می دهیم. که نام فایل ما parking بود.

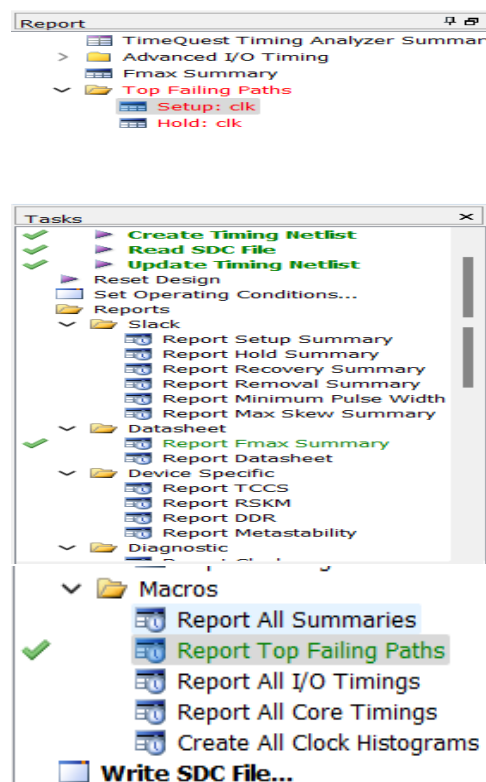


از بخش Device در بالا سمت راست هم می توانیم FPGA مدنظر را انتخاب کنیم. توجه کنید که باید از قبل آن را نصب کرده باشیم.

در نهایت در پایین همین پنجره بر روی گزینه apply کلیک می کنیم و OK را می زنیم. سپس روی فایل وریداگ راست کلیک می کنیم و از نوار بالای صفحه بر روی گزینه کامپایل می زنیم و صبر می کنیم تا این مرحله تمام شود.



سپس از نوار بالای صفحه از بخش tools، TimeQuest Timing Analyser را انتخاب می کنیم. و سپس گزینه های تیک زده شده در تصویر زیر را به ترتیب از بالا به پایین می زنیم.



Report Fmax Summary شامل اطلاعاتی از جمله ماکسیمم فرکانس مدار (کد ورپلاگ) است.

فایل Report Top Falling Paths هم تاخیر را به ازای مسیرهای مختلف نشان می دهد.

خروجی Fmax Summary به صورت زیر است:

Fmax Summary				
	Fmax	Restricted Fmax	Clock Name	Note
1	40.86 MHz	40.86 MHz	clk	

که نشان می دهد ماکسیمم فرکانس این طراحی برابر 40.86 مگاهرتز است. این ماکسیمم فرکانس ناشی از کمترین تاخیر داده ای است که در Report Top Falling Paths موجود است.

برای محاسبه ی ماکسیمم فرکانس از داده های موجود در فایل Report Top Falling Paths از رابطه ی زیر استفاده می کنیم.

$$T(\text{clk}) = T(\text{data}) + T(\text{setup}) + T(\text{skew}) - \text{Slack}$$

منفی بودن slack یعنی اینکه سیگنال ها به موقع به مقصد نمی رسند. پس یعنی وقتی منفی می شود در فرمول بالا به  $T(\text{clk})$  افزوده می شود.

فایل Report Top Falling Paths به صورت زیر است.

	Slack	From Node	To Node	Launch Clock	Latch Clock	Relationship	Clock Skew	Data Delay
1	-11.738	hour[1]	uni_is_vacated_space\$latch	clk	clk	0.500	0.575	12.180
2	-11.727	hour[1]	uni_is_vacated_space\$latch	clk	clk	0.500	0.575	12.169
3	-11.687	hour[1]	uni_is_vacated_space\$latch	clk	clk	0.500	0.575	12.129
4	-11.686	hour[1]	uni_is_vacated_space\$latch	clk	clk	0.500	0.575	12.128
5	-11.686	hour[1]	uni_is_vacated_space\$latch	clk	clk	0.500	0.575	12.128
6	-11.686	hour[1]	uni_is_vacated_space\$latch	clk	clk	0.500	0.575	12.128
7	-11.686	hour[1]	uni_is_vacated_space\$latch	clk	clk	0.500	0.575	12.128
8	-11.682	hour[1]	uni_is_vacated_space\$latch	clk	clk	0.500	0.575	12.124
9	-11.682	hour[1]	uni_is_vacated_space\$latch	clk	clk	0.500	0.575	12.124
10	-11.676	hour[1]	uni_is_vacated_space\$latch	clk	clk	0.500	0.575	12.118
11	-11.675	hour[1]	uni_is_vacated_space\$latch	clk	clk	0.500	0.575	12.117
12	-11.675	hour[1]	uni_is_vacated_space\$latch	clk	clk	0.500	0.575	12.117
13	-11.675	hour[1]	uni_is_vacated_space\$latch	clk	clk	0.500	0.575	12.117
14	-11.675	hour[1]	uni_is_vacated_space\$latch	clk	clk	0.500	0.575	12.117
15	-11.671	hour[1]	uni_is_vacated_space\$latch	clk	clk	0.500	0.575	12.113
16	-11.671	hour[1]	uni_is_vacated_space\$latch	clk	clk	0.500	0.575	12.113
17	-11.665	hour[0]	uni_is_vacated_space\$latch	clk	clk	0.500	0.497	12.029
18	-11.654	hour[0]	uni_is_vacated_space\$latch	clk	clk	0.500	0.497	12.018
19	-11.645	hour[1]	uni_is_vacated_space\$latch	clk	clk	0.500	0.575	12.087
20	-11.645	hour[1]	uni_is_vacated_space\$latch	clk	clk	0.500	0.575	12.087
21	-11.645	hour[1]	uni_is_vacated_space\$latch	clk	clk	0.500	0.575	12.087
22	-11.645	hour[1]	uni_is_vacated_space\$latch	clk	clk	0.500	0.575	12.087
23	-11.645	hour[1]	uni_is_vacated_space\$latch	clk	clk	0.500	0.575	12.087
24	-11.645	hour[1]	uni_is_vacated_space\$latch	clk	clk	0.500	0.575	12.087
25	-11.645	hour[1]	uni_is_vacated_space\$latch	clk	clk	0.500	0.575	12.087
26	-11.645	hour[1]	uni_is_vacated_space\$latch	clk	clk	0.500	0.575	12.087
27	-11.642	hour[1]	uni_is_vacated_space\$latch	clk	clk	0.500	0.575	12.084
28	-11.642	hour[1]	uni_is_vacated_space\$latch	clk	clk	0.500	0.575	12.084
29	-11.642	hour[1]	uni_is_vacated_space\$latch	clk	clk	0.500	0.575	12.084
30	-11.642	hour[1]	uni_is_vacated_space\$latch	clk	clk	0.500	0.575	12.084

همانطور که مشخص است تاخیر اکثر داده ها در حدود 12.053 است (این لیست خیلی بلند تر از این است و فقط 30 سطر اول در تصویر بالا قرار دارند). و slack هم برابر با 11.611- در نظر می گیریم.

این اعداد را در رابطه ای که گفتیم قرار می دهیم.

$$T(\text{clk}) = 12.053\text{ns} + 0.575\text{ns} - (-11.611\text{ns})$$

$$T_{\text{clk}} = 12.053 + 0.575 + 11.611$$

$$T_{\text{clk}} = 24.239\text{ns}$$

و می دانیم که کمترین تاخیر کلاک منجر به بیشترین فرکانس می شود. (تاخیر داده ها در یک رنج مشخص بودند. یعنی از 12.02 تا 12.12)

$$F_{\text{max}} = (1 / T(\text{clk}))$$

و با قرار داده  $T_{\text{clk}}$  در این رابطه داریم:

$$F_{\text{max}} = 1 / (14.239 * 10^{-9}) \rightarrow \text{HZ}$$

$$F_{\text{max}} \approx 41.26\text{MHz}$$

ماکسیمم فرکانسی که از این رابطه در آوردیم با تقریب دهگان برابر با همان ماکسیمم فرکانسی است که در فایل  $F_{\text{max}}$  وجود داشت.