

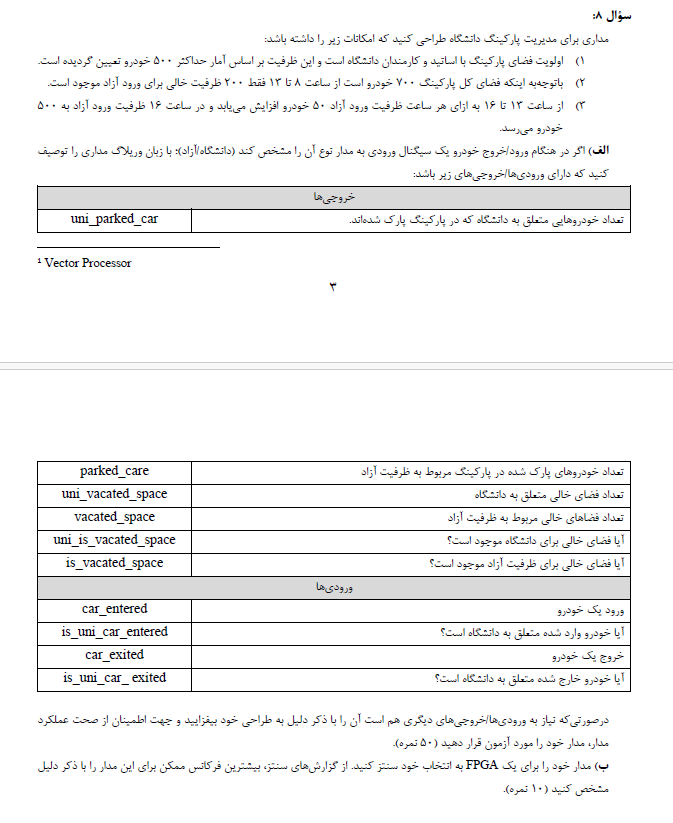
Digital System Design

طراحی سیستم‌های دیجیتال

نام استاد : دکتر فصحتی

نام دانشجو : امیررضا اینانلو

شماره دانشجویی : 401105667



الف) ورودی‌های به صورت زیر هستند.

start, clk, car\_entered, is\_uni\_car\_entered, car\_exited, 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.

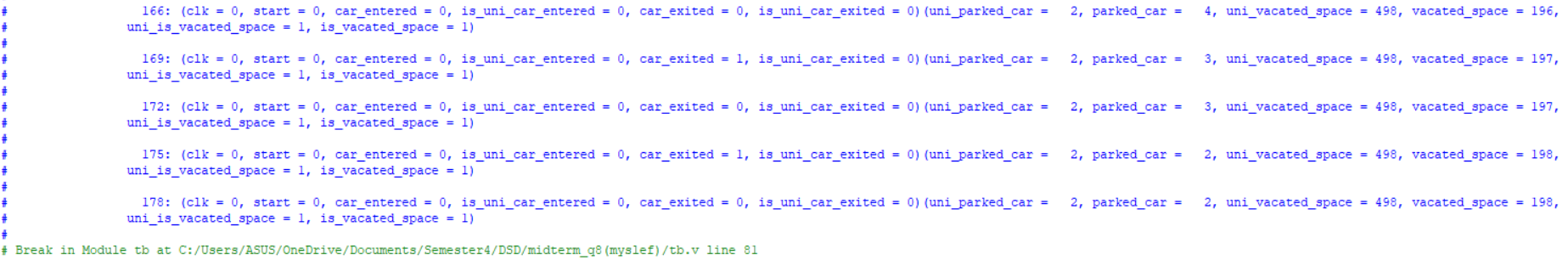
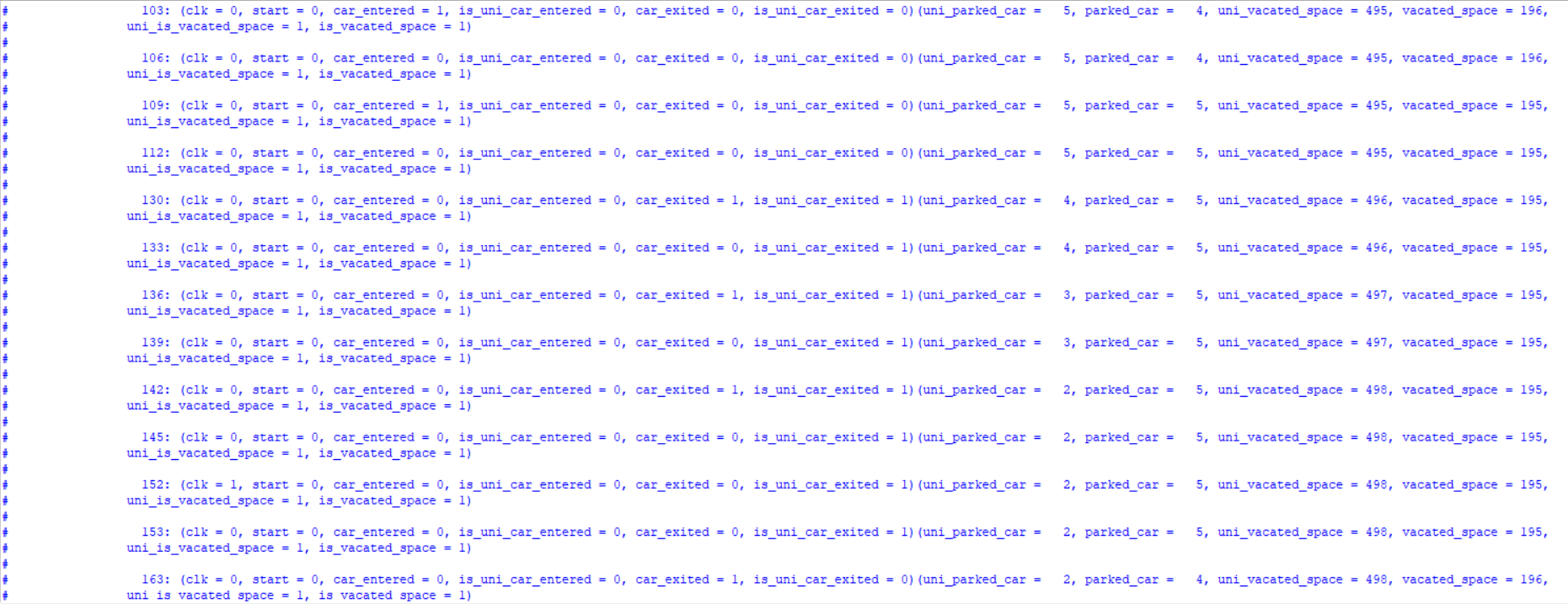
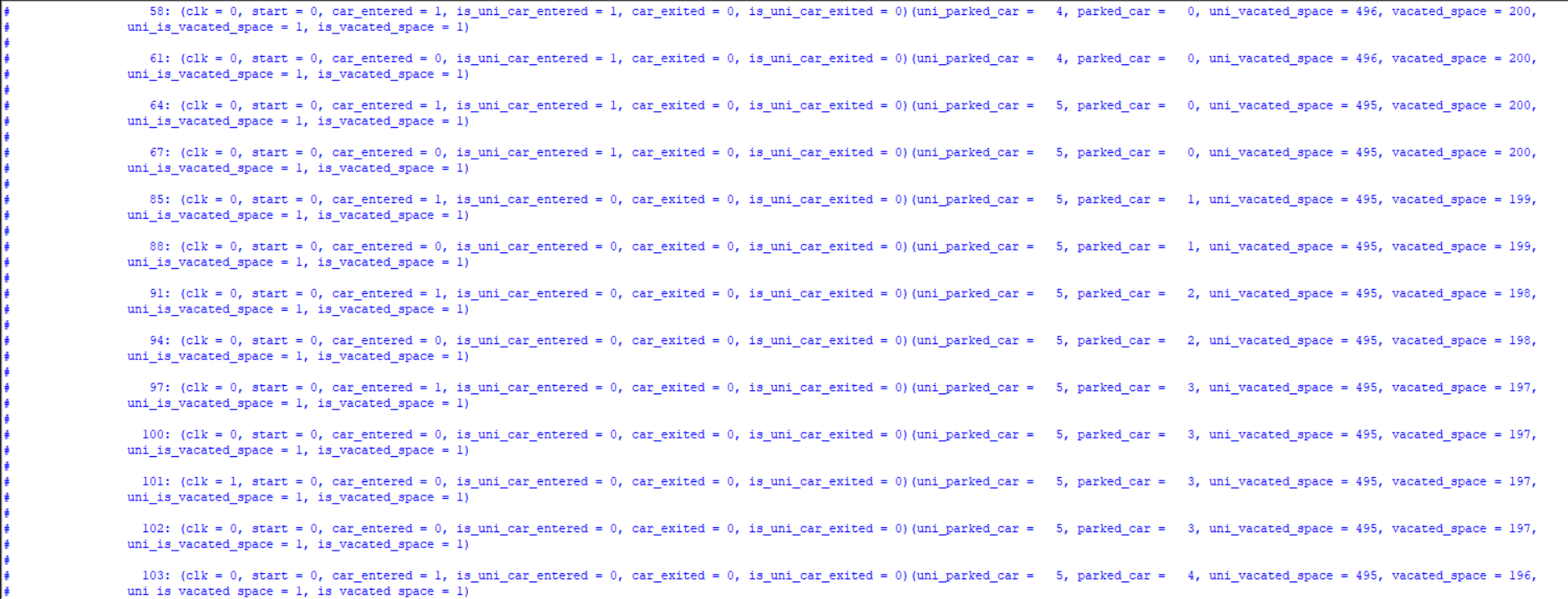
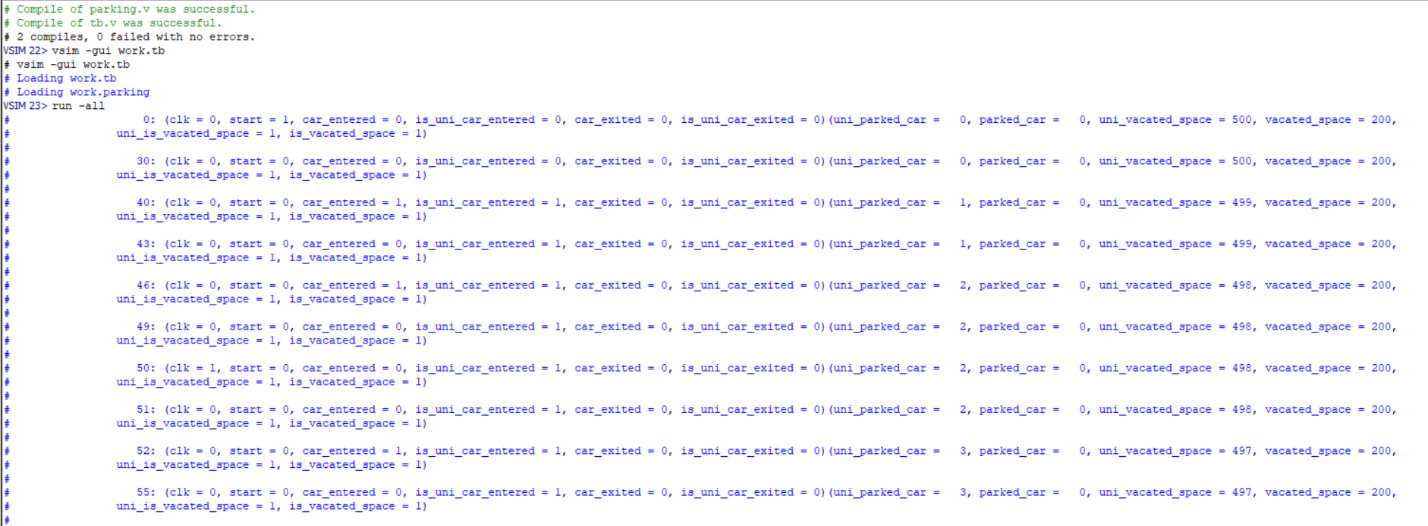


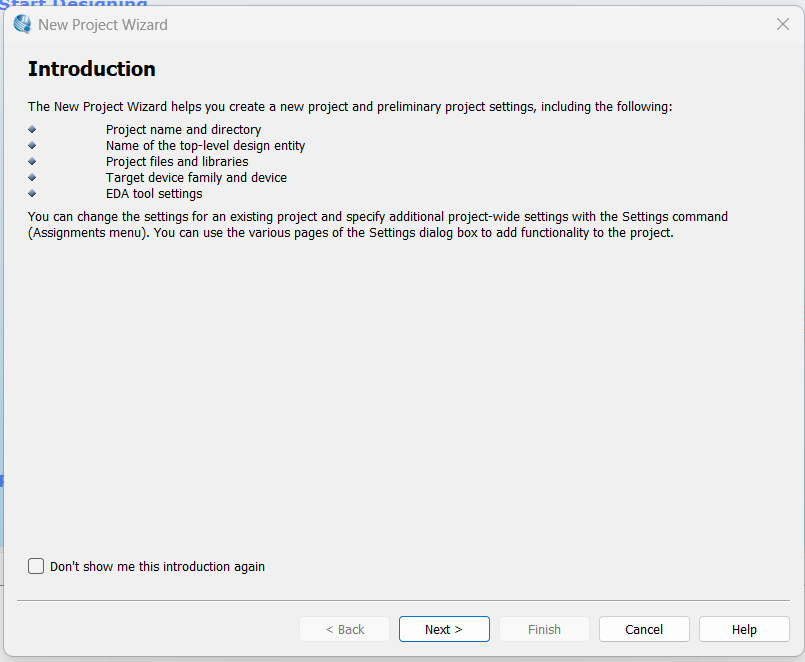
Figure 1خروجی

ب) در این قسمت از پروژه به صورت زیر عمل می‌کنیم.

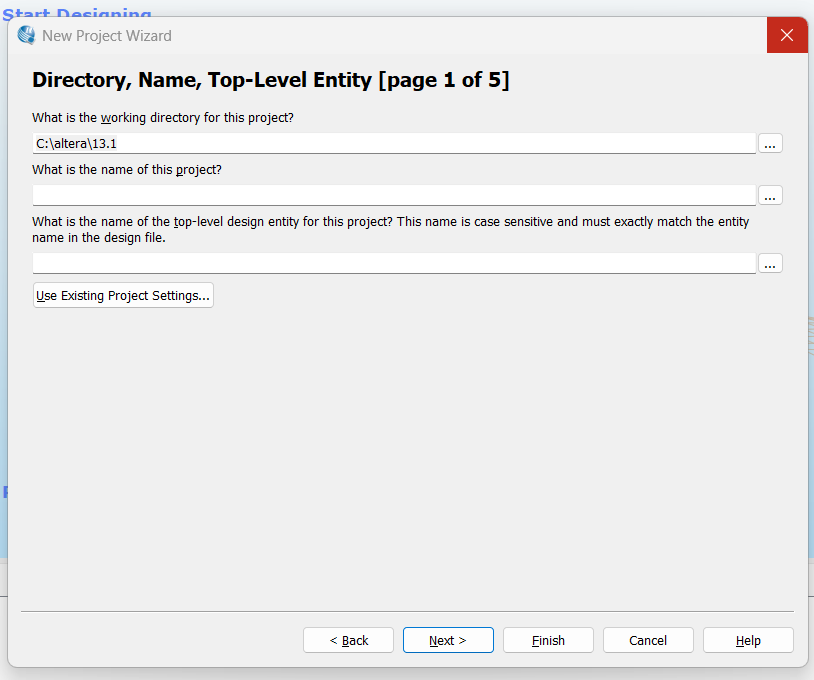
ابتدا یکی از نسخه‌های برنامه‌ی Quartus را نصب می‌کنیم. (نسخه‌ای که از آن استفاده کردم Quartus II 64\_bit بود.)

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 می شود.

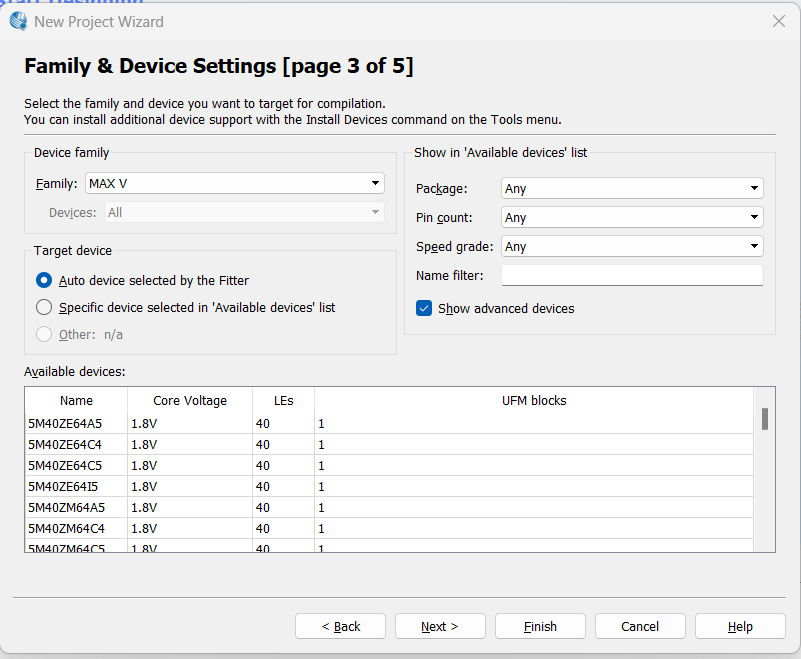
بر روی Next کلیک می کنیم. سپس در مرحله بعد فولدری را که می خواهیم پروژه در آن جا قرار بگیرد انتخاب میکنیم و یک اسم هم برای پروژه خود انتخاب می کنیم.(در تصویر زیر در بخش 1 فولدر مد نظر و در بخش‌های 2 و 3 نام دلخواهی که می‌خواهید به پروژه بدهید را قرار دهید.)



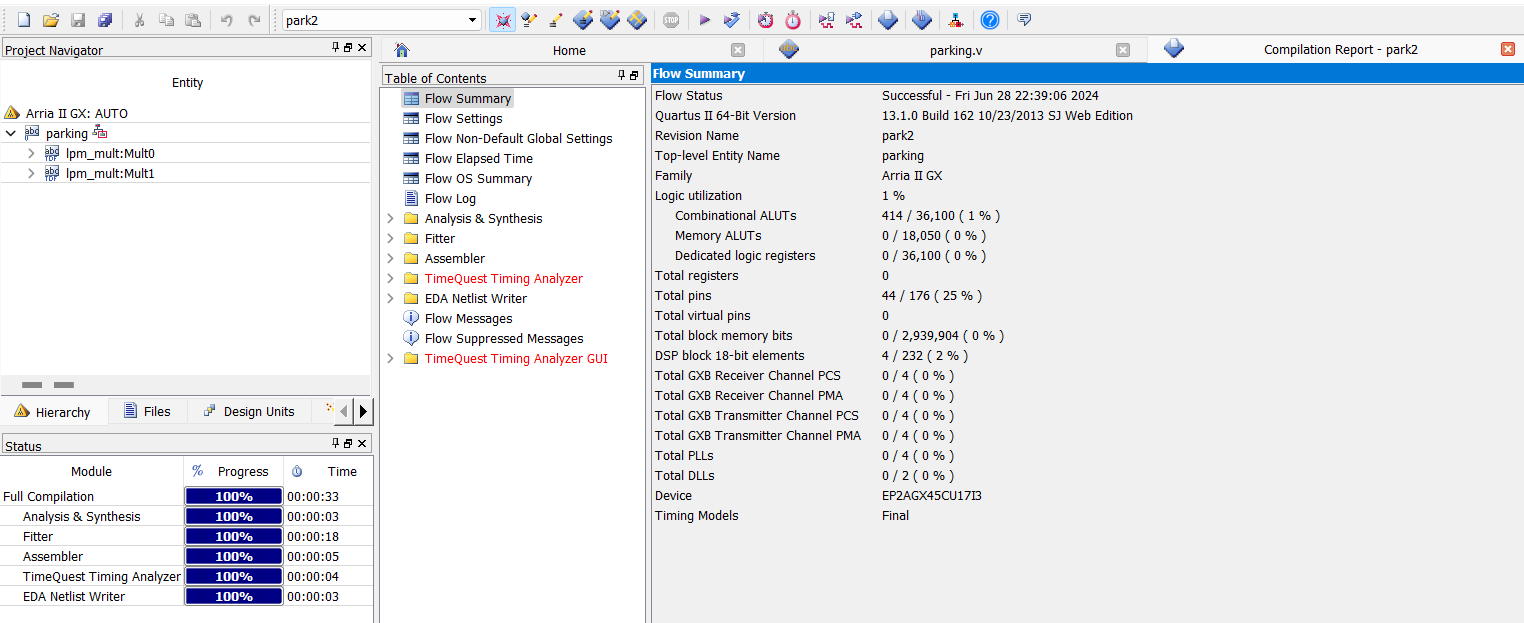
**1**

3

2

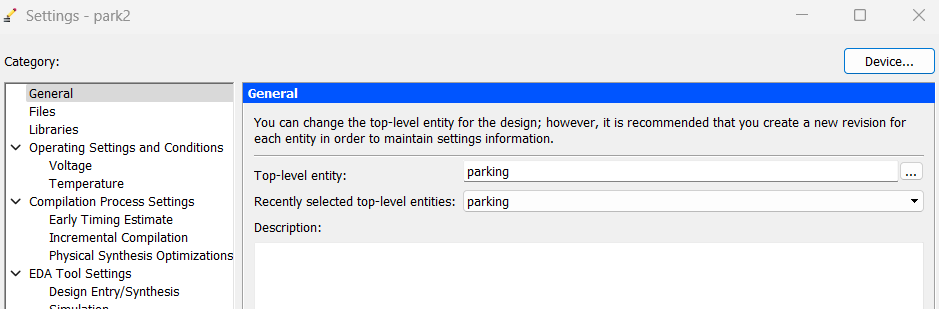
در مرحله بعد هم Next را بزنید. سپس در پنجره زیر Device family مد نظر را انتخاب کنید.(که ما از Arria II GX استفاده کردیم. این Devince را از قبل از سایت رسمی intel نصب کرده بودیم.)

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



1

2

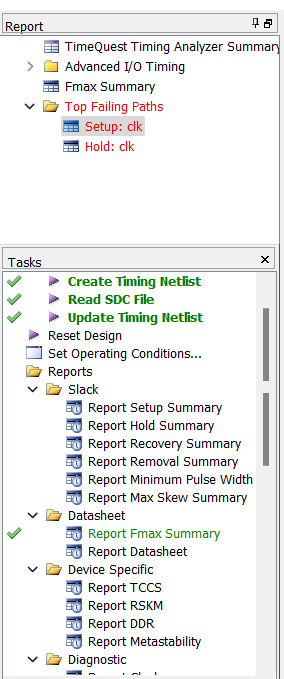
سپس کد وریلاگی که در بخش الف نوشته بودیم را در فایل ایجاد شده کپی می کنیم و این فایل را ذخیره می کنیم.سپس بر روی بخش 2 در تصویر بالا راست کلیک می کنیم و روی settings کلیک می کنیم. در پنجره ای که باز می شود به تب General می رویم و برای بخش‌های نشان داده شده نام فایل وریلاگ را به عنوان Top\_level\_entity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

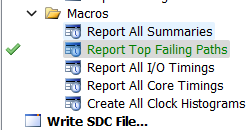
از بخش Device در بالا سمت راست هم می توانیم FPGA مدنظر را انتخاب کنیم. توجه کنید که باید از قبل آن را نصب کرده باشیم.

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



1

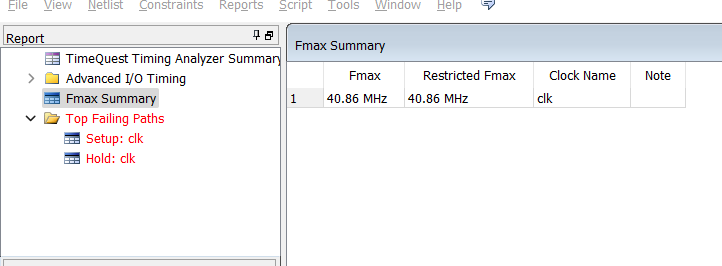
سپس از نوار بالای صفحه از بخش tools، TimeQuest Timing Analyser را انتخاب می کنیم. و سپس گزینه های تیک زده شده در تصویر زیر را به ترتیب از بالا به پایین می زنیم.



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

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

خروجی Fmax Summary به صورت زیر است:



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)clk(​=T)data​(+T)setup(​+T)skew(​−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(clk) افزوده می شود.

فایل Report Top Falling Paths به صورت زیر است.



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 1-11.61 در نظر می گیریم.

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

T)clk(​=12.053ns+0.575ns−(−11.611ns)

Tclk​=12.053+0.575+11.611

Tclk​=24.239ns

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)

Fmax​=(1 / T(clk))

و با قرار داده Tclk در این رابطه داریم:

Fmax​=1 / (14.239 \* 10^-9) 🡪 HZ

Fmax​≈41.26MHz

ماکسیمم فرکانسی که از این رابطه در آوردیم با تقریب دهگان برابر با همان ماکسیمم فرکانسی است که در فایل Fmax وجود داشت.