گزارش آزمایش سوم

اعضای گروه:

صادق محمدیان:401109477

متین محمدی:401110329

امیرحسین ملک محمدی:401106577

هدف آزمایش:

هدف از انجام این آزمایش، طراحی دو نوع مقایسه کننده است. در بخش اول یک مقایسه کننده 4 بیتی را به کمک 4 مقایسه کننده cascadable_1_bit_comparator طراحی می‌کنیم و در بخش دوم نیز به کمک یک مقایسه کننده سریال، با وارد کردن ارقام از رقم پر ارزش به کم ارزش، دو عدد را مقایسه می‌کنیم.

توضیح آزمایش:

قسمت اول:

برای این قسمت ابتدا ماژول cascadable_1_bit_comparator را طراحی می کنیم که شمای کلی آن به شکل زیر است:

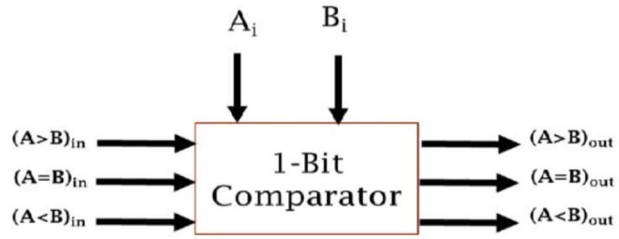


Fig 3: 1-Bit Cascaded Comparator

سپس با کنار هم قرار دادن ۴ cascadable_1_bit_comparator یک 4bit comparator می‌سازیم که شمای کلی آن به شکل زیر است:

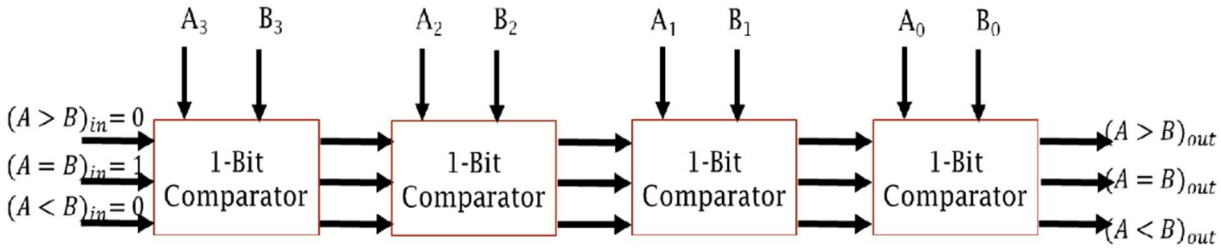


Fig 2: 4-Bit Comparator using 1-bit Cascaded Comparator

ماژول cascadable_1_bit_comparator :

```
module CASCADABLE_1_bit_COMPARATOR(input la, input eq, input le,input a, input b , output laa , output eqq , output lee);

    assign eqq = eq & (a==b);
    assign laa = (eq & (a>b)) | la ;
    assign lee = (eq & (a<b)) | le ;

endmodule
```

1 Figure

ماژول four_bit_comparator :

```
module FOUR_BIT_COMPARATOR(input [3:0]a, input [3:0]b, output la, output eq, output le);
    CASCADABLE_1_bit_COMPARATOR comp1(0,1,0,a[3],b[3],laa1,eqq1,lee1);
    CASCADABLE_1_bit_COMPARATOR comp2(laa1,eqq1,lee1,a[2],b[2],laa2,eqq2,lee2);
    CASCADABLE_1_bit_COMPARATOR comp3(laa2,eqq2,lee2,a[1],b[1],laa3,eqq3,lee3);
    CASCADABLE_1_bit_COMPARATOR comp4(laa3,eqq3,lee3,a[0],b[0],la,eq,le);
endmodule
```

2 Figure

روش کار به این صورت است که با مقایسه پر ارزش ترین بیت شروع می کنیم و بیت به بیت مقایسه می کنیم و و با استفاده از نتایج مقایسه بیت قبلی مقدار های la, eq, le را آپدیت می کنیم.باید توجه کنیم مقادیر ورودی

laa,eqq,lee ورودی به اولین cascadable_1_bit_comparator باید به ترتیب 0,1,0 باشند.

ماژول تست آن به صورت زیر می باشد:

```
module tb();
    reg [3:0]a;
    reg [3:0]b;
    wire la,eq,le;

    FOUR_BIT_COMPARATOR comp(a,b,la,eq,le);

    initial
    begin
    a=4'b1111 ;
    b=4'b1011 ;
    #10
    $display("a = ", a ," b = ", b , "=>" ," la = " , la , " eq = " , eq , " le = " , le);
    a=4'b1011 ;
    b=4'b1011 ;
    #10
    $display("a = ", a ," b = ", b , "=>" ," la = " , la , " eq = " , eq , " le = " , le);
    a=4'b1001 ;
    b=4'b1011 ;
    #10
    $display("a = ", a ," b = ", b , "=>" ," la = " , la , " eq = " , eq , " le = " , le);
    a=4'b1011 ;
    b=4'b0000 ;
    #10
    $display("a = ", a ," b = ", b , "=>" ," la = " , la , " eq = " , eq , " le = " , le);

    end

endmodule
```

4 Figure

و نتیجه حاصل از شبیه سازی نیز به شکل زیر است:

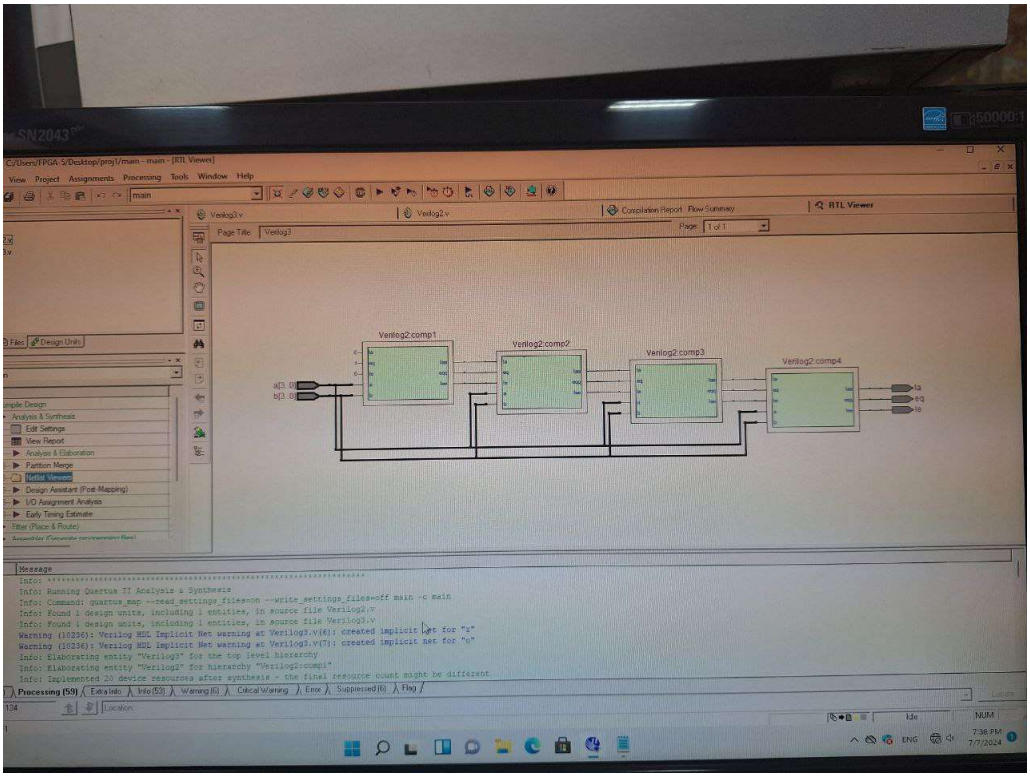
```
# a = 15 b = 11=> la = 1 eq = 0 le = 0
# a = 11 b = 11=> la = 0 eq = 1 le = 0
# a =  9 b = 11=> la = 0 eq = 0 le = 1
# a = 11 b =  0=> la = 1 eq = 0 le = 0
```

3 Figure

حال پس از توضیح کلیات کد (که در پیش گزارش نیز به همین صورت قرار گرفته است) به سراغ مراحلی می‌رویم که در آزمایشگاه سپری کردیم.

ابتدا بورد را روی حالت program قرار دادیم و سپس کد بالا را درون یک پروژه جدید در نرم افزار کوارتوس وارد کردیم. (کد تست بنچ را درون پروژه وارد نکردیم)

سپس پروژه را compile کردیم تا کد وریلاگ ما سنتز شود. در شکل زیر شماتیک مدار معادل با کدی که زدیم را مشاهده می‌کنید که پس از سنتز کردن توسط خود نرم افزار کوارتوس بدست آمد.



6 Figure

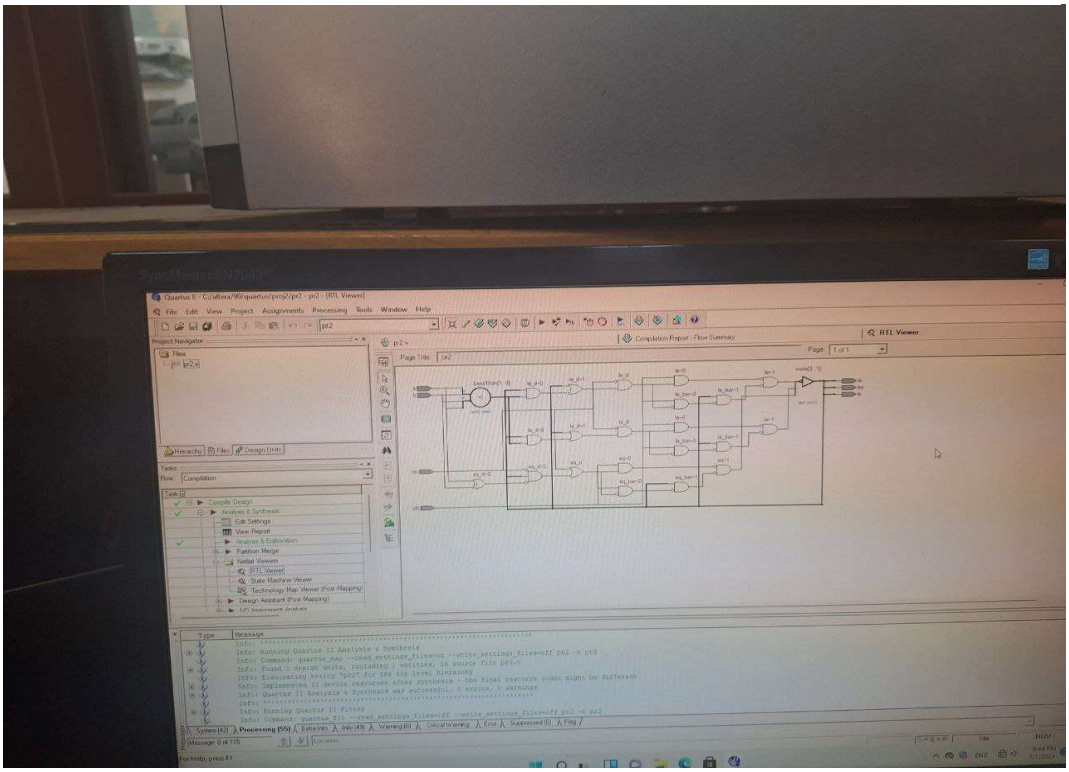
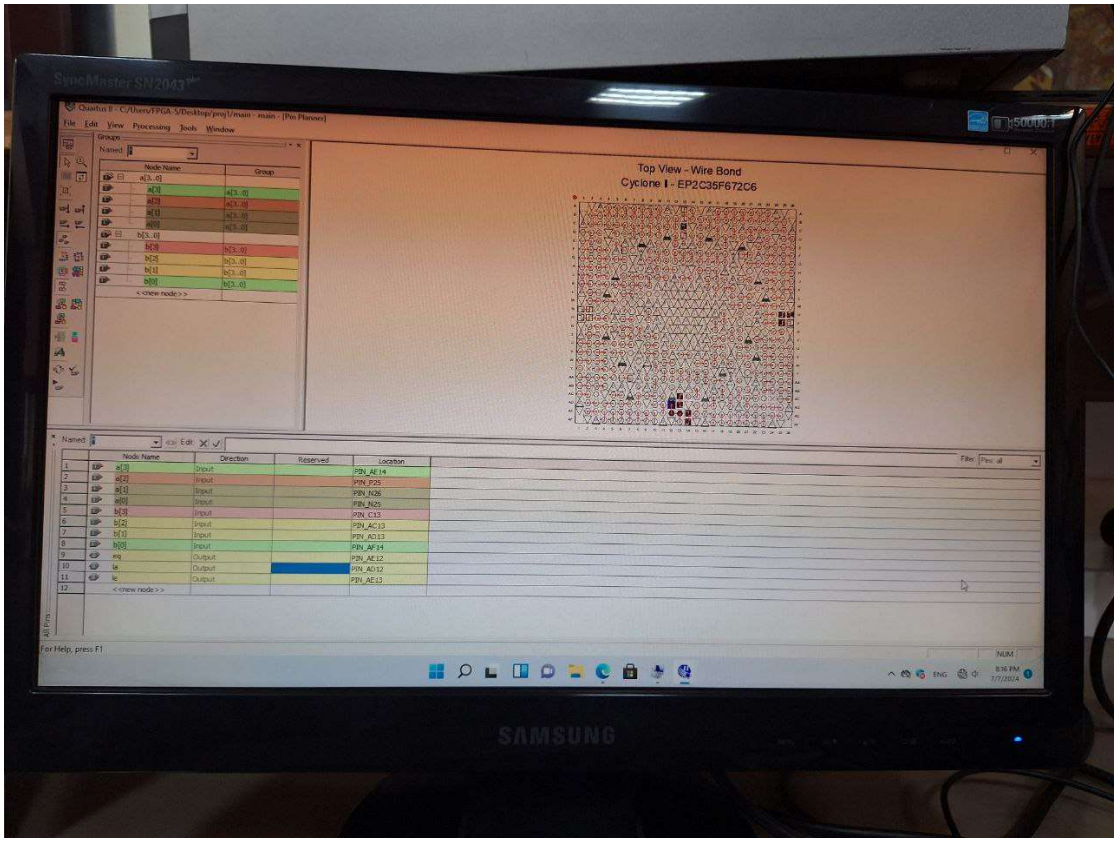


Figure 5

سپس به بخش pin planner (Ctrl+Shift+N) رفتیم و مشخص کردیم که هر کدام از ورودی های مدار به کدام سوویچ متصل باشد و هر کدام از 3 خروجی به کدام LED متصل باشد:

A = pin[3,2,1,0] و B = pin[7,6,5,4] و نشانگر A < B، A = B و A > B به ترتیب LED های 15 تا 17 هستند.

(اینکه هر آدرس پین مربوط به کدام پین است، در یک pdf راهنما که خود کوارتوس در اختیار ما گذاشت مشهود بود.)



7 Figure

پس از آن، به کمک programmer فایل main.scf را انتخاب کرده و پس از تغییر program به run به صورت فیزیکی، برنامه را استارت زدیم.

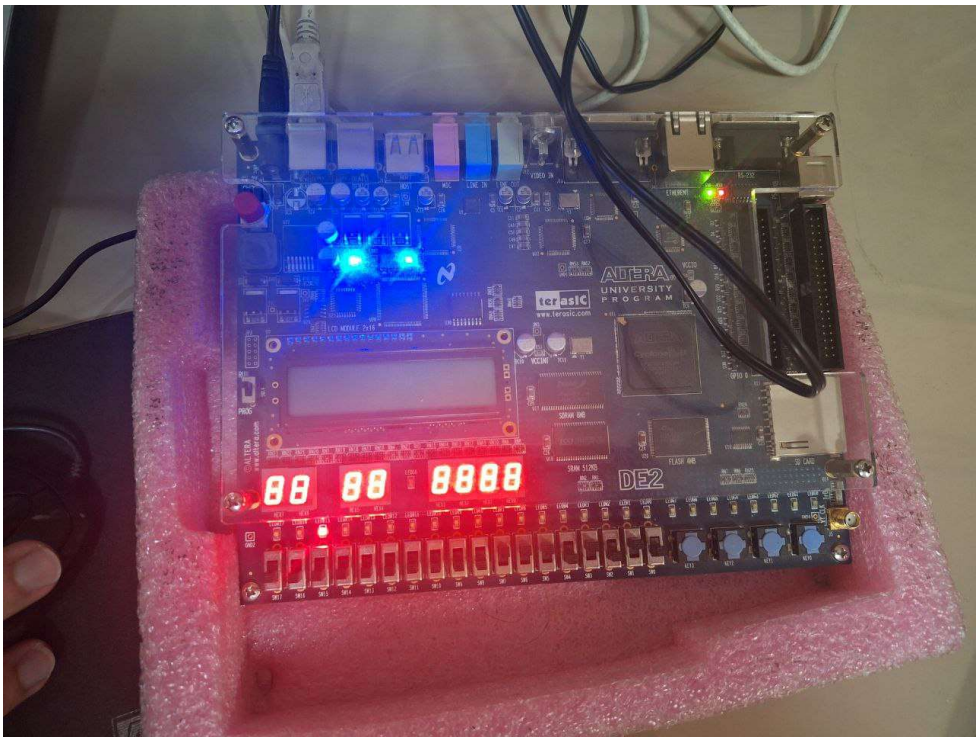
پس از آن، درون بورد، a و b را ست کردیم و هر بار دیدیم که کدام LED روشن می‌شود.

در زیر تصویر سه حالت از آن را ضمیمه می‌کنیم:

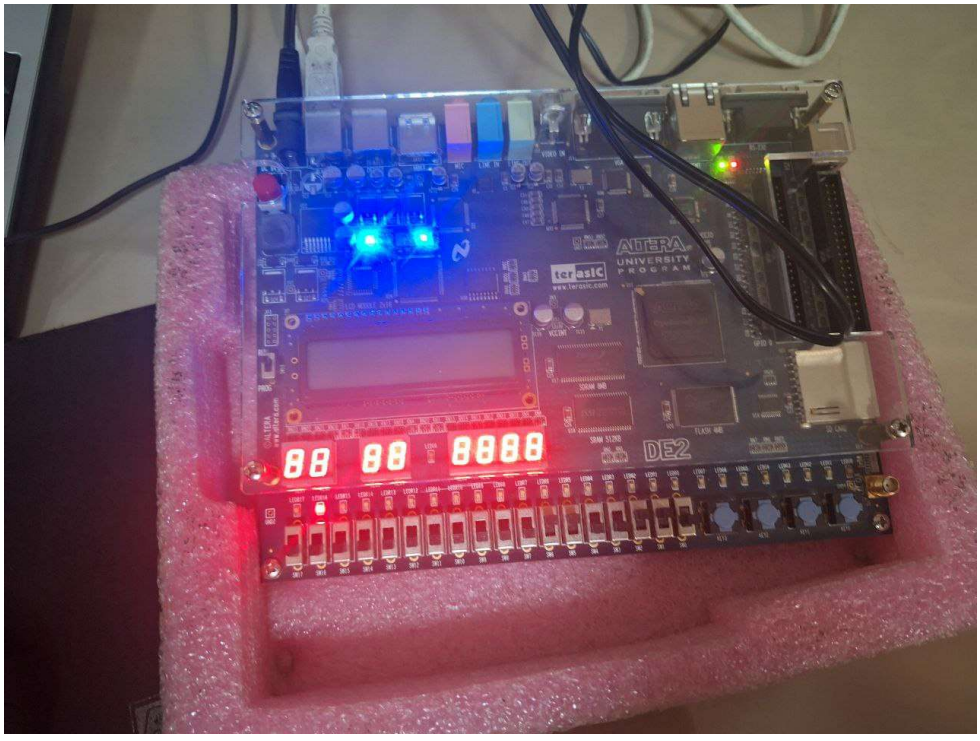
در تصویر 8، A = 1010 و B = 1110 است که نتیجه این شد که LED شماره 15 روشن شد.

در تصویر 9، هم A و هم B برابر با 0110 می‌باشند در نتیجه LED شماره 16 روشن شد.

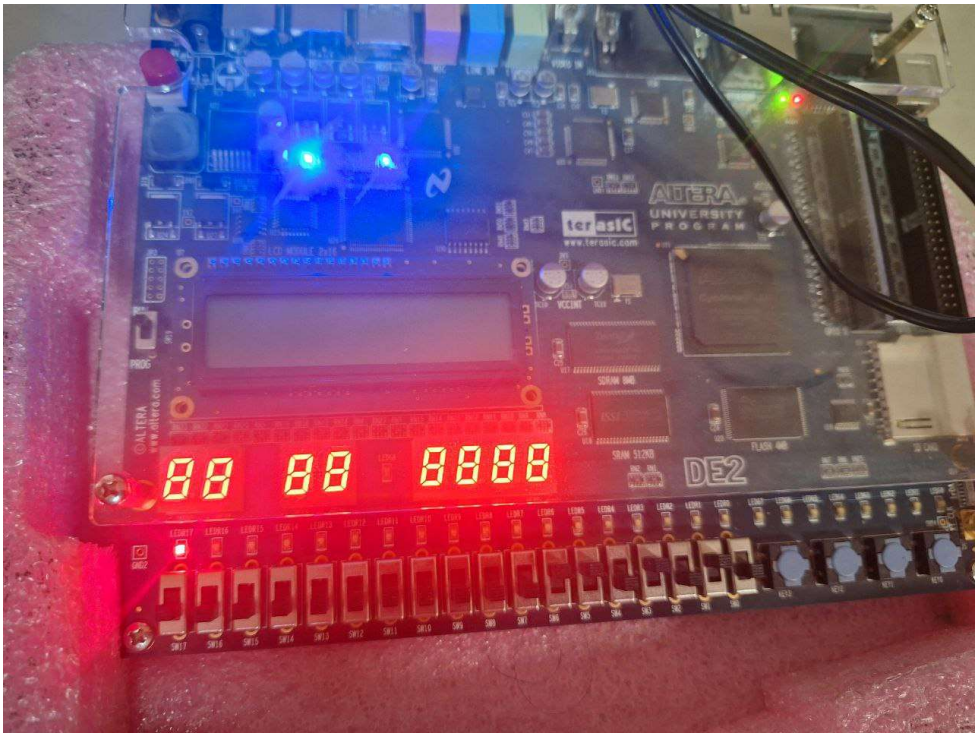
در تصویر 10، A = 1010 و B = 0110 است بنابراین LED شماره 17 روشن شد.



8 Figure



9 Figure

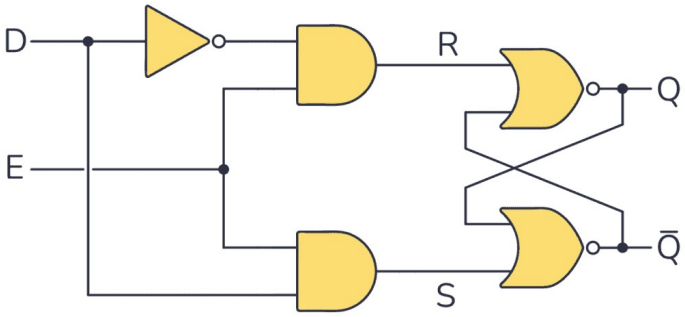


10 Figure

قسمت دوم:

طراحی ما باید شامل یک ماژول باشد و ما باید نتیجه مقایسه تا بیت قبلی را نگه داریم بدین منظور می توانیم از

Latch استفاده کنیم.



E or Clk	D	Q	Q'
0	0	Latch	
0	1	Latch	
1	0	0	1
1	1	1	0

با توجه به نتیجه مقایسه تا بیت قبلی و بیت های ورودی جدید ۳ حالت رخ می دهد:

1) عدد اول بزرگتر از عدد دوم باشد: در اینصورت با ورود بیت های جدید همچنان عدد اول بزرگتر می ماند.
2) عدد اول کوچکتر از عدد دوم باشد: در اینصورت با ورود بیت های جدید همچنان عدد اول کوچکتر می ماند.
3) دو عدد مساوی باشند: در این صورت

اگر بیت های ورودی عدد اول بزرگتر باشد عدد اول بزرگتر خواهد بود

اگر دو بیت برابر باشند عدد نهایی بزرگتر خواهد بود

و در صورتی که بیت عدد دوم بزرگتر باشد عدد اول کوچکتر خواهد بود.

طراحی ما برای این ماژول به صورت زیر خواهد بود:

```
module SERIAL_COMPARATOR(input clk, input rst, input a, input b,
    output la, output eq,output le);


    wire la_d, eq_d, le_d;

    assign la_d = (~rst) & ((eq & (a > b)) | la);
    assign eq_d =  rst | ((eq & (a == b)) & (~rst));
    assign le_d = (~rst) & ((eq & (a < b)) | le);


    //latches
    wire la_bar, eq_bar ,le_bar;

    assign la = ~(la_bar & ~(clk & la_d));
    assign la_bar = ~(la & ~(clk & ~(la_d)));

    assign eq = ~(eq_bar & ~(clk & eq_d));
    assign eq_bar = ~(eq & ~(clk & ~(eq_d)));

    assign le = ~(le_bar & ~(clk & le_d));
    assign le_bar = ~(le & ~(clk & ~(le_d)));



endmodule
```

11 Figure

ماژول آزمون ما نیز به صورت زیر خواهد بود:

```
module SERIAL_COMPARATOR_TB();
    reg rst, clk, a, b;
    wire la, eq, le;
```

```
    SERIAL_COMPARATOR cfomparator(clk, rst, a, b, la, eq, le);

    always  begin
        #5 clk = ~clk;

    end
    initial begin
        clk = 0;
        rst = 1;
        #10
        $display("time: %d, reset: %b a_in: %b, b_in: %b,  la: %b, eq: %b, le:
%b", $time, rst, a, b, la, eq, le);
        rst = 0;
        a = 1;
        b = 0;
        #10
        $display("time: %d, reset: %b a_in: %b, b_in: %b,  la: %b, eq: %b, le:
%b", $time, rst, a, b, la, eq, le);
        a = 0;
        b = 1;
        #10
        $display("time: %d, reset: %b a_in: %b, b_in: %b,  la: %b, eq: %b, le:
%b", $time, rst, a, b, la, eq, le);
        a = 0;
        b = 1;
        #10
        $display("time: %d, reset: %b a_in: %b, b_in: %b,  la: %b, eq: %b, le:
%b", $time, rst, a, b, la, eq, le);
        rst = 1;
        #10
        $display("time: %d, reset: %b a_in: %b, b_in: %b,  la: %b, eq: %b, le:
%b", $time, rst, a, b, la, eq, le);
        rst = 0;
        a = 1;
        b = 1;
        #10
        $display("time: %d, reset: %b a_in: %b, b_in: %b,  la: %b, eq: %b, le:
%b", $time, rst, a, b, la, eq, le);
        a = 0;
        b = 1;
        #10
        $display("time: %d, reset: %b a_in: %b, b_in: %b,  la: %b, eq: %b, le:
%b", $time, rst, a, b, la, eq, le);
        a = 1;
        b = 0;
```

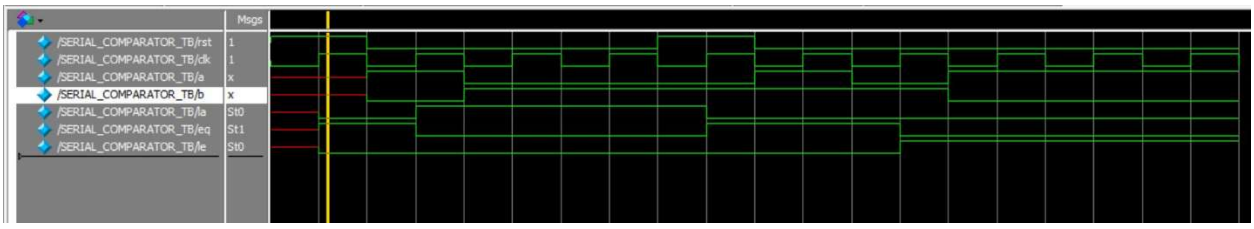
```
        #10
        $display("time: %d, reset: %b a_in: %b, b_in: %b,  la: %b, eq: %b, le:
%b", $time, rst, a, b, la, eq, le);

    end

endmodule
```

و نتایج آن به صورت زیر است که نشان می دهد کد ما به درستی کار می کند:

```
time:                10, reset: 1 a_in: x, b_in: x,  la: 0, eq: 1, le: 0
time:                20, reset: 0 a_in: 1, b_in: 0,  la: 1, eq: 0, le: 0
time:                30, reset: 0 a_in: 0, b_in: 1,  la: 1, eq: 0, le: 0
time:                40, reset: 0 a_in: 0, b_in: 1,  la: 1, eq: 0, le: 0
time:                50, reset: 1 a_in: 0, b_in: 1,  la: 0, eq: 1, le: 0
time:                60, reset: 0 a_in: 1, b_in: 1,  la: 0, eq: 1, le: 0
time:                70, reset: 0 a_in: 0, b_in: 1,  la: 0, eq: 0, le: 1
time:                80, reset: 0 a_in: 1, b_in: 0,  la: 0, eq: 0, le: 1
```

12 Figure

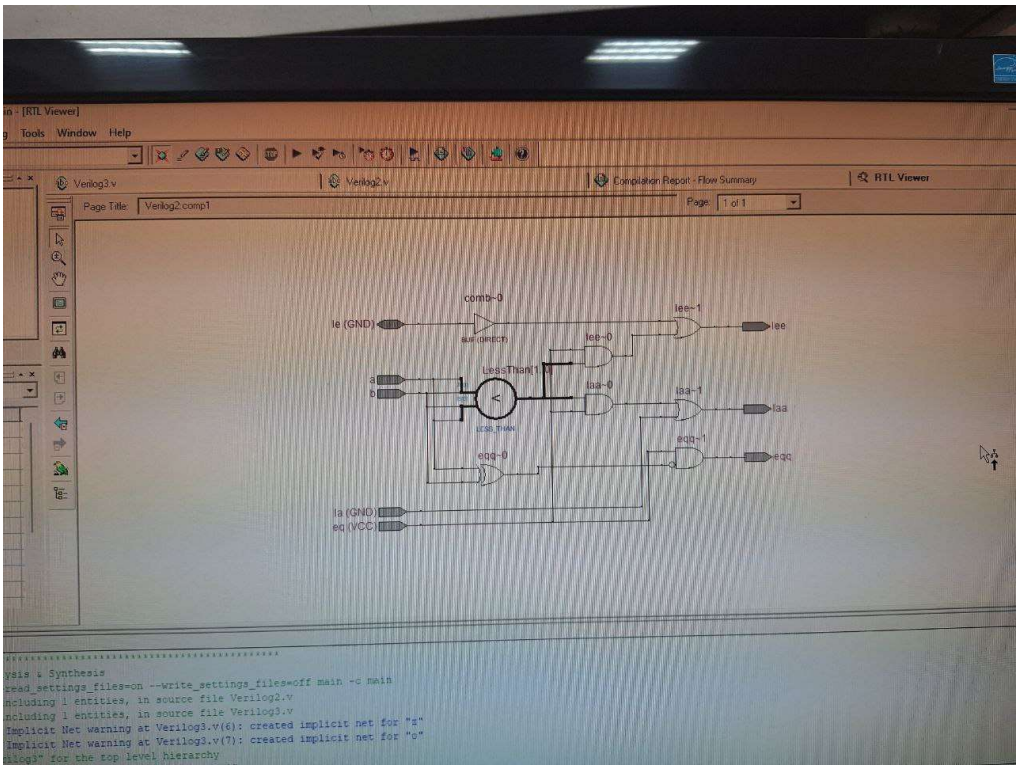
ابتدا دو سریال 100 و 011 را مقایسه می کنیم و سپس بعد از ریست کردن مدار دوسریال 101 و 110را تست می کنیم.



13 Figure

حال در ادامه کار، نتیجه فعالیت در آزمایشگاه را نمایش می‌دهیم:

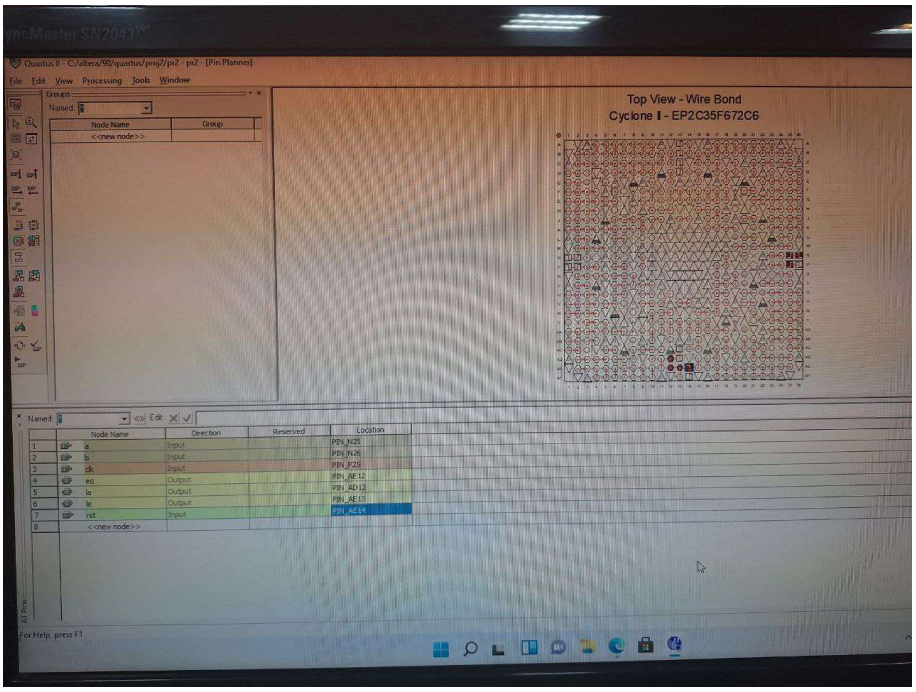
ابتدا بورد را روی حالت program قرار دادیم. سپس تمامی مراحل مانند بخش اول طی شدند و فایل ماژول در یک پروژه جدید کوارتوس بارگزاری شد (فایل تست را بارگزاری نکردیم) سپس پروژه را کامپایل کردیم تا کد سنتز شود و شماتیک زیر حاصل شد:



14 Figure

سپس به بخش pin planner رفتیم و پین ها را مشخص کردیم:

A = pin[0] و B = pin[1] و clk = pin[3] و reset = pin[4] و LED ها نیز مانند بالا برای a < b و a = b و a > b به ترتیب، LED های 15 تا 17 هستند.



15 Figure

حال به بخش programmer رفته و فایل را انتخاب می‌کنیم و سپس بورد را روی حالت run گذاشته و استارت می‌زنیم تا برنامه روی بورد اجرا شود.

سپس با انتخاب a و b در کلاک های متوالی، مقایسه کننده حاصل را به ما می‌دهد. (پس از ریست کردن، تا هر زمان که a = b، LED شماره 16 روشن است. سپس اولین جایی که a > b شود، LED شماره 17 روشن و بقیه خاموش می‌شوند و دیگر پس از آن تا قبل از ریست کردن، تغییری نمی‌کند. همچنین اگر وقتی LED شماره 16 روشن است، a < b شود، LED شماره 15 روشن می‌شود و مابقی خاموش می‌شوند و دیگر تا زمان ریست کردن، تغییری نمی‌کنند.)

از این مراحل تصویری برای بارگزاری نداریم زیرا مشخص کردن توالی در تصاویر، بسیار پیچیده بود ولی در آزمایشگاه صحت مدار مورد تایید TA قرار گرفت.