گزارش آزمایش طراحی سیستم های دیجیتال

آزمایش دوم:

طراحی مدار ترتیبی با امکانات شماتیک

نگارش :

سید ابوالفضل رحیمی ۹۷۱۰۵۹۴۱

امیرحسین باقری ۹۸۱۰۵۶۲۱

استاد :

دکتر علیرضا اجلالی

دستیار آموزشی :

آقای روزبه سیادت زاده

Contents

مقدمه

هدف از این آزمایش، طراحی مقایسه کننده به کمک توصیف جریان داده در وریلاگ است. می دانیم در توصیف جریان داده ی یک سیستم دیجیتال، سیستم از طریق معرفی روابط منطقی میان سیگنال‌ها(اتصالات) با ارائه ی عبارت‌های جبری توصیف می‌شود. در این توصیف، به کمک کلیدواژه assign مشخص می کنیم که خروجی بر اساس ورودی ها به چه شکل است. پس ازاین که یک واحد مقایسه‌کننده تک بیتی طراحی شد، به کمک قابلیت ماژول‌بندی در وریلاگ یک مدار ترکیبی ۴ بیتی در بخش یک ساخته خواهد شد. در قسمت ب اما، باتوجه به نبود ساختار سلسه‌مراتبی، در یک ماژول این کار صورت گرفت.

۱. مقایسه‌کننده تک و چهار بیتی

در فایل onebit_comparator.v و همچنین fourbit_comparator.v این پیاده‌سازی‌ها صورت گرفته است که در ادامه به بررسی و توضیح آن‌ها می‌پردازیم.

ابتدا خوب است به طراحی ساده مدار مقایسه کننده تک‌بیتی بپردازیم. این مدار به سادگی سه خروجی برابر بودن، بزرگ‌تر بودن و کوچک‌تر بودن را با استفاده از گیت‌های منطقی پیاده‌سازی کرده است. کد وریلاگ آن به همراه شمای RTL در کوارتوس را در زیر می‌توانید ببینید.

```
module onebit_comparator(input a, input b, output equal, output greater, output less);
    assign equal = (a == b);
    assign greater = (a > b);
    assign less = (a < b);
endmodule
```

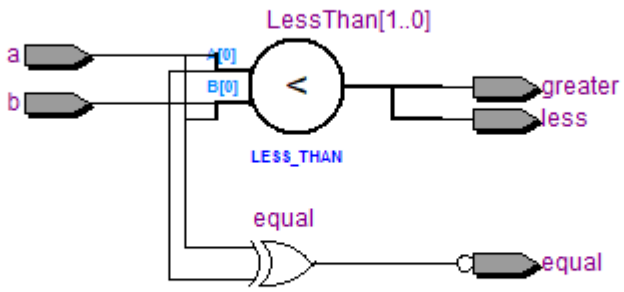

Figure 1 شمای RTL مقایسه‌گر تک‌بیتی در کوارتوس

برای تست این مدار ساده، از تست بنچ زیر استفاده شد که موج آن را هم در زیر می‌توانید ببینید:

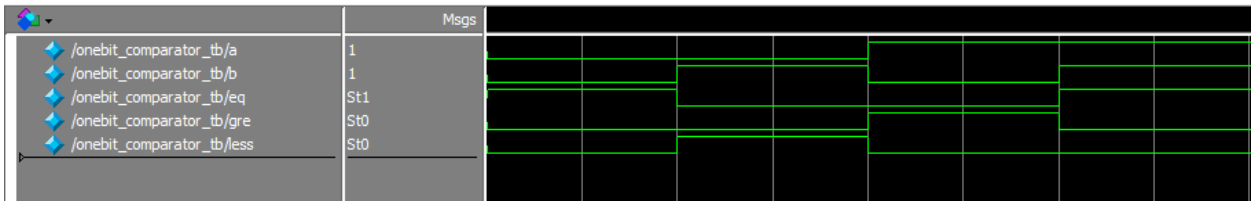

Figure 2 نتیجه تست‌بنچ برای مقایسه‌گر تک‌بیتی

```
module onebit_comparator_tb;
	reg a, b;
	wire eq, gre, less;
	onebit_comparator comparator(a, b, eq, gre, less);
	 initial
	begin
	  a <= 0;
	  b <= 0;
	  #10
	  a <= 0;
	  b <= 1;
	  #10
	  a <= 1;
```

```
        b <= 0;
        #10
        a <= 1;
        b <= 1;
      end
endmodule
```

در ادامه و با توجه به در دست داشتن یک مقایسه‌گر تک‌بیتی یک مقایسه‌گر چهاربیتی را با استفاده از چهار مقایسه‌گر تک‌بیتی ساختیم. مقایسه‌گری که پیاده‌سازی آن با وریلاگ و شبیه‌سازی آن با کوارتوس در ادامه آمده است:

```
module fourbit_comparator(input [3:0] a,
        input [3:0] b,
        output equal,
        output greater,
        output less);

        wire [3:0] g;
        wire [3:0] e;
        wire [3:0] l;

        onebit_comparator onebit_comparator0(a[0], b[0], e[0], g[0], l[0]);
        onebit_comparator onebit_comparator1(a[1], b[1], e[1], g[1], l[1]);
        onebit_comparator onebit_comparator2(a[2], b[2], e[2], g[2], l[2]);
        onebit_comparator onebit_comparator3(a[3], b[3], e[3], g[3], l[3]);

        assign equal = & e;
        assign greater = g[3] | (g[2] & e[3]) | (g[1] & e[3] & e[2]) | (g[0] & e[3] & e[2] & e[1]);
        assign less = ~(equal | greater);
endmodule
```

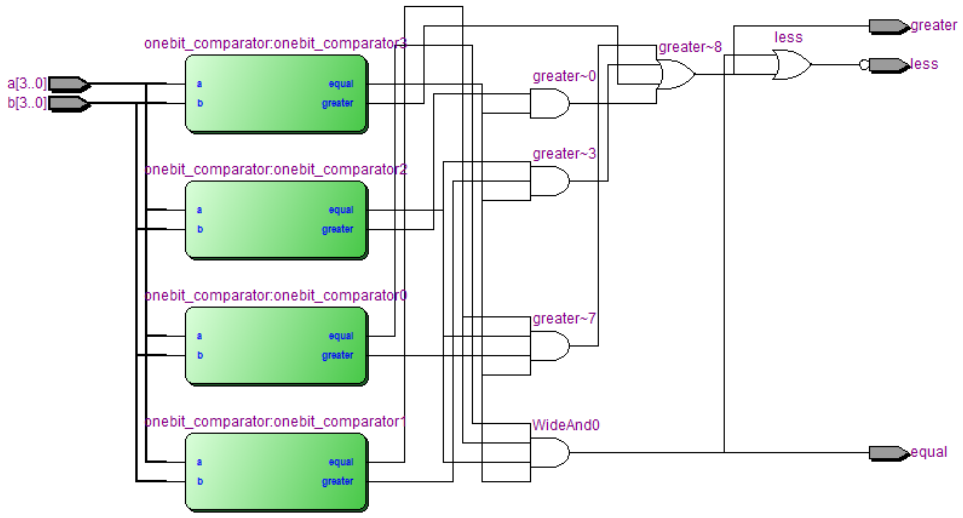

Figure 3 شمای RTL مقایسه‌گر چهاربیتی ما با استفاده از کوارتوس

همچنین برای تست کردن این مدار هم از تست‌بنچ زیر استفاده شد که نتایج مربوط به آن‌را می‌توانید در موج زیر ببینید:

```
module fourbit_comparator_tb;
  reg [3:0] a;
  reg [3:0] b;
  wire E, G, L;

  fourbit_comparator fcmp(a, b, E, G, L);

  always
  begin
    a <= {$random} % 16;
    b <= {$random} % 16;
    #10;
  end
endmodule
```

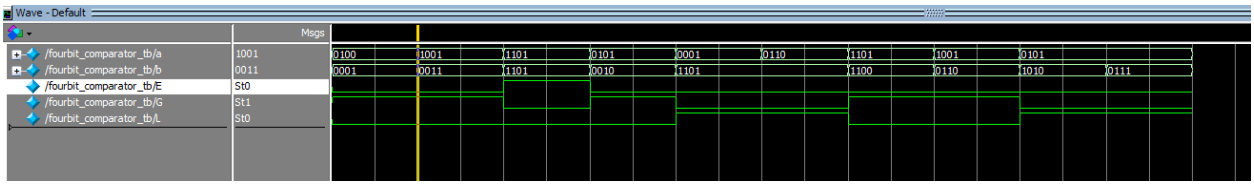

Figure 4نتایج تست‌بنچ مقایسه‌گر چهاربیتی

همانطور که در موج بالا دیده‌می‌شود با تغییر بیت‌ها هر سه حالت رخ داده است.

ابتدا دو بزرگ‌تر را می‌بینیم با توجه به اعداد ۹-۱و ۹-۳. پس از آن تساوی ۱۳ و به همین‌ترتیب تا انتها.

یک توضیح کوتاه هم درباره تست‌بنچ خوب است داده شود. این تست‌بنچ درواقع هربار یک عدد رندم برای هر دو انتخاب می‌کند و سپس آن‌ها را مقایسه می‌کند. این فرآیند هم تا انتهای شبیه‌سازی ادامه پیدا خواهد کرد. تنها نکته‌ای که ممکن است پیش بیآید این است که تعداد تساوی‌ها احتمال کمی دارند که به همین دلیل باید برای خوب عمل کردن این تست‌بنچ میزان زمان از کلاک صوری از اردر ۱۶ باشد. (احتمال تساوی دو عدد مستقل!)

۲. مقایسه‌کننده سریال

این مقایسه‌کننده سریال در بخش دوم همانطور که خواسته شده با کلاک و یک اکتیو های ریست طراحی شده است. کد وریلاگ آن را در ادامه می‌توانید در اینجا هم ببینید. البته که در فولد بخش دوم نیز این کد تولید شده است. همچنین خروجی سنتز کوارتوس هم برای درک بهتر اینجا آورده شده است:

در خطوط اول، ورودی ها و خروجی ها اعلان شده اند. x_inوy_in بیت هایی هستند که هر مرحله وارد می شوند تا مقایسه شوند وclk کلاک مدار است و reset هم در صورت سوال گفته شده است. Gو L وE هم خروجی های مقایسه کننده هستند . درباره ی wire های تعریف شده در بخش مربوطه توضیح داده خواهد شد. با L_dوE_d ورودی های فلیپ فلاپ ها را مشخص می کنیم(که پارامترهایE و L در آن ها مربوط به مرحله ی قبل اند و خود ورودی های فلیپ فلاپ ها برای این تنظیم شده اند که بتوانند به عنوان یک رجیستر در نبود امکان استفاده از دستوراتی غیر از دستورات از نوع توصیف جریان داده عمل کنند) E یک reset با or هم دارد؛ یعنی اگر reset یک شد یعنی می خواهیم عدد وارد کنیم و تا قبل از این، تنها اعداد، صفرهای اختیاری پشت اعداد هستند که با هم برابرند و در نتیجه E باید یک شود .ضمنا L با نقیض and reset شده است تا اگر reset یک بود، کل عبارت L برابر صفر شود(که با توجه به توضیحات فوق، چرایی آن مشخص است.) سپس، عبارت های بولی یک فلیپ فلاپ D حساس به لبه ی بالارونده ی کلاک نوشته شده است(عبارت های مرتبط، قبلا به صورتwireتعریف شده اند)به وضوح از شکل زیر، مشخص است که E_barوE خروجی های فلیپ فلاپ هستند .در کد، اتصالات مربوط بهNANDهای طبقه ی اول، با 1Ew تا4 Ew مشخص شده اند .حاصلnand های طبقه ی دوم(که خروجی های فلیپ فلاپ ما هستند)هم در اتصالات assign، E_barوE شده اند. سپس یک فلیپ فلاپ دیگر هم برای نگه داریL در نظر گرفته شده است .در نهایت، خروجی هایE و Lپس از هر مرحله درست آماده می شوند و در نتیجه خروجی نهایی مدار نیز خواهند بود . مدارهای هر دو بخش، توصیف جریان داده هستند، زیرا از ساختارهایی به جز assign استفاده نکرده اند .در نهایت، وقتی G یک است(یعنی بزرگ تر است)که نه L یک باشد نه E. در نتیجه OR آن ها باید صفر باشد و به این ترتیب، خروجی Gنیز ساخته می شود. در نهایت نیز پایان ماژول اعلان گشته است

```
module serial_comparator(input x_in, input y_in, input clk, input reset,
                         output E, output L, output G);

    wire E_d, Ew1, Ew2, Ew3, Ew4, E_bar;
    wire L_d, Lw1, Lw2, Lw3, Lw4, L_bar;

    // 1 bit comparator (combinational)
    assign E_d =  reset || ((x_in ~^ y_in) & (E));
    assign L_d = !reset &&((E & y_in & (~x_in)) | L);

    // D-flip-flop for E (input(d): E_d, output(Q) : E)
    assign Ew1 = !(Ew4 && Ew2);
    assign Ew2 = !(Ew1 && clk);
    assign Ew3 = !(Ew2 && clk && Ew4);
    assign Ew4 = !(Ew3 && E_d);

    assign E   = !(Ew2 && E_bar);
    assign E_bar = !(E && Ew3);

    // D-flip-flop for L (input(d): E_d, output(Q) : L)
    assign Lw1 = !(Lw4 && Lw2);
    assign Lw2 = !(Lw1 && clk);
    assign Lw3 = !(Lw2 && clk && Lw4);
    assign Lw4 = !(Lw3 && L_d);

    assign L   = !(Lw2 && L_bar);
    assign L_bar = !(L && Lw3);

    // greater is when not "equal or less"
    assign G = ~(E | L);

endmodule
```

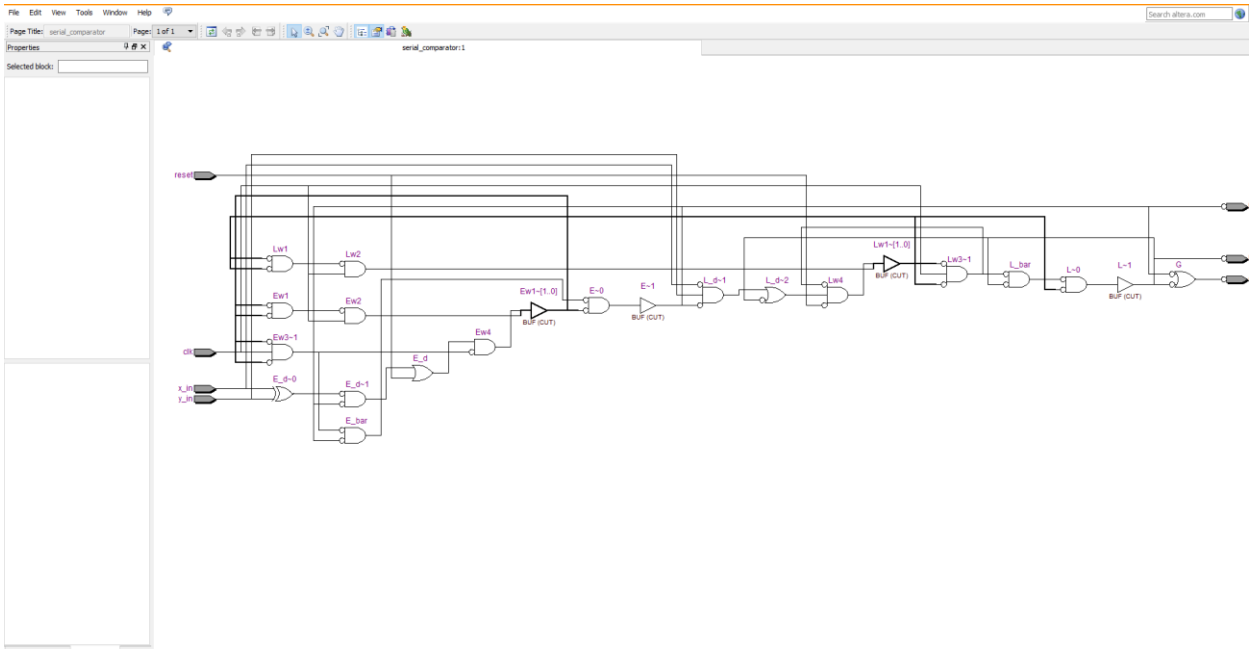

Figure 5 شمای RTL مقایسه‌گر سریال پیاده‌سازی شده در کوارتوس

این مدار و کد وریلاگ هم با استفاده از تست‌بنچ زیر که حالات اصلی را چک می‌کند تست شد. البته که غیر از آوردن سه حالت اصلی موج، سعی کردیم برای درک ساده‌تر خروجی ها را نیز در کنسول چاپ کنیم که در ادامه این‌ها آمده‌اند:

```
`timescale 1ns/1ns


module serial_comparator_tb;

    parameter num_tests = 30;
    integer i1, i2, i3, x_num, y_num;

    reg x_in, y_in, clk, reset;
    wire E, L, G;

    serial_comparator comparator(
        .x_in(x_in),
        .y_in(y_in),
        .clk(clk),
        .reset(reset),
```

```
        .E(E),
        .L(L),
        .G(G)
    );



    //  clock generator
    initial
    begin
        clk = 0;

        for (i1 = 0; i1 < num_tests * (5+1) * 2; i1 = i1+1) begin
            #100;
            clk = ~clk;
        end
    end

    // input generator
    initial
    begin

$srandom(100);

        for (i2 = 0; i2 < num_tests; i2 = i2+1) begin

            // reset for a clock cycle
            reset = 1;
            #200
            reset = 0;

            // generate two 5-bit numbers
            x_num = 0;
            y_num = 0;

            for (i3 = 0; i3 < 5; i3 = i3+1) begin
```

```
                #50;
                x_in = $urandom_range(1,0);
                y_in = $urandom_range(1,0);
                x_num = (x_num << 1) + x_in;
                y_num = (y_num << 1) + y_in;
                #150;
            end

            $display("x = %5b(%2d) , y = %5b(%2d) --> E = %b , L = %b , G = %b",
                x_num, x_num, y_num, y_num, E, L, G);


        end

    end

endmodule
```

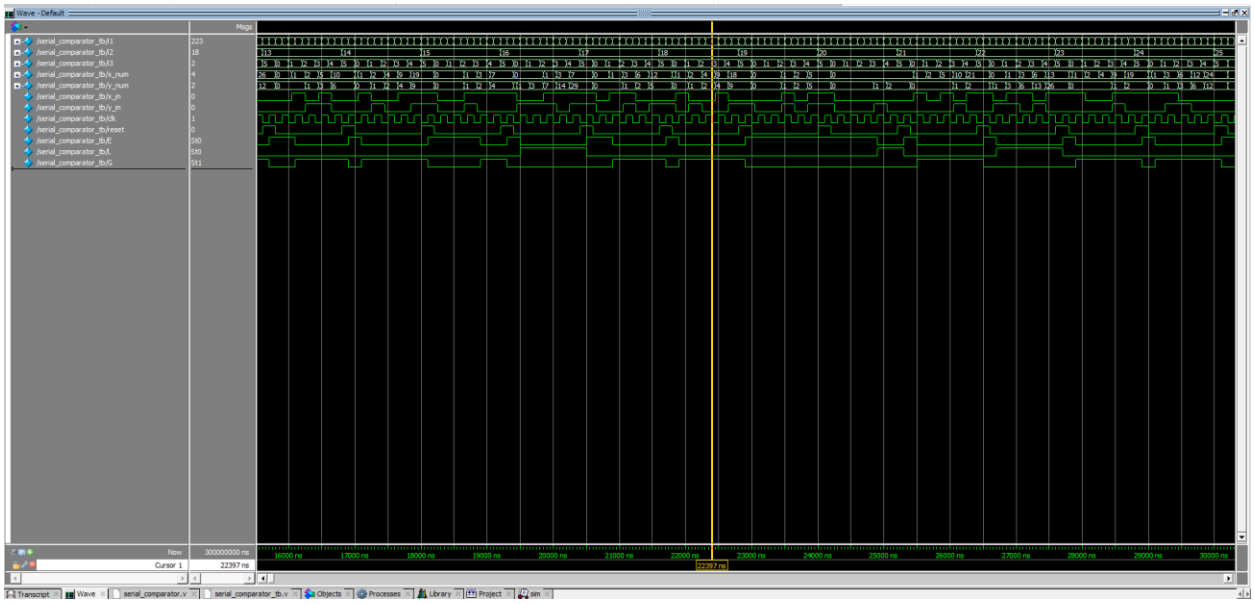

Figure 6بزرگ‌تر بودن در یک مقایسه‌گر سریال

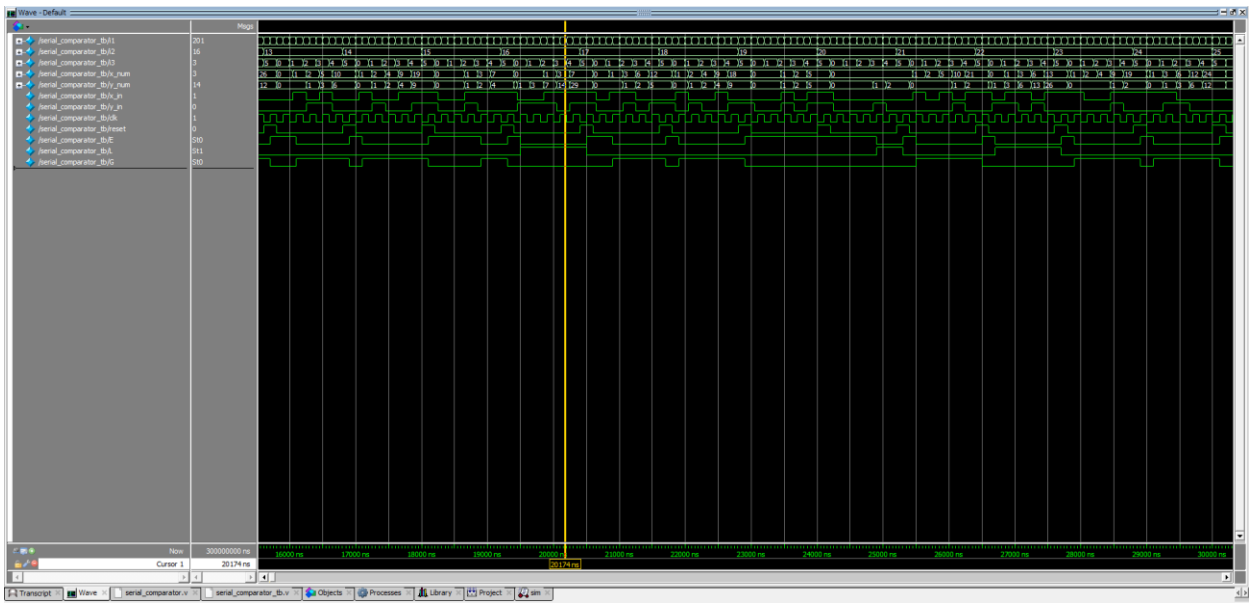

Figure 7 برابری در مقایسه‌گر سریال

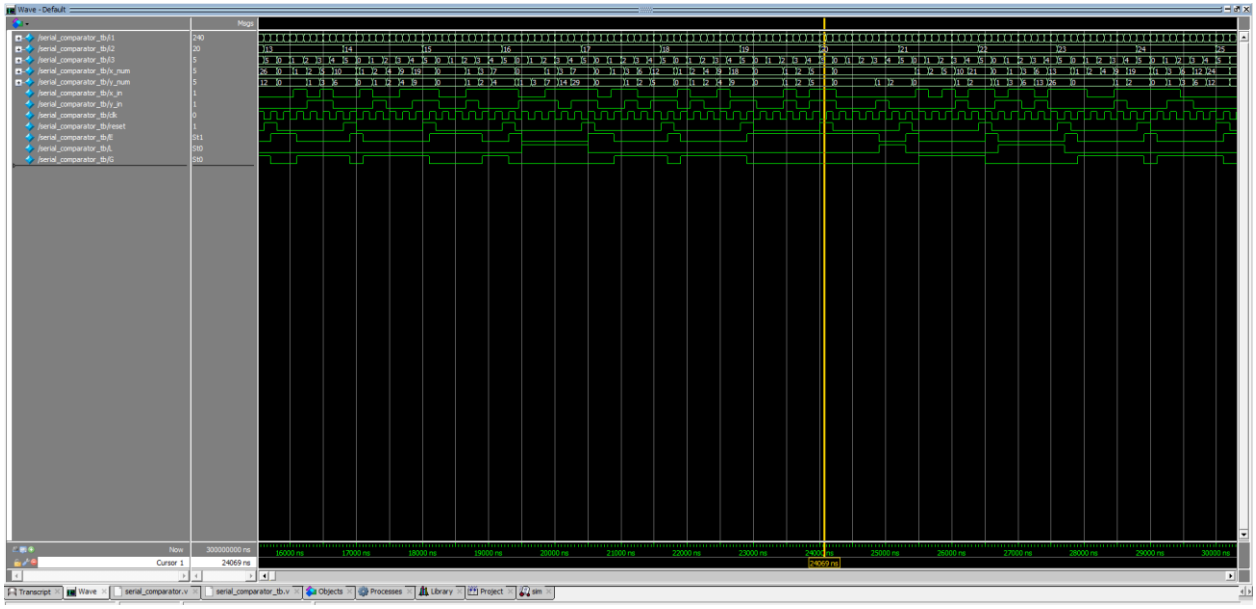

Figure 8 کوچک‌تر بودن در مقایسه‌گر سریال ما

```
VSIM 35> run
# x = 11000(24) , y = 11000(24) --> E = 1 , L = 0 , G = 0
# x = 01010(10) , y = 00111( 7) --> E = 0 , L = 0 , G = 1
# x = 11001(25) , y = 01110(14) --> E = 0 , L = 0 , G = 1
# x = 01101(13) , y = 01100(12) --> E = 0 , L = 0 , G = 1
# x = 11111(31) , y = 01101(13) --> E = 0 , L = 0 , G = 1
# x = 01011(11) , y = 01000( 8) --> E = 0 , L = 0 , G = 1
# x = 01100(12) , y = 10011(19) --> E = 0 , L = 1 , G = 0
# x = 10001(17) , y = 01110(14) --> E = 0 , L = 0 , G = 1
# x = 11110(30) , y = 00000( 0) --> E = 0 , L = 0 , G = 1
# x = 00001( 1) , y = 10011(19) --> E = 0 , L = 1 , G = 0
# x = 00010( 2) , y = 11111(31) --> E = 0 , L = 1 , G = 0
# x = 01010(10) , y = 00111( 7) --> E = 0 , L = 0 , G = 1
# x = 11010(26) , y = 01100(12) --> E = 0 , L = 0 , G = 1
# x = 01010(10) , y = 00110( 6) --> E = 0 , L = 0 , G = 1
# x = 10011(19) , y = 01001( 9) --> E = 0 , L = 0 , G = 1
# x = 00111( 7) , y = 00100( 4) --> E = 0 , L = 0 , G = 1
# x = 00111( 7) , y = 11101(29) --> E = 0 , L = 1 , G = 0
# x = 01100(12) , y = 00101( 5) --> E = 0 , L = 0 , G = 1
# x = 10010(18) , y = 01001( 9) --> E = 0 , L = 0 , G = 1
# x = 00101( 5) , y = 00101( 5) --> E = 1 , L = 0 , G = 0
# x = 00000( 0) , y = 00010( 2) --> E = 0 , L = 1 , G = 0
# x = 10101(21) , y = 00010( 2) --> E = 0 , L = 0 , G = 1
# x = 01101(13) , y = 11010(26) --> E = 0 , L = 1 , G = 0
# x = 10011(19) , y = 00010( 2) --> E = 0 , L = 0 , G = 1
# x = 11000(24) , y = 01100(12) --> E = 0 , L = 0 , G = 1
# x = 01000( 8) , y = 01011(11) --> E = 0 , L = 1 , G = 0
# x = 11010(26) , y = 11001(25) --> E = 0 , L = 0 , G = 1
# x = 10011(19) , y = 00001( 1) --> E = 0 , L = 0 , G = 1
# x = 11000(24) , y = 00001( 1) --> E = 0 , L = 0 , G = 1
# x = 01011(11) , y = 01011(11) --> E = 1 , L = 0 , G = 0
```

Figure 9 نتایج تست‌بنچ که همگی درست هستند.

نتایج کامپایل را هم در فایل مربوطه در فولدر screenshots می‌توانید ببینید.