گزارش آزمایش سوم طراحی سیستم‌های دیجیتال

گروه شش

اعضا:

احمد سلیمی

همیلا میلی

درنا دهقانی

شرح آزمایش

در این آزمایش به ساخت مقایسه‌گر میپردازیم.

در بخش اول آن به کمک مقایسه‌گرهای تک بیتی، یک مقایسه‌گر چهار بیتی با مدار ترکیبی می‌سازیم. پس حاصل طراحی، سلسله‌مراتبیست.

در بخش دوم آن باید توسط فقط یک module، یک مقایسه‌گر سریالی بسازیم که ورودی‌ها را بیت به بیت دریافت میکند و تا بیت دریافت شده تا کلاک فعلی، حاصل مقایسه آنها را اعلام میکند. این مقایسه‌گر یک ورودی reset برای بازگشت به حالت ابتدایی نیز دارد.

ورودی‌ها:

- a: عدد اول که در بخش اول ۴ بیتی و در بخش دوم یک بیتی‌ست.
- b: عدد دوم که در بخش اول ۴ بیتی و در بخش دوم یک بیتی‌ست.
- reset: فقط برای بخش دوم
- clk: فقط برای بخش دوم

خروجی‌ها:

- g: اگر عدد اول بزرگتر از عدد دوم باشد برابر با ۱ می‌شود.
- l: اگر عدد اول کوچکتر از عدد دوم باشد برابر با ۱ می‌شود.
- e: اگر عدد اول برابر با عدد دوم باشد برابر با ۱ می‌شود.

بخش اول: مقایسه‌گر چهار بیتی

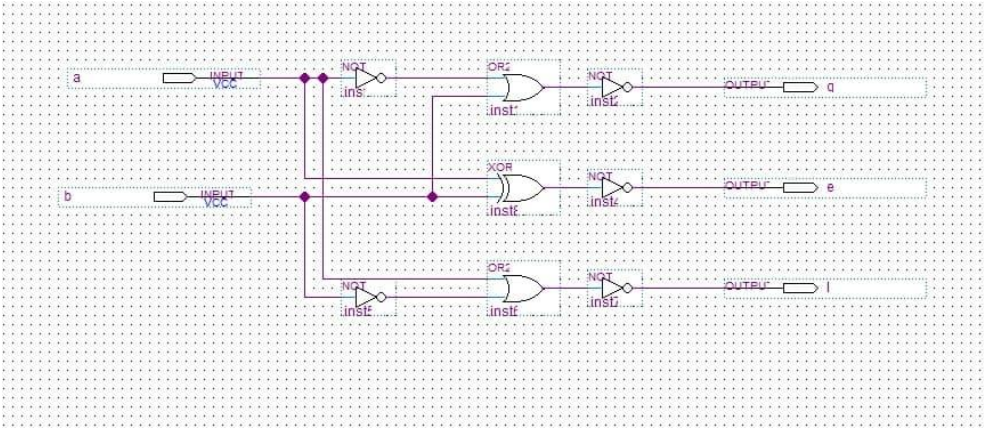
- Comparator: این ماژول ابتدایی‌ترین لایه‌ی این مقایسه‌گر است که خروجی آن حاصل مقایسه دو بیت ورودی a و b می‌باشد. با توجه به جدول زیر و به کمک جدول کارنو، می‌توان مقادیر سه بیت خروجی g، l و e را محاسبه کرد.

a	b	g	l	e
0	0	0	۰	۱
0	1	0	۱	۰
1	0	1	۰	۰
1	1	0	۰	۱

در نتیجه، در کد Verilog این مقایسه‌گر داریم:

```
module comparator (
    input   a,
    input   b,
    output  g,
    output  e,
    output  l
);

assign g = ~((~a) | b);
assign e = ~(a ^ b);
assign l = ~((~b) | a);

endmodule
```

نمایش شماتیک این بخش از مدار به شکل زیر است:



- Cascadable_comparator: در این ماژول به کمک سیگنال‌های ورودی g_in، e_in و l_in که بیانگر این هستند که تا بیت قبل، عدد اول از عدد دوم بزرگتر بوده یا خیر، و دو سیگنال ورودی a و b که دو بیت فعلی هستند، تعیین می‌کنیم که تا بیت فعلی کدام عدد بزرگتر است.

ابتدا یک instance از Comparator می‌سازیم که دو بیت a و b را مقایسه می‌کند.

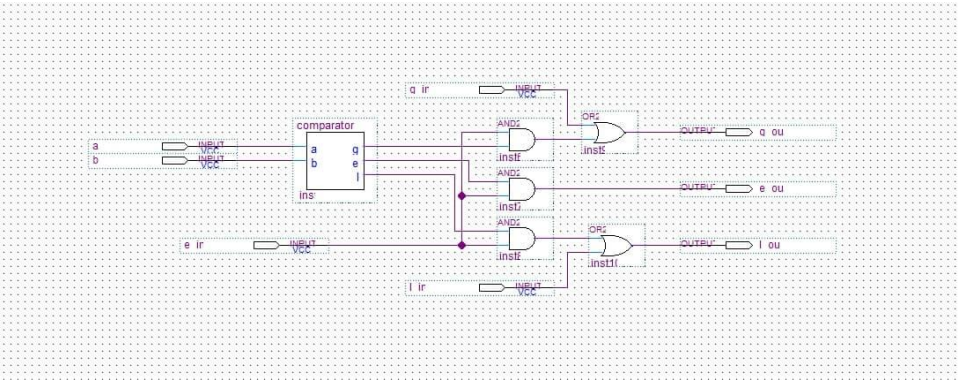
برای دو خروجی l_out و g_out، ابتدا بررسی میکنیم که سیگنال‌های l_in و g_in یک بودند یا خیر، که اگر یک باشند، حتما l_out و g_out نیز یک خواهند بود. در غیر اینصورت، اگر تا کنون دو عدد برابر باشند (یعنی e_in=1) و اکنون a از b بزرگتر باشد، g_out و اگر تا کنون دو عدد برابر باشند (یعنی e_in=1) و اکنون a از b کوچکتر باشد، l_out برابر با یک می‌شود.

برای خروجی e_out باید حتما دو عدد تا بیت قبل برابر باشند (e_in=1) و اکنون نیز a با b برابر باشد.

در نتیجه در کد Verilog این module داریم:

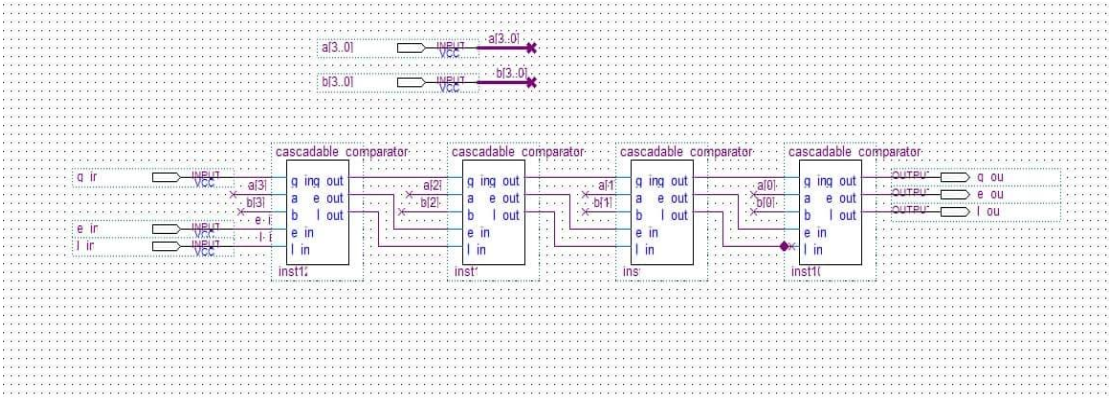
```
module cascadable_comparator (
    input   a,
    input   b,
    input   g_in,
    input   e_in,
    input   l_in,
    output  g_out,
    output  e_out,
    output  l_out
);

comparator COM(
    .a(a),
    .b(b),
    .g(g),
    .e(e),
    .l(l)
);

assign g_out = g_in | (e_in & g);
assign e_out = e_in & e;
assign l_out = l_in | (e_in & l);

endmodule
```

نمایش شماتیک این بخش از مدار به شکل زیر است:



- Four_bit_comparator: در این بخش از مدار، دو ورودی چهار بیتی a و b به این ماژول که ساختار ترکیبی دارد داده می‌شوند و حاصل مقایسه آنها در خروجی نمایش داده می‌شود. این ماژول متشکل از چهار ماژول Cascadable_comparator است که خروجی هر ماژول به عنوان ورودی به ماژول بعدی داده می‌شود.

ساختار شماتیک این مدار به شکل زیر است:



و درکد Verilog این module داریم:

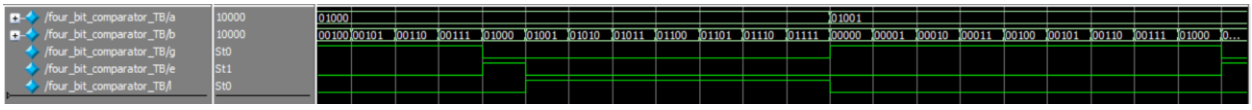
```
module four_bit_comparator (
    input   [3:0]   a,
    input   [3:0]   b,
    input           g_in,
    input           e_in,
    input           l_in,
    output          g_out,
    output          e_out,
    output          l_out
);

cascadable_comparator COM3(
    .a(a[3]),
    .b(b[3]),
    .g_in(g_in),
    .e_in(e_in),
    .l_in(l_in),
    .g_out(g3),
    .e_out(e3),
    .l_out(l3)
);

cascadable_comparator COM2(
    .a(a[2]),
    .b(b[2]),
    .g_in(g3),
    .e_in(e3),
    .l_in(l3),
    .g_out(g2),
    .e_out(e2),
    .l_out(l2)
);

cascadable_comparator COM1(
    .a(a[1]),
    .b(b[1]),
    .g_in(g2),
    .e_in(e2),
    .l_in(l2),
    .g_out(g1),
    .e_out(e1),
    .l_out(l1)
);
```

```
cascadable_comparator COM0(
    .a(a[0]),
    .b(b[0]),
    .g_in(g1),
    .e_in(e1),
    .l_in(l1),
    .g_out(g_out),
    .e_out(e_out),
    .l_out(l_out)
);

endmodule
```

- four_bit_comparator_TB: از این ماژول برای مقداردهی به a و b و امتحان کارکرد ماژول قبلی استفاده می‌کنیم و تمامی مقادیر ۴ بیتی ممکن برای a و b را بررسی می‌کنیم.

```
module four_bit_comparator_TB ();

reg [4:0]   a, b;
wire        g, e, l;

four_bit_comparator COM(
    .a(a[3:0]),
    .b(b[3:0]),
    .g_in(1'b0),
    .e_in(1'b1),
    .l_in(1'b0),
    .g_out(g),
    .e_out(e),
    .l_out(l)
);

initial begin
    $monitor("%d, %d, g = %d, e = %d, l = %d", a, b, g, e, l);
    for (a = 0; a < 16; a = a + 1) begin
        for (b = 0; b < 16; b = b + 1) begin
            #10;
        end
    end
    $stop;
end

endmodule
```

- Waveform: برای نمونه، بخشی از waveform را بررسی می‌کنیم. اگر مقدار a برابر ۸ و مقدار b نیز برابر ۸ باشد، فقط سیگنال e برابر با ۱ می‌شود و اگر مقدار b برابر با ۹ شود، فقط سیگنال l برابر با ۱ می‌شود. اگر مقدار a برابر با ۹ و مقدار b بین ۰ تا ۸ باشد، سیگنال g برابر با ۱ می‌شود.

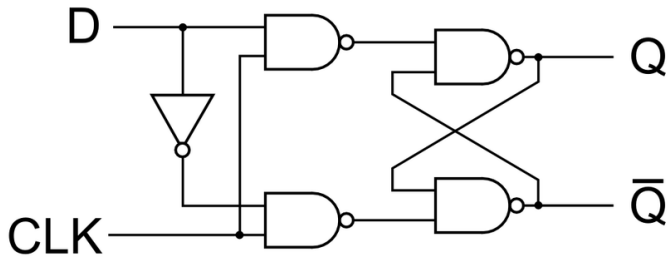


- سنتز: مدار نهایی توسط کوارتوس سنتز شد و summary آن به شرح زیر است:

Flow Summary	
Flow Status	Successful - Tue Apr 06 11:26:08 2021
Quartus II 64-Bit Version	13.0.1 Build 232 06/12/2013 SP 1 SJ Web Edition
Revision Name	Tootooleh
Top-level Entity Name	four_bit_comparator
Family	Cyclone IV GX
Total logic elements	12 / 14,400 (< 1 %)
Total combinational functions	12 / 14,400 (< 1 %)
Dedicated logic registers	0 / 14,400 (0 %)
Total registers	0
Total pins	14 / 81 (17 %)
Total virtual pins	0
Total memory bits	0 / 552,960 (0 %)
Embedded Multiplier 9-bit elements	0
Total GXB Receiver Channel PCS	0 / 2 (0 %)
Total GXB Receiver Channel PMA	0 / 2 (0 %)
Total GXB Transmitter Channel PCS	0 / 2 (0 %)
Total GXB Transmitter Channel PMA	0 / 2 (0 %)
Total PLLs	0 / 3 (0 %)
Device	EP4CGX15BF14C6
Timing Models	Final

بخش دوم: مقایسه‌گر سریالی

- serial_comparator: در این بخش تنها همین یک ماژول را داریم. در این بخش چون می‌خواهیم نتیجه هر بخش را ذخیره کنیم و مدار ترتیبی‌ست، نیاز به فلیپ‌فلاپ نوع D داریم که به کمک این لینک طراحی شده است:

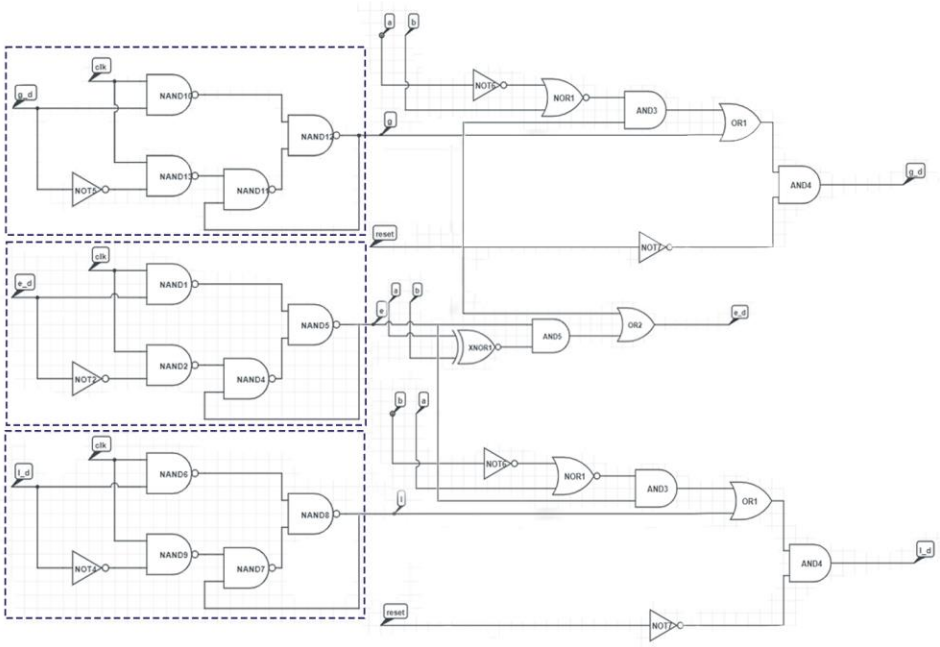


در این ماژول سه DFF داریم که هر یک برای یکی از سیگنال‌های خروجی‌ست. ورودی DFFهای مربوط ġ، l و e به ترتیب g_d، l_d و e_d و خروجی‌های آنها به ترتیب ġ، l و e می‌باشند. هر یک از سیگنال‌های g_d، l_d و e_d توسط مجموعه‌ای از مدارهای ترکیبی محاسبه می‌شوند. این مدارهای ترکیبی مشابه بخش اول هستند، ولی ساختار سلسله‌مراتبی ندارند.

برای مقایسه هر دو عدد ۴ بیتی در این ماژول، ابتدا پرارزش‌ترین بیت آن دو مقایسه می‌شود و به همین ترتیب تا کم‌ارزش‌ترین بیت پیش می‌رود.

چون در این بخش سیگنال reset نیز داریم، آن را باید با حاصل e_d، OR کنیم تا به حالت اولیه بازگردیم. برای ġ_d و l_d نیز چون در حالت اولیه هر دو صفرند، reset~ را با حاصل این مقدار AND می‌کنیم.

ساختار شماتیک این مدار به شکل زیر است:



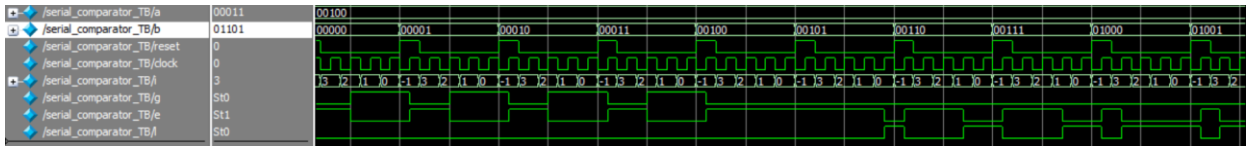
در کد Verilog این بخش داریم:

```
module serial_comparator (
    input   a,
    input   b,
    input   reset,
    input   clock,
    output  g,
    output  e,
    output  l
);

wire g_d, g_i1, g_i2, g_not;
wire e_d, e_i1, e_i2, e_not;
wire l_d, l_i1, l_i2, l_not;

/* g reg (g_d is input) */
assign g_i1     = ~(clock & g_d);
assign g_i2     = ~(clock & ~g_d);
assign g        = ~(g_i1 & g_not);
assign g_not    = ~(g_i2 & g);
/* g reg */

/* e reg (e_d is input) */
assign e_i1     = ~(clock & e_d);
assign e_i2     = ~(clock & ~e_d);
assign e        = ~(e_i1 & e_not);
assign e_not    = ~(e_i2 & e);
/* e reg */

/* l reg (l_d is input) */
assign l_i1     = ~(clock & l_d);
assign l_i2     = ~(clock & ~l_d);
assign l        = ~(l_i1 & l_not);
assign l_not    = ~(l_i2 & l);
/* l reg */


assign g_d = ~reset & (g | (e & ~((~a) | b)));
assign e_d = reset | (e & ~(a ^ b));
assign l_d = ~reset & (l | (e & ~((~b) | a)));

endmodule
```

- serial_comparator_TB: از این ماژول برای مقداردهی به a و b و امتحان کارکرد ماژول قبلی استفاده می‌کنیم و تمامی مقادیر ۴ بیتی ممکن برای a و b را بررسی می‌کنیم. چون مقایسه‌گر سریالیست، ورودی‌ها باید بیت به بیت داده شوند.

```
module serial_comparator_TB ();

reg [4:0]   a, b;
reg         reset = 1, clock = 0;
integer     i;
wire        g, e, l;

always #10 clock = ~clock;

serial_comparator COM(
    .a(a[i]),
    .b(b[i]),
    .reset(reset),
    .clock(clock),
    .g(g),
    .e(e),
    .l(l)
);

initial begin
    for (a = 0; a < 16; a = a + 1) begin
        for (b = 0; b < 16; b = b + 1) begin
            #20 reset = 0;
            for (i = 3; i >= 0; i = i - 1) begin
                #20;
            end
            $display("%d, %d, g = %d, e = %d, l = %d", a, b, g, e, l);
            reset = 1;
        end
    end
    $stop;
end

endmodule
```

- Waveform: برای نمونه بخشی از waveform این ماژول را بررسی می‌کنیم. در این بخش مقدار a برابر با ۴ است. اگر مقدار b برابر با ۳ باشد، با مقایسه پرارزش‌ترین بیت این دو عدد که هر دو صفر هستند، با لبه کلاک فقط سیگنال e برابر با 1 می‌شود و بقیه صفر می‌شوند. اما با مقایسه بیت بعدی، فقط سیگنال g برابر با ۱ می‌شود و دو سیگنال دیگر صفر می‌شوند و این مقادیر تا پایان مقایسه دو عدد ثابت می‌مانند. با فعال شدن reset، در کلاک بعد هر سه سیگنال خروجی به حالت اولیه خود باز میگردند.



- سنتز: مدار نهایی توسط کوارتوس سنتز شد و summary آن به شرح زیر است:

Flow Summary	
Flow Status	Successful - Tue Apr 06 11:30:26 2021
Quartus II 64-Bit Version	13.0.1 Build 232 06/12/2013 SP 1 SJ Web Edition
Revision Name	Tootooleh
Top-level Entity Name	serial_comparator
Family	Cyclone IV GX
Total logic elements	7 / 14,400 (< 1 %)
Total combinational functions	7 / 14,400 (< 1 %)
Dedicated logic registers	0 / 14,400 (0 %)
Total registers	0
Total pins	7 / 81 (9 %)
Total virtual pins	0
Total memory bits	0 / 552,960 (0 %)
Embedded Multiplier 9-bit elements	0
Total GXB Receiver Channel PCS	0 / 2 (0 %)
Total GXB Receiver Channel PMA	0 / 2 (0 %)
Total GXB Transmitter Channel PCS	0 / 2 (0 %)
Total GXB Transmitter Channel PMA	0 / 2 (0 %)
Total PLLs	0 / 3 (0 %)
Device	EP4CGX15BF14C6
Timing Models	Final