گزارش آزمایش سوم

دانشگاه صنعتی شریف – **زمستان ۹۹**

آزمایشگاه طراحی سیستم‌های دیجیتال – دکتر اجلالی

نویسندگان:
سروش جهان‌زاد – ۹۸۱۰۰۳۸۹
علی حاتمی تاجیک – ۹۸۱۰۱۳۸۵

۱ مقدمه

در این سند سعی شده تا گزارشی بر نحوه انجام آزمایش سوم درس آزمایشگاه طراحی سیستم‌های دیجیتال ارائه شود. کدهای مربوط به هر بخش و فایل تست بنچ آن در کنار این گزارش آمده است. از صبر و بردباری شما در مطالعه این سند بسیار سپاسگزاریم.

با احترام

سروش جهان‌زاد، علی حاتمی تاجیک – بهار ۱۴۰۰

۲ مقایسه‌کننده ۴ بیتی

Cascade-able 1-bit Comparator

طراحی

کد

تست

مقایسه‌کننده ۴ بیتی

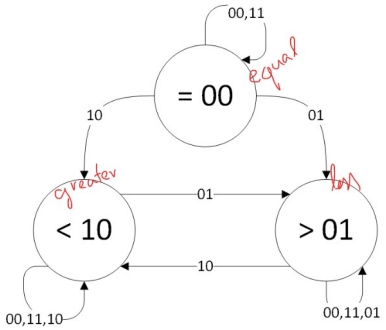
افزودن مقایسه کننده‌های یک بیتی و کد

تست

۳ مقایسه‌کننده سریال

طراحی مدار

این مدار به صورت زیر طراحی شده است که یک مدار ترتیبی سه وضعیتی است. به طور خلاصه اگر در هر وضعیتی بودیم اگر بیت با ارزش بیشتر بعدی با هم برابر بود در همان وضعیت میمانیم و غیر از این حالت اگر بیت اول صفر و دیگری یک بود به حالت کوچکتر و اگر برعکس بود به حالت بزرگتر خواهیم رفت. اگر از همان ابتدا نیز بیتها با یکدیگر برابر بودند در استیت صفر یا همان برابری باقی خواهیم ماند.



شکل ۱–حالتهای مقایسه کننده سریال

حال برای حالتهای بالا با استفاده از دو فلیپفلاپ نوع دی، جدول حالت و جداول کارنو ساده سازی ها را انجام میدهیم. از آنجایی که تضمین شده است که قبل از مقایسه حتما ماژول ریست خواهد شد به همین خاطر حالتی را که فلیپفلاپها هردو یک باشند را در نظر نمیگیریم.

Q_1	Q_0	a	b	Q_1^+	Q_0^+
0	0	0	0	0	0
0	0	0	1	0	1
0	0	1	0	1	0
0	0	1	1	0	0
0	1	0	0	0	1
0	1	0	1	0	1
0	1	1	0	1	0
0	1	1	1	0	1
1	0	0	0	1	0
1	0	0	1	0	1
1	0	1	0	1	0
1	0	1	1	1	0
1	1	0	0	0	X

جدول ۱–جدول حالت مقایسه کننده سریال

Q_1Q_0\ab	00	01	11	10
00	0	0	0	1
01	0	0	0	1
11	X	X	X	X
10	1	0	1	1

جدول ۲–جدول کارنو برای *D1*

Q_1Q_0\ab	00	01	11	10
00	0	1	0	0
01	1	1	1	0
11	X	X	X	X
10	0	1	0	0

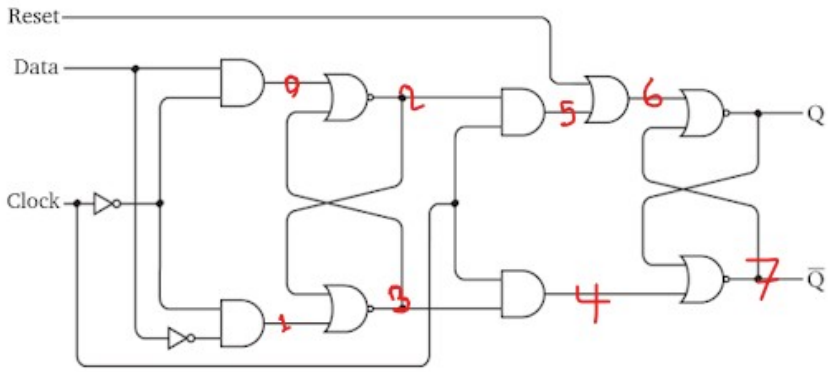
جدول ۳-جدول کارنو برای *D0*

که نتایج این دو جدول به صورت روبه‌رو خواهد بود:

$$D_1 = a\,\bar{b} + Q_1(a + \bar{b})$$
$$D_0 = \bar{a}\,b + Q_0(\bar{a} + b)$$

ساخت DFF

برای نگهداری از وضعیت فعلی مدار به دو فلیپ فلاپ نوع دی نیازمندیم. در ادامه شکلی از شماتیک فلیپ فلاپ پیادهسازی شده آمده است (ایندکس‌ـوایرها که داخل کد از آن‌ها به عنوان واسط استفاده شده است آمده‌است). پس از آن کد مربوط به یک فلیپ فلاپ آورده شده است که در ماژول مقایسه کننده این کد دو بار استفاده شده‌است و ورودی آن‌ها یعنی D1 و D2 با استفاده از خروجی آن‌ها تعیین شده است. در ابتدا نیز تمام سیمها را صفر میکنیم تا به مشکلی برنخوریم. برای صحت کارایی فلیپ فلاپ، آنرا به صورت یک ماژول درآورده‌ایم تا آنرا تست کنیم (فایل ماژول dff و تست آن نیز در پوشه مربوط به این بخش از آزمایش موجود است) ولی در ماژول اصلی دو دفعه از این کد آمده‌است تا فلیپ فلاپ ها ساخته شوند.



شکل ۲-*D Flip-Flop*

```
wire connectorFF[7:0];
wire q;
wire D;
assign connectorFF[0] = D && (~clk);
assign connectorFF[1] = (~D) && (~clk);
assign connectorFF[2] = ~(connectorFF[0] || connectorFF[3]);
assign connectorFF[3] = ~(connectorFF[1] || connectorFF[2]);
assign connectorFF[4] = connectorFF[3] && clk;
assign connectorFF[5] = connectorFF[2] && clk;
assign connectorFF[6] = connectorFF[5] || reset;
assign connectorFF[7] = ~(connectorFF[4] || q);
assign q              = ~(connectorFF[6] || connectorFF[7]);
```

ساخت مدار نهایی

حال که چهار سیم D_0, D_1 را داریم با توجه به نتایج بدست آمده‌از قسمت طراحی مدار این چهار سیم را مقدار دهی میکنیم و پس از آن نیز خروجی ها را اساین کرده (که عبارت بولی آن طبق شکل ۱ واضح است) و کارمان به پایان میرسد.

```
assign D1 = (a && (~b)) || (q1 && (a || (~b)));
assign D0 = (b && (~a)) || (q0 && (b || (~a)));
assign less = (~q1) && q0;
assign greater = (~q0) && q1;
assign equal = (~q1) && (~q0);
```

تست مدار

تمام کارایی های ماژول را درون یک تست می گنجانیم. مدار را برای سه جفت عدد تست می کنیم و بین هر کدام از reset ماژول استفاده می کنیم.

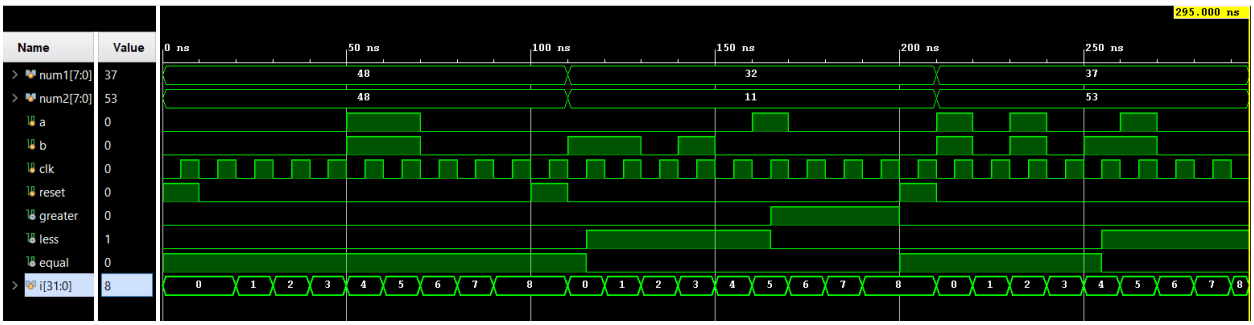
کد تست

در ادامه کد مورد استفاده برای تست ماژول را می‌بینیم.

```
module test_serial_comparator;
    reg [7:0]num1 = 48;
    reg [7:0]num2 = 48;
    reg a=0, b=0, reset = 0, clk = 0;
    wire greater, less, equal;
    serial_comparator uut(clk,a,b,reset,greater,equal,less);
    initial begin
    forever
        #5 clk = ~clk;
    end
    integer i=0;
    initial begin
        reset = 1; #10 reset = 0;
        for (i=0; i<8; i = i+1) begin
           a = num1[i];
           b = num2[i];
           #10;
        end
        #10 reset = 1;#10 reset = 0;
        num1 = 32; num2 = 11;
        for (i=0; i<8; i = i+1) begin
           a = num1[i];
           b = num2[i];
           #10;
        end
        #10 reset = 1;#10 reset = 0;
        num1 = 37;num2 = 53;
        for (i=0; i<8; i = i+1) begin
           a = num1[i];
           b = num2[i];
           #10;
        end
        #5 $finish;
    end
endmodule
```

نتایج تست

تصویر زیر نیز نتیجه اجرای تست بالاست. همانطور که مشخص است ریست به صورت سریع تأثیر خود را میگذارد اما تغییرات در a و b تنها در لبه‌های بالارونده است. همینطور زمانی که I برابر با ۸ می‌شود زمانی است که نتایج نهایی قابل مشاهده هستند. که ابتدا دو عدد برابر هستند، بعد عدد اول بزرگ‌تر است و سپس عدد اول کوچتر است. اما این‌ها تنها نتایج نهایی هستند و با توجه به a و b و num1 و num2 مشخص است که تأثیر گذاری بیت به بیت است (مثلاً در بخش دوم زمانی که اعداد ۱۱ و ۳۲ مقایسه می‌شوند، عدد ۱۱ در ۵ بیت ابتدایی بزرگ‌تر از عدد ۳۲ است اما در بیت ششم بزرگ‌تر شده است. اینکه در آن مرحله کدام بیت مورد برسی است از مقدار I مشخص می‌شود).



شکل ۳: نتیجه تست بر روی مقایسه‌کننده سریال