بسم الله الرحمن الرحیم

آزمایشگاه طراحی سیستم‌های دیجیتال

استاد اجلالی – مهندس سیادت‌زاده

آزمایش سوم

محمدحسین دولت‌آبادی ۹۸۱۰۵۷۷۳

زهره عباسی ۹۸۱۰۵۸۸۱

هدف آزمایش

در آزمایش اول می‌خواهیم یک مقایسه‌کننده چهار بیتی را به کمک ۴ مقایسه کننده یک بیتی با قابلیت انتشار پیاده‌سازی کنیم. در این آزمایش فقط از توصیف جریان داده استفاده می‌کنیم و تمامی پیاده‌سازی مدار به کمک دستور assign انجام می‌شود. در قسمت اول این آزمایش مدار به صورت ترکیبی طراحی شده و طراحی نیز سلسله‌مراتبی است.

در آزمایش دوم با استفاده از دستور assign و توصیف جریان داده یک مقایسه‌کننده سریال می‌سازیم. این مقایسه‌کننده یک مدار ترتیبی است که با استفاده از ورودی reset در اول کار reset می‌شود و پس از آن از دو ورودی خود بیت‌های دو عددی که باید مقایسه شوند را بیت به بیت گرفته و در هر پالس ساعت حاصل مقایسه را تا جایی که مقایسه کرده (تا بیتی که مقایسه شده) در خروجی سریال خود تحویل می‌دهد. برای پیاده‌سازی این مدار طراحی سلسله‌مراتبی انجام نمی‌دهیم و صرفا یک پیمانه (module) داریم که آن پیمانه نیز توصیف جریان داده شده است و از هیچ توصیف دیگری استفاده نشده است.

مدار اول

مقایسه‌کننده یک بیتی

در این مدار قصد داریم با اتصال ۴ مقایسه کننده یک بیتی یک مقایسه کننده چهار بیتی بسازیم. توصیف پیمانه مقایسه‌کننده یک بیتی به صورت زیر است:

```
module cascadable_1bit_comparator(
        input lt_in,
              eq_in,
              gt_in,
              a,
              b,
        output lt_out,
               eq_out,
               gt_out
);

        assign lt_out = lt_in | (eq_in & ~a & b);
        assign eq_out = eq_in & (a ~^ b);
        assign gt_out = gt_in | (eq_in & a & ~b);

endmodule
```

در این پیمانه برای ایجاد قابلیت انتشار ۳ ورودی برای نتیجه مقایسه قبلی خواهیم داشت که عبارتند از:lt_in ، eq_in و gt_in. و همچنین دو ورودی دیگر که دو بیتی هستند که باید مقایسه شوند.

۳ خروجی هم برای نمایش نتیجه مقایسه داریم که فعال شدن هریک نشان‌دهنده بزرگتربودن، مساوی‌بودن یا کوچکتر بودن بیت a از بیت b است.

خروجی مساوی بودن در صورتی فعال است که بیت‌های قبلی همه مساوی بوده باشند پس ورودی مساوی بودن هم یک است و همچنین بیت‌های a و b نیز با یکدیگر برابر باشند.

در حالت‌های کوچکتر یا بزرگتر بودن اگر از قبل مشخص شده باشد که A از B بزرگتر یا کوچکتر است کافی است اما در غیر این صورت باید مساوی باشند و همچنین بیت a از بیت b بزرگتر یا کوچکتر باشد. یعنی یا اینکه تکلیف از قبل مشخص شده است و این بیت کم ارزش دیگر تاثیری ندارد یا اینکه تا اینجا دو عدد بیت به بیت مساوی بوده اند و باید در این بیت تکلیف مشخص شود.

مقایسه‌کننده چهاربیتی

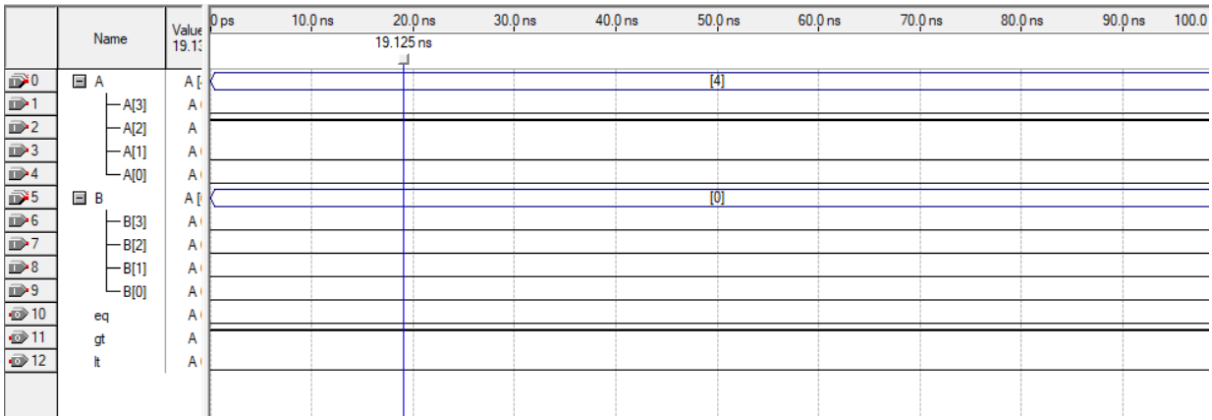
حال با گرفتن چهار نمونه از این پیمانه‌ها(یک نمونه به ازای هر بیت) و با طراحی سلسله‌مراتبی یک پیمانه بزرگتر می‌سازیم که دو عدد ۴ بیتی را مقایسه کند. برای این کار کافی است خروجی هر پیمانه یک بیتی را به ورودی پیمانه بعدی متصل کنیم و در نهایت خروجی‌های پیمانه آخر، خروجی‌های مدار است. پیاده‌سازی مدار مقایسه‌کننده چهاربیتی به شکل زیر است:

```
module comb_comp_4(
        input [3:0] A,
        input [3:0] B,
        output gt,
               eq,
               lt
);

        wire gt_3, gt_2, gt_1, gt_0;
        wire eq_3, eq_2, eq_1, eq_0;
        wire lt_3, lt_2, lt_1, lt_0;

        cascadable_1bit_comparator comp_3(.lt_in(1'b0), .eq_in(1'b1), .gt_in(1'b0), .a(A[3]), .b(B[3]), .lt_out(lt_3), .eq_out(eq_3), .gt_out(gt_3));
        cascadable_1bit_comparator comp_2(.lt_in(lt_3), .eq_in(eq_3), .gt_in(gt_3), .a(A[2]), .b(B[2]), .lt_out(lt_2), .eq_out(eq_2), .gt_out(gt_2));
        cascadable_1bit_comparator comp_1(.lt_in(lt_2), .eq_in(eq_2), .gt_in(gt_2), .a(A[1]), .b(B[1]), .lt_out(lt_1), .eq_out(eq_1), .gt_out(gt_1));
        cascadable_1bit_comparator comp_0(.lt_in(lt_1), .eq_in(eq_1), .gt_in(gt_1), .a(A[0]), .b(B[0]), .lt_out(lt_0), .eq_out(eq_0), .gt_out(gt_0));

        assign gt = gt_0;
        assign eq = eq_0;
        assign lt = lt_0;

endmodule
```

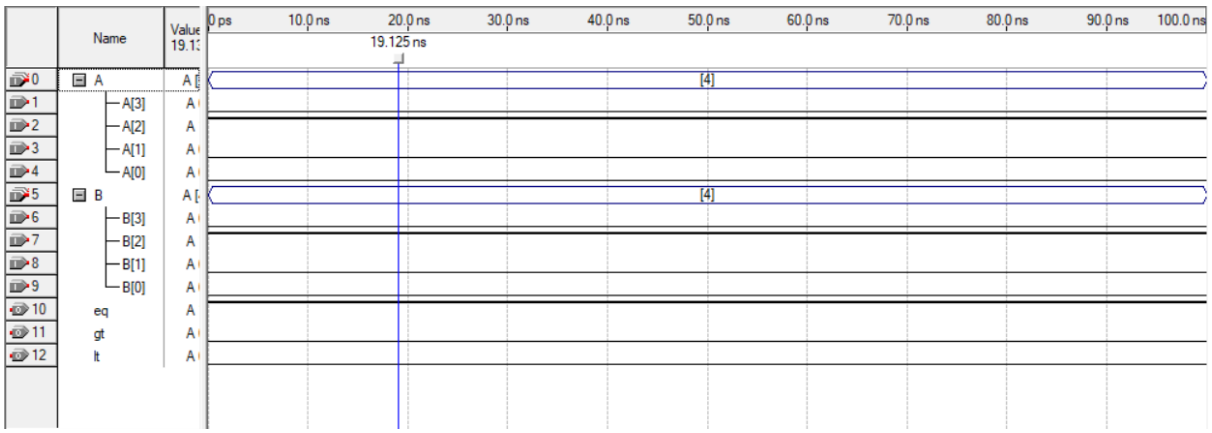
شبیه‌سازی و تست مدار

عکس‌های زیر شبیه‌سازی مدار را در نرم‌افزار Quartus نشان می‌دهد:

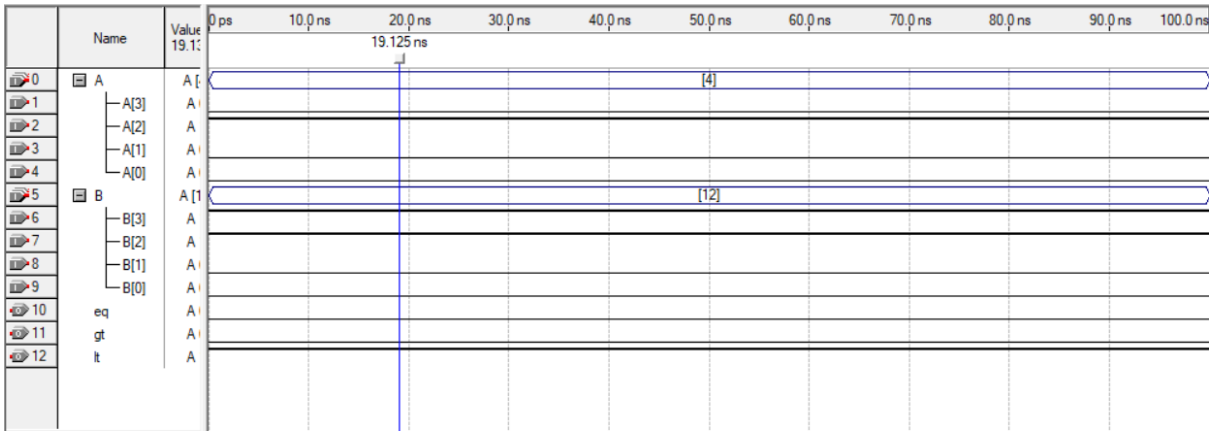
برای حالتی که A از B بزرگتر باشد:



حالت تساوی:



حالت کوچکتر


Name
Value
19.12
0 ps
10.0 ns
20.0 ns
30.0 ns
40.0 ns
50.0 ns
60.0 ns
70.0 ns
80.0 ns
90.0 ns
100.0 ns
19.125 ns
A
A[3]
A[2]
A[1]
A[0]
B
B[3]
B[2]
B[1]
B[0]
eq
gt
lt
[4]
[12]

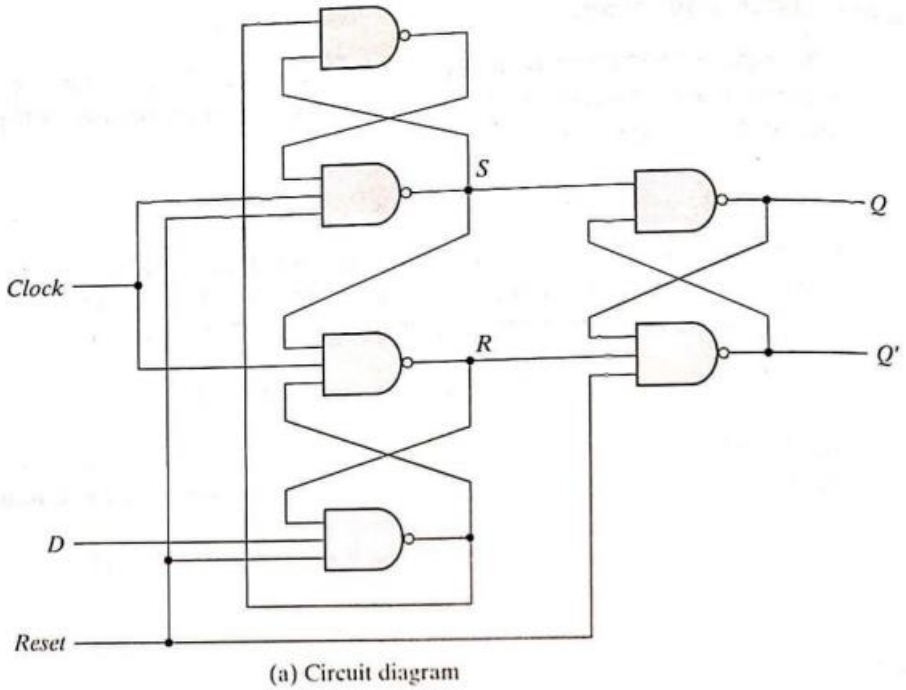
مدار دوم

مقایسه‌کننده تک‌پیمانه‌ای سریال

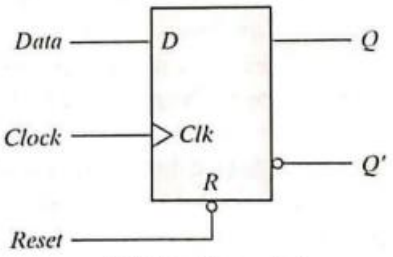
در این مدار قصد داریم با تنها یک پیمانه یک مقایسه کننده سریال بسازیم. پیاده‌سازی این مدار به صورت زیر است که در ادامه هر بخش به صورت مفصل توضیح داده شده است:

```
module seq_comp(
        input clk,
              reset,
              A,
              B,
        output gt,
               eq,
               lt
);

        wire FF1D, FF2D, FF1Q, FF2Q, FF1QP, FF2QP;
        wire FF1S, FF1R, FF11, FF14;
        wire FF2S, FF2R, FF21, FF24;

        //SR Dff data-flow 1
        assign FF11 = ~(FF1S & FF14);
        assign FF1S = ~(FF11 & clk & reset);
        assign FF1R = ~(FF1S & clk & FF14);
        assign FF14 = ~(FF1R & FF1D & reset);
        assign FF1Q = ~(FF1S & FF1QP);
        assign FF1QP = ~(FF1Q & FF1R & reset);

        //SR Dff data-flow 2
        assign FF21 = ~(FF2S & FF24);
        assign FF2S = ~(FF21 & clk & reset);
        assign FF2R = ~(FF2S & clk & FF24);
        assign FF24 = ~(FF2R & FF2D & reset);
        assign FF2Q = ~(FF2S & FF2QP);
        assign FF2QP = ~(FF2Q & FF2R & reset);

        // dff input
        assign FF1D = (A & ~B) | (FF1Q & (A | ~B));
        assign FF2D = (~A & B) | (FF2Q & (~A | B));

        //output
        assign eq = ~FF2D & ~FF1D;
        assign gt = ~FF2D & FF1D;
        assign lt = FF2D & ~FF1D;

endmodule
```

این مدار بین ۳ حالت کلی گذار می‌کند بنابراین نیاز به ۲ فیلپ‌فلاپ D داریم. چون قرار است که مدار به صورت تک پیمانه طراحی شود پیاده‌سازی این فیلپ‌فلاپ‌ها نیز در همان تک ماژول صورت می‌پذیرد. یک فلیپ‌فلاپ D با ریست آسنکرون به صورت زیر طراحی می‌شود:



(a) Circuit diagram



(b) Graphic symbol

R	Clk	D	Q	Q'
0	X	X	0	1
0	↑	0	0	1
0	↑	1	1	0

(b) Function table

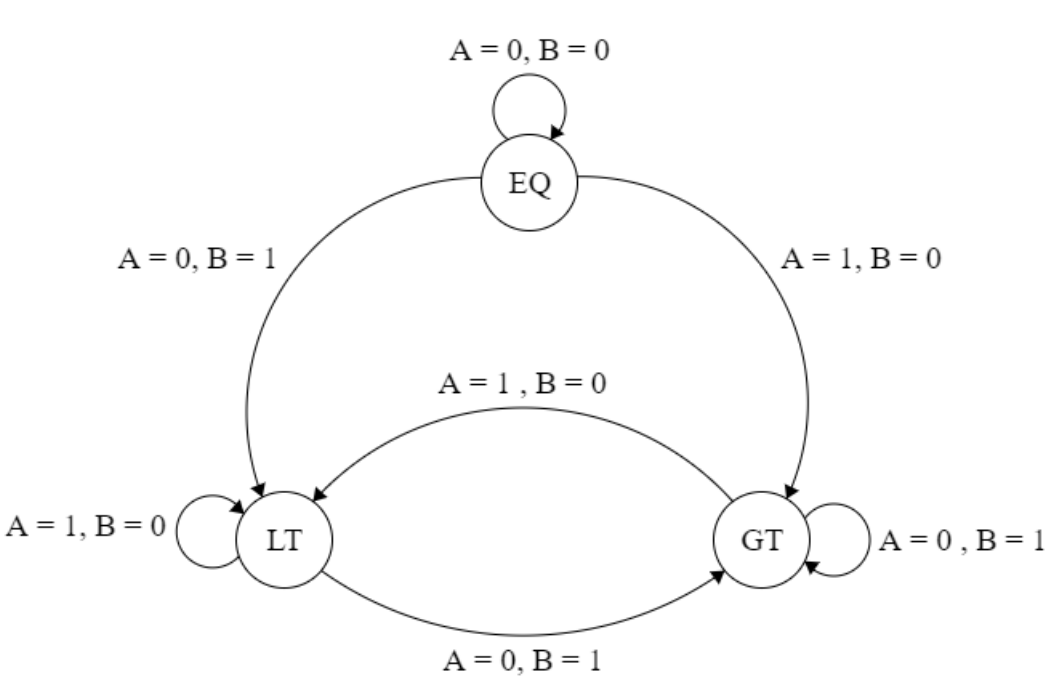
FIGURE 5.14
D flip-flop with asynchronous reset

توصیف پیمانه‌ی آن به صورت زیر است:

```
wire FF1D, FF2D, FF1Q, FF2Q, FF1QP, FF2QP;
wire FF1S, FF1R, FF11, FF14;
wire FF2S, FF2R, FF21, FF24;

//SR Dff data-flow 1
assign FF11 = ~(FF1S & FF14);
assign FF1S = ~(FF11 & clk & reset);
assign FF1R = ~(FF1S & clk & FF14);
assign FF14 = ~(FF1R & FF1D & reset);
assign FF1Q = ~(FF1S & FF1QP);
assign FF1QP = ~(FF1Q & FF1R & reset);

//SR Dff data-flow 2
assign FF21 = ~(FF2S & FF24);
assign FF2S = ~(FF21 & clk & reset);
assign FF2R = ~(FF2S & clk & FF24);
assign FF24 = ~(FF2R & FF2D & reset);
assign FF2Q = ~(FF2S & FF2QP);
assign FF2QP = ~(FF2Q & FF2R & reset);
```

با توجه به اینکه مدار ترتیبی طراحی شده، باید با استفاده از وضعیت فعلی و همچنین ورودی‌ها وضعیت بعدی مدار را تعیین کنیم. از آنجا که سه حالت نهایی برای مدار داریم، می‌دانیم در صورتی دو عدد مساوی هستند که بیت‌های آنها بیت به بیت با هم مساوی باشد. اگر در وضعیت بزرگتر قرار بگیریم تنها در صورتی به وضعیت کوچکتر می‌رویم که وضعیت قبلی یعنی حاصل مقایسه بیت‌های قبلی در وضعیت کوچکتر بوده باشد و به همین شکل برای وضعیت بزرگتر. و برای حالت مساوی هم در صورتی می‌مانیم که حالت قبلی در وضعیت مساوی بوده باشیم و ورودی‌های جدید هم به همین صورت باشند.

دیاگرام حالت‌های ماشین به صورت زیر است:



حال با استفاده از جداول کارنو به دو فرمول زیر می‌رسیم:

$$D_0 = Q_0^{n+1} = AB' + Q_0(A + B')$$

$$D_1 = Q_1^{n+1} = A'B + Q_1(A' + B)$$

و با توجه به فرمول‌های بدست آمده و شماره‌های هر حالت مساوی(۰۰) و بزرگتر (۰۱) و کوچکتر (۱۰) مقادیر خروجی‌ها را مشخص می‌کنیم:

```
// dff input
assign FF1D = (A & ~B) | (FF1Q & (A | ~B));
assign FF2D = (~A & B) | (FF2Q & (~A | B));

//output
assign eq = ~FF2D & ~FF1D;
assign gt = ~FF2D & FF1D;
assign lt = FF2D & ~FF1D;
```

شبیه‌سازی و تست مدار

مقایسه دو عدد ۱۱۰۱ و ۱۱۱۰: