به نام خدا

گزارش کار آزمایشگاه طراحی سیستم‌های دیجیتال

محمدحسین تاج‌الدینی ۹۴۱۷۰۸۲۸
محمدحسین حاجی سید سلیمان ۹۸۱۰۵۶۷۸
محمدعلی محمدخانی ۹۸۱۰۲۲۵۱

آزمایش سوم – توصیف جریان داده

بخش صفر – خلاصه کار جلسه

باید یک کد برای مقایسه کننده یک بیتی ترکیبی بنویسیم و آنرا تست کنیم. سپس با اتصال چهار عدد از این مقایسه‌گرها باید یک مقایسه‌گر چهار بیتی ساخته شود. در مرحله بعد باید یک مقایسه کننده سریال با محدودیت یک ماژول بسازیم که یک ورودی reset هم دارد و با پالس ساعت نتیجه مقایسه را اعلام می‌کند. برای توصیف جریان داده باید از دستور assign استفاده شود.

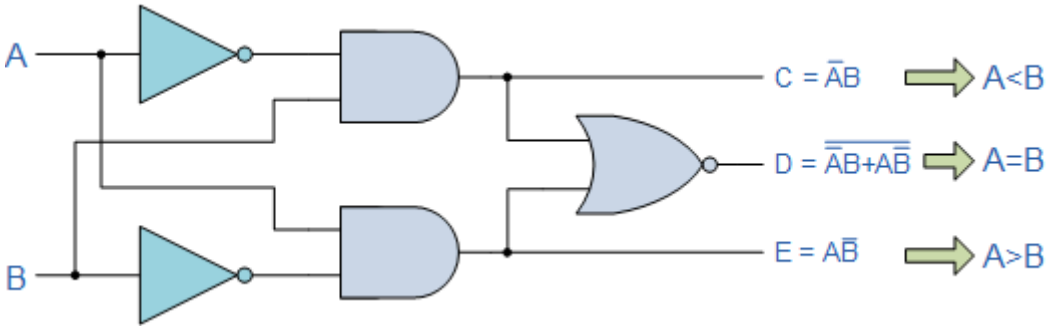
بخش یک – مقایسه‌گر ترکیبی

بخش ۱.۱ – پیدا کردن منطق مدار

یک مقایسه‌گر یک بیتی دو ورودی برای بیت‌های داده شده و سه خروجی برای حالات ممکن دارد. ورودی‌ها و خروجی‌ها طبق جدول زیر است:

نوع	نام	توضیح
ورودی	A	بیت اول
ورودی	B	بیت دوم
خروجی	A_le_B	بیت اول کمتر از بیت دوم
خروجی	A_eq_B	بیت اول مساوی بیت دوم
خروجی	A_gr_B	بیت اول بیشتر از بیت دوم

بلوک دیاگرام چنین مداری به شکل زیر است:



در نهایت کد وریلاگ این مدار به شکل زیر است:

```
module C31(
    input A,
    input B,
    output A_le_B,
    output A_eq_B,
    output A_gr_B
    );
    assign A_le_B = ~A & B;
    assign A_gr_B = ~B & A;
    assign A_eq_B = ~(A_le_B & A_gr_B);
endmodule
```

کد فوق با تست بنچ زیر بررسی شد و بدون ایراد اجرا شد:

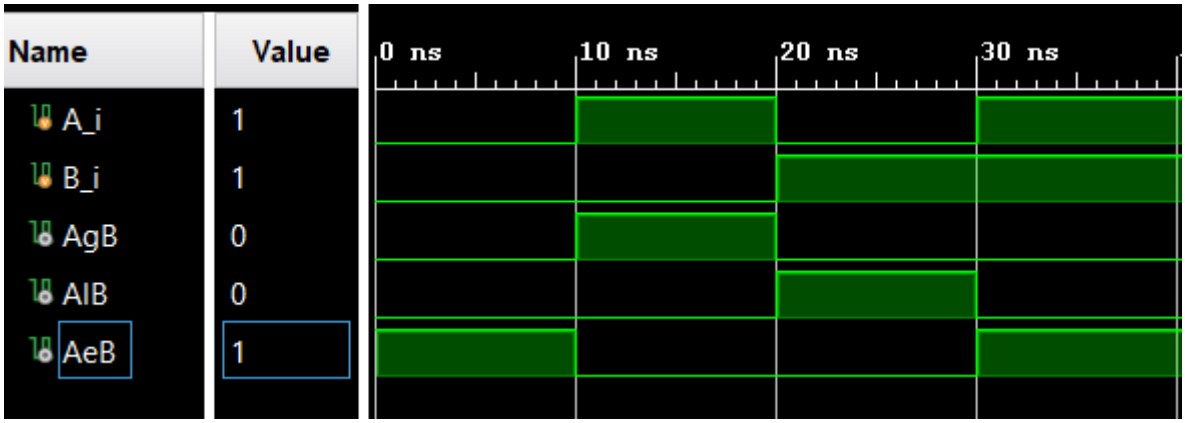
```
module tb();
reg A_i;
reg B_i;
wire AgB;
wire AlB;
wire AeB;

initial
begin
A_i <= 0;
B_i <= 0;
#10
A_i <= 1;
B_i <= 0;
#10
A_i <= 0;
B_i <= 1;
#10
A_i <= 1;
B_i <= 1;
end


C31 test_31 (
    .A(A_i),
    .B(B_i),
    .A_le_B(AlB),
    .A_eq_B(AeB),
    .A_gr_B(AgB)
    );

endmodule
```

نتایج سنتز به شکل زیر بود:



که با جدول درستی مد نظر تطابق دارد:

A	B	A < B	A = B	A > B
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0

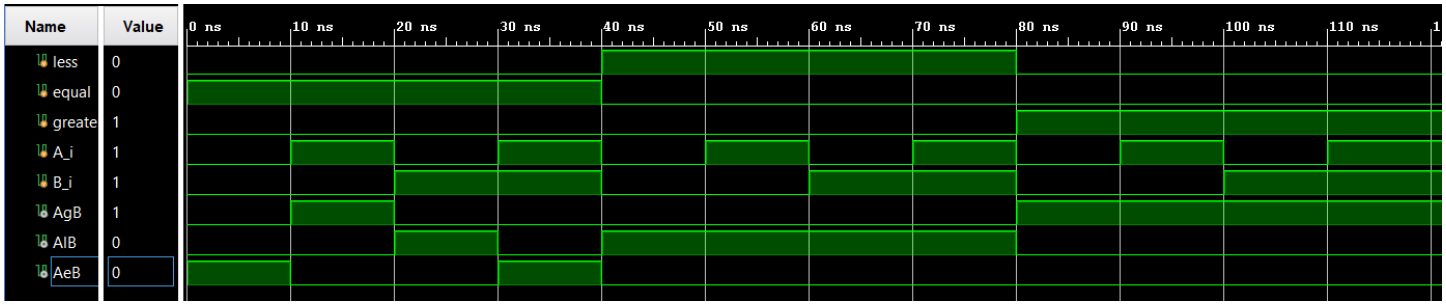
حال، برای منطق متصل کردن مدارها به یکدیگر برای روش cascade باید منطق را مشخص کنیم. برای این منظور، باید به ورودی مقایسه کننده، نتیجه مقایسه پیشین را بدهیم. در این صورت، مقایسه کننده تنها در صورتی که مقایسه پیشین به نتیجه تساوی رسیده باشد مقایسه می‌کند و در غیر این صورت، همان مقدار مقایسه پیشین را بازمی‌گرداند. روش کار بدین صورت خواهد بود که باید بیت‌های ورودی را به صورت زوج‌های با ارزش یکسان جدا کرده و از پر ارزش ترین زوج بیتها مقایسه را شروع کنیم و خروجی را تا مقایسه کم ارزش ترین بیت ها cascade کنیم. برای صحت کار به ورودی اولین مقایسه‌گر حالت تساوی (A>B:0, A=B:1, A<B:0) می‌دهیم.

نکته:
حالت ورودی حتما mutually exclusive است. پس برای ورودی‌های اشتباه، خروجی صحیحی نخواهیم گرفت.

کد ویرایش شده برای پشتیبانی از cascade کردن به صورت زیر شد:

```
module C31(
    input le,
    input eq,
    input gr,
    input A,
    input B,
    output A_le_B,
    output A_eq_B,
    output A_gr_B
    );
    assign A_le_B = eq ? ~A & B : le;
    assign A_gr_B = eq ? ~B & A : gr;
    assign A_eq_B = eq ? ~(A_le_B | A_gr_B) : eq;
endmodule
```

و کد تست آن در فایل tb1.v موجود است. برای صرفه‌جویی در فضا از تکرار تست پرهیز می‌کنم. در فایل مذکور حالات ممکن A و B را برای هر سه حالت ممکن ورودی بررسی کردم. توقع داریم در صورتی که ورودی حالت کمتر یا بیشتر باشد خروجی همان کمتر یا بیشتر شود و فقط در صورت تساوی مقادیر بررسی شوند. نتیجه شبیه‌سازی کد فوق به صورت زیر شد که با توقعات ما همخوانی دارد:



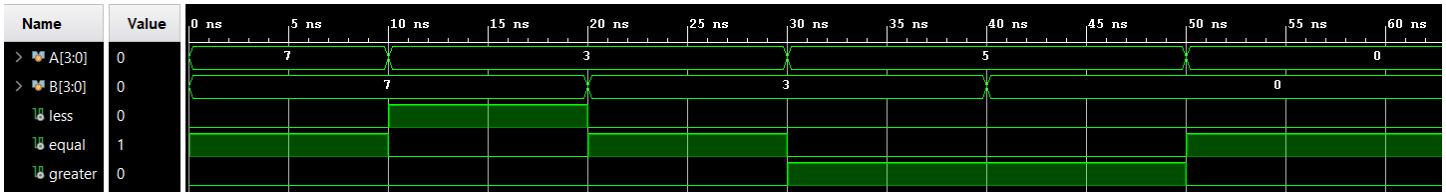
بخش ۱.۲ – ساخت مدار، شبیه سازی و سنتز

از ماژولی که در بخش پیشین طراحی شد چهار instance می‌گیریم و طبق گفته‌های قبل سیم کشی داخلی را با wire مشخص کرده و متصل می‌کنیم.

برای شبیه سازی حالات زیر بررسی شدند:

A	B	EXPECTATION
7	7	Equal
3	7	Less
3	3	Equal
5	3	Greater
5	0	Greater
0	0	Equal

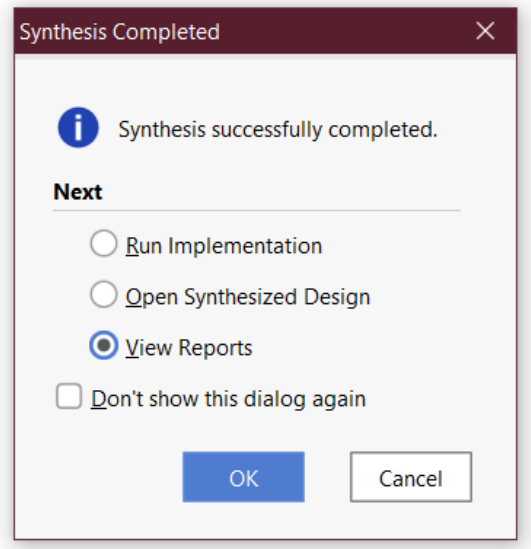
و نتیجه شبیه سازی به صورت زیر شد:



که مشاهده می‌شود صحیح است.

<< فایل های کد ماژول و تست بنچ به نامهای CascadableComparators.v و Sim1.v پیوست شده است.

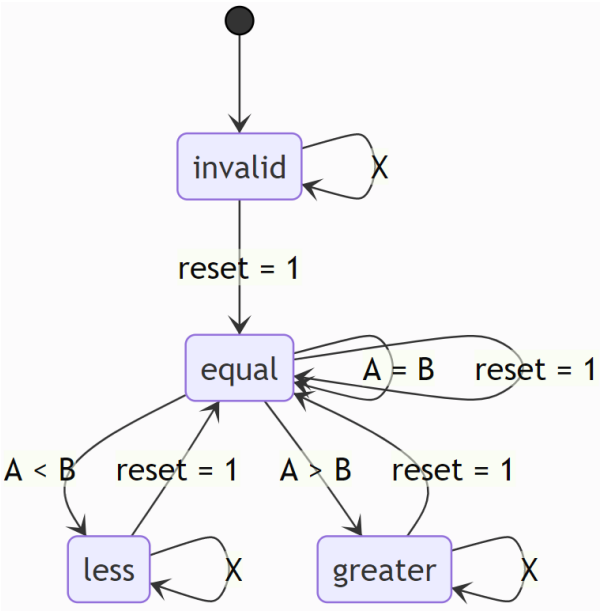
در نهایت سنتز پذیری را بررسی می‌کنیم که طبق پیام زیر موفقیت آمیز بوده است:



بخش دو – مقایسه کننده سریال

مشابه مسئله بخش پیشین، این بار باید یک مقایسه کننده سریال بسازیم. در این بخش محدودیت استفاده از دستور assign برای مشخص کردن جریان داده داریم. همچنین، تنها مجاز به تعریف یک ماژول هستیم. در این مدار فرض بر آن است که از MSB به LSB بیت‌های هم ارزش را دریافت کرده و در صورت نیاز zero-padding داریم. منطق مدار مطابق بخش بعد خواهد شد.

بخش ۲.۱ – منطق مدار



مدار سه ورودی A و B و reset را دریافت می‌کند و سه خروجی less و equal و greater را تولید می‌کند. در بخش ۱.۱ – پیدا کردن منطق مدار دیدیم که این مقادیر چگونه بدست می‌آیند. در این قسمت از آنجا که فقط یک ماژول داریم خروجی ماژول باید به ورودی آن برود و از آنجا که مدار باید به خودی خود کامل باشد این کار را نباید در بخش شبیه سازی انجام داد. پس از wire برای توصیف این اتصالات داخلی استفاده می‌کنیم. منطق کلی برای این مدار به شکل روبرو می‌شود:

نکته اینجاست که مدار به این شکل با دریافت ورودی ها خروجی خود را تغییر می‌دهد، پس سریال شده؛ اما هنوز مفهومی از clock برای آن تعریف نشده و به محض دریافت سیگنال‌ها در ورودی خروجی می‌دهد. از آنجا که در دستور کار از واژه پالس ساعت استفاده شده، می‌توانیم یک مرحله جلوتر هم برویم و کد را به نحوی تغییر دهیم که خروجی‌ها تنها با یک شدن مقدار کلاک تغییر کنند.

پس دو حالت را برای این مدار طراحی می‌کنیم: با کلاک و بدون کلاک.

بخش ۲.۲ – ساخت مدار، شبیه سازی و سنتز

برای حالت بدون کلاک کد مدار به شکل زیر است:

```
module Serial(
    input A,
    input B,
    input reset,
    output less,
    output equal,
    output greater
    );
    wire state_less;
    wire state_equal;
    wire state_greater;

    assign state_less = reset ? 0 : (state_equal ? (~A & B) : state_less);
    assign state_greater = reset ? 0 : (state_equal ? (A & ~B) : state_greater);
    assign state_equal = reset ? 1 : ~(state_less | state_greater);

    assign less = state_less;
    assign equal = state_equal;
    assign greater = state_greater;
endmodule
```

کد فوق در فایل Serial-NC.v پیوست شده است.

سپس، طبق جداول زیر شبیه سازی برای مدار انجام شد:

(بین هر تست یک سیگنال ریست داده شده)

تست ۱ (A=110 و B=100):

A	B	EXPECTATION
1	1	Equal
1	0	Greater
0	0	Greater
...	...	Greater

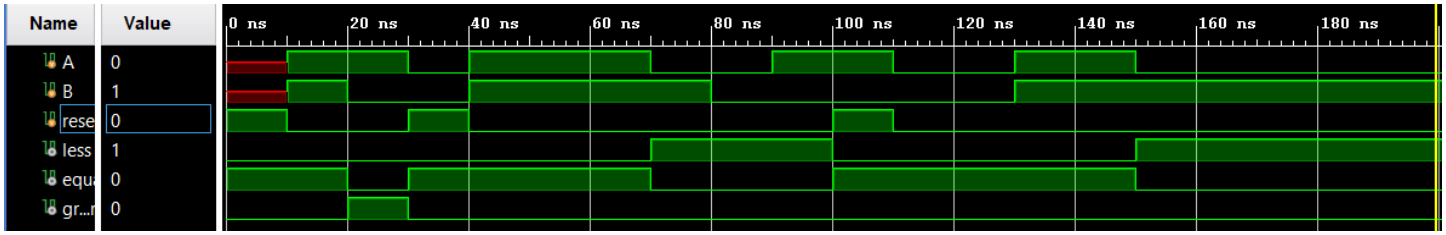
تست ۲ (A=1110011 و B=1111000):

A	B	EXPECTATION
1	1	Equal
1	1	Equal
1	1	Equal
0	1	Less
0	0	Less
1	0	Less
1	0	Less
...	...	Less

تست ۳ (A=00110 و B=00111):

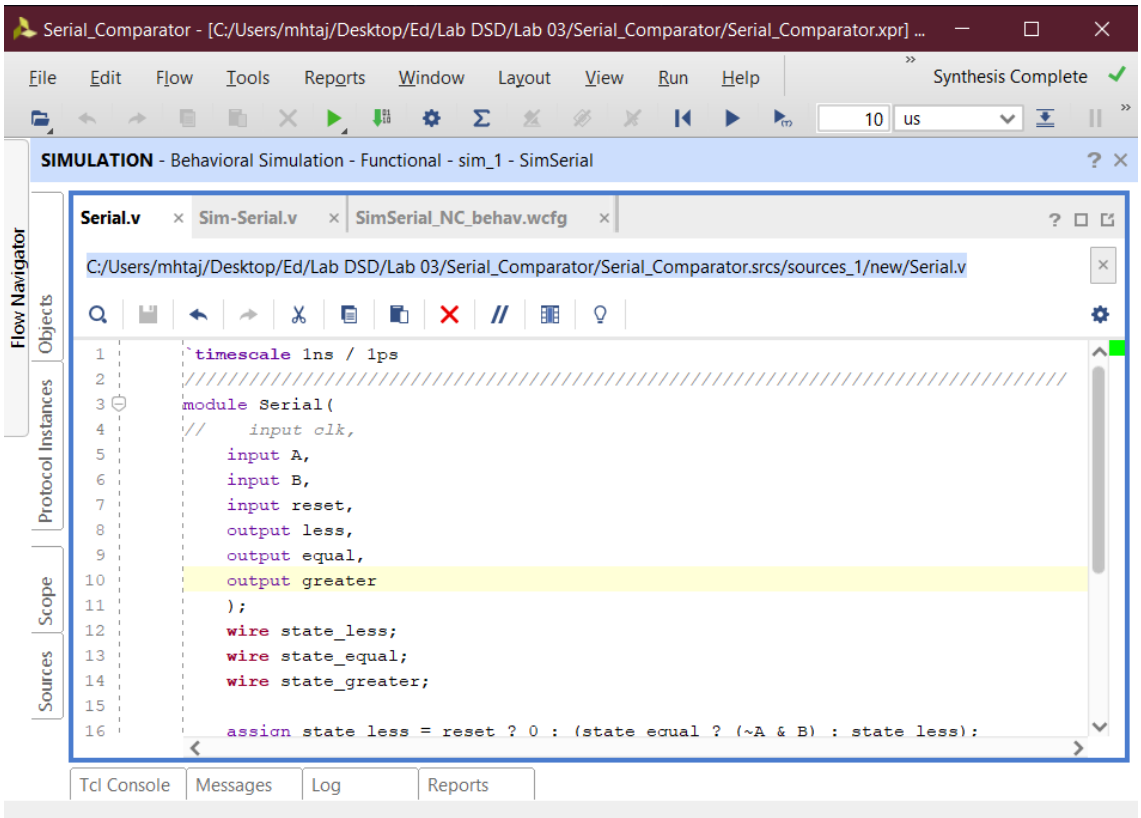
A	B	EXPECTATION
0	0	Equal
0	0	Equal
1	1	Equal
1	1	Equal
0	1	Less
...	...	Less

نتیجه شبیه سازی مطابق شکل زیر شد:



فایل شبیه سازی و تست به نام‌های Sim_Serial-NC.v و SimSerial_NC_behav.wcfg پیوست شده است.

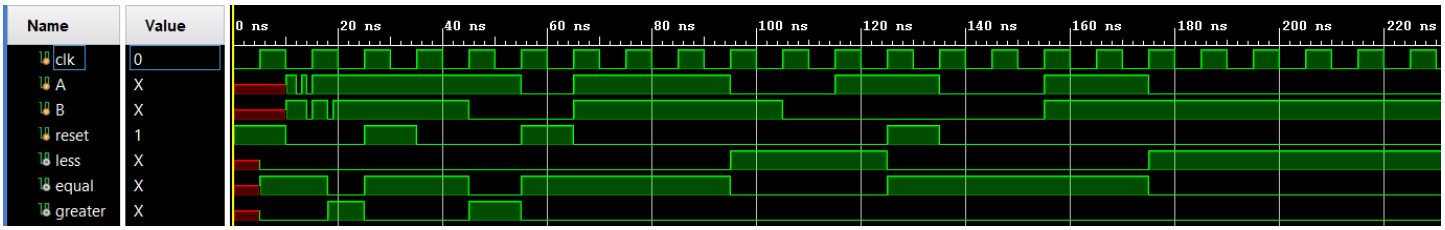
در نهایت، این کد با موفقیت سنتز شد:



در نهایت برای حالتی که کلاک داشته باشیم کد را به فرم زیر تغییر می‌دهیم:

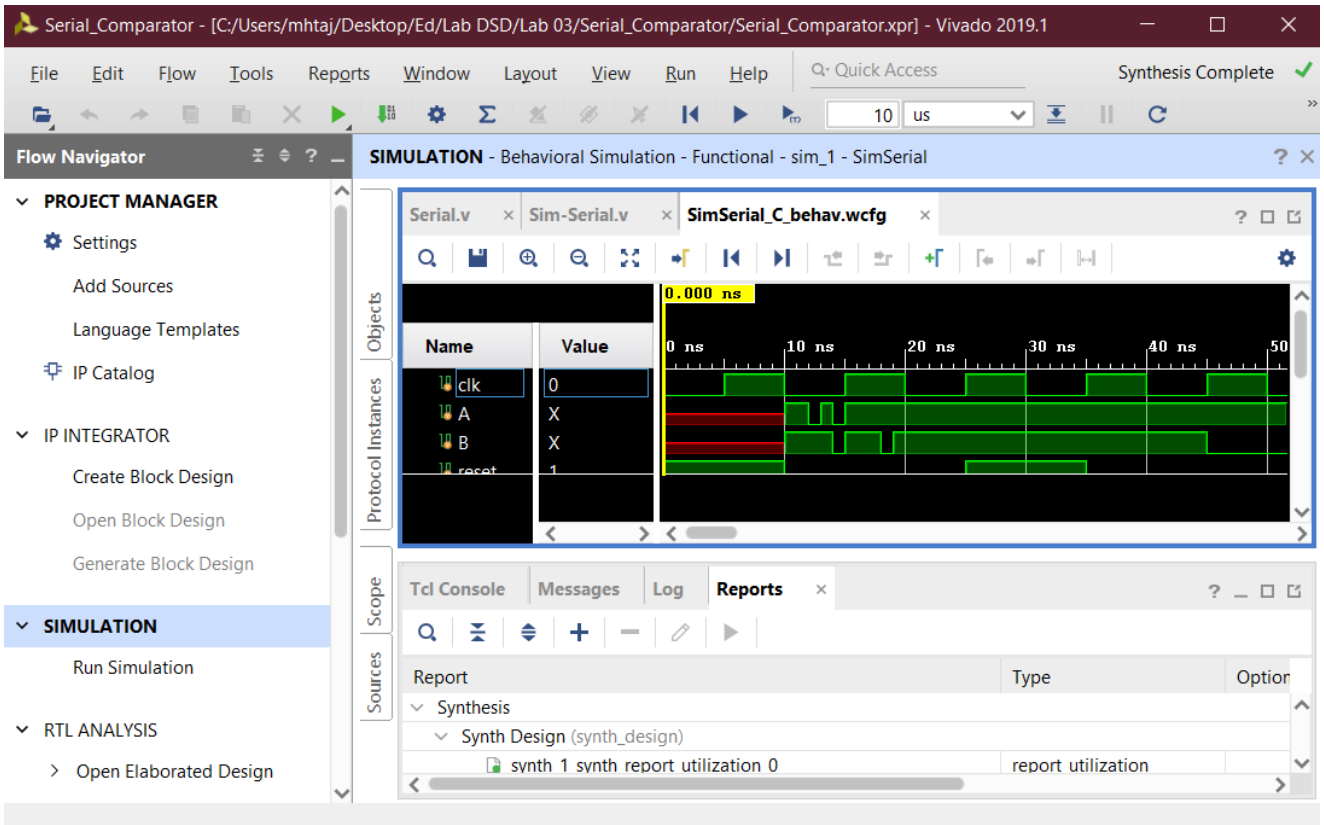
```
assign state = clk ? (reset ?) : state;
```

کد بالا فقط با کلاک یک کار می‌کند و با کلاک صفر ثابت می‌ماند. در نهایت تست کیس‌های پیشین را به علاوه تست برای ثابت ماندن با کلاک برای این مدار هم بررسی کرده و نتایج به شکل زیر شد:



در تست فوق ابتدا می‌بینیم که وقتی کلاک صفر است کاری انجام نمی‌شود. سپس می‌بینیم تمامی تغییرات زمانی که کلاک یک است تاثیر دارند. سپس مدار را ریست کرده و تست‌های سریال را برای آن بررسی می‌کنیم. مشاهده می‌شود نتایج طبق انتظارات است.

در نهایت، این کد با موفقیت سنتز شد:



تمامی فایل‌ها در پوشه‌های متناظر خود پیوست شده‌اند.