**به نام خدا**

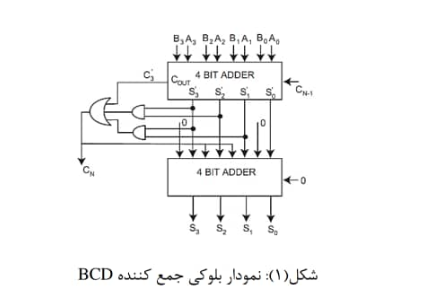
**آزمایش اول**

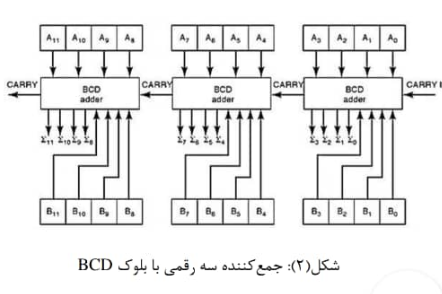
**امین ساوه دورودی – محمدرضا اسکینی – کامیاب عابدی**

**آزمایش الف**

از در درس معماری و مدار منطقی میدانیم که سیستم BCD یا همان Binary Coded Decimal صرفا دارای اعداد 0 تا 9 میباشد. ( یا به عبارت دیگر در مبنای 2 به میتوان را دامنه اعداد را از 0000 تا 1001 در نظر گرفت)

از نکات بالا میتوان نتیجه گرفت که در این سیستم و در 4 بیت نمی توان اعداد بزرگتر از 9 ( 10 به بالا) را در 4 بیت نمایش داد. پس به بیت های بیشتر نیاز داریم. برای مثال برای عدد 10 به 8 بیت ( یک بایت ) نیاز داریم تا برای هر رقم آن، 4 بیت در نظر بگیریم.و هر کدام از اعداد را در سیستم BCD حساب و در کنار هم قرار دهیم.

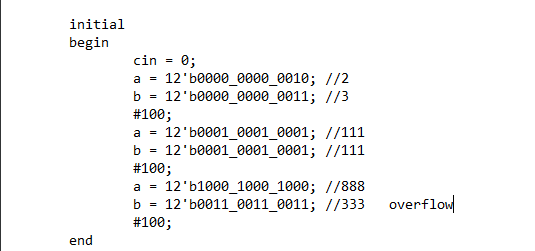




همانطور که در دستور کار آورده شده است شکل های ۱ و ۲ در بالا مشخص شده اند که طبق آن به طراحی ماژول ها میپردازیم و در ادامه از آنها برای ساختن جمع کننده سه رقمی استفاده میکنیم.

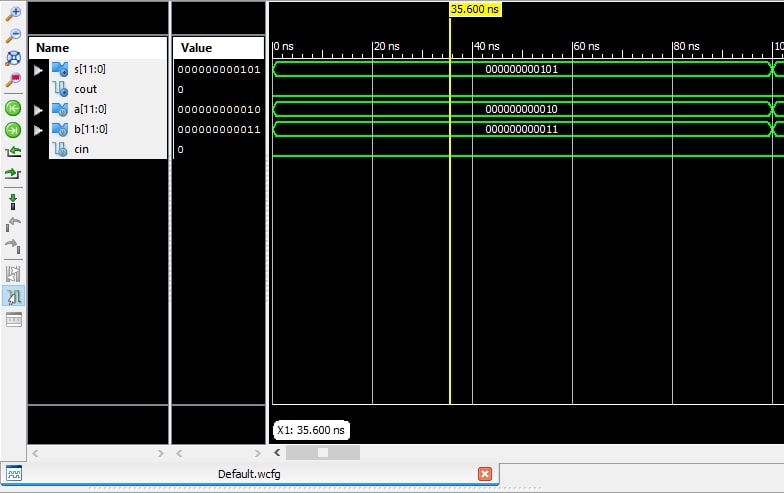
تست :

 در ادامه شماتیک های داده شده در دستور کار را با کمک Verilog شبیه سازی و بررسی میکنیم. در زیر تعدادی از خروجی ها و شبیه سازی ها آورده شده است:



داده های ورودی در تست بنچ

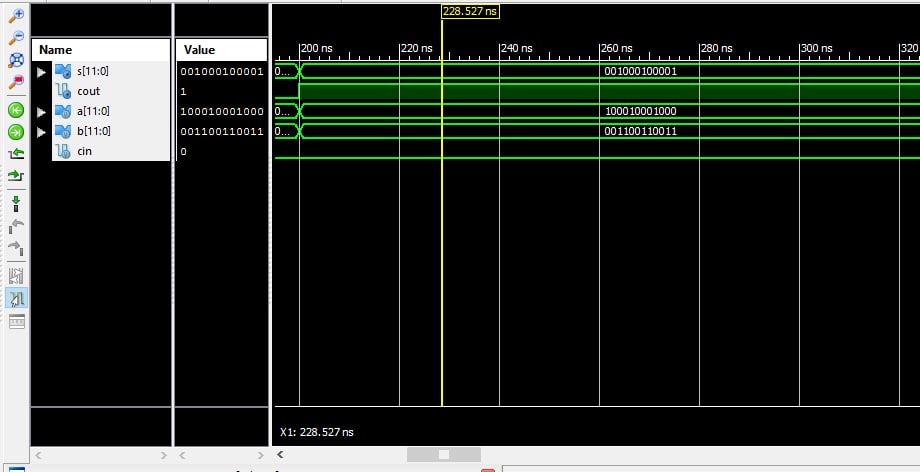
\* در خروجی های زیر، Radix بر روی Binary ست شده است. چرا که کدینگ ما به صورت BCD است.



خروجی برای جمع دو عدد 2 و 3 = 5



خروجی برای جمع دو عدد 111 و 111 = 222

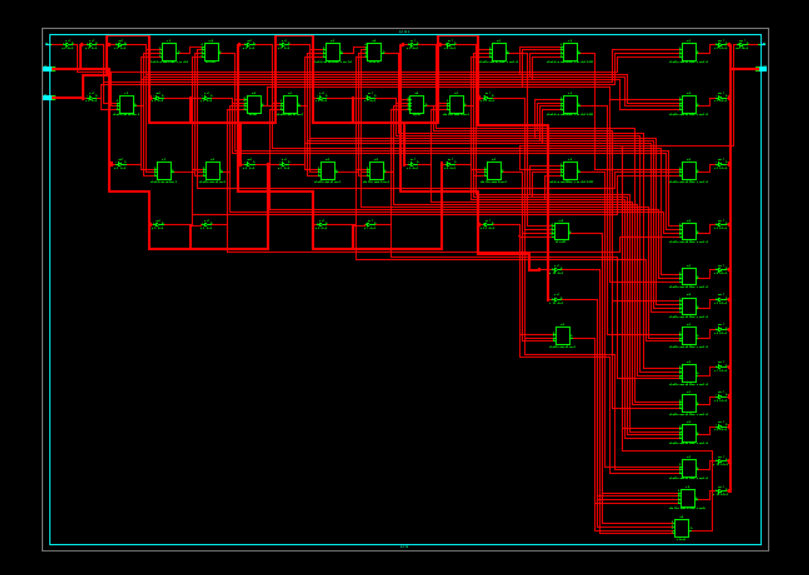


خروجی برای جمع دو عدد 333 و 888= 1221

در این خروجی همان طور که دیده میشود حاصل جمع چهار رقمی میباشد و overflow رخ داده است که

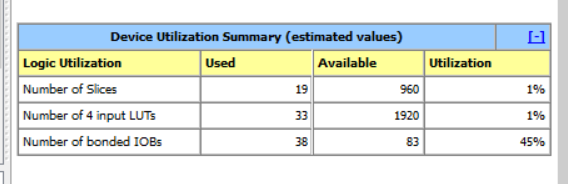
همان طور که مشخص است Cout نیز 1 شده است.

همانطور که در دستور کار نیز گفته شد است، قرار دادن مدار های سنتز شده کاملا اختیاری است برای این تمرین اما ما آن ها را نیز خروجی گرفتیم .



(RTL)

خلاصه طراحی :

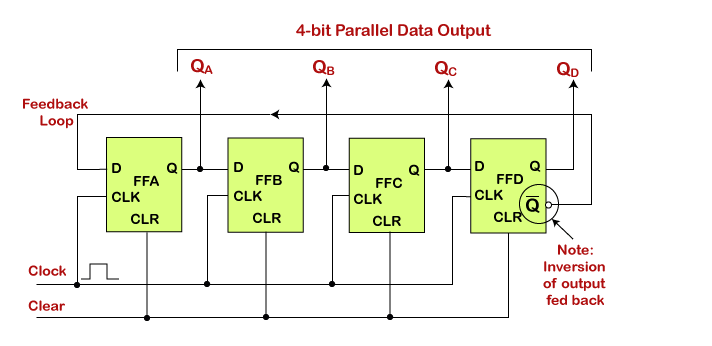
****

**آزمایش ب**

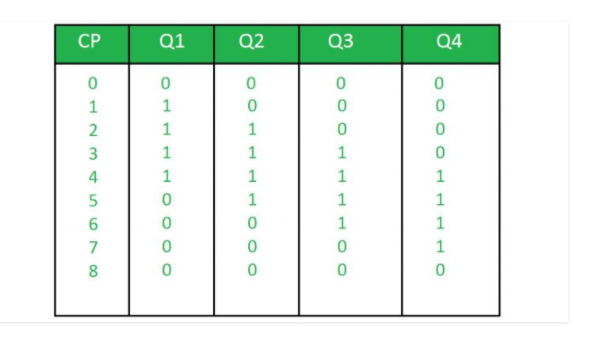
**Johnson Counter**

در این قسمت به پیاده سازی و شبیه سازی Johnson Counter به کمک زبان توصیف سخت افزار Verilog می پردازیم:

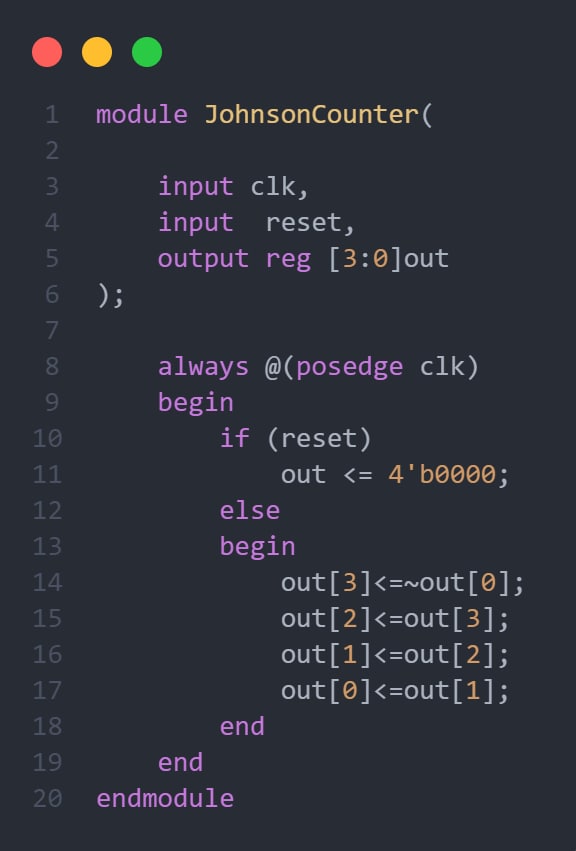
شیوه کاری این شمارنده به این صورت است که بیت آخر را (MSB) به صورت معکوس شده به  بیت اول ( LSB) بر میگرداند. به همین دلیل است که یک الگوی 8 تایی در طول زمان این شمارشگر ( با 4 بیت خروجی) ایجاد میکند که این 8 الگو در صورت دستور کار آورده شده است.



شماتیک



جدول درستی



کد استفاده شده برای شبیه سازی شمارنده

همانطور که در شکل بالا دیده میشود، ما از روش توصیف رفتاری استفاده کرده ایم. و همینطور یک سیگنال reset نیز برای شروع کار این دستگاه و راه اندازی مجدد آن در نظر گرفته شده است.

در و در ادامه در هر لبه بالارونده کلاک، 4 بیت را یک واحد به سمت چپ شیف میدهیم. نکته مهم در این قسمت این است که بیت آخر (که معمولا دور ریخته میشود) در اینجا نقیض و به بیت اول منتقل میشود..

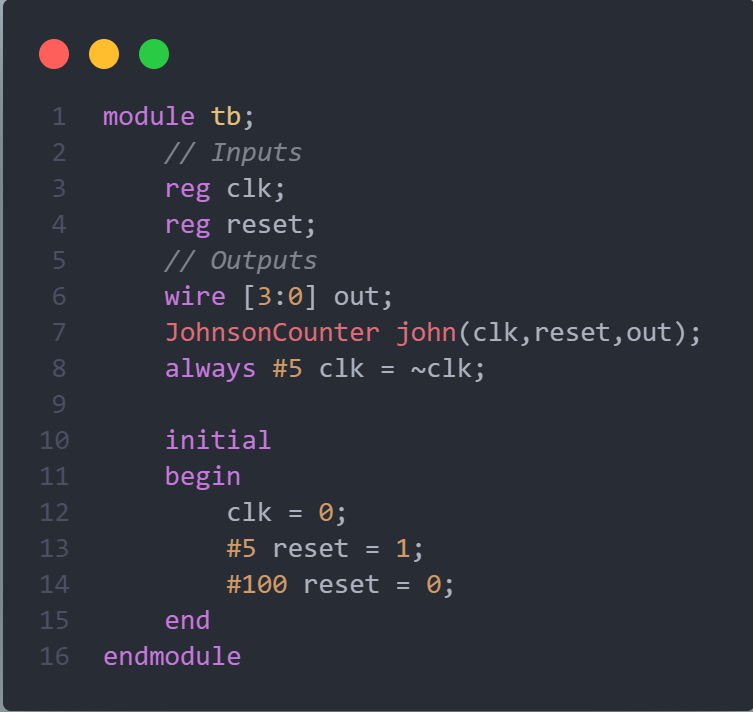
مزایا:

برای پیاده سازی یک الگوی تکرار شونده در 2n کلاک تنها به n فلیپ فلاپ نیاز داریم. در مقایسه با Ring Counter که در پایین توضیح داده خواهد شد نیز حالت های بیشتری را میتواند نشان دهد چرا که میتوانیم همزمان چند بیت با مقدار یک داشته باشیم.

معایب :

تمامی حالات ممکن از ۱۵ حالت را پوشش نمیدهد و تنها ۸ حالت را در بر میگیرد و تمامی حالاتی که چهاربیت میتوانند باشند را شامل نمیشود.

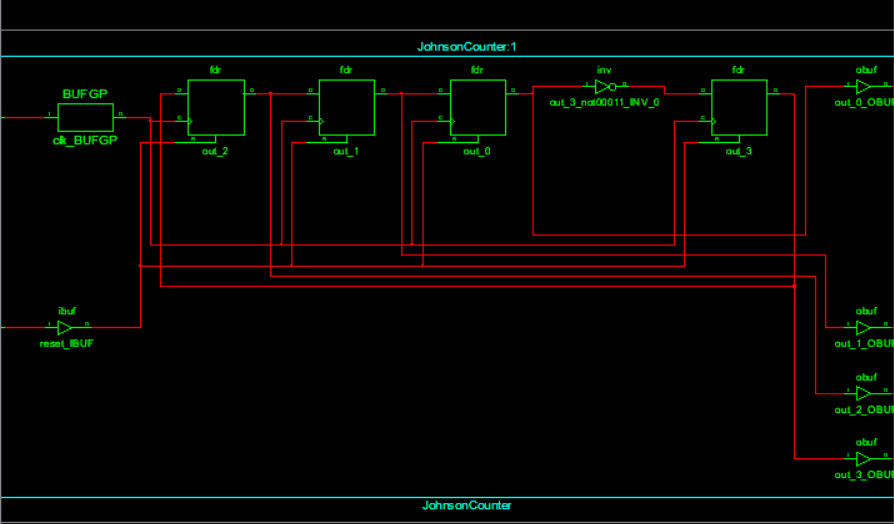
تست :



کد استفاده شده در تست بنچ

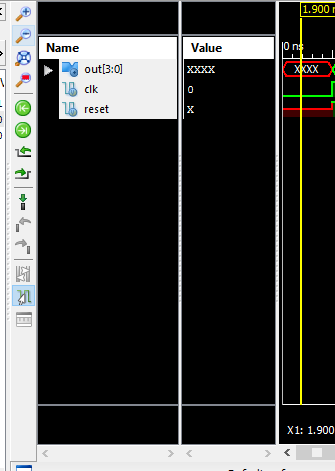
برای کلاک، یک سیگنال که هر 5 واحد زمانی عوض می شود در نظر گرفته ایم.

در ادامه به مدت یک لبه کلاک، مقدار reset را 1 کرده ایم تا دستگاه ما مقدار دهی اولیه شود.

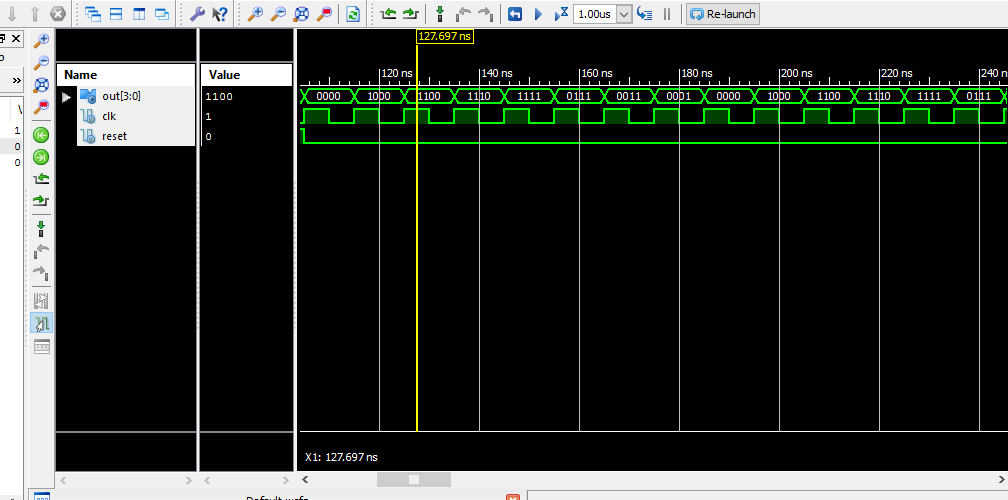


(RTL)

شبیه سازی:

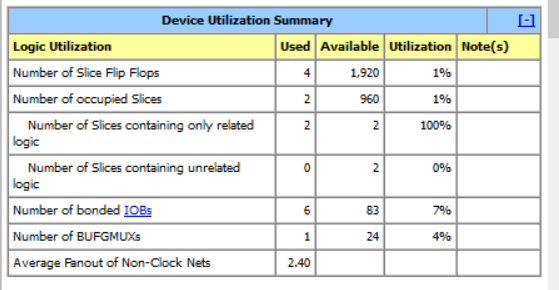


و همانطور که معلوم است در 5 واحد زمانی اول، چون مقدار خروجی ست نشده است، مقدار X نمایش داده میشود.



و بعد از آن که شمارنده شروع به کار می کند داریم که الگو های ذکر شده در صورت سوال، همگی خودشان را نشان میدهند.

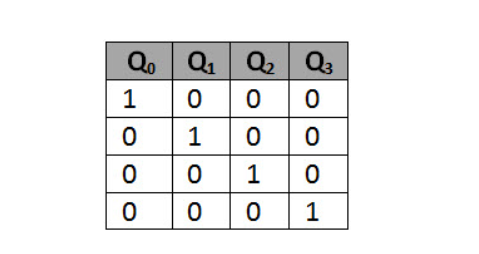
خلاصه طراحی :



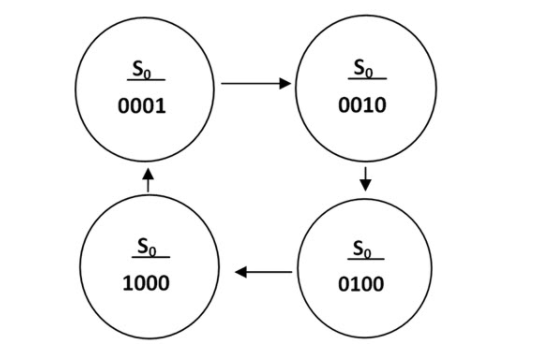
**Ring Counter**

طبق صورت پروژه, مدار تایمینگ هشت حالته مداری است که هشت خروجی دارد ( یک پایانه هشت بیتی ) که در هر لحظه فقط یکی از بیت ها میتواند یک و بقیه بیت ها صفر باشند. پس نیاز به کلاکی داریم که در هر کلاک ققط و فقط یکی از سیگنال ها روشن و بقیه خاموش باشند. با استفاده از الگوی شمارنده جانسون Ring Counter را پیاده سازی میکنیم.

جدول درستی و دیاگرام وضعیت Ring Counter به عنوان مثال چهارحالته به شکل زیر میباشد:



جدول درستی



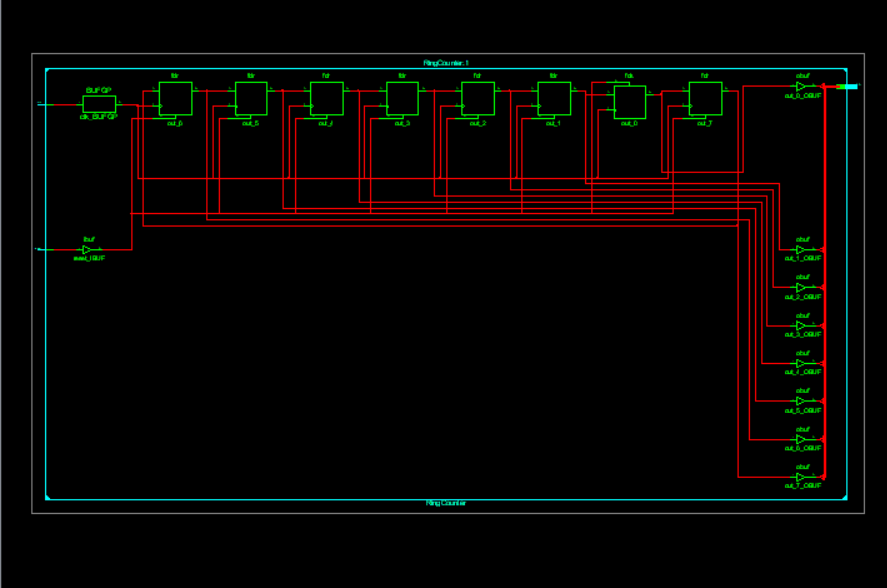
دیاگرام وضعیت

مزایا :

به تنهایی یک دیکودر میباشد چرا که در هر کلاک تنها یک بیت مقدار یک را دارد. میتواند با انواع فلیپ فلاپ ها ساخته شود.

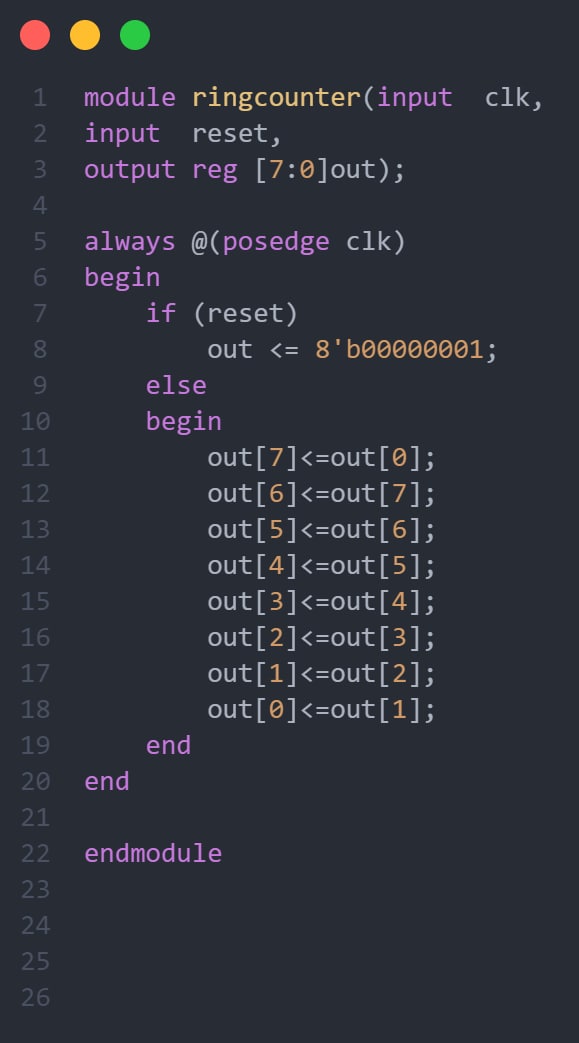
معایب :

فقط ۸ حالت از ۲۵۵ حالت ممکن برای مقدار بیت ها را میتواند تولید کند .



(RTL)

پیاده سازی :



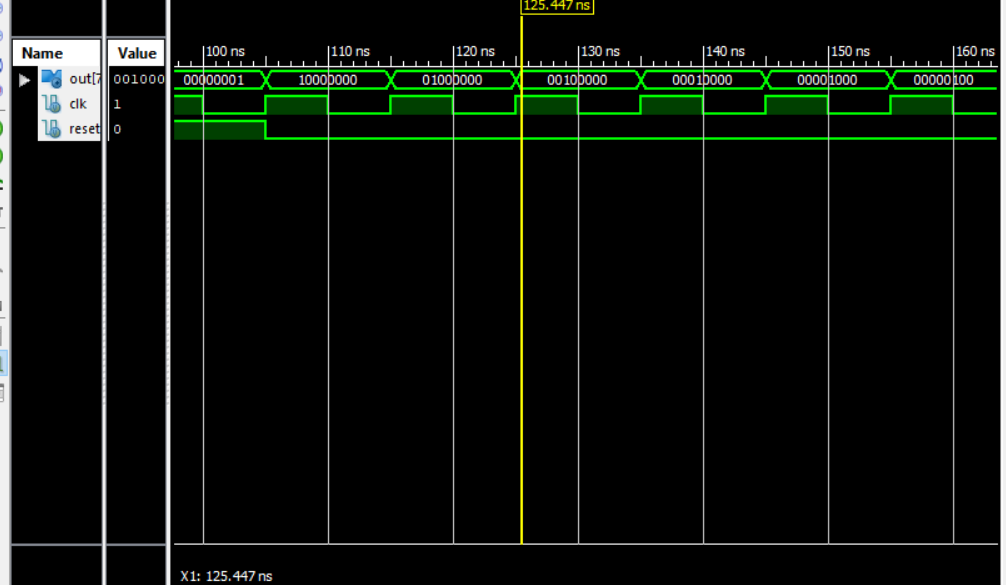
همانطور که مشاهده میشود با ریست کردن LSB خروجی برابر یک خواهد بود و همانند آنچه که در شمارنده جانسون پیاده سازی کرده ایم LSB خروجی به MSB خروجی داده میشود و یک حلقه ایجاد شده که در هر کلاک با حرکت مقدار ها به یک بیت به سمت LSB و دوباره تکرار حلقه به یک RingCounter یا مدار تایمینگ هشت حالته (هشت بیت خروجی) دست پیدا کرده ایم.

تست :



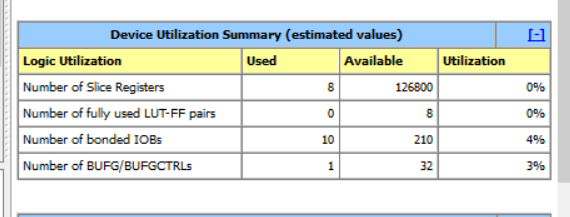
همانند شمارنده جانسون ابتدا ریست میکنیم( مقدار دهی اولیه خروجی ها) سپس در هر 5 نانوثانیه مقدار کلاک را تغییر میدهیم.

شبیه سازی :

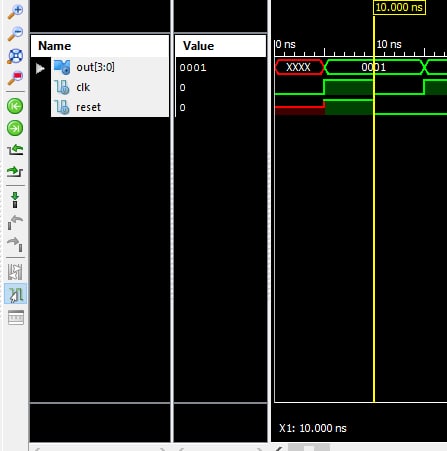
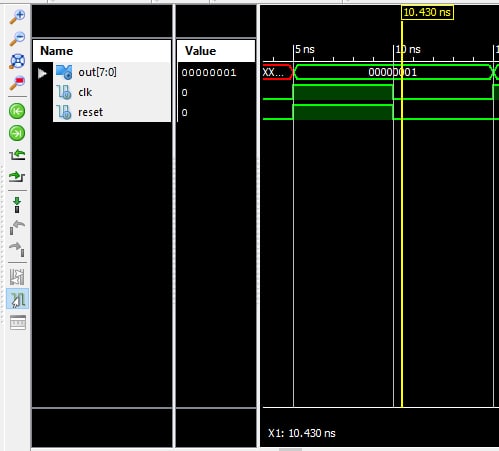


همانطور که انتظار داشتیم در شبیه سازی نیز میتوانیم مشاهده کنیم که در کلاک تنها یک بیت برابر یک است و مانند یک شیفت دهنده عمل میکند.

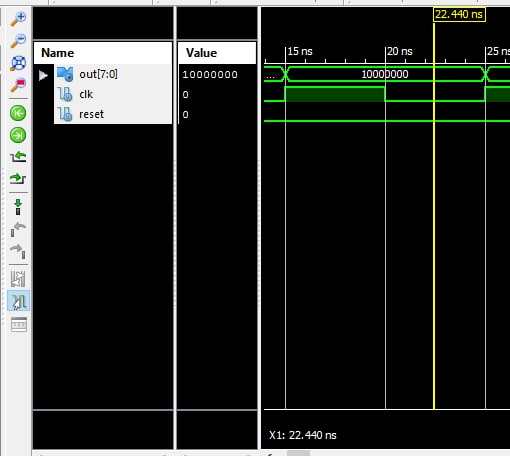
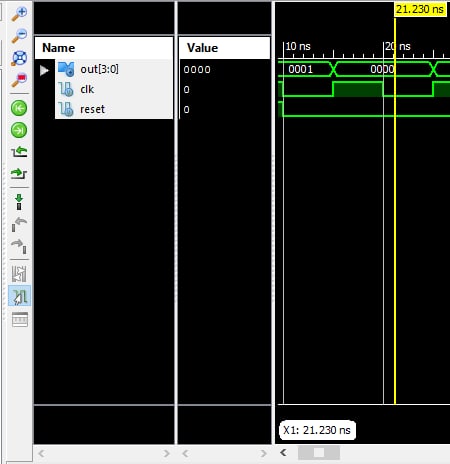
خلاصه طراحی :



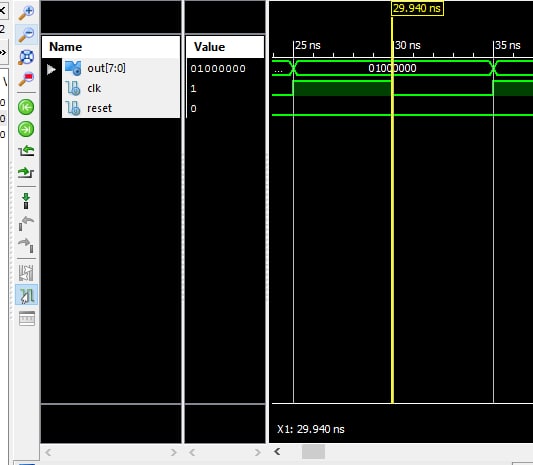
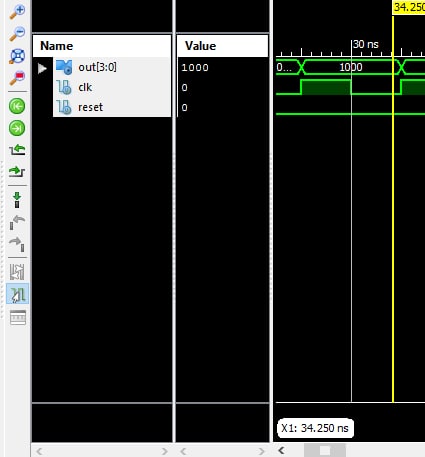
در پایان اگر شبیه سازی Johnson Counter و Ring counter را متناظرا در کنار هم قرار دهیم به راحتی میتوانیم تعداد حرکات Johnson Counter را با استفاده از مدار تایمینگ Ring Counter به دست آوریم.

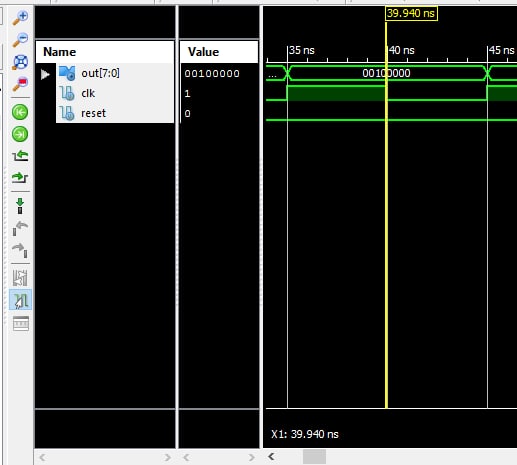
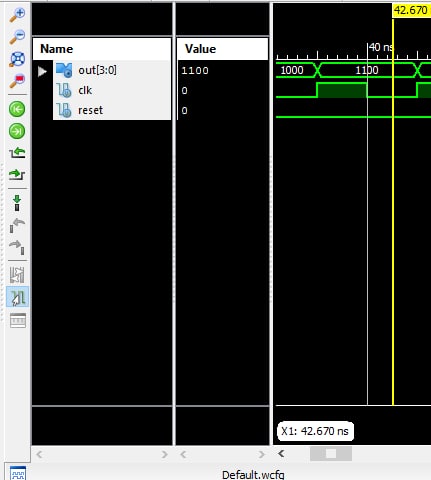
وضعیت اول



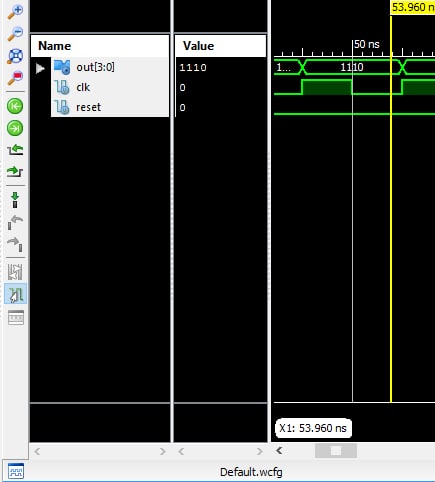
وضعیت دوم



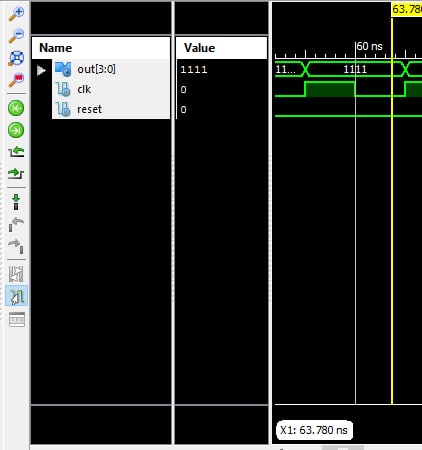
وضعیت سوم



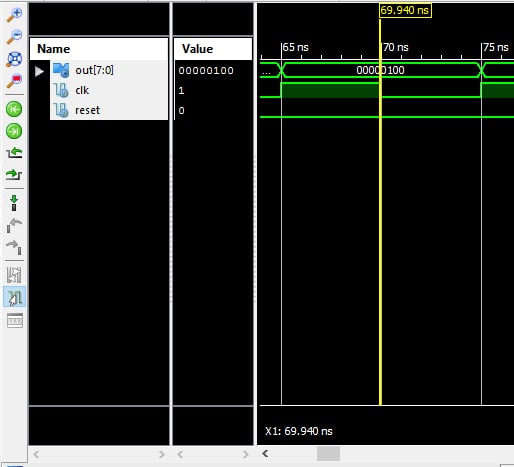
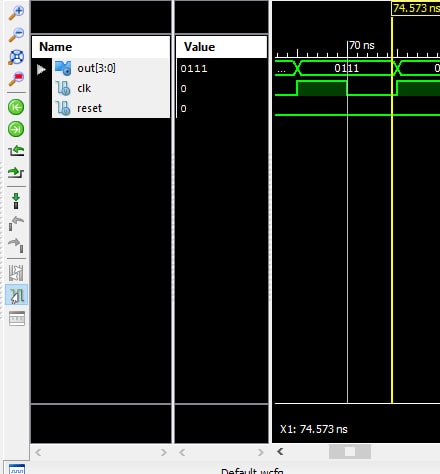
وضعیت چهارم



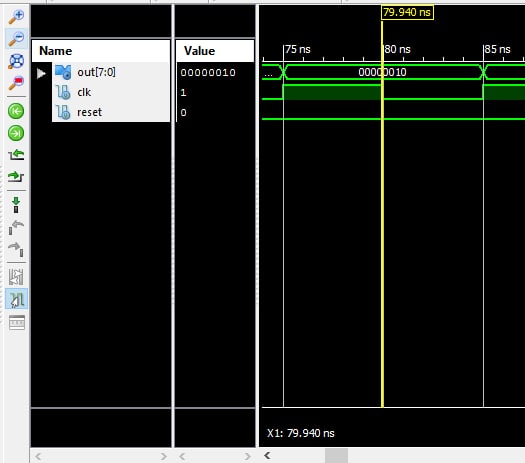
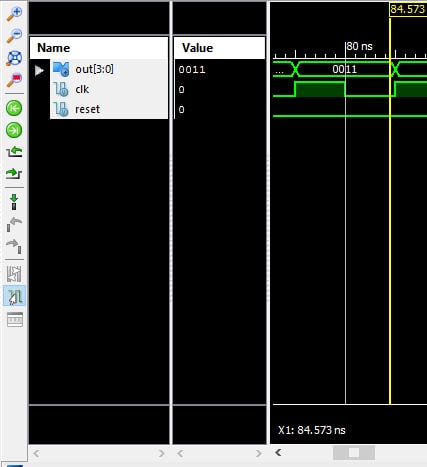
وضعیت پنجم



وضعیت ششم



وضعیت هفتم



وضعیت هشتم

همانطور که میبینم در هشت مرحله حلقه کامل میشود و تکرار میشود پس به درستی شبیه سازی و طراحی انجام شده است.