

دانشگاه صنعتي امیرکبیر  
(پلی­تکنیک تهران)

دانشكده برق

گزارشکار آزمایشگاه مدار منطقی

پروژه درس

نگارش

پارسا محمّدی 9923121

استاد درس

مهندس مهرابادی

تیر 1402

فهرست

[ساختار پروژه 3](#_Toc138469329)

[تشریح کد ALU 6](#_Toc138469330)

[تست بنچ 8](#_Toc138469331)

# ساختار پروژه

در این پروژه یک ALU طبق خواسته صورت پروژه ساخته شده است و در فایل ALU قرار دارد. و همچنین یک تست بنچ برای بررسی درستی عملکرد آن طراحی شده است.

در این پروژه نیاز به استفاده از چندین کد دستور مختلف برای مشخص کردن ورودی و خروجی می‌باشد. جدول کد ها به صورت زیر می باشند.

|  |  |
| --- | --- |
| Op code | Operation |
| 0000 | SUB |
| 0001 | AND |
| 0010 | ADD |
| 0011 | OR |
| 0100 | MUL |
| 0101 | XOR |
| 0110 | Shift Right |
| 0111 | Shift Left |
| 1000 | Write |
| 1001 | Read |

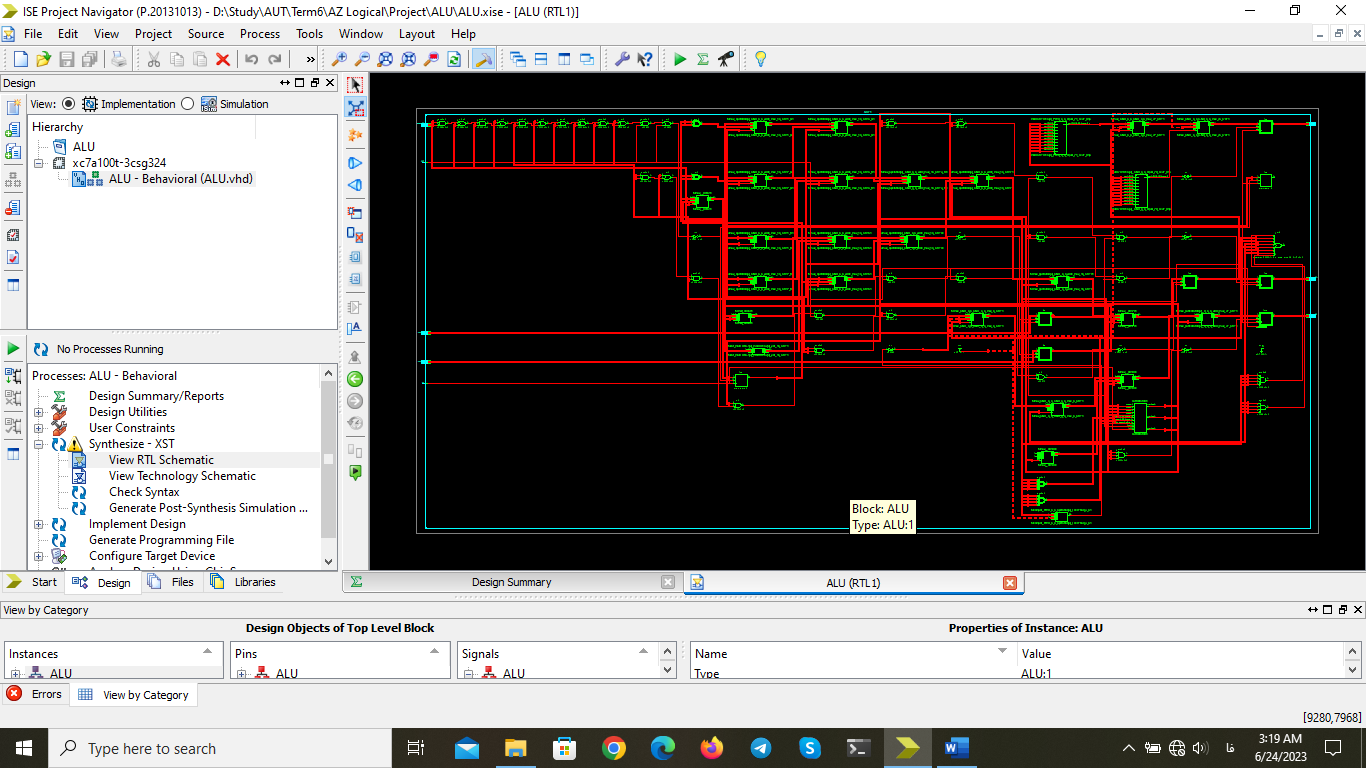
با توجه به اینکه این واحد پردازش باید 10 دستور مختلف را پردازش کند به همین منظور از دستورات 4 بیتی برای مپ کردن اپ کد و عملیات استفاده شده است. و از 0 شروع شده و تا 9 می روند.

این کد ها به ورودی ALU داده می‌شوند تا ALU بداند در ادامه چه کاری را بر روی ورودی هایش انجام دهد.

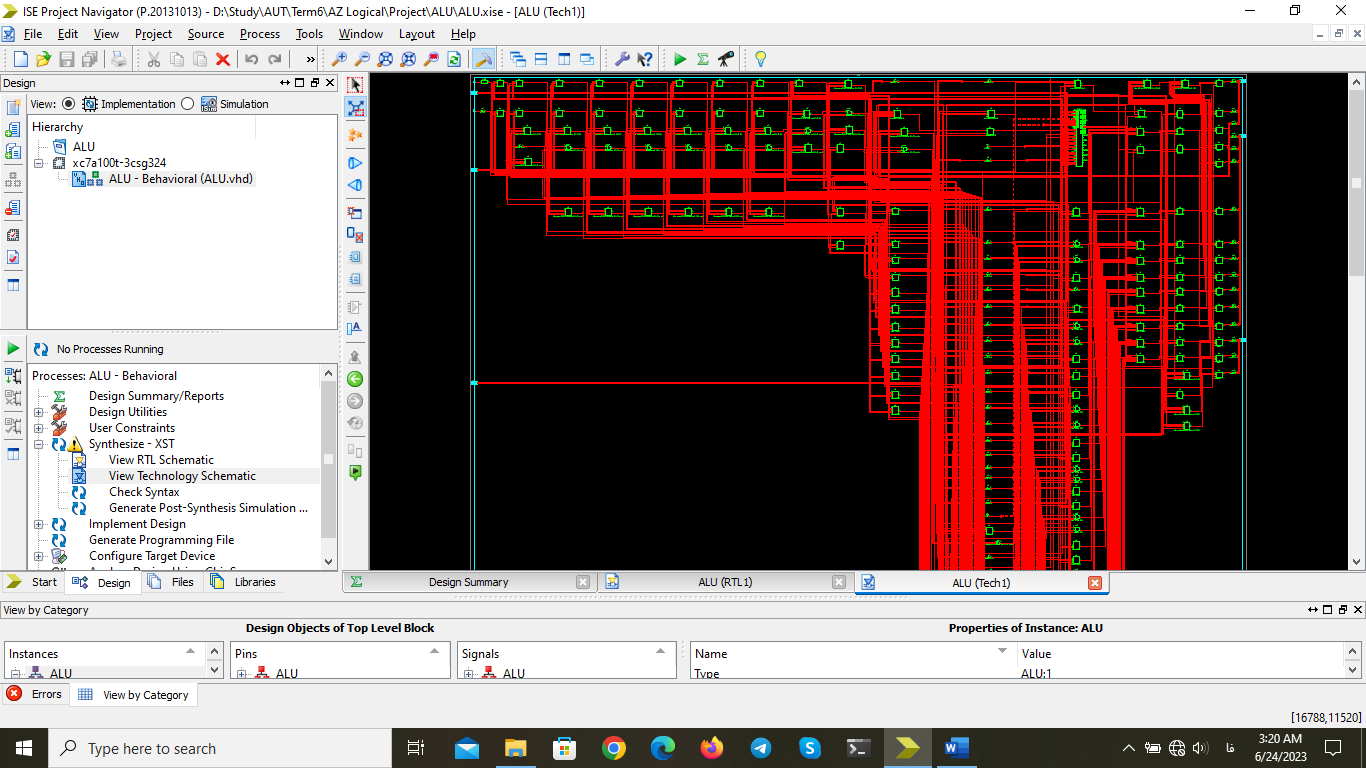
|  |  |
| --- | --- |
| code | State |
| 000 | Ready |
| 001 | In Prosses |
| 010 | Writing Process |
| 011 | Reading Process |
| 100 | In output |

جدول بالا کد های متناظر با هر کدام از استیت های قطعه را مشخص می‌کند چون در صورت پروژه آمده است که این قطعه باید استیت خود را مشخص کند این قطعه نیز در هر استیتی که باشد آن را خروجی می‌دهد و کاربر را از استیتی که در آن قرار دارد آگاه می‌کند. خروجی های 3 بیتی برای مشخص کردن استیت استفاده می‌شود.

|  |  |
| --- | --- |
| code | Flag |
| 000 | None |
| 001 | Z(zero) |
| 010 | C(carry) |
| 011 | N(negative) |
| 100 | V(overflow) |

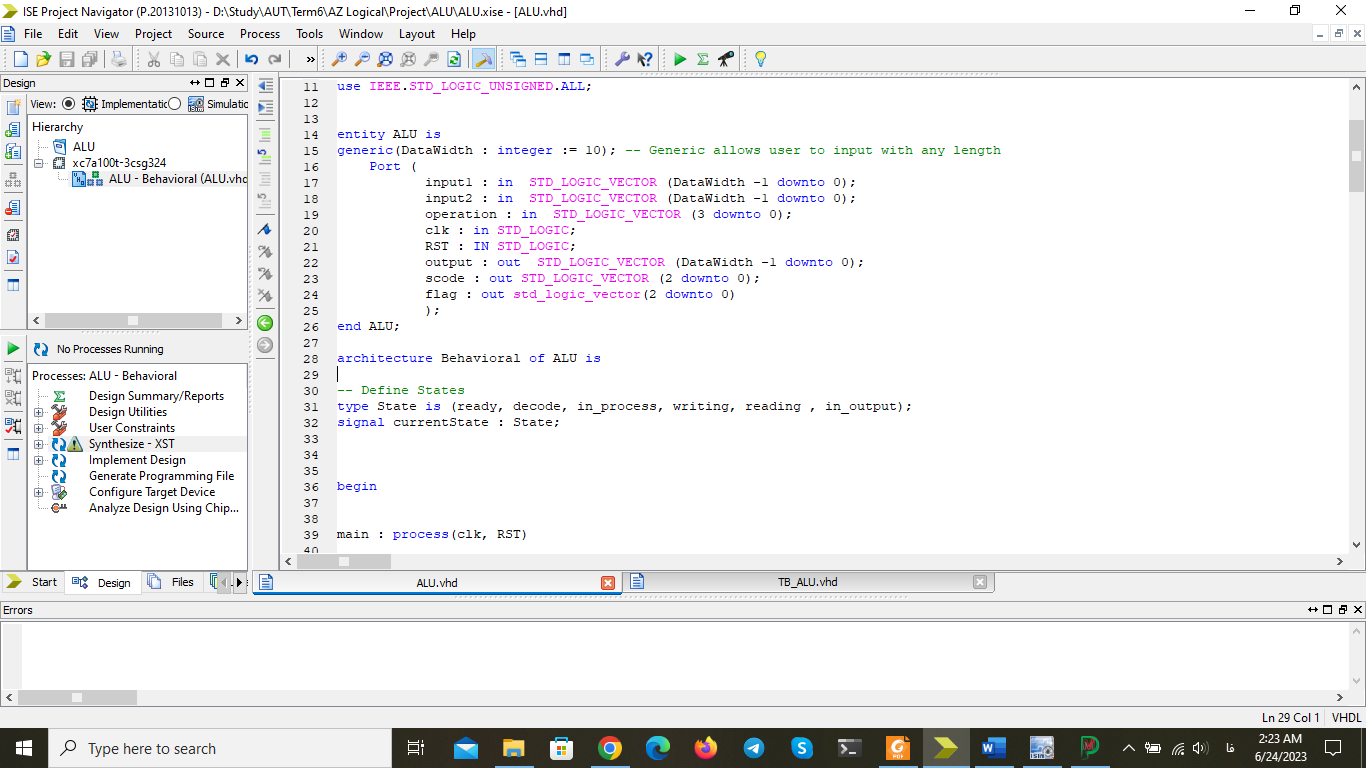
جدول بالا فلگ ها را مشخص می‌کند. با توجه به اینکه این قطعه باید چند فلگ را بعد از انجام محاسبات مشخص کند به همین دلیل جدول بالا طراحی شده است و این قطعه در صورت وقوع هرکدام خروجی متناظر با آن را می دهد. حالت None برای زمانی است که هیچ کدام از فلگ ها فعال نمی‌باشند.

تصویر RTL ماژول

بخشی از شمای تکنولوژی ماژول

# تشریح کد ALU

در ابتدا کد بخش اینتیتی کد می‌باشد و با توجه به خواست پروژه مبنی بر اینکه این قطعه بتواند ورودی با هر تعداد بیت را پردازش کند از generic استفاده شده است که به کاربر اجازه می دهد ورودی را با هر مقدار دلخواه به ماژول بدهد.

در ادامه هم ورودی ها و خروجی های ماژول مشخص شده اند.

در این قسمت همانطور که در صورت پروژه آمده است چندین استیت تعریف شده است که هرکدام وضعیت ماژول را مشخص میکند

* Ready

در این استیت ماشین آماده دریافت ورودی های جدید می‌باشد و ورودی ها را بافر میکند

* Decode

در این قسمت آپ کد آنالیز می‌شود و مشخص می‌شود که نوع دستور جیست و در ادامه به کدام استیت برود.

* In\_process

در این بخش محاسبات منطقی و ریاضی انجام می‌شوند

* Writing

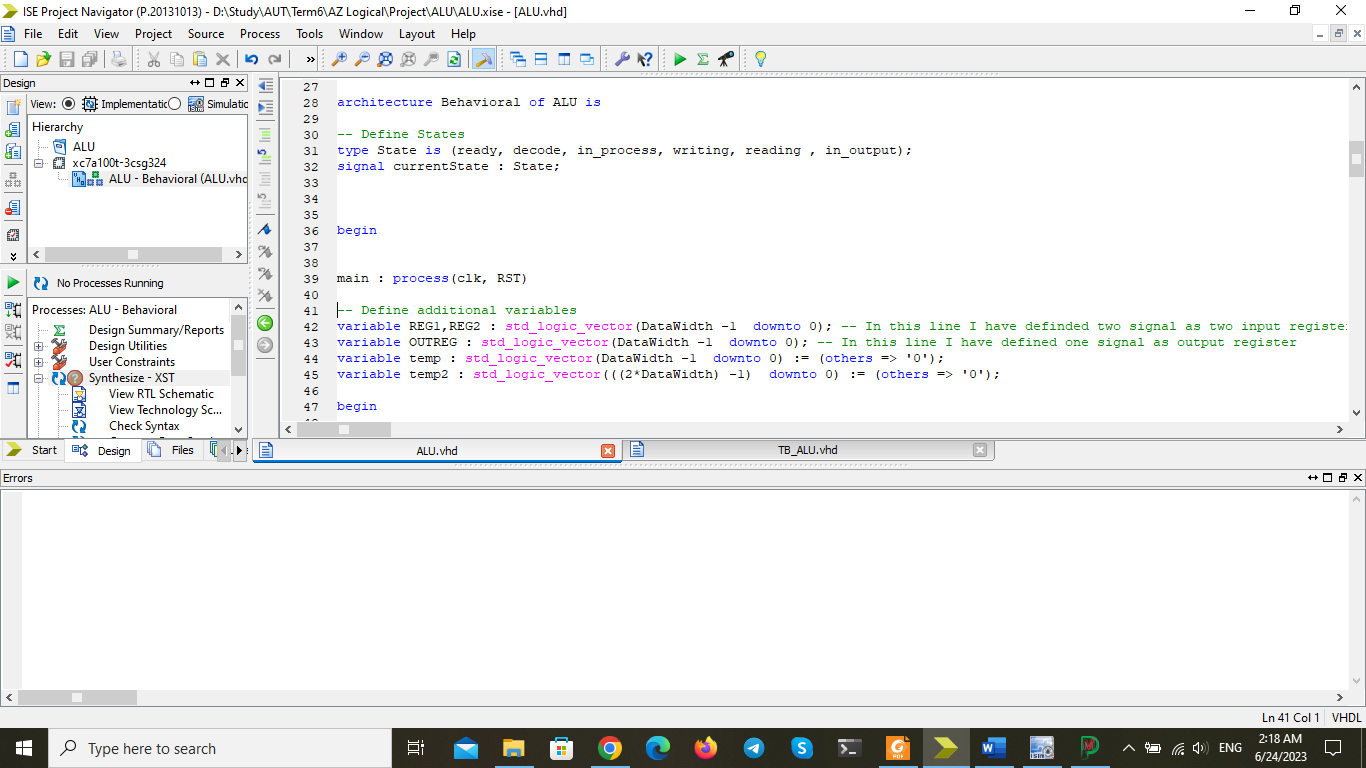
در این بخش در ورودی در حافظه نوشته می‌شود

* Reading

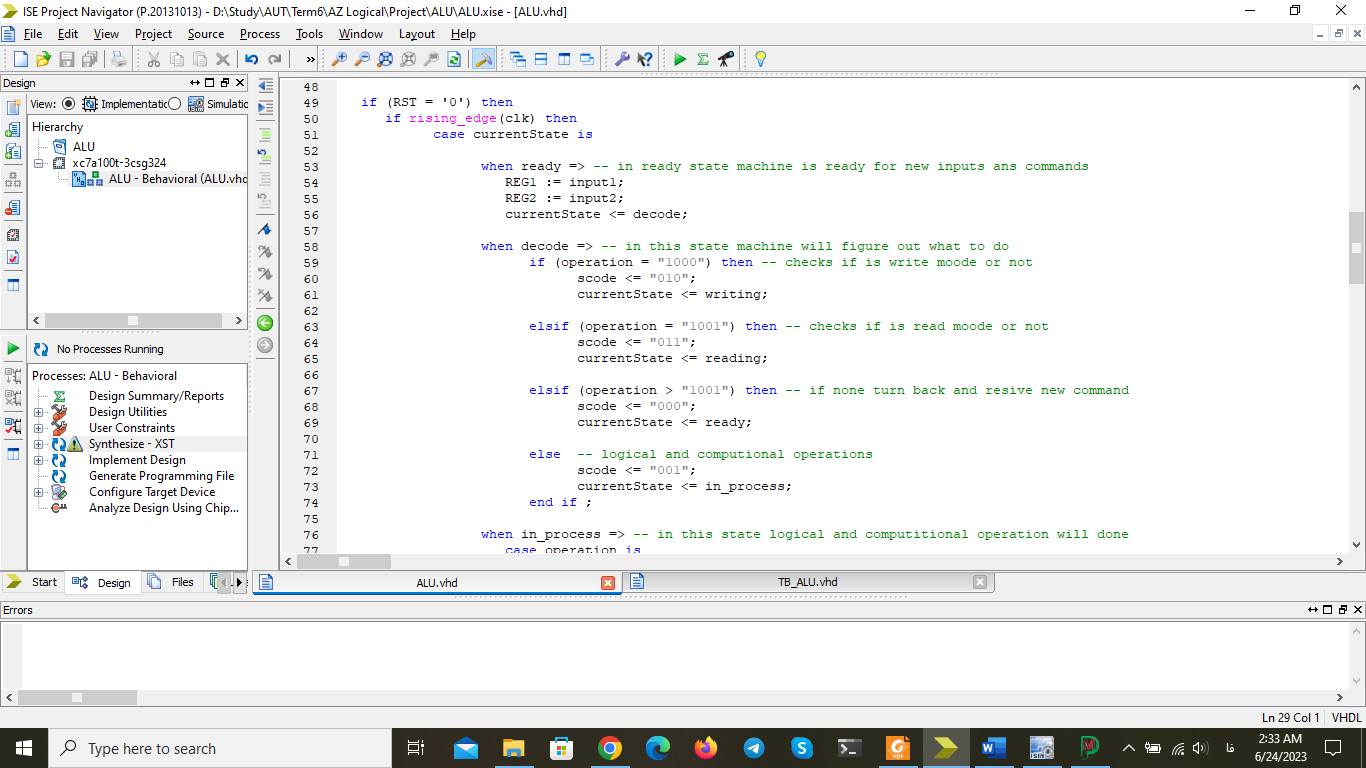
در این بخش حافظه خوانده و به خروجی داده می‌شود

* In\_OUTPUT

در این استیت حافظه به خروجی داده می‌شود.



با توجه به اینکه این ماژول حساس به کلاک می‌باشد یک پروسس برای آن تعرف شده است که به هر بار تغییر کلاک تغییرات اعمال شود این استیت به تغییرات ریست نیز حساس است.

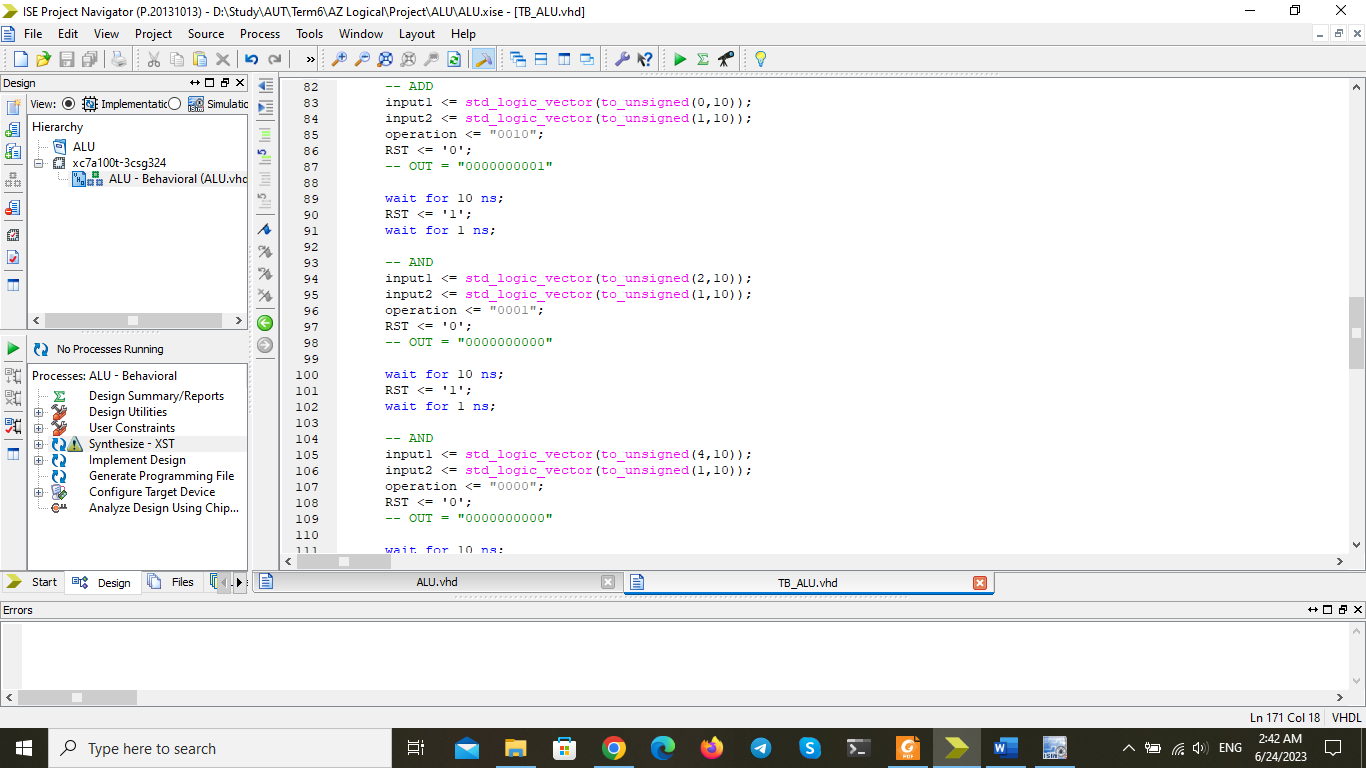
در ادامه چندین متغییر تعرف کرده ام. دو خط اول ریجیستر های ورودی و خروجی را مشخص میکنند و برای مشخص کردن آنها از متغییر استفاده شده است زیرا این متغییر ها میتوانند مقدار ورودی و خروجی ها را در خود ذخیره کنند و از آنها به عنوان بافر استفاده شده است.

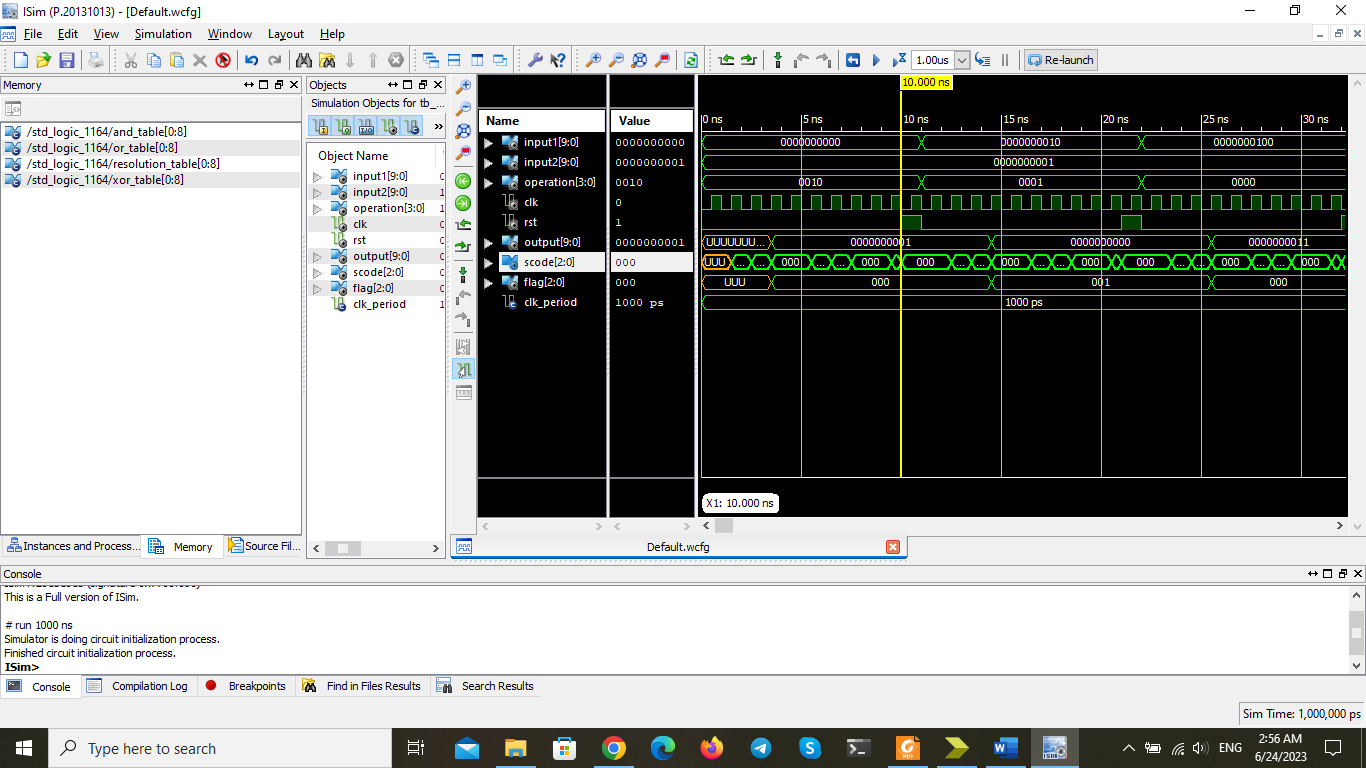
در ادامه یک ریست آسکرون برای ماژول طراحی شده است و همچنین این ماژول به لبه بالا رونده حساس می باشد.

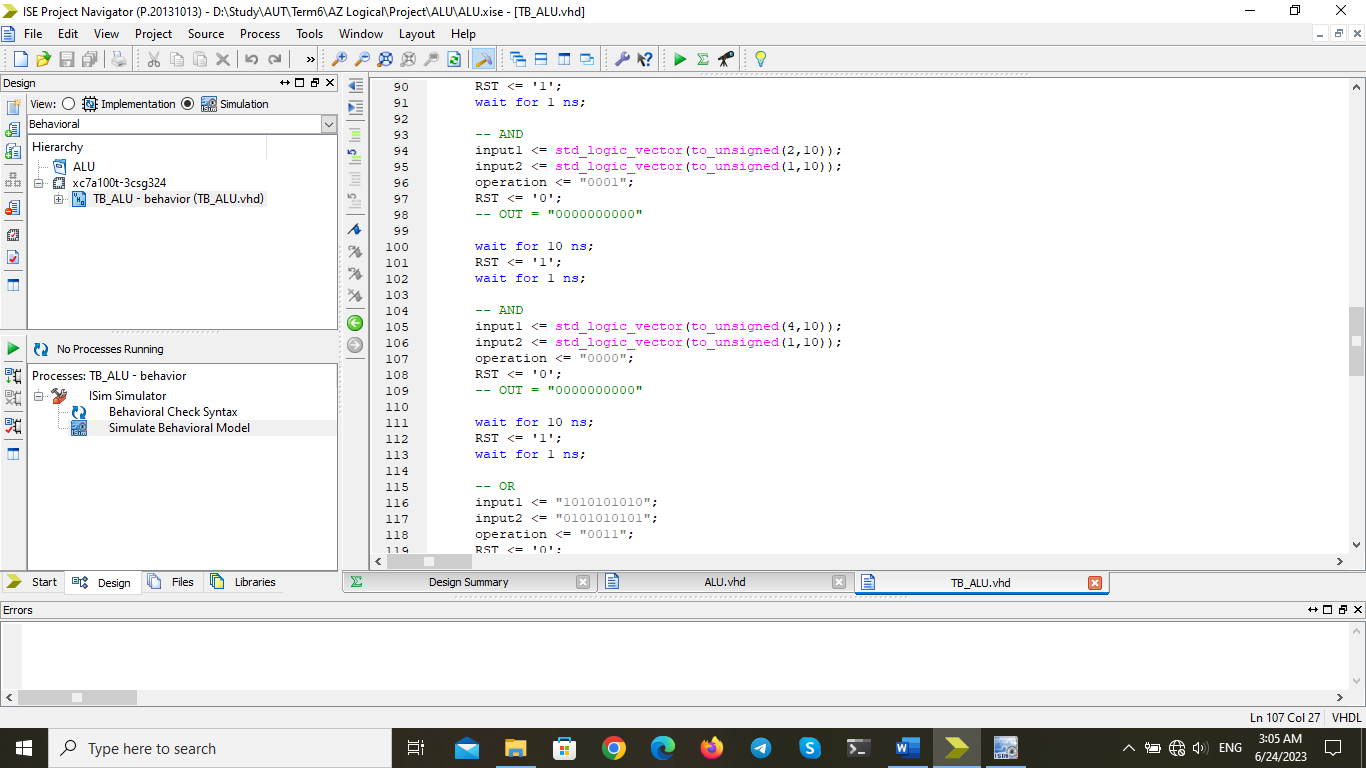
ادامه توضیحات عملکرد کد در خود کد کامنت شده است.

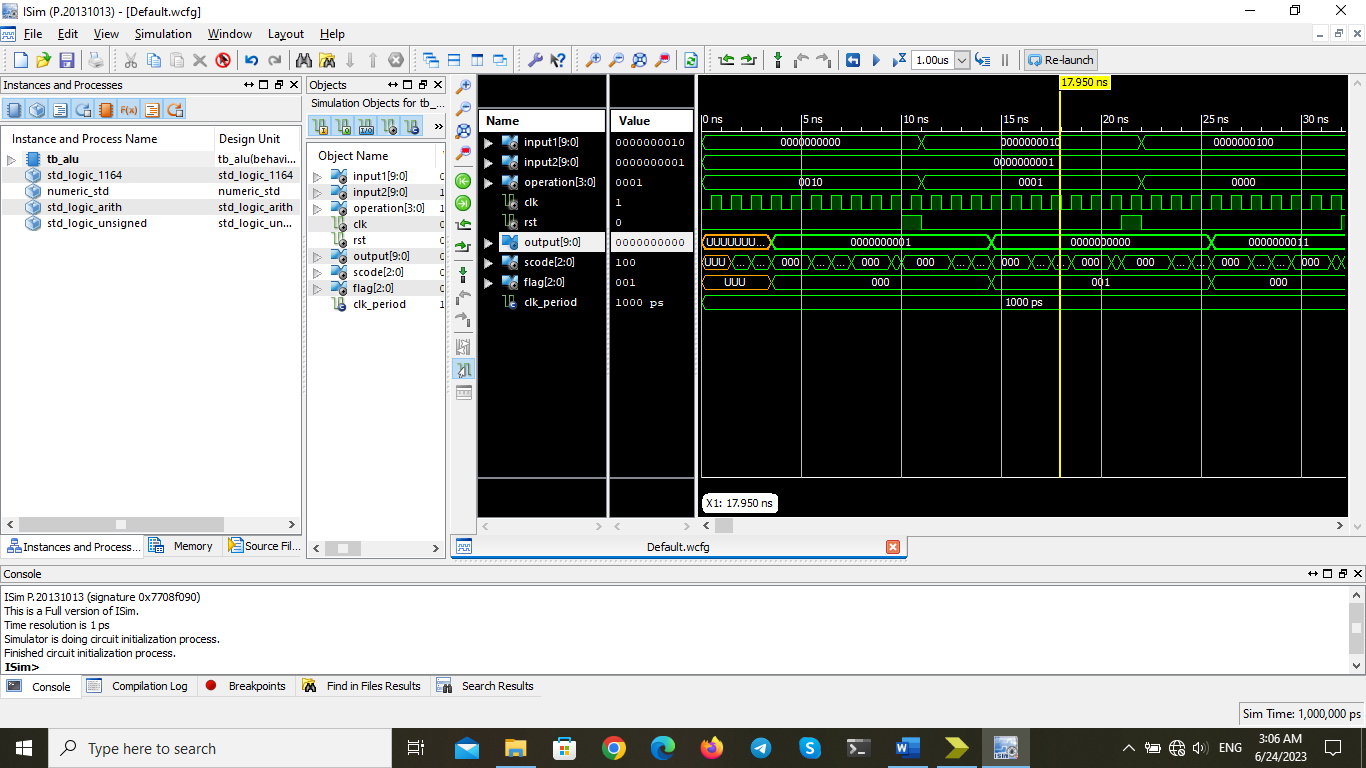
# تست بنچ

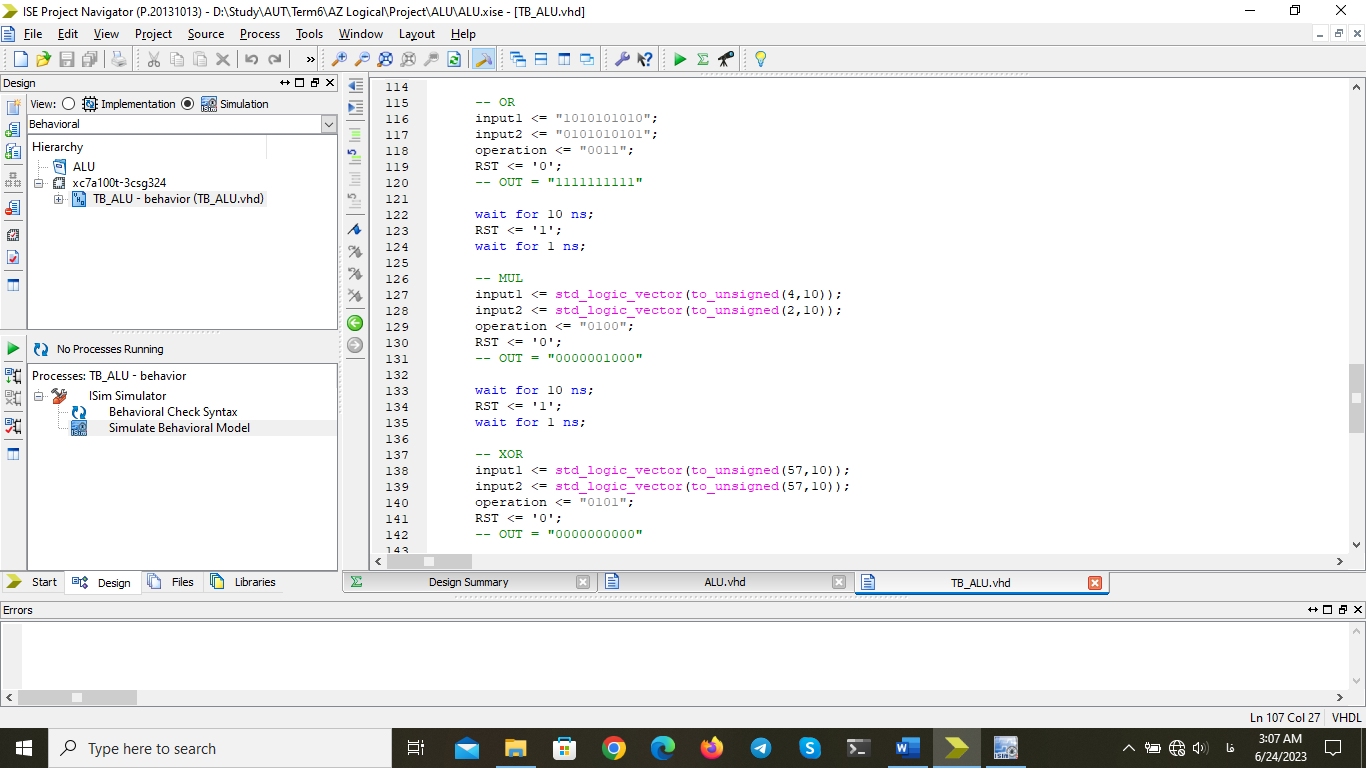
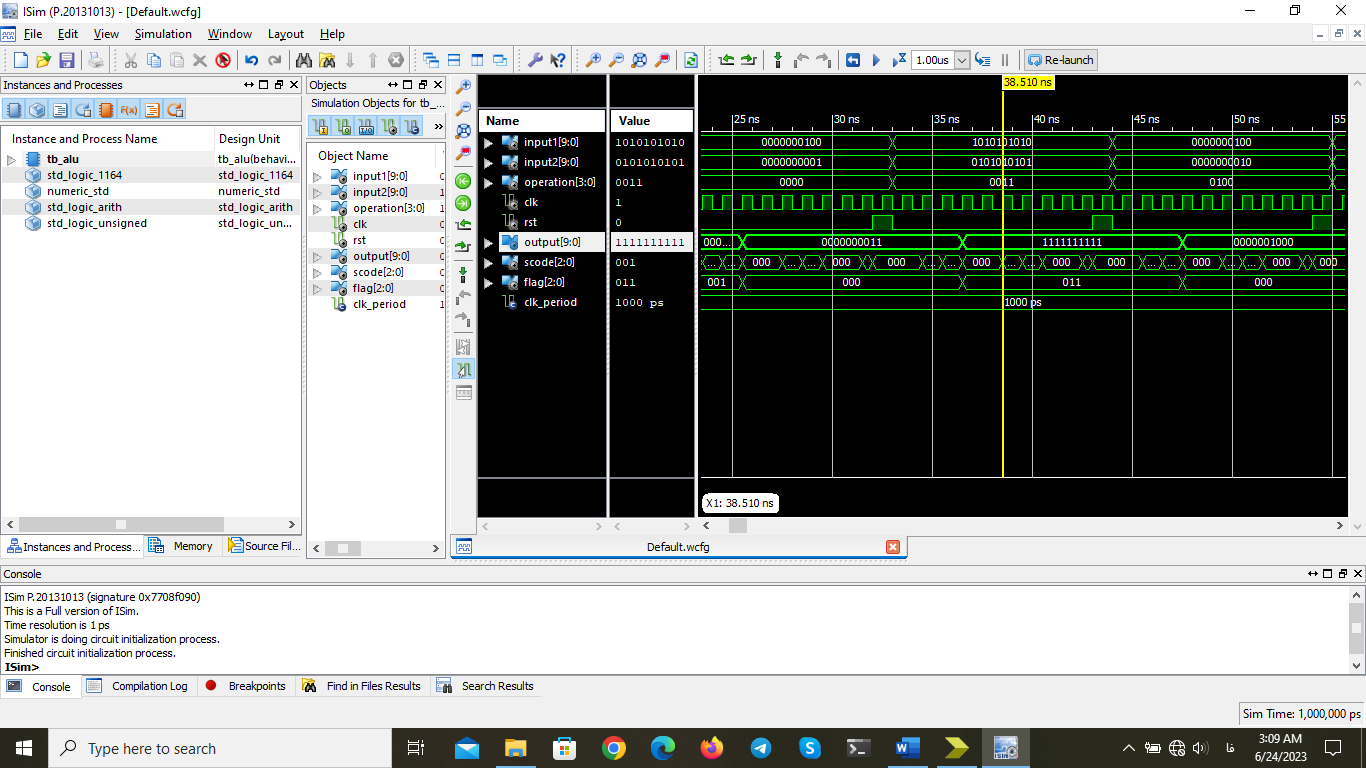
برای این ماژول یک تست بنچ طراحی شده است که بخش های مختلف آن را تست می‌کند.





همطور که مشخص است خروجی مجموع دو ورودی می باشد و درست است. همین طور که مشخص است زمانی که ریست آسکرون فعال می‌شود در همان زمان دستگاه ریست شده و از استیت اول شروع به کار می‌کند.



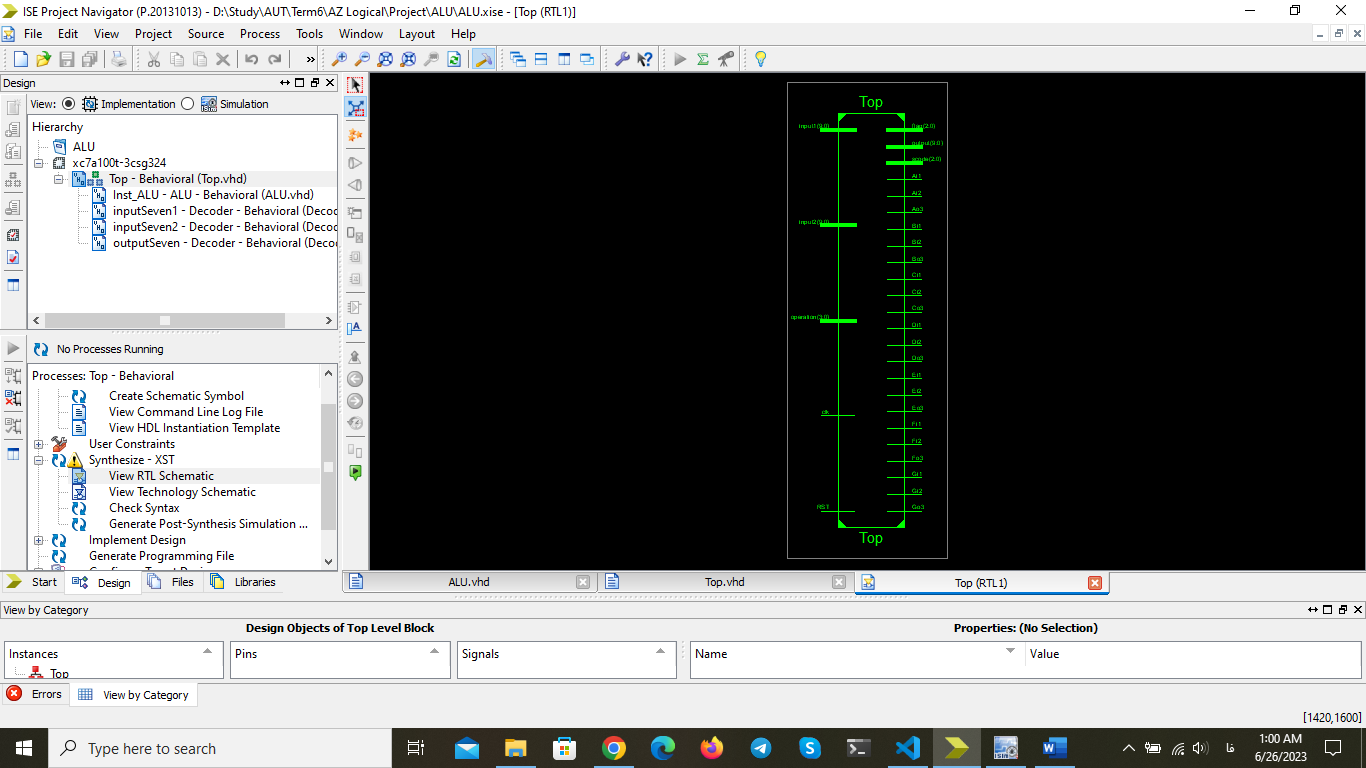
 همان طور ملاحظه می شود حاصل And 1 و 4 صفر شده است.

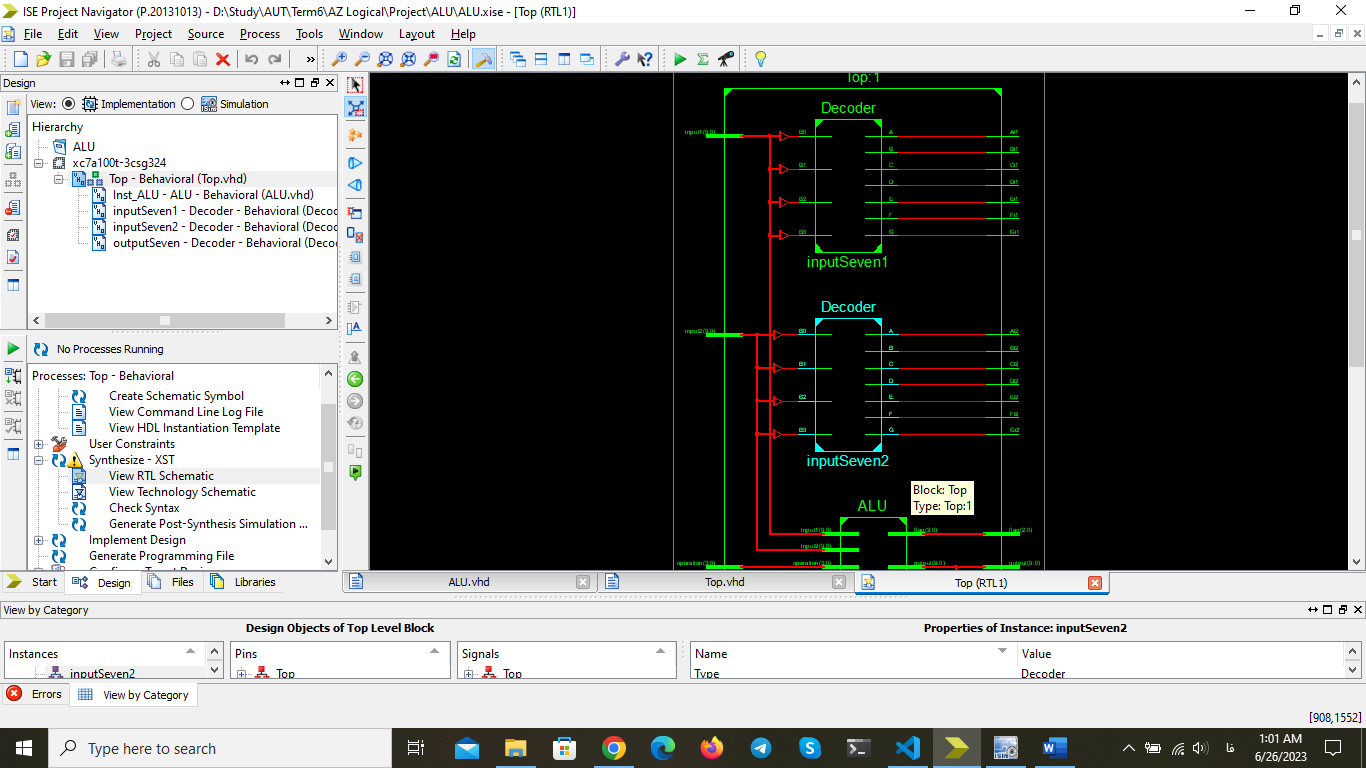
همان طور که مشخص است حاصل or این دو عدد باهم همه بیت هایسش 1 می شود.

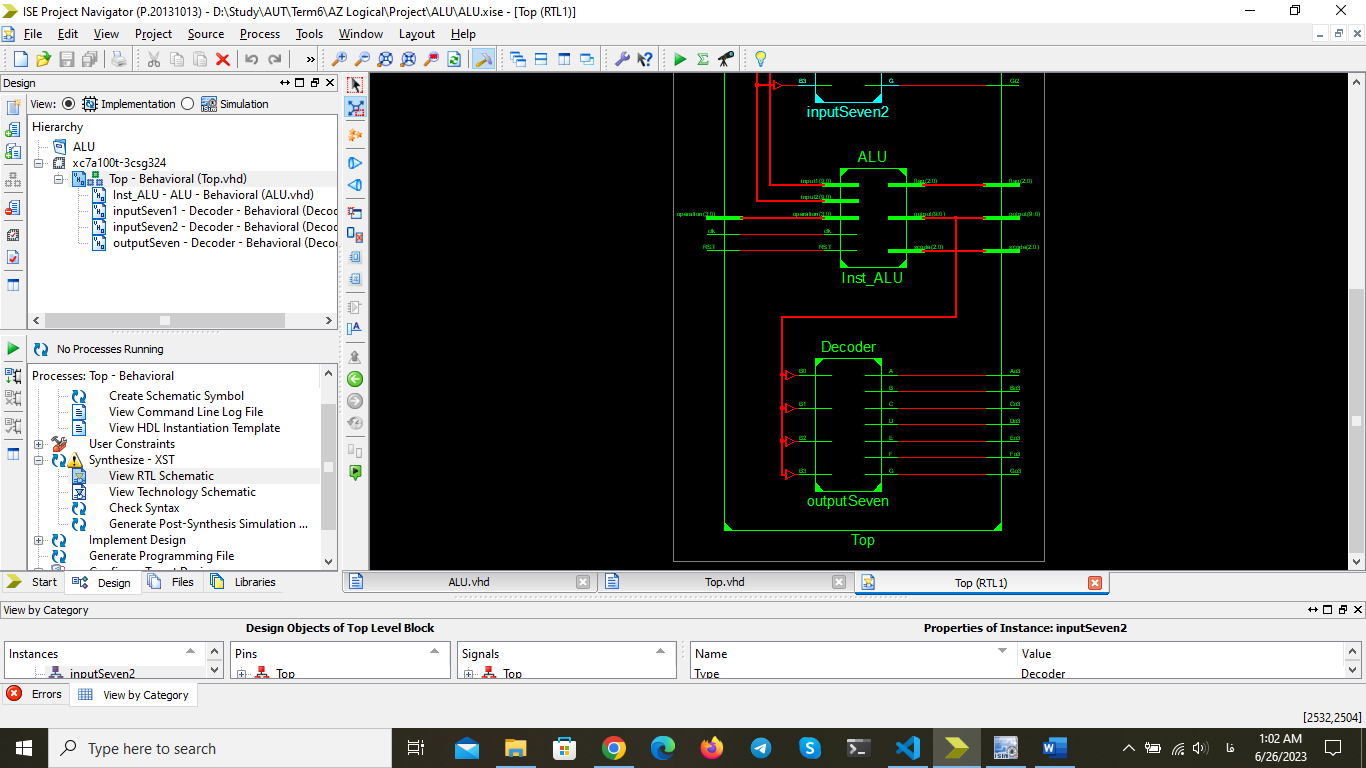
بقیه تست ها هم به همین ترتیب درست اند.

# اضافه کردن بخش امتیازی

در بخش امتیازی یک فایل Top تعریف شده است که هم شامل alu و هم شامل 7segment ها می‌باشد. و آنها را طبق صورت پروژه به هم وصل می‌کنیم.

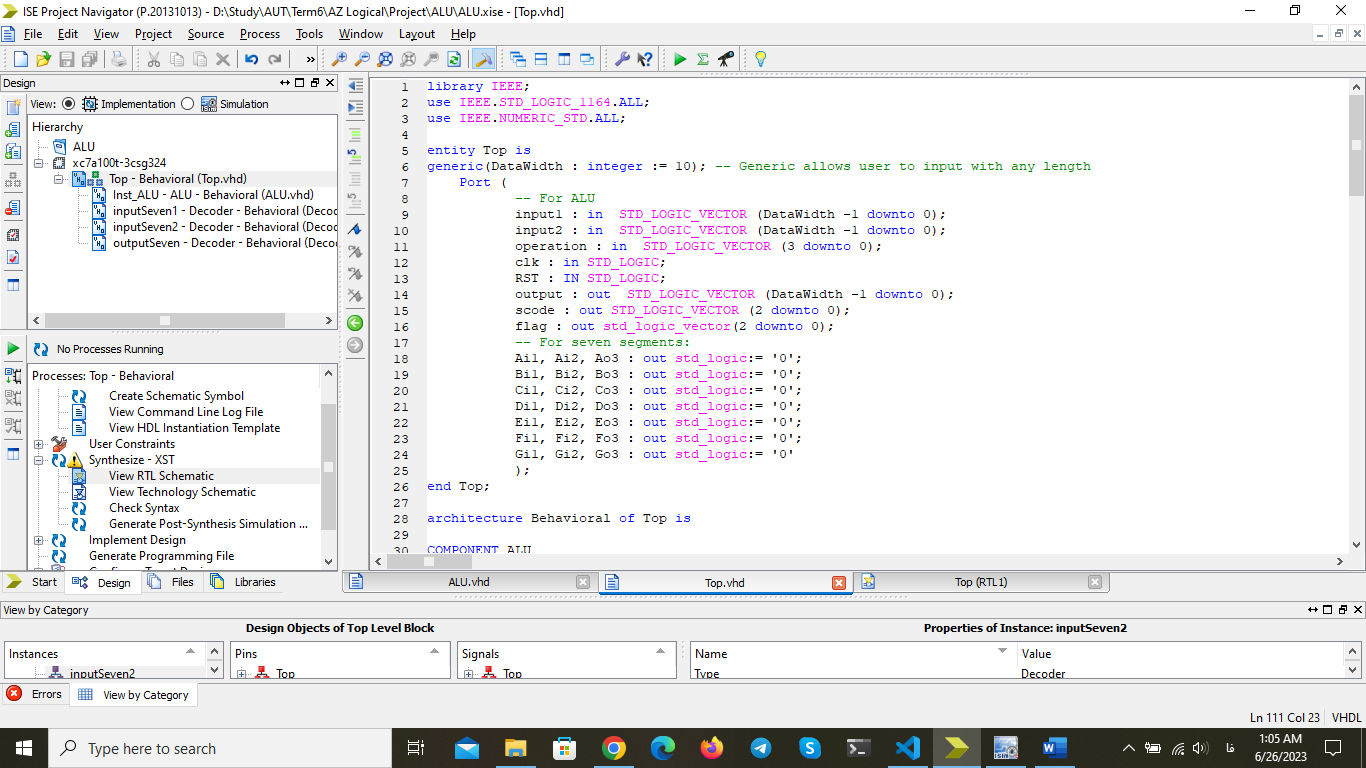
RTL کد تاپ





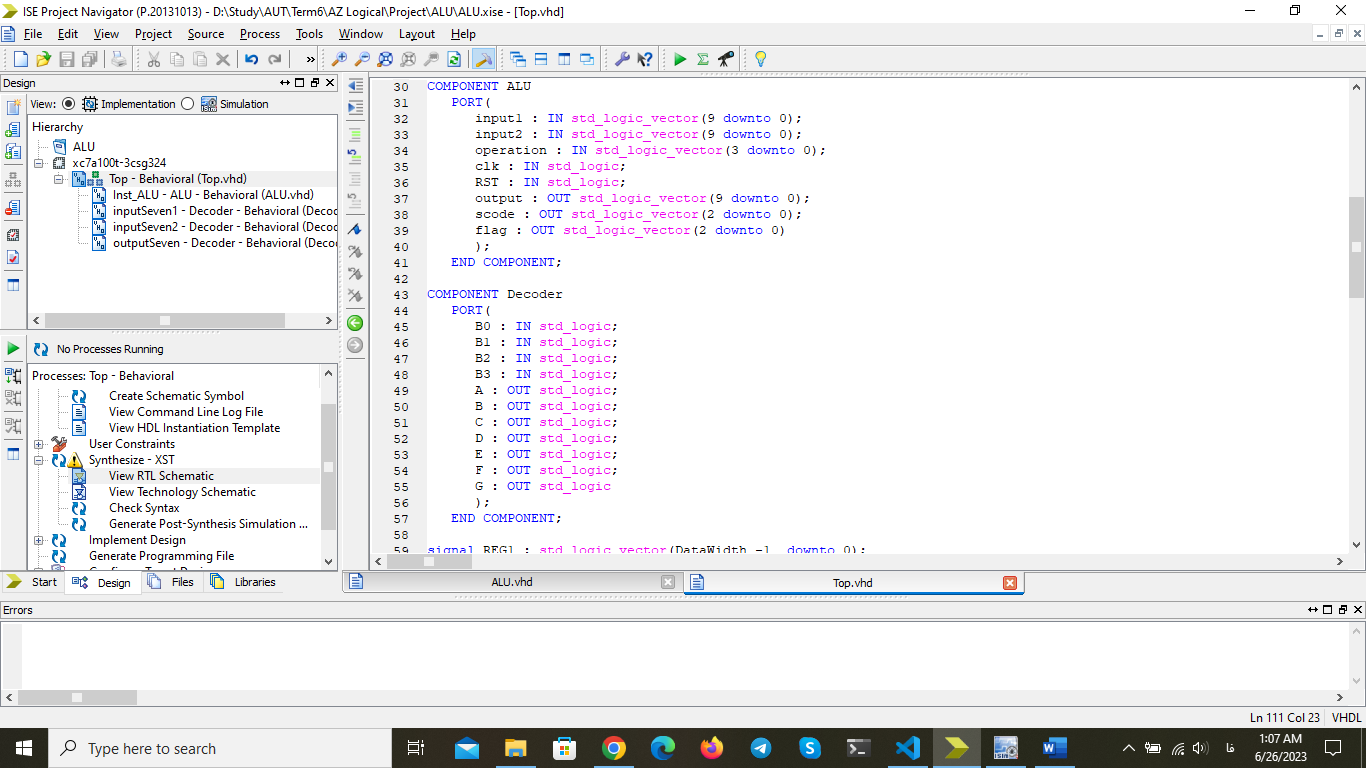
نقشه RTL مدار هرکدام از ماژول ها در بخش های قبل نمایش داده شده است.

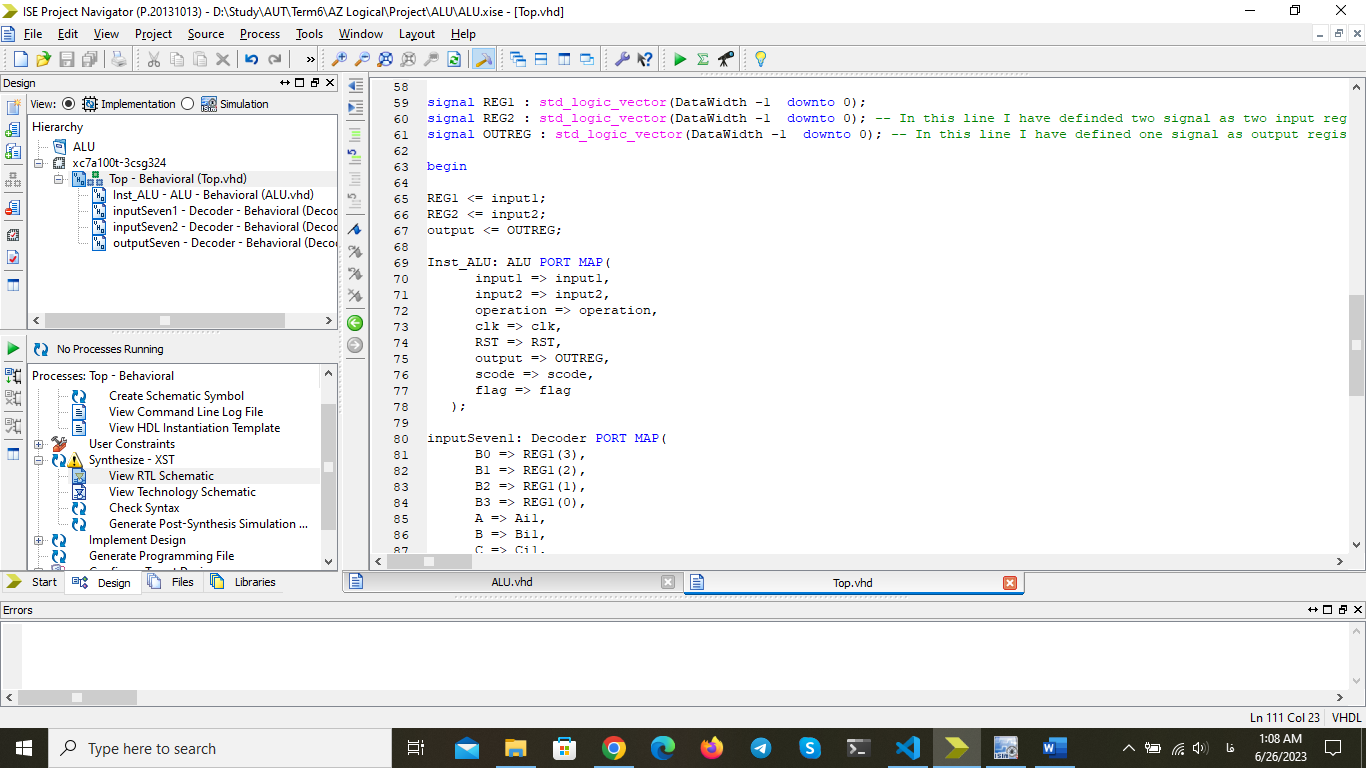
از دوتا دیکدر BCD به 7Segment برای ورودی و یکی هم برای خروجی در نظر گرفته شده است طبق خواست صورت پروژه.



در این پروژه تمام ورودی های مورد نیاز ALU به علاوه خروجی های جدید که برای 7segment است نیز نوشته شده است.

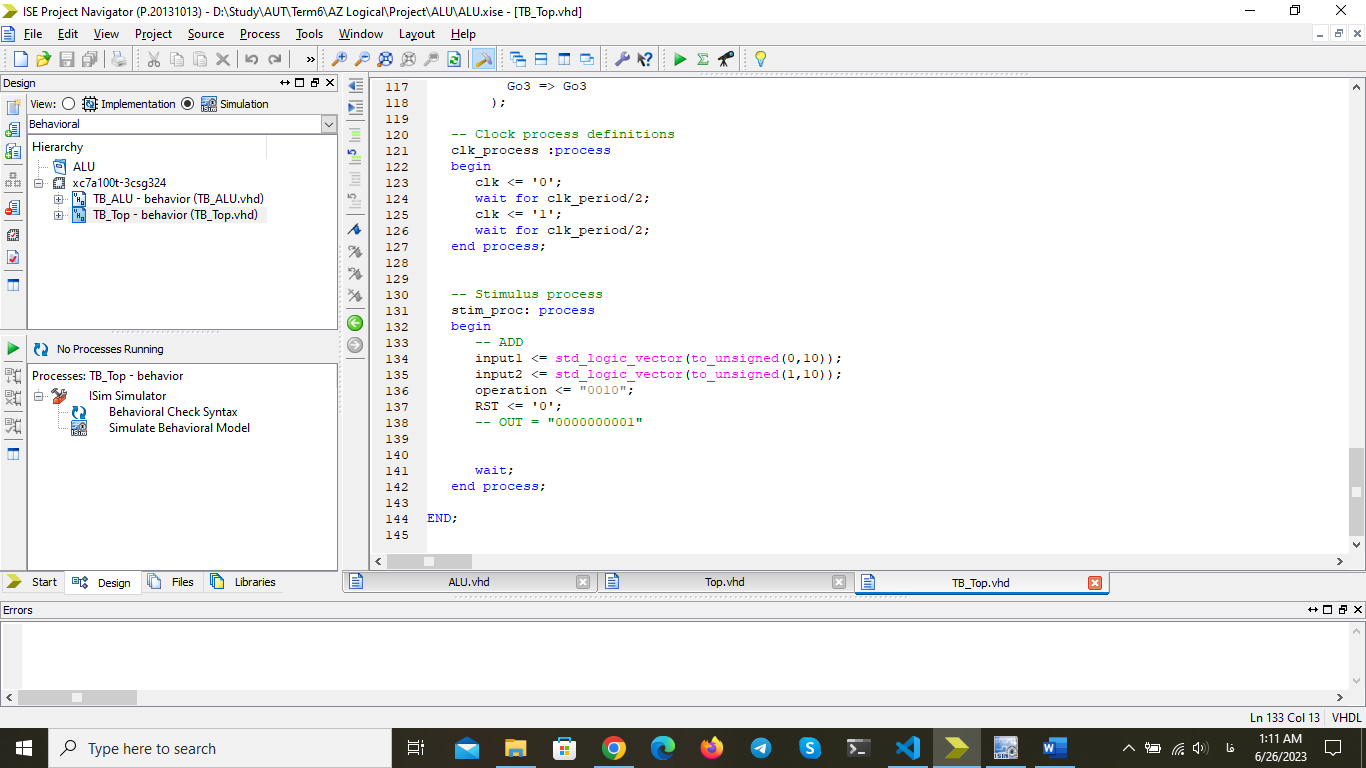
هم کدام از A تا G را برای هرکدام از خروجی ها مینویسیم تا خروجی های دیکدر ها نمایش داده شود.

ابتدا دیکدر و ALU اضافه مکنیم به صورت کامپوننت

سیگنال های تعریف شده کارایی رجیستر ها را دارد و مقدار ورودی ور خروجی را در خود بافرر می‌کند.

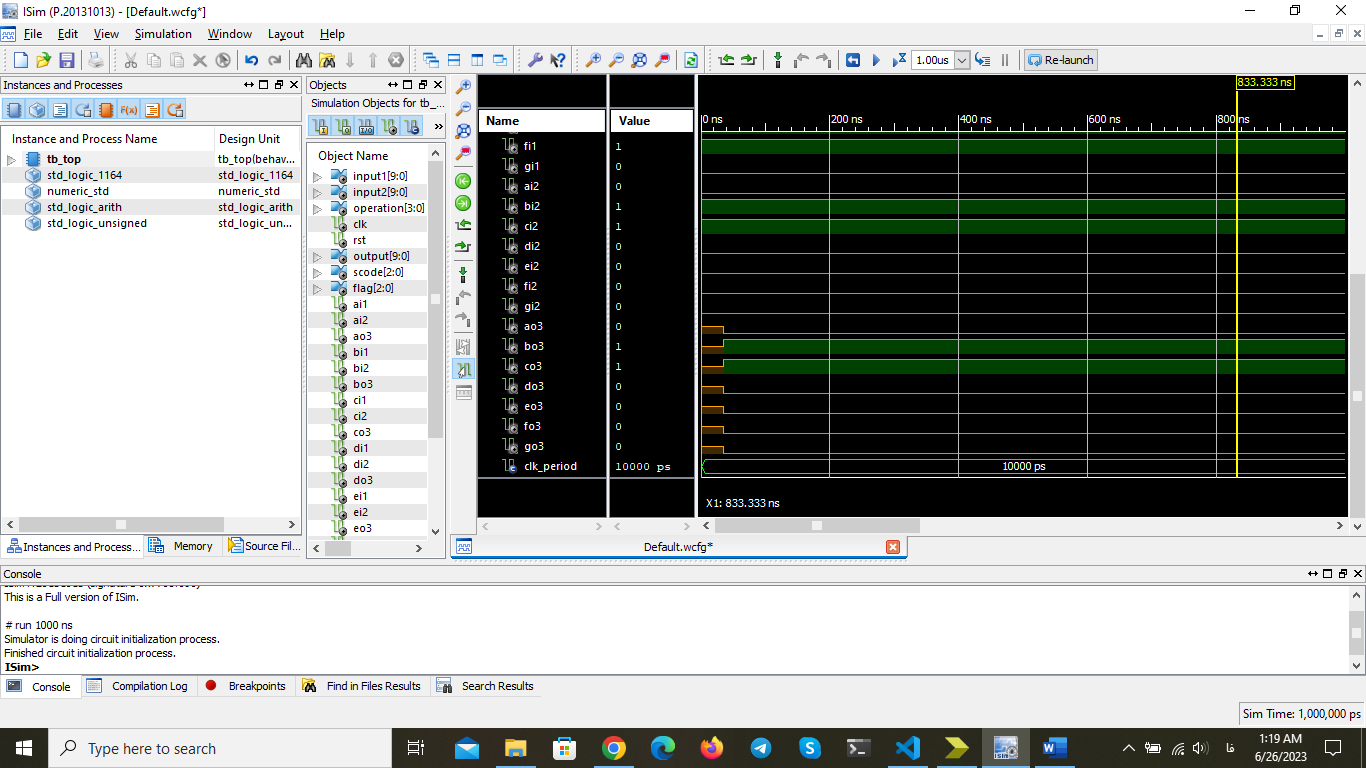
درادامه ماژول ها پورت مپ می‌شوند.

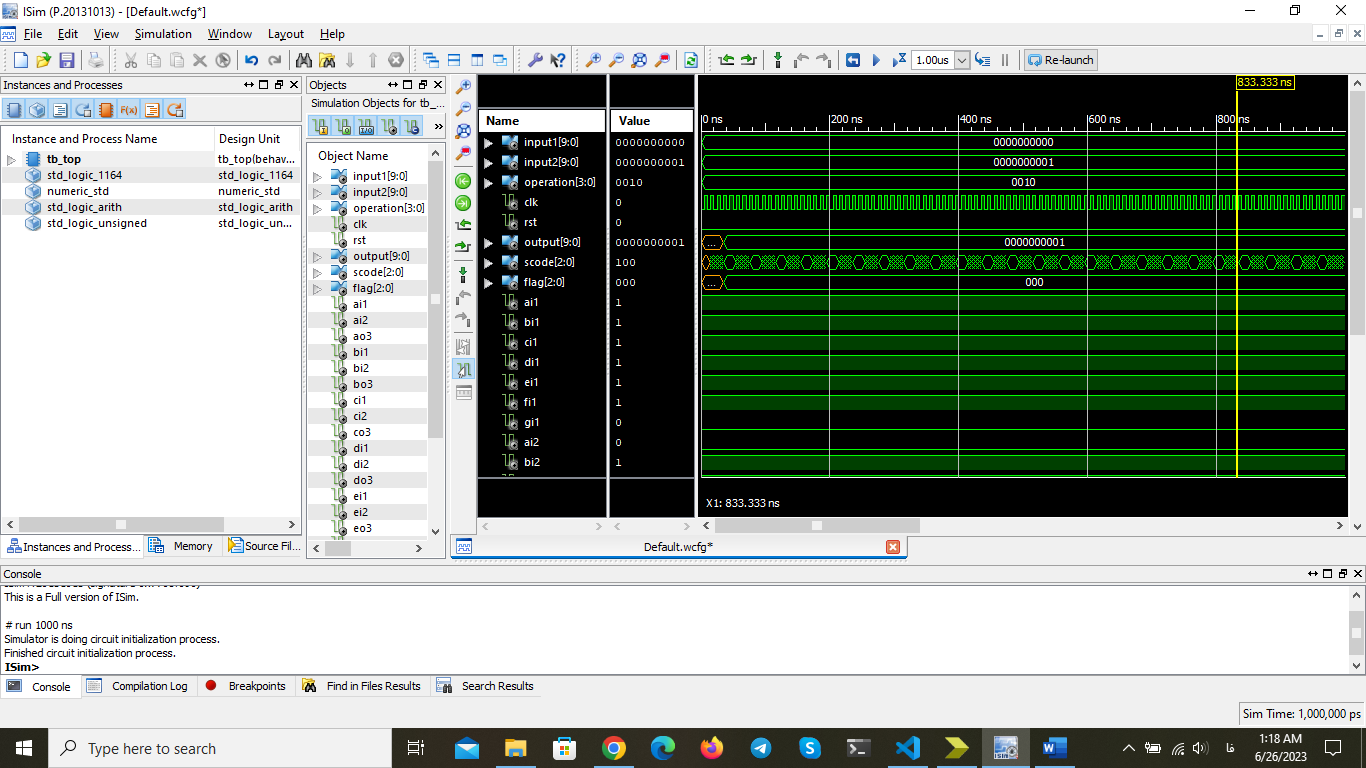
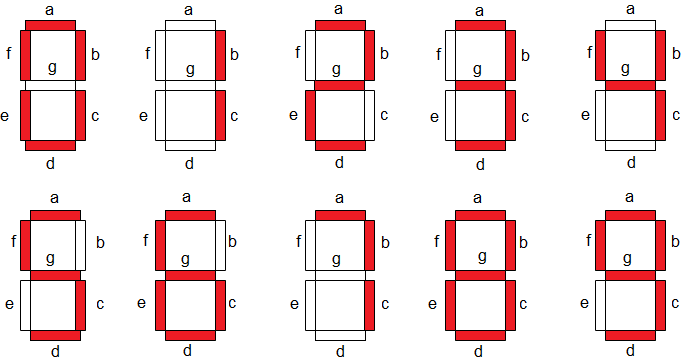
# تست نچ بخش امتیازی و بررسی 7segment ها

تمام بخش های فایل مانند بقیه پروژه می باشد در این بخش ورودی مورد نظر داده می‌شود.

ورودی اول و صفر 0 و ورودی دوم 1 می‌باشد انتظار می‌رود که دیکدر های ورودی 0 و 1 را نشان بدهند.

و خروجی هم با توجه به اینکه از درستور جمع استفاده شده است باید 1 باشد. بعد از شبیه سازی خواهیم داشت:



همین طور که مشخص است خروجی های 7segment ها درست می‌باشند.