به نام خدا

اعضاء گروه:

محمد جمشیدی 97105866

محمد مهدی اصمع 97106419

مهدی فرزادی 97106176

امین مقراضی 97106273

علیرضا حسن پور 97103208

سید ابوالفضل رحیمی 97105941

موضوع پروژه :

تقسیم و ریشه دوم عدد در نمایش ممیز شناور به صورت single precision و double precision با استاندارد IEEE\_754

مقدمه :

در عمل، همه کامپیوترهای مدرن از نمایش ممیز شناور که در استاندارد IEEE 754 مشخص شده برای تمامی اعدادی که با یک مانتیس و یک توان نمایش داده می شوند، بهره می‌برند. همانند نماد علمی، مقدار هر عدد ممیز شناور عبارتست از Mantissa \* 2 ^ exponent.

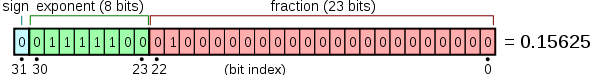
این نحوه ی نمایش اجازه می دهد تا محدوده ی وسیعی از مقادیر با تعداد نسبتا کمی از بیت ها نمایش داده شوند که هم شامل مقادیر کوچک و هم مقادیری است که اندازه ی آن بسیار بزرگ تر از آن است که در عدد صحیحی با همان تعداد بیت نمایش داده شود. با این همه، مشکلی که به وجود می آید این است که تعداد زیادی از مقادیر در محدوده ی نمایش ممیز شناور نمی توانند به درستی نمایش داده شوند. درست مانند تعداد زیادی از اعداد حقیقی که نمی توانند توسط یک عدد دهدهی با تعدادی رقم ثابت معنی دار به نمایش در آیند. وقتی یک محاسبه مقداری را ایجاد می کند که نمی تواند دقیقا به وسیله قالب ممیز شناور نمایش داده شود، سخت افزار باید نتیجه را به مقداری که به درستی نمایش داده می شود، گرد کند.

در استاندارد IEEE 754، روش پیش فرض برای این کار این است که به نزدیک ترین عدد گرد شود. در این روش، مقادیر به نزدیک ترین عدد قابل نمایش گرد می شوند و نتایجی که دقیقا در میان دو عدد قابل نمایش قرار دارند طوری انتخاب گرد می شوند که کم ارزش ترین رقم حاصل شان زوج شود. این استاندارد چندین شیوه گرد کردن دیگر را که قابل انتخاب توسط برنامه هستند، تعیین می کند که شامل گرد کردن به 0، گرد کردن به بی نهایت مثبت و گرد کردن به بی نهایت منفی است.

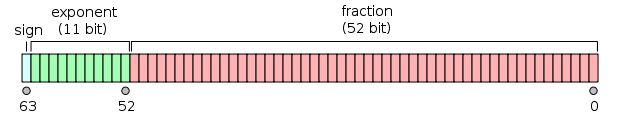
استاندارد IEEE 754، چند پهنای بیتی را برای اعداد ممیز شناور تعیین کرده است. دو پهنای که بیشتر از همه مورد استفاده قرار می گیرند، دقت ساده ( Single Precision ) و دقت مضاعف ( Double Precision ) هستند. اعداد با دقت ساده 32 بیت طول دارند و شامل 8 بیت برای نما، 23 بیت برای بخش کسر و 1 بیت علامت هستند که شامل علامت بخش کسری است. اعداد دقت مضاعف دارای 11 بیت برای نما، 52 بیت برای بخش کسر و 1 بیت برای علامت هستند.

هر دو فیلد نما و کسری در یک عدد ممیز شناور IEEE 754 به شکل متفاوتی نسبت به نمایش اعداد صحیح رمز گذاری می شوند. فیلد کسری، یک عدد مقدار-علامت است و بیان گر بخش کسری یک عدد دودویی است که فرض می شود بخش صحیح آن 1 می باشد. بر این اساس، مانتیس هر عدد ممیز شناور IEEE 754 مثبت یا منفی یک است. بخش کسری، به مقدار بیت علامت وابسته ایت. استفاده از فرض "1 مقدم" در این روش، تعداد ارقام معناداری را که می توانند به وسیله ی هر عدد اعشاری با پهنای مشخصی نمایش داده شوند، افزایش می دهد. همچنین مقدار بایاس درنظر گرفته شده برای این اعداد برابر 127 است. پس برای بدست آوردن مقدار دقیق باید در نظر داشته باشیم:

Single precision:



Double precision:

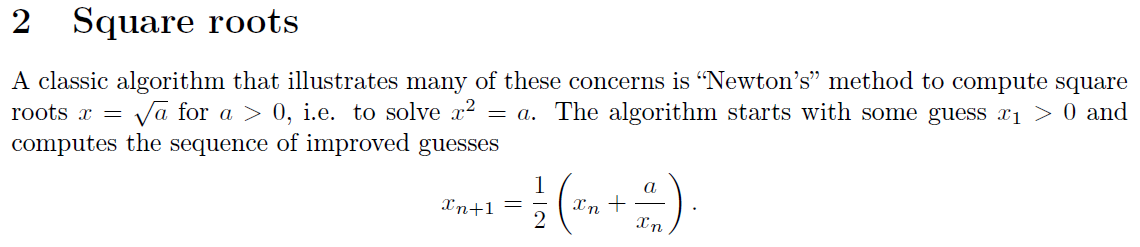


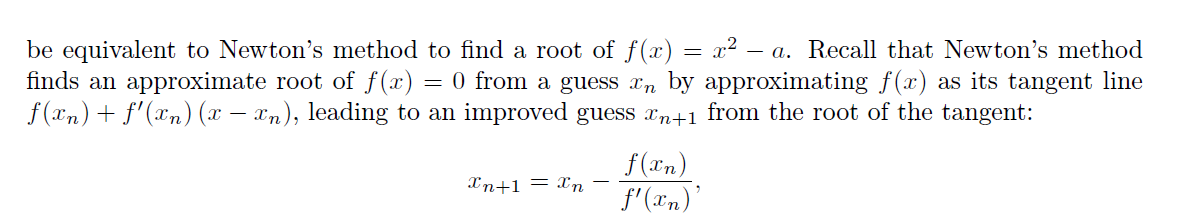
**پروژه اول** ریشه دوم :

الگوریتم:

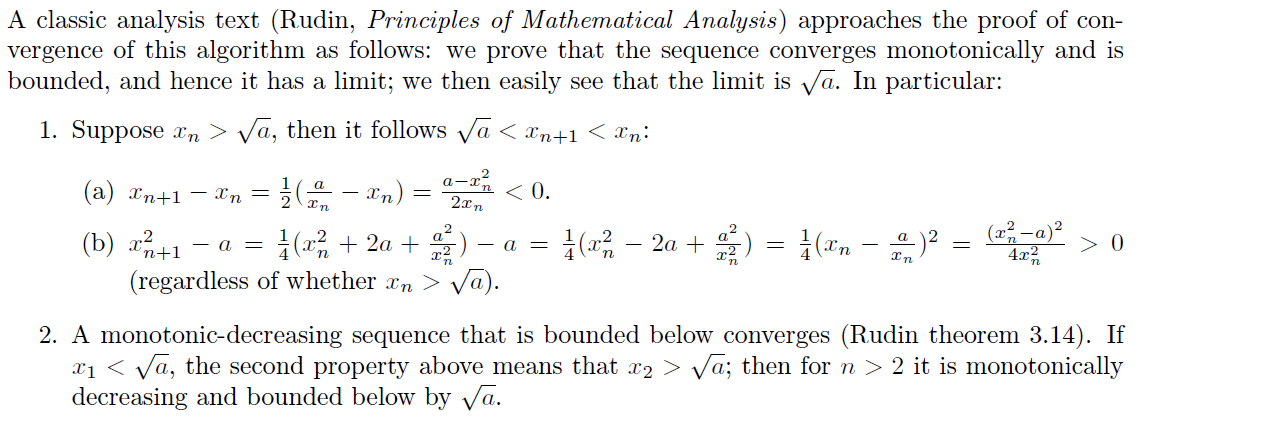
ابتدا منفی بودن عدد را بررسی می کنیم.

بعد با توجه به الگوریتم نیوتن عمل میکنیم :





و برای ROUND کردن خروجی نیز از الگوریتم زیر بهره می گیریم :



خلاصه:

در این الگوریتم از شبه کد مقابل که به الگوریتم نیوتن معروف است استفاده شده است :

r=ix-z^2

bit(N)=1/2^(N/2), and bit(n)=2^(N/2-n)

if bit is one:

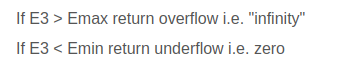
r(n+1) = ix - (z(n)+bit(n))^2 = r(n) - 2z(n)bit(n) - bit(n)^2

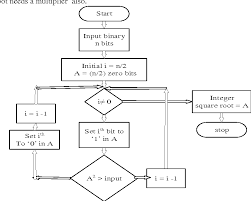
else

r(n+1) = r(n)

که در آنz نتیجه نهایی است و r باقی مانده از فرآیند در مرحله قبل است و این فرآیند N بار انجام میشود تا ریشه دوم عدد به دست بیاید . و در این فرایند در هر مرحله یک بیت از خروجی بدست میاید.

نرمالایز : اگر بیت MSB خارج قسمت صفر بود , خارج قسمت را یک بیت به چپ شیفت می‌دهیم و از exponent یه واحد کم می‌کنیم

در مرحله آخر , overflow یا underflow را چک می‌کنیم. اگر underflow رخ دهد خروجی صفر می‌شود و اگر overflow رخ دهد , خروجی بی‌نهایت می‌شود. شرط رخ دادن overflow و underflow هم به شکل زیر است : 

 در ادامه سعی شده تا شهود بهتری نسبت به الگوریتم داشته باشیم.

ابتدا یک Asm بلاک را مشاهده میکنیم. که نحوه کار رو توضیح داده.

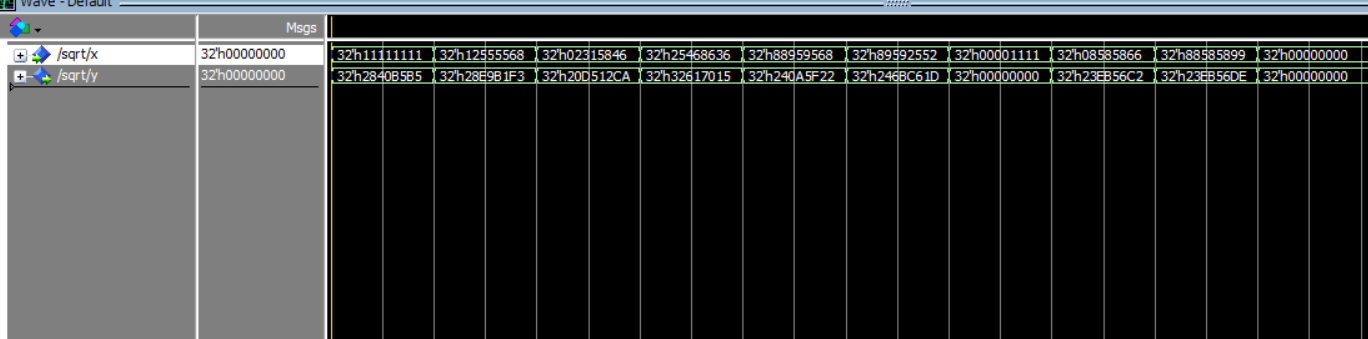
**پیاده سازی، گلدن مدل و سیمولیشن:**

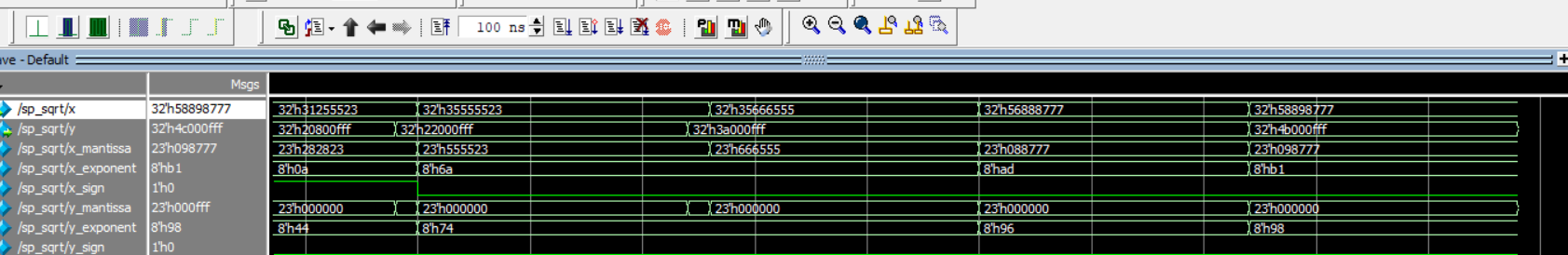
برای انتخاب مدار و نوع الگوریتم مناسب برای پیاده سازی و همچنین گلدن مدل از منابعی که در اختیار داشتیم استفاده شده است.

برای پیاده سازی جذر با دقت نوع اول از کد رفتاری وریلاگی که در اختیارتان قرار گرفته است؛ استفاده کردیم. همچنین با توجه به استاندارد خواسهته شده که برای هر کدام 14 و 28 سایکل مورد نیاز بود، سبب شد بیشتر از فایل پیاده سازی قبلی که در اختیارمان قرار داده شده، استفاده کنیم. همچنین برای تست بنچ؛ از آنجایی که تعداد بالایی تست مورد نیاز بود؛ از تابع random وریلاگ استفاده کردیم و با طولانی تر کردن زمان سیمولیشن؛ و از آنجایی که تابع random از توزیع یکنواخت میآید؛ پس با احتمال یک هر تست بررسی شده و در فایل متنی ذخیره شده است. (sim\_out.txt)

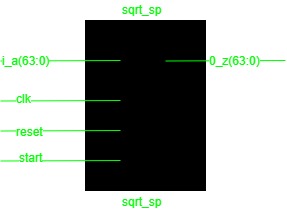
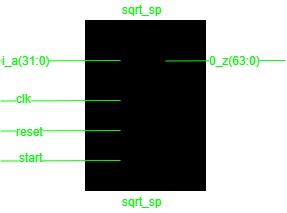
گلدن مدل

همچنین برای گلدن مدل...

فایل های موج حاصل از شبیه سازی نیز در ادامه آمده است. 



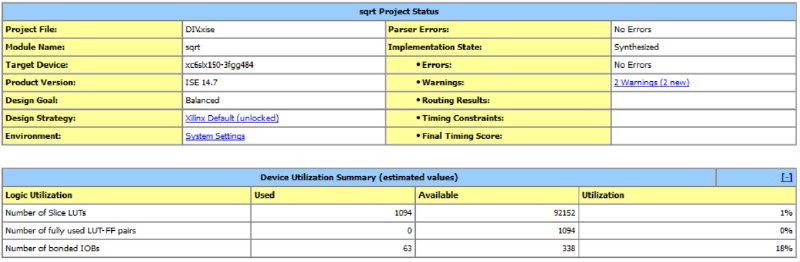
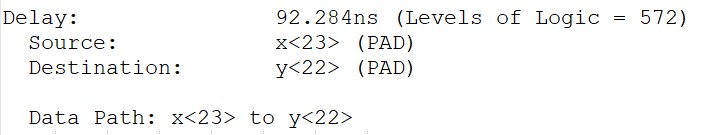
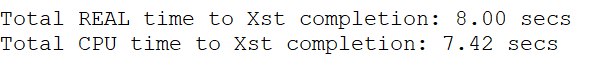
توصیف معماری سیستم :



همانطور که در تصویر مشخص است , دارای کلاک و سیگنال‌های کنترلی ریست و استارت است. همچنین دارای ورودی ۳۲ بیتی ( ۶۴ بیتی برای ماجول دابل ) به عنوان عملوند است. و دارای یک خروجی ۳۲ بیتی (۶۴ بیتی برای دابل ) است

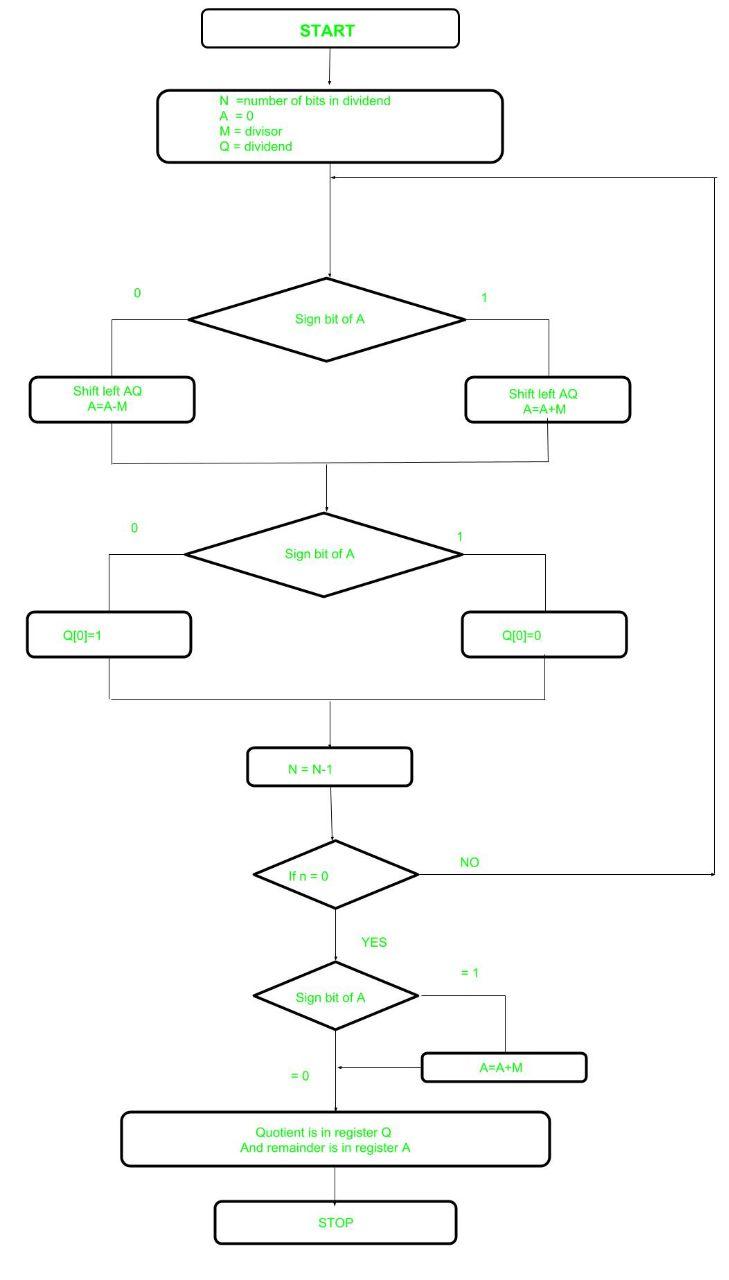
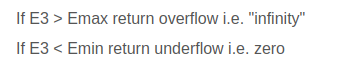
**روند شبیه سازی و نتایج:**

گزارش مربوط به کلاک و دیلی ها و مساحت ( شامل گزارش مساحت / تعداد فلیپ فلاپ ها / LUT ها )



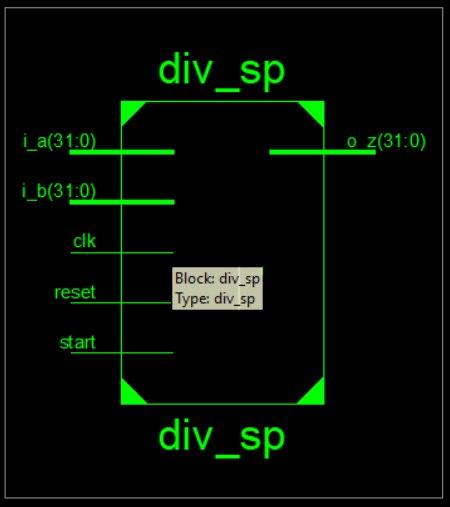
**پروژه دوم: تقسیم**

الگوریتم :

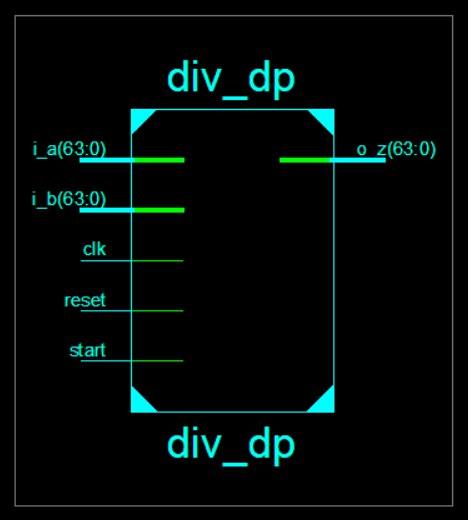
1. ابتدا صفر یا بی‌نهایت بودن مقسوم و مقسوم علیه چک می‌شود
2. بیت علامت خارج قسمت از xor بیت علامت مقسوم و مقسوم علیه بدست می‌آید
3. مقدار exponent خارج قسمت از رابطه زیر بدست می‌اید : 
4. تقسیم کردن دو مانتیس : برای تقسیم کردن از الگوریتم زیر استفاده شده است :  توضیح : ابتدا رجیسترها را مانند شکل مقداردهی می‌کنیم. سپس در هر مرحله اگه بیت MSB رجیستر A صفر بود , A و Q را یک واحد به چپ شیفت می‌دهیم و M را از A کم می‌کنیم. اگر MSB رجیستر A یک بود , A و Q را یک واحد به چپ شیفت می‌دهیم و M را به A اضافه می‌کنیم. این کار را تا زمانی که counter به عدد ۱۲ ( برای دابل به ۲۶ ) برسد تکرار می‌کنیم
5. نرمالایز : اگر بیت MSB خارج قسمت صفر بود , خارج قسمت را یک بیت به چپ شیفت می‌دهیم و از exponent یه واحد کم می‌کنیم
6. در مرحله آخر , overflow یا underflow را چک می‌کنیم. اگر underflow رخ دهد روجی صفر می‌شود و اگر overflow رخ دهد , خروجی بی‌نهایت می‌شود. شرط رخ دادن overflow و underflow هم به شکل زیر است : 

توصیف معماری سیستم :

اینترفیس :



همانطور که در تصویر مشخص است , دارای کلاک و سیگنال‌های کنترلی ریست و استارت است. همچنین دارای دو ورودی ۳۲ بیتی ( ۶۴ بیتی برای ماجول دابل ) به عنوان مقسوم و مقسوم علیه است. و دارای یک خروجی ۳۲ بیتی (۶۴ بیتی برای دابل ) است



توصیف ماشین حالت تقسیم کننده :

ماجول تقسیم کننده در کلاک اول در استیت S\_Start قرار میگیرد و رجیسترها مقدار دهی اولیه می‌شوند و صفر یا بی‌نهایت بودن مقسوم و مقسوم علیه چک می‌شود. و در نهایت به استیت S\_DIVIDE می‌رود

در استیت S\_DIVIDE مانتیس ها در ۱۲ کلاک (برای Double precision در ۲۶ کلاک ) تقسیم می‌شوند و در نهایت به استیت S\_FINISH می‌رود

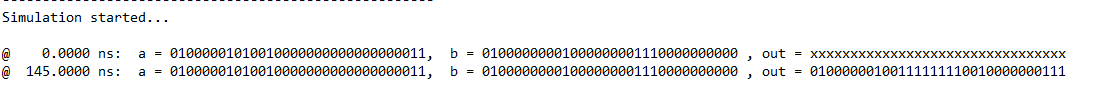
در استیت S\_FINISH , اگر مانتیس نیاز به نرمالایز داشته باشد , نرمالایز می‌شود و بعد overflow , underflow چک می‌شوند و خروجی ست می‌شود و به استیت S\_IDLE می‌رود.

در استیت S\_IDLE منتظر می‌ماند تا سیگنال start , یک شود. از آنجا مراحل گفته شده در بالا تکرار می‌شود

روند شبیه سازی و نتایج :

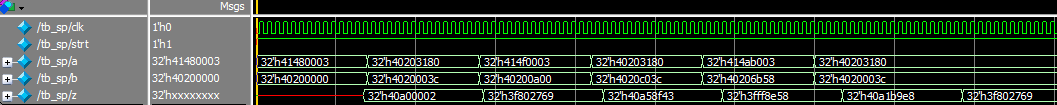
برای شبیه سازی و تست کردن کد , تست کیس ها توسط گلدن مدل تولید می‌شوند و به فرمت‌های دهدهی و single precision و double precision در فایل های جداگانه نوشته می‌شوند . تست بنچ های طراحی شده مقادیر ورودی مقسوم و مقسوم علیه را از این فایل های تولید شده می‌خوانند و نتایج تقسیم را در فایل می‌نویسند.

یکی از خروجی های نمونه ( مربوط به single precision ) :

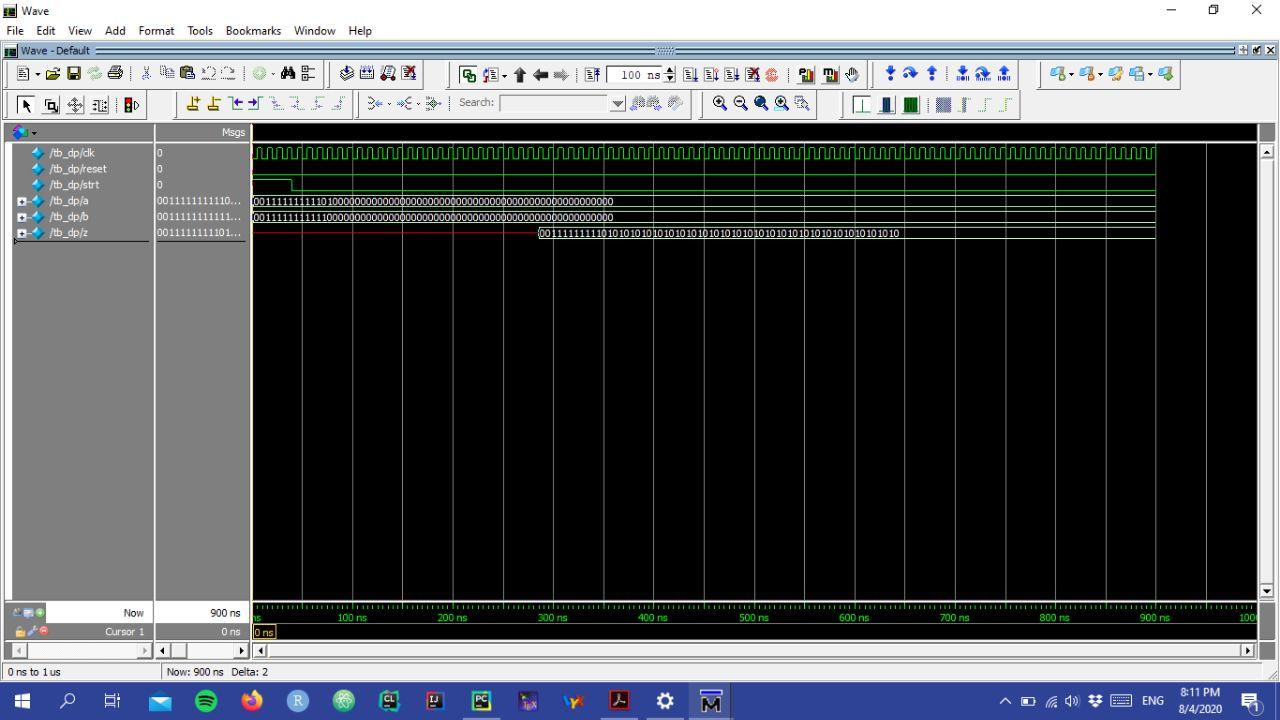


در تست بنچ پریود کلاک ۱۰ واحد زمانی درنظر گرفته شده است

شکل موج مربوط به single precision :



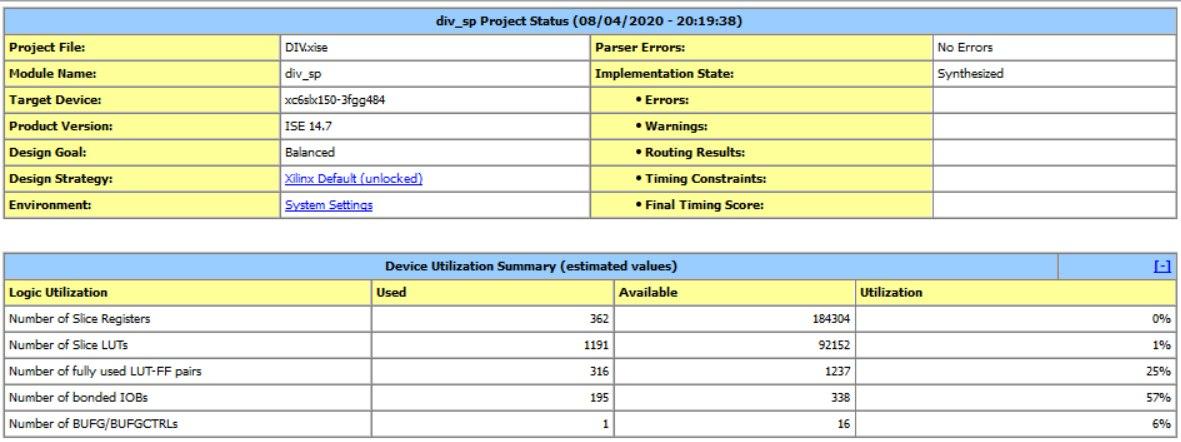
شکل موج مربوط به double precision :



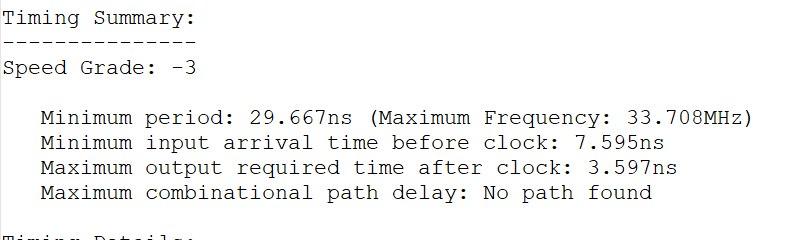
پیاده سازی و نتایج حاصله :

گزارش مربوط به سنتز ماجول single precision :

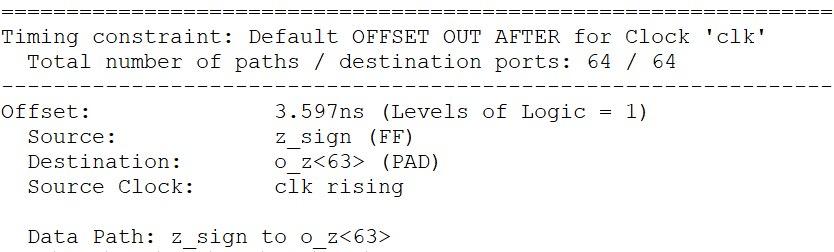
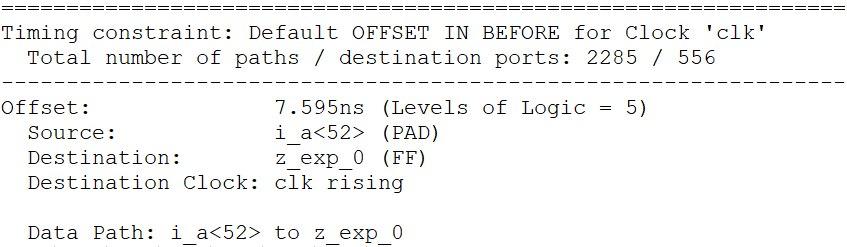
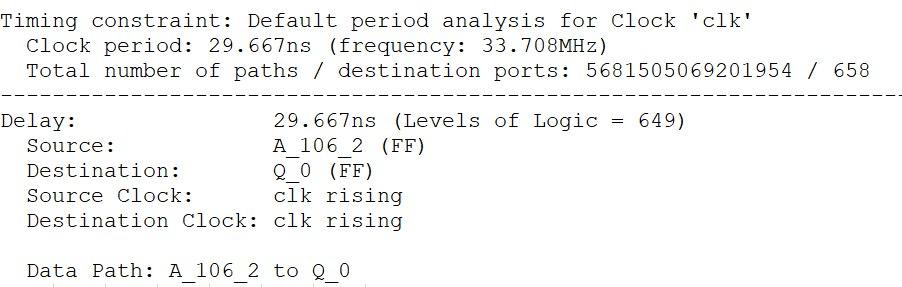
( شامل گزارش مساحت / تعداد فلیپ فلاپ ها / LUT ها )



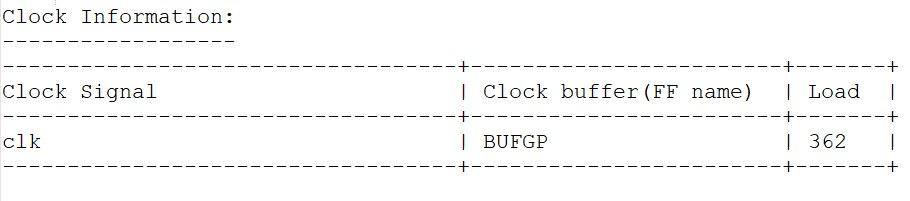
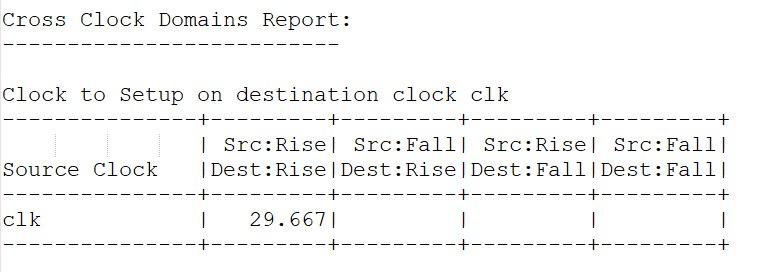
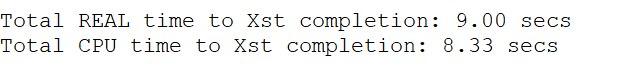
گزارش حداکثر فرکانس کاری :



گزارش دیلی ها و مسیر حیاتی :



گزارشات مربوط به کلاک :



تصاویر مربوط به سنتز ماجول Double precision به پروژه پیوست شده است