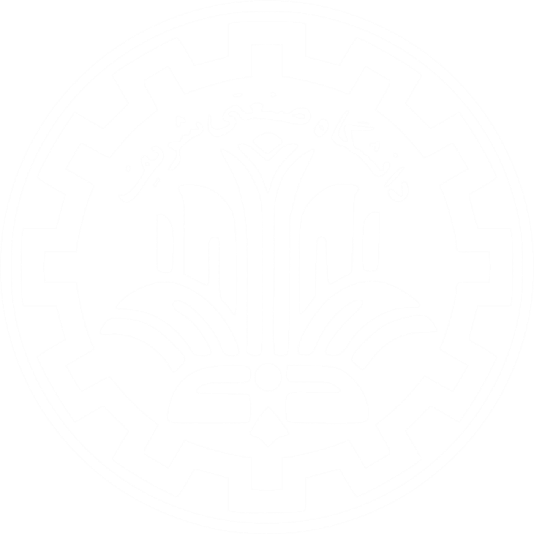
به نام خدا

**فاز سوم پروژه طراحی سیستم های دیجیتال**



**نیمسال دوم سال تحصیلی ۱۴۰۰-۱۴۰۱**

**دانشگاه صنعتی شریف**

**دانشکده مهندسی کامپیوتر**

**توضیحات پروژه:**

**یک واحد پردازشی ضرب دو ماتریس را با استفاده از الگوریتم تقسیم و حل پیاده سازی کرده ایم.دراین طراحی مولفه های ماتریس مطابق با استاندارد IEEE754 دریافت می شوند. این طراحی با استفاده از تقسیم و حل به گونه ایی انجام شده که انجام عملیات ضرب ماتریس به صورت موازی و بهینه انجام شود.**

**اعضای تیم:**

* ***سهیل نظری مندجین***
* ***علیرضا حقی***
* ***مهدی سعادت بخت***

**چکیده**

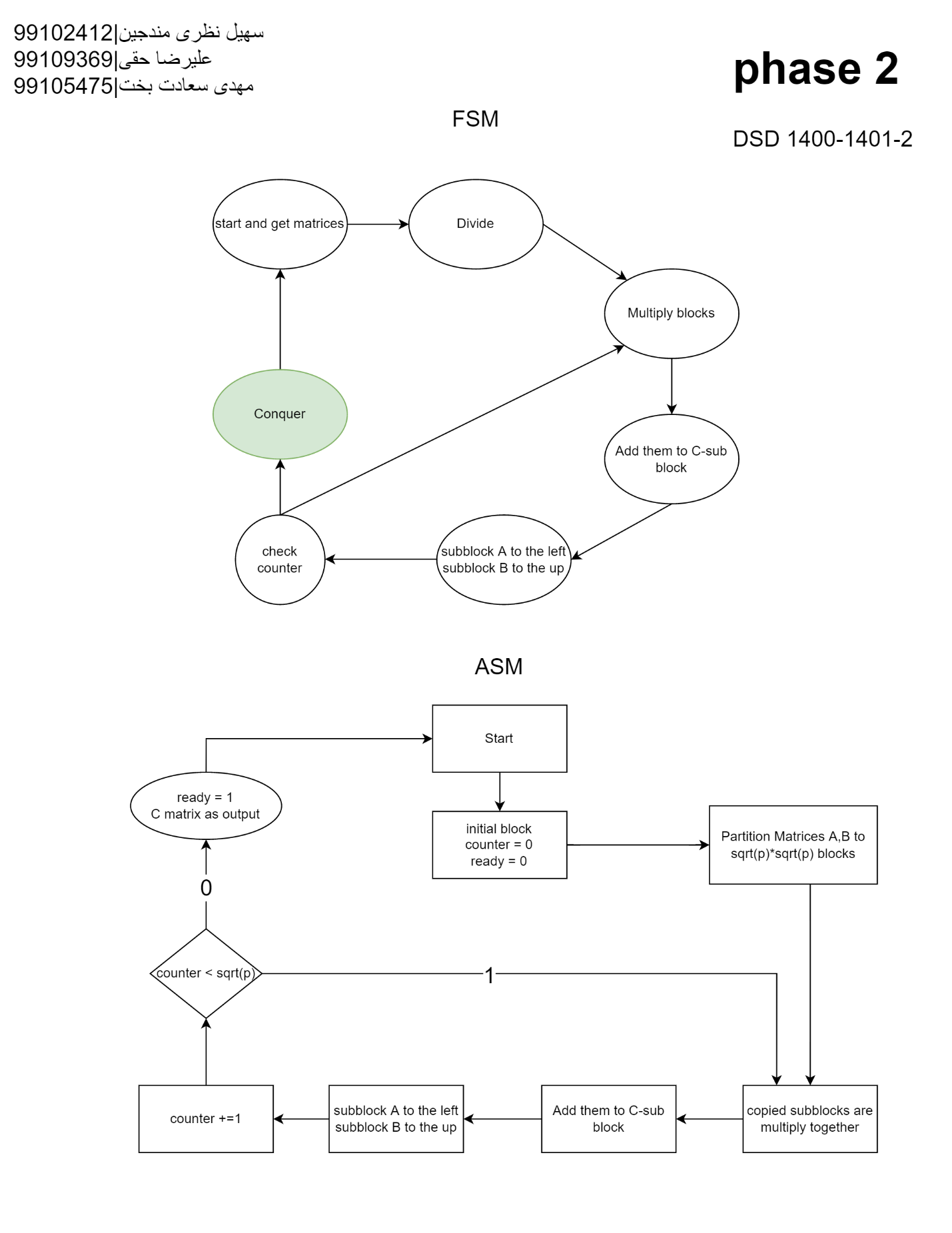
**با توجه به محاسبات سنگین محاسباتی ماتریسی در حوزه های مختلف، پردازش موازی ضرب ماتریس ها در راستای کمینه کردن میزان انرژی و مساحت مورد استفاده اهمیت ویژه ای پیدا کرده است.**

**در این پروژه، بنابر دغدغه ی الزام ضرب موازی ماتریس ها، به طراحی یک مدل تقریبا بهینه با استفاده از زبان برنامه نویسی verilog بر اساس مساحت و تعداد ماژول ها رسیده ایم. در مدل ما از واحد های محاسباتی استفاده شده است که قادر است در کلاک محاسبات موازی دو ماتریس را انجام دهد که در این جا بیانگر مجذور تعداد واحد های پردازش موازی است. کد های ما کاملا قابل سنتز است و این امر بوسیله ی نرم افزار سنتز vivado صورت گرفته است. همچنین صحت کد و کارایی آن توسط آزمایش کننده ای توسط verilator صورت گرفته است. نکته ی حائز اهمیت دیگر مدل ما، پارامتریک بودن است که می توان بوسیله آن برای هر ابعاد دلخواه و هر تعداد دلخواه پردازنده آن را در نظر گرفت که در آن تعداد پردازنده مربع کامل است و ابعاد بر مجذور پردازنده ها بخش پذیر است.**

**مقدمه**

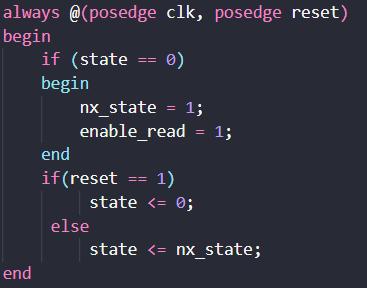
**Fsm and Asm:**

**در این بخش ما شمایی کلی از حالت و وضعیت های محاسباتی ممکن برای ماژول ضرب کننده نشان می دهیم. که از 7 حالت تشکیل شده است**

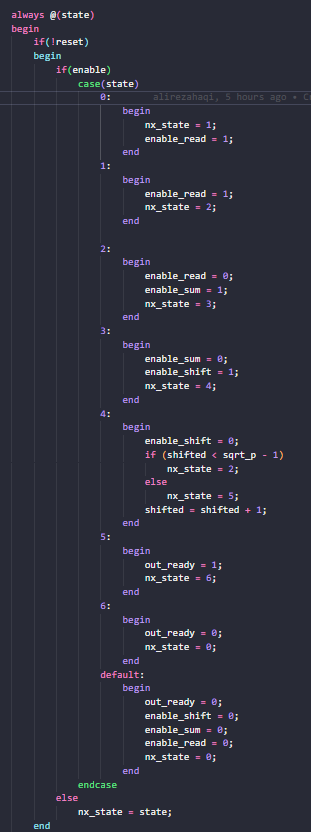


**با استفاده از طراحی Fsm انجام شده یک ماژول کنترلر تعریف کرده ایم که به وسیله آن استیت های مختلف ماشین ما تغییر کرده و سه بیت کنترلی برای خواندن، جمع کردن و شیفت دادن در اختیار داریم که میان استیت ها مقادیر شان را تنظیم میکنیم. نکته ی دیگر این است که ماشین کنترلی مد نظر از نوع Moore است. بنابراین به ورودی برای حرکت بین حالات حساس نیست.**

**تکه کد ریست و استارت در استیت 0**:



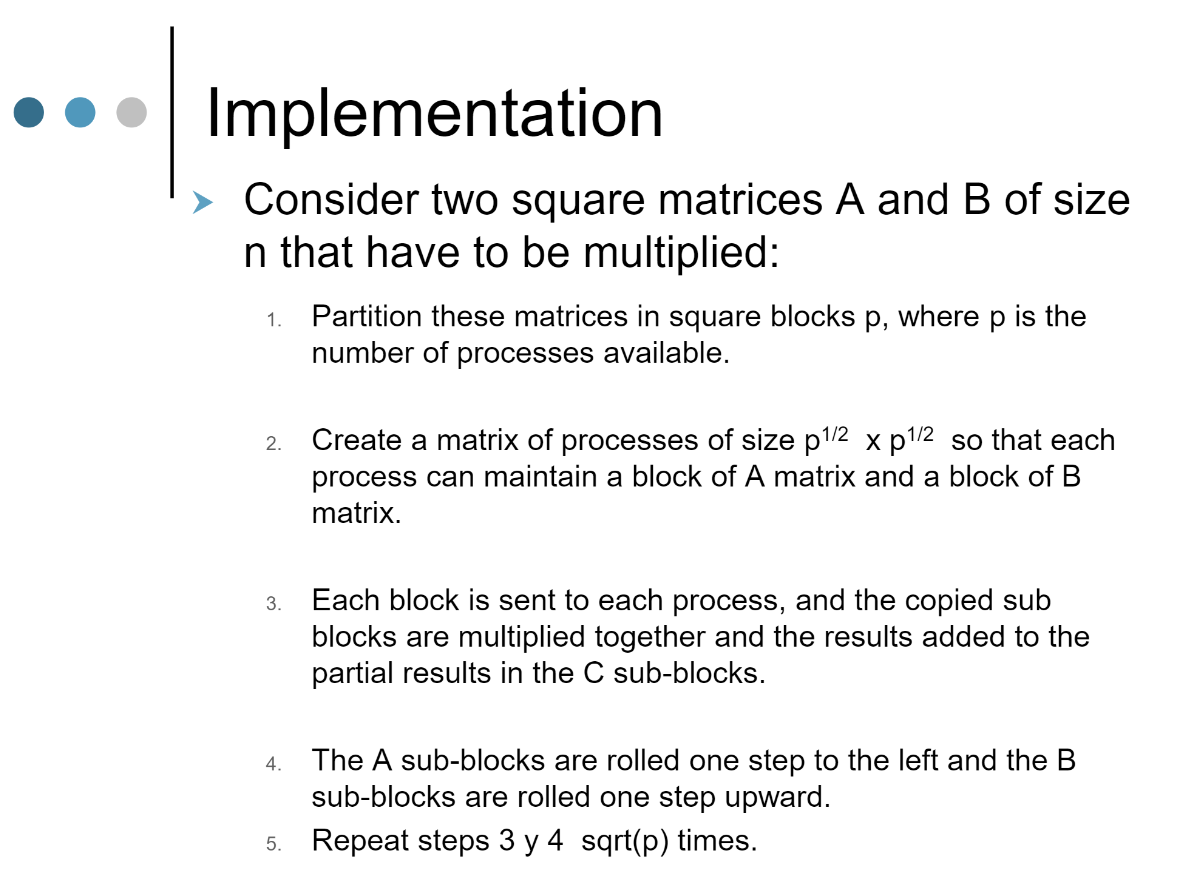
**تکه کد انتقال میان استیت ها و ست کردن بیت های کنترلی:**



**الگوریتم ضرب ماتریس**

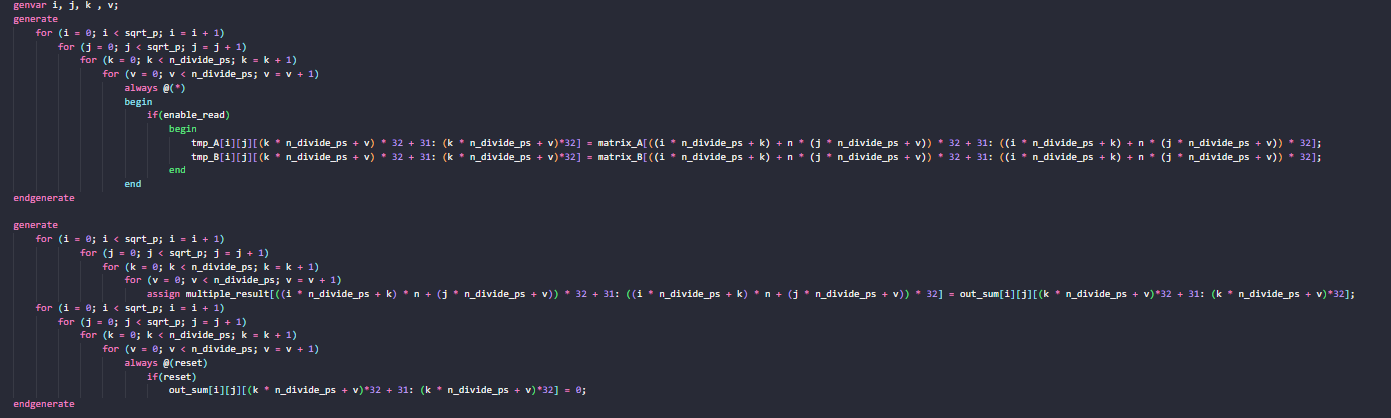
**الگوریتم کلی بدین شکل است که هر دو ماتریس n \* n را به ماتریس با ابعاد تقسیم می کنیم. سپس در طی p مرحله با استفاده از شیفت و ضرب اجزای متناظر ماتریس های بدست آمده و در نهایت جمع آنها جواب نهایی حاصل می شود. در عکس پایین نیز شبه کدی برای الگوریتم ارائه شده است. ( را تعداد پردازنده ها است).**

**عملیات های ضرب دو ماتریس به شکل تقسیم و حل به 5 مرحله تقسیم می شود که به شکل زیر است.**

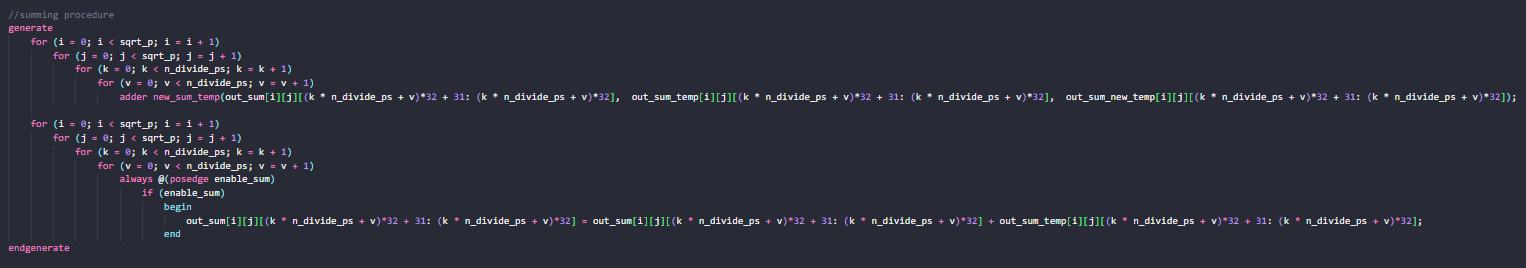


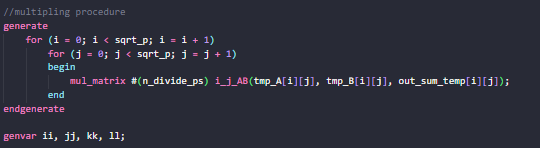
**ما در یک ماژول به اسم array\_divider این مراحل را با استفاده از جنریتور ها پیاده سازی کرده ایم تکه کد مراحل مختلف به ترتیب آمده اند. در این بخش بواسطه جنریتور ها توانستیم برنامه ی کاملا کارا و پارامتریک که هیچ وابستگی به متغیر و عدد ثابتی ندارد را پیاده سازی کنیم که هر چند منجر به پیچیدگی های قابل ملاحظه ای در پیاده سازی کد شد.**

**مرحله اول و دوم ( پارتیشن بندی ماتریس به ساب بلاک های سایز )**

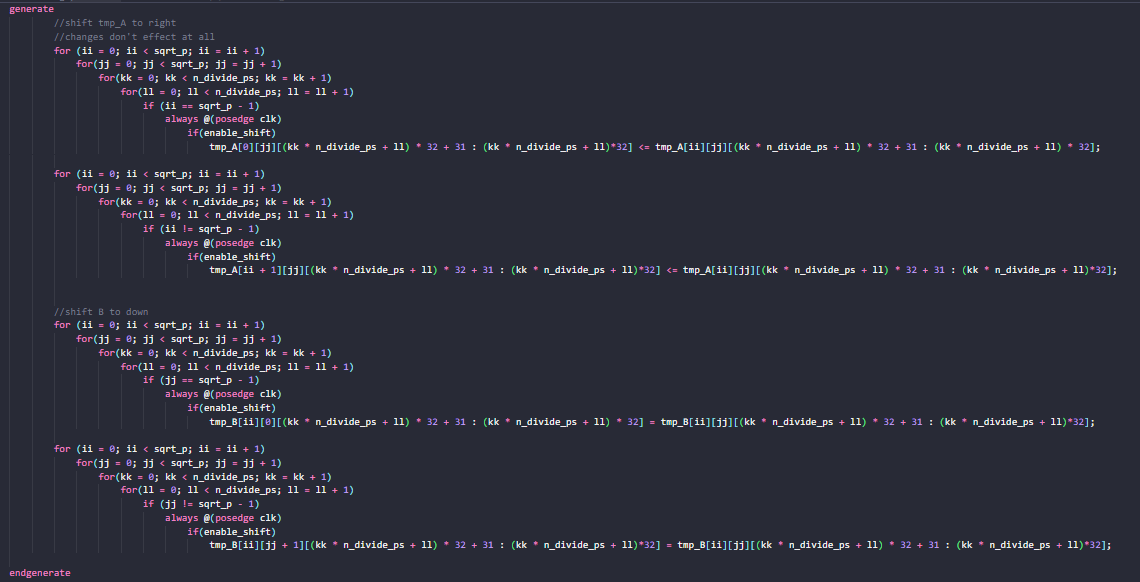


**مرحله سوم (ضرب ساب بلاک ها در هم و سپس جمع ان ها با هم**)





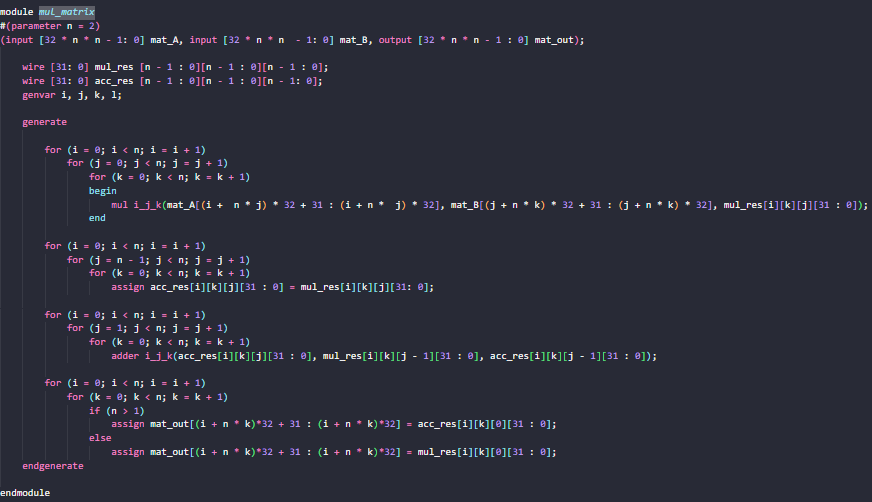
**مرحله چهارم (شیفت دادن ماتریس a به راست و ماتریسb به پایین**)



**مرحله پنجم نیز در تصاویر نشان داده است همانطور که میبینید داخل یک حلقه به اندازه قرار دارند.**

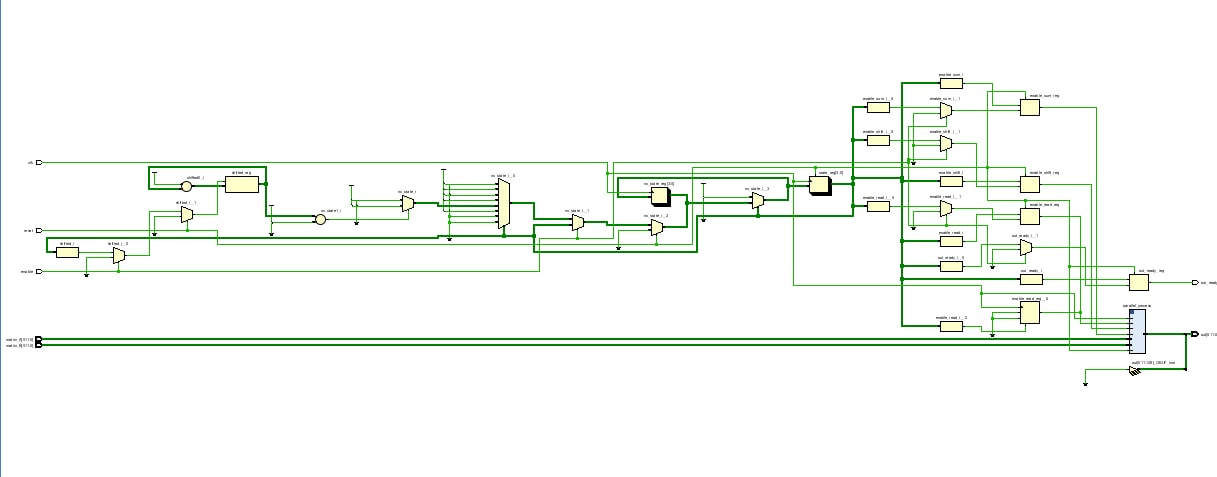
**ضرب ماتریس**

**یکی دیگر از بخش های مورد نیاز برای پیاده سازی ضرب موازی ماتریس ها، ضرب دو ماتریس با ابعاد دلخواه است که در آن برای ضرب دو ماتریس از واحد ضرب و واحد جمع استفاده شده است که delay که در این عملیات وجود دارد برابر است با n برابر delay یک واحد جمع کننده بعلاوه ی یک واحد ضرب کننده. هر چند اگر بهینه تر پیاده سازی کرد می توان این delay را به بار delay جمع کننده و یک delay ضرب کننده رساند که با توجه به بزرگ تر شدن n می توان به شکل قابل توجهی سرعت کلاک را کمتر کرد و در غیر اینصورت می توان سرعت کلاک را پایین نگه داشت ولی تعداد دور هایی که کلاک در استیت ضرب کردن است بنابر آزمایشات زمانی بدست آورد.**

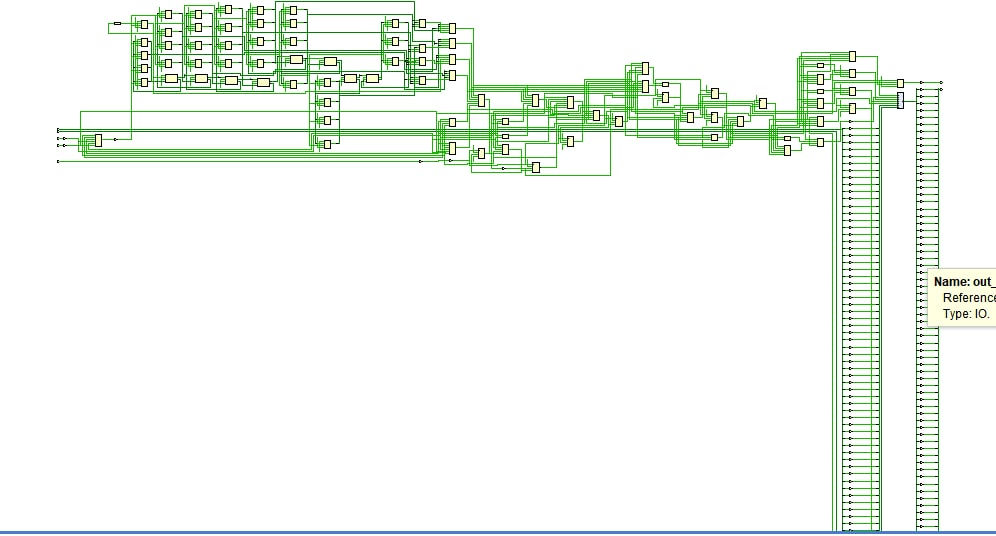


**ما با استفاده از ابزار سنتز Vivado طراحی خود در وریلاگ را سنتز کرده و کد های ما کاملا سنتز پذیر بوده و خروجی های آن به شکل زیر می باشد.**

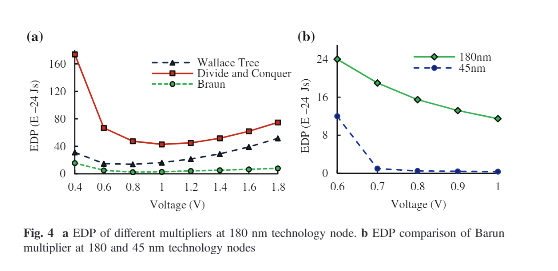
**RTL Schematic**



**Synthesis schematic**



**ما با استفاده از روش تقسیم و حل و همچنین موازی سازی عملیات های پیدا کردن هر یک از درایه های ماتریس خروجی در توان مصرفی نهایی قطعه خود صرفه جویی بسیاری انجام داده ایم طبق آزمایش ها و محاسباتی که بر روی این قضیه انجام شده است جدول های زیر برای سه الگوریتم مختلف انجام اینکار نشان داده شده است.**



**EDP**

The proposed work uses energy-delay product (EDP), where energy the total energy consumption of cores and delay is the amount of time for executing applications.