

پروژه‌ی درس طراحی سیستم‌های دیجیتال

نام استاد: مهندس فصحتی

بهار ۱۴۰۳

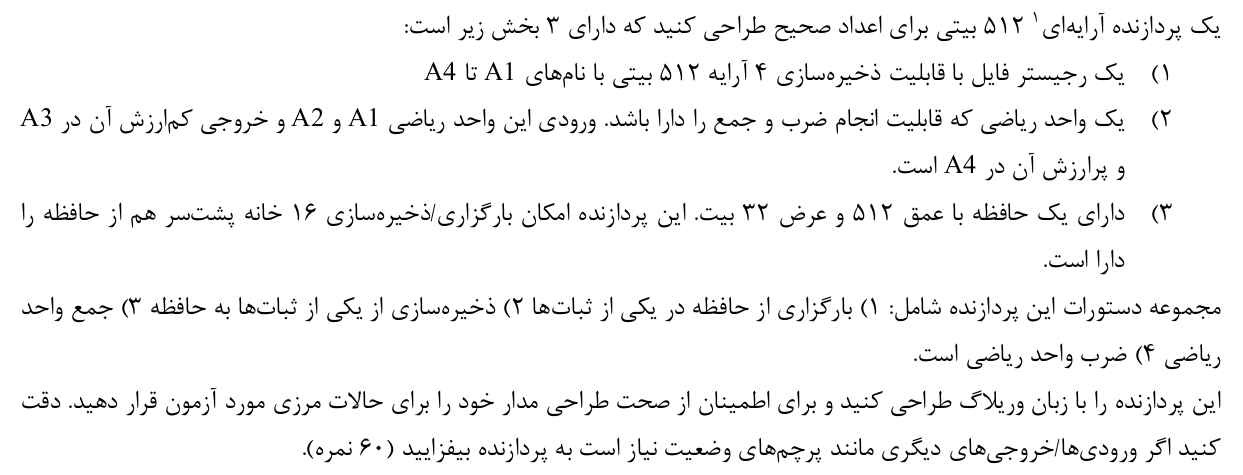
دانشگاه صنعتی شریف

─

حورا عابدین

شماره دانشجویی: ۴۰۱۱۰۶۲۰۹

در این پروژه قصد داریم یک پردازنده‌ی آرایه‌ای ۵۱۲ بیتی مطابق توضیحات زیر پیاده سازی کنیم:



با توجه به مسئله‌ی مطرح شده، نیاز است ۴ ماژول طراحی کنیم:

1. **RegisterFile:**

از این ماژول و رجیسترهای A1, A2, A3, A4 جهت انجام محاسبات و نگه‌داری مقادیر جهت نمایش و … استفاده خواهیم کرد.

1. **ALU:**

درون این ماژول، منطق ریاضی مورد نظر پیاده سازی می‌شود. مانند پردازنده‌های موجود، برای هر یک از عملیات‌های جمع و ضرب خواسته شده یک opcode در نظر گرفته شده است.

1. **Memory:**

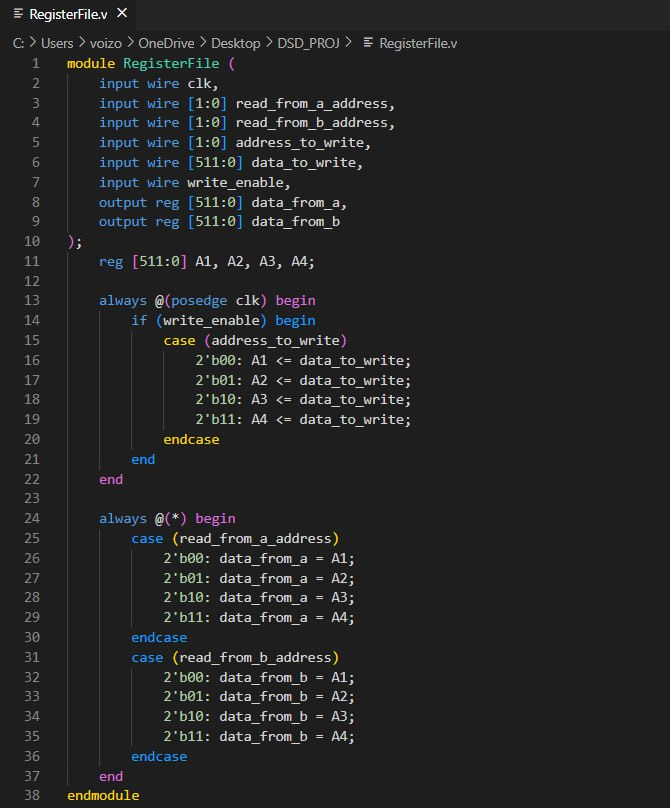
این ماژول جهت خواندن و نوشتن از روی حافظه مورد استفاده قرار می‌گیرد.

1. **Vector Processor:**

در نهایت، این ماژول با استفاده از ۳ ماژول اول که در واقع بخش‌های تشکیل‌دهنده‌ی پردازنده هستند، خواسته‌ی مسئله را برآورده می‌کند.

همچنین نیاز است یک TB برای پردازنده‌ی خود بنویسیم که کارکرد آن را در حالت‌های مختلف مورد بررسی قرار دهد.

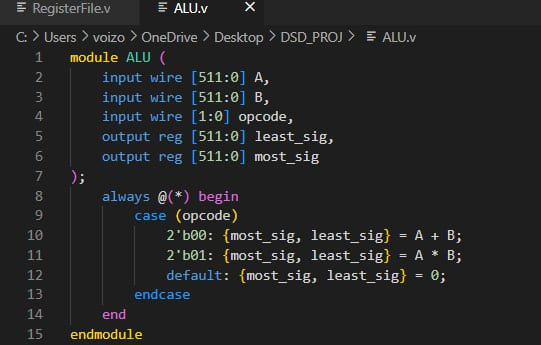
در ادامه به طراحی هر یک از ماژول‌ها و توضیحاتی در مورد هر یک می‌پردازیم.

**طراحی ماژول‌ها:**

**RegisterFile module:**

توضیحات:

مطابق با توصیف وریلاگ بالا، ماژول RegisterFile دارای چندین ورودی و ۲ خروجی است.

* ورودی اول (clk) سیگنال کلاک را مشخص می‌کند.
* از آن جایی که در پردازنده با دو ورودی و عملیات روی آن مواجهیم، ورودی دوم و سوم آدرس رجیستری است که باید دیتا از آن خوانده شود. (طبق خواست سوال، ۴ رجیستر با نام‌های A1, … , A4 داریم که برای هر یک آدرسی دوبیتی معادل شماره‌ آن (۰۰ تا ۱۱) در نظر گرفته شده است.)
* ورودی چهارم آدرس رجیستر مقصد است که می‌خواهیم حاصل درون آن ریخته شود.
* ورودی پنجم مقداری است که قرار است در رجیستر مقصد نوشته شود.
* ورودی ششم مشخص می‌کند که آیا نوشتن خواهیم داشت یا نه (۰ یا ۱)
* دو خروجی نیز مقدار خوانده شده از آدرس‌های داده شده هستند.

همچنین در بدنه‌ی ماژول، ۴ رجیستر ۵۱۲ بیتی تعریف شده‌اند و با گذاشتن switch برای تعیین شماره‌ی رجیستر، خواندن و نوشتن بر رجیسترها پیاده‌سازی شده است.

**ALU module:**

توضیحات:

مطابق با توصیف وریلاگ بالا، ماژول ALU دارای ورودی و خروجی‌های زیر است:

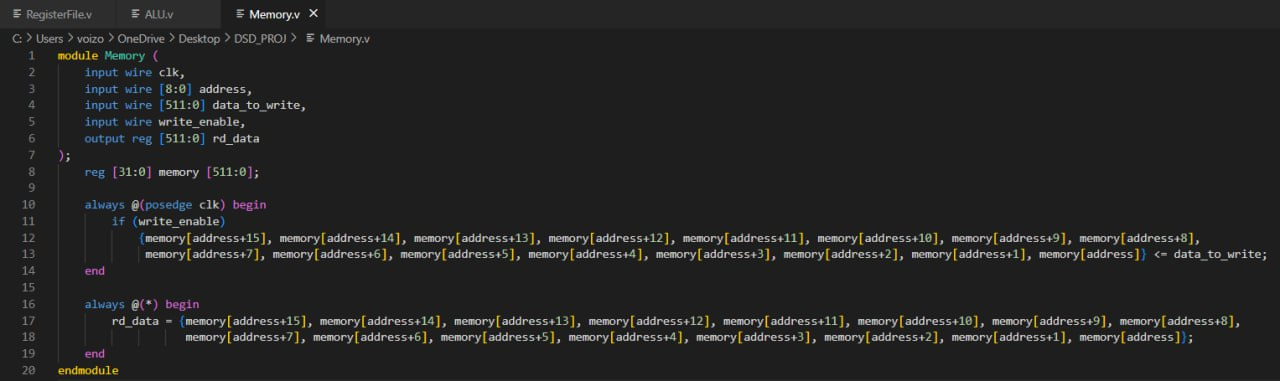
* ورودی اول و دوم، مقادیر A و ‌B هستند که می‌خواهیم روی آن عملیات انجام دهیم.
* ورودی سوم، opcode است که تعیین می‌کند کدام عملیات باید انجام شود.

- ضرب: 01 / جمع: 00

* خروجی اول بخش کم ارزش جواب و خروجی دوم بخش پرارزش جواب است.

همچنین در بدنه‌ی ماژول، منطق محاسباتی پیاده‌سازی شده است که با کدام opcode چه عملیاتی انجام شود و حاصل برگردانده شود.

**Memory module:**

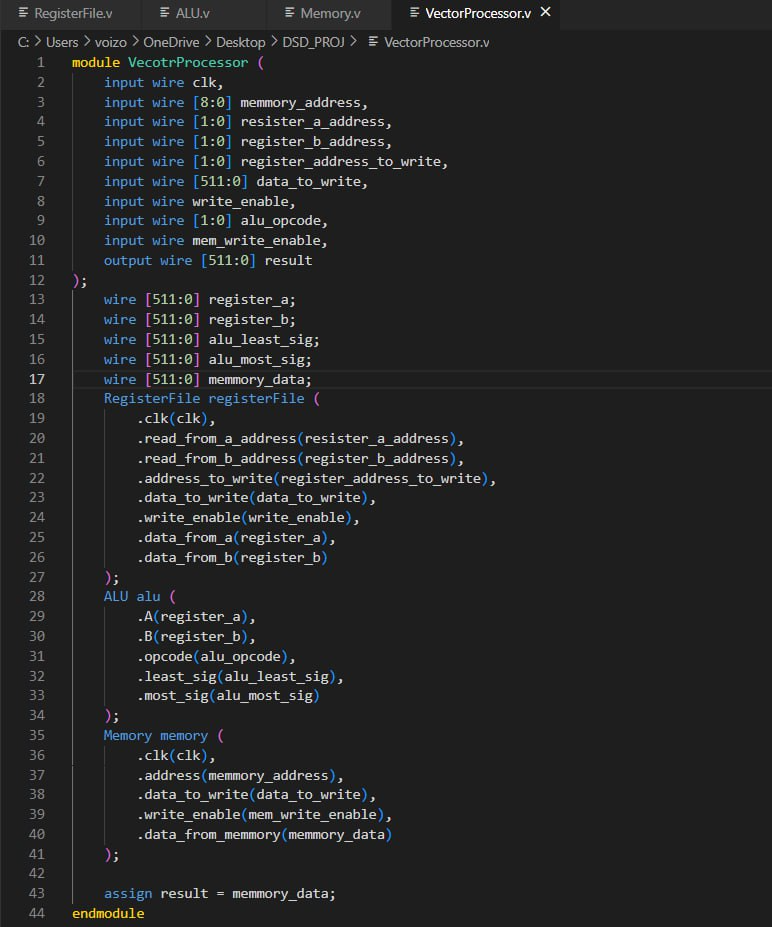


توضیحات:

مطابق با توصیف وریلاگ بالا، ماژول Memory دارای ورودی و خروجی‌های زیر است:

* ورودی اول، آدرس خانه‌ مورد برای شروع در حافظه را مشخص می‌کند.
* ورودی دوم، مقداری است که می‌خواهیم در حافظه نوشته شود.
* ورودی سوم مشخص می‌کند که آیا نوشتن روی حافظه داریم یا خیر. (۰ و ۱)
* خروجی نیز مقدار خوانده شده از خانه‌ی مشخص شده در حافظه است.

همچنین در بدنه‌ی ماژول، مطابق خواست سوال، حافظه‌ای با عمق ۳۲ بیت و قابلیت نوشتن تا ۱۶ خانه‌ی پشت سر هم پیاده‌سازی شده است.

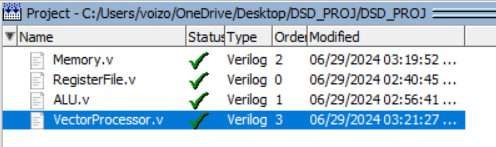
**VectorProcessor module:**

توضیحات:

این ماژول از به هم پیوستن زیرماژول‌ها به دست آمده که خواسته‌ی سوال را به طور کامل برآورده می‌کند. ورودی و خروجی‌ها دقیقا منطبق بر خواست پروژه است. همچنین درون آن، از هر یک از زیرماژول‌های طراحی شده به ترتیب instance گرفته شده است تا وظیفه‌ی خود را جهت عملکرد صحیح پردازنده انجام داده و خروجی تولید شده را به بخش بعدی بدهند.

**تست کردن و مورد آزمون قرار دادن پردازنده در حالات مختلف:**

پیش از نوشتن TB، ماژول‌های نوشته شده را در Modelsim کامپایل می‌کنیم تا از صحت آن‌ها اطمینان پیدا کنیم:

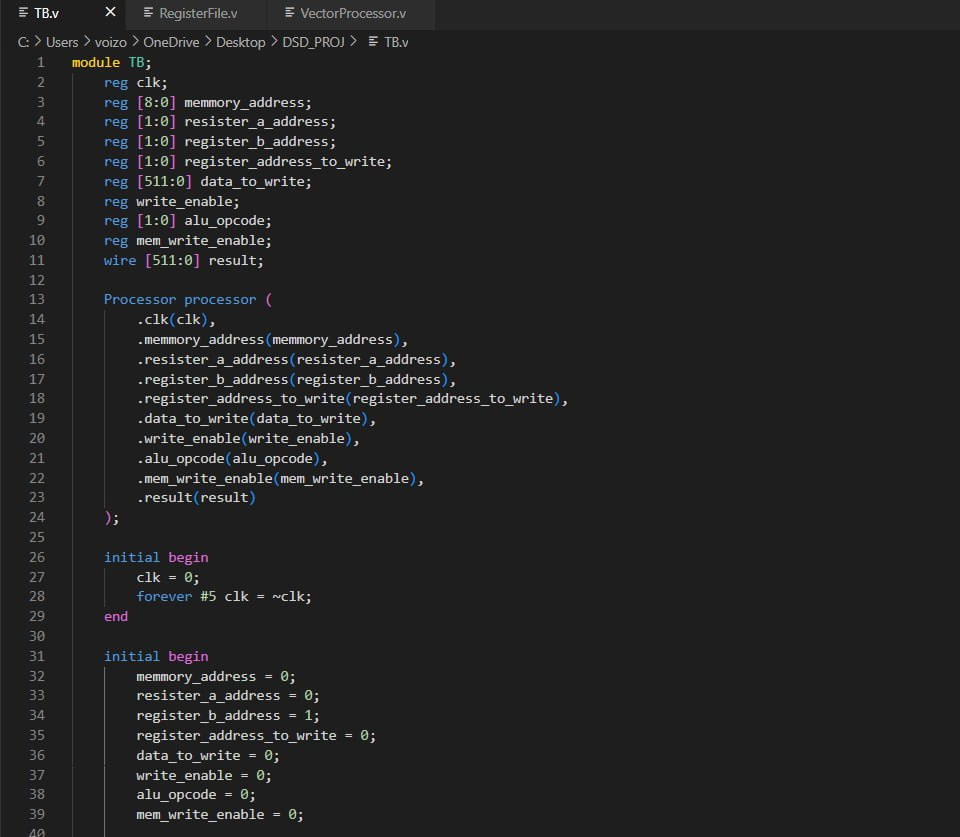


اکنون یک TB می‌نویسیم که فرایند ورودی دادن و خروجی مورد نظر را شبیه سازی کند. در این تست، خروجی مورد انتظار را در کنار خروجی حاصل از پردازنده چاپ می‌کنیم تا بتوانیم صحت خروجی را چک کنیم.

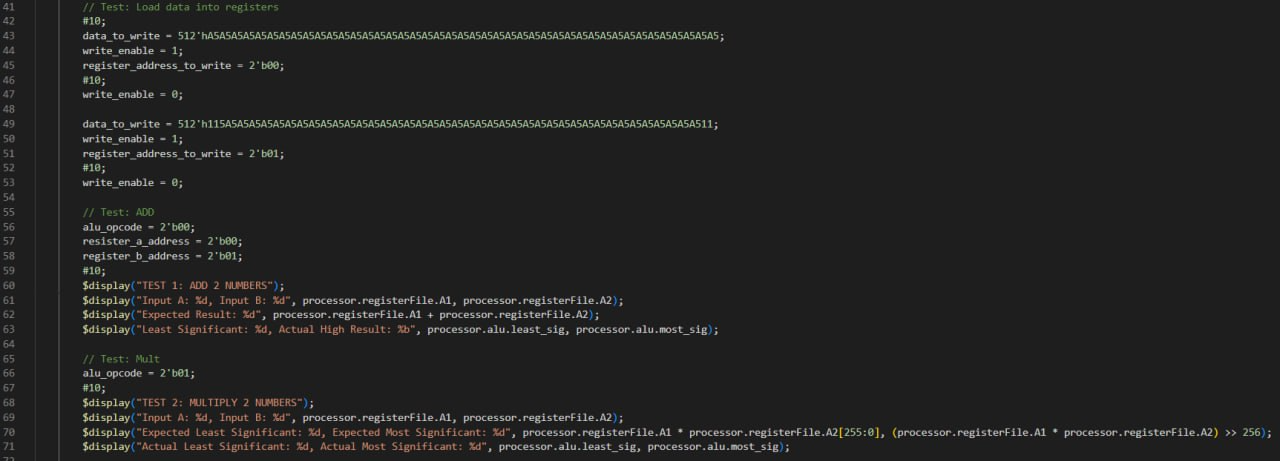
\* اعداد 512 بیتی به صورت hex در تست ورودی داده شده اند اما جهت سهولت خواندن، در پرینت به صورت decimal چاپ شده‌اند.

\* از آن جایی که کد TB طولانی است، آن را به بخش‌های کوچک‌تر تقسیم کرده و توضیح داده‌ایم. سپس نتایج حاصل از شبیه‌سازی آورده شده است.

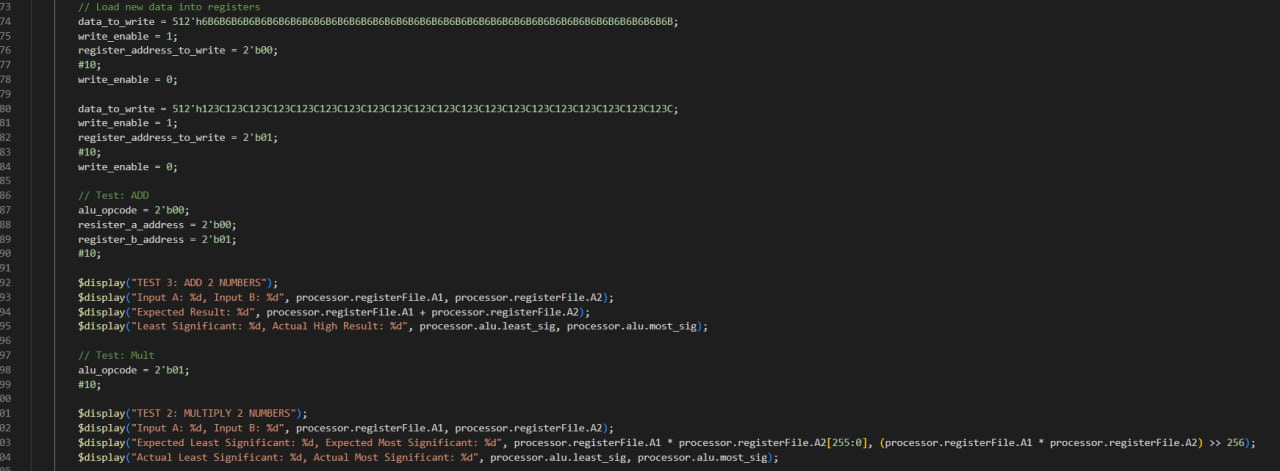
بخش اول: نمونه سازی از processor و مقداردهی‌های اولیه:



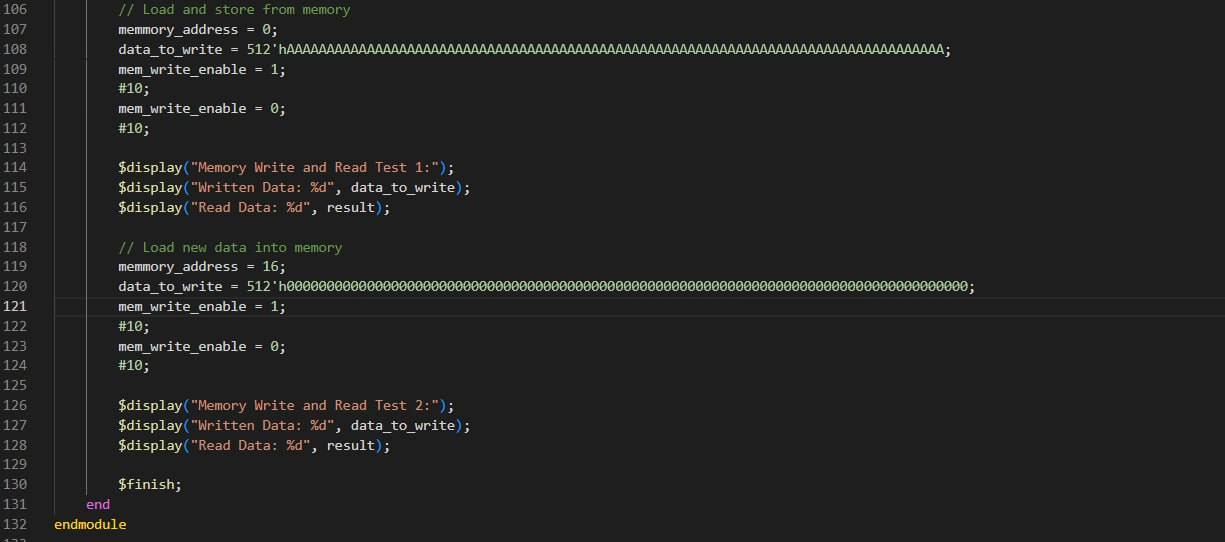
بخش دوم: تست اول برای جمع و ضرب برای مقادیر مشخص شده و مقایسه با حاصل مورد نظر:



بخش سوم: تست دوم برای جمع و ضرب برای مقادیر مشخص شده و مقایسه با حاصل مورد نظر:



بخش چهارم: تست خواندن از حافظه و نوشتن روی آن:



**نتیجه‌ی شبیه سازی:**

با انجام شبیه سازی می‌بینیم که فرایند با موفقیت انجام می‌شود و نتایج تست‌ها به درستی نمایش داده می‌شود.

\* از آن جایی که اعداد بسیار بزرگ هستند، wave نمایش خوبی برای نتیجه نیست و به نتیجه‌ی display اکتفا می‌کنیم.

\* از آن جایی که اعداد بسیار بزرگ هستند، در عکس نتیجه کاملا واضح نیست اما با runکردن و مشاهده‌ی دقیق به صحت نتایج می‌رسیم.

