بسمه تعالي

آزمایشگاه سیستم‌های دیجیتال2

گزارش کار آزمایشگاه سیستم‌های دیجیتال2

پردیس دانشکده‌های فنی دانشگاه تهران

دانشکده مهندسی برق و كامپيوتر

علی ایمانقلی 810197692 – عمید مهرجو 810198333

نیمسال اول 1402-1401

استفاده از در پردازنده به عنوان حافظه داده:

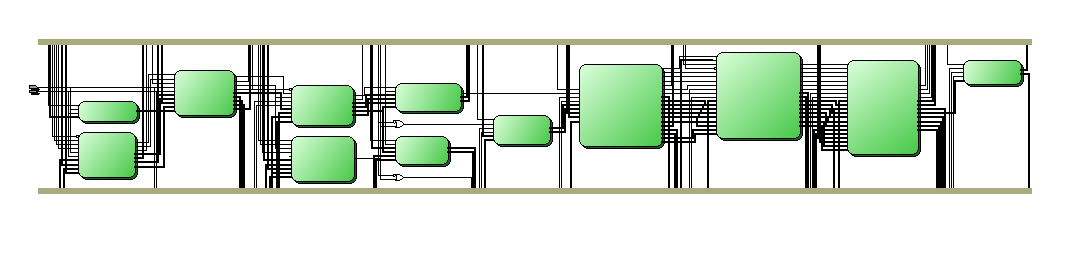
در بخش های قبل حافظه‌ی پردازنده را به صورت ایده‌آل در نظر می گرفتیم، به نحوی که در همان سیکلی که آدرس را بر روی پورت آدرس حافظه قرار می دادیم، عملیات خواند و نوشتن داده انجام می شد. اما در این قسمت می خواهیم حافظه‌ی به کار رفته در پردازنده را به واقعیت نزدیک تر نماییم، فلذا از حافظه‌ی استفاده می نماییم. همانطور که در صورت گزارش عنوان شده است، برای انجام عملیات های خواندن و نوشتن هنگامی که از حافظه‌ی استفاده نمی نماییم، نیازمند به 6 سیکل کلاک می باشیم.

با توجه به اینکه دیگر عملیات دسترسی به حافظه در یک کلام انجام نمی شود و به 6 کلاک نیاز داریم، باید در هنگام دسترسی به حافظه، پردازنده را نماییم تا دستورات بعدی اجاره نشوند و منتظر باشند تا دسترسی به حافظه به صورت کامل انجام شود و سپس دستورات بعدی اجرا شوند.

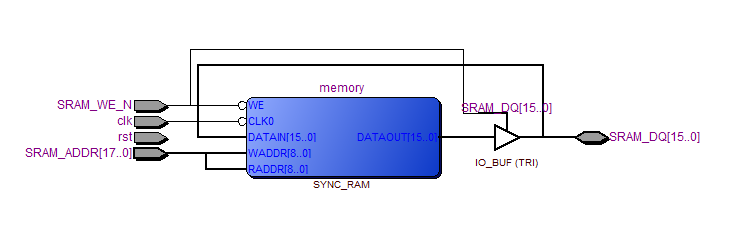
روند دسترسی به حافظه در عملیات های خواندن و نوشتن نیز متفاوت می باشند و به علاوه داده های ذخیره شده در خانه‌های به صورت 16 بیتی می باشند ولی پردازنده‌ی طراحی شده با داده های 32 بیتی سروکار دارد. این موارد بیانگر این است که نیازمند ماژولی برای مدیریت دسترسی به حافظه می باشیم.

پس علاوه بر اینکه به ماژول حافظه نیاز داریم، به یک ماژول کنترلر نیز برای کنترل کردن دسترسی به حافظه و کردن پردازنده در مواقع مورد نظر، نیازمندیم.

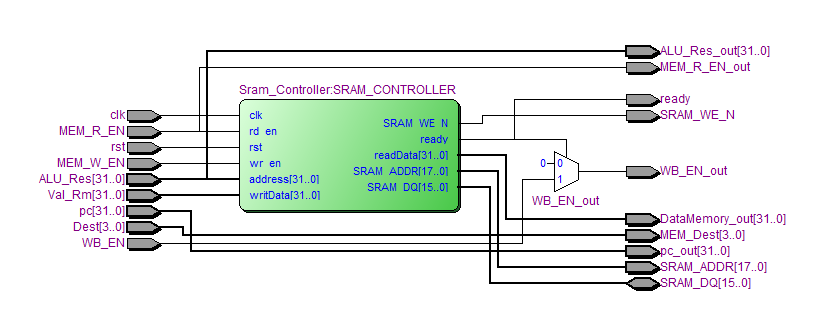
کلی پردازنده:



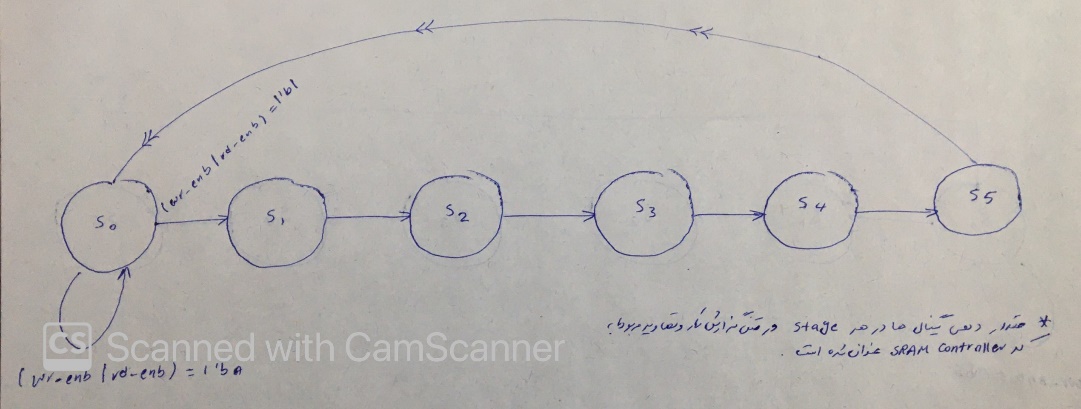
مربوط به :



مربوط به :

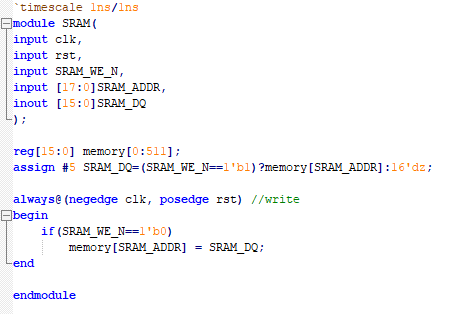


تصویر مربوط به :



اطلاعات مربوط به مقدار دهی سیگنال ها در هر قابل بیان در تصویر بالا نمی باشد به همین علت در تصویر بالا نشان داده نشده است، ولیکن در ادامه گزارش کار به تفصیل اطلاعات مربوط توضیح داده شده است.

در ابتدا ماژول کلی حافظه‌ی را ایجاد می نماییم:



در قطعه کد بالا، در ابتدا ورودی و خروجی ماژول را تعیین می نماییم. کلاک و ریست از سیگنال های ورودی به منظور همگام کردن این ماژول با باقی ماژول های پردازنده می باشند. علاوه بر کلاک و ریست، 3 سیگنال دیگر نیز داریم:   
: زمانی که بخواهیم در حافظه عملیات نوشتن را انجام دهیم این سیگنال 0 می شود. () و اگر بخواهیم عملیات خواندن را انجام دهیم این سیگنال 1 می شود.

: این سیگنال آدرس محل خواندن با نوشتن از حافظه را نشان می دهد.

: این سینال هم به صورت ورودی و هم به صورت خروجی می باشد. این ماژول هنگامی مقدار دهی می شود که بخواهیم عملیات خواندن را انجام دهیم، در این صورت ماژول مقدار این سیگنال را درایو می نمایم. در غیر این صورت ماژول این سیگنال را یا می نماید.

در هنگام عملیات نوشتن نیز، ماژول مقدار داده‌ی موردنظر را از این سگنال می خواند و در خانه‌ی موردنظر از حافظه قرار می دهد.

در ادامه‌ی قطعه کد بالا، حافظه 512 خانه‌ی 16 بیتی تعریف می شود که این حافظه ساختار کلی را تشکیل خواهد داد.

سپس در ادامه‌ی کد عملیات خواندن از حافظه مدریت می شود:



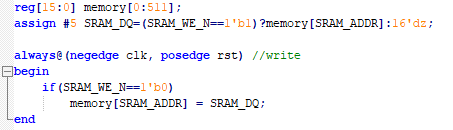
همانطور که از کد بالا مشخص است، زمانی که سیگنال برابر با 1 باشد، پس از 5 نانوثانیه مقدار خانه‌ی مشخص شده توسط بر روی سیگنال قرار می گیرد و به عنوان خروجی به پردازنده تحویل داده می شود. ولی اگر مقدار برابر با 0 باشد مقدار 16 بیت بر روی سیگنال خروجی قرار می گیرد.

دو نکته‌ای که باید توجه داشت:

\* علت اینکه خواند از حافظه 5 نانو ثانیه طور می کشد در این است که طبق مطالب عنوان شده در صورت آزمایش، عملیات خواندن از حافظه در سیکل یکسانی صورت نمیگیرد، بدین معنی که در ابتدا آدرس خانه‌ی مورد نظر برروی پورت آدرس حافظه قرار می گیرد و سپس در سیکل بعدی مقدار مورد نظر از پورت خوانده می شود.

\* علت اینکه سیگنال را هنگامی برابر با 0 است 16 بیت z می کنیم در این است که سیگنال به صورت پورت تعریف شده است، به نحوی که یا توسط مقدار دهی می شود و یا توسط مقدار دهی می شود. حال برای اینکه در هنگام مقدار دهی این سیگنال توسط ماژول های و تعارضی رخ ندهد، باید این سیگنال همواره باشد مگر در صورتی که ماژولی بخواهد آن را مقدار دهی نماید.

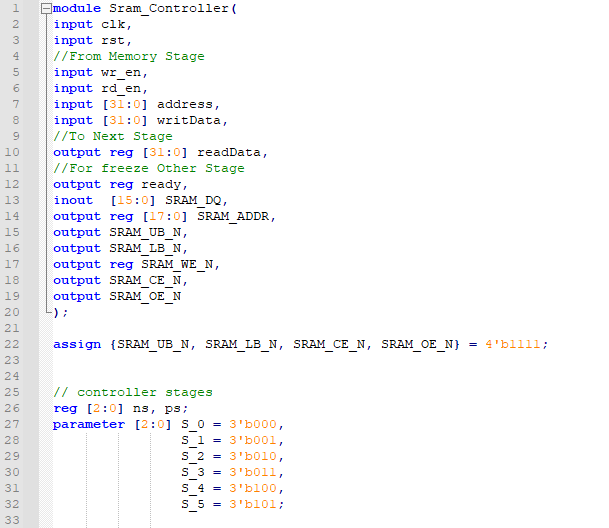
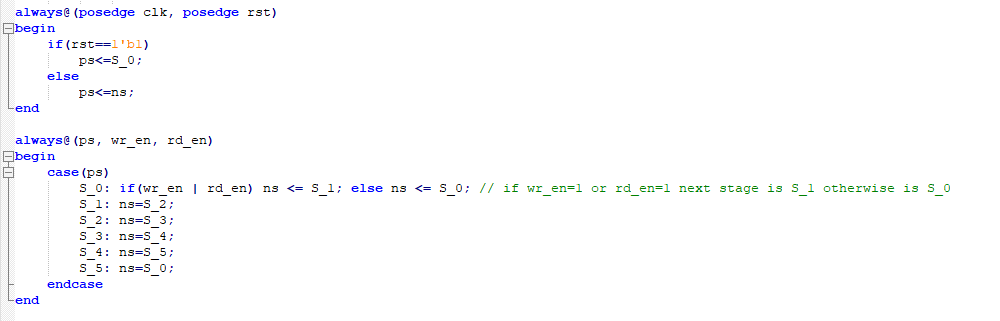
در ادامه‌ی کد عملیات نوشتن در حافظه تعریف می گردد:

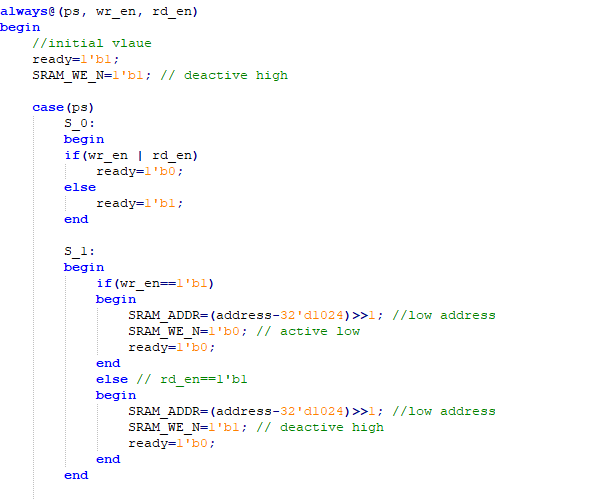
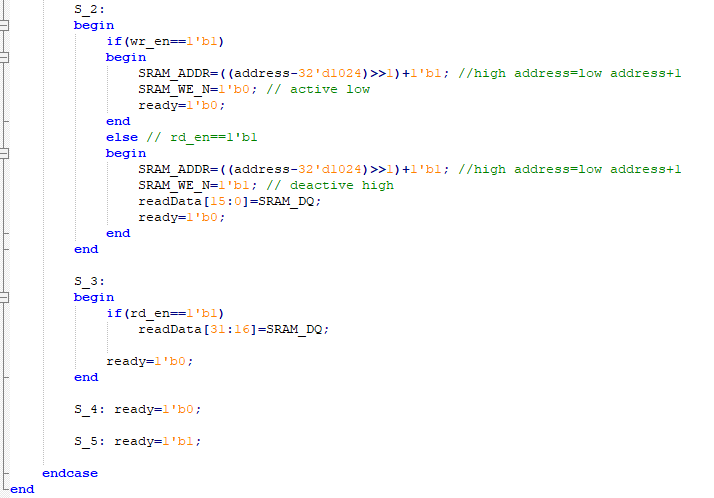
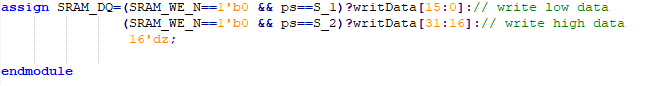


با توجه به کد بالا در لبه‌ی پایین رونده‌ی کلاک، اگر مقدار سیگنال برابر با 0 باشد، مقدار در خانه‌ی موردنظر از حافظه نوشته می شود.

حال در ادامه ماژول را بررسی می نماییم:

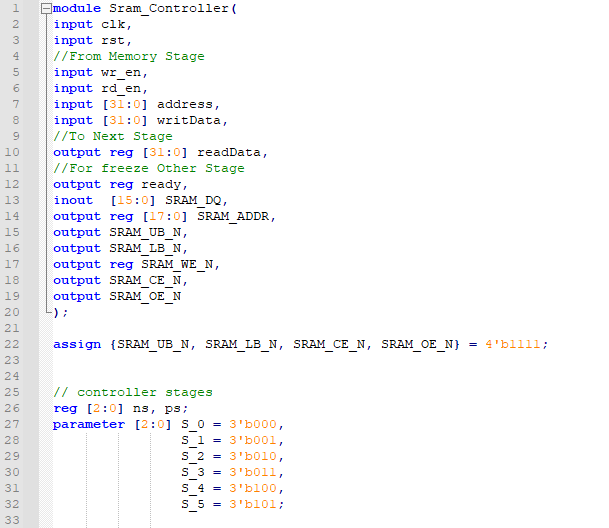
کد ماژول موردنظر به صورت زیر می باشد:

حال به تفکیک هر بخش را بررسی می نماییم:

ورودی و خروجی های ماژول:



کلاک و ریست برای همگام سازی ماژول با باقی ماژول های پردازنده.

و : این سیگنال های ورودی که توسط دستور موردنظر مشخص می شوند، تعیین می نمایند که باید عملیات خواندن از حافظه انجام شود و یا عملیات نوشتن از حافظه

: سیگنال ورودی ای که آدرس محل نوشتن و یا خواند از حافظه را تعیین می نماید.

: سیگنال ورودی که مقدار داده ی موردنظر برای نوشتن در حافظه را تعیین می نماید.

: سیگنال خروجی ای که حاوی مقدار حافظه در خانه‌ی مورد نظرمی باشد.

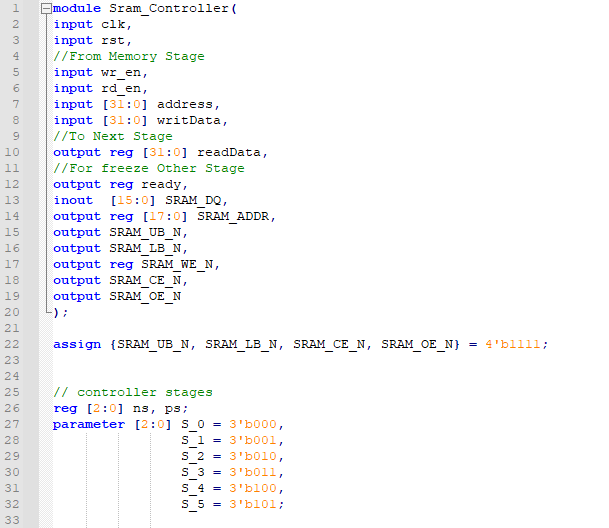
: این سیگنال تعیین می نماید که آیا عملیات خواندن و نوشتن از حافظه به صورت کامل صورت گرفته است یا خیر. توسط این سیگنال می تواینم تا زمانی که عملیات دسترسی به حافظه به صورت کامل صورت نگرفته، پردازنده را نماییم.

: سیگنالی از جنس که انتقال داده میان و را مدیریت می نماید.  
: آدرس دسترسی به خانه‌های

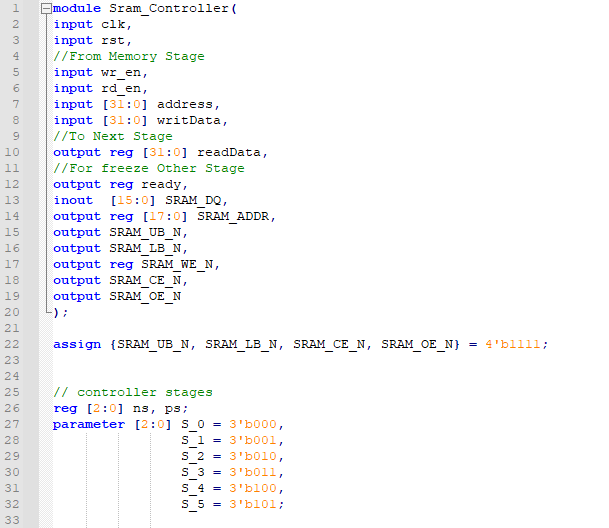
: سیگنالی برای مشخص نمودن عملیات نوشتن و یا خواندن در حافظه (به صورت active low می باشد.)

و و و : برخی از سیگنال های کنترلی SRAM که در این آزمایش به منظور سادگی، آن ها را فقط مقدار دهی اولیه می نماییم.

در ادامه 4 پین حافظه را مقدار دهی می نماییم( طبق خواسته‌ی صورت آزمایش):



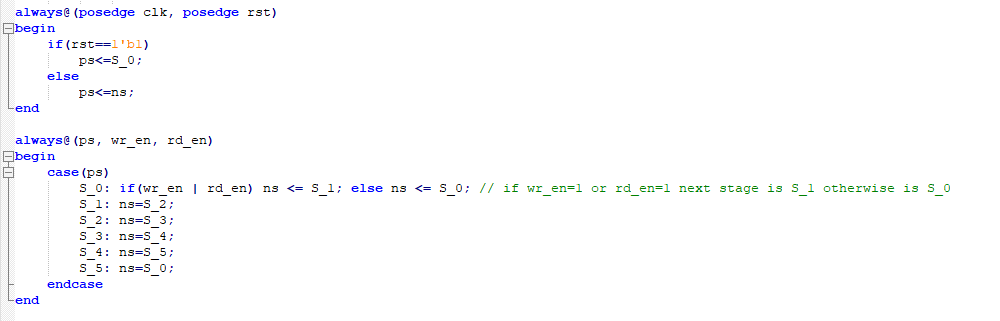
در ادامه های کنترلی را تعریف می نماییم:



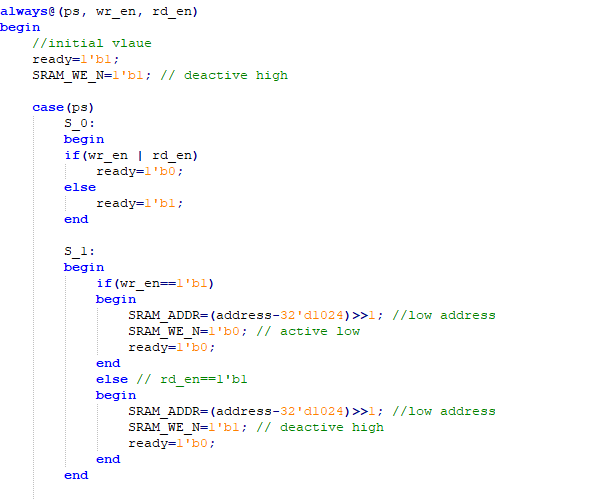
ماژول دسترسی به حافظه را در 6 سیکل مدیریت می نماید( طبق مطالب عنوان شده در صورت آزمایش)

صفر: تا زمانی که سیگنال های و یا فعال نشده اند، در این باقی می مانیم.  
این بدین معنی است که پردازنده قصد دسترسی به حافظه را ندارد. ولیکن به محض فعال شدن هر کدام از این سیگنال ها در لبه‌ی بالا رونده کلاک وارد بعدی می شویم.

های 1 تا 5: با هربار لبه‌ی بالا رونده‌ی کلاک وارد بعدی می شویم.

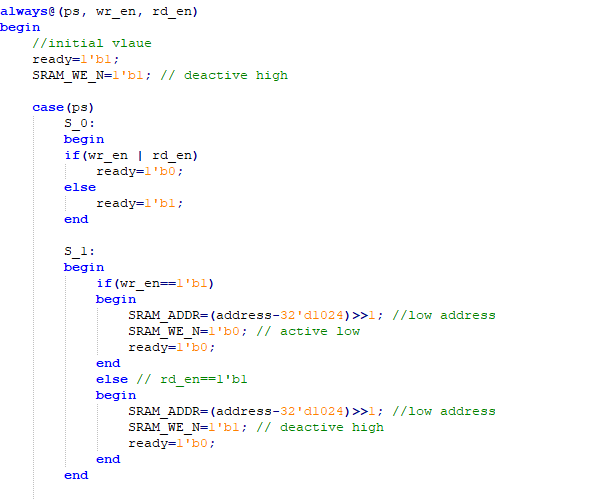


حال بررسی نماییم که در هر چه پردازشی صورت می گیرد:  
باید توجه داشت که خواندن از و نوشتن در آن به صورت 16 بیتی انجام می شود و از آنجاییکه پردازنده ی طراحی شده با داده های 32 بیتی سروکار دارد، باید این 32 بیت در چند سیکل منتقل شود.



در ابتدا سیگنال های و را به حالت غیر فعال مقدار دهی اولیه می نماییم. باید توجه داشت که سیگنال های و به صورت می باشند.

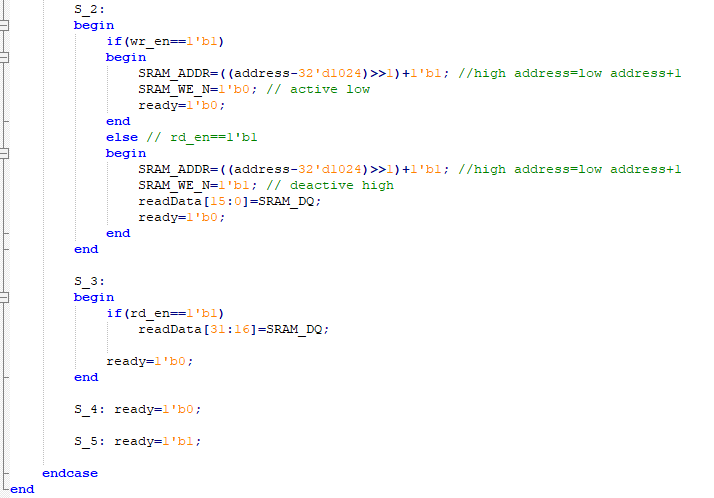
سپس در بعدی، صفر، بررسی می نماییم که اگر سیگنال های و یا فعال بودند یعنی می خواهیم دسترسی به حافظه داشته باشیم پس باید سیگنال را صفر نماییم تا در پردازنده ایجاد نماییم، در غیر این صورت مقدار سیگنال را یک می نماییم بدین معنی که دسترسی به حافظه نداریم و نمی خواهیم در پردازنده ایجاد نماییم.



در بعد، یک، مطمئن هستیم که می خواهیم دسترسی به حافظه انجام دهیم، پس تنها بررسی می کنیم که آیا می خواهیم عملیات نوشتن را انجام دهیم و یا عملیات خواندن را.

در عملیات نوشتن، سیگنال را مقدار دهی می نماییم (بر اساس سیگنال ورودی )، سپس مقدار سیگنال را صفر و یا فعال می نماییم. به علاوه سیگنال را نیز در مقدار 0 حفظ می نماییم.

اگر عملیات از نوع خواند بود: سیگنال را مقدار دهی می نماییم (بر اساس سیگنال ورودی )، سپس مقدار سیگنال را یک و یا غیر فعال می نماییم. به علاوه سیگنال را نیز در مقدار 0 حفظ می نماییم.



در بعدی، دوم، همان عملیات های یک را انجام می دهیم. تنها تفاوت هایی که وجود دارد در این است که مقدار آدرس را یک واحد زیاد می نماییم تا 16 بیت بعدی را دریافت نماییم.  
و اگر عملیات از نوع خواند باشد، مقدار داده‌ی مشخص شده توسط آدرس قبل را در قرار می دهیم(16 بیت )

تذکر: همانطور که می دانیم، در عملیات خواندن از حافظه در ابتدا آدرس مورد نظرتعیین می گردد و در لبه‌ی بالا رونده‌ی کلاک، داده‌ی موردنظر خوانده می شود.

فرایند ذکر شده برای دوم، در سوم نیز تکرار می شود.

در چهارم عملیاتی انجام نمی شود و تنها مقدار سیگنال برابر با 0 تعیین می گردد( حالت غیر فعال)

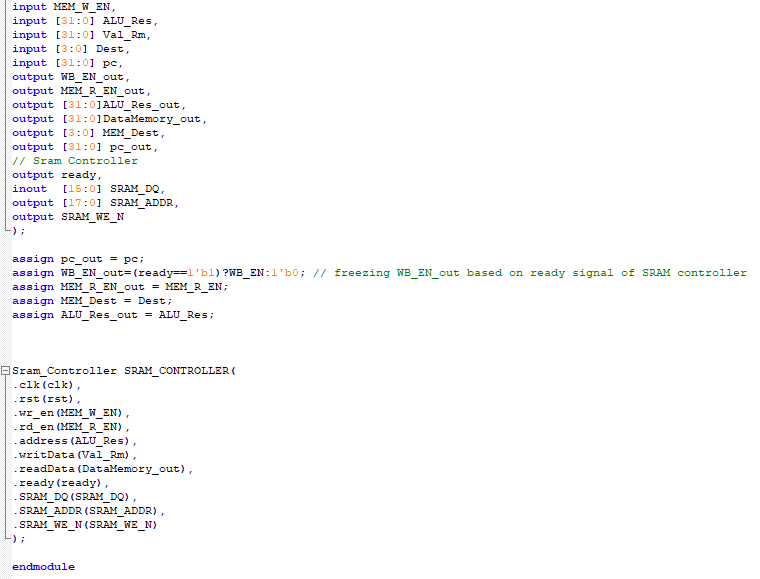
در state پنجم، عملیات دسترسی به حافظه تکمیل شده است، بنابراین مقدار سیگنال را 1 می نماییم تا پردازنده داده ی موردنظر را دریافت نماید و دچار نگردد.

در هنگام نوشتن داده در باید مقدار را به صورت 16 بیت، 16 بیت درچند برروی پورت قرار دهیم. پورت از جنس می باشد و نمی توان آن را در مقدار دهی نمود، پس با استفاده از و تعیین شرط های مناسب مقدار را بر روری قرار می دهیم.

قطعه کد زیر نحوه‌ی این فرایند را نشان می دهد:



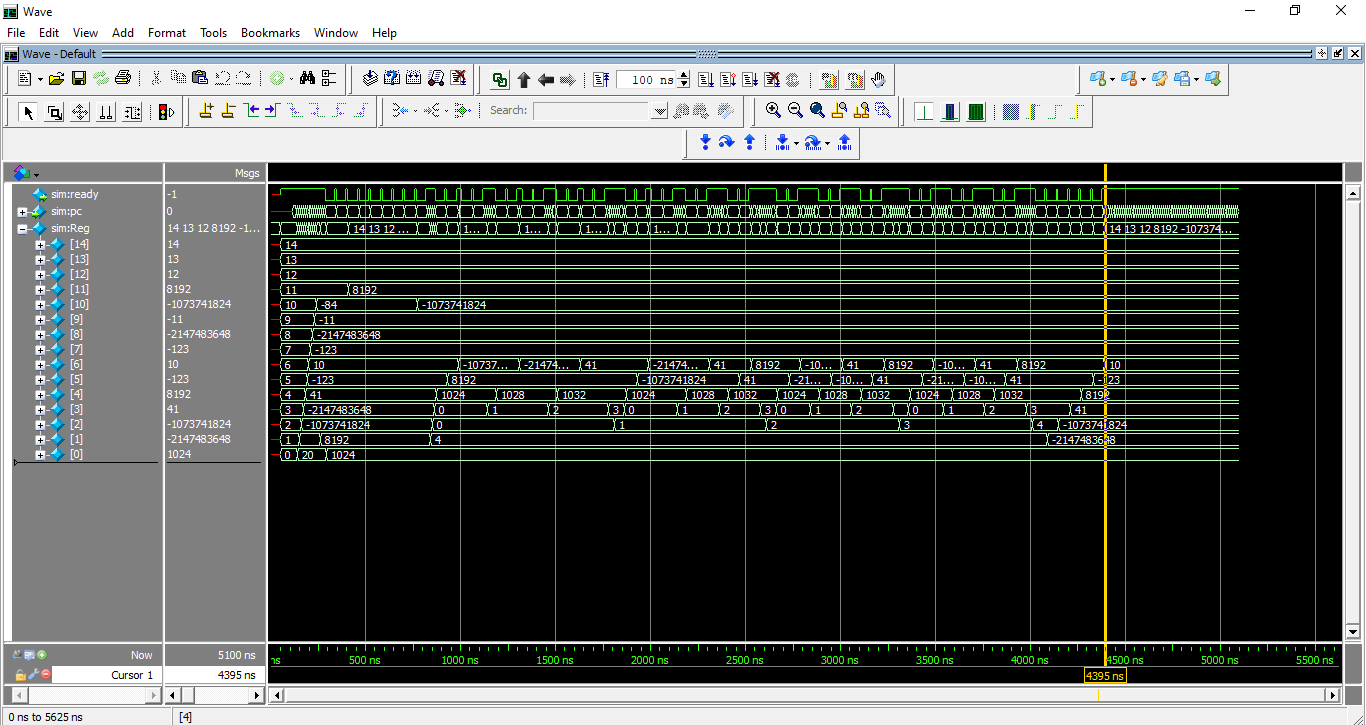
در انتها باید ماژول های و را به یکدیگر متصل نماییم و سیگنال را نیز به های دیگر پردازنده ارسال نماییم تا بتواند در مواقع دسترسی به حافظه آن ها را نماید.



پس برای متصل نمودن ماژول‌های و آن ها را درون

فراخوانی می نماییم و سیگنال های موردنظر را نیز برای آن ها تعریف می نماییم. قطعه کد بالا این فرایند را نشان می دهد.

اجرای برنامه‌ی محک و بررسی صحت عملکرد پردازنده:

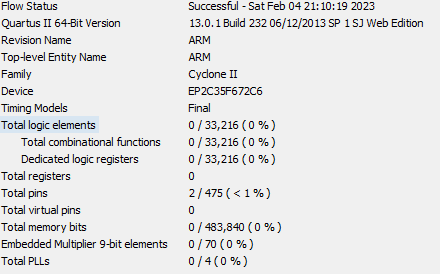


همانطور که از تصویر بالا مشخص است، خروجی برنامه محک مطابق با انتظار، صحیح می باشد.

به میزان طول کشیده است تا تمامی خروجی های برنامه‌ی محک آماده شوند؛ این درحالی است که مرحله‌ی قبل، یعنی پردازنده به همراه حافظه ایده آل ، پس از خروجی آماده می شود.

این یعنی پردازنده به همراه به میزان نسبت پردازنده به همراه حافظه ایده آل **کاهش**  داشته است.

نتایج سنتز پردازنده طراحی شده تا بدین مرحله:



همانطور که از نتایج سنتز مشخص است، تعداد المان های به کار رفته در پردازنده طراحی شده (تا بدین جا)

افزایش یافته است، که البته منطقی نیز می باشد، زیرا در این مرحله 2 ماژول جدید تحت عنوان و به پردازنده قبلی اضافه شده است که منجر به پیچیدگی سخت افزاری و افزایش تعداد المان های به کار رفته می شود.

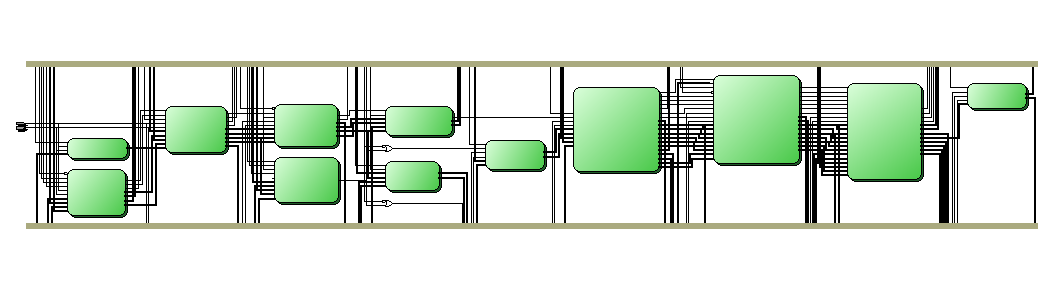
**بخش امتیازی:**

به منظور افزایش کارایی پردازنده ( با وجود و بدون ) می توانیم از رویکرد  
 بهره ببریم. بدین ترتیب که اگر برای انجام عملیاتی نیاز داریم ابتدا داده ی حافظه را بخوانیم و سپس آن را در ذخیره نماییم و توسط دستور بعدی بر روی داده ی مورد نظر پردازش انجام دهیم. به جای این کار، یک دستور جدید به پردازنده معرفی نماییم که هنگامی که داده ی موردنظر را از حافظه می خواند پردازش مورد نظر را نیز انجام دهد و بدین ترتیب در زمانی کمتر و تنها توسط یک دستور عملیات موردنظر انجام می شود. البته نیازمند اضافه کردن واحد محاسبات () به و تغییر در می باشیم.

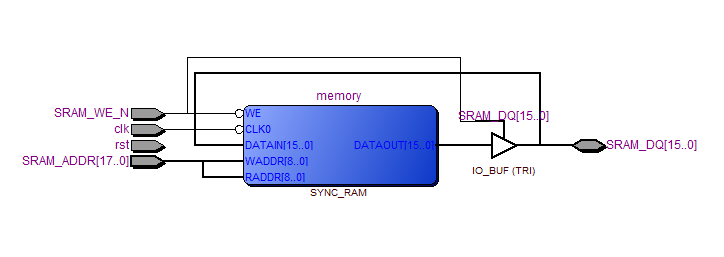
استفاده از حافظه نهان () در پردازنده :

از انجاییکه استفاده از منجر شد تا کارایی پردازنده طراحی شده کاهش یابد، به عنوان یک راه حل به منظور افزایش کارایی، از به عنوان حافظه‌ی نهان استفاده می نماییم.

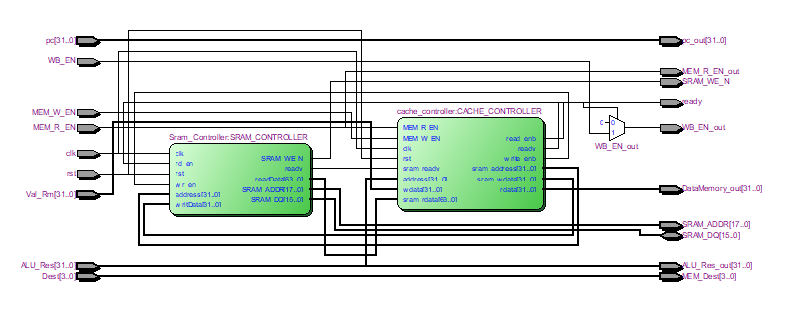
کلی پردازنده:



مربوط به :

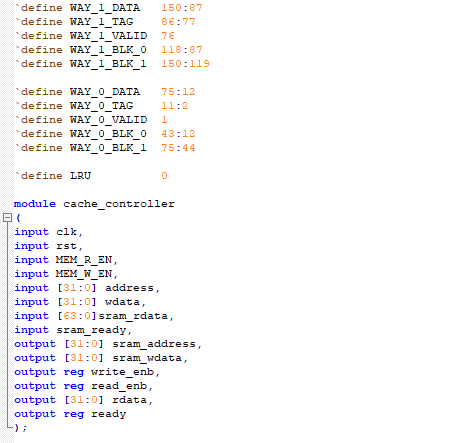
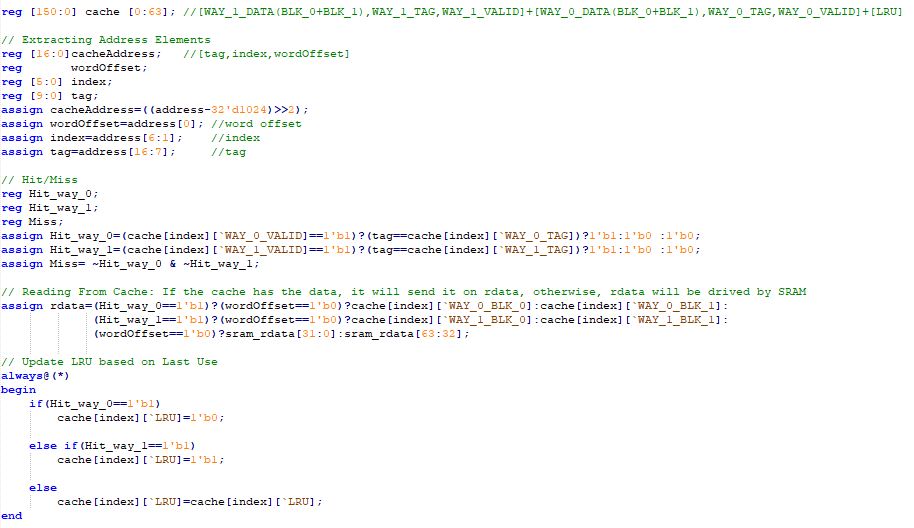


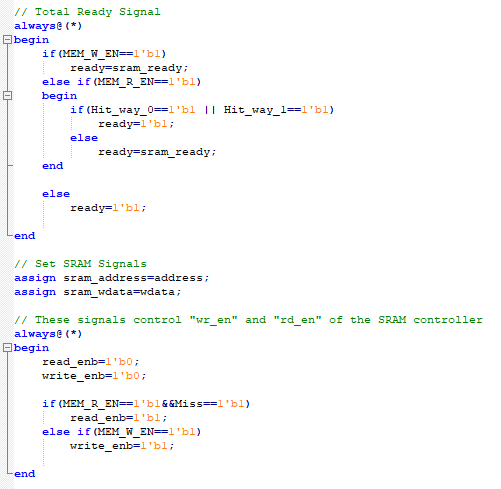
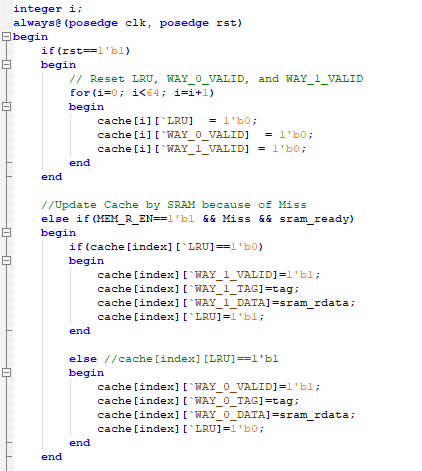
مربوط به و :

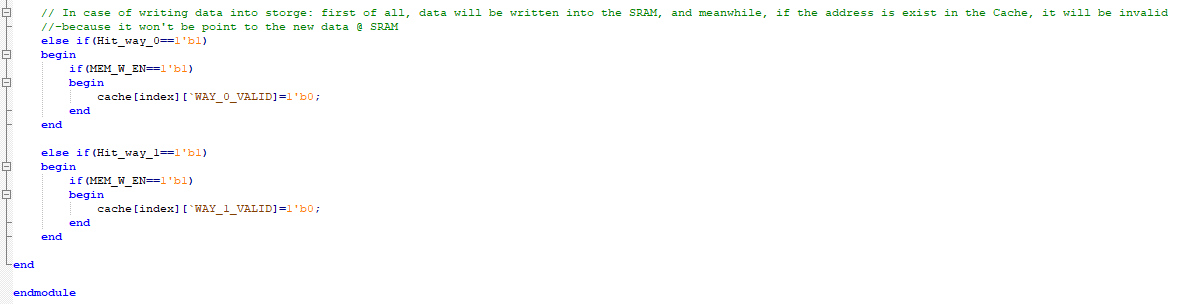


حال بدین منظور ماژولی تحت عنوان r به پردازنده اضافه می نماییم:

قطعه کد زیر ماژول نشان می دهد:



حال به تفصیل هر بخش را توضیح می دهیم:

در ابتدا یکسری پارامتر تعریف می نماییم، که حاوی اعدادی است که به دفعات در کد مورد نظر استفاده می شود.

سپس ورودی ها و خروجی های ماژول تعریف می گردند.  
کلاک و ریست برای همگام سازی این ماژول با باقی بخش های پردازنده.

و : سیگنال های ورودی برای مشخص کردن عملیات خواندن و یا نوشتن

: سیگنال ورودی برای مشخص کردن آدرس محل خواندن یا نوشتن

: سیگنال ورودی، مشخص کننده‌ی داده ای که می خواهیم در حافظه نوشته شود.

: مقدار خوانده شده از   
باید توجه داشت که مقدار این سیگنال ورودی برابر با 64 بیت است که علت آن در این است که با هربار اس که رخ می دهد به جای آن که تنها 32 بیت داده از بخوانیم، 64 بیت می خوانیم. با اینکار می خواهیم احتمال رخ دادن را کاهش دهیم علاوه بر اینکه داده‌ی موردنظر را داریم، داده های کناری آن را نیز داشته باشیم.

: سیگنالی که نشان می دهد اگر رخ داد یا خواستیم در بنویسیم، پردازنده تا چه زمانی باید باشد.

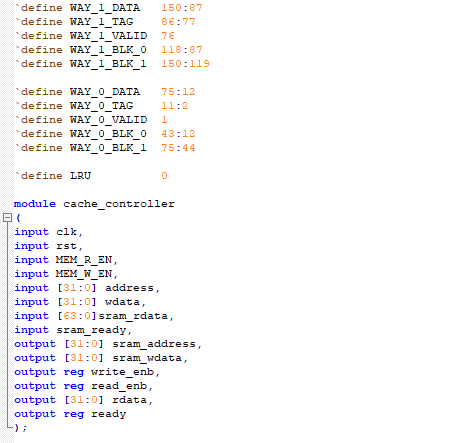
: آدرسی که مخواهیم داده را از آن بخوانیم یا در آن بنویسیم.

: داده ای که می خواهیم در بنویسیم.

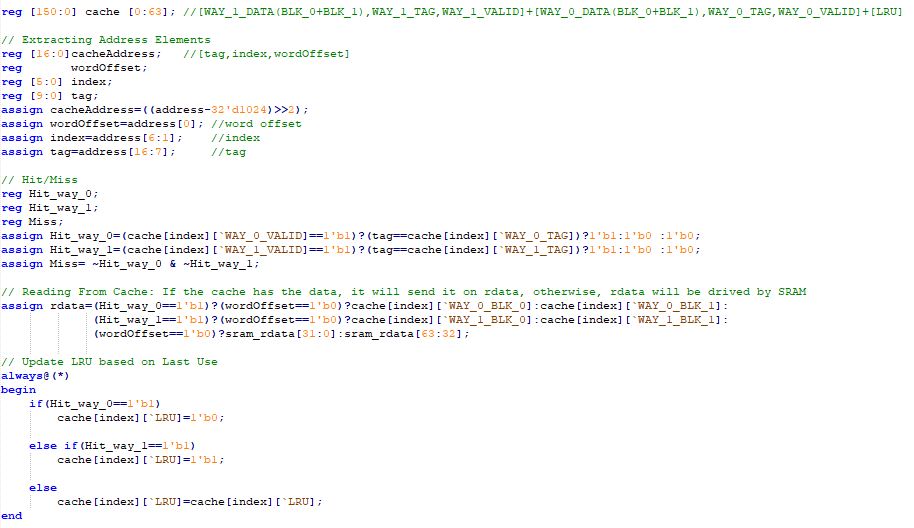
و : سیگنال های خروجی عملیات خواندن و نوشتن که میان و تبادل می شوند.

: سیگنال خروجی که داده‌ی حاصل از عملیات خواندن را مشخص می نماید.

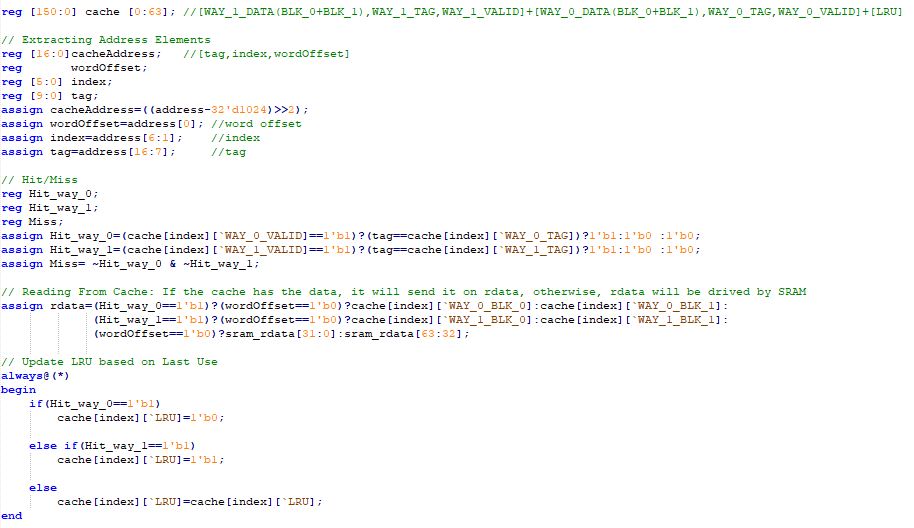
: سیگنالی برای مشخص کردن اینکه ایا عملیات به صورت کامل انجام شده است یا خیر.



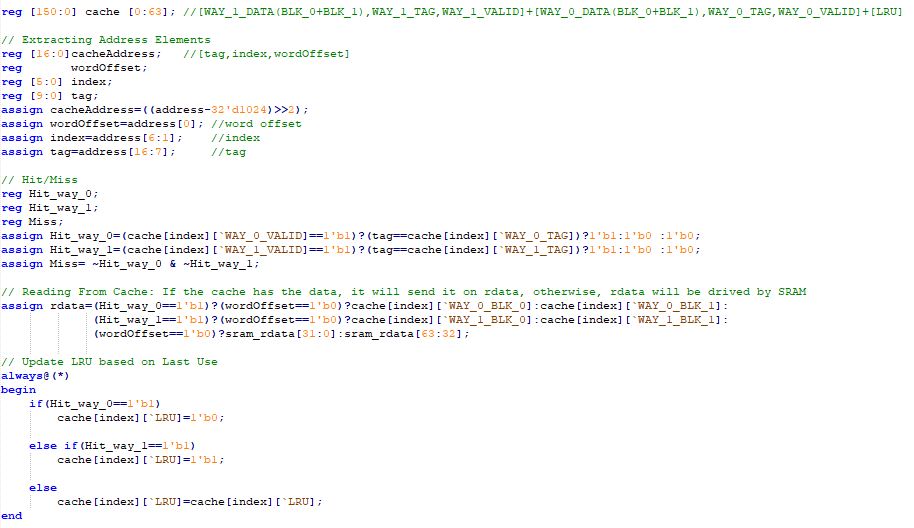
در ادامه حافظه‌ی کش، به صورت 151 خانه‌ی 64 بیتی تعریف می گردد.



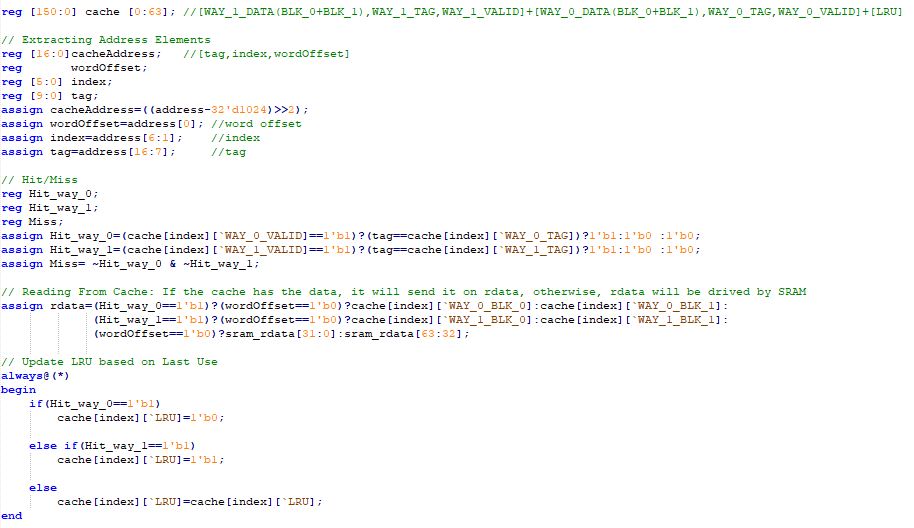
سپس و و و براساس مطالب عنوان شده در صورت آزمایش تعیین می گردد.



درادامه طبق کد زیر، مشخص می شود که آیا رخ می دهد یا و اگر رخ می دهد در کدامین داده‌ی موردنظر وجود دارد:



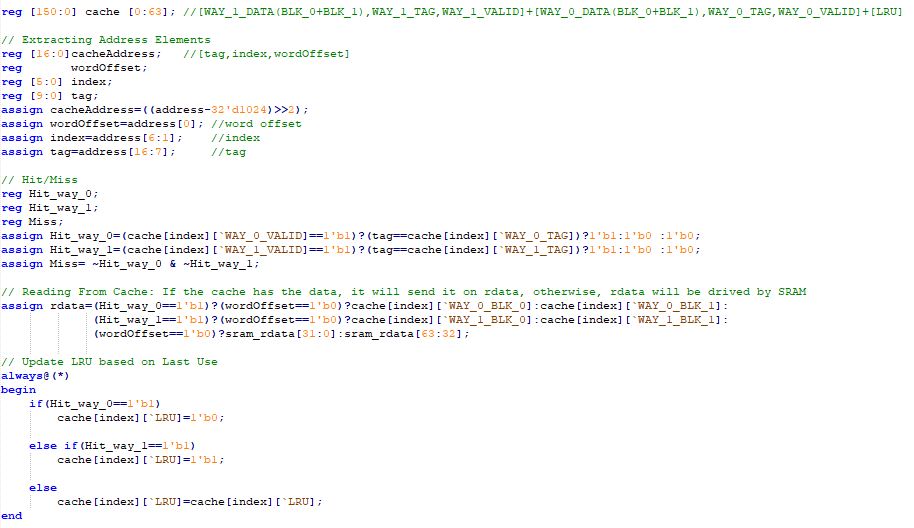
سپس عملیات خواندن از کش مدیریت می شود:



اگر داده در کش موجود بوده باشد و یا به عبارتی رخ داده باشد، وابسته به اینکه در کدام یک از بلاک ها رخ داده و چه می باشد، داده ی مورد نظر از کش خوانده می شود.

ولی اگر رخ داده باشد داده ی موردنظر از خوانده می شود( داده توسط سیگنال درایو می شود.)

با توجه به اینکه اگر رخ دهد نیازمند جیاگزینی داده ها می باشیم، با توجه به سیاست جایگرینی عمل می نماییم، وابسته به اینکه آخرین مرتبه در کدام یک از بلاک ها رخ داده است، سیگنال آن را فعال و برای دیگری را غیر فعال می نماییم.

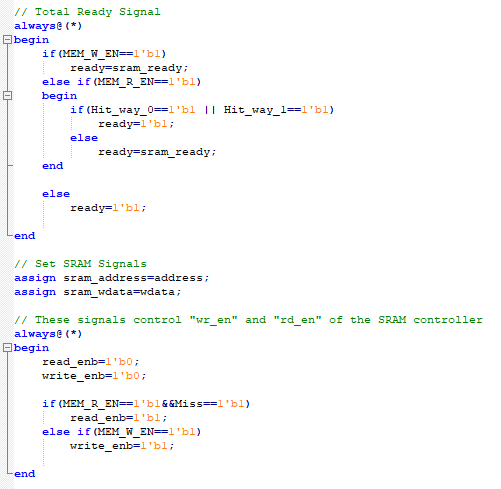


قطعه کد زیر نحوه‌ی مدیریت سیگنال را نشان می دهد:

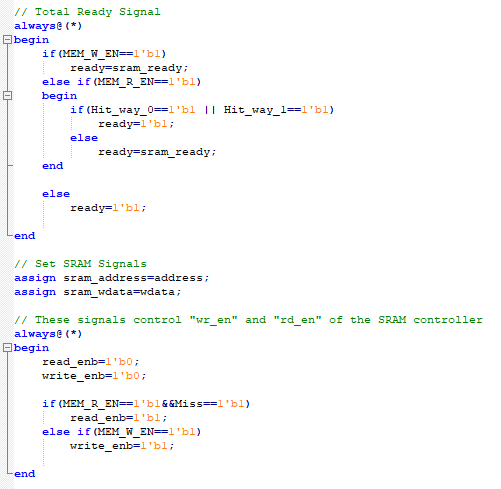
وابسته نوع دسترسی ای که به حافظه داریم( خواندن یا نوشتن) سیگنال را مشخص می نماییم.

اگر می خواهیم در حافظه بنویسیم به 6 سیکل کلاک نیاز داریم، پس باید 6 سیکل پردازنده را کنیم.

اگر می خواهیم از حافظه بخوانیم، پس باید بررسی شود که آیا رخ داده است یا ، اگر رخ دهد نیز باید 6 سیکل پردازنده شود تا داده ی مورد نظر از خوانده شود ولی اگر رخ دهد می توانیم داده موردنظر را در همان سیکل بخوانیم و نیازی با نباشد.

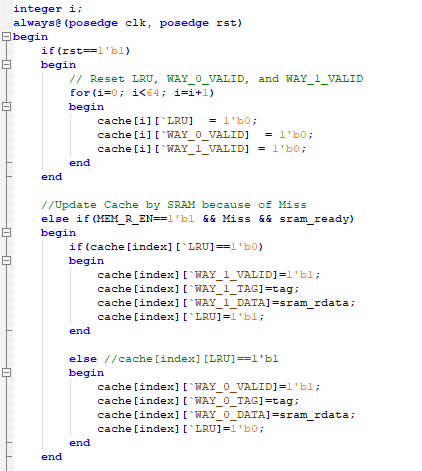


در ادامه سیگنال های کنترلی و و و که میان و در زمان هایی که رخ می دهد یا می خواهیم در حافظه عملیات نوشتن را انجام دهیم، تبادل می شوند را مدیریت می نماییم.



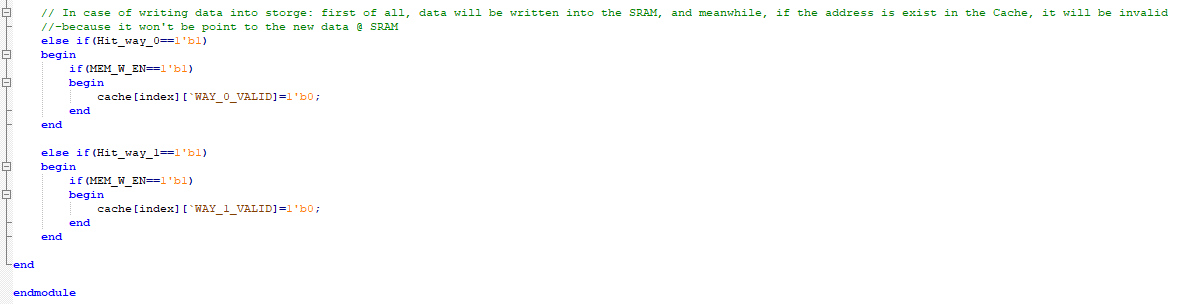
درقطعه کد زیر اگر سیگنال ریست آمده باشد، مقادیر موجود در کش را پاک می نماییم ( تمامی موارد 0 می شوند.)

سپس بررسی می شود، اگر رخ داده باشد، مقدار موردنظر از خوانده می شود و سپس کش می شود.



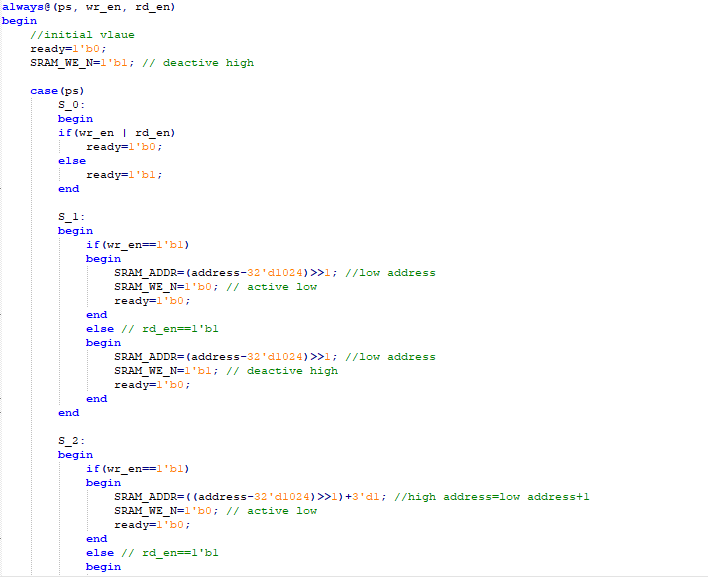
در ادامه قطعه کد زیر عملیات نوشتن در حافظه را مدیریت می نماید:

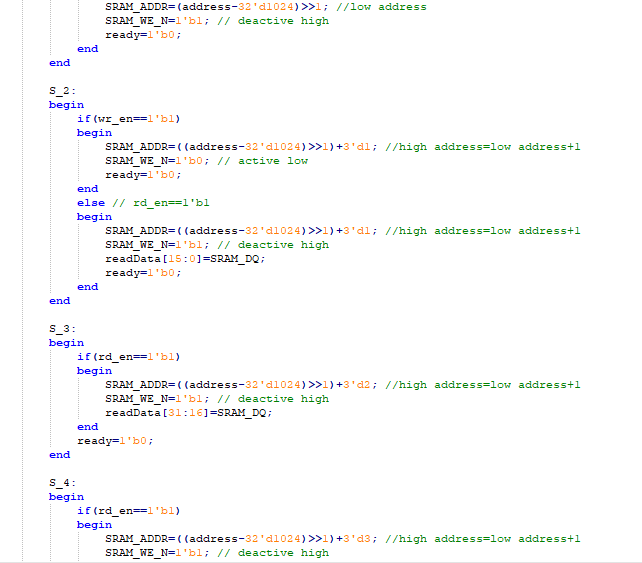
اگر دسترسی به حافظه از نوع نوشتن باشد، ابتدا داده‌ی مورد نظر در نوشته می شود و سپس اگر آدرس موردنظر درون کش موجود بود، داده‌ی آن پاک می شود زیر دیگر حاوی مقداری درستی نخواهد بود.

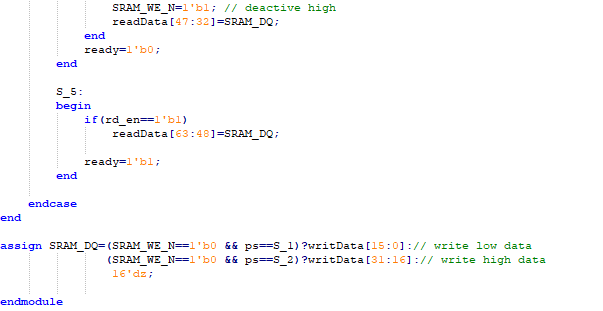


همانطور که پیشتر نیز ذکر گردید، هنگامی که رخ می دهد، 64 بیت داده‌ی از می خواند، ولیکن با توجه به طراحی شده، تنها می توان 32 بیت داده از خواند. بنابراین نیازمند آن هستیم که SRAM Controller را تغییر دهیم تا بتوانیم 64 بیت داده از بخوانیم:

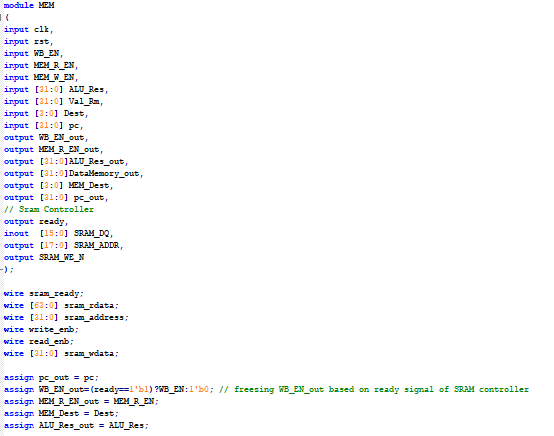
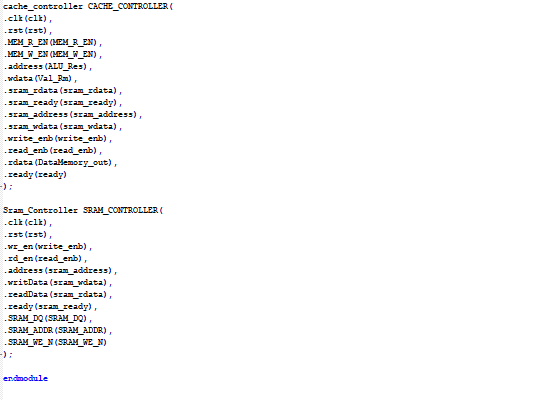
بدین منظور عملیات خواندن را در های 4 و 5 از ادامه می دهیم. پس در نهایت در های 2 و 3 و 4 و 5 عملیات خواندن انجام می شود و در هر مرتبه 16 بیت خوانده می شود که مجموعا 64 بیت داده از خواند می شود:



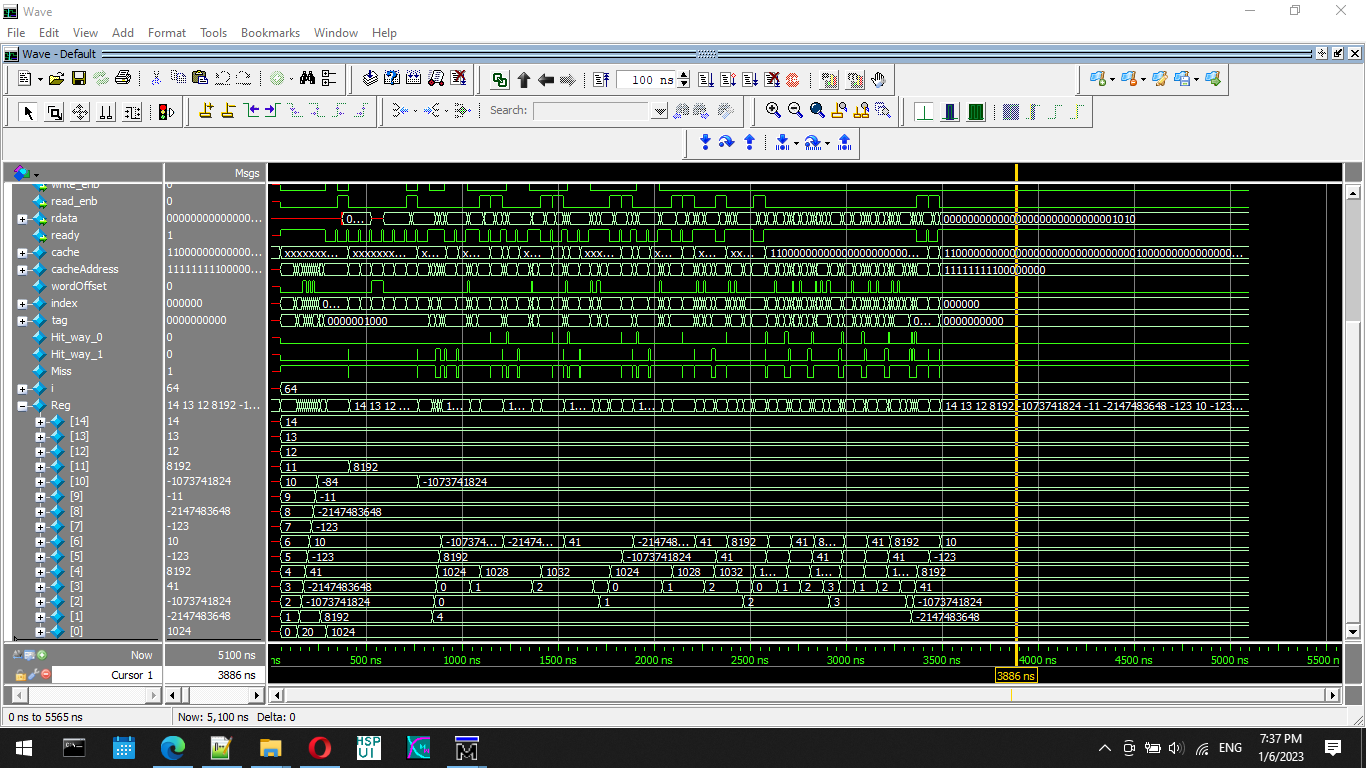




حال باید ماژول و را در به یکدیگر متصل نماییم:

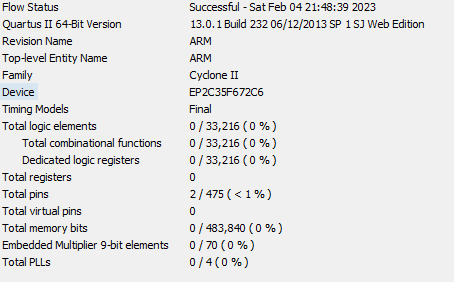
اجرای برنامه‌ی محک و بررسی صحت عملکرد پردازنده:



همانطور که از تصویر بالا مشخص است، خروجی برنامه محک مطابق با انتظار، صحیح می باشد.  
به میزان طول کشیده است تا تمامی خروجی های برنامه‌ی محک آماده شوند؛ این درحالی است که مرحله‌ی قبل، یعنی پردازنده به همراه ، پس از خروجی آماده می شود.

این یعنی پردازنده به همراه به میزان نسبت به پردازنده به همراه بهبود داشته است.

نتایج سنتز پردازنده طراحی شده تا بدین مرحله:



همناطور که از نتایج سنتز مشخص است، تعداد المان مورد استفاده توسط پردازنده طراحی شده ( تا بدین جا)، افزایش یافته است که البته مطابق با انتظار می باشد. منطقا با اضفه شده به پردازنده طراحی شده در مرحله‌ی قبل، تعداد المان های مورد استفاده افزایش می یابد.

**بخش امتیازی:**

در مکانیزم هنگامی که داده ای به دفعات تغییر می کند، کارای پردازنده افت کاهش می یابد زیرا زمان زیادی را برای هر تغییر مقدار دورن را نیز تغییر می دهد. ولی اگر از مکانیزم

استفاده نماییم، حتی اگر داده ای به دفعات نیز تغییر کند این تغییرات فقط در کش ثبت می شود و فقط هنگامی که قرار است از کش حذف شود، را آپدیت می نماید. در این حالت به علت سرعت بالای کش کارایی پردازنده کاهش کمتری نسبت به حالت خواهد داشت.

راه حل دیگر به منظور افزایش کارایی پردازنده، استفاده از چندین کش به صورت موازی است، بدین صورت که داده ی مورد نظر در چندین کش جست ‌و جو شود. بدین ترتیب می توانیم داده های زیادی را در کش ذخیره نماییم و از آن جایی که کش ها به صورت موازی عملیات را انجام می دهند، سرعت  
جست و جو و احتمال شده افزایش می یابد.

راه حل دیگر افزایش ساز کش می باشد، مثلا از کشی با تعداد بالاتر استفاده نمود.