**Cache:**

در داخل کش یک حلقه‌ی بینهایت وجود دارد که با لبه‌ی بالارونده‌ی کلاک کار می‌کند و به نسبت مقادیر comp و write استیت فعلی کش را مشخص می‌کند و تسک مناسب آن استیت را اجرا می‌کند و داخل تسک‌های ۴ استیت همان کار‌هایی که در تعریف پروژه آمده است نوشته شده است.

task validzero:

این تسک مواقعی اجرا می‌شود که سیگنال ریست برابر یک شود و وقتی اجرا شود اعتبار تمام خانه‌های کش را ۰ کرده و در مواقع به معنی خالی شدن کش است.

**Testbench:**

مدیریت کننده‌ی کش و از طریق ورودی‌های آن تست بنچ است. در تست بنچ روند عادی این‌گونه است که آرایه‌ای از comp و writeهایی که به ترتیب قرار است به کش داده شوند وجود دارند و آرایه‌ای نیز از address‌های حافظه‌ای که به کش داده می‌شوند وجود دارد و با هر لبه‌ی بالارونده‌ی کلاک داده‌های این آرایه ها به ترتیب به کش داده می‌شوند و کش عملیات مورد نظر را انجام می‌دهد. ( داده‌های درون رم نیز در ابتدا با اعداد تصادفی پر می‌شوند ).

اما ۳ حالت خاص وجود دارد که نیاز به بیش از روند عادی ورودی دادن به کش دارند. اگر در حالت ۲ داده میس بخورد باید به طور مستقیم درون رم ریخته شود در نتیجه این حالت با یک if درون تست بنچ هندل شده است.

دو حالت دیگر خاص حالت داده نوشتن یا خواندن از کش هستند. به این معنی که وقتی می‌خواهیم یک ورد را درون کش بریزیم باید توجه کنیم که کل بلاک آن باید انتقال یابد. در نتیجه ی برای هندل این اتفاق، از تعدادی متغیر تحت کاربرد flag استفاده شده است که وقتی یک ورد داخل کش می‌رود تا ۴ لبه‌ی بالارونده‌ی بعدی کلاک باقی ورد‌های بلاک مورد نظر نیز به کش منتقل شوند. از طرف دیگر وقتی بخواهیم داده‌ای را داخل کش بریزیم و بیت dirty داده‌ای که قبلاً داخل کش در همان سطر بوده است ۱ باشد باید تمام بلاک داخل کش را به رم منتقل کنیم و سپس کل بلاک داده‌ی جدید را درون کش بریزیم، که این حالت نیز مانند حالت خاص قبل با تعدادی متغیر تحت عنوان flag هندل شده است.

از طرف دیگر چون به طور منطقی بعد از استیت ۳ در پروژه باید استیت ۴ بیاید. این حالت نیز لحاظ شده است، که بعد از استیت ۳ تست بنچ اتوماتیک استیت را به ۴ تغییر می‌دهد و تحت ۴ لبه‌ی بالارونده‌ی کلاک کل بلاک را به داخل کش انتقال می‌دهد.