

دانشكده مهندسي برق

گزارش پروژه درس VHDL

راه اندازی و شبیه سازی واحد SPI قطعه ADF4351

علی رضازاده - ۴۰۱۶۱۱۱۴۸

استاد:

دكتر ستار میرزاکوچکی

دی ماه 1401

**فهرست مطالب**

۱**: مقدمه ............................................................................................................................... ۳**

**۲: تشریح کد ....................................................................................................................... ۴**

**۲-۱: موجودیت .................................................................................................................................... ۴**

**۲-۲: معماری ........................................................................................................................................ ۵**

**۳:تست بنچ .......................................................................................................................... ۹**

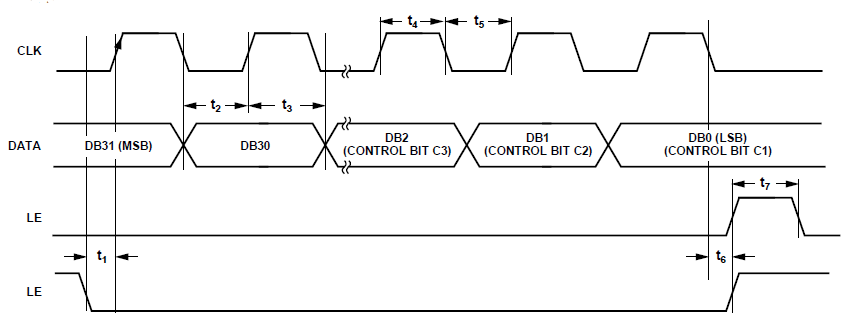
**۴:شبیه‌سازی ...................................................................................................................... ۱۰**

**۱: مقدمه**

در این بخش، به توضیحاتی ابتدایی در رابطه با واحد SPI قطعه ADF4351 می‌پردازیم.

ارتباط سریال این قطعه از ساختار ساده‌ای تشکیل شده است. پايه‌های مربوط به ارتباط سریال این قطعه عبارتند از CLK، LE و DATA. یکی دیگر از پایه‌های مهم مربوط به CE می‌باشد که در صورت فعال نبودن قطعه کاری انجام نخواهد داد.

قطعه ADF4351 دارای یک رجیستر ۳۲ بیتی بوده که اطلاعات دریافتی از طریق ارتباط سریال را در آن ذخیره می‌کند، سپس با توجه به بیت‌ها اطلاعات مورد نظر را به رجیستر دیگری فرستاده و یا کار خاصی انجام می‌دهد که مربوط به حوزه بررسی ما نمی‌باشند.

وظیفه پایه LE نیز به این شکل است که هنگامی که مقدار ‘0’‌رو آن قرار گرفت، ارسال ۳۲ بیت شروع شده و پس از آخرین کلاک دوباره مقدار ‘1’ روی آن قرار می‌گیرد. شکل زیر نحوه عملکرد پایه‌ها را به خوبی نشان می‌دهد.

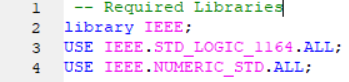
شکل ۱) دیاگرام زمانی مربوط به پایه‌های ارتباط سریال

طبق زمان بندی‌های مشخص شده در جدول دیتا شیت، دوره تناوب مناسب برای این واحد ۵۰ نانو ثانیه بوده که معادل فرکانس کاری 20 Mbps برای واحد SPI می‌باشد.

**۲: تشریح کد**

کد مربوطه در برنامه ISE از شرکت Xilinx نوشته و شبیه‌سازی شده است.

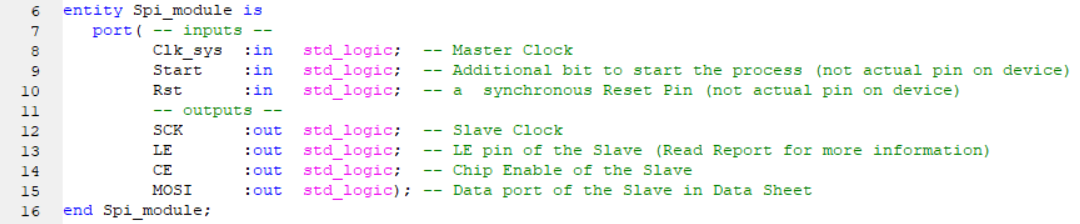
در اولین بخش کد کتابخانه‌های مورد نیاز برای انجام پروژه را اضافه می‌کنیم. در شکل زیر این کتابخانه‌ها قابل مشاهده می‌باشند.



شکل ۲) کتابخانه‌های افزوده شده

**۲-۱: موجودیت**

در این بخش ابتدا کتابخانه استاندارد IEEE افزوده شده، سپس با افزودن بخش‌های بعد قابلیت استفاده از تایپ‌های Std\_Logic و عملیات‌های محاسباتی را فراهم ساخته‌ایم.

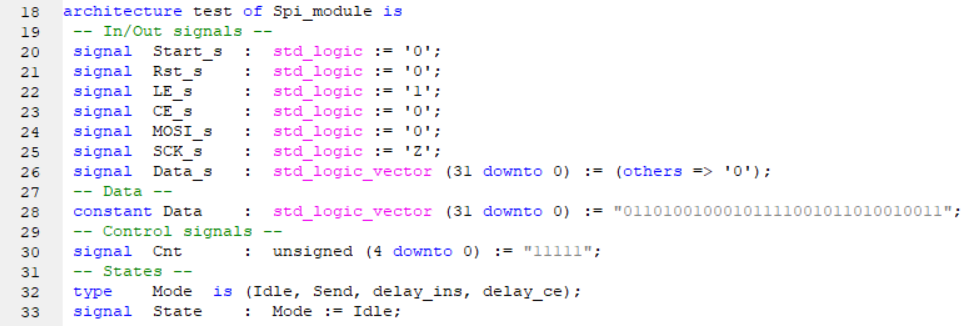
بخش بعدی کد مربوط به موجودیت پروژه بوده که در آن پورت‌‌های مورد نظر را تعریف می‌کنیم. پورت‌های Start‌ و Rst تنها برای بررسی شبیه‌سازی بوده و در قطعه اصلی وجود خارجی ندارند. با ‘1’‌شدن Start‌ شبیه سازی آغاز شده و Rst نیز عملیات ارسال را متوقف می‌کند.

شکل ۳) موجودیت واحد SPI نوشته‌ شده

\*توضیحات مربوط به باقی پایه‌ها در بخش مقدمه داده شده است.

پس از پایان بخش موجودیت، بخش معماری کد نوشته شده را مورد بررسی قرار می‌دهیم.

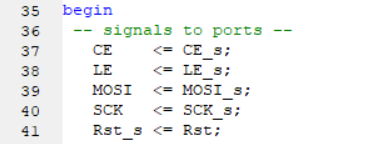
**۲-۲: معماری**

بخش معماری به دو قسمت تقسیم می‌شود. بخش ابتدایی مربوط به تعریف سیگنال‌ها و مشخص کردن برخی از پارامتر‌ها بوده که با نام Declarative Part‌ شناخته می‌شود.

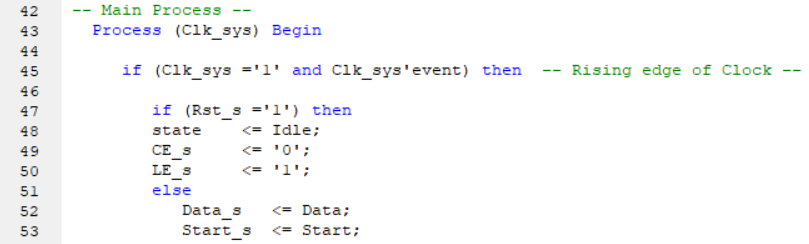
شکل ۴) بخش Declarative معماری

در قسمت ابتدایی،‌ سیگنال اتصالی پورت‌ها تعریف شده است. این عملیات در شبیه‌سازی تأثیر خاصی نداشته اما در بهبود عملیات روتینگ توسط خود برنامه اثر قابل توجهی دارد. بیت‌هایی که قرار است ارسال شوند توسط دستور constant به شکل یک مقدار ثابت تعریف شده که بعدا آن را روی سیگنال Data\_s قرار می‌دهیم. سیگنال Cnt نیز یک عدد ۵ بیتی با مقدار اولیه ۳۱ بوده که تعداد بیت‌هایی که قرار است ارسال شوند را مشخص می‌کنند. پس از ارسال هر بیت مقدار آن یکی کم می‌شود تا در نهایت برابر صفر بشود. صفر شدن این سیگنال تمام شدن عملیات ارسال را نشان داده و همانطور که مشاهده کردیم باید پس از آن سیگنال مربوط به LE دوباره ‘1’ بشود. این موارد را در بخش‌های بعد مشاهده خواهیم کرد. در بخش States ابتدا تایپی جدید به نام Mode را تعریف کرده که نشان دهنده هر مرحله از انجام کل عملیات می‌باشد. سیگنال State نیز از همین تایپ تعریف شده و مقدار اولیه Idle را به آن داده‌ایم. هنگامی که در حالت Idle هستیم هیچ کار خاصی را انجام نمی‌شود و تنها سیگنال‌‌های کنترلی برای شروع عملیات بررسی می‌شوند.

بخش بعدی نیز مربط به قسمت Instantiation بوده که بیشتر بخش کد را به خود اختصاص داده است. این قسمت را ریز به ریز مورد بررسی قرار می‌دهیم.



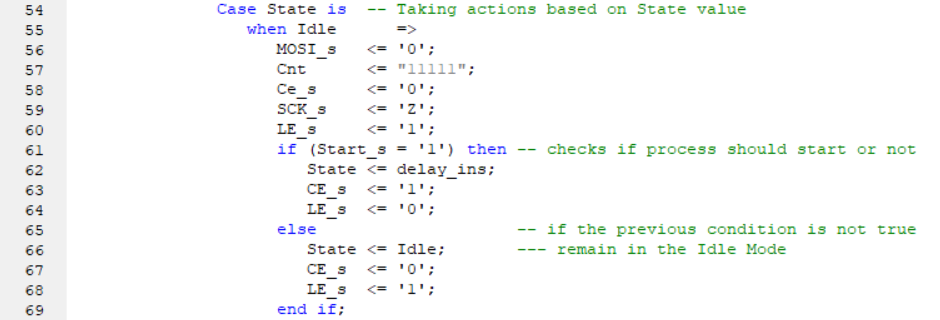
شکل ۵) شروع بخش Instantiation

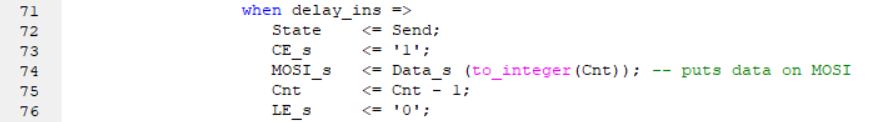
دستور Begin نشان دهنده شروع این بخش از معماری می‌باشد، سپس سیگنال‌های تعریف شده در بخش پیشین را به پورت‌های مربوطه متصل کرده‌ایم.

شکل ۶) شروع بخش Process

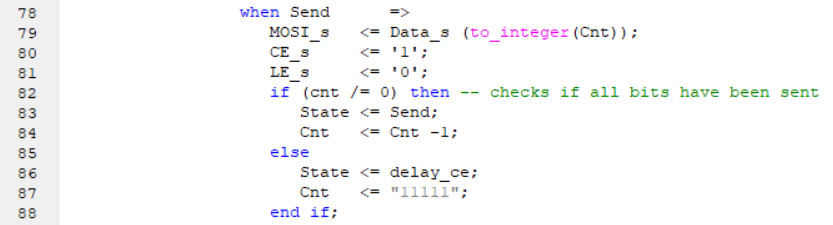
بدنه اصلی کد درون یک Process قرار دارد که حساس به تغییرات کلاک سیستم می‌باشد. شرط درون دستورif‌ بالا رونده بودن لبه کلاک را بررسی می‌کند و در صورت بر قراری شرط دستورات درون آن اجرا می‌شود. در اولین مرحله بالاترین اولویت بررسی مربوط به Reset سنکرون می‌باشد. در صورتی که این سیگنال مقداری برابر ‘1’‌داشت، سیگنال‌های اصلی را ریست کرده و به حالت Idle که حالت اولیه می‌باشد برمی‌گردیم و تا زمانی که دوباره دستور Start داده نشود در این حالت باقی می‌مانیم.

در صورتی که شرط Reset‌ برقرار نبود دستور Case مورد بررسی قرار می‌گیرید. دستور Case حالت‌‌های مختلف سیگنال State‌ را چک کرده و با توجه به حالتی که در آن قرار دارد دستورات مربوطه را اجرا می‌کند. سیگنال State‌ در حالت اولیه خود Idle می‌باشد. در این State سیگنال‌ها در حالت اولیه خود قرار داشته و شمارنده مقدار اولیه خود را دارد. در صورت ‘1’‌شدن مقدار سیگنال Start، فرایند ارسال دیتا شروع می‌شود. در این مرحله با ‘1’ کردن CE و ‘0’ کردن LE فرایند را آغاز کرده و سپس وارد حالت Delay\_ins می‌شویم. در صورت ‘0’ بودن Start‌ نیز در حالت اولیه Idle باقی می‌مانیم.

شکل ۷) بررسی حالت Idle

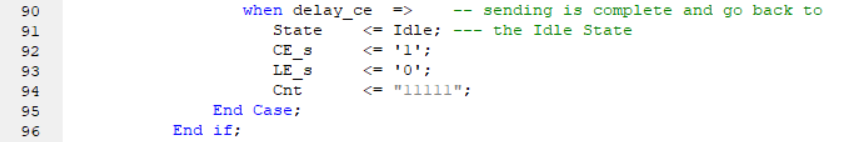
در مرحله Delay\_ins اولین بیت آماده ارسال شده و روی MOSI قرار می‌گیرد، یکی از شمارنده کم شده و State وارد مرحله Send‌ می‌شود. در صورت نبود این مرحله یک بیت کمتر برای Slave فرستاده خواهد شد.

شکل ۸) بررسی حالت Delay\_ins

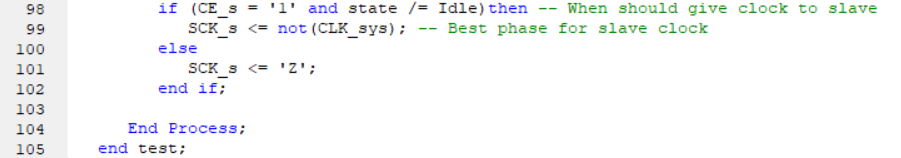
در مرحله Send یک شرط برای بررسی مقدار شمارنده وجود دارد و عملیات انتقال را تا زمانی که به صفر نرسیده انجام می‌دهد. دستور to integer مقدار binary این شمارنده را به integer تبدیل می‌کند. زمانی که مقدار Cnt برابر صفر شد آنرا به مقدار اولیه خود باز گردانده و وارد مرحله نهایی یعنی Delay\_ce‌ می‌شویم.

شکل ۹) بررسی حالت Send

مرحله Delay\_ce نیز زمان کافی برای انتقال آخرین بیت را فراهم کرده و دقت شود که سیگنال‌های کنترلی مقدار خود را حفظ می‌کنند. پس از این مرحله دوباره وارد حالت Idle شده تا زمانی که دوباره سیگنال Start مقدار ‘1’ گرفته و دیتای بعدی طبق مراحل شرح داده شده ارسال شود.

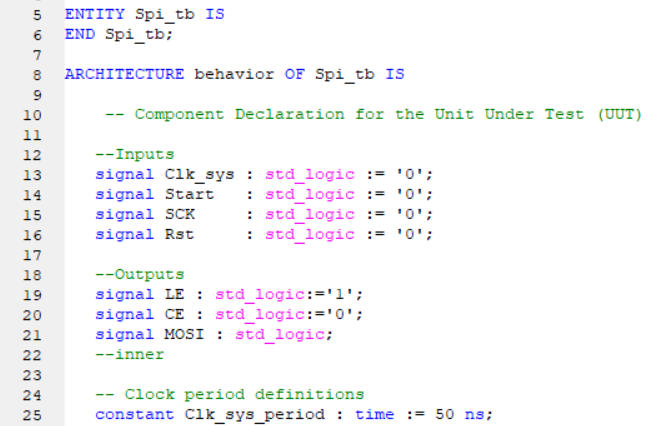
شکل ۱۰) بررسی حالت Delay\_ce

تنها بخش باقی مانده تولید کلاک Slave می‌باشد. از آنجایی که تولید کلاک باعث مصرف توان شده نمی‌توان این کلاک را همیشه فعال نگه داشت. به همین دلیل تا زمانی که در مرحله Idle هستیم کلاکی برای Slave ارسال نمی‌کنیم. با خارج شدن از حالت Idle‌ و فعال شدن CE این کلاک را با اختلاف فاز ۱۸۰ درجه نسبت به کلاک سیستم به قطعه مورد نظر می‌دهیم. این کار باعث می‌شود تا دیتا در پایدار ترین زمان ممکن توسط Slave خوانده شود.

شکل ۱۱) تولید سیگنال SCK

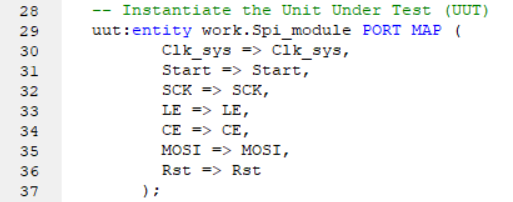
**۳: تست بنچ**

تست بنچ بر خلاف کد اصلی تنها از بخش معماری تشکیل شده و موجودیت ندارد. در این بخش به سیگنال‌ها مقادیری برای بررسی شبیه سازی و اطمینان از صحت عملکرد کد داده می‌شود. در بخش Declarative سیگنال‌ها را تعریف و مقدار اولیه می‌دهیم. دوره تناوب کلاک اصلی را نیز به شکل یک مقدار ثابت تعریف کرده‌ایم.

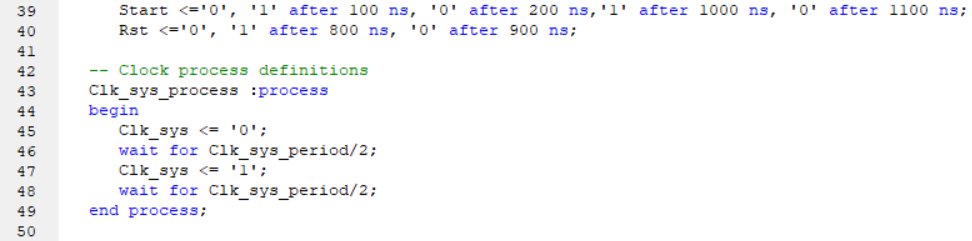


۱۲) بخش Declarative معماری تست بنچ

در بخش ابتدایی Instantiation معماری تست بنچ، قطعه طراحی شده در بخش پیشین را از کتابخانه Work فرخوانی کرده و سیگنال‌های تعریف شده را به پورت‌های معادل متصل می‌کنیم.



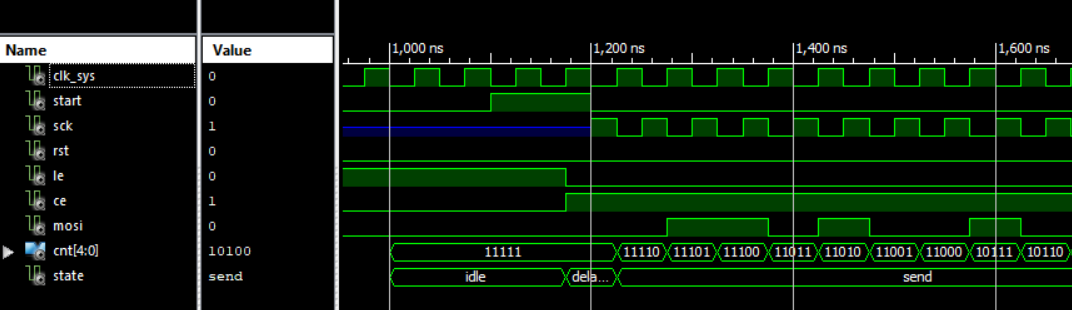
۱۳) اتصال سیگنال‌ها به پورت‌های قطعه

در آخر نیز به دو سیگنال Start و Rst سیگنال هایی را داده که در شبیه سازی می‌توان نتیجه آنرا مشاهده کرد. سیگنال کلاک سیستم نیز بر اساس دوره تناوب تعریف شده ساخته می‌شود. به این شکل که نصف دوره تناوب مقدار ‘1’ و نصف دیگر مقدار ‘0’ به آن داده شده است.

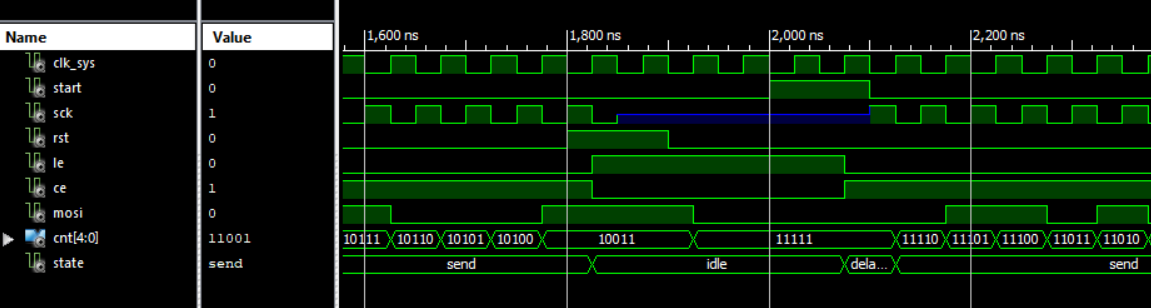
۱۴) مقدار دهی به سیگنال‌ها و کلاک سیستم

**۴: شبیه‌سازی**

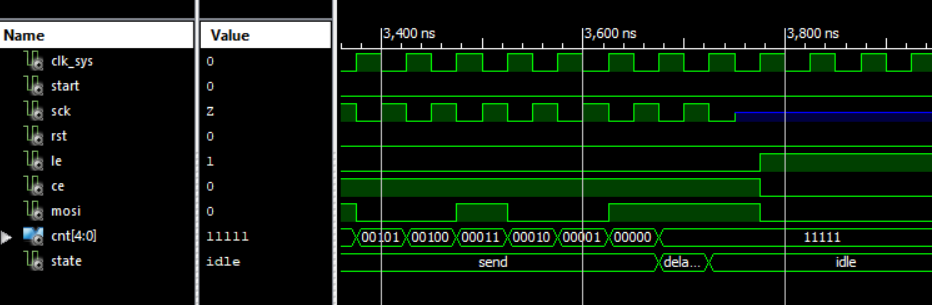
برای مشاهده نتیجه خروجی نیز تنها کافیست از بخش Simulation فایل تست بنچ نوشته شده را اجرا کرده، سیگنال‌های اضافی جهت بررسی را از بخش uut افزوده و با گذراندن زمان خروجی را مشاهده کنیم.

همانطور که در نتیجه شبیه‌‌سازی مشاهده می‌کنیم تا زمانی که سیگنال Start مقدار ‘0’ داشته در حالت Idle باقی‌ مانده‌ایم. پس از ‘1’ شدن Start سیگنال‌های کنترلی و کلاک Slave‌ فعال شده و به State بعدی رفته‌ایم. باقی مراحل نیز همانطور که در تشریح کد بیان شد مرحله به مرحله طی شده است.

۱۵) بخش ابتدایی نتیجه شبیه‌سازی

در بخش زیر نیز نحوه عملکرد پایه ریست را مشاهده می‌کنیم. با فعال شدن ریست به حالت Idle رفته، سیگنال‌ها مقدار اولیه خود را دریافت کرده و سیگنال کلاک Slave را نیز قطع می‌کنیم. با دریافت دوباره سیگنال Start ارسال دیتا برای بار دیگر انجام شده و پس از نهایی شدن عملیات متوقف می‌شود.

۱۶) نحوه عملکرد Reset



۱۷) پایان ارسال دیتا در شبیه‌سازی