بنام خدا

گزارش پروژه VHDL

میلاد کلوندی

استاد درس: دکتر میرزا کوچکی

معرفی پروتکل SPI:

--پروتکل سریال بسیار پرکاربرد که با ۴ سیم دیتا را به صورت سنکرون را منتقل میکند

  معنی سنکرون بودن یعنی به همراه دیتا کلاک هم برای گیرنده ارسال میشود و گیرنده دیتا را به نحو مناسبی نمونه برداری می کند معمولاً در لبه بالا رونده کلاک محل نمونه برداری بیت های دیتا است

--برای انتقال دیتا در فواصل نزدیک و سرعت نسبتاً زیاد

--انتقال دیتا به صورت فول دوبلکس انجام میشود

پکت SPI:

--برعکس RS232 پکت های SPI فقط شامل بیت های دیتا هستند

--همچنین طول پکت ها می توانند متفاوت باشند ۸ ۱۶ ۲۴ و ۳۲

--سیگنال cs مشخص کننده ابتدا و گاهی انتهای پکت است  در پکت های SPI ما فقط دیتا را ارسال میکنیم چیزی در این پکت ها وجود ندارد که مشخص کند ابتدا و انتهای پکت کجاست

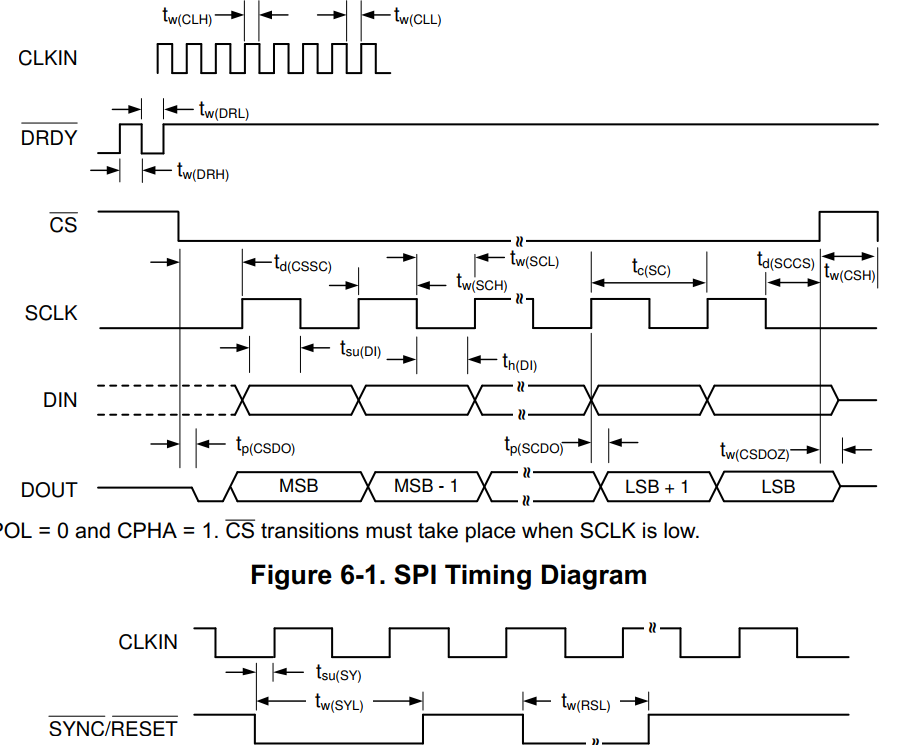
تایمینگ پکت SPI:

--سرعت بیشتر نسبت به RS232

 --برخلاف RS232 که دارای یکسری سرعت های استاندارد است و گیرنده و فرستنده باید روی این سرعت به توافق برسند و بر مبنای آن سرعت ارسال انجام می‌شود ولی هر پریفرالی که از پروتکل SPI استفاده می کند یک ماکسیمم سرعت کلاک برای SPI مشخص میکند

--طول پاکت معمولاً ضریبی از ۸ ارسال ممکن است به صورت MSB\_FIRST یا LSB\_FIRST فرصت انجام شود

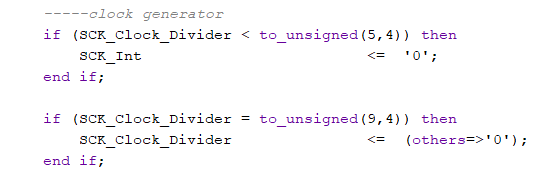
|  |  |
| --- | --- |
| Min 64 | Tc(sc) |
| Min 16 | Td(cssc) |
| Min 10 | Td(sccs) |
| Min 15 | Tw(csh) |
| Min 5 | Tsu(di) |
| Min 10 | Th(di) |
| Min 4 | Tw(drh) |
| Min 4 | Tw(drl) |
| Min 5 | Tp(scdo) |



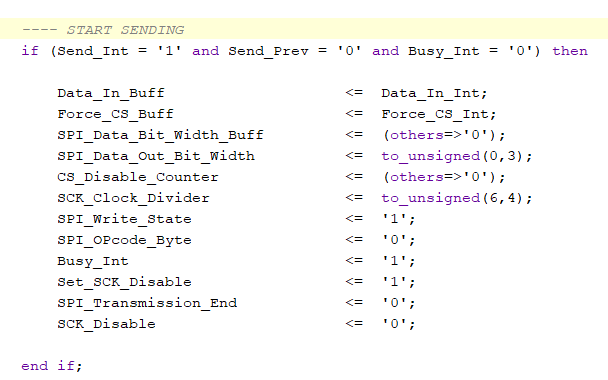
ما در اینجا فرکانس سیستم را ۱۰۰ مگاهرتز در نظر گرفتیم فرکانس خروجی SCK را برابر مقدار ۱۰ مگاهرتز در نظر گرفتیم  یک پریود ده نانو ثانیه به ما میدهد بنابراین شرط پریود SCK برقرار میشود

در اینجا ما دو بلوک به نام‌های SPI و ADS داریم بلوک SPI برای ارسال داده های سریال استفاده می شودو بلوکه ADS برای ارسال کامند رجیستر های این بلوک استفاده می شود

ابتدا بلوک مربوط به SPI را توضیح می دهیم :

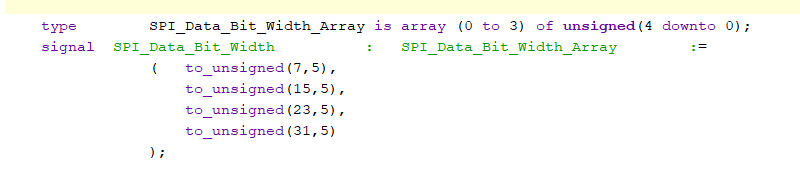


در اینجا ما دو تا شرط داریم که کلاک spi نوشتیم علیرغم تایمینگی که در دیتاشیت داریم ماکسیمم فرکانس کلاک میتونست 15.625 مگاهرتز باشد اما من در پیاده سازی ۱۰ مگاهرتز در نظر گرفتم که پریود کلاک 100 نانو ثانیه ای به ما میدهد با انتخاب کلاک مدار ۱۰۰ مگاهرتز و پریود ۱۰ نانو ثانیه است اگر یک کانتر درست کنیم هر واحدی که به این کانتر اضافه میشه در واقع ده نانو ثانیه است پس برای اینکه ما صد نانو ثانیه رو ایجاد کنیم که معادل ده مگاهرتز است نیاز داریم که کانتر ما ده تا شمارش انجام بده کانتری که در اینجا تعریف کردیم برای تولید کلاک اسمش SCK\_Clock\_Divider است در بدنه process می‌بینیم که این شمارنده بصورت free runingهمیشه  در حال شمارش است در داخلی if شرطی گذاشتیم که اگر کانتر کوچکتر از ۵ بود یعنی ۰ ۱ ۲ ۳ و ۴ مقدار SCK\_Int برابر صفر باشد در غیر اینصورت هیچ شرطی قرار ندادیم و شرط تمام شده اما در خط ۸۲ مقدار SCK\_Int برابر '1' قرار دادیم که در صورت اجرا نشدن شدن شرط بالا مقدار SCK\_Int  ‘1’خواهد شد در شرط دوم نوشتیم که اگر کانتر ما برابر ۹ شد حالا کانتر ریست شود



در اینجا شرطی داریم که شروع ارسال است که با لبه بالا رونده Send کار میکند در داخل شرط اگر لبه بالارونده Send رخ دهد و سیگنال Busy\_Int هم برابر '0' باشد فرایند ارسال شروع میشود در این فرآیند ابتدا یک سری سیگنال ها مقدار دهی اولیه می شوند مقدار Data و Force\_Cs بافر می شوند که اگر در ادامه تغییر هم کرد مقدار آن لحظه را داشته باشیم چون ممکنه فرمان Send بیاید و ما شروع به کار با دیتا کنیم ولی در حین کار مقدار دیتا کسی که با ماژول استفاده می کند در بیرون عوض کنه این نباید باعث خراب شدن عملکرد ماژول ما بشه برای اینکه این اتفاق نیفته به محض دیدن لبه بالارونده send مقدار دیتای ورودی و مقداری که برای Force\_Cs بوده رو در سیگنال میانی قرار میدیم و با این دو سیگنال میانی در پروژه کار میکنیم

ورودی دیگری بود به اسم Command\_Type\_Int که نشان میده طول پکت ما چند بیتی است (00 برای 8 - 01 برای 16 - 10 برای 23 - 11 برای 32 بیت)



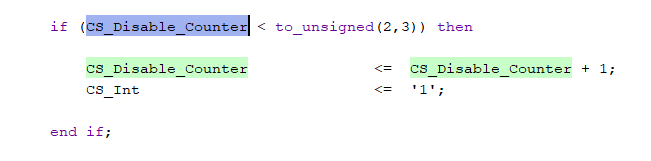
در اینجا آرایه تعریف کردیم به اسم SPI\_Data\_Bit\_Width  و یک تاییپی تعریف کردیم به صورت آرایه ای که چهار خونه داره که هر خونه یک unsined پنج بیتی است یک خط پایین تر یک سیگنال SPI\_Data\_Bit\_Width کردیم که اولی ۷ دومی ۱۵ سوم ۲۴ و آخرین ۳۱ که همان‌ طول پکت‌های ۸ بیتی ۱۶ بیتی و ... است

حالا با توجه به اینکه Command\_Type\_Int چه بوده و تبدیل به اینتیجر من از اون آرایه مقدار متناظر اون رو میریزم در یک سیگنال دیگه به اسم SPI\_Data\_Bit\_Width\_Buff (مثلاً اگر Command\_Type\_Int روکاربر برابر صفر قرار داده باشه پس خونه شماره صفر آرایه خونده میشه که مقدار ۷ در آن بوده پس طول دیتای ما ۸ بیتی خواهد بود یعنی سیگنال دیتای ما که ۳۲ بیتی بود خونه های7 تا 0 ما نیاز داریم و استفاده میکنیم  )

سیگنال دیگری به اسم SPI\_Data\_Out\_Bit\_Width که کانتر است برای شمردن بیت های خوندن از اسلیو  چون ما بیت هارو ۸ بیت ۸ بیت از اسلیو می خونیم و در داخل شرط send مقدار ثابت صفر به این کانتر به عنوان مقدار اولیه دادیم  زیرا سیگنال ما LSB\_First است

دو خط پایین تر یک کانتر دیگر که مقدار آن را صفر گذاشتیم و خود کانتری است که کلاک را می شمارد SCK\_Clock\_Divider و مقدار آن را شش قرار دادیم و سیگنال دیگری به نام SPI\_Write\_State رو برابر یک کردیم   چون قراره به استیت نوشتن ببریم

 سیگنال دیگری به نام SPI\_OPCode\_Byte  را برابر صفر کردیم Busy\_Int را برابر  یک کردیم ( دلیل آن مشخص است)  سیگنال دیگه ای به نام Set\_SCK\_Disable  برابر یک شده  و سیگنال SPI\_Transmission\_End  و SCK\_Disable  را برابر صفر گذاشتیم

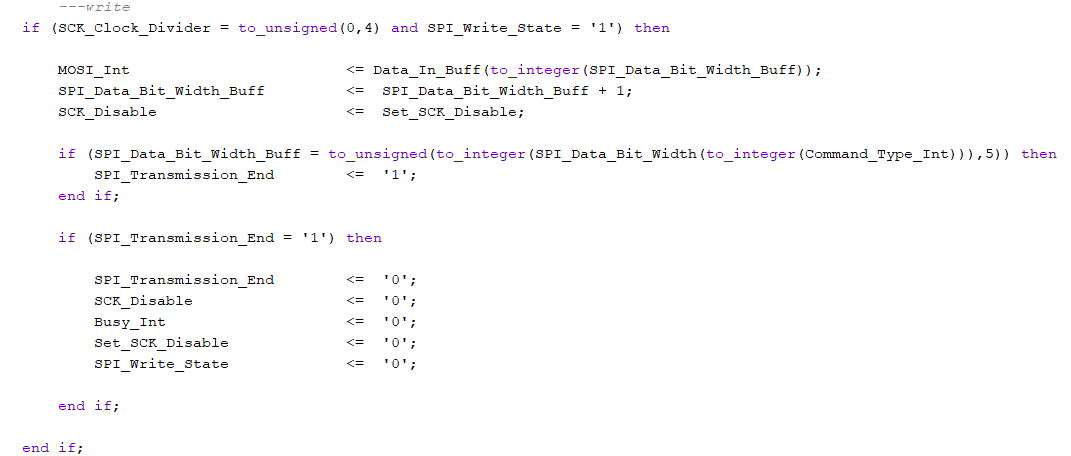


 در ابتدای کد یک سیگنال داریم که CS\_Disable\_Counter نام دارد ( در دیتاشیت گفتیم که CS  فعال پایین است و همیشه برابر صفر است ولی وقتی فرمان Send  ارسال می شود یک مدت کوتاهی CS  یک می شود که slave  ریست بشود و ابتدای پکت معلوم شد دوباره CS  برابر صفر میشه و ارسال انجام میشه  مقدار اولیه این کانتر ( CS\_Disable\_Counter ) هفت است  چون مقدار هفت  داره  شرط برقرار نیست وقتی فرمان send ارسال میشه ما کانتر رو ریست میکنیم بعد شرط ما برقرار میشه  به مدت ۲ کلاک و تا زمانی که شرط برقراره CS\_Int  برابر یک است CS\_Int رو ما همیشه برابر صفر قرار دادیم در زمان هایی که شرط  برقراره اون رو برابر یک قرار  میدهیم

به این ترتیب به محض این که فرمان Send  اعمال شد ما اون  CS رو مدتی Disable  میکنیم و دوباره enaable  میکنیم حالا چرا دو کلاک چون در دیتاشیت نوشته بود حداقل باید ۱۶ns  بشه  دوکلاک میشه 20ns

 یک سیگنال دیگه ای به نام SCK\_Disable  برابر صفر قرار دادیم یعنی غیر فعال کردن کلاکSPI در قسمت concurent  ماکلاک SPI رو داریم که SCK\_Int رو بهش ارجاع دادیم ولی اون رو با SCK\_Disable اند کردیم وقتی SCK\_Disable  برابر صفر باشه فرقی نداره کهSCK\_Int چی باشهSCK برابر صفر میشه در زمانی که ما داریم سیگنالCS رو غیر فعال می کنیم نیاز به کلاک زدن نیست بنابراین در این لحظات ما کلاک رو غیرفعال کردیم

بخش نوشتن:

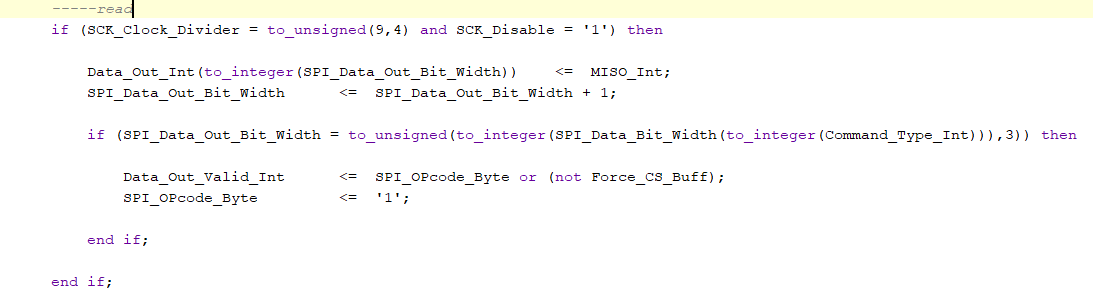


اما بعد از اون میرسیم  به شرطی که عکس آن در بالا قرار داده شده گفتیم که اگرSCK\_Clock\_Divider( همان کانتری که کلاکSPIرو تولید می کرد) برابر صفر باشه(زمانی برابر صرف میشه که مقدار اون از صفر تا ۴ باشه پس وقتی صفر هست یعنی شروع پریود این کلاک هستیم  که صفره  پس ما در ابتدای پریود  هستیم) وSPI\_Write\_State برابر یک باشد( که همین طور هم هست در زمان ارسال که شرطsend  برقراره مقدار اون رو یک کردیم)کانتر ما شمرد از ۶ تا ۹ و که شرط برقرار شد مقدار اون رو صفر قراردادیم شرط بالا  برقرار می شود و درست در اولین پریود  اولین کلاک  هستیم میخواهیم اولین بیت دیتا رو خروجی قرار بدیم و اون رو رویMOSI قرار بدیم قبلا دیتا رو رویData\_In\_Buff ریخته بودیم که ۳۲ بیتی هست سیگنال شمارندهSPI\_Data\_Bit\_Width\_Buff که شماره بیت را مشخص می‌کرد

با فرض اینکه مقدار این سیگنال صفر باشه پس از اول بیت صفرم رو  روی خروجی می ریزیم چون قرار سیستم  LSB\_First  رو پیاده سازی کنیم ضمن این کار یکی به اون اضافه میکنیم که وقتی شرط برقرار شد بیت بعدی روی خروجی قرار بگیره در عین حال وقتی وارد این شرط میشیم سیگنال SCK\_Disable  رو برابر میکنیم با Set\_SCK\_Disable  که سیگنال ای است که در قسمت پایین کد برابر یک قرار دادیم پس اینجا SCK\_Disable برابر یک میشه حالا که این سیگنال برابر یک شد تازه کلاک شروع به فعالیت می کند (چون در بالا این سیگنال رو باSCK\_Int اند کردی بهSCK  دادیم )  پس پس کل  شرط بالا تمام کارهای مربوط به نوشتن است

در داخل  شرط  دونه دونه بیتا نوشته میشه خط بعد یکی به کانتر اضافه میشه خط بعدی اون هم کلا کرو فعال میکنه در داخل این شرط یک شرط دیگه داریم گفتی اگر  SPI\_Data\_Bit\_Width\_Buff برابر طول دیتا شد ( وقتی برابر طول دیتا میشه که سنگین ترین بیت رو ارسال کردید ) باید یک پریود دیگه صبر کنیم که یک سیگنال دیگه به نام SPI\_Transmision\_End رو  که در ابتدای کار برابر صفر کرده بودیم اینجا اونو یک می کنی تا یک پریود دیگه بگذره و این شرط برقرار بشه وقتی این  شرط  برقرار شد شرط بعدی اون هم برقرار میشه چونSPI\_Transmision\_End  رو دفعه قبل برابر یک قرار دادیم وقتی این شرط برقرار بشه اولا اینکه همین سیگنال رو برابر صفر می کنیم  چون ارسال تموم شده و میخواهیم SCK\_Disable رو هم دوباره صفر می کنیم چون سال تموم شده و می خواهیمSCK متوقف بشه وBusy  رو برابر صفر می کنیم  چون ارسال تموم شده و می توانیم ارسال جدیدی داشته باشیم Set\_SCK\_Disable رو  هم برابر صفر می کنیم چون دیگه قراره از کلاک بعدی این شرط برقرار نباشه

بخش مربوط به خواندن:



اگر فاز مربوط به خواندن داشته باشیم این بخش به کار ما می آید یک شرطی گذاشتیم که اگر SCK\_CLOCK\_Divider ( کانتر مربوط به تولید کلاک SPI) برابر نه شد شرط برقرار بشه به نظر خودم بهترین جا برای خوندن دیتا انتهای پریود کلاک هست درست قبل از لبه پایین رونده کلاک اگر کانتر برابر نه باشد یعنی آخرین لحظه هستی که هنوز مقدار کلاک یک است و در انتهای کلاک است و در کلاک بعد کلاکSPI میره صفر و لبه پایین رونده اول ایجاد میشه و تازه دیتا شروع میکنه به خوندن و شروع میکنه به برگشتن به مستر بنابراین قبل از اینکه لبه پایین رونده رخ بده دیتا رو بخونم یعنی دیتایی که لبه پایین رونده قبلی خارج شده الان به صورت پایدار در دست من است است و در اینجا میتونم اون رو بخونم( دلیل اینکه اینجا مقدار و نه قرار دادیم) البته شرط دیگری قرار دادیم که SCK\_Disable  برابر یک است ماSCK\_CLOCK\_Divider رو مقدار اولیه ۶ به اون دادیمبعد سه کلاک مقدار اون به نه میرسه و شرط برقرار میشه اما قرار نیست بعدسه کلاک این شرط انجام بشه بنابراین and دیگری گذاشتیم و اگرSCK\_Disable  برابر یک شد شرط خوندن انجام بشه چون ابتدا برابر صفر است شرط انجام نمیشه تا اینکه برسه به ۹ و بعد بشه صفر  بعد بیاد داخل شرط  و مقدار اون تازه برابر یک بشه  و سیکل بعد شرط خوندن برقرار بشه.

رسیدیم به انتهای پریود  کلاک SPI می خواهیم دیتا رو بخونیم و MISO\_Int  رو در یک سیگنال میانی Data\_Out\_Int میریزیم که ۸ بیتی است .

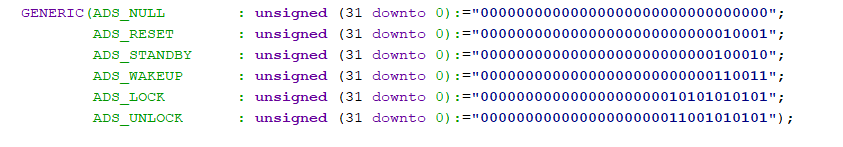
یک سیگنال کانتر تعریف کردیم به اسم SPI\_Data\_Out\_Bit\_Width  که قبلاً مقدار اون برابر صفر بود  اونو در سبک ترین قرار میدهیم  LSB\_First هست  یکی به کانتر اضافه میکنیم و شرطی هم گذاشتیم هر وقت مقدار این کانتر به هفت رسید کار خوندن تمامه  اگر شرط برقرار بشه یک سیگنالValid رو به مدت یک کلاک برابر با یک قرار میدهیم که به صورت دائمی برابر صفر است فقط در این لحظه در یک کلاک برابر یک قرار می دهیم این لحظه وقتی است که آخر این بیت رو دریافت کردیم و به خروجی ارجاع داد همزمان ولید رو هم برابر یک قرار می دهیم ولی به جای اینکهData\_Out\_Valid\_Int رو مستقیماً برابر یک قرار بدیم آن را برابر SPI\_OPCode\_Byte OR (NOT Force\_CS\_Buff) قرار دادیم که سیگنال SPI\_OPCode\_Byte   وقتی سیگنالsend اومد برابر صفر کردیم  و اولین باری که SCK\_CLOCK\_Divider  برابر نه میشه و همزمان شرط خط ۱۱۸ برقراره مقدارSPI\_OPCode\_Byte    برابر صفره زیرا در فاز اول خوندن هستیم برای مثال:  فرض کنیم ارسال ما ۱۶ بیتی است  بایت اول رایت و بایت دوم رید هست یعنی میخواهیم از اسلیو بخونیم  اولین باری که این شرط اجرا می شود تازه  فاز رایت هستیم یعنی اگر بخونیم دیتا ها بی اهمیت هستند چون در فاز رایت بودیم چون این دیتاها معنی ندارد موقعی که شرط خط ۱۲۳ میرسه نباید valid برابر یک بشه باید هنوز صفر بمونه تا دیتای اشتباهی  نخونیم پس از یک سیگنال ای به نامSPI\_OPCode\_Byte  استفاده کردیم که در واقع مشخص میکند ارسالیم که به بایت اول ارسال  میگیمOPCode ۱۱ کد دستورالعمل داخل اون هست پس در اینجا صفر و بهData\_Out\_Valid\_Int ارجاع می دهیم پس هیچ اتفاقی نیفتاده و کار ادامه پیدا میکنه و در همین شرط مقدارSPI\_OPCode\_Byte  برابر یک می کنیم چون در بایت های بعدی میتونیم عمل خوندن انجام بدیم مثلاً در بایت دوم دیتا ای که میخونم valid  برابر یک میشه.

 همین با یک سیگنال دیگه ای NOT Force\_CS\_Buff اور کردیم  که وقتی سیگنالsend  اومدForce\_CS\_Int رو داخل این سیگنال قرار دادیم تا نگه داریم.

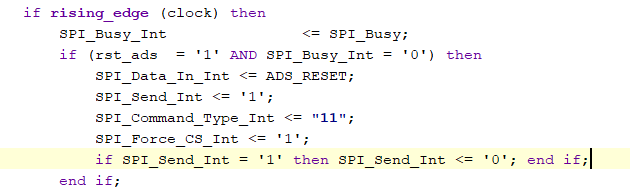
کاربردForce\_CS\_Int:

اگر بخواهیم  ارسالی داشته باشیم و درون ارسال cs،Didable نشه  Force\_CS\_Buff باید برابر صفر بشه حالا فرض کنیم قرار بیشتر از هزار دیتا از اسلیو بخونیم قاعدتاًCS بایدDisable بشود و بعد اونOPCode رو ارسال کنم و بعد هزار بایت دیتا من از اسلیو بیاد تا بخونم وقتی اون هزار بایت رو بخونم با توجه به اینکه پرورت دیتا ورودی من ۳۲ بیتی است میتونم اونو در دیتاهای بلوک ۳۲ بیتی قرار بدم و در این ۳۲ بیت ها نبایدCS ماDisable بشه اما کد اینگونه است که هر بار یک دیتا ۳۲ بیتی قرار بدم و فرمانsend رو ارسال بکنم یکبارCS رو Disable بکنم برای رفع این مشکل  ما از سیگنالForce\_CS استفاده کردیم یکبار یک فرمانی رو ارسال میکنیم  که قاعدتاً باعث میشهCS همDisableبشه و فرض کنیم بعد این فرمان قرار هزار بار بخونیم بدون اینکهCS ما Disable بشود بایدForce\_CS روی یک بکنیم اگر به ابتدای کد ببریمCS\_Int اند شده باForce\_CS\_Buff که اند  هر چیزی باصفر میشه صفر که باعث میشه هرچندCS\_Intرو یک می کنم ولی اثری نمی گذارد چون اون رو با  ۰ and کردیم

بلوک مربوط به کامند رجیستر های ADS :



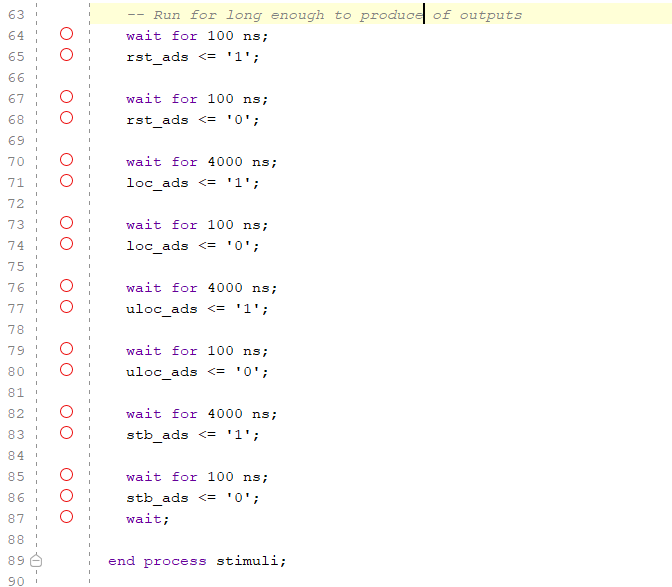
در کد بالا مقادیر هر کامند به انها اختصاص داده شده که در واقع 16 بیت ان مورد نیاز است این حذف 16 بیت اضافی را SPI\_Command\_Type انجام میدهد



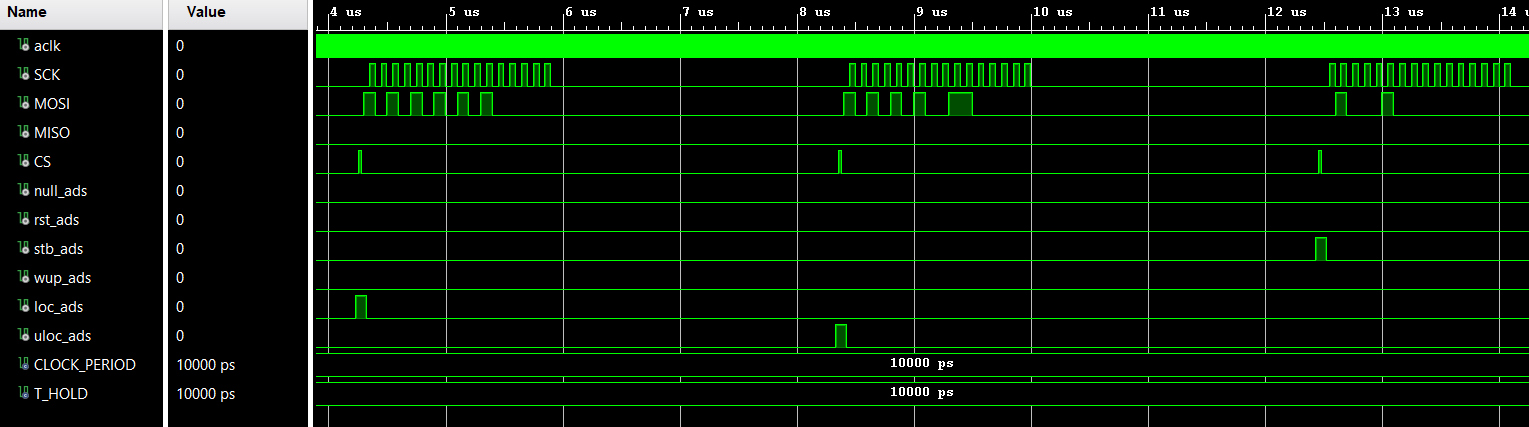
در بالا یک نمونه کد ارسال دیتا داریم اگر rst\_ads برابر یک شد و SPI هم بیزی نبود انگاه مقدار دیتا ریست ارسال شده و send هم برابر 1 میشود SPI\_Command\_Typeهم برابر 11 شده یعنی کل 32 بیت برای ما اهمیت دارد اگر میخواستیم 16 بیت ارسال کنیم میبایست مقدار ان را برابر 01 قرار میدادیم

در شرط بعدی ما مقدار send را برابر صفر کردیم تا برای ارسال دیتا بعدی اماده باشد

تست بنچ و سیمولیشن :



در کد تست بنچ بالا چند کامند به صورت تصادفی و در زمان های مختلف داده شده و نتیجه این کد رو در سیمولینک میتوان دید:



در کد شبیه سازی شده زیر یک سیگنال در زمان ارسال و برای ایجاد اختلال به SPI دادیم اما کد این را ریجکت کرد و در ارسال خود هیچ خللی به وجود نیامد.