دانشکده مهندسی برق

پروژه پایانی VHDL
LTC2369-18

نام دانشجو

عاطفه بهادری

شماره دانشجویی

۴۰۱۶۱۱۵۰۳

IUST_9


بسم الله الرحمن الرحیم

فهرست مطالب

فهرست اشکال

فهرست جداول

فصل ۱ :

مقدمه

۱-۱- مقدمه

در این پروژه، هدف پیاده سازی پروتکل ارتباطی SPI با زبان برنامه نویسی VHDL است. قطعه مورد بررسی آی سی LTC2369-18 است که چندین حالت مختلف برای اتصال قطعه به کنترلر(Host) را معرفی میکند. قطعه ذکر شده برای برقراری ارتباط از طریق پروتکل SPI دارای سه سیم SDO(MISO) ، SDI(MOSI) و SCK است. همچنین پایه هایی از جمله CHAIN ،CNV و BUSY نیز برای بخش کنترلی آی سی و ارتباط چندین آی سی با کنترلر تعبیه شده است.

برای پیاده سازی SPI بر بستر FPGA اطلاعاتی از جمله تعداد بیت های مورد نیاز قطعه ، فرکانس کلاک SPI و همچنین نحوه کارکرد سیگنال های کنترلی مورد نیاز است که این اطلاعات از datasheet قطعه مورد نظر بدست آمده است.

فصل ۲ خلاصه ای از datasheet و توضیحات قطعه را شامل میشود ، فصل ۳ توضیحات مورد نیاز در مورد سیگنال های اصلی به کار برده شده، فصل ۴ شامل توضیح بخش های مختلف کد VHDL و فصل ۵ نتایج شبیه سازی SPI با VHDLدر برنامه ISE را نشان میدهند.

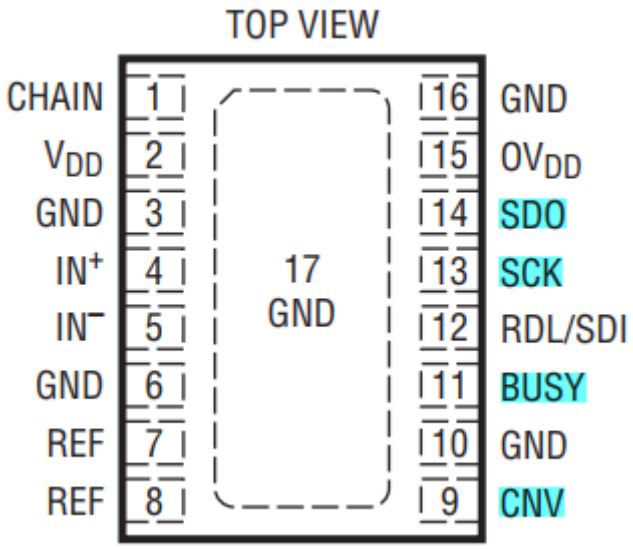
فصل ۲ :

خلاصه datasheet قطعه LTC2369-18

۱-۲- مشخصات کلی

- عملکرد قطعه تبدیل آنالوگ به دیجیتال با دقت ۱۸ بیت است.
- نرخ نمونه برداری این قطعه ۱٫۶msps است که این مقدار ماکسیمم فرکانس نمونه برداری برای این آی سی میباشد.
- قابلیت تطابق با پروتکل SPI چهار سیمه و همچنین دارای قابلیت اتصال زنجیر وار (Daisy-chain).

۲-۲- پایه های مورد نیاز



شکل (۱-۲) پایه های قطعه LTC2369-18

2-2-2- توضیح پایه های مورد نیاز

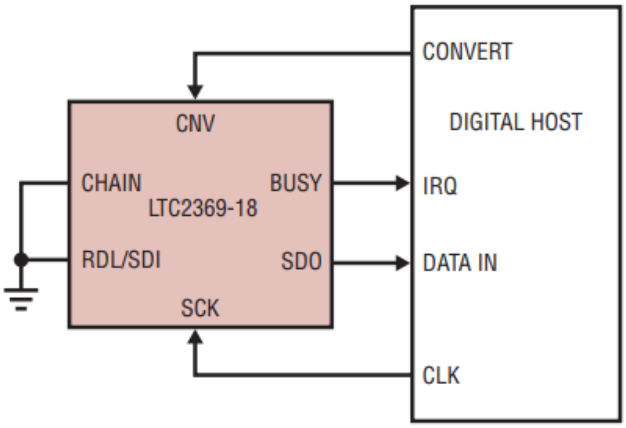
پایه CNV : لبه بالارونده این پایه همچون Chip Select عمل میکند ، قطعه را روشن میکند و همچنین فرمان شروع تبدیل آنالوگ به دیجیتال را نیز صادر میکند. این پایه در کد به عنوان خروجی تعریف شده است تا بتوان توسط آن قطعه را روشن و آماده به کار کرد.

پایه BUSY : این پایه هنگام شروع تبدیل مقدار ۱ را به خود میگیرد و تا پایان تبدیل ۱ باقی میماند . صفر شدن این پایه به معنای اتمام عمل تبدیل آنالوگ به دیجیتال است. این پایه در کد به عنوان ورودی تعریف میشود تا آی سی زمان اتمام تبدیل را به کنترلر اعلام کند.

پایه SCK : پایه سریال کلاک ورودی. در هر لبه بالارونده این سیگنال، یک بیت از داده به پایه SDO منتقل میشود .

پایه SDO : پایه داده سریال خروجی. در هر لبه بالارونده کلاک SCK، اطلاعات تک بیت تک بیت بر روی این پایه به صورت MSB First قرار میگیرند.

پایه CHAIN : برای اتصال چندین قطعه به کنترلر استفاده میشود. در این حالت پایه SDI فرمان فعال و غیر فعال شدن SDO را بر عهده دارد. در این پروژه این پایه مورد استفاده ما نخواهد بود و و همراه با SDI به منطق صفر (زمین) متصل میشوند.



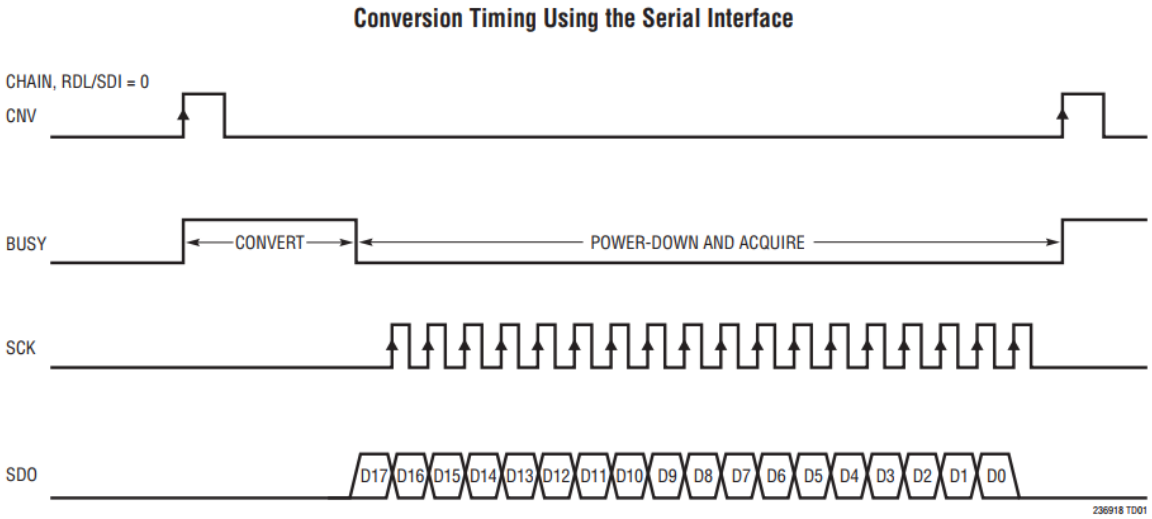
شکل (۲-۲) شماتیک نحوه اتصال قطعه به کنترلر

فصل ۳ :

سیگنال ها و زمان بندی

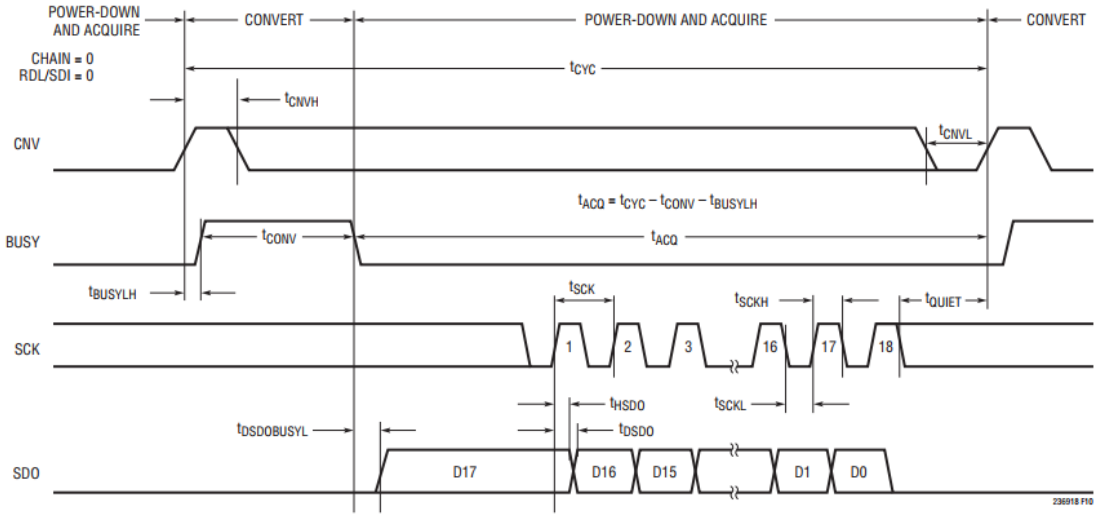
۱-۳- سیگنال های قطعه LTC2369-18 در حالت معمول

در حالت معمول(Normal mode) سیگنال های مربوطه همانند شکل (۱-۳) است.



شکل (۱-۳) شماتیک کلی سیگنال های اصلی

برای بررسی دقیق‌تر سیگنال ها و زمان بندی آنها میتوان به شکل ۲-۳ مراجعه کرد.



شکل (۲-۳) سیگنال های اصلی و زمان‌بندی دقیق آنها

در جدول ۱-۳ زمان بخش های مختلف روی سیگنال های اصلی آورده شده است.

SYMBOL	PARAMETER	CONDITIONS		MIN	TYP	MAX	UNITS
f_{SMPL}	Maximum Sampling Frequency		●			1.6	Msps
t_{CONV}	Conversion Time		●	360		412	ns
t_{ACQ}	Acquisition Time	$t_{ACQ} = t_{CYC} - t_{CONV} - t_{BUSYLH}$ (Note 10)	●	200			ns
t_{CYC}	Time Between Conversions		●	625			ns
t_{CNVH}	CNV High Time		●	20			ns
t_{BUSYLH}	CNV↑ to BUSY Delay	C_L = 20pF	●			13	ns
t_{CNVL}	Minimum Low Time for CNV	(Note 11)	●	20			ns
t_{QUIET}	SCK Quiet Time from CNV↑	(Note 10)	●	20			ns
t_{SCK}	SCK Period	(Notes 11, 12)	●	10			ns
t_{SCKH}	SCK High Time		●	4			ns

جدول (۱-۳) بخش های زمانی مختلف سیگنال ها

زمانی که پالس Busy صفر میشود نشان دهنده آماده شدن داده برای انتقال است. همانطور که از جدول ۱-۳ مشخص است ، زمان مورد نیاز برای انجام عمل تبدیل((Convert(CONV) ۳۶۰ نانو ثانیه و زمان لازم برای فرستادن داده حداقل ۲۰۰ نانو ثانیه است.

طول پالس مثبت کانورت که نقش CS را در این قطعه اجرا میکند حداقل ۲۰ نانو ثانیه است.

مهم ترین سیگنال در این بخش SCK میباشد که دوره زمانی(Period) ای برابر ۱۰ نانو ثانیه دارد که معادل فرکانسی برابر ۱۰۰ مگاهرتز است. بنابراین کلاک اصلی سیستم نیز روی ۱۰۰ مگاهرتز تنظیم شده است.

فصل ۴ :

کد VHDL

۴-۱- کداصلی

شامل موجودیت و معماری اصلی کد است. طبق شکل ۲-۲ پورت های لازم برای این قطعه شامل کلاک سیستم(CLK_SYS)، پورت شروع(start) برای دستور به آغاز پروتکل SPI ، SDO ، BUSY و CNV است که در فصل۲ توضیح داده شد. علاوه بر پورت های گفته شده، پورت reset نیز برای استفاده در کد اصلی و test bench به موجودیت اضافه شده است.

بخش معماری (Architecture) کد شامل بدنه های case when برای بخش‌بندی حالات مختلف عملکرد قطعه با پروتکل SPI است. قبل از شروع پراسس اصلی برنامه، سیگنال های حیاتی به سیگنال های درون-برنامه ای مربوطه ارجاع داده میشوند تا به صورت موازی(Concurrent) با پراسس اصلی اجرا شود(شکل۱-۴).

```
BEGIN

    CNV       <= CNV_INT;  -- chip select
    SDO_INT   <= SDO ;     -- MISO
    Busy_INT  <= Busy;
    state_INT <= state;
    reset     <= reset_INT;

    process(CLK_SYS)
    begin
```

شکل (۱-۴) ارجاع سیگنال های حیاتی

در بخش ابتدایی پراسس بعد از شرط کلاک اصلی سیستم، شرط reset اجرا میشود، چرا که اولویت سیگنال ریست از بقیه سیگنال های معرفی شده بالا تر است.

```
process(CLK_SYS)
begin

    if (rising_edge(CLK_SYS)) then
       if (reset_INT  = '1') then
             state    <= idle;
             CNV_INT  <= '0';
     else
```

شکل (۲-۴) شرط اجرای reset در برنامه

در بخش Case when حالت های کاری قطعه به چهار بخش idle, CONV, ACQ, QUIET تقسیم میشود.

idle : حالتی که در آن هیچ عملیاتی صورت نمیگیرد لذا قطعه منتظر میماند تا درصورت ۱ شدن سیگنال start (تعیین کردن زمان شروع به کار سیستم)، سیگنال کانورت از کنترلر فرستاده شده و عملیات آغاز شود. در غیر اینصورت وضعیت(state) در حالت idle میماند.

```
case state is

    when idle =>

        SDO_INT <= '0';
        bit_CNT <= "10001";
        if(start_INT = '1') then
            state    <= CONV;
            CNV_INT  <= '1';
        else
            state    <= idle;
            CNV_INT  <= '0';
        end if;
```

شکل (۴-۳) حالت idle

CONV : دراین حالت کنترلر منتظر میماند تا قطعه تبدیل را انجام دهد و اعلام کند که داده آماده ارسال است.

```
when CONV =>

    if(CNV_INT = '1' or Busy = '1') then
        state <= CONV;
        CNV_INT  <= '0' after 10 ns;
    else
        state    <= ACQ;
        CNV_INT  <= '0';
    end if;
```

شکل (۴-۴) حالت CONV

ACQ : این حالت زمانی رخ میدهد که سیگنال Busy صفر شده و داده ها آماده برای ارسال است.

```
when ACQ =>

    CNV_INT <= '0';
    --SDO := data_in(bit_CNT);
    data_in_INT (to_integer(bit_CNT)) <= SDO_INT ;

    if(bit_CNT /= 0) then -- if bit_CNT not 0
        state    <= ACQ;
        bit_CNT  <= bit_CNT -1 ;
    else
        state   <= QUIET;
        bit_CNT <= "10001";
    end if ;
```

شکل (۴-۵) حالت ACQ

QUIET : مدت زمانی که قطعه بعد از ارسال داده ها نیاز دارد تا دوباره دستور کانورت را دریافت کند. این حالت باید به مدت ۲۰ نانو ثانیه به طول بیانجامد ، لذا نیازمند ۲ سیکل پشت هم از کلاک سیستم است. اما پیاده‌سازی این بخش در کد با مشکل مواجه شد چرا که لبه بالا رونده سیگنال start_INT با دستور‘1’ = start and event’start در این حالت دیده نمیشود (به دلیل اینکه این سیگنال در بدنه پراسس آپدیت میشود). برای حل این مشکل میتوان حالت idle بعد از حالت QUIET را با هم به مدت ۲۰ نانو ثانیه در نظر گرفت. این راه در اجرای کد مشکلی به وجود نمی‌آورد و زمان‌بندی سیگنال ها به درستی صورت میگیرد.

```
when QUIET =>

    state <= idle;
    CNV_INT <= '0';
    SDO_INT  <= '0';
    bit_CNT  <= "10001";
```

شکل (۴-۶) حالت QUIET

4-1-2- نحوه تولید کلاک SCK

دو روش برای تولید کلاک SPI وجود دارد که روش اول بهینه تر و کوتاه تر است.

```
if (Busy_INT = '0' and state_INT = ACQ ) then
 SCK <= not (CLK_SYS);
else
 SCK <= '0';
end if;
```

شکل (۴-۷) روش اول در تولید SCK

روش اول : در این رویکرد به دلیل یکسان بودن فرکانس کلاک سیستم و کلاک SCK میتوان با اختلاف ۱۸۰ درجه کلاک SCK را از روی کلاک سیستم ساخت و تنها شرط مورد نیاز را اعمال کرد..
در این قطعه شرط تولید کلاک طبق شکل ۲-۳ ، صفر بودن سیگنال Busy و همچنین قرار داشتن در حالت ACQ است. بنابراین با اعمال این شرط در بدنه پراسس اصلی میتوان SCK را به درستی تولید کرد.

روش دوم : تولید یک سیگنال کنترلی برای شروع عملیات تولید SCK در پراسسی جدا و موازی با پراسس اصلی (شکل ۸-۴).

```
-- SCK_start generator

SCK_start_pro : process
begin
    SCK_start  <= '0','1' after 870 ns, '0' after 1045 ns, '1' after 1430 ns, '0' after 1605 ns;
wait;
end process;
```

شکل (۸-۴) تولید سیگنال کمکی برای ساخت SCK

و سپس تولید سیگنال SCK با اعمال شرط سیگنال SCK_start در پراسسی جدا و موازی با دو پراسس دیگر (شکل ۹-۴).

```
 -- SCK generator
SCK_pro : process
    begin
        if(SCK_start = '1') then
            SCK <= '0';
            wait for SCK_period/2;
            SCK <= '1';
            wait for SCK_period/2;
        else
            SCK <= '0';
            wait until SCK_start = '1';
        end if;
end process;
```

شکل (۹-۴) تولید SCK با اعمال شری سیگنال کمکی

مزیت روش اول نسبت به دومی در خودکار بودن آن است چرا که در روش دوم باید سیگنال SCK_start را به صورت دستی مطابق با مابقی سیگنال ها زمان دهی کرد.

۴-۲- کد تست بنچ

در کد تست بنچ سیگنال های پورت های ورودی باید ساخته شوند. لذا چندین پروسس به صورت موازی سیگنال ها را تولید میکنند.

```
-- Clock process definitions
CLK_SYS_process :process
begin
    CLK_SYS <= '0';
    wait for CLK_SYS_period/2;
    CLK_SYS <= '1';
    wait for CLK_SYS_period/2;
end process;
```

شکل (۴-۱۰) پراسس تولید سیگنال کلاک سیستم

```
  -- Busy generator
Busy_pro : process
  begin
    if(reset = '0') then
      Busy <= '0','1' after 505 ns , '0' after 865 ns,'1' after 1065 ns, '0' after 1425 ns ;
    else
      -- Busy <= '0','1' after 505 ns , '0' after 865 ns,'0' after 1065 ns, '0' after 1425 ns ;
      Busy <= '0','1' after 505 ns , '0' after 865 ns,'1' after 1065 ns, '0' after 1425 ns ;
    end if;
  wait;
  end process;
```

شکل (۴-۱۱) پراسس تولید سیگنال Busy

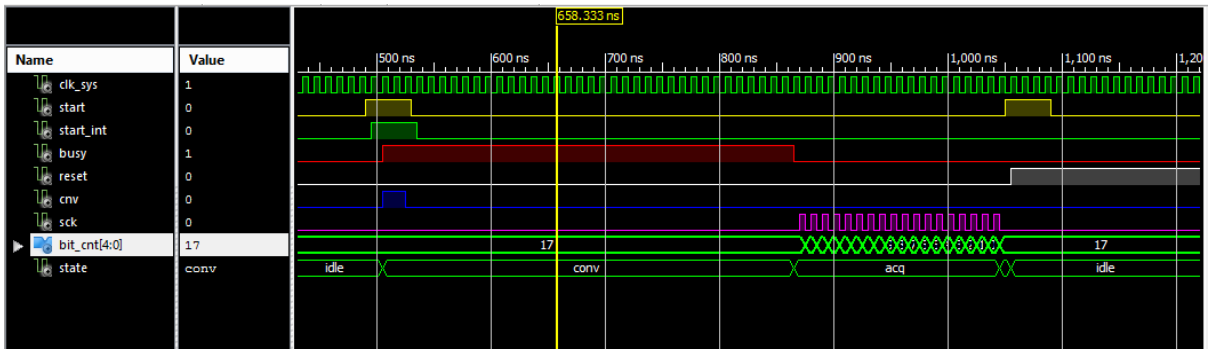
```
-- start genertor
start_pro : process
    begin
        start <= '0', '1' after 490 ns, '0' after 530 ns, '1' after 1050 ns, '0' after 1090 ns;
    wait;
end process;
```

شکل (۴-۱۲) پراسس تولید سیگنال start برای آغاز به کار کل برنامه

فصل ۵ :

شبیه سازی و خروجی ها

در این فصل خروجی های به‌دست آمده از کد بررسی میشود.
در این پروژه سیگنال SDO باید از قطعه به کنترلر وارد شود بنابراین به جای قرار دادن SDO در شبیه سازی تنها اندیس داده ورودی در شبیه سازی آورده شده است(bit_cnt).

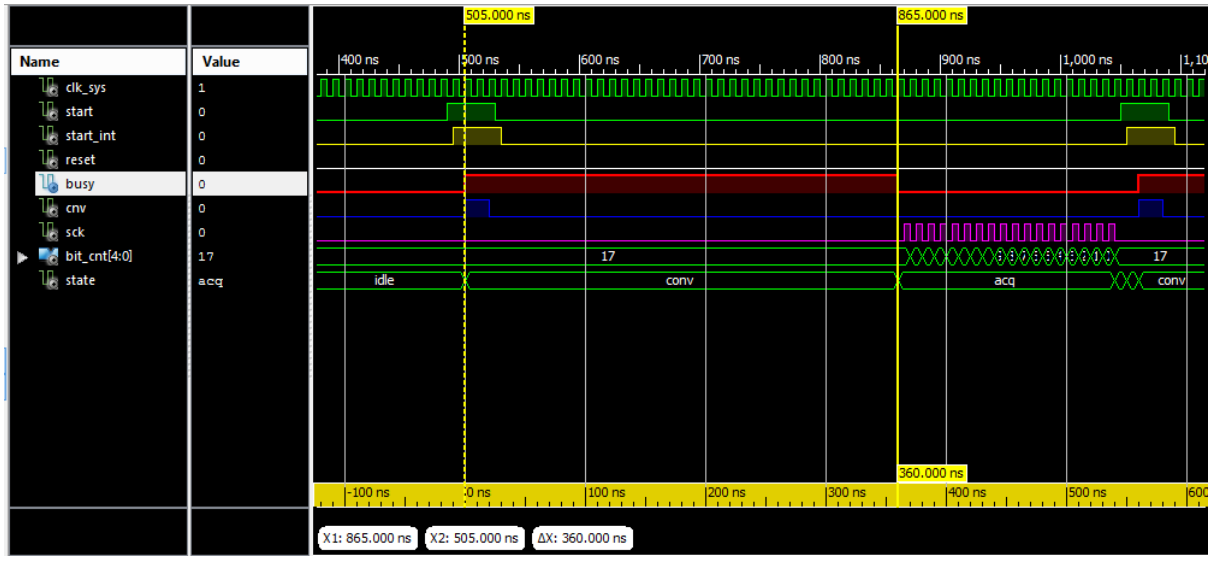


شکل (۵-۱) سیگنال های مورد نیاز قطعهLTC2369-18

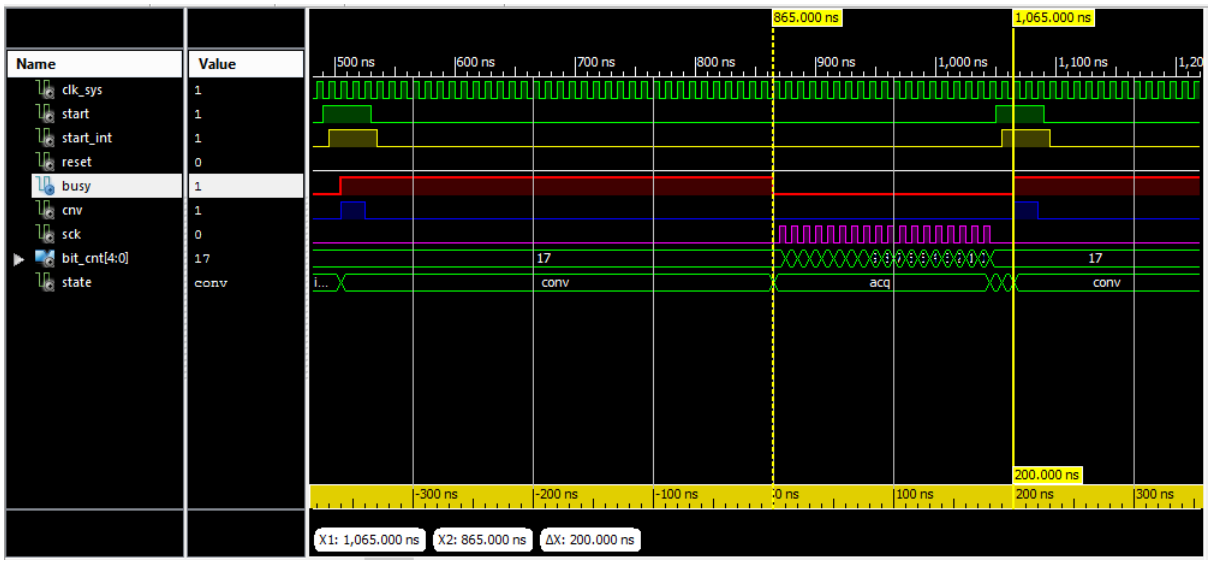
۵-۲- زمان‌بندی سیگنال ها در شبیه‌سازی

5-2-1- سیگنال Busy

همانطور که گفته شد، مدت زمان ۱ بودن در سیگنال Busy برابر ۳۶۰ نانوثانیه و زمان صفر شدن آن که در واقع همان حالت ACQ است برابر ۲۰۰ نانو ثانیه میباشد.

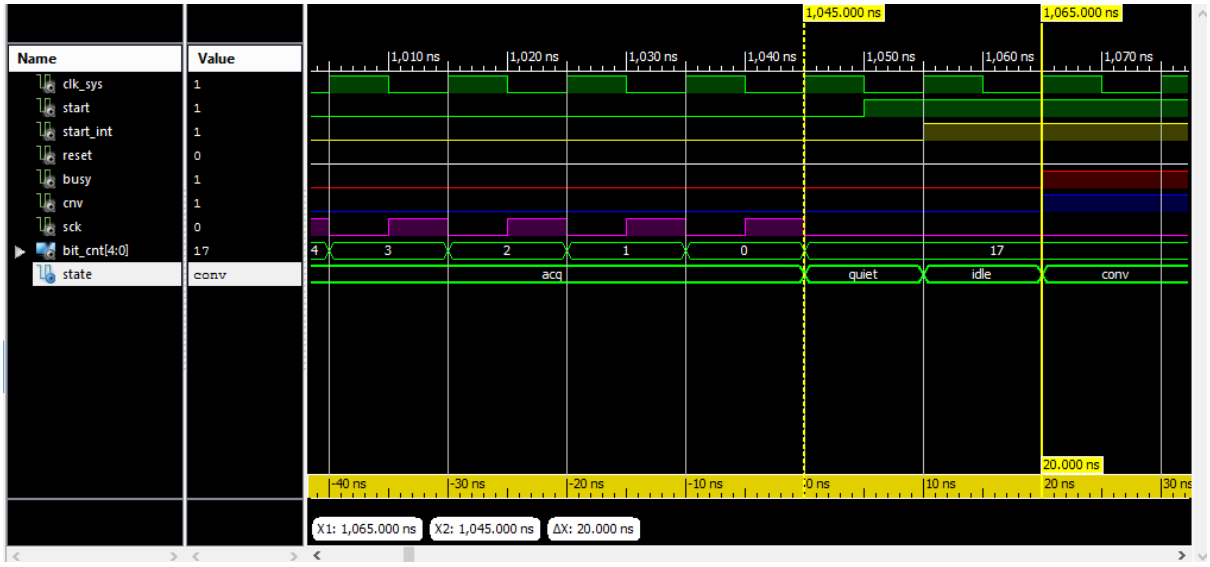


شکل (۵-۲) زمان convert و ۱ بودن سیگنالBusy



شکل (۳-۵) زمان ACQ و صفر بودن سیگنال Busy

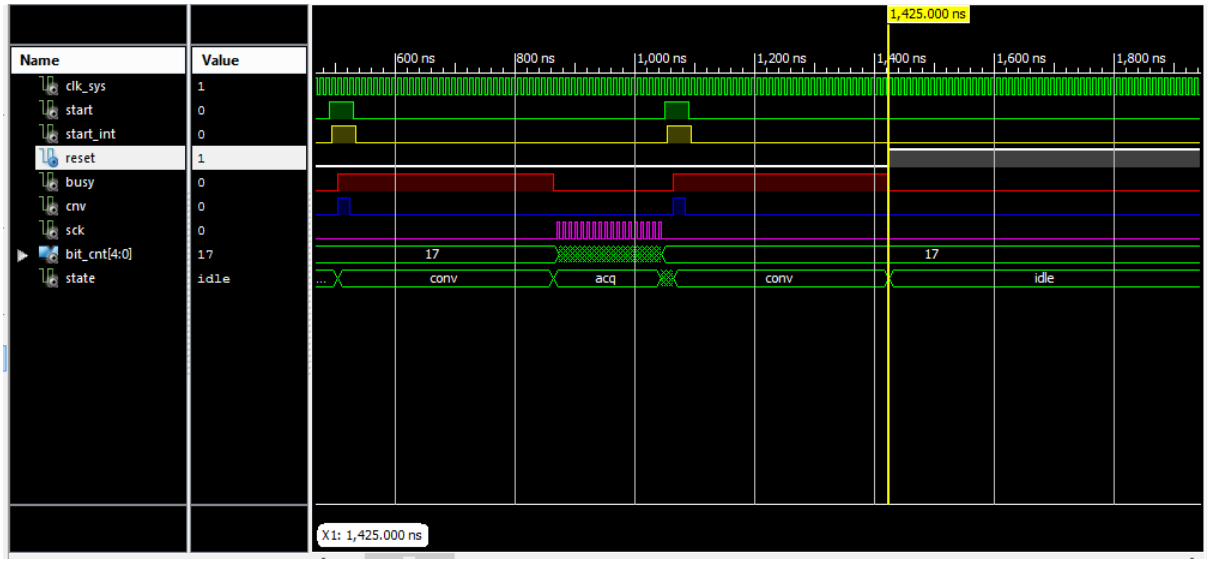
در این شبیه سازی بخش ACQ همان acq+quiet+idle میباشد(۲۰۰ نانو ثانیه) که بعد از آن دوباره سیگنال CNV یک میشود و عملیات تبدیل بعدی آغاز میشود(شکل ۳-۵ و شکل ۴-۵).



شکل (۴-۵) مدت زمان QUIET که ۲۰ نانوثانیه میباشد (quiet+idle)

۵-۳- سیگنال Reset

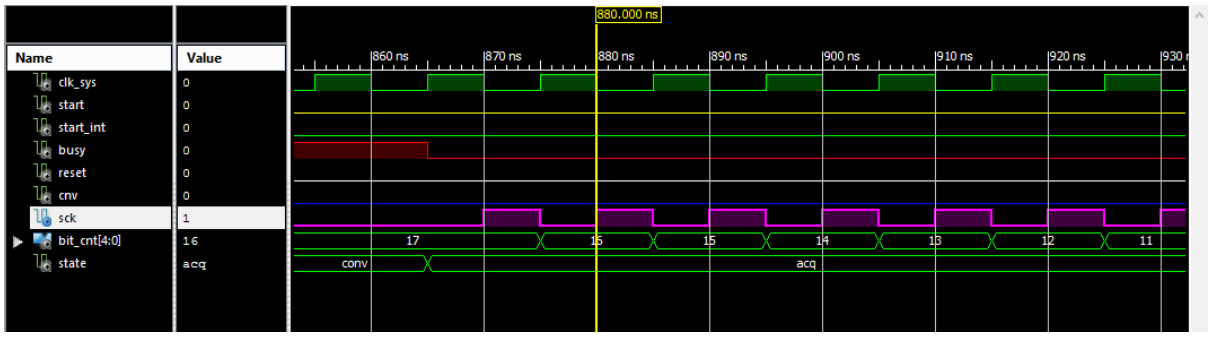
بعد از ۱ شدت سیگنال reset هیچ سیگنال دیگری به جز کلاک سیستم تولید نمیشود.



شکل (۵-۵) قبل و بعد ۱ شدن سیگنال reset

۵-۴- سیگنال SCK

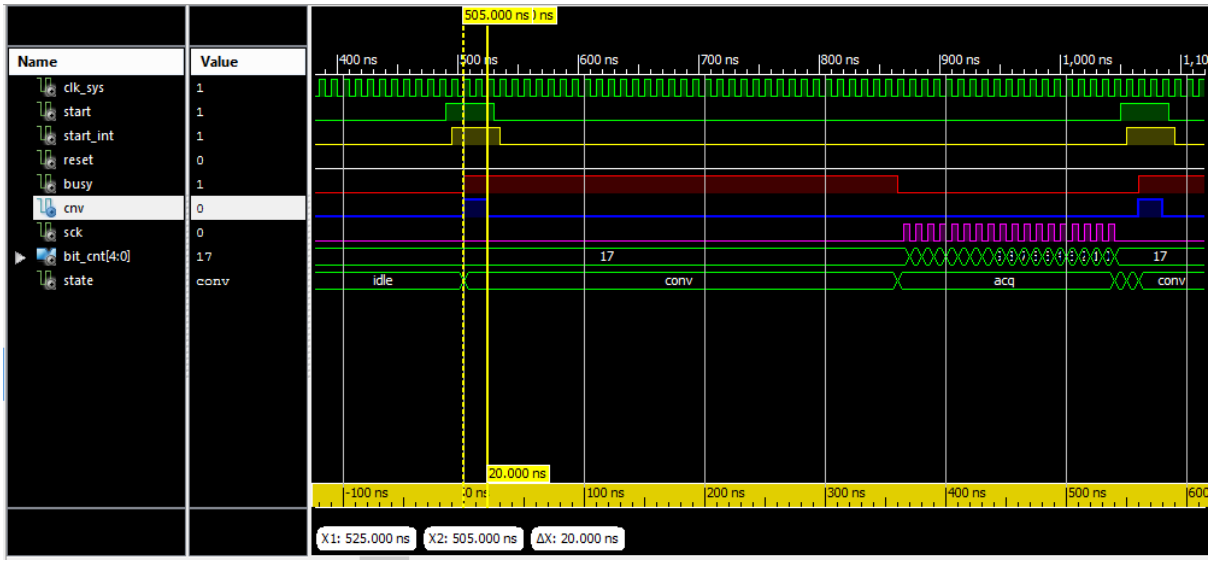
همانطور که در شکل ۵-۶ نشان داده شده است، پریود سیگنال SCK، ۱۰ نانو ثانیه است و هر لبه بالارونده درست در وسط هر بیت داده ارسالی قرار دارد. همچنین اختلاف فاز ۱۸۰ درجه SCK و clk_sys نیز در تصویر دیده میشود.



شکل (۵-۶) ارسال داده در لبه بالا رونده SCK

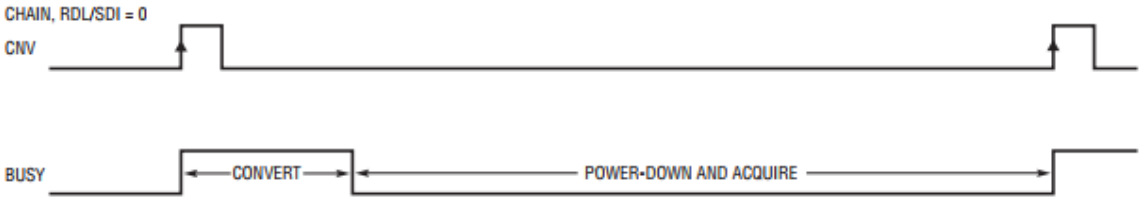
۵-۵- سیگنال CNV

همانطور که گفته شد سیگنال convert نقش Chip select را اجرا میکند و بعد از ۱ شدن آن عملیات تبدیل ADC شروع شده و بلافاصله (مدت زمان پاسخ آی سی به سیگنال CNV که همان ۱ شدن سیگنال Busy میباشد، میتواند از صفر تا ماکسیمم ۱۳ نانو ثانیه متغیر باشد. که در اینجا بلافاصله بعد از ۱ شدن CNV ، Busy نیز ۱ میشود) سیگنال Busy نیز ۱ میشود.



شکل (۵-۷) مدت زمان ۱ بودن CNV (t_{cnvh})

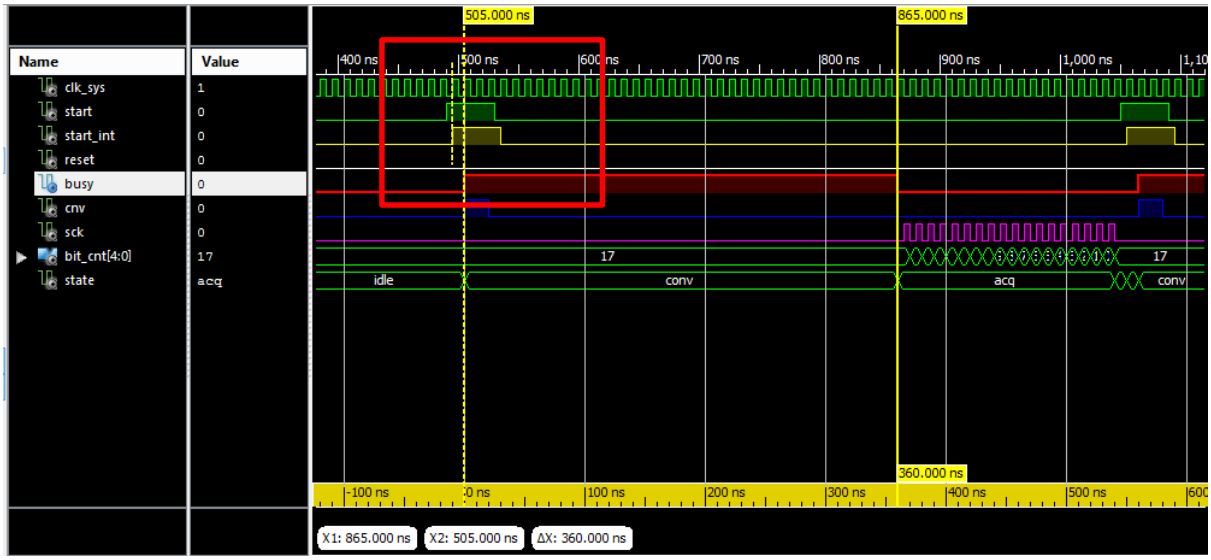
مقایسه سیگنال CNV و Busy در شبیه سازی (شکل ۵-۷) و دیتاشیت (شکل۵-۸).



شکل (۵-۸) سیگنال های CNV و Busy در دیتاشیت

۶-۵- سیگنال Start و Start_INT

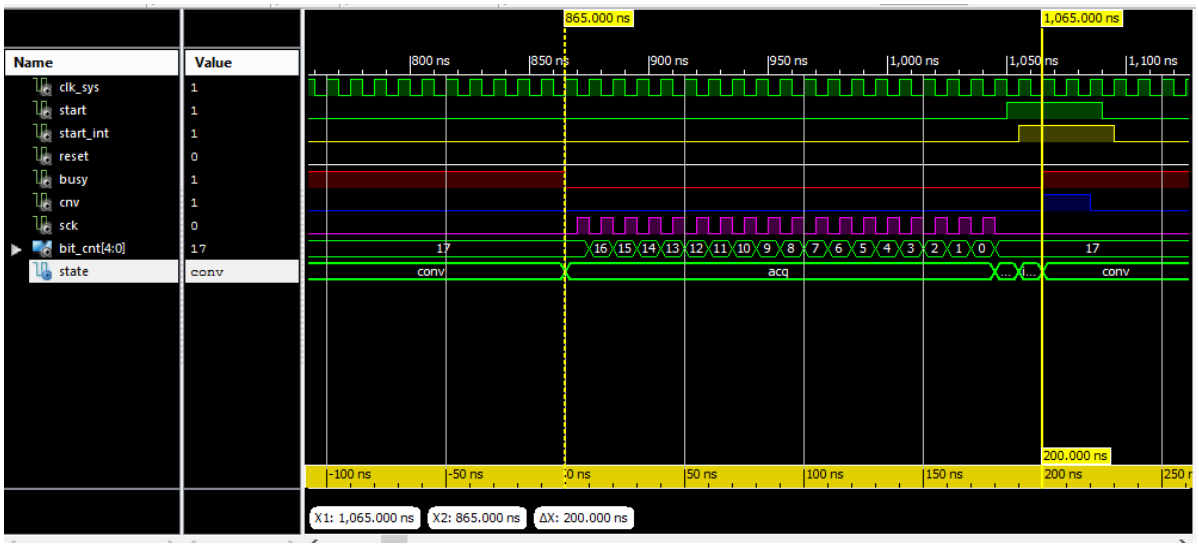
همانطور که از شکل ۹-۵ پیداست، سیگنال start برای دستور به شروع به کار کل سیستم است و تا قبل از آن سیستم بعد از عمل تبدیل و ارسال در حالت idle قرار میگیرد ، سیگنال start_INT ارجاع شده سیگنال start درون برنامه است، بنابراین تغییرات سیگنال start در لبه بالارونده بعدی (خط چین زرد رنگ در شکل ۹-۵) به سیگنال های دیگر منتقل میشود.عملیات ترتیبی در پراس اصلی نیز با توجه به سیگنال start_INT صورت میگیرد.



شکل (۹-۵) سیگنال Start وStart_INT

۷-۵- انتقال بیت های داده

طبق شکل ۱۰-۵ هر بیت داده در هر لبه بالا رونده SCK به صورت MSB First منتقل میشود.



شکل (۱۰-۵) نشان دهنده انتقال MSB First داده ها در زمانی که ACQ برابر با ۲۰۰ نانو ثانیه است.

Iran University of Science and Technology
Department Electrical Engineering

VHDL Final Project

Student Name
Atefeh Bahadori

Student Number
401611503
IUST_9