دانشکده مهندسی مهندسی برق

راه اندازی فرکانس سینتی سایزر LMX2531 به وسیله FPGA

استاد مربوطه:

دکتر میرزاکوچکی

دانشجو:

سید سعید رضوی ظریفی پسند

۴۰۱۶۱۱۸۵۴


بسم الله الرحمن الرحيم

فهرست مطالب

فهرست اشکال

فهرست جداول

فصل ۱:

مقدمه

۱-۱- مقدمه

سیستم سینتی سایزر فرکانس[1] سیستم با فیدبکی است که قادر است با استفاده از یک مرجع ثابت و کاملا پایدار، فرکانسهای متعددی را در یک حلقه قفل شده فاز[2] تامین نماید. سینتی سایزر برای تولید نوسان دلخواه از یک نوسانساز[3] دقیق و یا یک سیگنال زمانبندی مرجع به عنوان ورودی استفاده می کند و در یک بازه فرکانسی می تواند نوسانهایی با فرکانس مطلوب را بر اساس سیگنال مرجع به صورت دقیق و قفل شده تولید کند. سینتی سایزرها برای تولید نوسانات تک فرکانس قابل تنظیم یا مجموعه ای از نوسانات ترکیب شده به کار می روند. این مدارات امروزه در وسایل الکترونیکی زیادی از قبیل انواع گیرنده‌ها و فرستنده های رادیویی، سیستم های رادیویی ماهواره ای و جی پی اس و تجهیزات صوتی دیجیتال استفاده می‌شوند.

[1] Frequency Synthesizer System

[2] Phase-Locked Loop

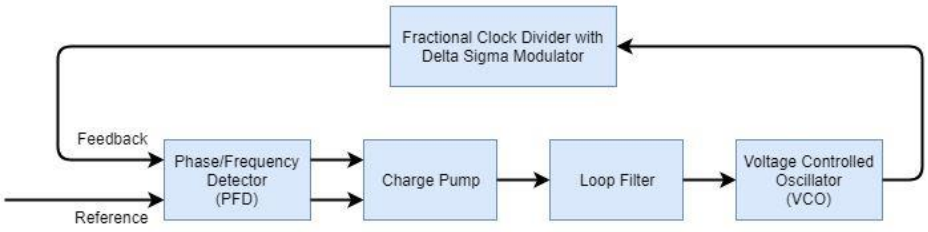
[3] Oscillator

فصل ۲:

ای سیLMX2531

۲-۱- مقدمه

LMX2531 یک ای سی[1] سینتی سایزر فرکانس، با کارایی بالا و کم مصرف می باشد که از مودلاتور دلتا- سیگما[2] و ولتاژ کنترل اسیلاتور[3] استفاده میکند



شکل (۱-۲) Fractional N PLL- با مودلاتور دلتا-سیگما

همچنین این ای سی شامل رگولاتور های ولتاژ با افت اندک[4] با دقت بالا و بسیارکم نویز برای بخش های PLL و VCO میباشد که موجب مصونیت در برابر نویز می شود

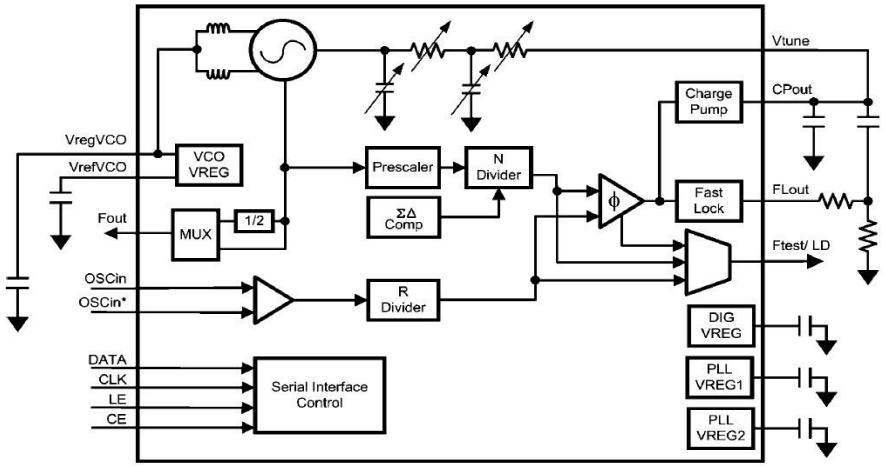
جهت برقراری ارتباط و برنامه ریزی این قطعه از پروتکل 3-wire MICROWIRE استفاده میشود.

۲-۲- ویژگی های قطعه

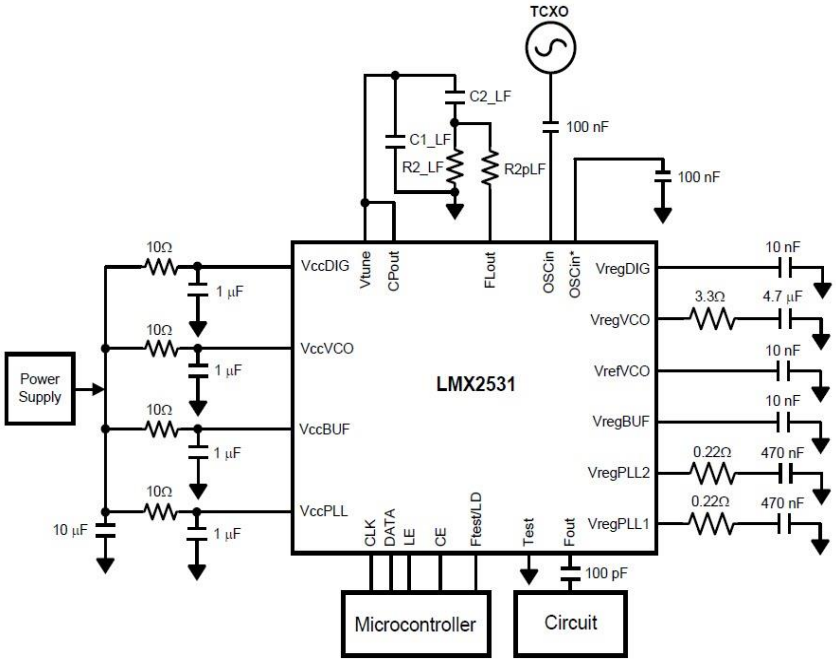
- فرکانس خروجی از 553MHz تا 3132 MHz
- Fractional N PLL با مودلاتور دلتا-سیگما
- قابلیت برنامه ریزی
- FastLock/Cycle Slip Reduction
- Partially Integrated, Adjustable Loop Filter
- Integrated Tank Inductor
- Low Phase Noise

[1] IC
[2] Delta-Sigma Modulator
[3] VOC
[4] Low-Dropout Regulator

۳-۲- بلوک دیاگرام



شکل (۲-۲) بلوک دیاگرام داخلی LMX2531



شکل (۳-۲) پیکربندی پایه های قطعه

۴-۲- برقراری ارتباط و پیکربندی

برقراری ارتباط با قطعه توسط پروتکل[1] MICROWIRE انجام می شود که یک نوع رابط سریال[2] ۳ سیمه با سرعت ۳ مگابیت بر ثانیه است و زیرمجموعه ای از رابط SPI می با شد، MICROWIRE در وا قع یک پورت[3] ورودی/خروجی سریال است.

کنترل ای سی از طریق پایه های LE, CLK, DATA انجام میشود که شرح آن به صورت ذیل می باشد:

جدول (۱-۲) توضیح پین های کنترلی

DATA	I	MICROWIRE serial data input. High impedance CMOS input. This pin must not exceed 2.75V. Data is clocked in MSB first. The last bits clocked in form the control or register select bits.
CLK	I	MICROWIRE clock input. High impedance CMOS input. This pin must not exceed 2.75V. Data is clocked into the shift register on the rising edge.
LE	I	MICROWIRE Latch Enable input. High impedance CMOS input. This pin must not exceed 2.75V. Data stored in the shift register is loaded into the selected latch register when LE goes HIGH.

فرمول محاسبه فرکانس خروجی این ای سی به شرح زیر است:

$$f_{VCO} = f_{PD} \times N = f_{OSCin} \times N/R$$

$$N = N_{Integer} + N_{Fractional}$$

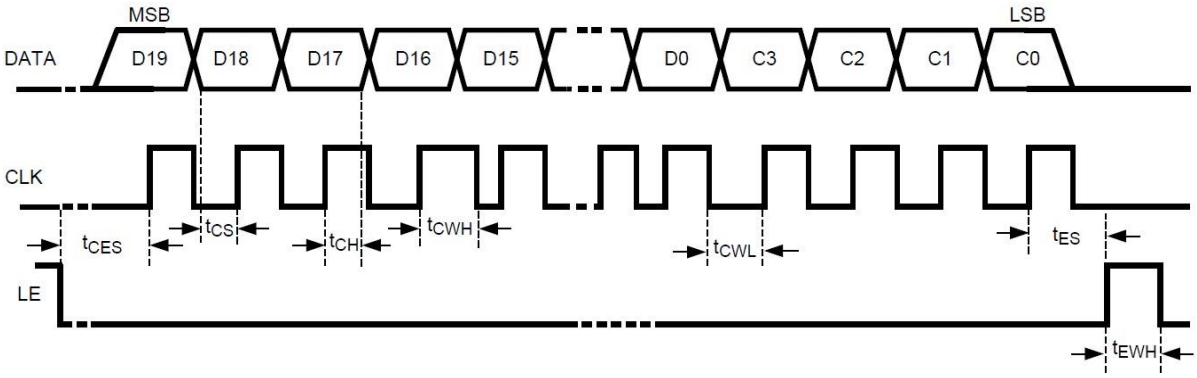
که در آن f_{OSCin} فرکانس کریستال خارجی[4]، R ضریب تقسیم فرکانسی[5] که مقادیر مناسب آن می تواند ۱-۲-۴- ۸- ۱۶ و ۳۲ باشد.

[1] Protocol
[2] Serial
[3] Port
[4] External Crystal
[5] Frequency division coefficient

۱-۴-۲- الزامات زمانبندی انتقال اطلاعات

جدول (۲-۲) MICROWIRE Timing Requirements

		MIN	NOM	MAX	UNIT
t_{CS}	Data to Clock Set-Up Time	25			ns
t_{CH}	Data to Clock Hold Time	20			ns
t_{CWH}	Clock Pulse Width High	25			ns
t_{CWL}	Clock Pulse Width Low	25			ns
t_{ES}	Clock to Enable Set-Up Time	25			ns
t_{CES}	Enable to Clock Set-Up Time	25			ns
t_{EWH}	Enable Pulse Width High	25			ns



شکل (۴-۲) – نحوه عملکرد پین ها جهت انتقال داده ها

۲-۴-۲- برنامه ریزی

برای کنترل عملکرد این ای سی از ۱۱ رجیستر[1] ۲۴ بیتی آن استفاده می شود. این قطعه در واقع دارای ۱۴ رجیستر میباشد که رجیستر های R10,R11 و R13 به صورت پنهان بوده و غیر قابل دستر سی برای کاربر هستند.

شیفت رجیستر[2] ۲۴ بیتی برای برنامه ریزی غیر مستقیم رجیستر استفاده می شود. شیفت رجیستر از یک بخش داده و یک بخش آدرس تشکیل شده است.

برای انتقال داده پایه LE باید Low باشد تا داده های سریال بر روی لبه بالارونده کلاک[3] در شیفت رجیستر منتقل می شوند (داده ها به صورت MSB برنامه ریزی می شوند).

[1] Register
[2] Shift Register
[3] Clock

۳-۴-۲- رجیسترها

نحوه قرارگیری داده ها داخل رجیستر به صورت زیر است:

DATA[19:0]																				CONTROL[3:0]			
MSB																							LSB
D19	D18	D17	D16	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	C3	C2	C1	C0

شکل (۵-۲) -ساختار رجیستر

چهار بیت آخر این رجیستر ، بخش آدرس را تشکیل می دهد CTRL[3:0] ، که برای تعیین آدرس رجیستر داخلی استفاده می شود. ۲۰ بیت باقیمانده بخش داده را تشکیل می دهند.DATA [19:0] .

جهت راه اندازی ای سی ، توالی برنامه ریزی رجیستر ها باید رعایت شود و ترتیب آن به صورت زیر است

جدول (۳-۲) توالی مقدار دهی اولیه به رجیستر ها

REG.	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	DATA[19:0]																				C3	C2	C1	C0
R5 INIT1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
R5 INIT2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
R5	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0	1	0	1
R12	Program R12 as shown in the complete register map.																				1	1	0	0
R9	Program R9 as shown in the complete register map.																				1	0	0	1
R8	See individual section for Register R8 programming information. Programming of this register is necessary under specific circumstances.																				1	0	0	0
R7	See individual section for Register R7 programming information.																				0	1	1	1
R6	See individual section for Register R6 programming information.																				0	1	1	0
R4	See individual section for Register R4 programming information. Register R4 only needs to be programmed if FastLock is used.																				0	1	0	0
R3	See individual section for Register R3 programming information.																				0	0	1	1
R2	See individual section for Register R2 programming information.																				0	0	1	0
R1	See individual section for Register R1 programming information.																				0	0	0	1
R0	See individual section for Register R0 programming information.																				0	0	0	0

جهت اطمینان از عملکرد صحیح تراشه LDO زمان نوشتن بین رجیستر(init3) R5 و رجیستر R1 نباید کمتر از ۱۰ میلی ثانیه باشد.

جدول (۴-۲) - تمامی بیت های قابل برنامه ریزی رجیستر ها

REGISTER	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	DATA[19:0]																				C3	C2	C1	C0
R0	N [7:0]								NUM [11:0]												0	0	0	0
R1	0	0	1	ICP [3:0]			N [10:8]		NUM [21:12]												0	0	0	1
R2	0	1	DEN [11:0]												R [5:0]						0	0	1	0
R3	DIV2	FDM	DITHER [1:0]		ORDER [1:0]		FoLD [3:0]		DEN [21:12]												0	0	1	1
R4	0	0	ICPFL [3:0]				TOC [13:0]														0	1	0	0
R5	1	0	0	0	0	REG_RST	0	0	0	0	0	0	0	EN_DIG LDO	EN_PLL LDO 2	EN_PLL LDO 1	EN_VCO LD	EN_OSC	EN_VCO	EN_PLL	0	1	0	1
R6	0	XTLSEL [2:0]			VCO_ACI_SEL [3:0]				EN_LPF LTR	R4_ADJ [1:0]		R4_ADJ_FL [1:0]		R3_ADJ [1:0]		R3_ADJ_FL [1:0]		C3_4_ADJ [2:0]			0	1	1	0
R7	0	0	XTLMAN [11:0]												XTLDIV [1:0]		0	0	0	0	0	1	1	1
R8	0	0	0	0	0	0	1	LOCK MODE		0	0	0	0	0	0	0	0	0	0	XTL MAN 2	1	0	0	0
R9	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	0	1	0	1	0	0	1
R12	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	1	0	0	0	1	1	0	0

فصل ۳:

طراحی ماژول ارسال داده

۱-۳- مقدمه

SPI مخفف عبارت Serial Peripheral Interface است و یک پروتکل ارتباطی سریع میباشد. از این پروتکل در ارتباطات با فاصله کوتاه استفاده می‌شود و نوع ارتباط آن سنکرون بوده و در کامل‌ترین حالت، چهار سیم یا مسیر ارتباطی برای پیاده‌سازی این پروتکل به کار می‌رود.

به کمک این چهار سیم، امکان ایجاد یک ارتباط Full duplex به صورت Master/Slave فراهم می‌شود. اگر بیش از دو وسیله به کمک این پروتکل به یکدیگر متصل شوند، همواره یکی از آنها master و بقیه slave خواهند بود .یکی از چهار خط ارتباطی پروتکل SPI سیگنال کلاک است. وسیله‌ای که سیگنال کلاک را ارسال می‌کند master نامیده می‌شود و وسیله‌ای که آن را دریافت می‌کند slave نام دارد. دومین خط ارتباطی، سیگنال انتخاب slave یا SS (slave select) است.

از آنجاییکه SPI یک ارتباط سنکرون است، هر بیت داده در master و slave همزمان با کلاک ارسال یا دریافت می‌شود.. طول این داده‌ها بستگی به قراردادی دارد که در یک وسیله خاص وجود دارد و ممکن است ۸ بیت، ۱۰ بیت، ۱۶ بیت، ۳۲ بیت یا هر مقدار دیگری باشد.

میکرو وایر یک رابط سریال ۳ سیمه با سرعت ۳ مگابیت بر ثانیه است که زیرمجموعه ای از رابط SPI است و در واقع یک پورت ورودی/خروجی سریال است. این گذرگاه در EEPROM ها و سایر تراشه های جانبی نیز یافت می شود.

در این فصل با مراجعه به دیتا شیت ای سیLMX2531 به بررسی زمانبندی ارسال داده بر مبنای این پروتکل پرداخته و در ادامه به طراحی ماژول این ارتباط می پردازیم.

از آنجایی که ای سی LMX2531 پس از دریافت دیتا ، هیچ پاسخی ارسال نمیکند در نتیجه در طراحی این ماژول ارسال دیتا، هیچ پورتی جهت دریافت و ذخیره سازی دیتای ارسالی از این ای سی در نظر گرفته نشده است.

۲-۳ – زمانبندی

در ارتباط سنکرون با هر لبه بالا رونده کلاک ، دیتا را ارسال میکنیم
ارسال دیتا برای ماژول نیازمند توجه و رجوع به جدول زمانبندی (جدول ۲-۲) است تا دیتای ارسالی به شکل صحیح و قابل دریافت توسط قطعه، ارسال گردد .
حال به شرح مقادیر داخل جدول زمانبدی می پردازیم:

t_{CES} : فاصله لبه پایین رونده LE تا اولین لبه بالا رونده کلاک SPI که براساس جدول این مقدار برابر ۲۵ ns میباشد.

t_{CWH} : نیم سیکل مثبت کلاک

t_{CWL} : نیم سیکل منفی کلاک

با توجه به حداقل مقادیر داده شده برای پارامتر های فوق، دوره تناوب کلاک ۵۰ ns بوده که در نتیجه فرکانس آن حداکثر ۲۰ MHz می باشد

t_{ES} : بیانگر این است که از آخرین کلاک تا غیر فعال شدن LE(یک شدن LE) حداقل باید ۲۵ ns زمان وجود داشته باشد

t_{EWH} : زمانی که ما به انتهای پکت ارسالی میرسیم باید LE را غیر فعال کنیم و برای ارسال دوباره باید LE مجدداً فعال شود، جهت اعمال غیر فعال بودن LE به دستگاه جانبی حداقل باید ۲۵ ns مقدار LE را غیر فعال نگه داریم تا غیر فعال بودن آن به Slave اعمال شود

هنگامی که دیتایی به صورت سنکرون با کلاک ارسال می شود زمانی که بخواهیم در لبه بالا رونده کلاک SPI دیتا را برداریم برای اینکه را در لبه بالارونده به درستی رایت کنیم دیتا باید یک مدت زمانی قبل از لبه بالارونده در ورودی ماژول آماده باشد و همچنین جهت پایداری و اطمینان از دریافت درست دیتا در سمت گیرنده، پکت ارسالی باید زمانی کوتاه بعد از لبه بالا رونده کلاک در ورودی ثابت بماند.

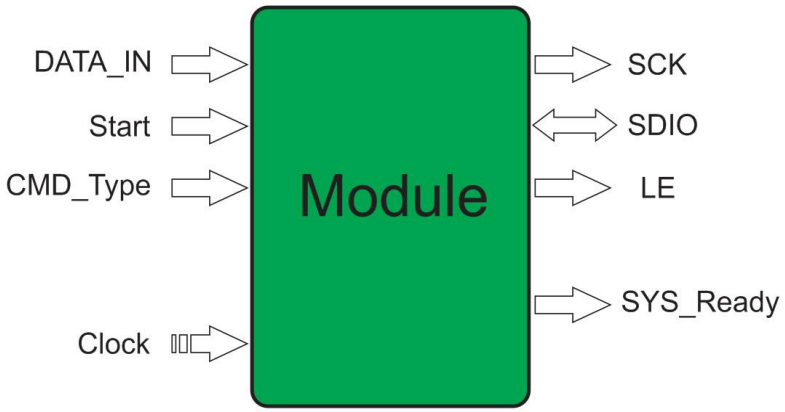
t_{CS} : مدت زمان لازم جهت آماده سازی دیتای ارسالی قبل از لبه بالا رونده

t_{CH} : مدت زمان لازم جهت پایداری و ثابت ماندن دیتای ارسالی بعد از لبه بالا رونده

ایجاد زمان بندی های فوق بر اساس کلاک FPGA انجام میشود که ما در اینجا کلاک FPGA را ۸۰MHz در نظر گفته ایم که در نتیجه دوره تناوب آن ۱۲.۵ ns میگردد که در نتیجه واحد های زمانی سازنده این زمانبندی ها ۱۲.۵ ns است.

۳-۳ – طراحی

برای انجام طراحی و پیاده سازی ماژول ابتدا نیاز است علاوه پورتهای استاندارد مورد نیاز برای پروتکل SPI با در نظر گرفتن ماهیت عملکردی پروتکل ،پورت ها و رجیستر های مورد نیاز دیگری را به آن اضافه کنیم .
اولین پورت مورد نیاز برای طراحی این ماژول Data_IN میباشد که برای نگه داری دیتایی است که قرار است به LMX2531 ارسال شود طول آن با توجه به طول پکت ارسالی به ای سی تعیین میشودکه برای این کاربرد خاص اندازه آن ۲۴ بیتی است
پورتی با ورودی ۲ بیتی در نظر گرفته شده تا طول پکت ارسالی را مشخص کند .(CMD_Type)
نشان میدهد که نوع ارسال داده ها ۸ بیتی، ۱۶ بیتی ، ۲۴ بیتی و یا ۳۲ بیتی خواهد بود.
برای اینکه شروع فرایند ارسال داده را کنترل کنیم پورت دیگری با نام Start تعریف می کنیم عملکرد آن به دین صورت است که با فعال شدن آن(یک لبه بالارونده داشته باشیم) فرایند ارسال بر مبنای طول پکتی که در بالا مشخص کردیم آغاز می شود.
قبل از ارسال مقدار LE برای لحظاتی یک شده و دوباره صفر میگردد با این کارماژول گیرنده ریست شده و ابتدای پکت مشخص میگردد
برای تعیین وضعیت آماده به کار بودن و یا قرار داشتن در حین فرایند ارسال نیاز به پین دیگری داریم به نام SYS_Ready و نحوه عملکرد آن بدین صورت است که مقدار آن در کل پروسه ارسال یک می ماند و با پایان یافتن فرایند ارسال مقدار آن صفر میشود. و در هر زمان با خواندن مقدار آن متوجه میشویم که فرایند ارسال به پایان رسیده و یا در حال انجام است.
پورت دیگر Clock می باشد که وجود آن ضروری است
نکته ای که باید مد نظر قرار داده شود این است که با توجه به اینکه ای سی LMX2531 هیچ مقداری به عنوان پاسخ و یا به عبارتی فیدبک بر نمیگرداند در نتیجه از گذاشتن پورت هایی جهت دریافت خروجی slave و یا جهت اطلاع از آماده شدن دیتا های دریافتی اجتناب شده است
و در پایان ماژول طراحی شده برای ارسال دیتای ما بدین شکل خواهد بود.



شکل (۱-۳) – ماژول طراحی شده

فصل ۴:

پیاده سازی

۱-۴- مقدمه

پس از طراحی ماژول ارسال دیتا نوبت به کد نویسی آن می رسد که به زبان توصیف سخت‌افزار VHDL و در نرم افزار ISE انجام شده که در این فصل به آن می پردازیم

ISE مخفف Integrated Software Environment یک ابزار سنتز[1] و آنالیز[2] طراحی های توصیف شده توسط یکی از زبان های سخت افزاری می باشد. این نرم افزار متعلق به شرکت Xilinx بوده و به طور کلی میتوان قابلیت های آن را به صورت زیر خلاصه نمود

- امکان سنتز و کامپایل طراحی ها
- آنالیز زمانی طراحی ها
- بررسی دیاگرام های RTL[3]
- شبیه سازی عملکرد طراحی در ازای ورودی های مختلف
- پیکربندی و برنامه ریزی دستگاه مقصد.

ورژن برنامه استفاده شده ۱۴.۷ می باشد

[1] Synthesis

[2] Analyze

[3] Register Transfer Level

۲-۴- کدنویسی

در شروع کد نویسی بر اساس ماژولی که در فصل قبل طراحی کردیم ابتدا پورت های ماژول را در بخش موجویت[1] تعریف می کنیم و ادامه کار را در بخش معماری[2] که توصیفی رفتاری[3] از عمل کرد ماژول است ،کد نویسی را ادامه می دهیم در این قسمت که محیطی هم زمان[4] میباشد ابتدا ارجاعاتی که باید به خروجی داده شود را به صورت رجیستر[5] تعریف می کنیم

کلاک FPGA ما ۸۰ MHz است که در نتیجه دوره تناوب[6] آن ۱۲.۵ نانو ثانیه است پس اگر ما شمارنده[7] ای درست کنیم هر واحدی که به شمارنده اضافه میشود ۱۲.۵ ns می باشد

ابتدا مقادیر ورودی را بافر میکنیم تا در صورتی که حین انجام فرایند دیتای جدیدی وارد شد تاثیری بر روی دیتای در حال ارسال نگذارد

شمارنده ای که قرار است کلاک پروتکل ما را از روی کلاک اصلی سیستم ایجاد کند SCK_clock_Divider نام دارد ،این کانتر در قسمت ابتدای برنامه قرار دارد و با هر کلاک در حال افزایش بوده و همواره در حال شمارش است اگر این شمارنده بزرگتر از ۱ شود رجیستری با عنوان SCK_INT صفر می شود در غیر این صورت با توجه به مقدار اولیه تعیین شده برابر ۱ میشود و اگر کانتر به ۳ رسید شمارنده ریسیت می شود

برای تشخیص لبه بالا رونده کلاک از سیگنال های Start_INT و Start_PREV همراه با شرط زیراستفاده کردیم

If (Start _INT = ‘1’ AND Start _PREV = ‘0’ AND Busy_Int ‘0’)

که ارجاعات زیر برای انها وجود دارد:

Send_Int <= Send و Send_Prev <= Send_Int

اگر ماژول در حال ارسال داده باشد شرط فوق برقرار نمیشود.

به کمک CMD_Type و ارایه SPI_Data_Bit_Width طول پکت ارسالی را مشخص میکنیم.

طراحی بدین صورت انجام شد که در ابتدای ارسال برای مدت کوتاهی LE برابر یک شود و سپس صفر گردد ، از انجایی که پریود کلاک ما ۱۲.۵ نانو ثانیه است و حداقل پالس داخل دیتاشیت ۲۵ ns بیان شده به ۲ کلاک نیاز داریم که با اجرای کد زیر فراهم می شود

```
if (LE_Disable_Counter < to_unsigned(2,3))then
    LE_Disable_Counter          <=  LE_Disable_Counter + 1;
    LE_INT                          <=  '1';
end if;
```

شکل (۱-۴) – بخشی از کد-۱

[1] Entity
[2] Architecture
[3] Behavioral
[4] Concurrent
[5] Register
[6] Period
[7] Counter

شمارنده SCK_Clock_Divider از ۰ تا ۱ می شمارد که در این زمان مقدار کلاک SPI صفر است در نتیجه زمانی که صفر است ما در لحظه شروع پریود این کلاک هستیم.

```
if (SCK_Clock_Divider < to_unsigned(2,2))then
    SCK_INT                         <=  '0';
end if;
if (SCK_Clock_Divider = to_unsigned(3,2))then
    SCK_Clock_Divider               <=  (others=>'0');
end if;
```

شکل (۲-۴) – بخشی از کد -۲

زمان برقراری شرط زیر درست در اولین پریود اولین کلاک هستیم حال دیتای بافر شده به خروجی انتقال می یابد.

```
if (LE_Disable_Counter < to_unsigned(2,3))then
    LE_Disable_Counter          <=  LE_Disable_Counter + 1;
    LE_INT                          <=  '1';
end if;
```

شکل (۳-۴) – بخشی از کد -۳

زمانی که آخرین بیت را به خروجی ارسال کردیم باید به اندازه یک کلاک صبر کنیم تا آخرین بیت هم بدرستی ارسال شود و بعد ارسال را متوقف کنیم

```
if (SPI_Data_Bit_Width_Buffer = to_unsigned(0,5))then
    SPI_End_Send_Data       <=  '1';
```

شکل (۴-۴) – بخشی از کد-۴

زمانی که فرایند ارسال داده به پایان رسید لبه پایین رونده اتفاق می افتد و کلاک SCK هم به انتهایش میرسد.

```
if (SPI_End_Send_Data = '1')then
    SYS_Ready_INT           <=  '0';
    SCK_Disable             <=  '0';
    Set_SCK_Disable         <=  '0';
    SPI_Write_State         <=  '0';
    SPI_End_Send_Data       <=  '0';
end if;
```

شکل (۵-۴) – بخشی از کد-۵

فصل ۵:

تست و شبیه سازی

۱-۵- مقدمه

جهت تست ماژول نوشته شده نیاز است تا ماژولی دیگر تحت عنوان تست بنچ نوشته شود تا سیگنال‌های لازم برای ماژول دیگری که به آن وصل می‌شود تا عملکرد آن مورد بررسی قرار گیرد را تامین کند.
اساس طراحی تست بنچ توانایی شبیه‌سازی برنامه‌های HDL می‌باشد. شبیه‌سازی اجازه توصیف سخت‌افزاری HDL برای گذر از تأیید طراحی و همچنین اجازه اکتشاف معماری را می‌دهد. طراح می‌تواند با تغییرات متعدد طراحی از پایه طرح را آزمایش کند. سپس رفتار آن‌ها را در شبیه‌سازی مقایسه کند؛ بنابراین در شبیه‌سازی برای طراحی موفق بسیار مهم است.

۲-۵- تست بنچ

برای تست عملکردی ماژول طراحی شده و شبیه سازی مقدار دهی به ای سی از شبیه سازISIM استفاده میکنیم برای این کار بعد از درست کردن فایل تست بنچ از روی کد نوشته شده ابتدا سیگنال های مورد نظر را تعریف میکنیم

```
--Inputs
signal Clock         : std_logic := '0';
signal Start         : std_logic := '0';
signal Data_IN    : unsigned(31 downto 0) := (others => '0');
signal CMD_Type   : unsigned(1 downto 0) := (others => '0');

 --BiDirs
signal SDIO : std_logic;

 --Outputs
signal SCK : std_logic;
signal LE : std_logic;
signal SYS_Ready : std_logic;
```

شکل (۱-۵) – بخشی از کد شبیه سازی- ۱

پریود کلاک را روی 12.5ns تنظیم میکنیم تا فرکانس کلاک ما 80MHz شود

سپس با استفاده از فرایند PORT MAP سیگنال های تعریف شده را به پورت خروجی متصل می کنیم

```
  -- Instantiate the Unit Under Test (UUT)
uut: LMX2531 PORT MAP (
        Clock         => Clock,
        SCK           => SCK,
        SDIO       => SDIO,
        LE            => LE,
        SYS_Ready     => SYS_Ready,
        Start         => Start,
        Data_IN    => Data_IN,
        CMD_Type   => CMD_Type
      );
```

شکل (۲-۵) – بخشی از کد شبیه سازی- ۳

سپس از با استفاده از پروسس"clock_process " کلاک FPGA را تولید میکنیم.

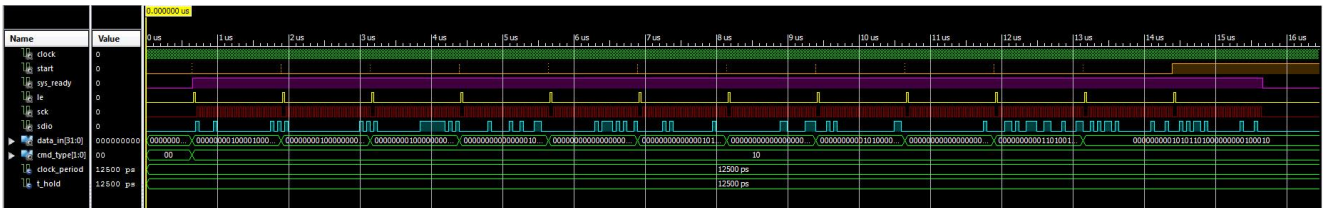
```
-- Clock process definitions
Clock_process :process
begin
     Clock <= '0';
     wait for Clock_period/2;
     Clock <= '1';
     wait for Clock_period/2;
end process;
```

شکل (۳-۵) – بخشی از کد شبیه سازی- ۳

با توجه که ارسال ما از نوع MSB First است و ۲۴ بیتی است مقدار "CMD_Type " را برابر ۲ قرار دادیم

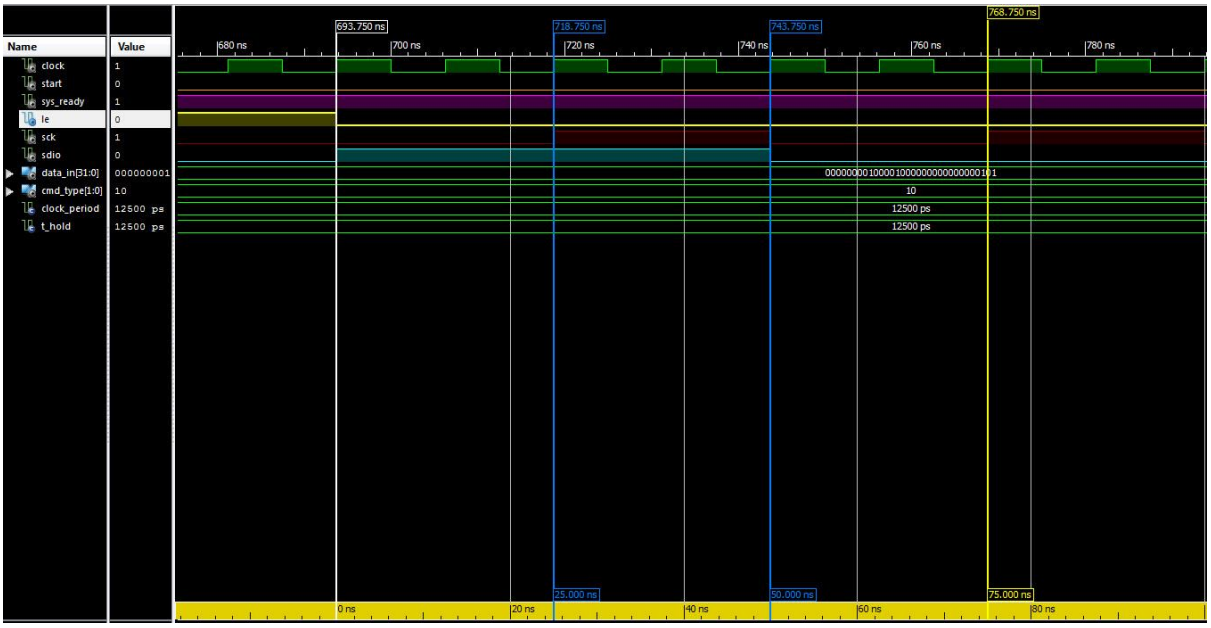
۳-۵- شبیه سازی

مقادیر ارسالی جهت مقدار دهی به رجیستر های قطعه LMX2531 را بر اساس دیتا شیت برای داشتن فرکانس 1800MHz در خروجی درون پروسس "stim_proc" تنظیم میکنیم.



شکل (۴-۵) –شبیه سازی

مشاهده میشود که زمانبندی ارسال و فاصله سیگنال ها بر اساس جدول زمان بندی دیتاشیت است.



شکل (۵-۵) – زمانبندی شبیه سازی

مراجع

مراجع

[1] LMX251. Datasheet

[2] Programming Multiple Devices Using a Single SPI Bus. Murata Company Application note 87

[3] https://nl.mathworks.com/help/msblks/ref/fractionalnpllwithdeltasigmamodulator.html

[4] State Machine using VHDL FPGA Implementation :Orhan Gazi, A Çağrı Arlı

[5] Digital Fundamentals: Floyd, Thomas, Buchla, David

[6] Circuit Design with VHDL Volnei A. Pedroni

[۷]