طراحی سیستم های مبتنی بر

FPGA

آزمایش اول

طراحی و پیاده سازی یک سیستم برای انتقال و پردازش اطلاعات

اعضای تیم:

علیرضا جابری راد

مهدی رادمان

نیما سلیمی

فهرست:

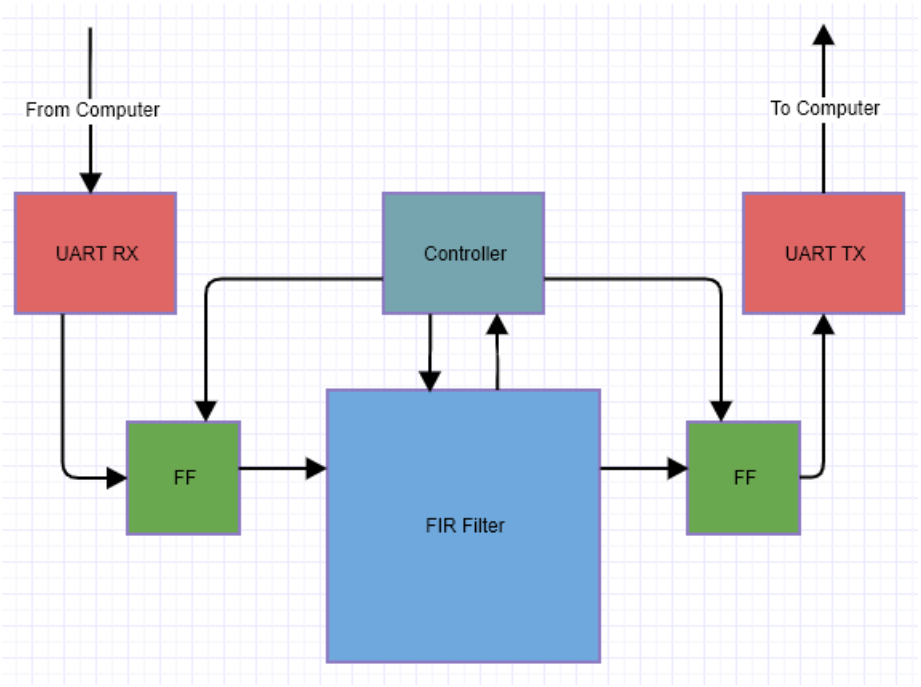
مقدمه:

در این آزمایش به طراحی یک سیستم UART کامل و فرستادن و دریافت اطلاعات از طریق یک کد MATLAB و پیاده سازی این سیستم بر روی برد FPGA موجود در آزمایشگاه (Cyclone II DE2:2C35) و دریافت و مشاهده اطلاعات می پردازیم.

Altera DE2 board (Cyclone 2C35)

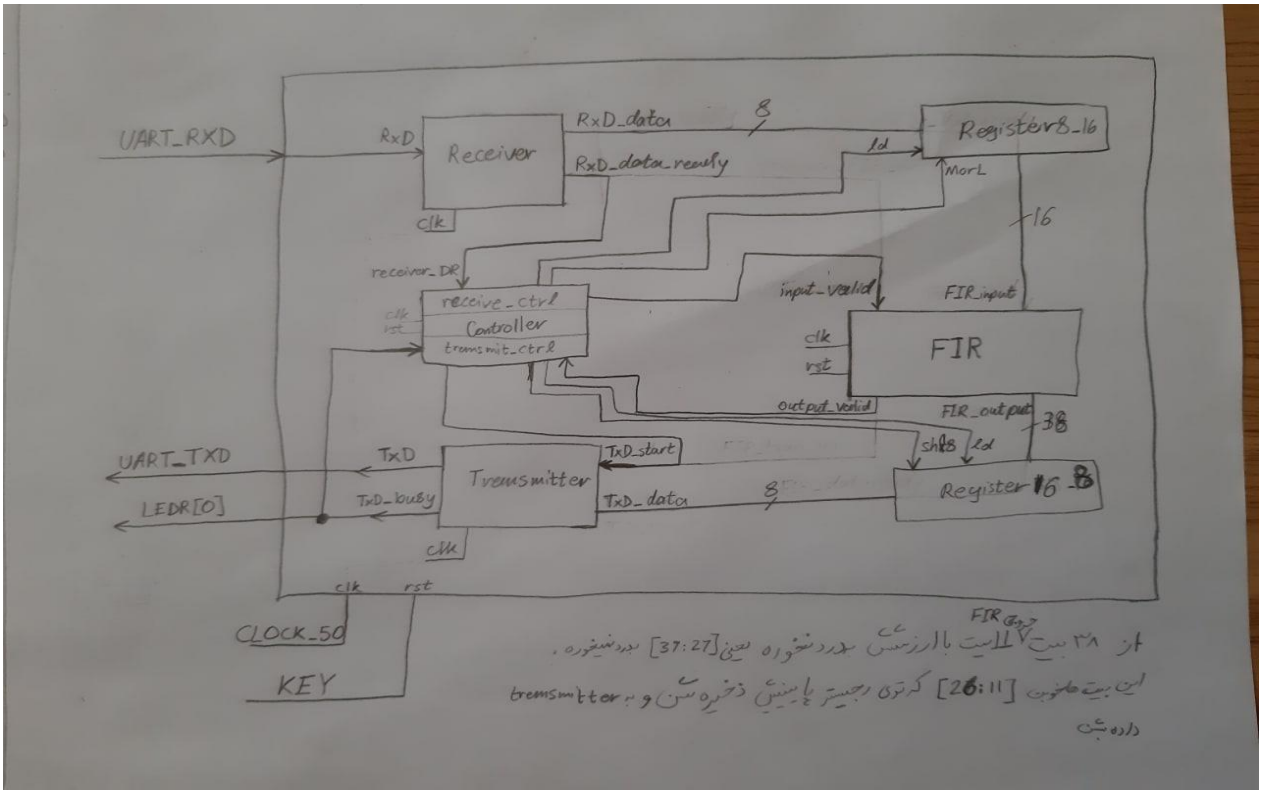
شرح کلی آزمایش:

در این آزمایش یک سیستم فرستنده و گیرنده کامل شامل تمامی واحد های transmitter ، receiver ، controller ، datapath ، BaudTickGen و دو Register متصل به واحد گیرنده و فرستنده و همچنین FIR استفاده شده در پروژه اول، طراحی شده است. پس از اطمینان از عملکرد سیستم در محیط quartus II سنتز شده و برای تست بر روی برد FPGA آزمایشگاه استفاده شده است. ابتدا به توضیحات مسیر داده و واحد کنترل پرداخته و سپس هر کدام از واحد های مربوطه را توضیح می دهیم. همچنین فایل گزارشکار شامل کد های وریلاگ ، تست بنچ ها و فایل سنتز کوارتوس می باشد.



شمای کلی سیستم شامل مسیر داده و واحد کنترل:

شمای کلی سیستم مطابق مسیر داده زیر است که سیگنال های ورودی خروجی سیستم مشخص شده اند. همچنین تمامی اسامی ورودی و خروجی ها و سیم های نشان داده شده، در کد وریلاگ نیز همینطور نام گذاری شده اند. مسیر داده نیز دقیقا مانند شمای کلی نشان داده شده در فایل گزارشکار شکل 1 طراحی شده است. مسیر داده نوشته شده در کد وریلاگ شامل تمامی واحد ها به جز کنترل بوده است و واحد BaudTickGen نیزدر کد فرستنده و گیرنده استفاده شده است.



واحد گیرنده:

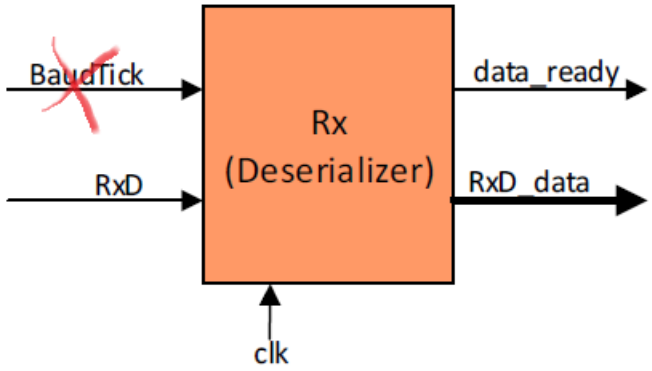
واحد گیرنده طراحی شده شبیه به واحد گیرنده زیر از گزارشکار است(از توضیحات دوباره صرف نظر می کنیم و فقط تفاوت ها و عملکرد را توضیح می دهیم) ولی با این تفاوت که ورودی ها شامل BuadTick نمیشوند و کد BuadTickGen در داخل ماژول گیرنده آورده شده است.همچنین گیرنده ورودی rst را نمی گیرد بلکه ورودی rst مربوطه در واحد کنترل اعمال می شود. کارایی ماژول دقیقا مطابق گزارشکار آزمایشگاه بوده و 8 بیت RxD را به صورت تک تک می گیرد و در خروجی 8 بیته RxD_data قرار می دهد که به ورودی رجیستر رسیور برود و همچنین پس از آماده شدن به مدت یک کلاک سایکل data_ready را 1 می کند و به واحد کنترل می فرستد.

-این ماژول با نام async_receiver ذخیره شده است.پارامتر های کد رسیور:

```
parameter Oversampling = 4;

parameter ClkFrequency = 50000000;

parameter Baud = 115200;
```



واحد فرستنده:

واحد فرستنده طراحی شده شبیه به واحد فرستنده گزارشکار است با این تفاوت که ماژول BuadTickGen در خود کد فرستنده آورده شده است. و همانند گیرنده ، rst به واحد کنترل ارسال می شود و در فرستنده ورودی rst نداریم. فرستنده وظیفه دریافت 8 بیت TxD_data از رجیستر فرستنده و تحویل آن به صورت بیت به بیت به خارج از ماژول با خروجی TxD را دارد. و همچنین وقتی در حال انجام کار است خروجی TxD_busy(Busy) را 1 می کند و نگه می دارد. همچنین با دریافت 1 از ورودی TxD_start کار خود را شروع می کند. مابقی عملکرد سیستم کاملا مشابه با توضیحات گزارشکار آزمایشگاه بوده است.

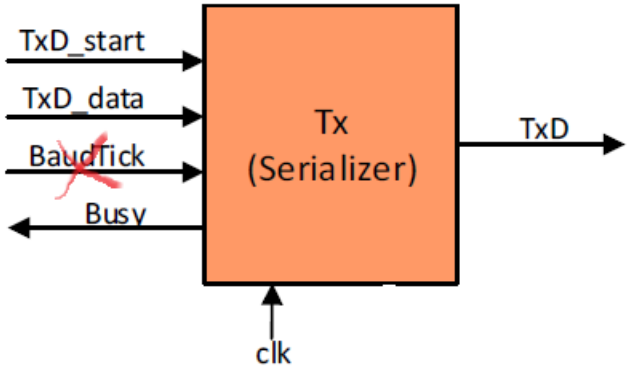
کد فرستنده با نام async_transmitter ذخیره شده است.

پارامتر های مربوط به کد وریلاگ ترنسمیتر:

parameter ClkFrequency = 50000000

parameter Baud = 115200

parameter Oversampling = 1;



رجیستر رسیور:

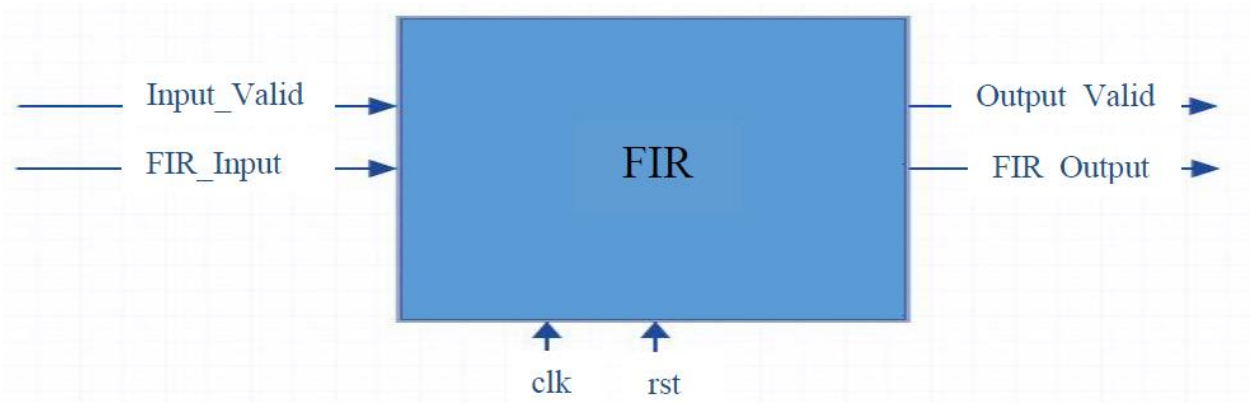
رجیستر رسیور و ترنسمیتر بسیار ساده طراحی شده اند و مطابق شکل1 مسیر داده، ورودی های ld وmorl (بزرگ تر یا کوچکترless or more) را از واحد کنترل می گیرند. که ld در واقع شروع به کار این رجیستر را مشخص می کند و اگر ورودی کنترلی morl یک بود آن را در 8 بیت بالایی Rxd_data و اگر صفر بود در 8 بیت پایینی Rxd_data ذخیره می کند.این رجیستر 8 بیت Rxd_data را از رسیور گرفته و خروجی را در 16 بیت FIR_input به ورودی FIR می فرستد.

رجیستر ترنسمیتر:

این رجیستر نیز بسیار ساده طراحی شده است و ورودی های ld و shf8 را مطابق مسیر داده شکل 1 می گیرد. خروجی FIR ، 38 بیت است که ما فقط بیت [26:11] آن را می گیریم و مابقی را احتیاج نداریم. و ورودی را در یک reg 16 بیتی ذخیره می کنیم با ld شروع به کار می کند و اگرshf8 یک بود 8 بیت اول وگرنه 8 بیت دوم را در دو مرحله به خروجی 8 بیتی می دهد تا به ترنسمیتر برود.

سیستم FIR :

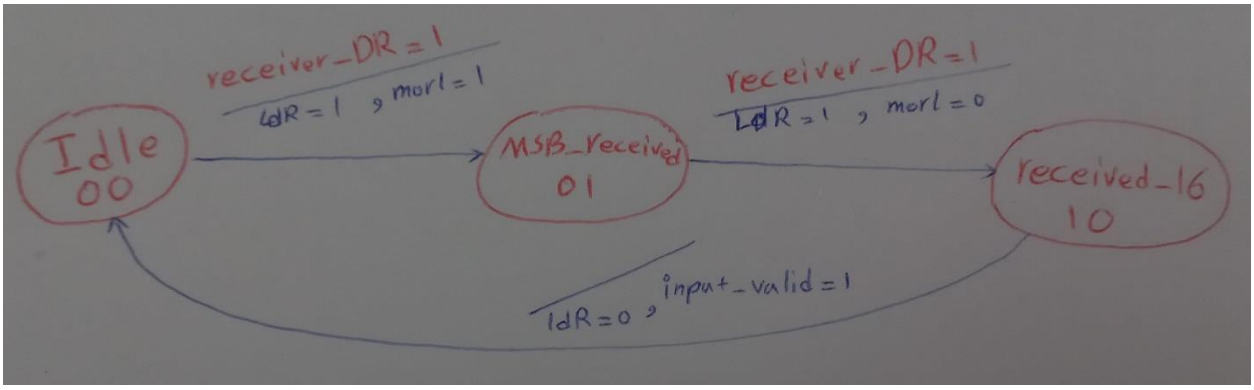
در آزمایش از FIR پروژه قبل استفاده شده که توضیحات لازم در فایل زیپ گذاشته شده است و صرفا به عملکرد کلی مدار اکتفا می کنیم. این واحد ورودی 16 تایی از رجیستر گیرنده می گیرد و خروجی 38 بیتی به رجیستر فرستنده تحویل می دهد که البته فقط بیت [26:11] آن را استفاده می کنیم. کلیه فایل های v. موجود در پوشه به غیر از فایل های ذکر شده در بالا کنترلر و مسیر داده و تاپ ماژول UART که جلوتر توضیح می دهیم برای فایل FIR است. همچنین با فرستادن ورودی input_valid این واحد شروع به کار کرده و پس از انجام عملیات output_valid را برابر 1 کرده و به منظور خاتمه کار به واحد کنترل می فرستد.



واحد کنترل:

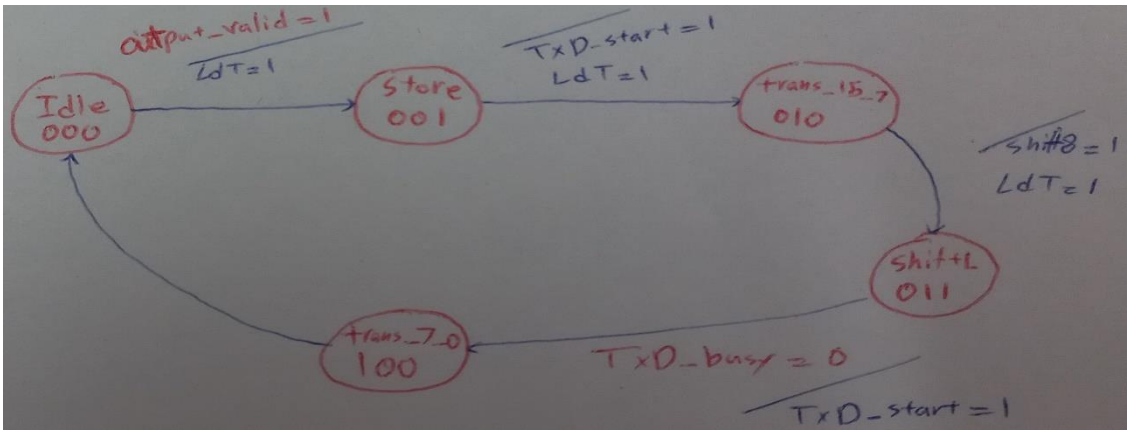
واحد کنترلر به صورا mealy state machine طراحی شده است و شامل دو قسمت است که جدا از هم کار می کنند. قسمت receive_ctrl و قسمت transmit_ctrl که عملکرد این دو به شکل زیر است.

Receive_ctrl



همان طور که مشخص است استیت با گرفتن receiver_DR = 1 ، ldT وmorl را یک کرده و به رجیستر رسیور می فرستد و به استیت 01 می رود. سپس ldR را یک نگه داشته و morl را صفر می کند و به استیت 10 می رود. سپس input_valid را یک می کند و به ورودی FIR می فرستد و ldR یک می شود و به استیت 00 برمیگردد.

Transmit_cntl

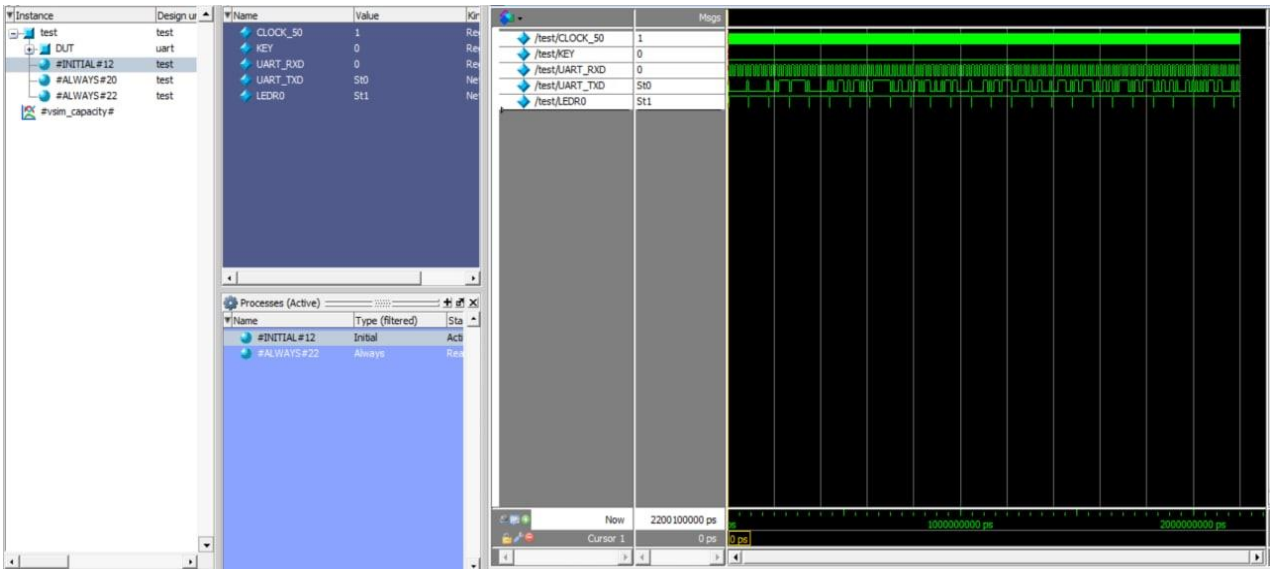


وقتی ورودی output_valid یک شود ldT یک شده وبه استیت بعد میرود. سپس TxD-start یک شده و به استیت بعد میرود.سپس shf8 به مدت یک سیکل یک میشود و به استیت بعد میرود. هر وقت TxD-busy صفر شد TxD_start یک میشود و به ترنسمیتر فرمان شروع صادر میکند وبه استیت آخر میرود.

ماژول کلی و تست بنچ ها:

ماژول cu واحد کنترل است و ماژول dp شامل تمامی واحد ها به جز کنترلر بوده و ماژول uart تاپ ماژول شامل این دو ماژول است. که تست بنچ uart با نام test.v ذخیره شده است.

Final test module uart:

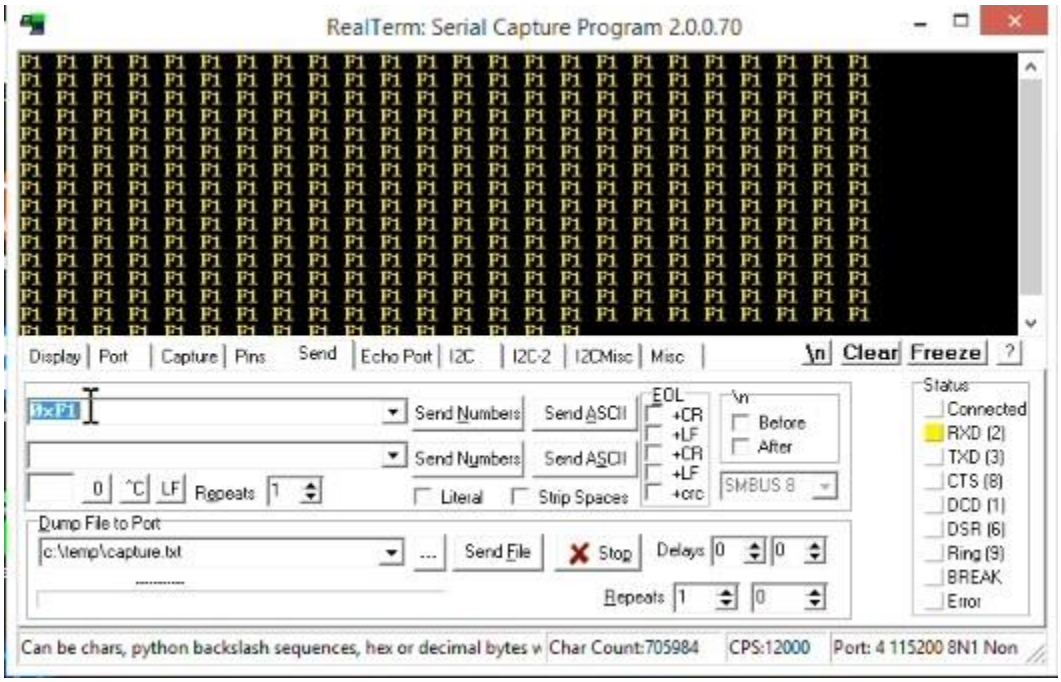


این سیمولیشن مربوط به آخرین ویرایش کد بوده است که مشکلی در مسیر داده داشت و برطرف شد. سیمولیشن بالا کاملا به درستی کار میکند(هم روی فایل کد نویسی شده و هم روی فایل خروجی سنتز vo. به همراه فایل sdo.) و به درستی در نرم افزار Quartus II سنتز میشود ولی به دلیل نبود فرصت روی برد تست نشده است. نتیجه سنتز کد بالا در Quartus II را در زیر مشاهده میکنید.

Flow Summary	
Flow Status	Successful - Wed Nov 17 18:03:25 2021
Quartus II 64-Bit Version	13.0.1 Build 232 06/12/2013 SP 1 SJ Web Edition
Revision Name	uart
Top-level Entity Name	uart
Family	Cyclone II
Device	EP2C35F672C6
Timing Models	Final
Total logic elements	1,358 / 33,216 (4 %)
Total combinational functions	804 / 33,216 (2 %)
Dedicated logic registers	1,192 / 33,216 (4 %)
Total registers	1192
Total pins	5 / 475 (1 %)
Total virtual pins	0
Total memory bits	1,024 / 483,840 (< 1 %)
Embedded Multiplier 9-bit elements	2 / 70 (3 %)
Total PLLs	0 / 4 (0 %)

توضیحات آزمایش روی برد دانشگاه:

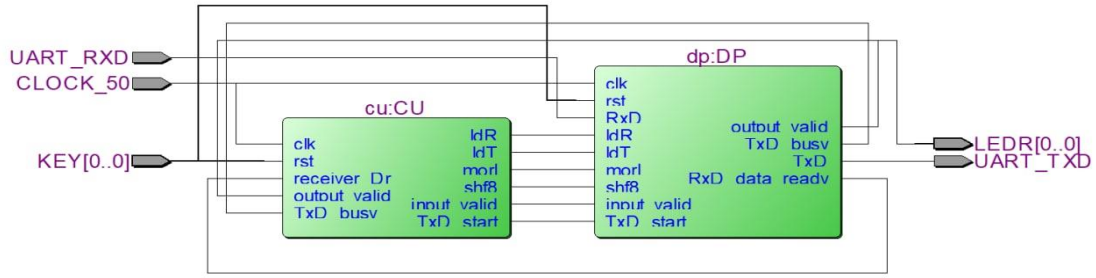
در جلسه اول آزمایش کد ماژول های گیرنده و فرستنده و BuadTickGen زده شده بود که با wire کردن این ها در جلسه اول صرفا یک ورودی داده و همان را دریافت کردیم که همین موضوع صحت کلی این سه ماژول را به ما نشان می دهد. ویدوی ضبط شده این عملیات نیز در فایل گزارش کار موجود است که دو عدد hex به عنوان ورودی داده می شود و همان عدد دریافت شده و در نمایشگر قابل مشاهده است فقط به علت نبود کد واحد کنترل، این عملیات متوقف نشده و بار ها عدد خروجی چاپ میشود.



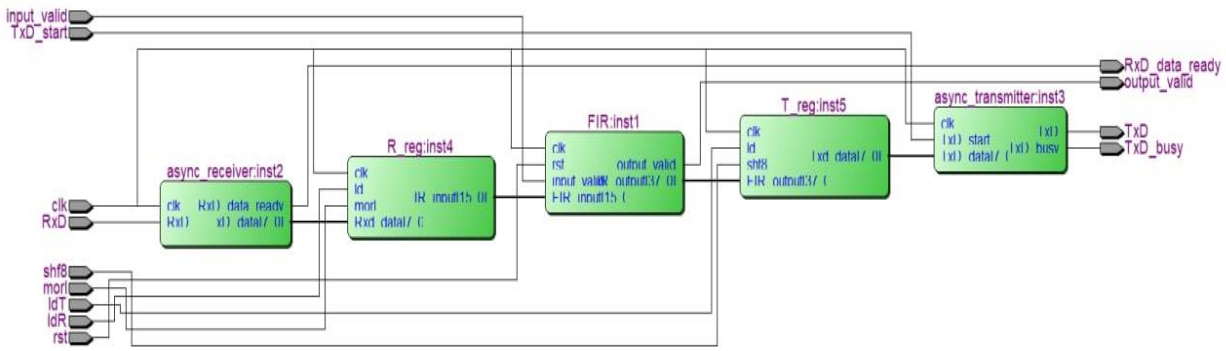
در جلسه آخر نیز تمام کد ها به درستی سنتز شد ولی در اجرا دچار مشکل شدیم که در ادامه مشکل برطرف شد و از فایل سنتز شده هم در وریلاگ تست گرفته شد و عملکردش کاملا مورد تایید بود، ولی فرصت اجرا روی برد آزمایشگاه پیدا نشد.

شمای مدارها در کوارتوس:

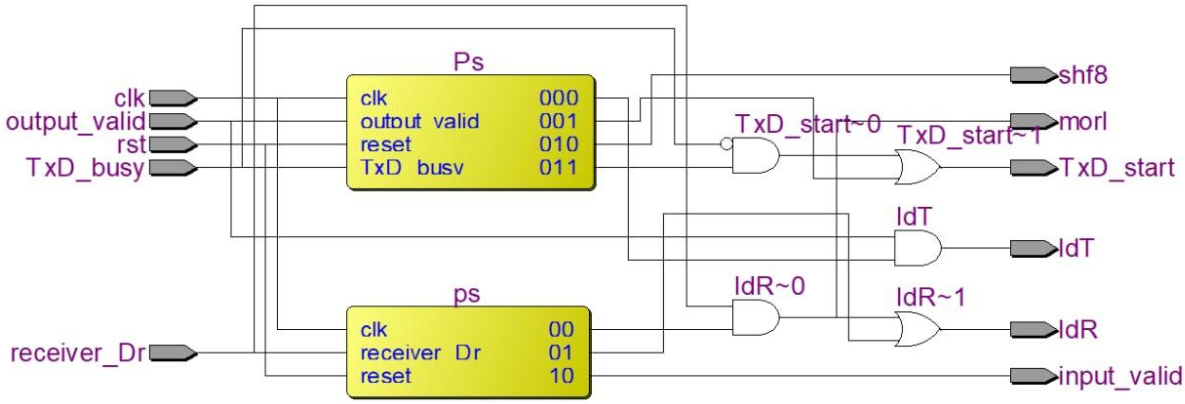
شمای کلی مدار



مسیر داده

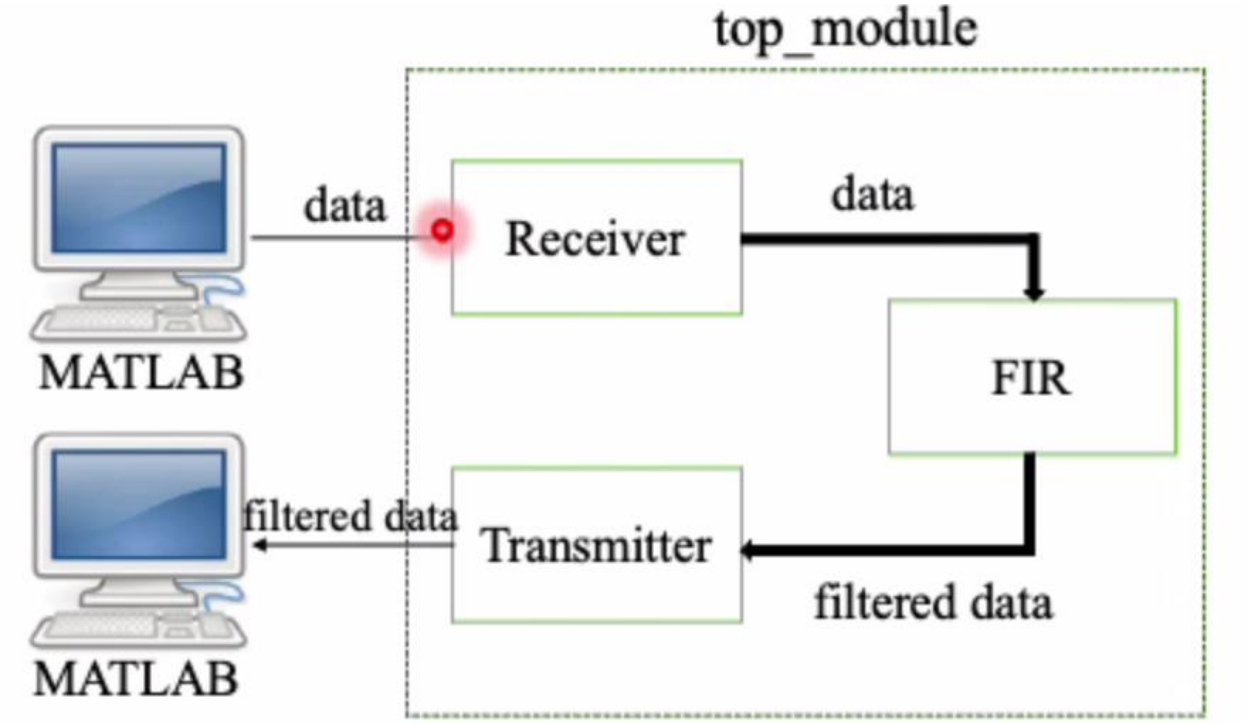


واحد کنترل



توضیحات کد MATLAB:

کد متلب اول پورت هارا مشخص میکند و سپسفایل صوت input.wav را می خواند و ذخیره می کند و به صورت double به کد اصلی ما می فرستد و پیامی مبنی بر این که همه اطلاعات را به درستی ارسال کرده است چاپ می کند. سپس آن را به صورت double دریافت می کند و پس از تعیین علامت نسبت به اندازه آن عملیاتی انجام می دهد و سپس در فایل Filtered_signal.wav به صورت صوت ذخیره می کند و در نهایت چاپ میکند که عملیات را درست انجام داده است.



بخش آخر(نکات پیشرفته):

بعد از گام اول فرکانس کاری مدار در slow model (بعد از تعریف کردن فرکانس از طریق timing analyzer wizard)

Slow Model Fmax Summary

	Fmax	Restricted Fmax	Clock Name	Note
1	128.75 MHz	128.75 MHz	CLOCK_50	

پ.ن: قبل از اضافه کردن گام اول فرکانس 133.9MHz نشان داده می شد.

پس از گام دوم و اضافه کردن pll (از طریق MegaWizard Plug-In Manager) به top module (که اسمش uart است) و وصل کردن کلاک تولیدی pll به واحد کنترل و مسیر داده، و انجام تغییراتی در setting جهت بهبود سرعت مدار(مانند تغییر برخی تنظیمات fitter و تغییر اولویت نحوه ی سنتز از حالت بالانس به سرعت)، به نتایج زیر دست یافتیم:

Flow Summary

Flow Status	Successful - Wed Nov 17 21:35:25 2021
Quartus II 64-Bit Version	13.0.1 Build 232 06/12/2013 SP 1 SJ Web Edition
Revision Name	uart
Top-level Entity Name	uart
Family	Cyclone II
Device	EP2C35F672C6
Timing Models	Final
Total logic elements	1,379 / 33,216 (4 %)
Total combinational functions	807 / 33,216 (2 %)
Dedicated logic registers	1,192 / 33,216 (4 %)
Total registers	1192
Total pins	5 / 475 (1 %)
Total virtual pins	0
Total memory bits	1,024 / 483,840 (< 1 %)
Embedded Multiplier 9-bit elements	2 / 70 (3 %)
Total PLLs	1 / 4 (25 %)

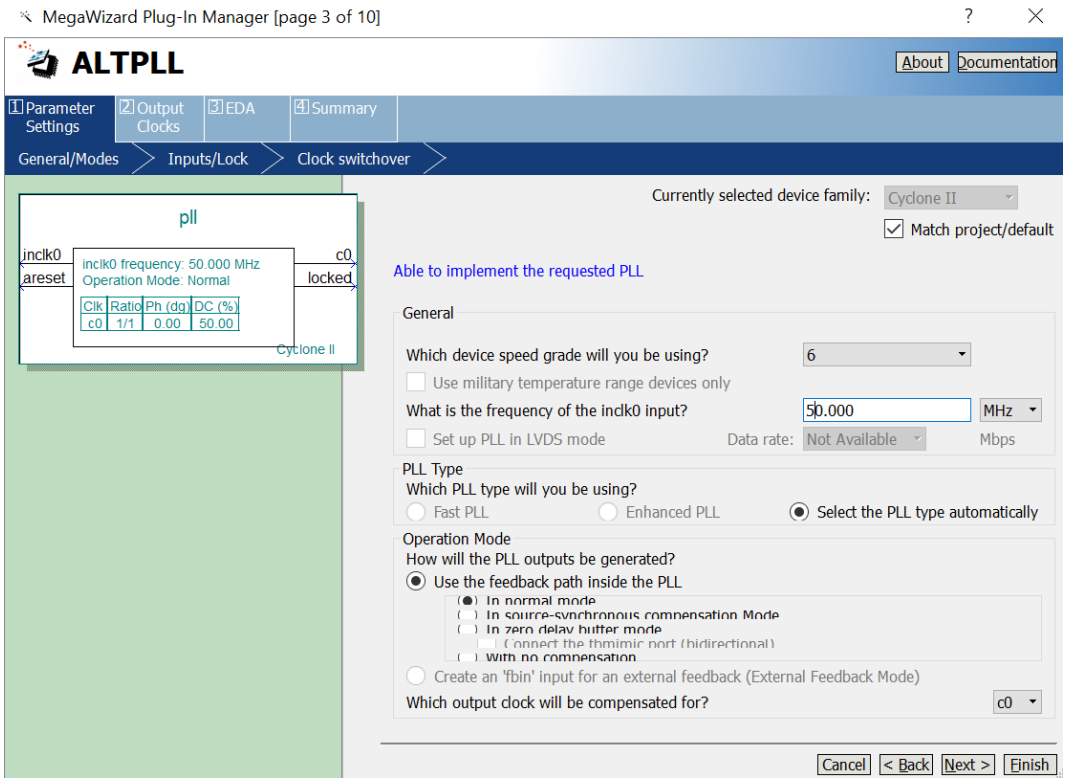
Slow Model Fmax Summary

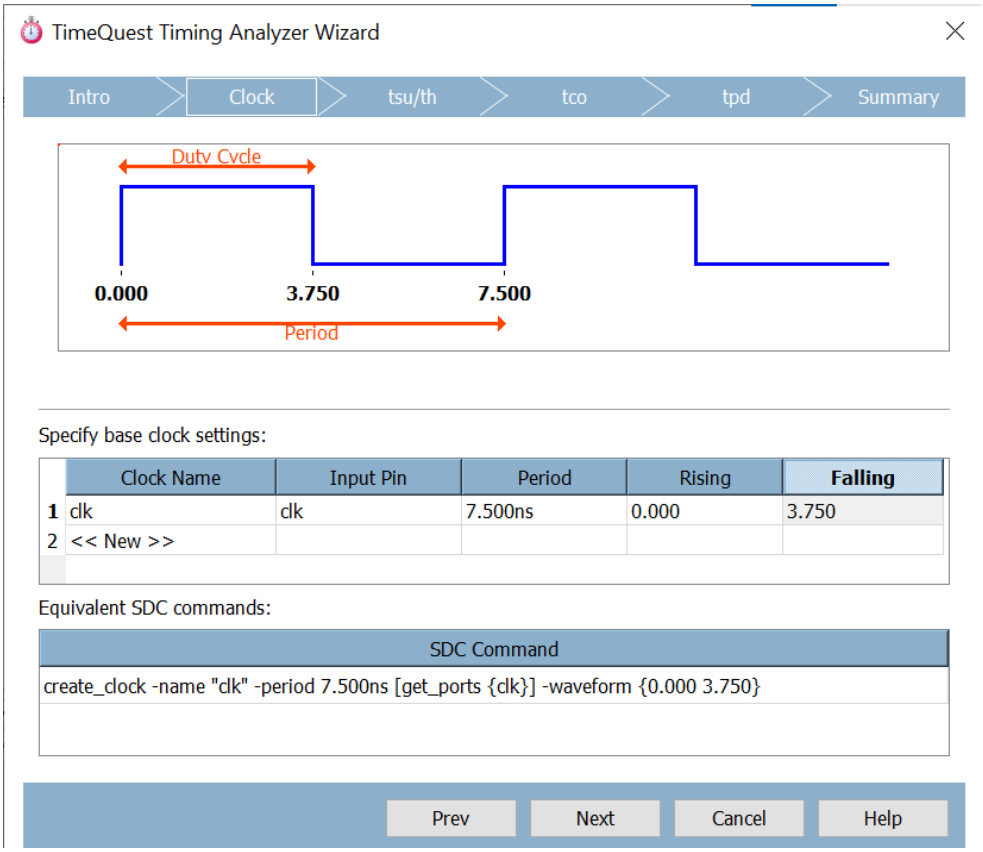
	Fmax	Restricted Fmax	Clock Name	Note
1	133.99 MHz	133.99 MHz	pl|altpll_component|pll|clk[0]	

Clocks

	Clock Name	Type	Period	Frequency	Rise	Fall	Duty Cycle	Divide by	Multiply by
1	CLOCK_50	Base	20.000	50.0 MHz	0.000	10.000			
2	pl|altpll_component|pll|clk[0]	Generated	7.500	133.33 MHz	0.000	3.750	50.00	3	8

در ادامه برخی از مراحلی که برای انجام این کار طی شده است را مشاهده میکنید:


MegaWizard Plug-In Manager [page 3 of 10]
ALTPLL
About
Documentation
1 Parameter Settings
2 Output Clocks
3 EDA
4 Summary
General/Modes
Inputs/Lock
Clock switchover
pll
inclk0
areset
c0
locked
inclk0 frequency: 50.000 MHz
Operation Mode: Normal
Clk Ratio Ph (dg) DC (%)
c0 1/1 0.00 50.00
Cyclone II
Currently selected device family: Cyclone II
Match project/default
Able to implement the requested PLL
General
Which device speed grade will you be using? 6
Use military temperature range devices only
What is the frequency of the inclk0 input? 50.000 MHz
Set up PLL in LVDS mode
Data rate: Not Available Mbps
PLL Type
Which PLL type will you be using?
Fast PLL
Enhanced PLL
Select the PLL type automatically
Operation Mode
How will the PLL outputs be generated?
Use the feedback path inside the PLL
In normal mode
In source-synchronous compensation Mode
In zero delay buffer mode
With no compensation
Create an 'fbin' input for an external feedback (External Feedback Mode)
Which output clock will be compensated for? c0
Cancel
< Back
Next >
Finish


TimeQuest Timing Analyzer Wizard
Intro
Clock
tsu/th
tco
tpd
Summary
Duty Cycle
0.000
3.750
7.500
Period
Specify base clock settings:
Clock Name
Input Pin
Period
Rising
Falling
1 clk
clk
7.500ns
0.000
3.750
2 << New >>
Equivalent SDC commands:
SDC Command
create_clock -name "clk" -period 7.500ns [get_ports {clk}] -waveform {0.000 3.750}
Prev
Next
Cancel
Help

در گام سوم، طبق روند پاسخ سوال ششم بخش "پاسخ به سوالات گزارشکار" که در ادامه مشاهده میکنید، شناسایی خودکار کلاک غیرفعال شده و کلاک تولیدی pll را به عنوان کلاک به برنامه معرفی کردیم(از طریق timing analyzer wizard). در این مرحله پیش از سنتز مجددا تغییرات جدید را در بخش Settings اعمال کردیم و تمام تنظیمات ممکن برای افزایش سرعت مدار را به کار بردیم، در تصویرهای زیر نتایج بدست آمده را مشاهده میکنید:

Flow Summary	
Flow Status	Successful - Wed Nov 17 22:15:53 2021
Quartus II 64-Bit Version	13.0.1 Build 232 06/12/2013 SP 1 SJ Web Edition
Revision Name	uart
Top-level Entity Name	uart
Family	Cyclone II
Device	EP2C35F672C6
Timing Models	Final
Total logic elements	1,379 / 33,216 (4 %)
Total combinational functions	807 / 33,216 (2 %)
Dedicated logic registers	1,192 / 33,216 (4 %)
Total registers	1192
Total pins	5 / 475 (1 %)
Total virtual pins	0
Total memory bits	1,024 / 483,840 (< 1 %)
Embedded Multiplier 9-bit elements	2 / 70 (3 %)
Total PLLs	1 / 4 (25 %)

Slow Model Fmax Summary

	Fmax	Restricted Fmax	Clock Name	Note
1	133.99 MHz	133.99 MHz	pl|altpll_component|pll|clk[0]	

پاسخ به سوالات گزارشکار:

1_چه محدودیت های زمانی دیگری را می توان توسط این Wizard اعمال کرد؟

بله، در سه مرحله ی دیگری که پس از مرحله ی تعریف کلاک می آیند میتوان محدودیت های زمانی دیگری را اعمال کرد که شامل موارد مقابل هستند: tsu یعنی setup time کلاک، th یعنی hold time کلاک، tco به معنای تاخیر کلاک تا خروجی و minimum tco حداقل میزان تاخیر کلاک تا خروجی، tpd به معنای propagation delay از پورتی که به عنوان ورودی تعریف میکنیم تا پورتی که به عنوان خروجی تعریف میکنیم و Minimum tpd به معنای حداقل زمان propagation delay بین دو پورت مذکور است.

2_آیا نیازی به اضافه کردن فرکانس جدید مدار (فرکانس خروجی) PLL به constraint فایل می باشد یا خیر؟(چرا؟)

بله زیرا میخواهیم با فرکانس متفاوتی مدارمان کار کند (در مسئله ی ما، چون مدارمان توانایی کار کردن با فرکانس های بالاتر از 50 مگاهرتز را دارد، میخواهیم از حداکثر سرعت آن با استفاده از تعریف فرکانس ورودی جدید استفاده کنیم.)

3_ **Routing** کلاک معمولا با همه سیگنال ها متفاوت است چرا؟

چون کلاک باید با کمترین میزان اختلاف زمانی بین کامپوننت های مختلف توزیع شود به گونه ای که همه ی کامپوننت ها لبه ی بالا (به طور مشابه پایین) کلاک را همزمان با هم مشاهده کنند. لذا مسیر جداگانه ای برای کلاک طراحی میشود و معمولا به مسیر های عادی کلاک را وصل نمیکنند، مگر آنکه به چندین کلاک نیاز داشته باشیم و تعداد مسیرهای تعبیه شده برای کلاک کم باشد.

4_ ابزار **Quartus II** چگونه کلاک را شناسایی می کند و آن را روی مسیر های از پیش مشخص **FPGA** سنتز میکند؟

با بررسی کد و دیدن ورودی **always** بلاک ها که به صورت **posedge**(یا **negedge**) در لیست حساسیت قرار میگیرند، کلاک را شناسایی میکند. یعنی با شناسایی سیگنال هایی که برای فلیپ فلاپ ها به عنوان کلاک عمل میکنند.

5_اگر کلاک های شناسایی شده در مدار بیشتر از مسیرهای مشخص شده در برد **FPGA** باشد، با چه منطقی مناسب ترین سیگنال ها برای انتقال روی شبکه های کلاک انتخاب می شود؟

سیگنال هایی که برای تعداد بیشتری فلیپ فلاپ عملکرد کلاک را اجرا میکنند به عنوان سیگنال های مناسب برای انتقال روی شبکه های کلاک انتخاب میکند.

6_چگونه انتخاب مسیر خودکار **Quartus II** را خاموش کنیم؟

با طی کردن مسیر زیر:

Settings/Fitter Settings/More Settings

و **off** کردن **Auto Global Clock**

7_آیا کلاک ورودی **50Hz** باید روی مسیر مشخص کلاک **Rout** شود؟چرا؟

چون تنها به عنوان ورودی **pll** داریم از آن بهره میبریم، خیر، زیرا در مسئله ی ما کلاک فقط به یک ماژول وصل میشود و چون به ماژول دیگری وصل نیست، مسئله ی اختلاف زمانی بین دیدن لبه های کلاک در ماژول های مختلف مطرح نیست، پس لازم نیست از مسیر ویژه ی کلاک برای وصل کردن آن به **pll** استفاده کرد.

8_اگرکلاک را به **PLL5** وصل کنیم...

چون به طور مثال، در بوردی که ما داریم با آن کار میکنیم، همانطور که در **flow summary** ها هم دیده میشود، فقط امکان بهره مندی از حداکثر **4 pll** در مسیرهای ویژه ی کلاک است، و اگر بخواهیم در مدارمان از **5 pll** استفاده کنیم، باید سیگنال تولیدی **pll** پنجم را از مسیری غیر از مسیرهای ویژه ی کلاک در سطح مدار توزیع کنیم.