
دانشکده مهندسی برق

# مدارهای منطقی و سیستم‌های دیجیتال

## آزمایش 9 - کنترل سخت‌افزار (FPGA) از طرف پردازنده (PS) به وسیله پورت GP و AXI GPIO

# آزمایش ۹:
# کنترل سخت افزار (FPGA) از طریق پردازنده (PS) به وسیله پورت GP و AXI GPIO

هدف از این آزمایش آشنایی اولیه با قسمت PS تراشه و برقراری ارتباط میان PL و PS با کمک AXI GPIO ipcore است.

## قطعات و تجهیزات مورد نیاز

- بورد Zynq (axpz 7010)
- یک عدد کابل USB و سیم پاور بورد
- نرم افزار Vivado 2019.1

## ۱ - پیش مطالعه

در آزمایش های قبلی از زبان توصیف سخت افزار (HDL) برای پیاده سازی مدار دیجیتال توصیف شده بر روی FPGA استفاده کردیم.

FPGA ها برای ایجاد یک سخت افزار خاص موردنظر ما استفاده می شوند.تقریبا هیچ سخت افزار ثابتی برروی تراشه وجود ندارد و برای ایجاد یک کاربرد خاص باید سخت افزار مربوط به آن را ایجاد کرد. از آنجایی که سخت افزار مورد نظر برای عملکرد خاصی طراحی شده است از نظر سرعت و عملکرد بسیار خوب عمل می کند اما ممکن است ازنظر مصرف توان بهینه نباشد.
از سوی دیگر میکروکنترلرها مجموعه‌ای از مدارهای ثابت هستند که هر کدام برای انجام برخی وظایف اساسی و پرکاربرد طراحی شده‌اند.با برنامه ریزی میکروکنترلرها شما تراشه چندمنظوره را برای انجام عملیات خود پیکربندی میکنید.
پردازنده های چندمنظوره (مثل میکروکنترلرها) انعطاف پذیر هستند در حالی که پردازنده های تک منظوره (مثل FPGA) سریع تر هستند.بیشتر کاربردها نیازمند پردازنده های چندمنظوره در کنار پردازنده های تک منظوره هستند.برای پاسخگویی به این نیاز در ماه March سال ۲۰۱۱ شرکت xilinx خانواده zynq-7000 را معرفی کرد. در این خانواده از یک پردازنده ARM (بخش PS – Processing System) در کنار FPGA (بخش PL – Programable Logic) استفاده می شود.
در این آزمایش با بخش PS برد zynq بیشتر آشنا خواهید شد و همچنین برقراری ارتباط بین بخش PS و PL به وسیله GPIO (با استفاده از پورت $M\_AXI\_GP$ زینک و ipcore ، AXI GPIO ) را خواهید آموخت.

# ۲ – پیش گزارش

قبل از انجام آزمایش دستور کار را دقیق مطالعه کنید. همچنین نیاز است ویدیوهای 8 ، 10.a و 10.b از سری ویدیوهای آموزشی zynq را ببینید.

# ۳ – دستورکار

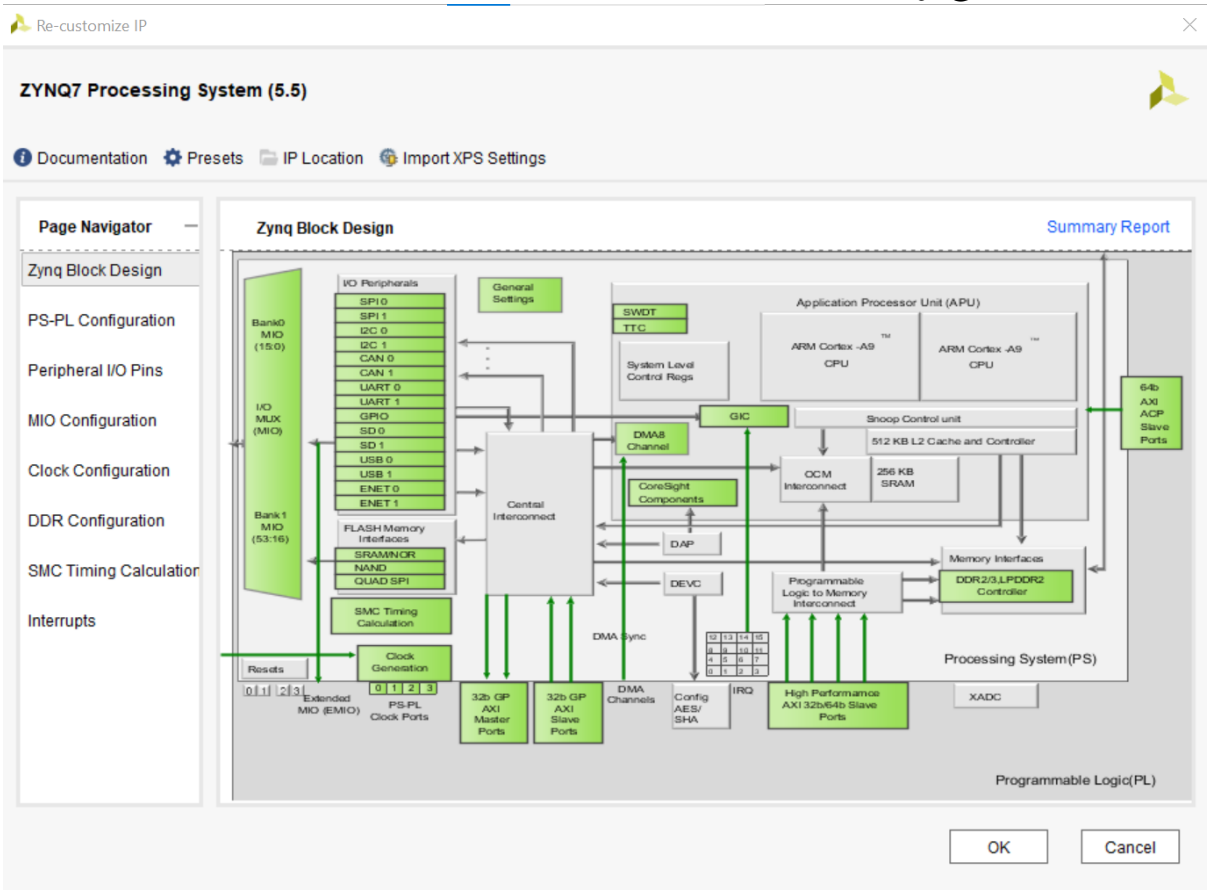
## ۱.۳ – راه اندازی قسمت PS

در این بخش با تنظیمات اولیه PS آشنا خواهیم شد و با استفاده از پریفرال uart طرف PS پیامی را برروی صفحه لپ تاپ نمایش خواهیم داد.

دقت کنید که در اسم پروژه خود و همچنین محلی که آن را داخل کامپیوتر ذخیره کرده اید هیچ حرف فارسی، space یا هرگونه کاراکتر نامعمول وجود نداشته باشد. در غیر این صورت در اجرای نرم افزار SDK به مشکل خواهید خورد.

پس از بازکردن نرم افزار vivado و ایجاد یک پروژه جدید:

- یک block design ایجاد کنید
- ipcore ، zynq را اضافه کنید
- برروی ipcore دو بار کلیک کنید تا به صفحه تنظیمات آن بروید. در این قسمت می توانید ساختار داخلی تراشه را مشاهده کنید.

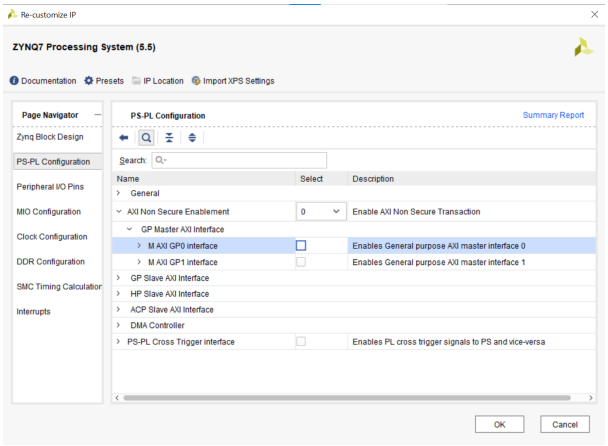



**شکل ۱: تنظیمات zynq**

- برخی از پورت ها به صورت پیش فرض در تراشه فعال شده اند اما در این بخش ما به آنها نیازی نداریم بنابراین بهتراست آنها را غیر فعال کنیم:

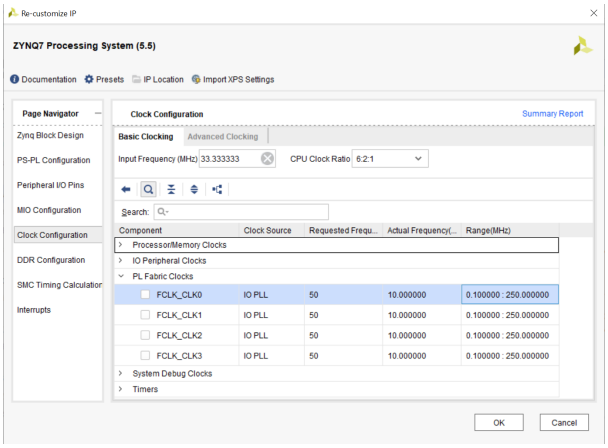
- از منوی سمت چپ PS-PL configuration را انتخاب کنید سپس داخل قسمت General و در قسمت Enable Clock Resets ، FCLK RESET0 N (سیگنال reset که PS برای PL می سازد)را غیرفعال کنید.
- در همان منوی PS-PL configuration ، AXI Non Secure Enablement را انتخاب و در زیر آن داخل قسمت GP Master AXI Interface ، M AXI GP0 Interface را غیرفعال کنید. از این پورت در بخش های بعدی آزمایش برای برقراری ارتباط میان PS و PL استفاده خواهیم کرد.

.




**شکل ۲: تنظیمات zynq**

- حال از منوی کنار Clock Configuration را انتخاب کرده و درزیر قسمت PL Fabric Clocks ، FCLK CLK0 (کلاکی که PS برای PL می سازد)را غیرفعال کنید.

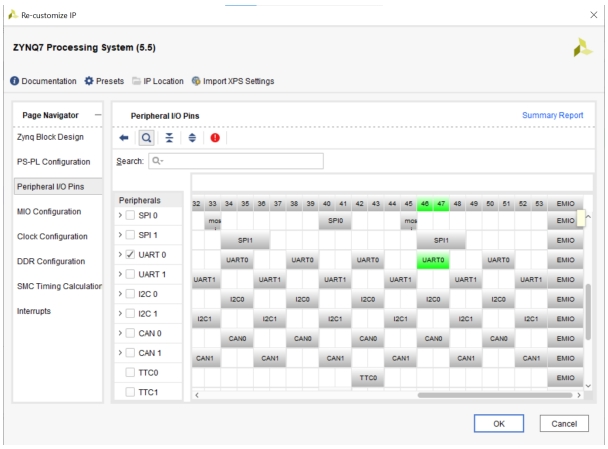



**شکل ۳: تنظیمات zynq**

- در کنار تراشه zynq یک حافظه DDR3 هم وجود دارد که وقتی با داده هایی با حجم نسبتا زیاد کار می کنیم یا به کارایی(performance) بالایی نیاز داریم از آن استفاده می کنیم. در این آزمایش به DDR3 نیازی نداریم پس بهتر است پورت آن را غیرفعال کنیم. در منوی کنار از بخش DDR Configuration ، DDR3 را غیرفعال کنید.

- همانظور که در آزمایش های قبل مشاهده کردید برای پروگرم کردن FPGA از پورت Jtag استفاده کردیم. برای اجرا کردن برنامه روی پردازنده ARM نیز از پورت Jtag استفاده می کنیم.

برای ارسال و دریافت پیام بین بورد و کامپیوتر به منظور عیب یابی (debug) از پروتوکل uart و پریفرال uart0 که از طرف PS به پورت micro USB بورد که مخصوص استفاده از uart است متصل شده استفاده می کنیم. برای فعال سازی پورت uart در منوی کناری بخش Peripherals IO Pins را انتخاب کنید و از بین Peripheral های موجود UART0 را انتخاب کنید. دقت کنید که پورت انتخاب شده همانند شکل زیر حتما روی پین های ۴۶ و ۴۷ قرار گرفته باشد. درغیر این صورت نیاز است که آن را به صورت دستی پورتUART0 قرار گرفته برروی پین های ۴۶ و ۴۷ را انتخاب کنید. به دلیل اینکه دو سیگنال rx و tx از UART0 به پین های ۴۶ و ۴۷ از از ۵۴ پین طرف PS متصل شده اند.




**شکل ۴: فعال کردن uart**

برای آشنایی بیشتر با پروتکل uart و نحوه کارکرد آن می توانید به این مطلب مراجعه کنید.

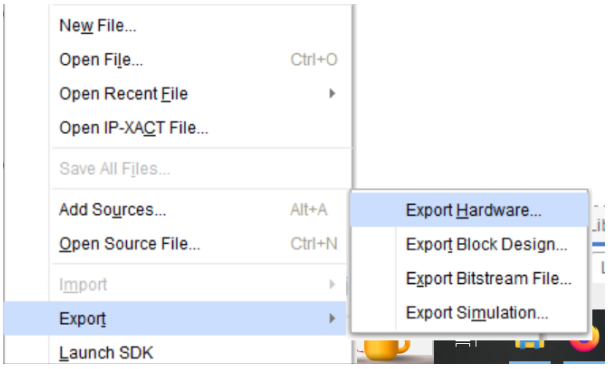
- پس از پایان تنظیمات ipcore برروی Run Block Automation کلیک کنید.

- برای پروگرم کردن FPGA نیاز به برنامه ای به زبان توصیف سخت افزار (HDL) داریم. درحالی که تا به این جای کار فقط یک block design درست کرده ایم و هنوز کد HDL تولید نشده است. برای این کار در منوی sources بر روی design کلیک راست کرده و Create HDL Wrapper را انتخاب کنید.با انتخاب Create HDL Wrapper ، نرم افزار Vivado به صورت خودکار از روی دیاگرام کد وریلاگ متناظر آن را درست می کند.
اگر روی Wrapper ایجاد شده کلیک کنید می توانید کد وریلاگ تولید شده توسط نرم افزار را مشاهده کنید.این کد وریلاگ صرفا شامل کانفیگ ها و تنظیمات لازم برای کارکرد درست پردازنده است و قرار نیست logic خاصی از FPGA را درگیر خودش کند زیرا پردازشی برروی FPGA طراحی نشده است.

- حال generate bitstream را انتخاب کنید.

- برای استفاده از پردازنده ARM موجود در تراشه و برنامه نویسی آن به زبان C از نرم افزار SDK استفاده می کنیم. برای استفاده از نرم افزار SDK قدم های زیر را طی کنید:

  - در منوی بالای صفحه در قسمت File ، Export وسپس Export Hardware را انتخاب کنید. با این کار اطلاعاتی از سخت افزار ایجاد شده در Vivado تهیه شده و در اختیار نرم افزار SDK قرار داده می شود.



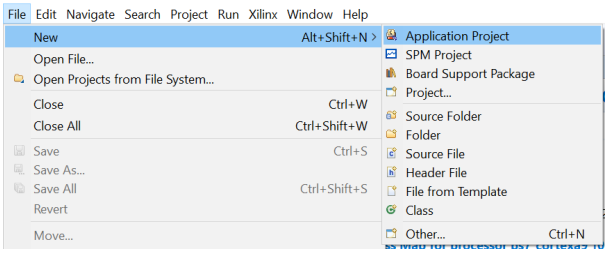

**شکل ۵: Export Hardware**

فراموش نکنید که تیک include bitstream را انتخاب کنید؛ تا بتوانیم تراشه را توسط نرم افزار SDK پروگرم کنیم.

  - اکنون از همان منوی File ، Launch SDK را انتخاب کنید.

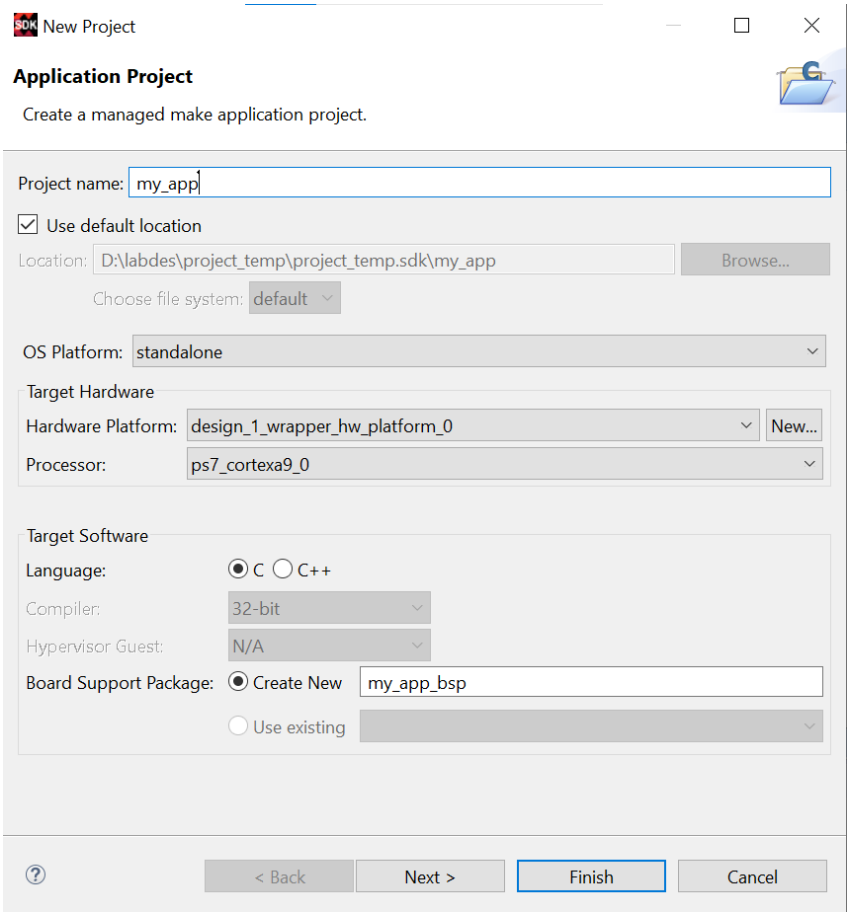
- بعد از باز شدن نرم افزار SDK این مراحل را انجام دهید:

– در ابتدا نیاز است همانند شکل زیر یک Application Project ایجاد کنید.



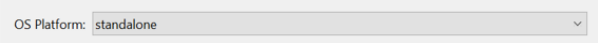

**شکل ۶: ساخت Application Project**

– یک اسم برای Application Project خود انتخاب کنید. دقت کنید که OS Platform حتما روی Standalone قرار داشته باشد. سپس روی next کلیک کنید.




**شکل ۷: تنظیمات Application Project**

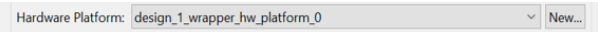
- برروی پردازنده zynq می توان سیستم عامل لینوکس را اجرا کرد و یکی از قابلیت های پردازنده توانایی کار با سیستم عامل لینوکس است. اما در این آزمایش ما با لینوکس کار نخواهیم کرد و هدف ما اجرای برنامه به زبان C بر روی پردازنده به صورت Bare-metal است بنابراین در این قسمت OS platform را Standalone انتخاب می کنیم.




**شکل ۸: OS Platform**

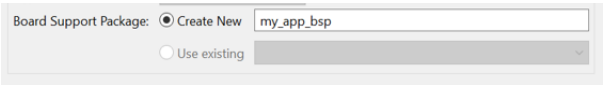
- در قسمت Hardware Platform سخت افزار توصیف شده توسط Vivado که توسط Export Hardware در اختیار نرم افزار SDK قرار داده شده بود ؛ import شده است و اطلاعات آن رامی توان از همین پنجره مشاهده کرد.

.



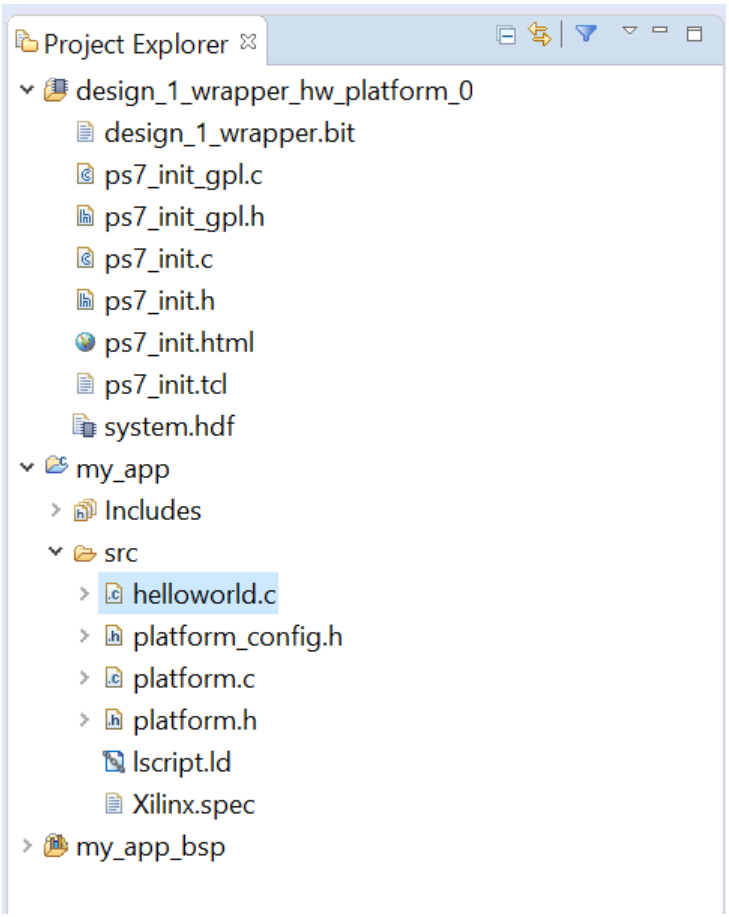

**شکل ۹: Hardware Platform**

- Board Support Package دربردارنده کتابخانه هایی است که در کنار پروژه قرار می گیرند و می توان از آنها استفاده کرد. در این جا ما به همراه Application Proect یک Board Support Package ساخته ایم . اما در برخی مواقع برای اینکه Board Support Package امکانات بیشتری داشته باشد می توانیم آن را به صورت جداگانه ایجاد کنیم.



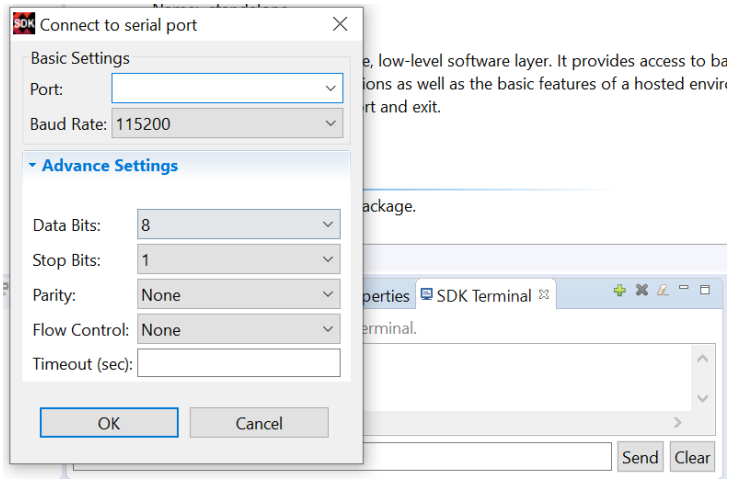

**شکل ۱۰: Board Support Package**

- در این قسمت از آزمایش می خواهیم از پروژه تمپلیت Hello World استفاده کنیم. آن را انتخاب کرده و Finish را بزنید.
- برای دیدن فایل اجرایی پروژه Hello world در منوی سمت چپ در بخش src آن را انتخاب کنید.



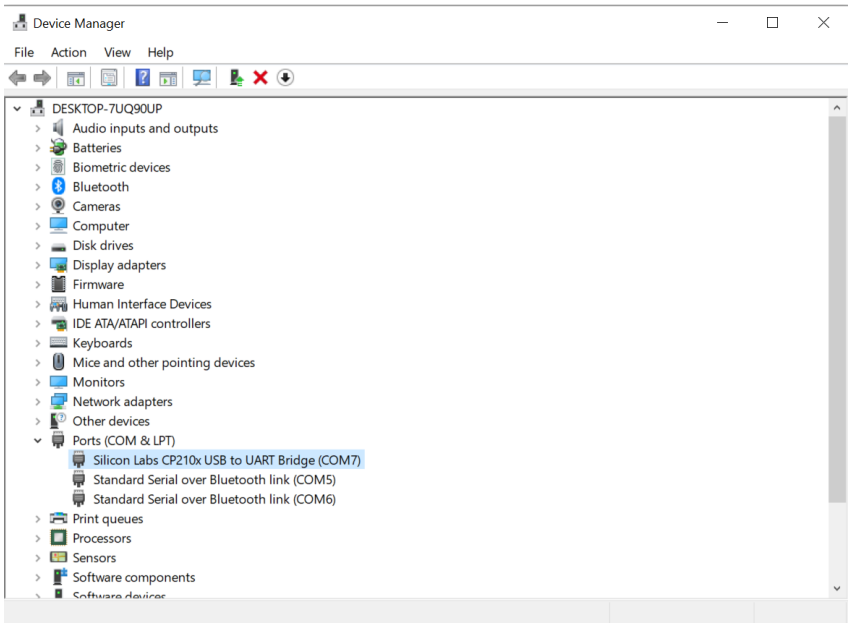

**شکل ۱۱: کد Hello World**

- اکنون می خواهیم به جای برنامه Hello World برنامه دیگری بنویسیم. برنامه شمارنده ای را بنویسید که با شروع از عدد ۱ شمارش را آغاز کرده و هر ۱ ثانیه مقدارش یک واحد افزایش می یابد.
- برای تعیین پورت uart در قسمت SDK terminal پورت uart را مشخص کنید. دقت کنید که مقدار Baud Rate برابر با ۱۱۵۲۰۰ قرار داده شده باشد. این مقدار در تنظیمات uart در تراشه zynq تنظیم شده است.




**شکل ۱۲: ترمینال SDK**

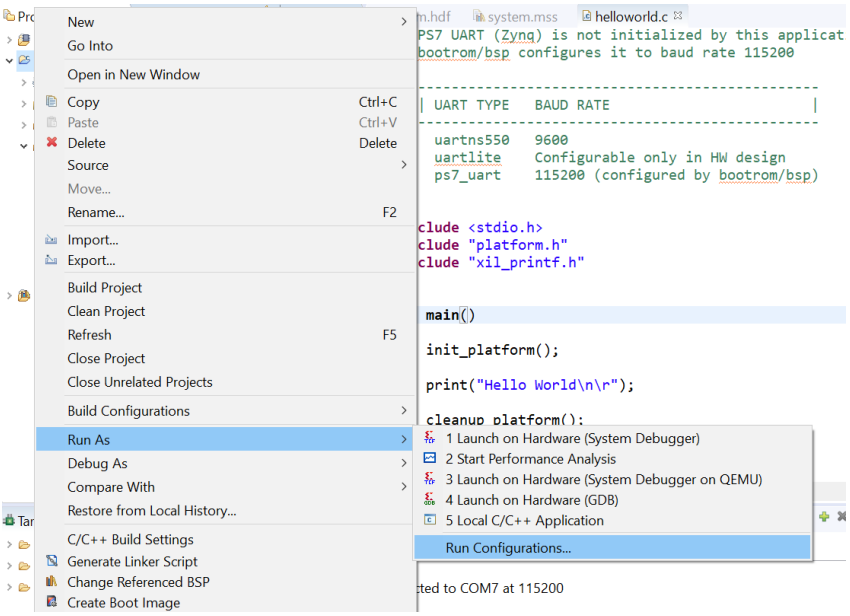
برای فهمیدن اینکه uart به کدام یک از پورت های کامپیوتر شما وصل است در device manger قسمت ports را نگاه کنید.




**شکل ۱۳: محل اتصال uart**

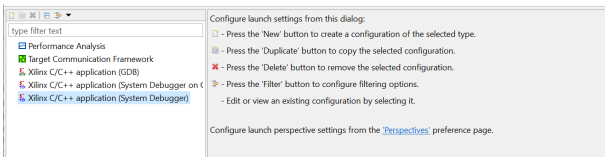
- در نهایت برای پروگرم کردن بورد گام های زیر را طی کنید:

* در منوی سمت چپ برروی application project کلیک راست کرده و از قسمت Run as ، Run Configuration را انتخاب کنید.

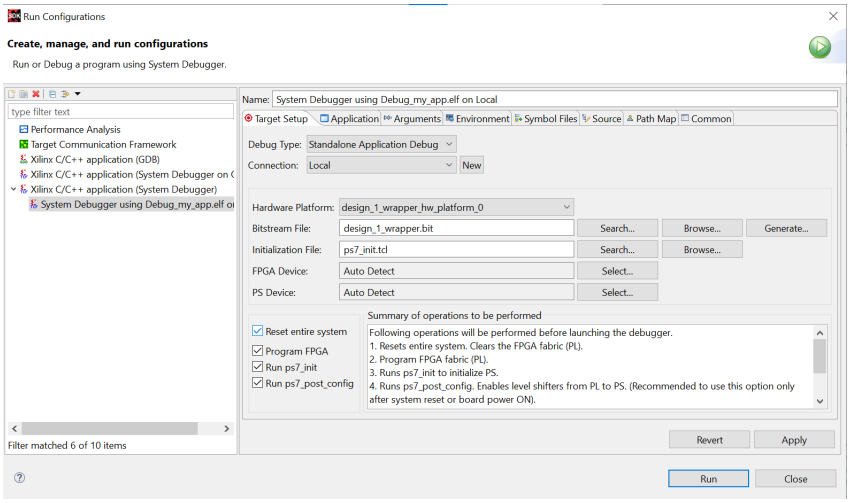



**شکل ۱۴: Run as Configuration**

* یک System Debugger ایجاد کنید.




**شکل ۱۵: ایجاد System Debugger**




**شکل ۱۶: تنظیمات System Debugger**

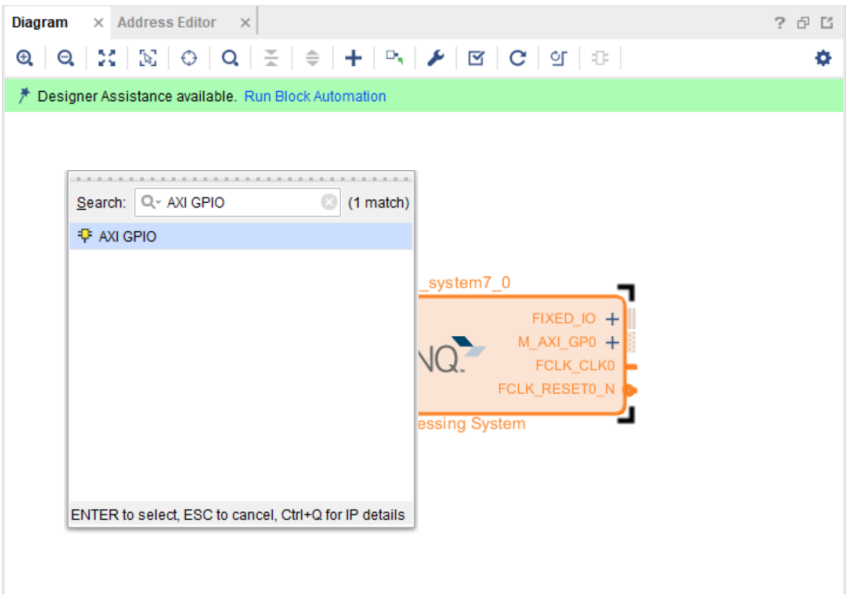
* Run را بزنید و بورد را پروگرم کنید

دقت کنید که این برنامه برروی پردازنده کامپیوتر شما اجرا نمی شود بلکه برروی پردازنده بورد اجرا می شود.

## ۲.۳ - برقراری ارتباط بین PS ، PL

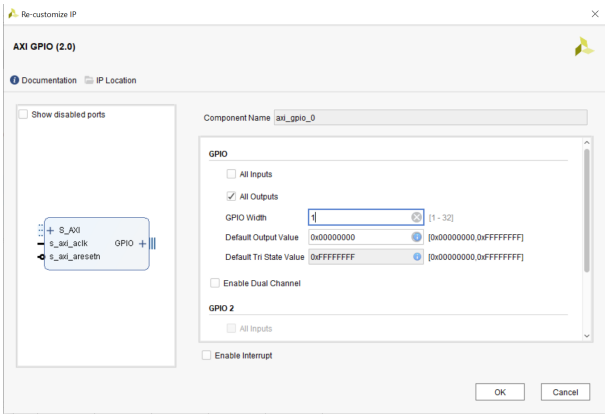
در این بخش از آزمایش با استفاده از ipcore ، AXI GPIO ارتباط ساده ای بین PL و PS برقرار می کنیم. هدف از این بخش آزمایش این است که از طریق PS یکی از LED های بخش PL را به صورت چشمک زن روشن و خاموش کنیم. AXI GPIO به یکی از پورت های $M\_AXI\_GP$ متصل می شود و داده به این صورت بین PS و PL منتقل می شود. AXI GPIO سرعت زیادی ندارد و در مواقعی که داده مورد انتقال کوچک است از آن استفاده می شود.

- یک block design ایجاد کرده و ipcore ، zynq را در آن قرار دهید. در تنظیمات DDR3 را غیر فعال کنید.
- یک ipcore ، AXI GPIO را داخل design قرار دهید.




**شکل ۱۷: قرار دادن AXI GPIO**

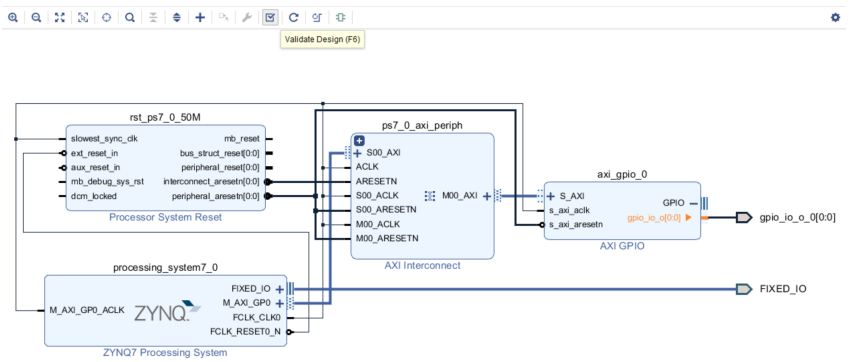
- تنظیمات ipcore ، AXI GPIO را همانند شکل زیر تعیین کنید.




**شکل ۱۸: تنظیمات AXI GPIO**

GPIO Width نشان دهنده اندازه داده های تبادل شده است. در این قسمت داده ها یک بیتی (۱ روشن و ۰ خاموش) هستند.همچنین چون از AXI CPIO به عنوان خروجی PS استفاده می شود (داده از PS به PL منتقل می شود) آن را روی حالت All Outputs قرار می دهیم.

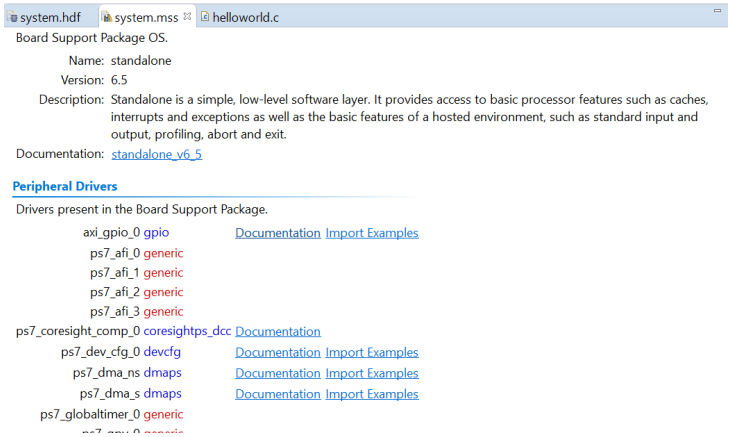
- Run Block Automation را بزنید.
- Run Connection Automation را بزنید. و تیک $S\_AXI$ را بزنید.
- روی پورت GPIO راست کلیک کرده و make external را بزنید.
- اکنون validate design را بزنید تا از صحت طراحی خود اطمینان پیدا کنید.




**شکل ۱۹: صحت سنجی دیاگرام نهایی**

- بعد از ساخت HDL Wrapper از منوی سمت چپ Open Elaborated Design را انتخاب کنید.
- در قسمت IO Ports با مراجعه به دیتاشیت ، پین خروجی GPIO را به یکی از LED هی بورد متصل کنید.IO Std را روی LVCMOS33 قرار دهید.
- اکنون نیاز است تا عملیات generate bitstream را انجام دهید. سپس نرم افزار SDK را باز کنید.
- در این بخش باید در قسمت PS با استفاده از GPIO با PL ارتباط برقرار کرده و یک LED را روشن و خاموش کنیم.برای این کار از توابع موجود در کتابخانه "$xgpio.h$" استفاده می کنیم. برای آشنایی بیشتر با توابع و نحوه عملکرد آنها می توانید از Documentation و example های موجود استفاده کنید.

.




**شکل ۲۰: کتابخانه های قابل استفاده**

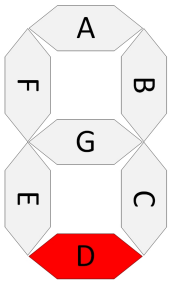
فایل "part2.c" به همراه دستور کار در اختیار شما قرار داده شده است. این برنامه با برقراری ارتباط میان PL و PS توسط Gpio یکی از LED های سمت PL را روشن می کند.این برنامه را برروی FPGA پروگرم کرده و عملکرد آن را مشاهده کنید.حال برنامه را به نحوی تغییر دهید که LED به صورت متناوب روشن و خاموش شود.

## ۳.۳ - طراحی ماشین حالت

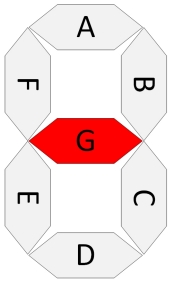
در این قسمت شما باید ماشین حالت شبیه ساز یک چراغ راهنمایی را طراحی کنید.
پروژه تمپلیت "part3" که همراه دستورکار آزمایش در اختیار شما قرار داده شده است را باز کنید. سپس Block design آن را مشاهده کنید.
در ماژول traffic light عملکرد یک چراغ راهنمایی و رانندگی پیاده می شود و باید توسط شما تکمیل شود. ساختار کلی پروژه را بررسی کرده و باتوجه به نکاتی که در بخش های قبلی آزمایش آموختید نحوه کار آن را برای دستیار آزمایشگاه توضیح دهید.
تنظیمات ipcore ، zynq را بررسی کرده و از صحت تنظیمات اطمینان حاصل کنید.

### ۱.۳.۳ - بخش PL

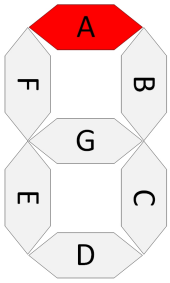
ماژول traffic light عملکرد یک چراغ راهنمایی و رانندگی را پیاده می کند. این ماژول با دریافت دستور از ورودی رنگ چراغ راهنمایی را تغییر می دهد. اگر مقدار این ورودی (ورودی command) برابر با صفر باشد چراغ سبز می شود و اگر مقدار ورودی برابر با یک شود ؛ چراغ ابتدا به مدت ۵ ثانیه زرد شده و سپس قرمز می شود. وضعیت چراغ در لحظه را توسط یکی از seven-seg های برد کمکی همانند شکل های زیر نمایش دهید.




**شکل ۲۱: چراغ سبز**




**شکل ۲۲: چراغ زرد**




**شکل ۲۳: چراغ قرمز**

همچنین یکی از LED های برد کمکی را به وضعیت عبور عابران پیاده اختصاص دهید. اگر چراغ قرمز باشد LED باید روشن شود که به معنای مجاز بودن عبور عابران پیاده است. در غیر این صورت LED باید خاموش شود. پس از تکمیل این ماژول نیاز است تا پورت های خروجی را به پین مربوطه خود در FPGA متصل کنید. پس از اتمام این کار bitstream را تولید کرده و نرم افزار SDK را با استفاده از گزینه launch SDK باز کنید.

### ۲.۳.۳ - بخش PS

بخش نرم افزاری و کد C آن به صورت آماده و تکمیل شده در پروژه تمپلیت موجود است. با استفاده از این کد و صادر کردن دستورات مناسب شما می توانید وضعیت چراغ راهنمایی را تغییر دهید. ساختار کد را به دقت بررسی کرده و نحوه عملکرد آن را به صورت خلاصه برای دستیاران آموزشی توضیح دهید.
برنامه را برروی FPGA پروگرم کرده و ماژول خود را با استفاده از دستورات زیر تست کنید.

- با استفاده از دستور "GREEN" چراغ را سبز کنید.

- با استفاده از دستور "RED" چراغ را قرمز کنید.

### ۳.۳.۳ - امتیازی

علاوه بر نمایش وضعیت چراغ بر روی seven-seg ، با استفاده از یک ipcore ، AXI GPIO وضعیت چراغ را به PS منتقل کرده و با کمک Uart آن را نمایش دهید.

# ۴ - گزارش

### ۱.۴

در خصوص آنچه از این آزمایش آموختید گزارش ۱ الی ۲ صفحه ای (مطابق تمپلیت) بنویسید.

### ۲.۴

دیدگاه خود نسبت به بخش های مختلف آزمایش را به انضمام پیشنهادهایتان بنویسید.