دانشگاه صنعتی امیرکبیر

دانشکده کامپیوتر

ضرب به روش Co-Design

درس سیستم‌های قابل بازپیکربندی

استاد:

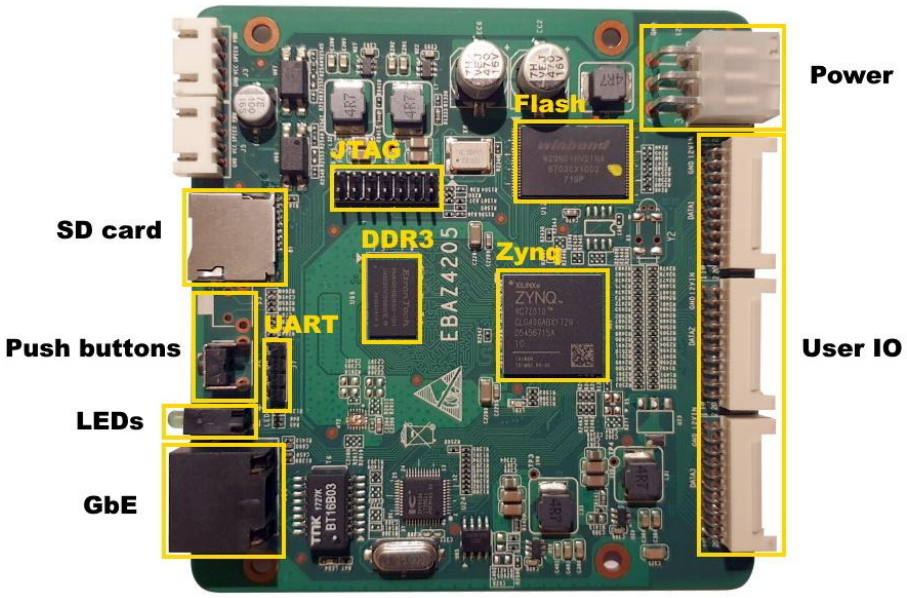
دکتر مرتضی صاحب الزمانی

مرحله‌ی اول پروژه: آشنایی اولیه با ابزار ویوادو

مرحله‌ی اول پروژه بیشتر برای آشنایی با طرز کار ابزار ویوادو است. در این مرحله، یک سیستم ساده روی ابزار ویوادو پیاده سازی می‌شود که دو عدد ۴ بیتی را به روش بوث در هم ضرب می‌کند. از آنجایی که عمل ضرب الگوریتمِ مشخص دارد، باید برای آن یک سخت‌افزار مشخص پیاده‌سازی شود (می‌توانید از کدهای متن باز موجود در سایت https://opencores.org یا https://github.com/freecores استفاده کنید). بنابراین، ورودی‌های ۴ بیتی باید به یک ضرب‌کننده سخت‌افزاری ارسال شوند و نتیجه باید به پردازنده بازگردانده شود. عملیاتی که به صورت نرم‌افزاری پیاده‌سازی می‌شود بسیار ساده است و شامل انجام عملیات لازم برای فراهم کردن داده و تبادل اطلاعات با سخت‌افزار است.

مراحل اجرایی پروژه :

برای انجام این بخش همانگونه که در صورت پروژه اشاره شده است بایستی از روش ایجاد یک مجموعه سخت‌افزاری، نرم‌افزاری همزمان بر روی FPGA استفاده نمود. برد مورد استفاده برای این منظور برد EBAZ4205 می‌باشد که دارای یک تراشه مرکزی Zynq می‌باشد.



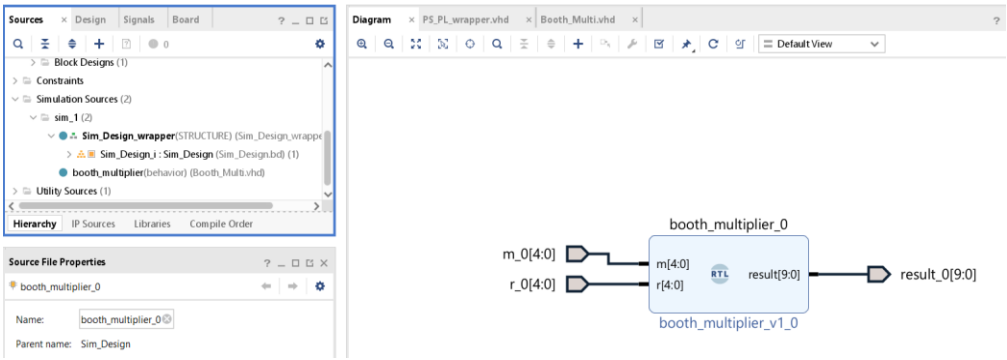
این برد در حقیقت یک کنترل‌کننده ماینر مدل Ebit E9+ BTC miner می‌باشد که به دلیل قدیمی شدن این ماینر به صورت جداگانه به فروش می‌رسد و با توجه به مشخصات فنی آن قیمت مناسبی دارد.
مشخصات اجمالی این برد در ادامه آورده شده است :

- XC7Z010CLG400, Dual Core Cortex A9 @ 666.66MHz and Artix-7 FPGA with 28k LEs
- 256MB DDR3 Dynamic Memory
- 128MB SLC NAND FLASH
- 10/100Mbps Ethernet transceiver: IP101GA
- On Board Switching Power Supply

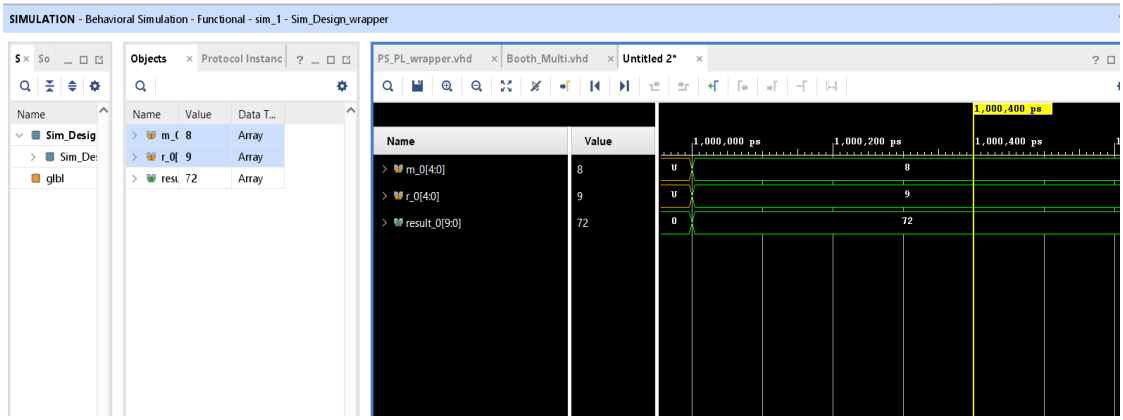
برای ایجاد پروژه از نسخه Vivado 2022.1 برای طراحی سخت افزار و Vitis 2022.1 برای ایجاد نرم‌افزار استفاده شده است.

برای بخش ضرب‌کننده از کدهای آماده استفاده شده است. این کد از سایت github[1] دانلود شده است. این کد طبق توضیحات آن یک ضرب کننده بوث برای اعداد علامت دار با فرمت متمم دو می‌باشد. تعداد بیت‌های ورودی این ضرب‌کننده در داخل کد قابل تنظیم می‌باشد. به دلیل اینکه دامنه اعداد ۴ بیتی به نسبت کوچک است ورودی‌ها را بدون علامت در نظر می‌گیریم اما در صورت نیاز به علامت‌دار بودن نیز مراحل کار چندان تفاوتی نخواهد داشت. با توجه به اینکه تعداد ورودی‌ها در کد قابل تنظیم است، تعداد ورودی را برابر با ۵ قرار می‌دهیم که یک بیت برای علامت بوده و به همین دلیل استفاده نمی‌گردد.

پس از تنظیم پارامترهای کد یک شبیه‌سازی از کد انجام می‌گردد و نتایج مورد بررسی قرار می‌گیرد. برای این منظور یک بلاک برای این منظور ایجاد می‌گردد و سپس Wrapper مربوطه آن توسط نرم‌افزار ایجاد می‌شود.



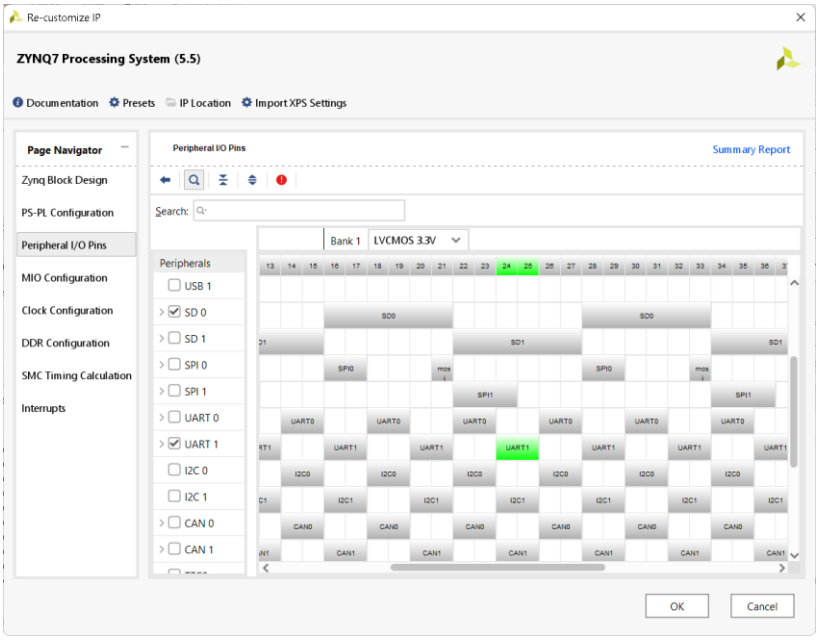
سپس با چند مقدار نتیجه بلاک مورد بررسی قرار می‌گیرد، به عنوان مثال نتیجه ضرب ۸ در ۹ در تصویر زیر نمایش داده شده است :



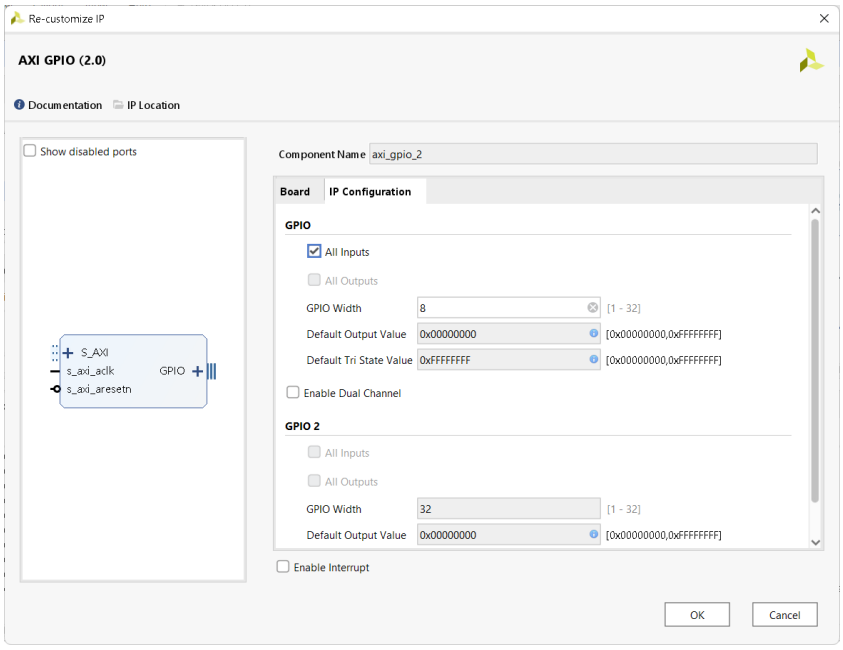
پس از بررسی عملکرد صحیح کد ضرب‌کننده قسمت مربوط به سخت‌افزار را ایجاد می‌نماییم. در این بخش با توجه به وجود هسته پردازشی سخت ARM ، از این پردازنده به عنوان قسمت پردازشی نرم‌افزار استفاده می‌گردد. در قسمت پیکربندی با توجه به سادگی برنامه مورد نظر نیازی به فلش، رم خارجی DDR و کارت حافظه نمی‌باشد و تنها نیاز به فعال نمودن USART1

[1] - https://github.com/gustavohb/booth-multiplier

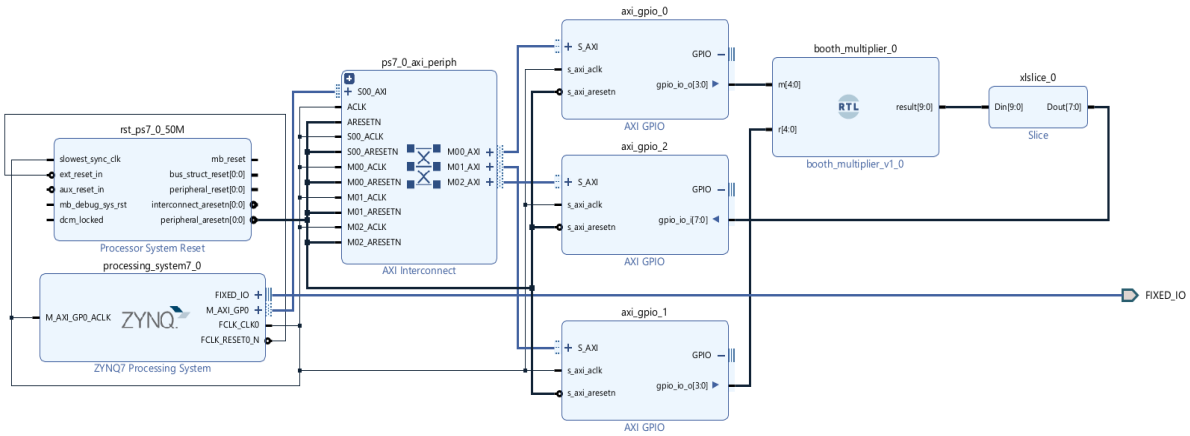
می‌باشد که با توجه به شماتیک برد به پایه‌های ۲۴ و ۲۵ بخش MIO متصل شده است. لذا این بخش در تنظیمات Zynq فعال می‌گردد.



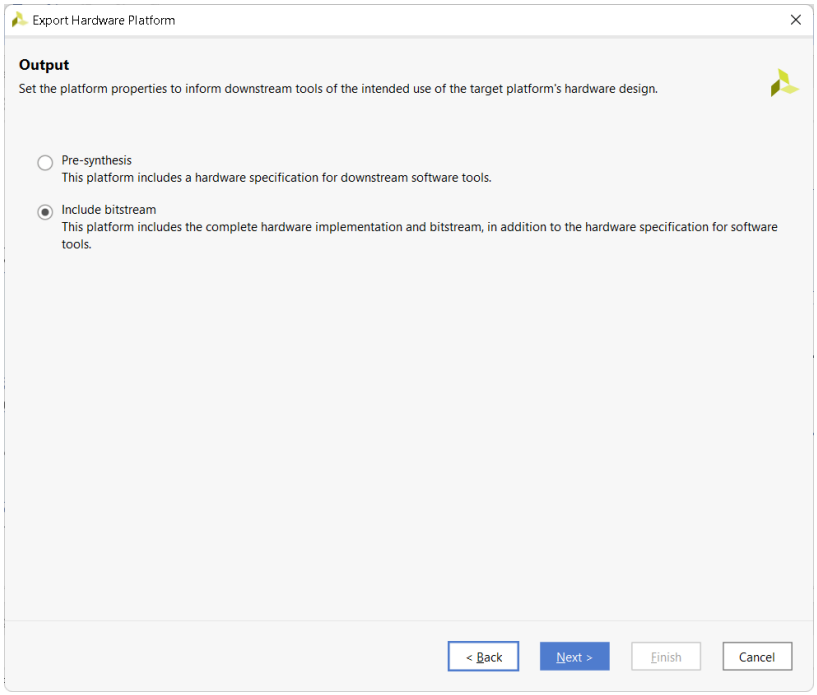
همچنین با توجه به نیاز به ارتباط با ماژول ضرب‌کننده این بخش به بلوک دیاگرام اضافه می‌شود و با استفاده از AXI GPIO ها ورودی و خروجی‌های مورد نیاز ماژول ضرب‌کننده ایجاد می‌گردد. پس از این مرحله با استفاده از ابزار Automation نرم‌افزار ارتباطات مورد نیاز به طور خودکار ایجاد می‌گردد. بایستی توجه داشت از آنجا که پورت‌های ماژول ضرب‌کننده به صورت ورودی و یا خروجی هستند، GPIO ها نیز در طراحی سخت‌افزار به صورت ورودی و خروجی یک‌طرفه تنظیم می‌گردند، این امر در بخش نرم‌افزار ارتباط ساده‌تری را ایجاد خواهد کرد. به عنوان مثال تنظیمات GPIO مربوط به خروجی ماژول ضرب‌کننده به صورت زیر می‌باشد :



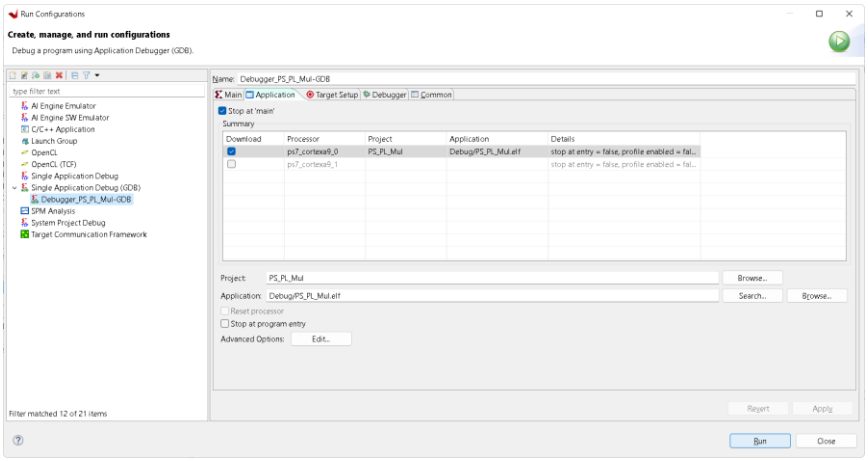
همچنین برای جلوگیری از ایجاد Warning در خصوص پهنای گذرگاه‌های ارتباطی نیز از بلوک Slice استفاده شده است. طراحی سخت‌افزار نهایی به صورت زیر می‌باشد :



پس از این مرحله با استفاده از Generate Bitstream فایل Bitstream ایجاد شده و با استفاده از Export Hardware و انتخاب گزینه Include Bitstream فایل XSA را برای استفاده در برنامه Vitis جهت استفاده بخش نرم‌افزار ایجاد می‌نماییم.



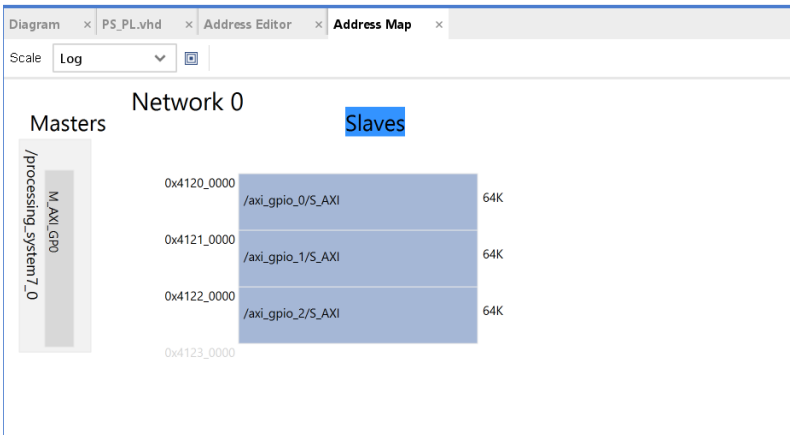
برای ایجاد بخش نرم‌افزار از بسته Vitis استفاده می‌گردد. پس از باز نمودن این نرم‌افزار با استفاده از بخش New Application Project و وارد نمودن فایل XSA مرحله قبلی یک پروژه نرم‌افزاری برای اجرا بر روی سخت‌افزار مرحله قبلی ایجاد می‌گردد. در انتهای ابزار کمک کننده گزینه برنامه Hello World نمونه جهت کمک به تست کلیه موارد تا کنون انتخاب می‌گردد. سپس کل پروژه Build می‌گردد و پس از اضافه کردن محیط اجرای مناسب در بخش Run و اتصال برد به سیستم با استفاده از Xilinx Platform Cable II برنامه به FPGA منتقل شده و بر روی آن اجرا می‌گردد. در زیر نحوه تنظیم اجرا آورده شده است :



در صورت اجرای صحیح متن Hello World در رابط ترمینال نمایش داده می‌شود. از برنامه Hterm به عنوان رابط ترمینال برای ارتباط با سخت‌افزار استفاده شده است. در صورت اجرای صحیح این برنامه تغییر یافته و برنامه نهایی ایجاد می‌گردد.
در ابتدا با استفاده از فایل "xparameters.h" آدرس‌های رابط‌های GPIO استخراج می‌گردد. این آدرس‌ها در تعاریف زیر قرار گرفته‌اند :

```
#define XPAR_AXI_GPIO_0_BASEADDR 0x41200000

#define XPAR_AXI_GPIO_1_BASEADDR 0x41210000

#define XPAR_AXI_GPIO_2_BASEADDR 0x41220000
```

این آدرس‌ها بایستی با موارد نمایش داده شده در نرم‌افزار Vivado یکسان باشند :

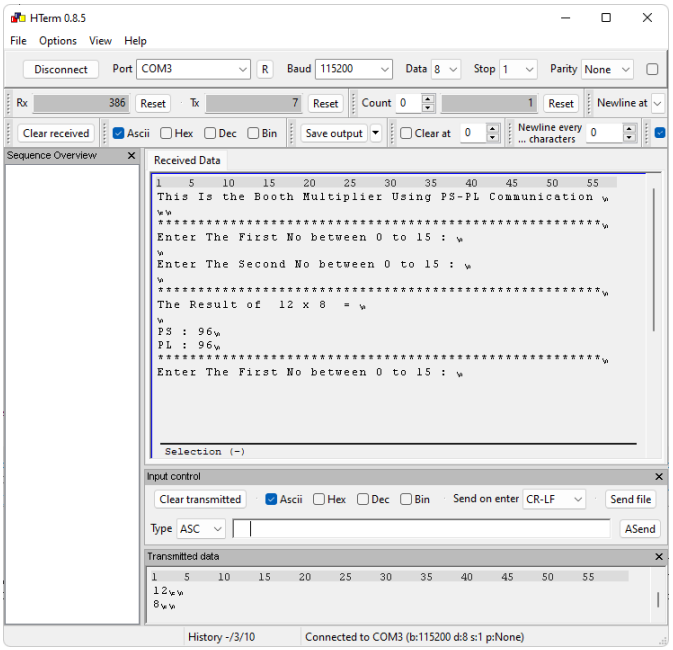


با تغییر برنامه Hello World برنامه ای به صورت زیر ایجاد می‌گردد که پس از گرفتن ورودی‌ها از کاربر و بررسی بودن آن‌ها در محدوده صحیح با استفاه از دستور Xil_Out8 مقادیر خوانده شده از ورودی به GPIO ها انتقال می‌یابد و با استفاده از Xil_In8 نتیجه حاصله از ضرب‌کننده سخت‌افزاری خوانده می‌شود

```c
#include <stdio.h>
#include "platform.h"
#include "xil_printf.h"
#include "xgpio.h"
#include "xparameters.h"


int main()
{
	long int delay;
	unsigned int a=0,b=0;
	unsigned int ps_mul=0;
	unsigned int pl_mul=0;
	init_platform();

    xil_printf("This Is the Booth Multiplier Using PS-PL Communication \n\r");

    while(1)
    {
       xil_printf("\n*****************************************************");
       do
       {
              xil_printf("\nEnter The First No between 0 to 15 : \n");
              scanf("%u",&a);
       }
       while(a<0 || a>16);

       do
       {
              xil_printf("\nEnter The Second No between 0 to 15 : \n");
              scanf("%u",&b);
       }
        while(b<0 || b>16);


        Xil_Out8( XPAR_AXI_GPIO_0_BASEADDR,(unsigned char)a);
        Xil_Out8( XPAR_AXI_GPIO_1_BASEADDR,(unsigned char)b);

       xil_printf("\n*****************************************************");

       xil_printf("\nThe Result of  %u x %u  = \n",a,b);
       xil_printf("\nPS : ");
       ps_mul=a*b;
       xil_printf("%u",ps_mul);
       xil_printf("\nPL : ");

       pl_mul= Xil_In8( XPAR_AXI_GPIO_2_BASEADDR);
       xil_printf("%u",pl_mul);
       for(delay=0;delay<1000000;delay++);
    }
    cleanup_platform();
    return 0;
}
```

ارتباط با GPIO ها با توابع Xilinx برای AXI GPIO نیز امکان پذیر است اما به دلیل تنظیم در بخش سخت‌افزار و سادگی روش ارتباط مستقیم طبق Datasheet مربوط به IP استفاده شده است.

در نهایت نتایج حاصل از ضرب در قسمت پروسسور و سخت‌افزار در دو متغیر ps_mul و pl_mul ذخیره می‌گردد و در خروجی برای مشاهده کاربر انعکاس می‌یابند. نتیجه یک اجرا در زیر آورده شده است. در این مثال دو عدد ۱۲ و ۸ به سیستم داده شده است.



در شکل زیر لوازم مورد استفاده برای ایجاد پروژه نمایش داده شده است: