

Electrical and Computer Engineering  
PO Box 23874 | Doha, Qatar  
341G Texas A&M Engineering Building | Education City  
(Office) +974.4423.0194 | (Fax) +974.4423.0064 | GMT+3

# Quick Start Guide

## ZedboardOLED Display Controller IP v1.0

### Introduction

This document provides instructions to quickly add, connect and use the ZedboardOLED v1.0 IP core. A test application running on an ARM processor system is used to communicate with the IP through its driver's functions. Vivado design suite is used as the development environment. Hardware verification is done on the Zedboard, however, this IP can be easily tailored to target other boards or embedded systems.

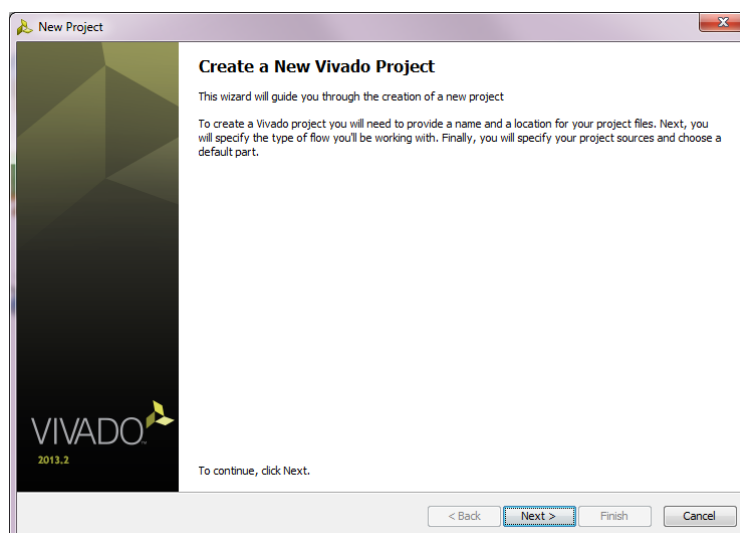
### Requirements

- **Hardware:** Zedboard.
- **Software:** Vivado 2014.2 or above.

### Procedures

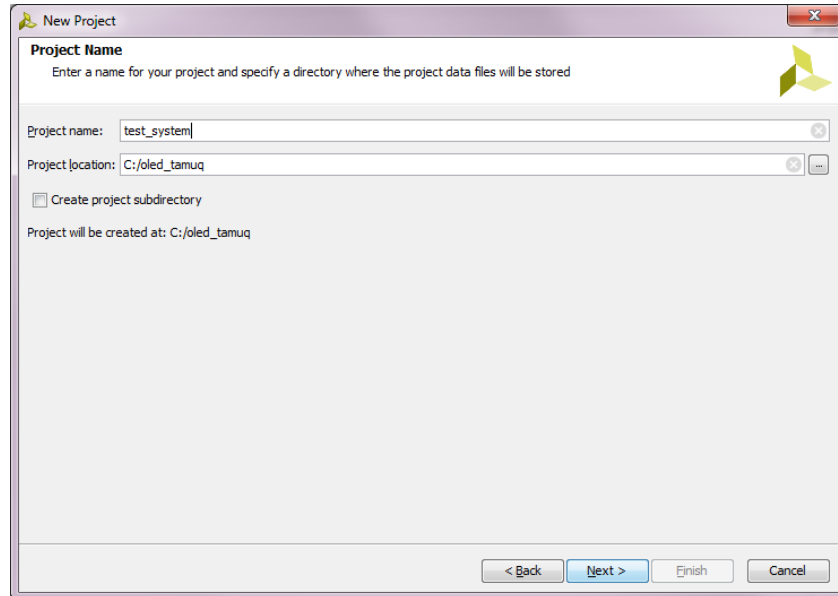
#### A- Creating a project in Vivado to target the Zedboard

1. Launch the **Vivado Design Suite**
2. Click on **File-> Create project**, and select **Create New Project** to open the **New Project** wizard.

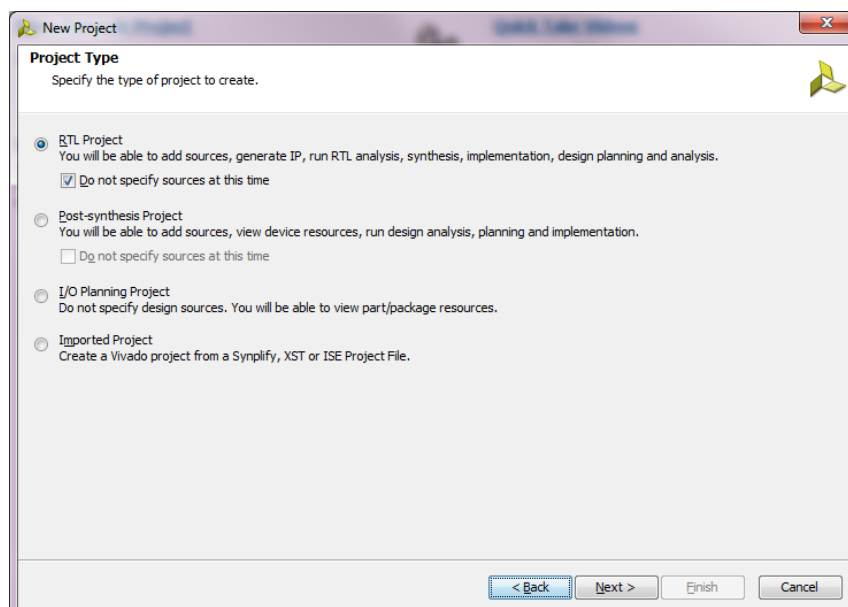


Electrical and Computer Engineering  
PO Box 23874 | Doha, Qatar  
341G Texas A&M Engineering Building | Education City  
(Office) +974.4423.0194 | (Fax) +974.4423.0064 | GMT+3

- In the next window enter “*test\_system*” as the **Project name**, specify the directory in which to store the project’s files “*c:/oled\_tamuq*”- create the folder *oled\_tamuq* in your c:\ drive(Windows machine assumed), uncheck **Create project subdirectory**.

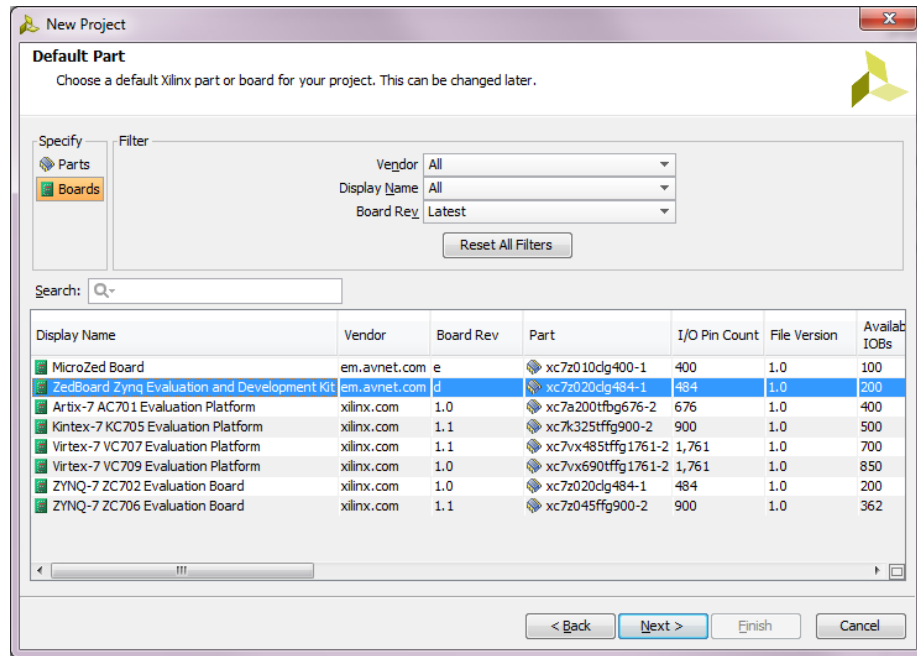


- Next, specify the project type, Use the default selection **RTL Project**, and check **Do not specify sources at this time**. RTL stands for Register Transfer Language, by choosing this option we will have the flexibility to add/modify source files later on.



Electrical and Computer Engineering  
PO Box 23874 | Doha, Qatar  
341G Texas A&M Engineering Building | Education City  
(Office) +974.4423.0194 | (Fax) +974.4423.0064 | GMT+3

- Next, specify the board that you want to test your project on, select the **Zedboard Zynq Evaluation and Development Board**. Click **Next** then click **Finish**.

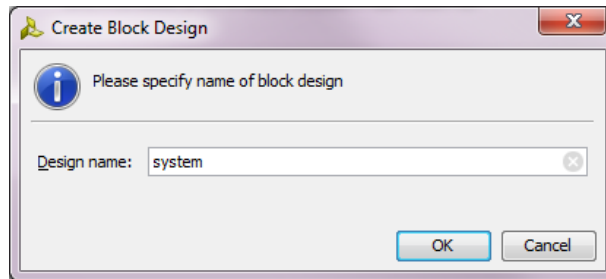



The New Project wizard closes and the project you just created opens in Vivado.

Electrical and Computer Engineering  
PO Box 23874 | Doha, Qatar  
341G Texas A&M Engineering Building | Education City  
(Office) +974.4423.0194 | (Fax) +974.4423.0064 | GMT+3

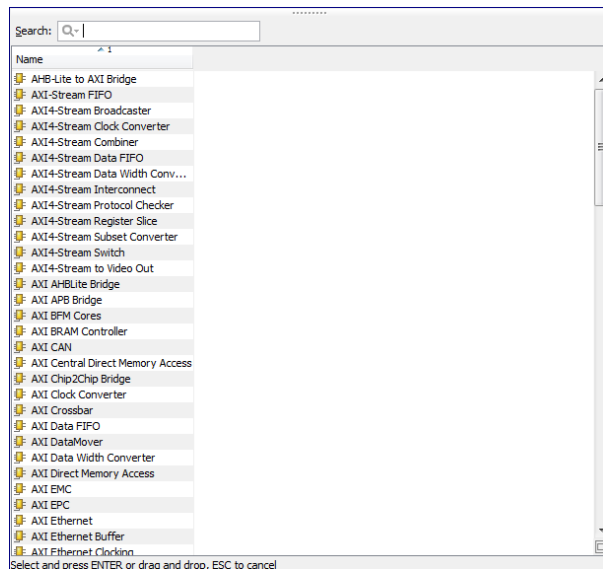
### B- Creating the ARM processor system using the IP Integrator

1. Click on the **Create Block Design** in the **Project Manager** available on the left top side of Vivado.
2. Type a name for the module and click **OK**. For this example, use the name: **system**.



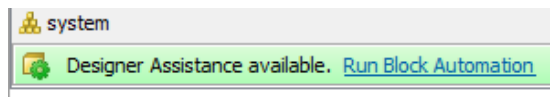
3. After clicking **OK**, you will be presented a blank block diagram view in the Vivado graphical user interface. In the diagram we will add the hardware blocks needed for the system.
4. Now, we will add **ZYNQ7 Processing System** block containing the ARM processor , This is done by launching the **Add IP**  wizard, or you can right click on a blank space in the block diagram, The IP catalog window will appear showing you all the IP that can be added in this design.

Electrical and Computer Engineering  
PO Box 23874 | Doha, Qatar  
341G Texas A&M Engineering Building | Education City  
(Office) +974.4423.0194 | (Fax) +974.4423.0064 | GMT+3

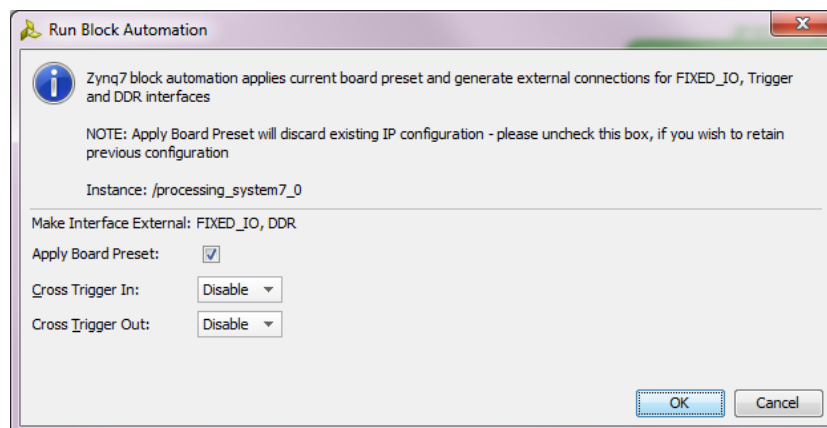


Either scroll down to the very bottom or search using the keyword *zynq*, double click on **ZYNQ7 Processing System**. The ZYNQ7 Processing System block has been placed in the block diagram view. The ports shown on the block diagram are defined by the default settings for this block as specified by target development board.

Click **Run Block Automation** in the green

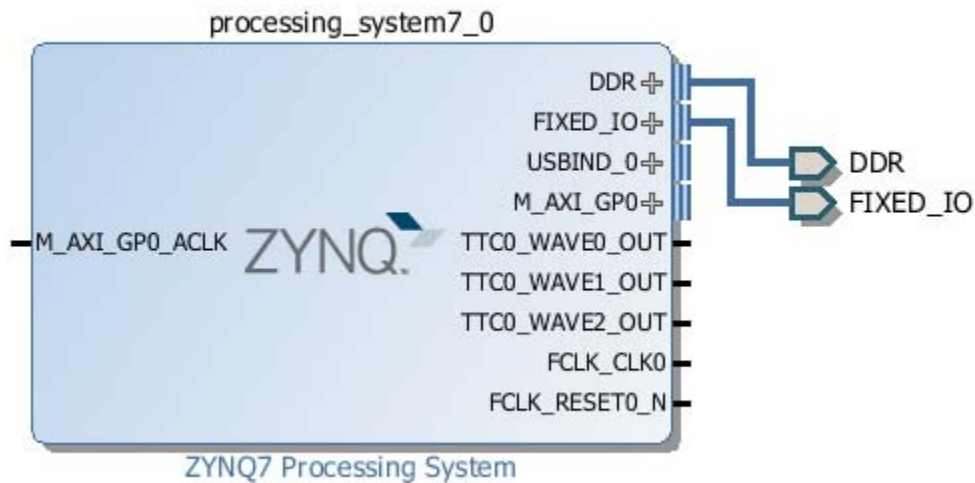


information bar. Select **processing\_system7\_0** and make sure **Apply Board Preset** is checked, then click **OK**.



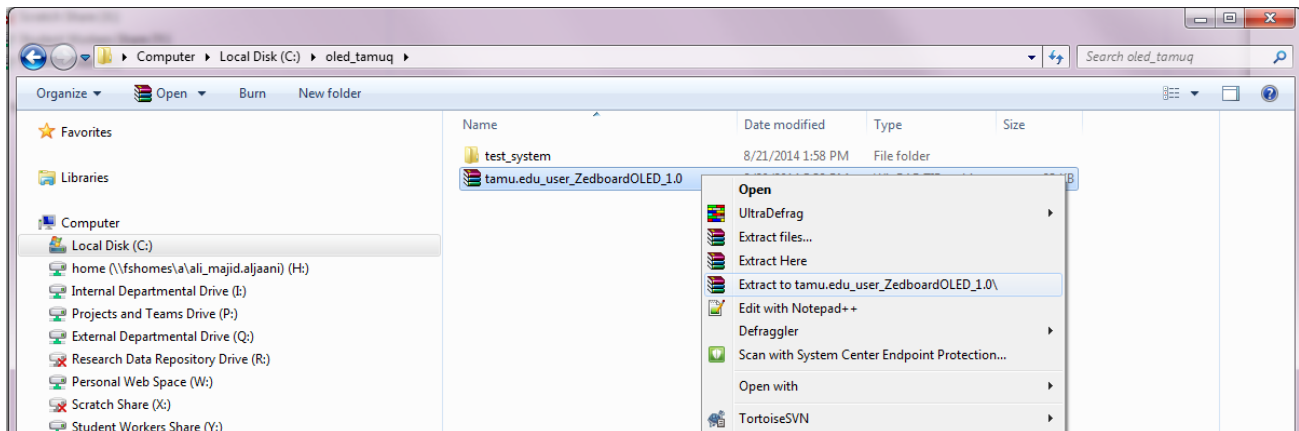
Electrical and Computer Engineering  
PO Box 23874 | Doha, Qatar  
341G Texas A&M Engineering Building | Education City  
(Office) +974.4423.0194 | (Fax) +974.4423.0064 | GMT+3

This will create external ports for the processing system, and apply physical constraints to these ports as shown in the figure below.



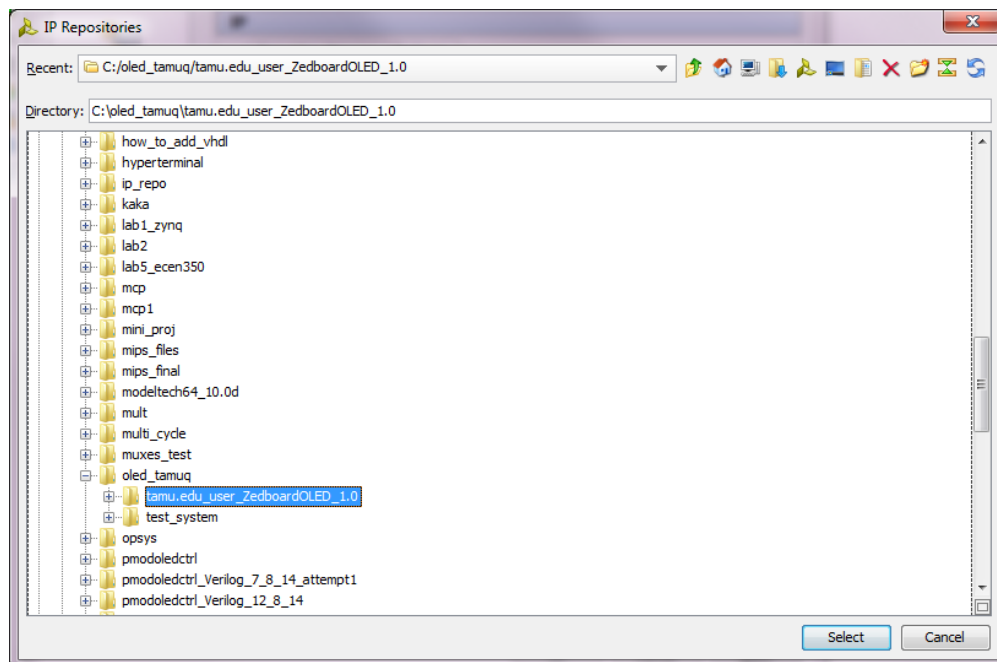
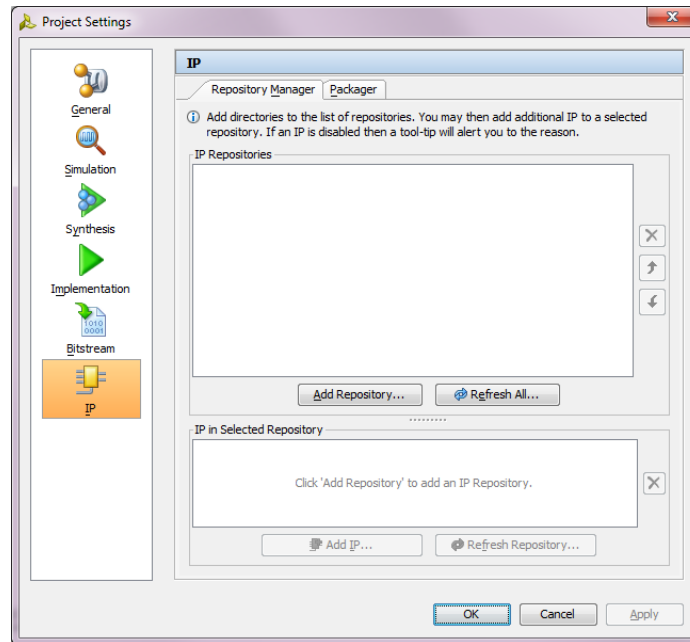
## C- Adding the ZedboardOLED IP to the IP repository

Before proceeding with this part, make sure you extract the archive file of the ZedboardOLED IP core (*tamu.edu\_user\_ZedboardOLED\_1.0.zip*) to the “C:\oled\_tamuq” directory.



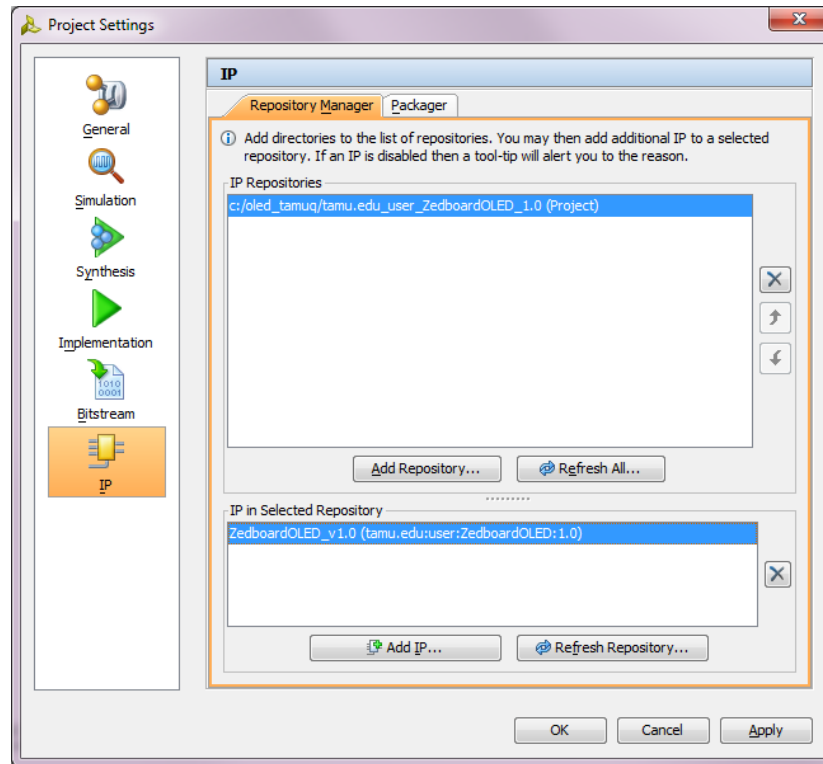
Electrical and Computer Engineering  
PO Box 23874 | Doha, Qatar  
341G Texas A&M Engineering Building | Education City  
(Office) +974.4423.0194 | (Fax) +974.4423.0064 | GMT+3

In the flow navigator pane, click on the **Project Setting** tab, select **IP**, then click on the **Add Repository...** button. A browsing window will open up, browse to the location where you extracted the IP core (`C:\oled_tamuq\tamu.edu_user_ZedboardOLED_1.0`), click **Select**.



Electrical and Computer Engineering  
PO Box 23874 | Doha, Qatar  
341G Texas A&M Engineering Building | Education City  
(Office) +974.4423.0194 | (Fax) +974.4423.0064 | GMT+3

Notice how the tool has detected a new IP in the directory, click **Apply** then **OK**.




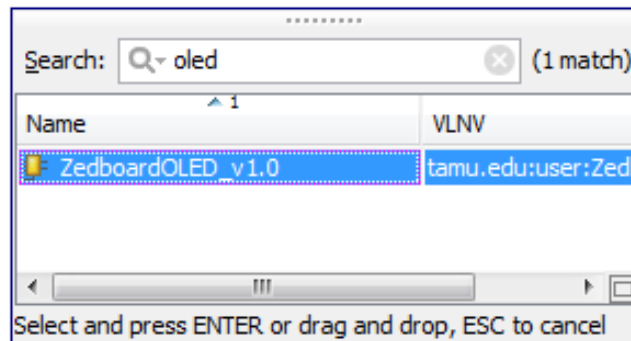
With this you have added the **ZedboardOLED\_v1.0** to the IP repository of the current project, the next step is to add it to the block design and connect it to the **Zynq processing system** from one side, and to the OLED panel from the other side using external ports.



Electrical and Computer Engineering  
PO Box 23874 | Doha, Qatar  
341G Texas A&M Engineering Building | Education City  
(Office) +974.4423.0194 | (Fax) +974.4423.0064 | GMT+3

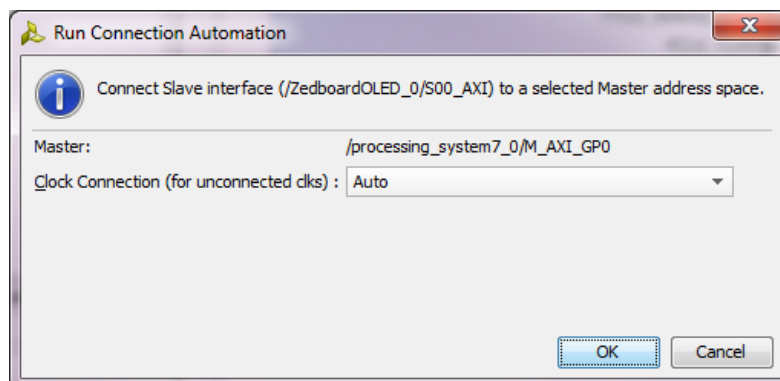
## D- Connecting the ZedboardOLED IP core

- 1- Launch the **Add IP**  wizard again, and type “oled”. The ZedboardOLED\_V1.0 will show up, double click on the IP to add it to the block design.



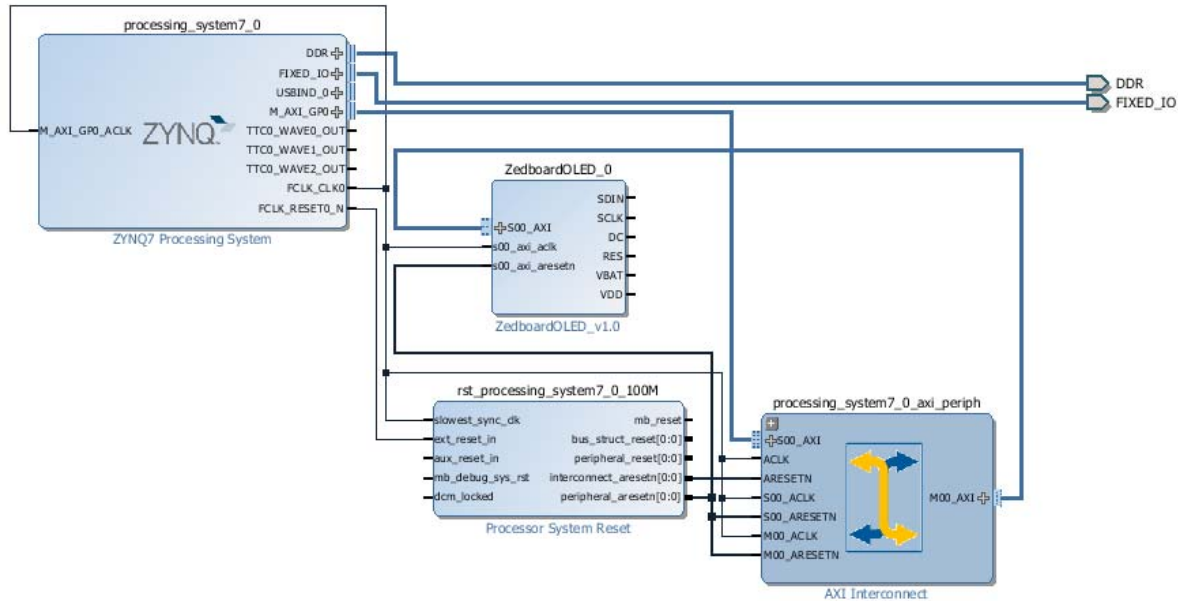
- 2- Notice that the **Designer Assistance** is available now in the green information bar, it will take care of connecting the IP core to the AXI subsystem of the processing system, configuring the clock and reset, assigning base address (0x43C00000) to the IP, and adding the necessary hardware for the interconnect and synchronization.

Click on the **Run Connection Automation** and select **/ZedboardOLED\_0/S00\_AXI**. In the **Run Connection Automation** window that pops up, leave the **Clock Connection** set to **Auto**, and click **OK**.



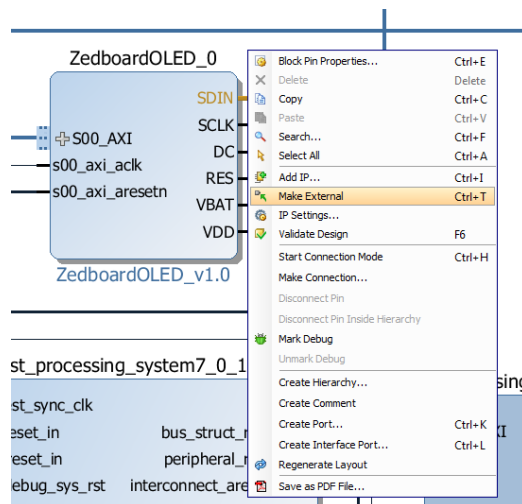
Electrical and Computer Engineering  
PO Box 23874 | Doha, Qatar  
341G Texas A&M Engineering Building | Education City  
(Office) +974.4423.0194 | (Fax) +974.4423.0064 | GMT+3

The block diagram should look something similar to this:



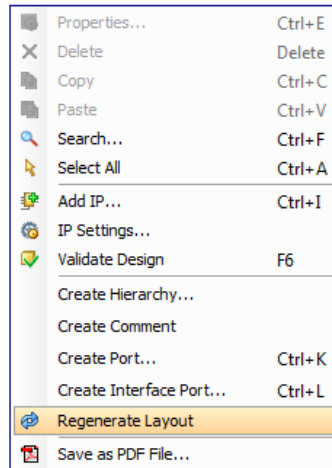
Notice that the **Designer Assistance** has added the **AXI Interconnect** and the **Processor System Reset** automatically.

Hover the mouse on the ZedboardOLED SDIN port until it changes to a pencil shape, then right click and select **Make External**. Repeat the same process for (SCLK, DC , RES ,VBAT ,VDD) ports.

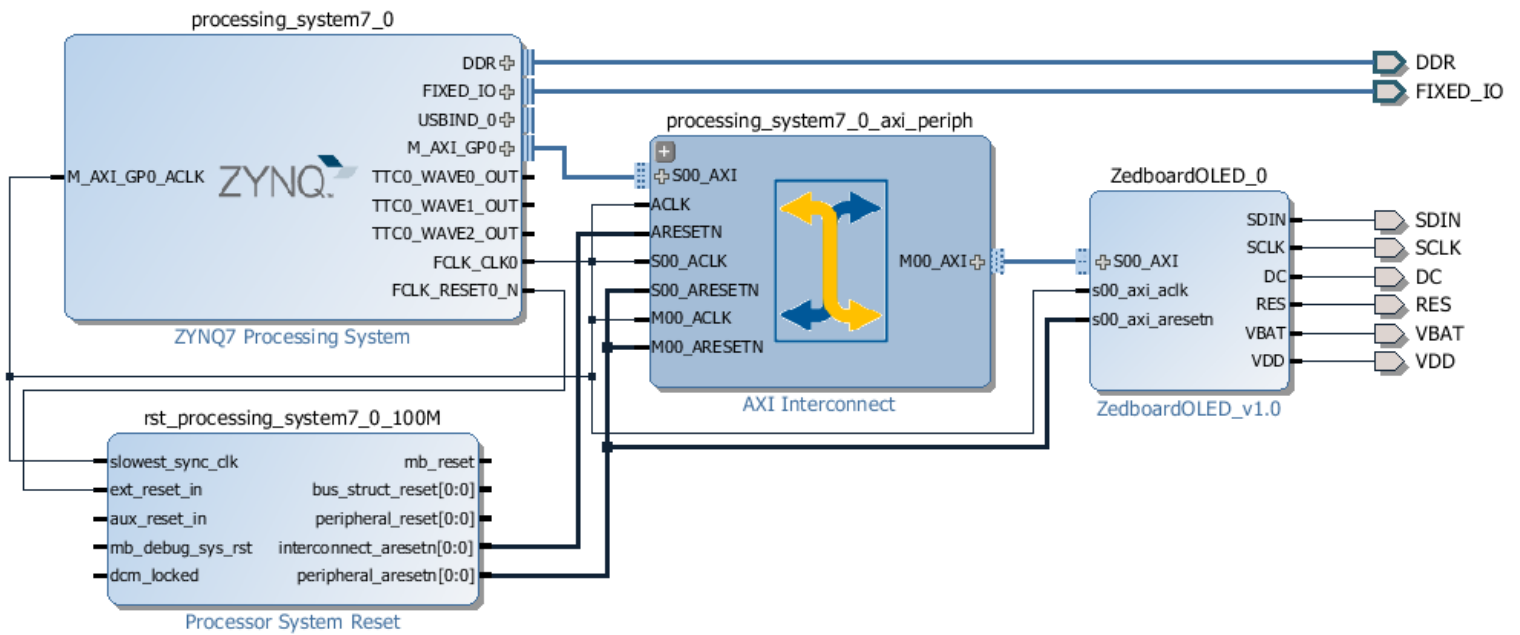


Electrical and Computer Engineering  
PO Box 23874 | Doha, Qatar  
341G Texas A&M Engineering Building | Education City  
(Office) +974.4423.0194 | (Fax) +974.4423.0064 | GMT+3

- 3- Right click on any free location on the block diagram and select **Regenerate Layout**, this will organize the blocks in the design in a neat way.

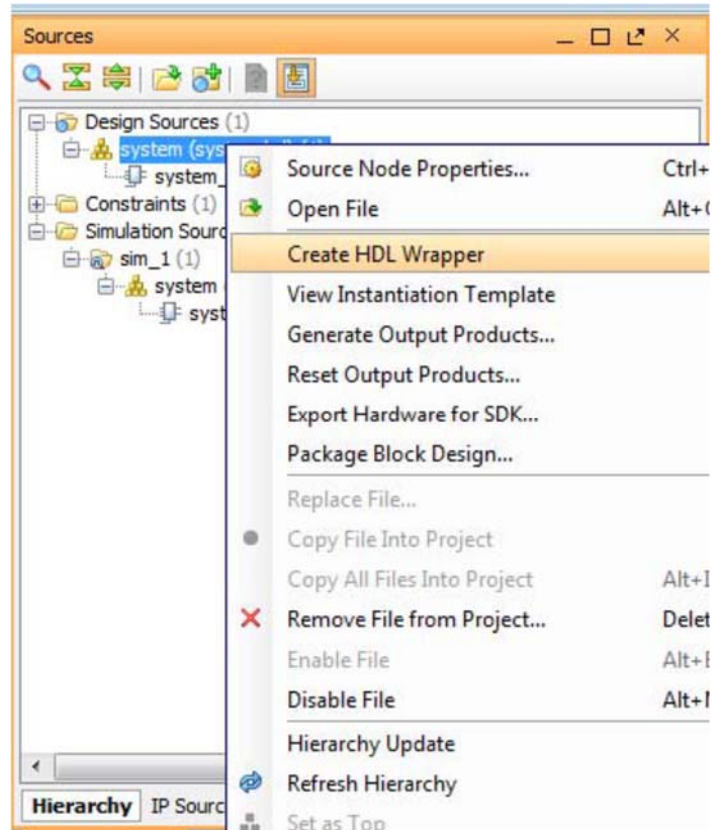


The final layout might look like the one below:



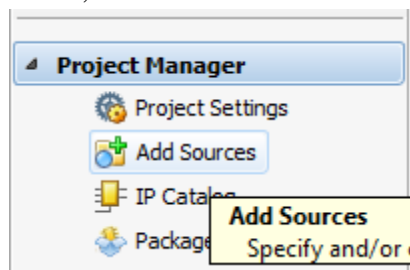
Electrical and Computer Engineering  
PO Box 23874 | Doha, Qatar  
341G Texas A&M Engineering Building | Education City  
(Office) +974.4423.0194 | (Fax) +974.4423.0064 | GMT+3

- 4- In the source pane, select the system block diagram “*system.bd*”, right-click and select **Create HDL Wrapper**, in the next window that shows up leave the option **Let Vivado manage wrapper and auto-update** selected. Click **OK**.



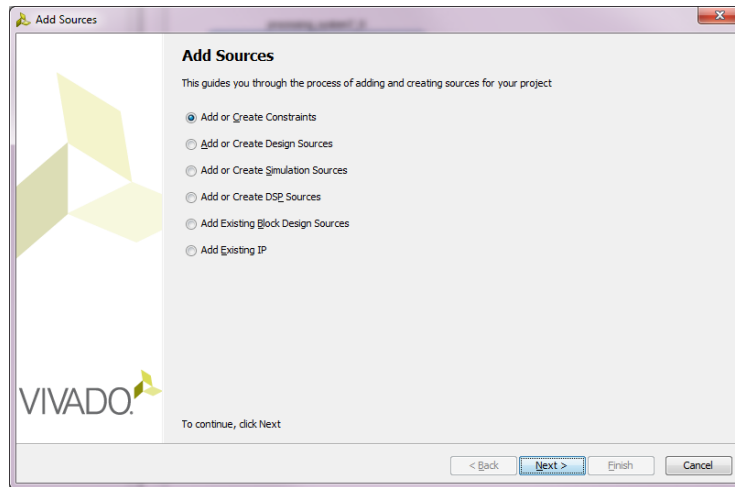
Now we finished connecting the IP to the processing system through AXI-interface, next step is to connect the external ports of the IP core to the actual Zynq pins hardwired to the OLED panel.

- 5- In the Flow Navigator window, select **Add Sources** from the Project Manager section.

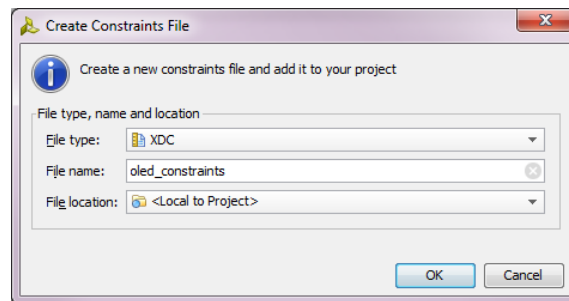


The Add Sources dialogue will open. Select **Add or Create Constraints**.

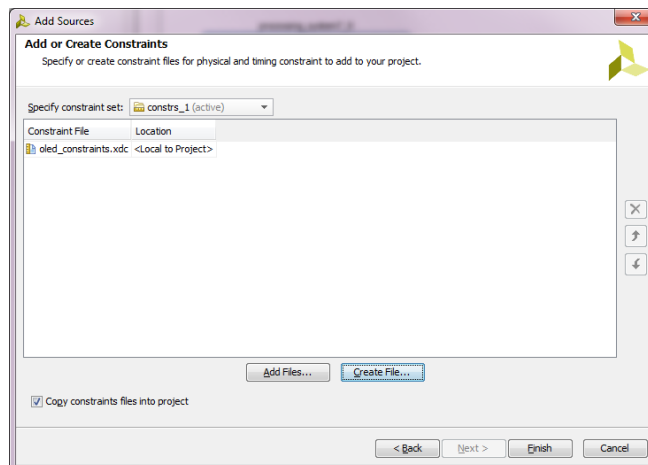
Electrical and Computer Engineering  
PO Box 23874 | Doha, Qatar  
341G Texas A&M Engineering Building | Education City  
(Office) +974.4423.0194 | (Fax) +974.4423.0064 | GMT+3



Click **Next**, then click **Create File...**, Select **XDC** as the **File type** and enter *oled\_constraints* as the **File name**.

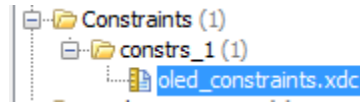


Click **OK**, Click **Finish** to create the file and close the dialogue.



Electrical and Computer Engineering  
PO Box 23874 | Doha, Qatar  
341G Texas A&M Engineering Building | Education City  
(Office) +974.4423.0194 | (Fax) +974.4423.0064 | GMT+3

In the **Sources** tab, expand the **Constraints** entry and open the newly created XDC file by double clicking on *oled\_constraints.xdc*



Add the following lines to the constraints file, alternatively, they can be copied from the source file *oled\_constraints.xdc* available with the package

```
set_property PACKAGE_PIN U10 [get_ports DC]
set_property PACKAGE_PIN U9 [get_ports RES]
set_property PACKAGE_PIN AB12 [get_ports SCLK]
set_property PACKAGE_PIN AA12 [get_ports SDIN]
set_property PACKAGE_PIN U11 [get_ports VBAT]
set_property PACKAGE_PIN U12 [get_ports VDD]
set_property IOSTANDARD LVCMOS33 [get_ports DC]
set_property IOSTANDARD LVCMOS33 [get_ports RES]
set_property IOSTANDARD LVCMOS33 [get_ports SCLK]
set_property IOSTANDARD LVCMOS33 [get_ports SDIN]
set_property IOSTANDARD LVCMOS33 [get_ports VBAT]
set_property IOSTANDARD LVCMOS33 [get_ports VDD]
```

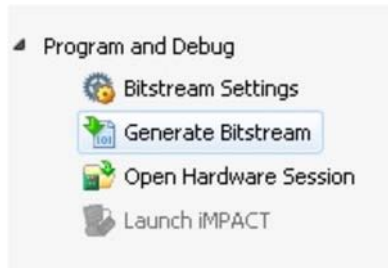
This connects the external ports of the ZedboardOLED IP core to specific pins on the Zynq device. The specific pins are connected to the OLED panel on the Zedboard.

Save the constraints file by pressing (Ctrl+S).

Electrical and Computer Engineering  
PO Box 23874 | Doha, Qatar  
341G Texas A&M Engineering Building | Education City  
(Office) +974.4423.0194 | (Fax) +974.4423.0064 | GMT+3

### E- Generating Bitstream

In the Program and Debug section in the Flow Navigator pane, click **Generate Bitstream**.



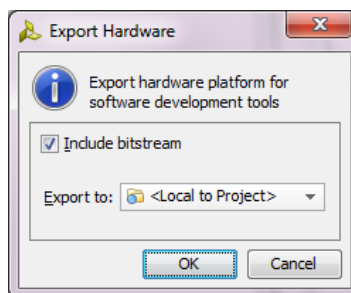
A dialog box will appear to ask you to save the modification you made, click **Save**, also you might get a dialog that says “*No Implementation Result Available*”, click **Yes** to run synthesis and implementation.

Generating the Bitstream may invoke the entire implementation process after synthesis, click **Yes** to run implementation when prompted. Implementation may take a while to complete, depending on the performance of your machine.

After the bitstream generation completes select **View Reports** in the dialog box, and click **OK**.

### F- Exporting hardware design to SDK

- 1- Click **File > Export > Export Hardware**, make sure you leave **Include bitstream** Checked.

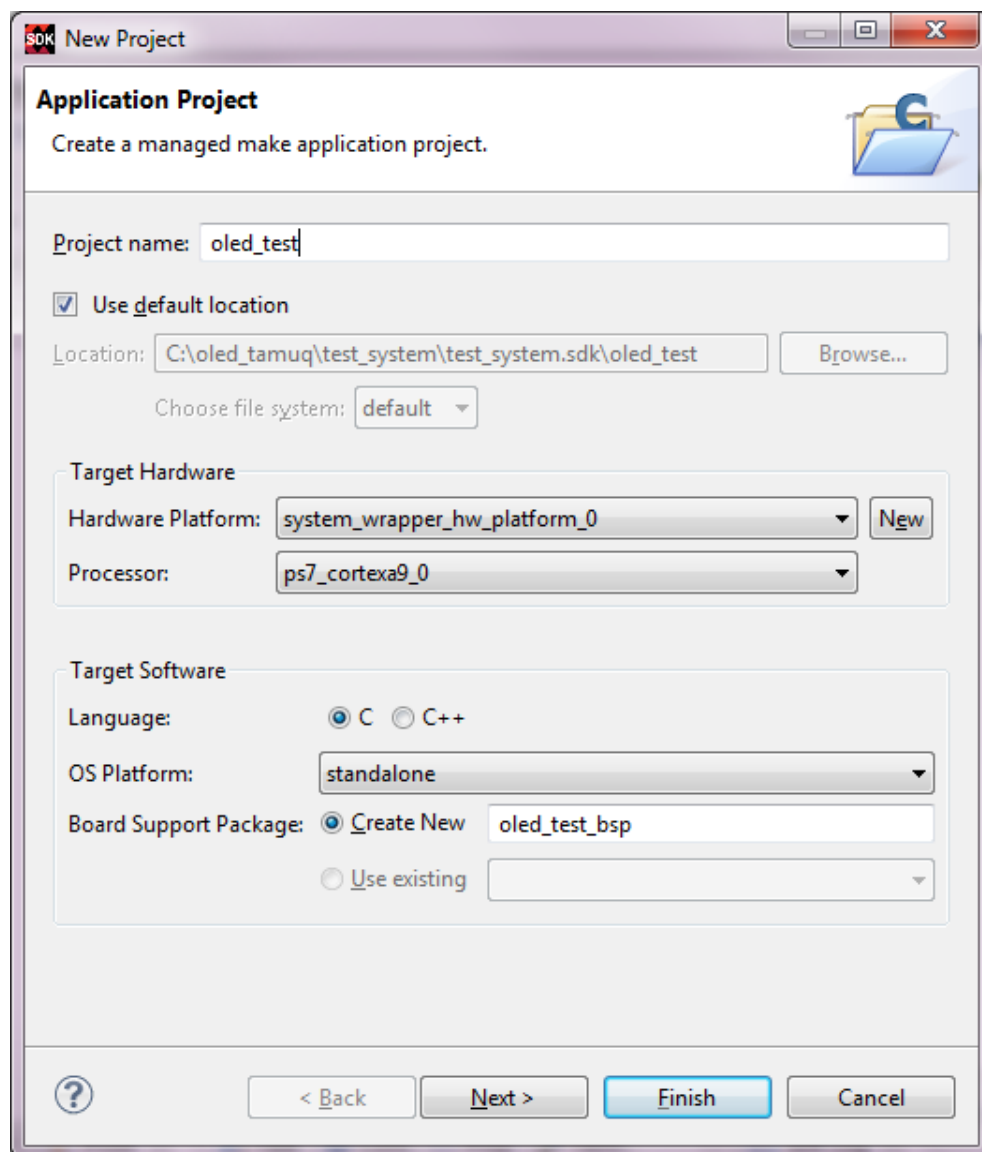


- 2- Select **File>Launch SDK**, this will open up SDK, notice that all the files related to design including the IP has already been exported to SDK in the previous step, among the resources that got exported is the driver of the ZedboardOLED\_v1\_0 IP core.

Electrical and Computer Engineering  
PO Box 23874 | Doha, Qatar  
341G Texas A&M Engineering Building | Education City  
(Office) +974.4423.0194 | (Fax) +974.4423.0064 | GMT+3

## G- Working with SDK

1. In SDK, select **File > New > Application Project**.
2. In the next window, enter the parameters as provided in the snapshot below:



**SDK New Project**

**Application Project**  
Create a managed make application project.

Project name:

☒ Use default location

Location:

Choose file system:

**Target Hardware**

Hardware Platform:

Processor:

**Target Software**

Language: ☒ C ☐ C++

OS Platform:

Board Support Package: ☒ Create New  ☐ Use existing

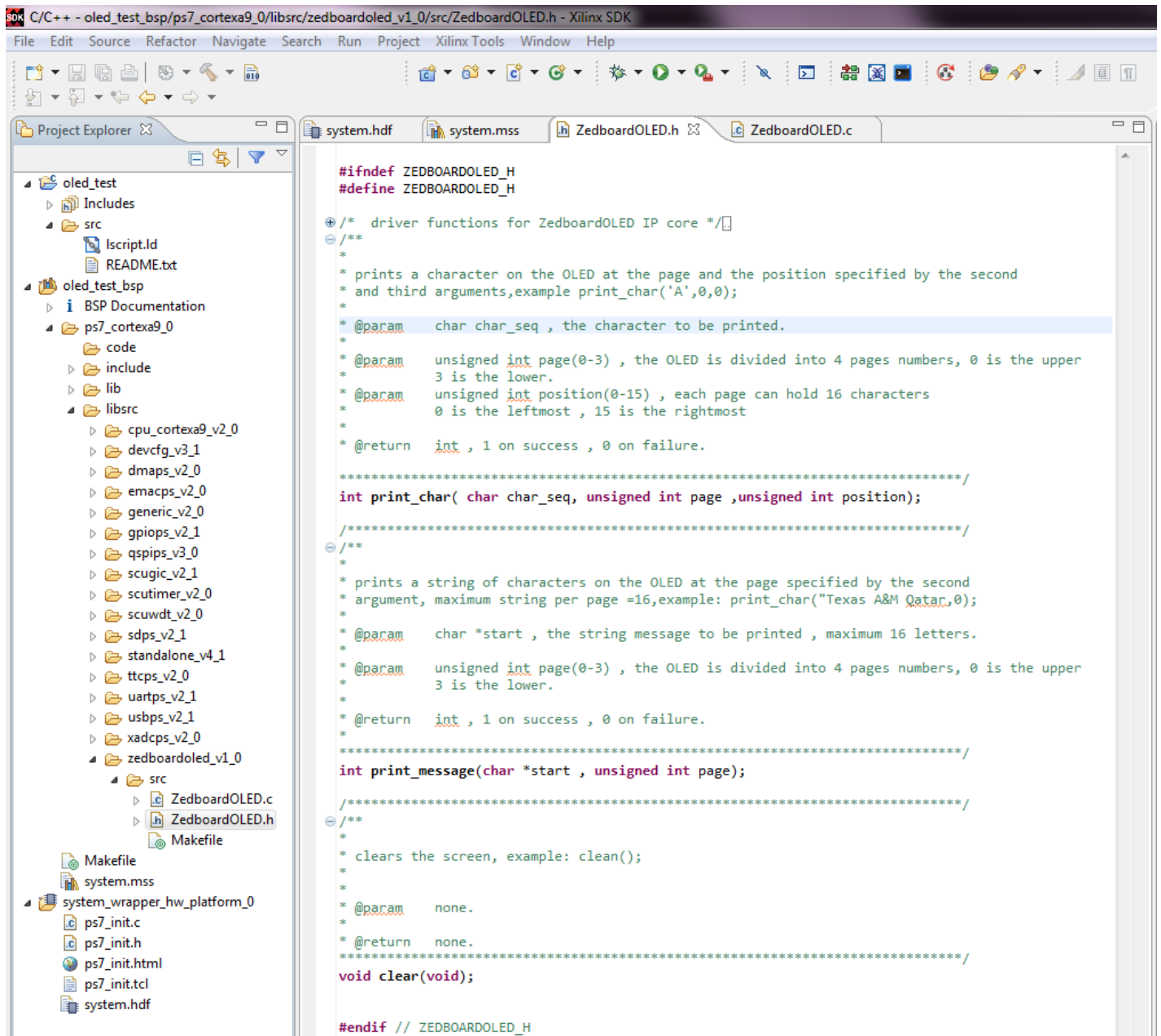
In the next window, select **Empty Application** from the available templates and click **Finish**.



Electrical and Computer Engineering  
PO Box 23874 | Doha, Qatar  
341G Texas A&M Engineering Building | Education City  
(Office) +974.4423.0194 | (Fax) +974.4423.0064 | GMT+3

This will compile the BSD and the related drivers.

- Expand **oled\_test\_bsp>ps7\_cortexa9\_v2\_0>libsrc>zedboardoled\_v1\_0>src**, and examine both **ZedboardOLED.c**, and **ZedboardOLED.h** for the functions implemented to be used with the OLED screen.



The screenshot shows the Xilinx SDK IDE with the project **oled\_test\_bsp** expanded. The **ps7\_cortexa9\_v2\_0** component is selected, and the **libsrc** folder is expanded, showing the **zedboardoled\_v1\_0** component. The **src** folder of **zedboardoled\_v1\_0** is expanded, showing the **ZedboardOLED.c** and **ZedboardOLED.h** files. The **ZedboardOLED.h** file is open in the editor, showing the following code:

```
#ifndef ZEDBOARDOLED_H
#define ZEDBOARDOLED_H

/* driver functions for ZedboardOLED IP core */
/**
 * prints a character on the OLED at the page and the position specified by the second
 * and third arguments, example print_char('A',0,0);
 *
 * @param char char_seq , the character to be printed.
 *
 * @param unsigned int page(0-3) , the OLED is divided into 4 pages numbers, 0 is the upper
 * 3 is the lower.
 * @param unsigned int position(0-15) , each page can hold 16 characters
 * 0 is the leftmost , 15 is the rightmost
 *
 * @return int , 1 on success , 0 on failure.
 */
int print_char( char char_seq, unsigned int page ,unsigned int position);

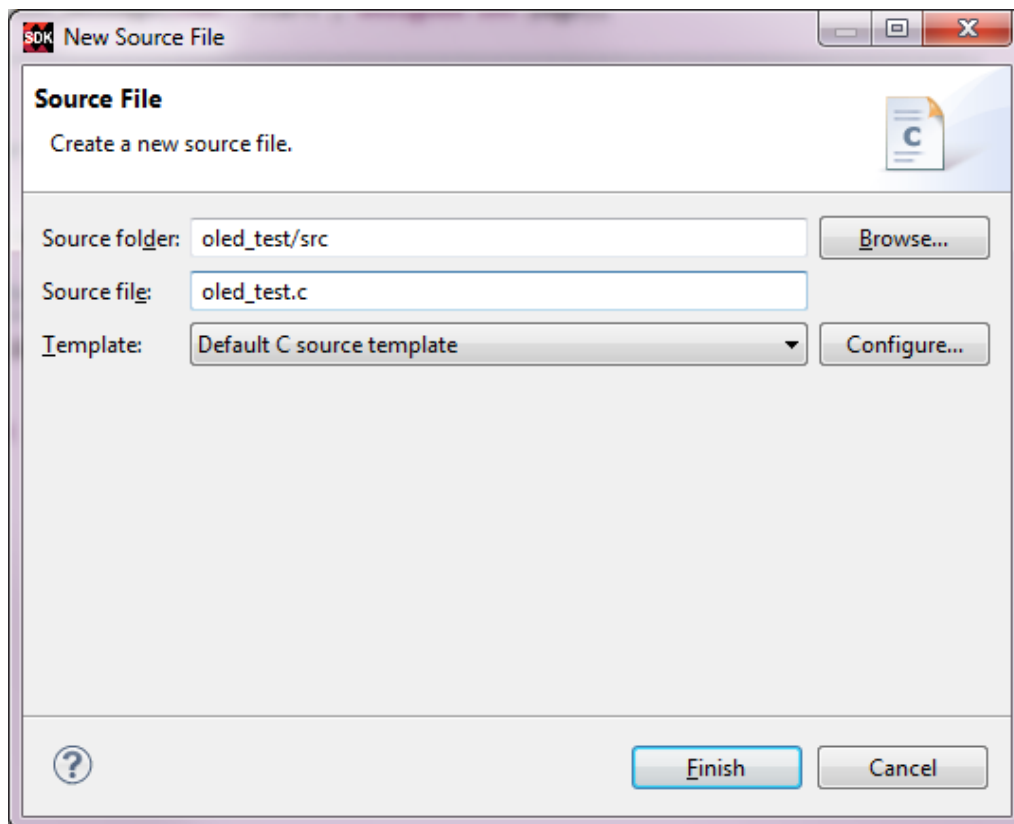
/**
 * prints a string of characters on the OLED at the page specified by the second
 * argument, maximum string per page =16, example: print_char("Texas A&M Qatar,0);
 *
 * @param char *start , the string message to be printed , maximum 16 letters.
 * @param unsigned int page(0-3) , the OLED is divided into 4 pages numbers, 0 is the upper
 * 3 is the lower.
 *
 * @return int , 1 on success , 0 on failure.
 */
int print_message(char *start , unsigned int page);

/**
 * clears the screen, example: clean();
 *
 * @param none.
 *
 * @return none.
 */
void clear(void);

#endif // ZEDBOARDOLED_H
```

Electrical and Computer Engineering  
 PO Box 23874 | Doha, Qatar  
 341G Texas A&M Engineering Building | Education City  
 (Office) +974.4423.0194 | (Fax) +974.4423.0064 | GMT+3

- Now we will write a simple application that uses these functions, Expand *oled\_test* directory, right click on the *src* directory , select **New->Source File** , In the next window that shows up, type “*oled\_test.c*” in the source file and click **Finish**.



Copy and paste the following C code into the editor view of “*oled\_test.c*”, click **Save** or hit (Ctrl+S), by doing so, both the *oled\_test* application, and its BSP are compiled automatically and the executable .elf file is generated.

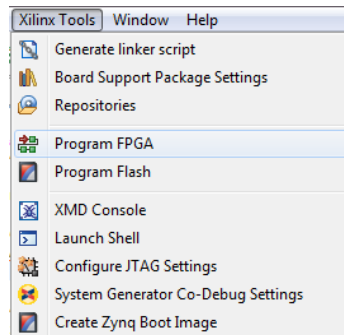
```
int main(void){
    clear();
    print_message("Texas A&M Qatar",0);
    print_message("ECEN Department",2);
    print_message("    ECEN449    ",3);
    return (1);
}
```

Click **Project-> clean** (in case you get any errors with the BSD).

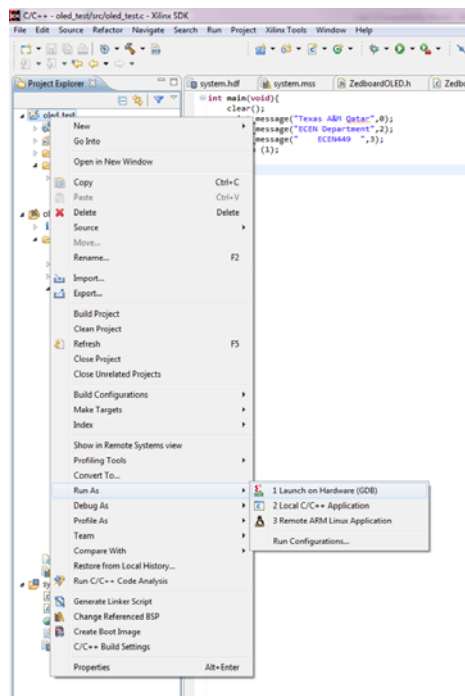
Electrical and Computer Engineering  
PO Box 23874 | Doha, Qatar  
341G Texas A&M Engineering Building | Education City  
(Office) +974.4423.0194 | (Fax) +974.4423.0064 | GMT+3

## H- Downloading the bitstream and running the application (Hardware verifications)

- 1- Select **Xilinx Tools-> Program FPGA** to download the Bitstream ( this will take few seconds ).



- 2- Select *oled\_test* project directory-> **Run As-> Launch on Hardware (GDB)** to run the *oled\_test* application on the ARM processor.



Electrical and Computer Engineering  
PO Box 23874 | Doha, Qatar  
341G Texas A&M Engineering Building | Education City  
(Office) +974.4423.0194 | (Fax) +974.4423.0064 | GMT+3

You should see the following message on the OLED screen.

