# A Hello World tutorial for the MYIR Z-turn board (Zynq 7020 SoC)

Thanks to Mr. Juan Abelaira of Akteevy to write this tutorial and share with us.

In this tutorial you will learn to configure the Processing System (PS) for the Z-turn board with an xc7z7020, create a Hello World software application with the Xilinx SDK and run it using the JTAG interface.

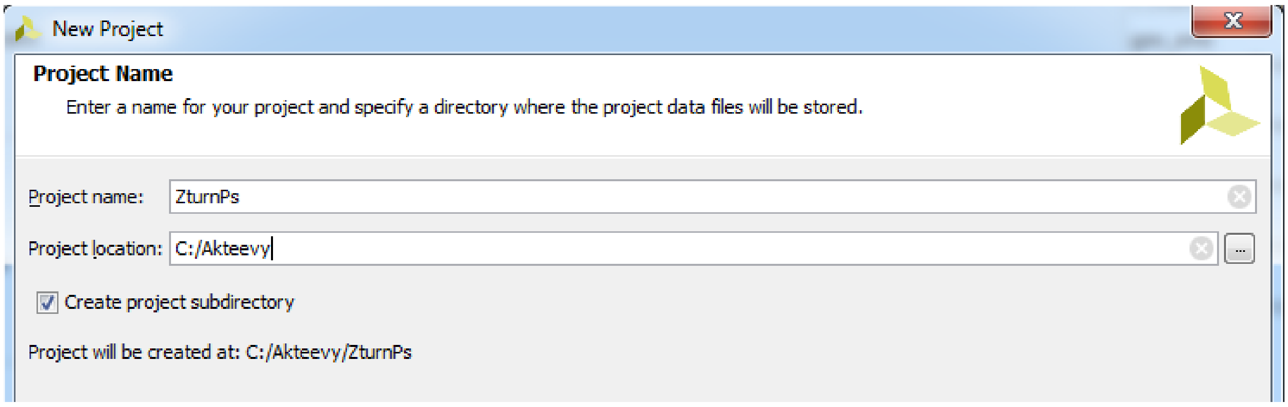
Be sure you have the following before starting:

- A Z-turn board, mounted or not on the Cape board
- A Xilinx programming cable with standard 14-pin JTAG connector
- An A - mini B type USB cable for supply and serial communications with the Z-turn
- Xilinx Vivado and SDK installed (I'm using version 2015.4 but it will be similar with other releases)
- HyperTerminal, Tera Term or any console application.

Connect the hardware as in the picture above. Use that specific USB port of the Z-turn, not the other one. As for the programming cable, check your connections and cable as your programmer might differ from this one. Refer to the Z-turn schematics and your programmer documentation. The blue LED on the Z-turn should be on and the programmer LED should turn green if correctly connected.

## 1. Create and configure the Processor System (PS)

Launch Vivado and create a new project. Enter the name *ZturnPs* as the project name and choose the project location (in this case C:/Akteevy/ but any other will do).


New Project

**Project Name**

Enter a name for your project and specify a directory where the project data files will be stored.

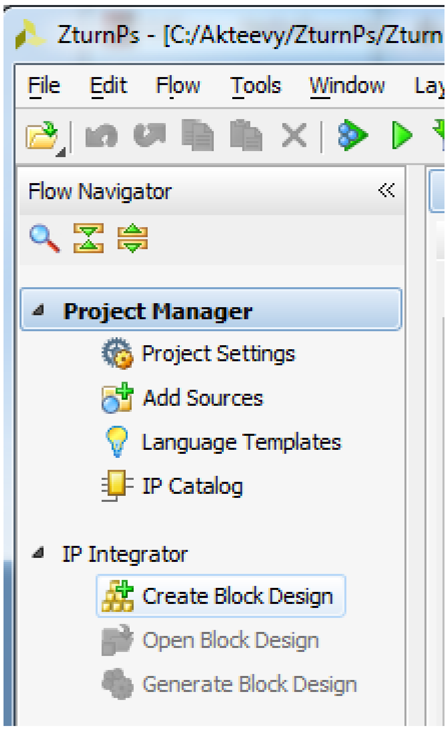
Project name: ZturnPs

Project location: C:/Akteevy

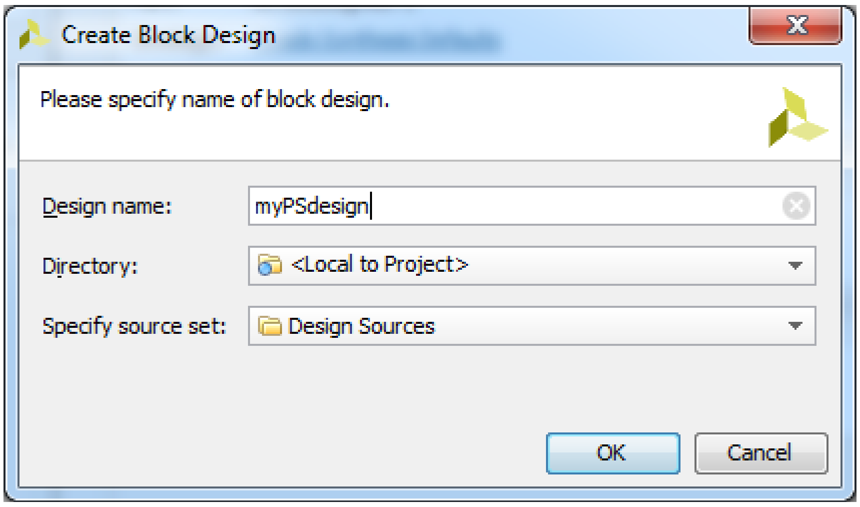
[✓] Create project subdirectory

Project will be created at: C:/Akteevy/ZturnPs

In the next window, select the 'RTL project' option and click 'Do not specify sources at this time'. Then select the part xc7z020clg400-1 (or xc7z010clg400-1 if you have that board version). When Vivado opens, in the flow navigator pane (left), click on 'Create Block Design'



On the pop up window, change the design name to myPSdesign and click OK


Create Block Design

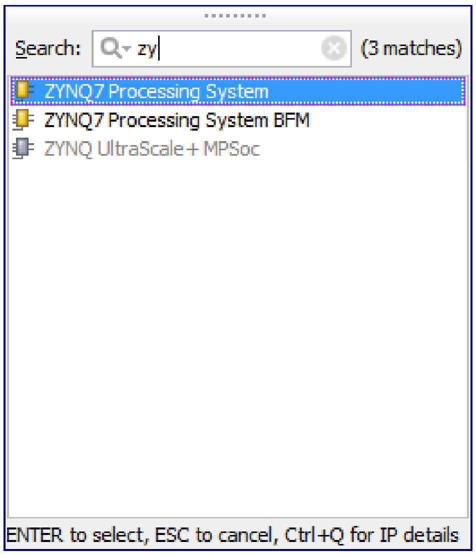
Please specify name of block design.

Design name: myPSdesign

Directory: <Local to Project>

Specify source set: Design Sources

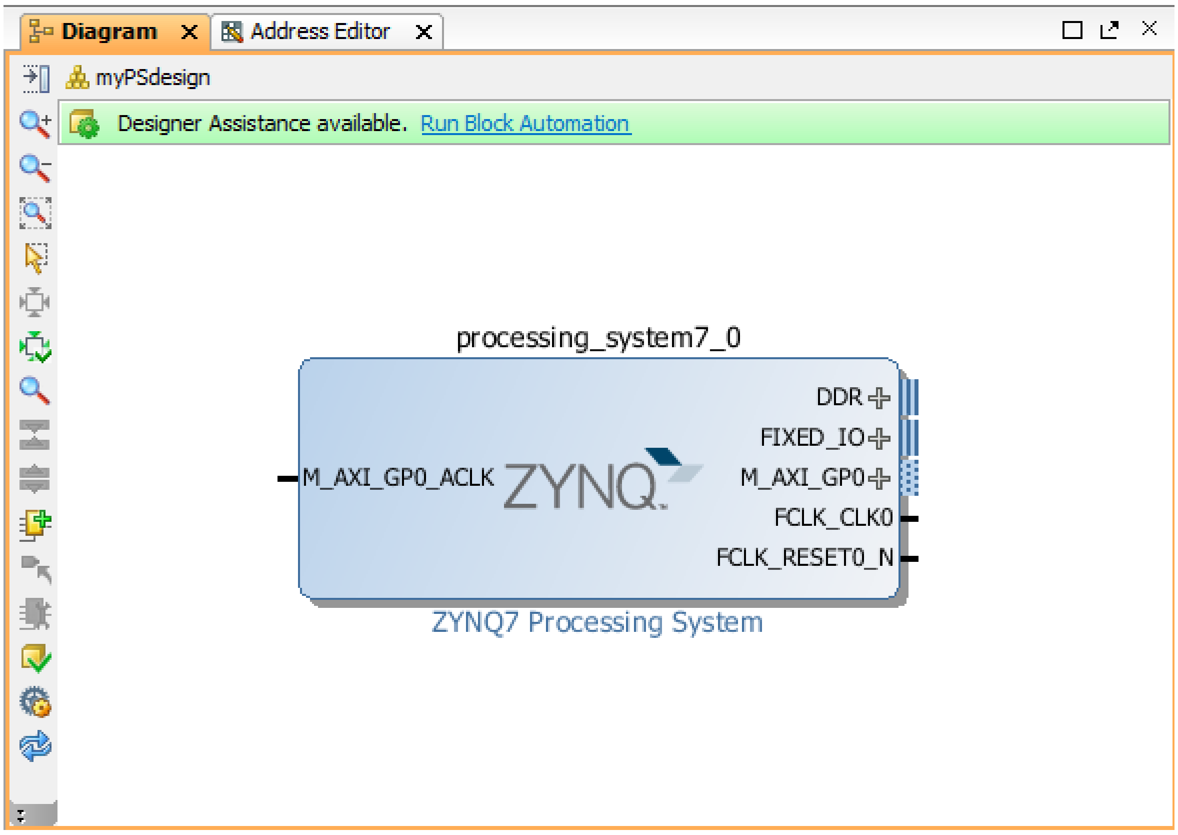
An empty diagram opens, on its left, click on 'Add IP', on the window that appears type just 'zy' and the two entries below will appear. Click on 'Zynq Processing System' and hit ENTER to add it to the diagram.


Search: zy (3 matches)

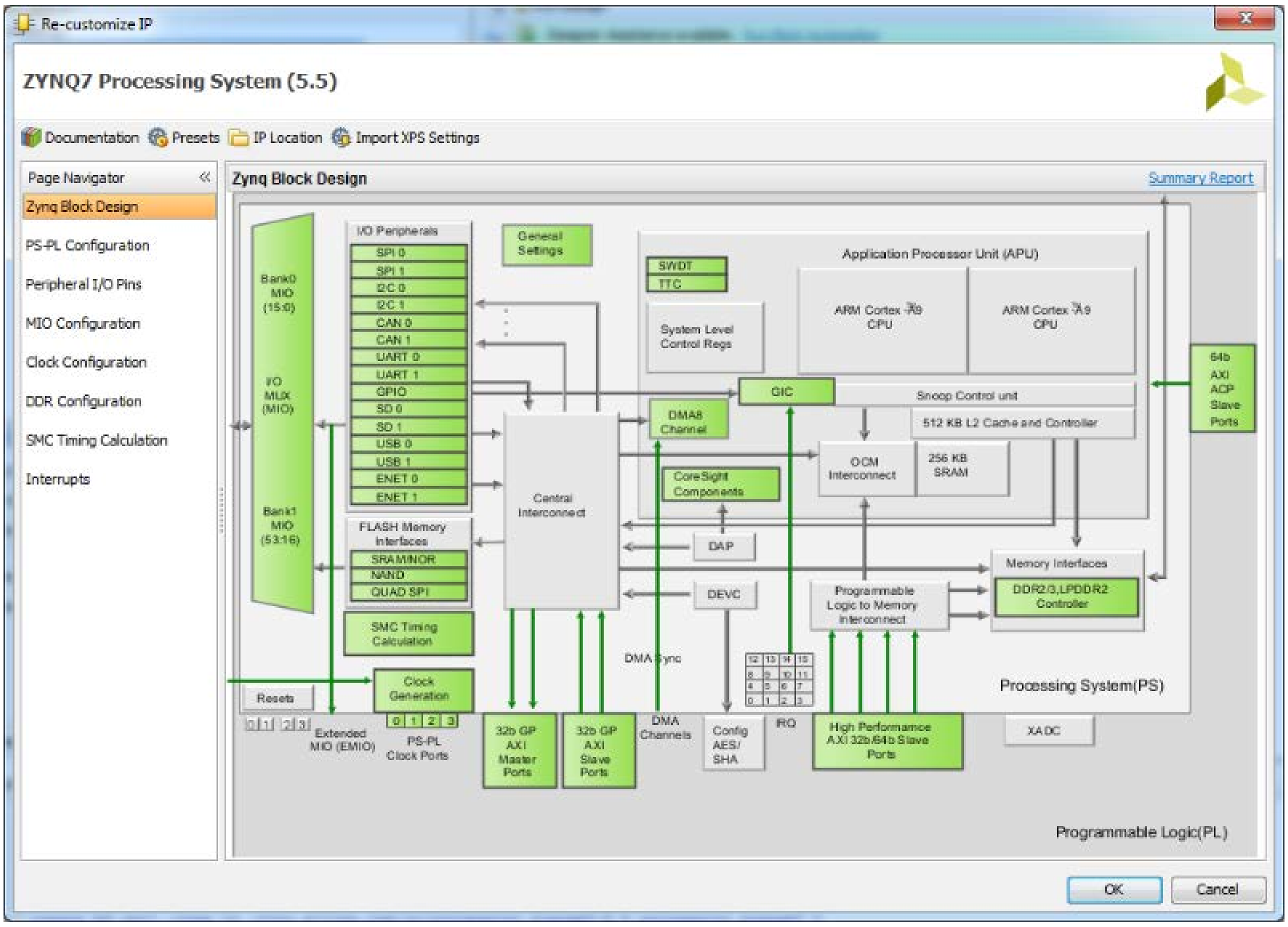
ZYNQ7 Processing System
ZYNQ7 Processing System BFM
ZYNQ UltraScale+ MPSoc

ENTER to select, ESC to cancel, Ctrl+Q for IP details

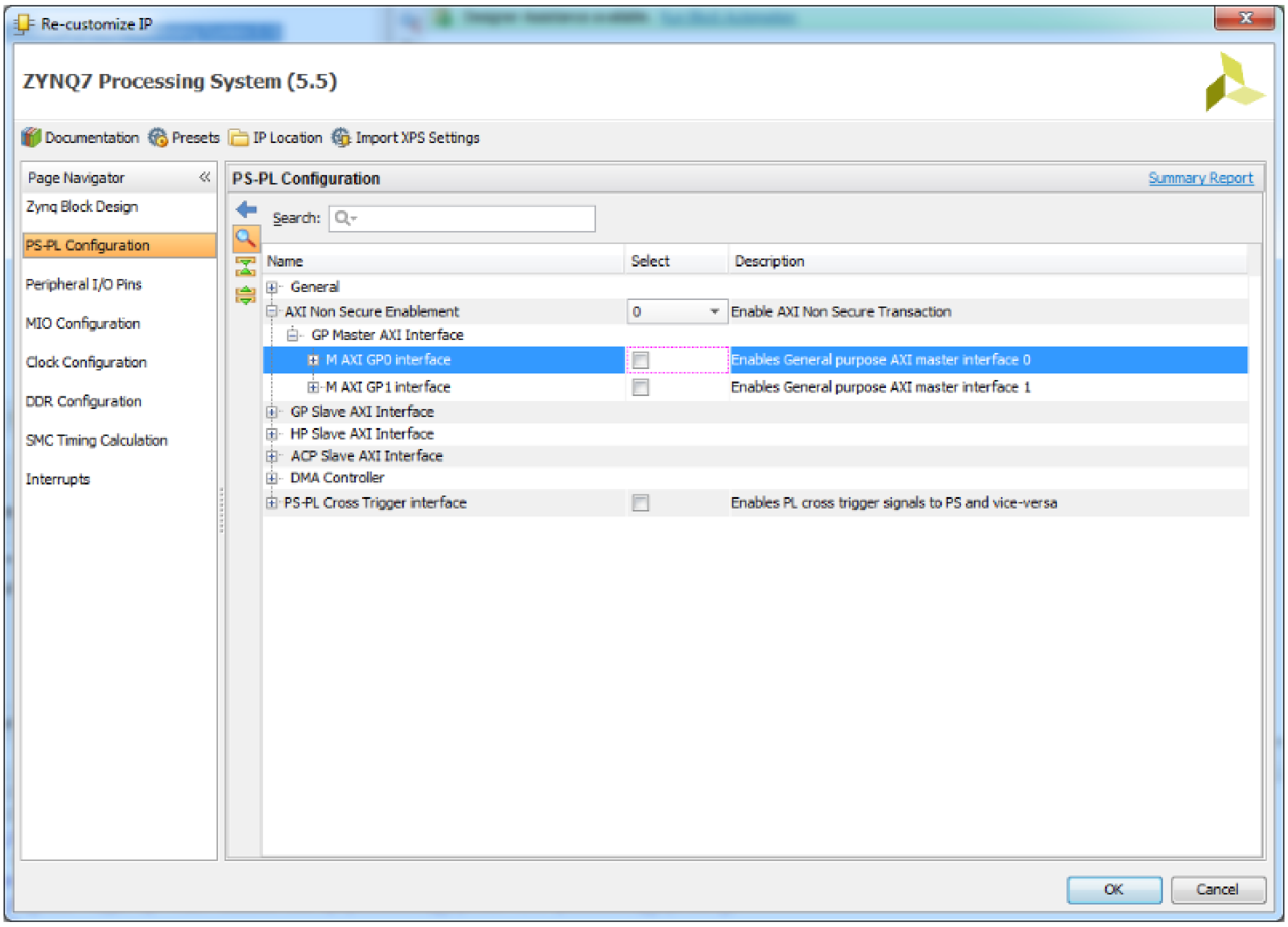
We have just added it and should look like as below.



Now double click on it to configure it. The following window will appear.

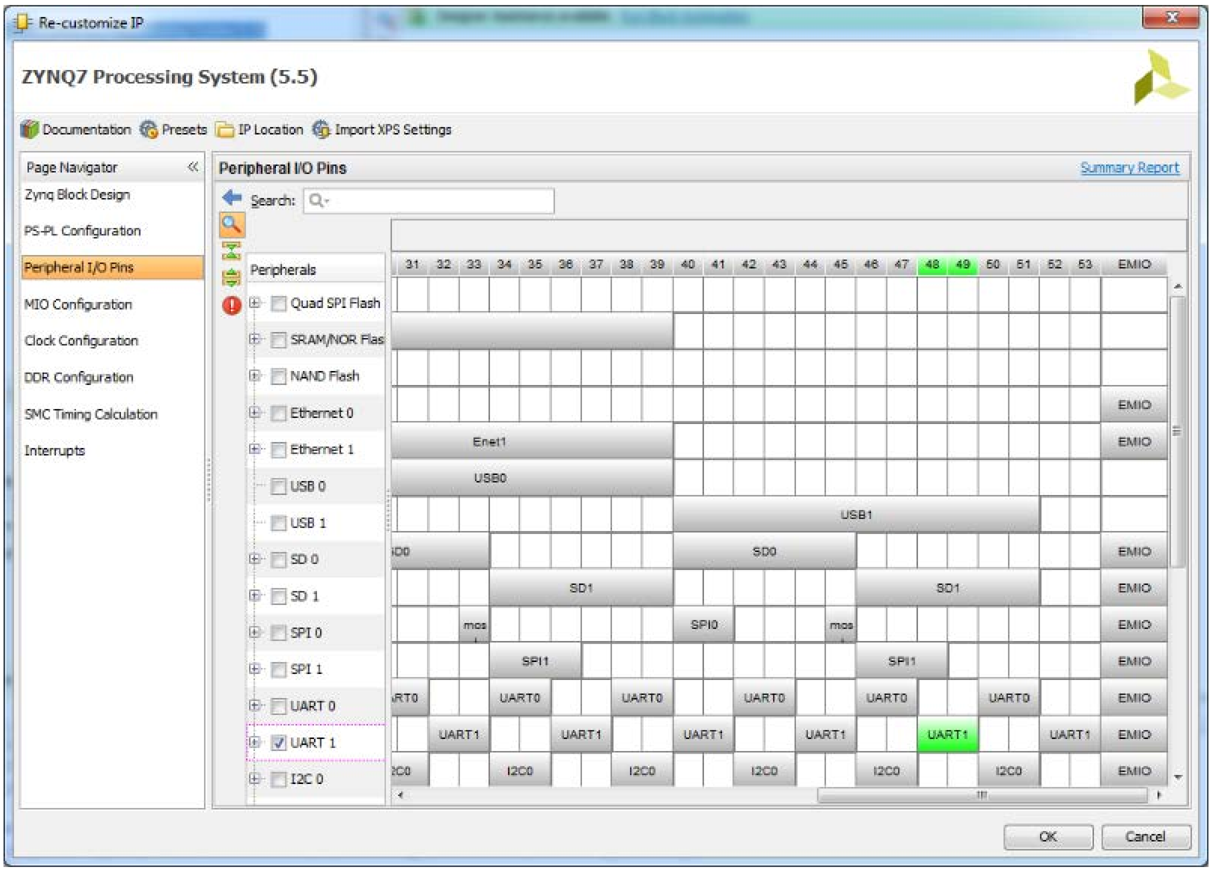


The first screen is just a summary. Now we'll go through all of them revising the settings. Click on PS-PL Configuration. Make sure under General, the UART1 speed is 115200 bps, under General > Enable Clock Resets the FCLK_RESET0_N is unticked and under 'AXI non secure enablement' the M-AXI GP0 interface is disabled.



| Name | Select | Description |
|---|---|---|
| General | | |
| AXI Non Secure Enablement | 0 | Enable AXI Non Secure Transaction |
| GP Master AXI Interface | | |
| M AXI GP0 interface | ☐ | Enables General purpose AXI master interface 0 |
| M AXI GP1 interface | ☐ | Enables General purpose AXI master interface 1 |
| GP Slave AXI Interface | | |
| HP Slave AXI Interface | | |
| ACP Slave AXI Interface | | |
| DMA Controller | | |
| PS-PL Cross Trigger interface | ☐ | Enables PL cross trigger signals to PS and vice-versa |

On the Peripheral I/O pins section, make sure only the UART1 is enabled and click on the UART1 button under column 48-49 to connect it to them.   Change the Bank1 voltage to 1.8V and leave the Bank0 to 3.3V


Re-customize IP

ZYNQ7 Processing System (5.5)

Documentation | Presets | IP Location | Import XPS Settings

Page Navigator: Zynq Block Design, PS-PL Configuration, Peripheral I/O Pins, MIO Configuration, Clock Configuration, DDR Configuration, SMC Timing Calculation, Interrupts

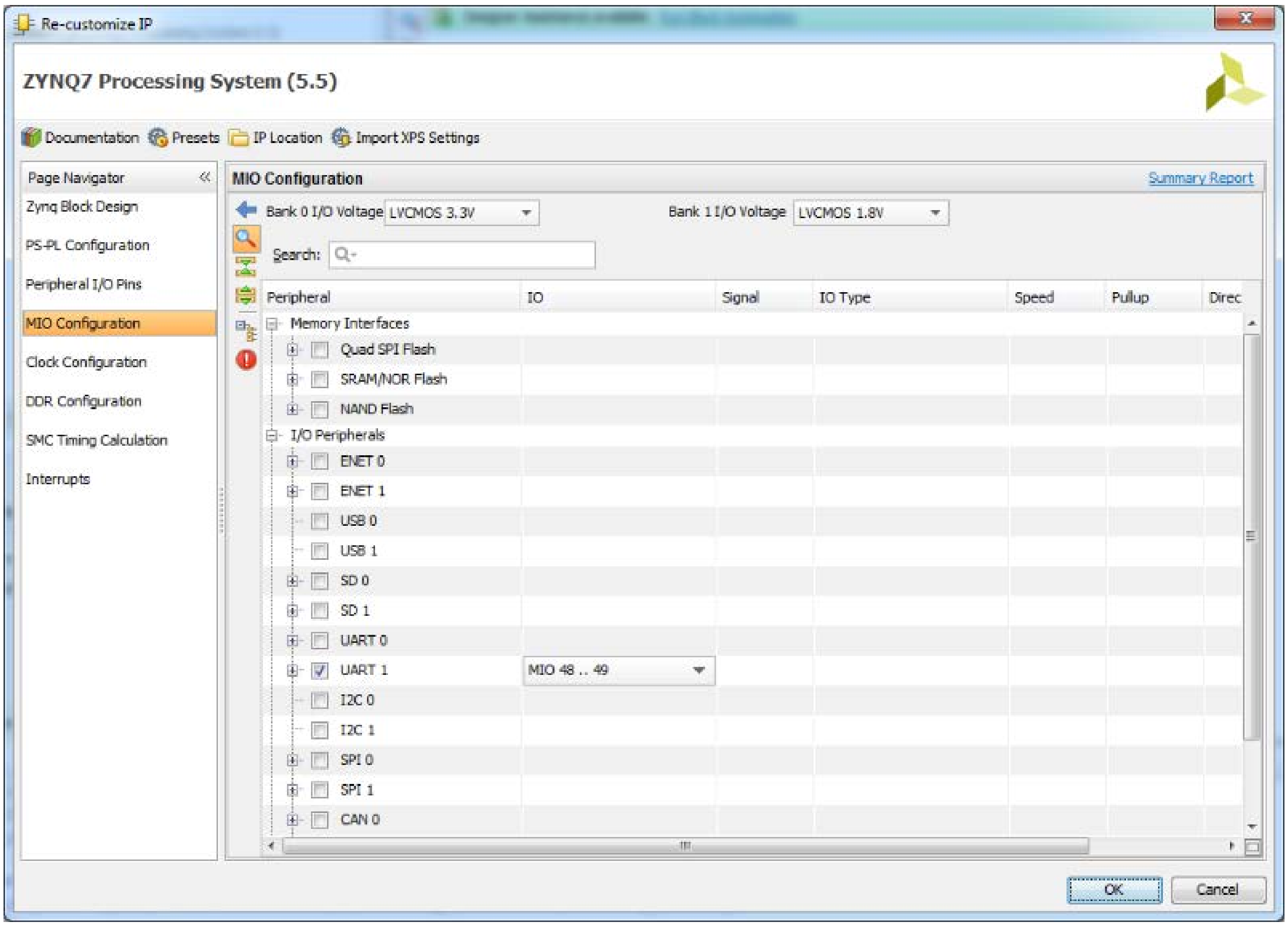
Peripheral I/O Pins — Summary Report

Search:

| Peripherals |
|---|
| Quad SPI Flash |
| SRAM/NOR Flash |
| NAND Flash |
| Ethernet 0 |
| Ethernet 1 |
| USB 0 |
| USB 1 |
| SD 0 |
| SD 1 |
| SPI 0 |
| SPI 1 |
| UART 0 |
| ✓ UART 1 |
| I2C 0 |

OK | Cancel

On the MIO Configuration screen, check all the entries are disabled except UART1 and the Bank voltages are as above.


Re-customize IP

ZYNQ7 Processing System (5.5)

Documentation | Presets | IP Location | Import XPS Settings

Page Navigator: Zynq Block Design, PS-PL Configuration, Peripheral I/O Pins, MIO Configuration, Clock Configuration, DDR Configuration, SMC Timing Calculation, Interrupts

MIO Configuration — Summary Report

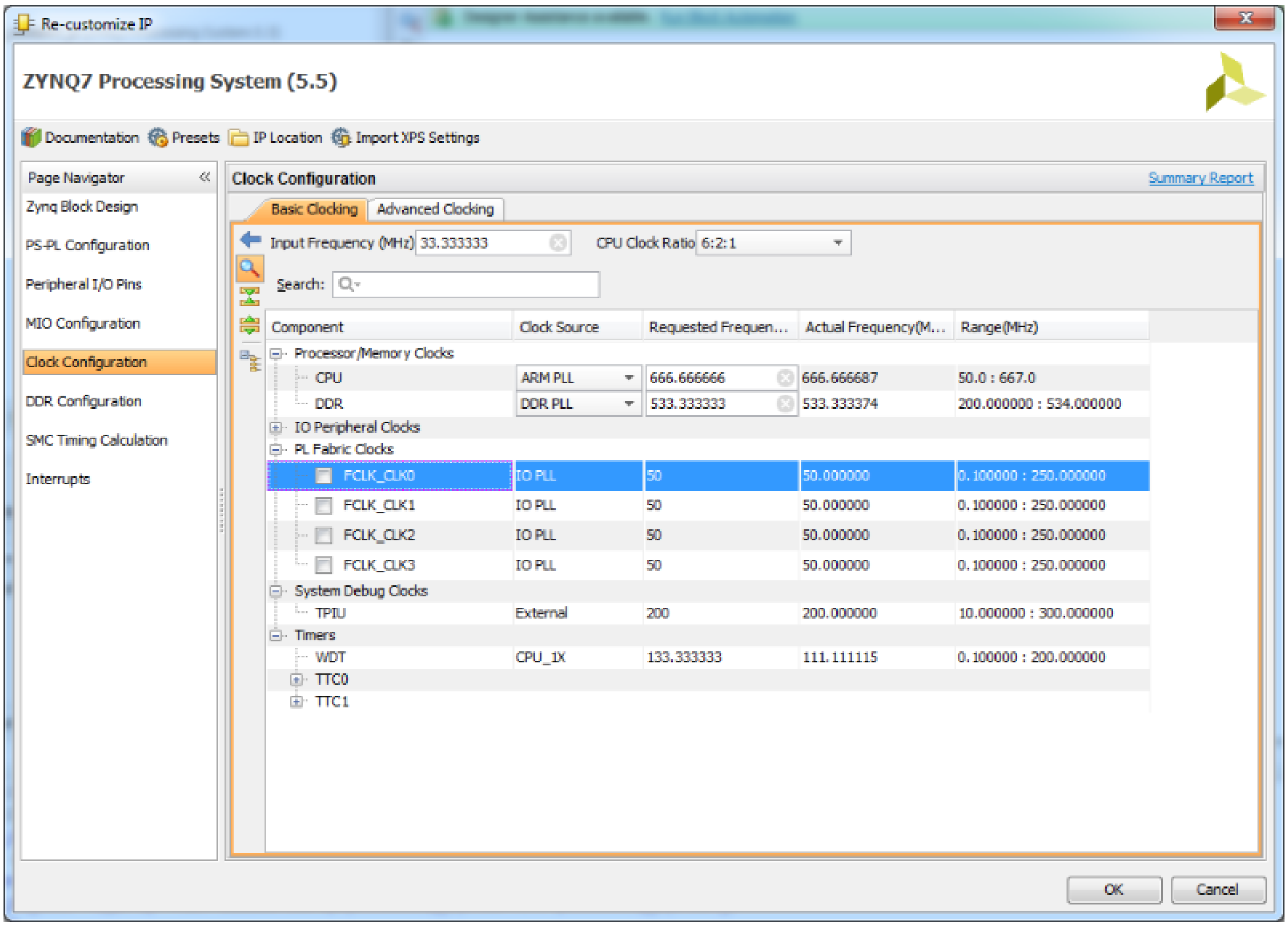
Bank 0 I/O Voltage: LVCMOS 3.3V    Bank 1 I/O Voltage: LVCMOS 1.8V

Search:

| Peripheral | IO | Signal | IO Type | Speed | Pullup | Direc |
|---|---|---|---|---|---|---|
| Memory Interfaces | | | | | | |
| Quad SPI Flash | | | | | | |
| SRAM/NOR Flash | | | | | | |
| NAND Flash | | | | | | |
| I/O Peripherals | | | | | | |
| ENET 0 | | | | | | |
| ENET 1 | | | | | | |
| USB 0 | | | | | | |
| USB 1 | | | | | | |
| SD 0 | | | | | | |
| SD 1 | | | | | | |
| UART 0 | | | | | | |
| ✓ UART 1 | MIO 48 .. 49 | | | | | |
| I2C 0 | | | | | | |
| I2C 1 | | | | | | |
| SPI 0 | | | | | | |
| SPI 1 | | | | | | |
| CAN 0 | | | | | | |

OK | Cancel

On Clock Configuration, make sure ther input frequency is 33.333 MHz, the CPU clock source is ARM PLL and disable the FCLK_CLK0


Re-customize IP

ZYNQ7 Processing System (5.5)

Documentation  Presets  IP Location  Import XPS Settings

Page Navigator: Zynq Block Design, PS-PL Configuration, Peripheral I/O Pins, MIO Configuration, Clock Configuration, DDR Configuration, SMC Timing Calculation, Interrupts

Clock Configuration — Summary Report

Basic Clocking | Advanced Clocking

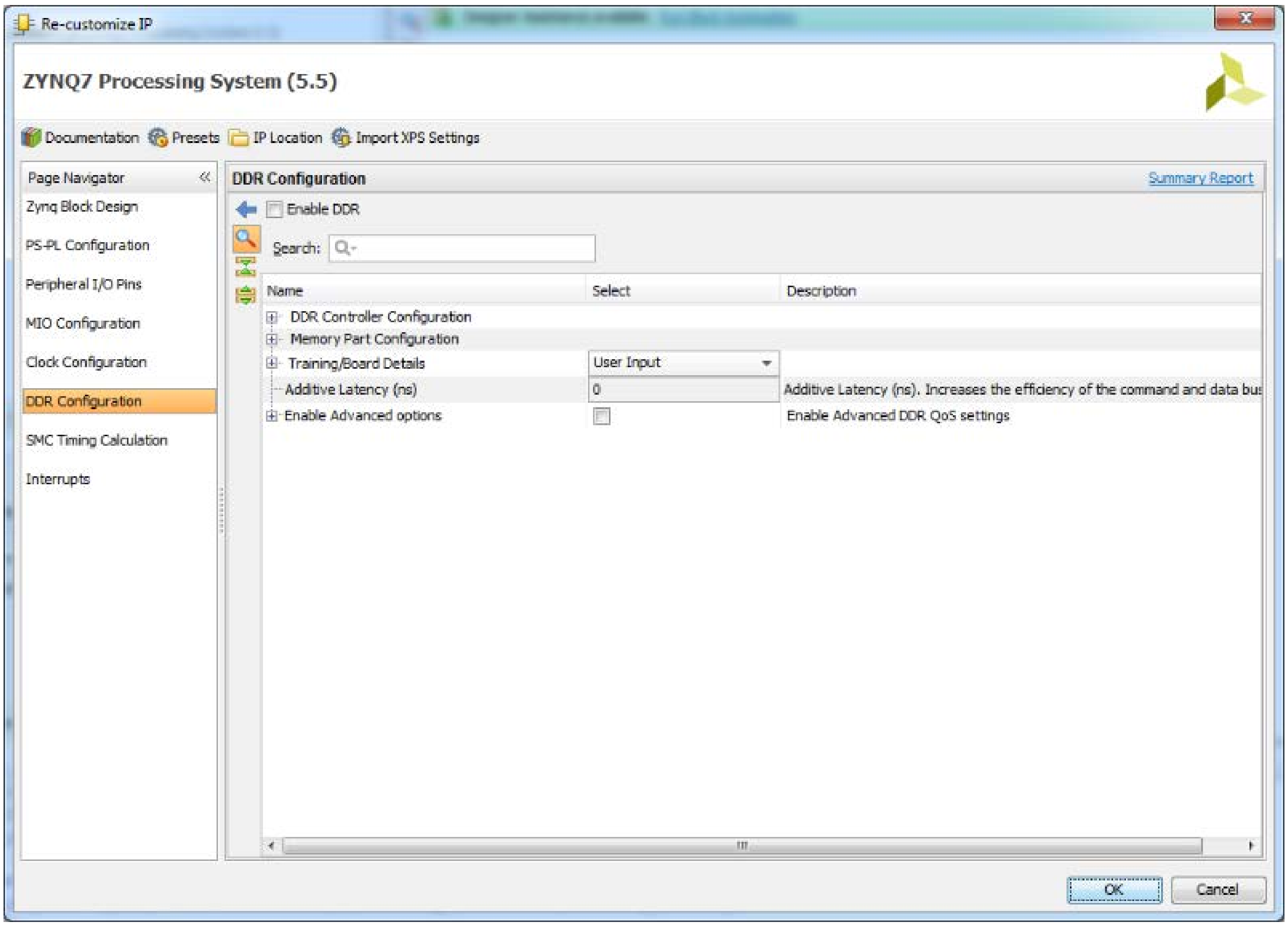
Input Frequency (MHz) 33.333333    CPU Clock Ratio 6:2:1

Search:

| Component | Clock Source | Requested Frequen... | Actual Frequency(M... | Range(MHz) |
|---|---|---|---|---|
| Processor/Memory Clocks | | | | |
| CPU | ARM PLL | 666.666666 | 666.666687 | 50.0 : 667.0 |
| DDR | DDR PLL | 533.333333 | 533.333374 | 200.000000 : 534.000000 |
| IO Peripheral Clocks | | | | |
| PL Fabric Clocks | | | | |
| ☐ FCLK_CLK0 | IO PLL | 50 | 50.000000 | 0.100000 : 250.000000 |
| ☐ FCLK_CLK1 | IO PLL | 50 | 50.000000 | 0.100000 : 250.000000 |
| ☐ FCLK_CLK2 | IO PLL | 50 | 50.000000 | 0.100000 : 250.000000 |
| ☐ FCLK_CLK3 | IO PLL | 50 | 50.000000 | 0.100000 : 250.000000 |
| System Debug Clocks | | | | |
| TPIU | External | 200 | 200.000000 | 10.000000 : 300.000000 |
| Timers | | | | |
| WDT | CPU_1X | 133.333333 | 111.111115 | 0.100000 : 200.000000 |
| TTC0 | | | | |
| TTC1 | | | | |

OK  Cancel

On DDR configuration just untick the Enable DDR option as we won't use it for bare-metal applications.


Re-customize IP

ZYNQ7 Processing System (5.5)

Documentation | Presets | IP Location | Import XPS Settings

Page Navigator: Zynq Block Design, PS-PL Configuration, Peripheral I/O Pins, MIO Configuration, Clock Configuration, DDR Configuration, SMC Timing Calculation, Interrupts

DDR Configuration | Summary Report

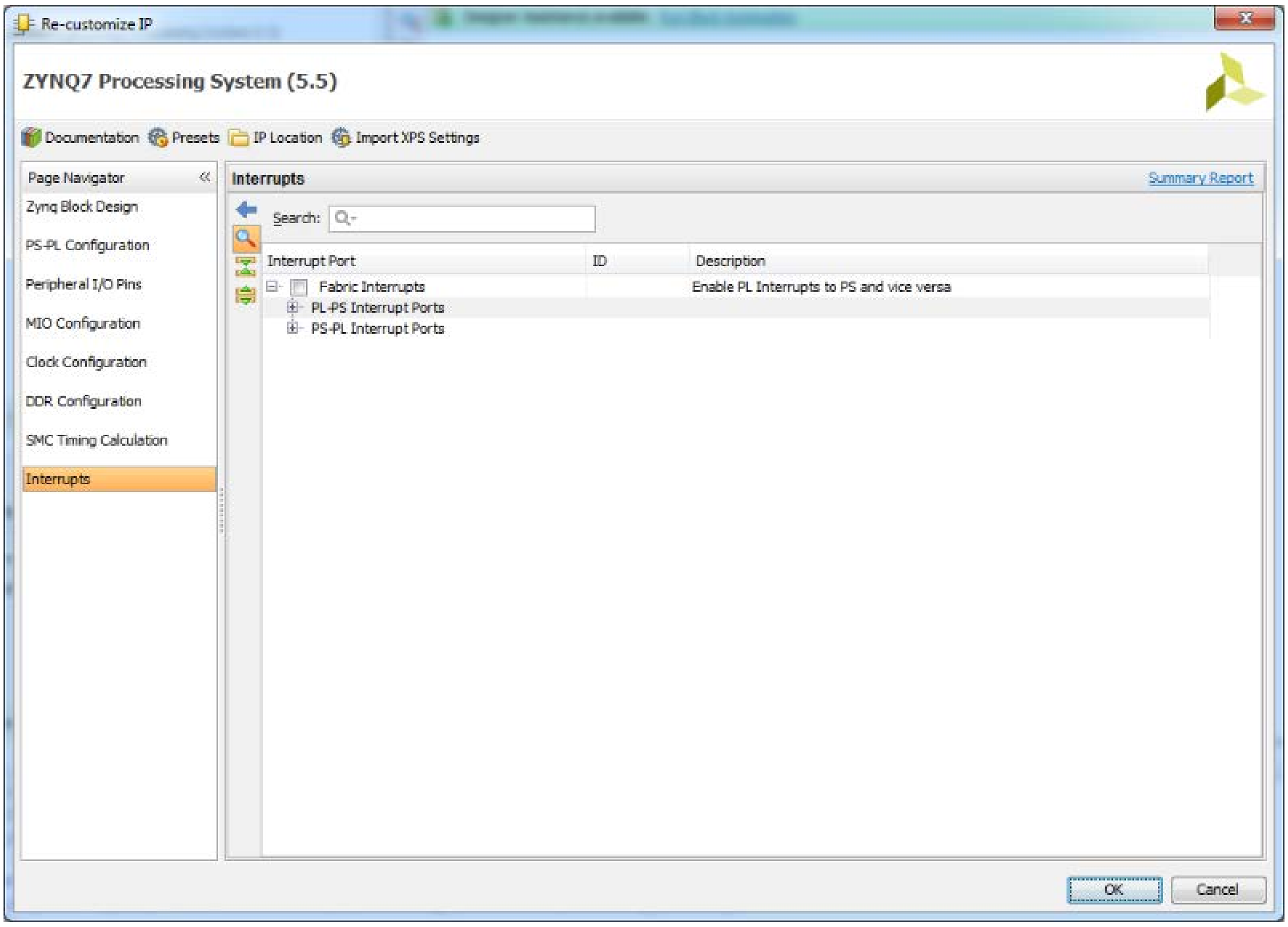
☐ Enable DDR

Search:

| Name | Select | Description |
|---|---|---|
| ⊞ DDR Controller Configuration | | |
| ⊞ Memory Part Configuration | | |
| ⊞ Training/Board Details | User Input | |
| Additive Latency (ns) | 0 | Additive Latency (ns). Increases the efficiency of the command and data bus |
| ⊞ Enable Advanced options | ☐ | Enable Advanced DDR QoS settings |

OK | Cancel

Skip the SMC Timing Calculation and on the Interrupts screen ensure all are disabled. Then click OK.


Re-customize IP

ZYNQ7 Processing System (5.5)

Documentation | Presets | IP Location | Import XPS Settings

Page Navigator: Zynq Block Design, PS-PL Configuration, Peripheral I/O Pins, MIO Configuration, Clock Configuration, DDR Configuration, SMC Timing Calculation, Interrupts

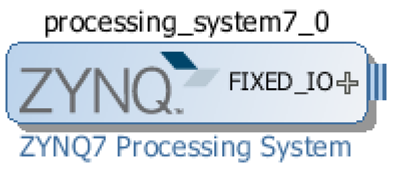
Interrupts | Summary Report

Search:

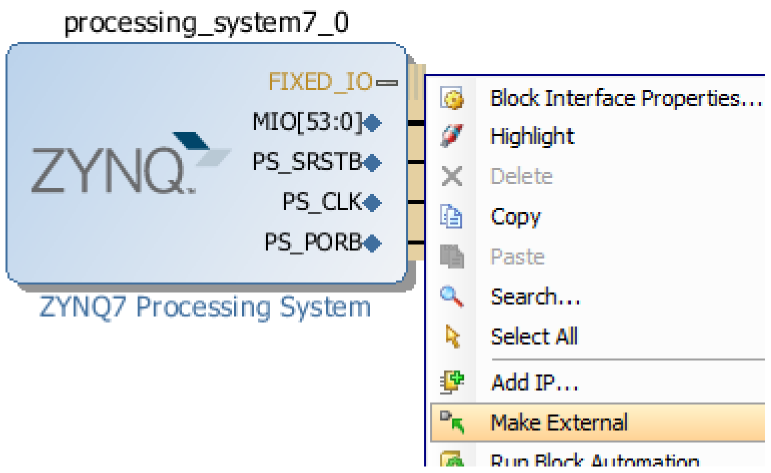
| Interrupt Port | ID | Description |
|---|---|---|
| ⊟ ☐ Fabric Interrupts | | Enable PL Interrupts to PS and vice versa |
| ⊞ PL-PS Interrupt Ports | | |
| ⊞ PS-PL Interrupt Ports | | |

OK | Cancel

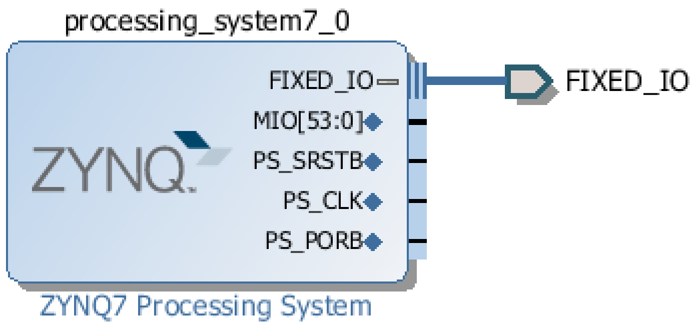
It should now look like this:



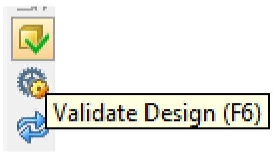
Click on the '+' sign to expand the signals and go over the three vertical lines next to FIXED_IO, right click and select 'Make External'



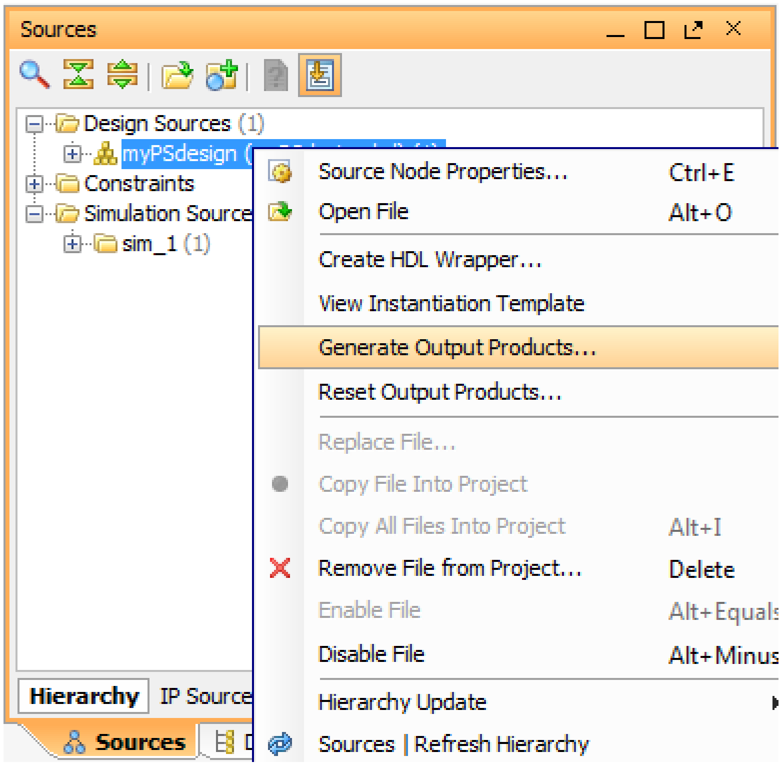
We'll end up with this:



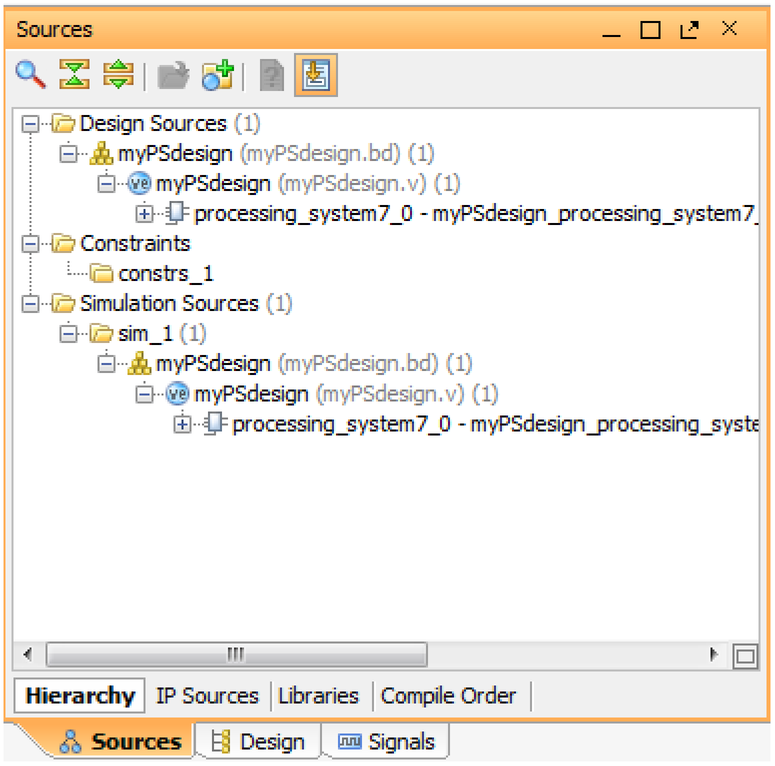
To validate the design (check for errors) hit F6 or click on the validate icon. You should get a message saying there are no critical warnings.



On the Sources view, right click on the design and select 'Generate Output Products'

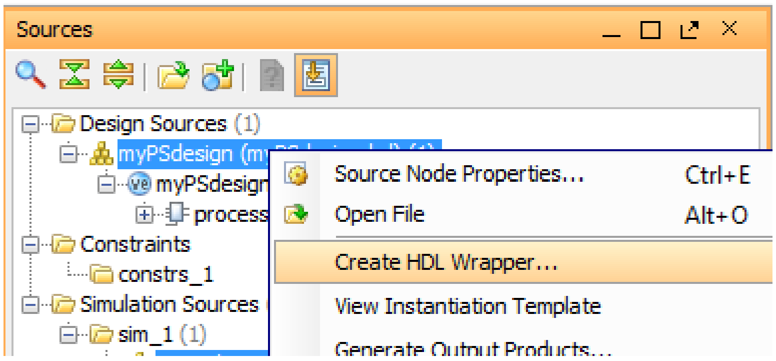


Leave the Synthesis Options 'Global' and click Generate.  After completion, the hierarchy in the Sources panel is updated:

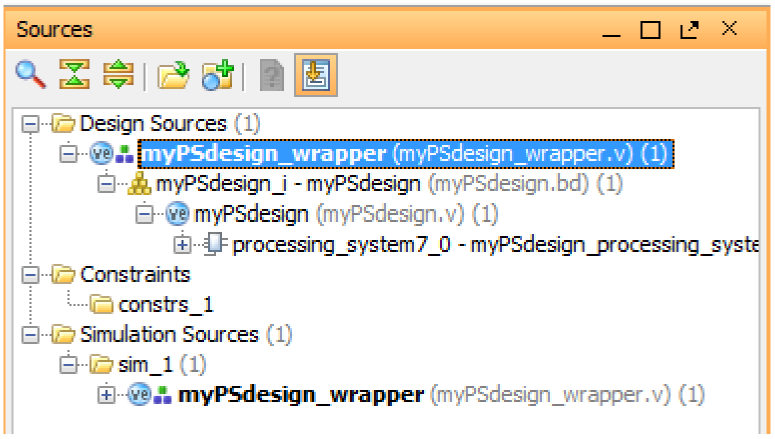

Sources
Design Sources (1)
myPSdesign (myPSdesign.bd) (1)
myPSdesign (myPSdesign.v) (1)
processing_system7_0 - myPSdesign_processing_system7_
Constraints
constrs_1
Simulation Sources (1)
sim_1 (1)
myPSdesign (myPSdesign.bd) (1)
myPSdesign (myPSdesign.v) (1)
processing_system7_0 - myPSdesign_processing_syste
Hierarchy
IP Sources
Libraries
Compile Order
Sources
Design
Signals

### 2. Wrap up the design with HDL and generate the bitstream

Right click on the design in the sources panel and click on Create HDL wrapper.

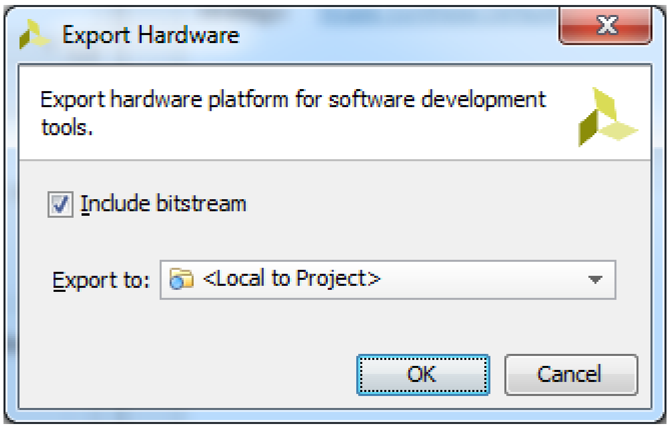


On the pop up window, select the option 'Let Vivado manage wrapper' and click OK. We have a now a Verilog top file that instantiates our design. If you generated a VHDL one instead because of your project settings, don't worry it will do anyways.


Sources
Design Sources (1)
myPSdesign_wrapper (myPSdesign_wrapper.v) (1)
myPSdesign_i - myPSdesign (myPSdesign.bd) (1)
myPSdesign (myPSdesign.v) (1)
processing_system7_0 - myPSdesign_processing_syste
Constraints
constrs_1
Simulation Sources (1)
sim_1 (1)
myPSdesign_wrapper (myPSdesign_wrapper.v) (1)

Now you can either click on Synthesize, Implement and Generate Bitstream or straight away the last option. It will take a while. Click on the 'Project Summary' button to see the results. You should get a few harmless warnings and the message that the bitstream was generated.

Now click on File > Export > Export Hardware and in the pop up window click on the 'Include bitstream' option. Leave the option <Local to Project>



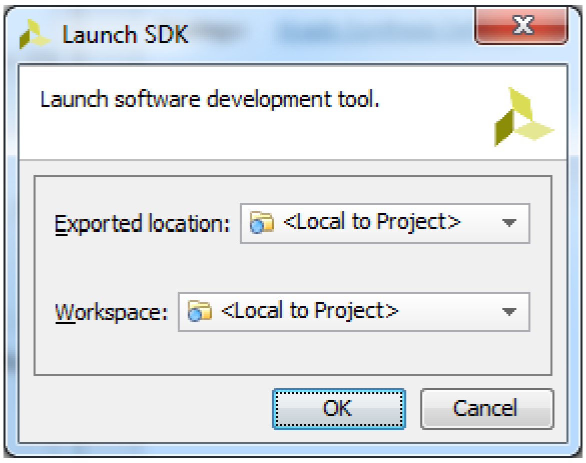
Export Hardware

Export hardware platform for software development tools.

[✓] Include bitstream

Export to: <Local to Project>

Now click on File > Launch SDK and leave the options as below. This will load the exported files from Vivado.


Launch SDK

Launch software development tool.

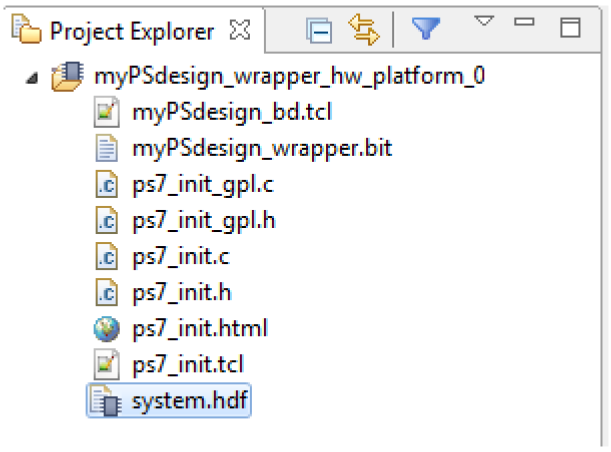
Exported location: <Local to Project>

Workspace: <Local to Project>

After SDK is open you can close Vivado or leave it open.

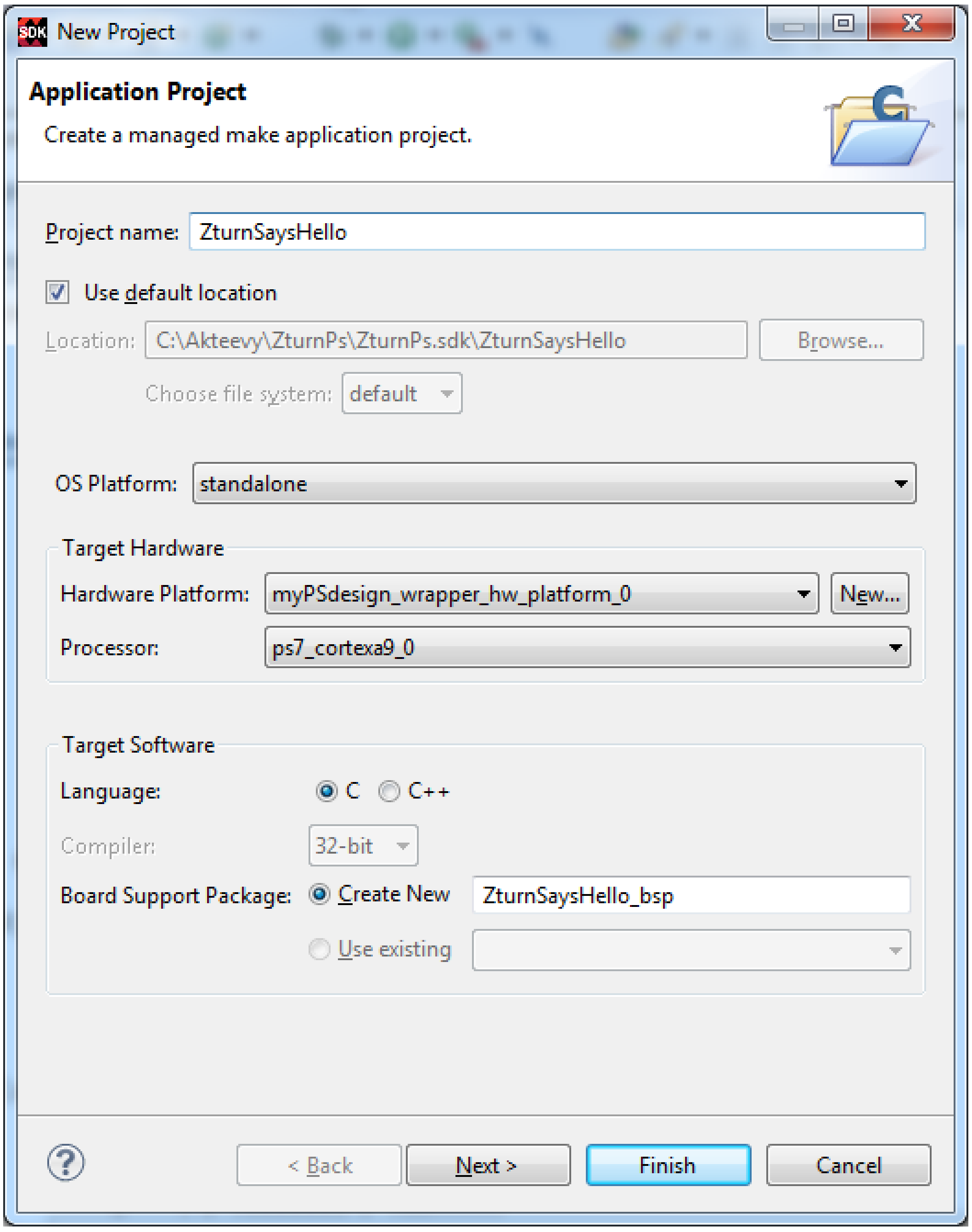
### 3. Start the software application in SDK

In SDK we should have on the left panel the hardware platform specification and one of its files (system.hdf) open.


Project Explorer

```
myPSdesign_wrapper_hw_platform_0
    myPSdesign_bd.tcl
    myPSdesign_wrapper.bit
    ps7_init_gpl.c
    ps7_init_gpl.h
    ps7_init.c
    ps7_init.h
    ps7_init.html
    ps7_init.tcl
    system.hdf
```

Now click on File > New > Application Project. On the pop up window type*ZturnSaysHello* as project name and check the other options are as below. Note the Board Supprt Package name has been automatically generated for you after the project name.



New Project

**Application Project**

Create a managed make application project.

Project name: ZturnSaysHello

[✓] Use default location

Location: C:\Akteevy\ZturnPs\ZturnPs.sdk\ZturnSaysHello

Choose file system: default

OS Platform: standalone

Target Hardware

Hardware Platform: myPSdesign_wrapper_hw_platform_0

Processor: ps7_cortexa9_0

Target Software

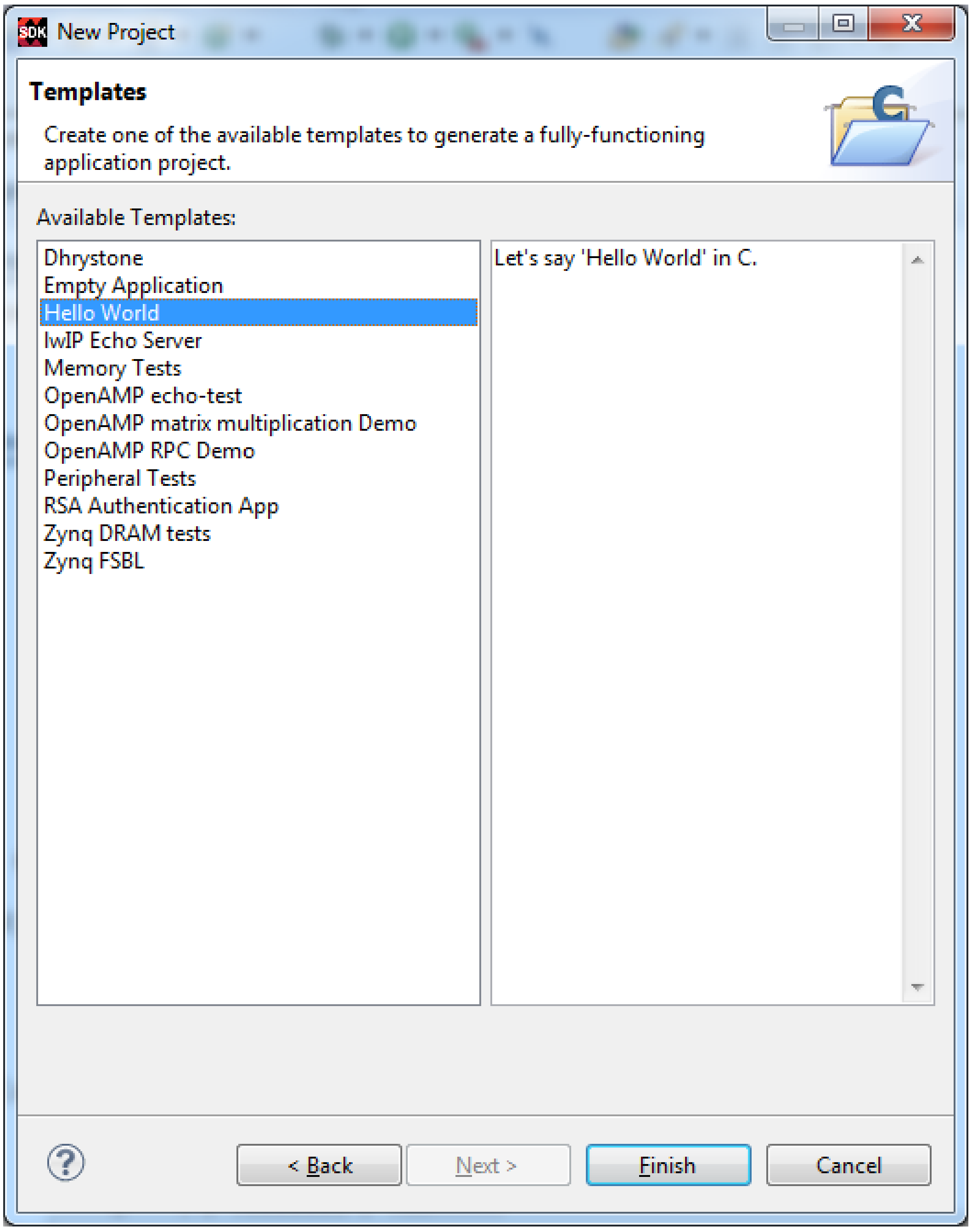
Language: [✓ C] C++

Compiler: 32-bit

Board Support Package: [✓ Create New] ZturnSaysHello_bsp

Use existing

Click Next, select the HelloWorld template and click Finish



**Templates**

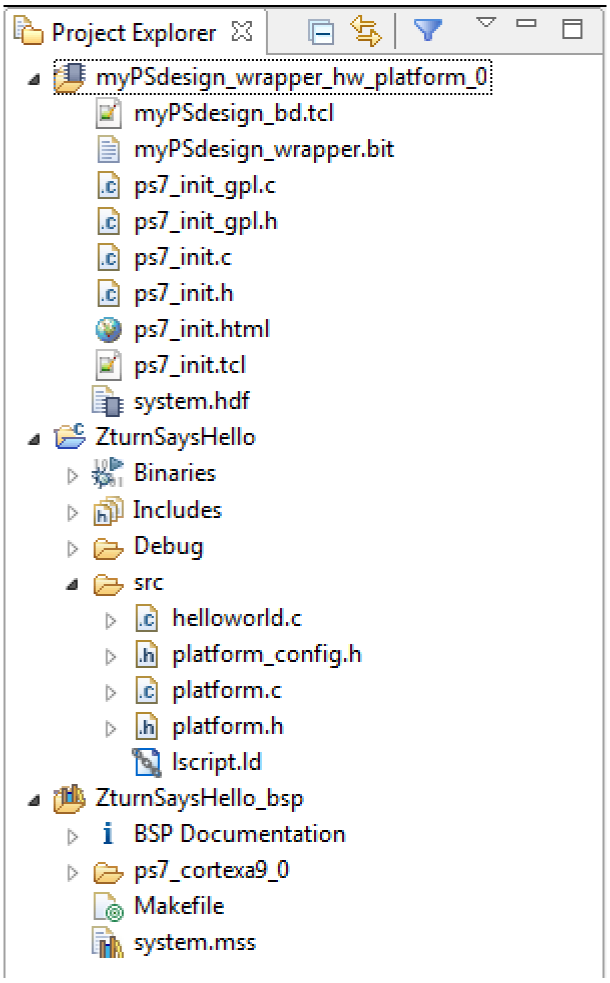
Create one of the available templates to generate a fully-functioning application project.

Available Templates:

- Dhrystone
- Empty Application
- Hello World
- lwIP Echo Server
- Memory Tests
- OpenAMP echo-test
- OpenAMP matrix multiplication Demo
- OpenAMP RPC Demo
- Peripheral Tests
- RSA Authentication App
- Zynq DRAM tests
- Zynq FSBL

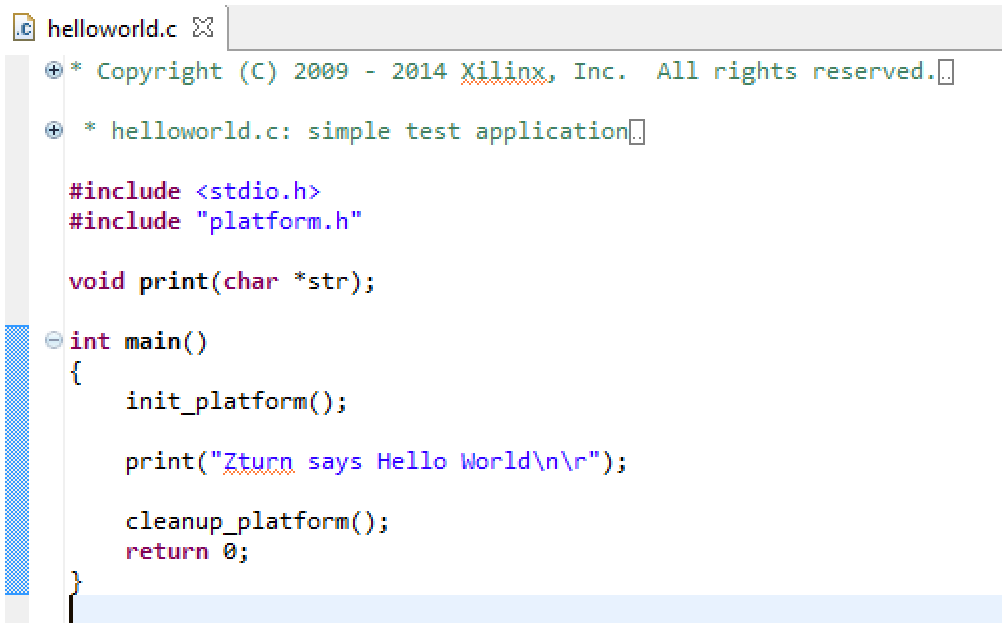
Let's say 'Hello World' in C.

After that, two additional folders are created, the application (ZturnSaysHello) with our .c source files and the BSP (Board Support Package).



```
Project Explorer
◢ myPSdesign_wrapper_hw_platform_0
    myPSdesign_bd.tcl
    myPSdesign_wrapper.bit
    ps7_init_gpl.c
    ps7_init_gpl.h
    ps7_init.c
    ps7_init.h
    ps7_init.html
    ps7_init.tcl
    system.hdf
◢ ZturnSaysHello
  ▷ Binaries
  ▷ Includes
  ▷ Debug
  ◢ src
    ▷ helloworld.c
    ▷ platform_config.h
    ▷ platform.c
    ▷ platform.h
      lscript.ld
◢ ZturnSaysHello_bsp
  ▷ BSP Documentation
  ▷ ps7_cortexa9_0
    Makefile
    system.mss
```

Double click the helloworld.c file to open it and change the message to be printed to make it a bit more personal (or just skip that if you prefer)



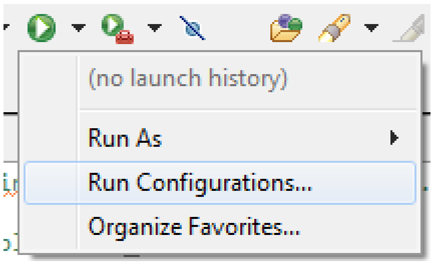
```
helloworld.c
* Copyright (C) 2009 - 2014 Xilinx, Inc.  All rights reserved.

 * helloworld.c: simple test application

#include <stdio.h>
#include "platform.h"

void print(char *str);

int main()
{
    init_platform();

    print("Zturn says Hello World\n\r");

    cleanup_platform();
    return 0;
}
```

Click on the down arrow next to the build icon on the top and select the Release option (the Debug will also do). It should automatically build the project (you will notice a window with a progress bar for a second). If not, click again on the 'hammer' button.

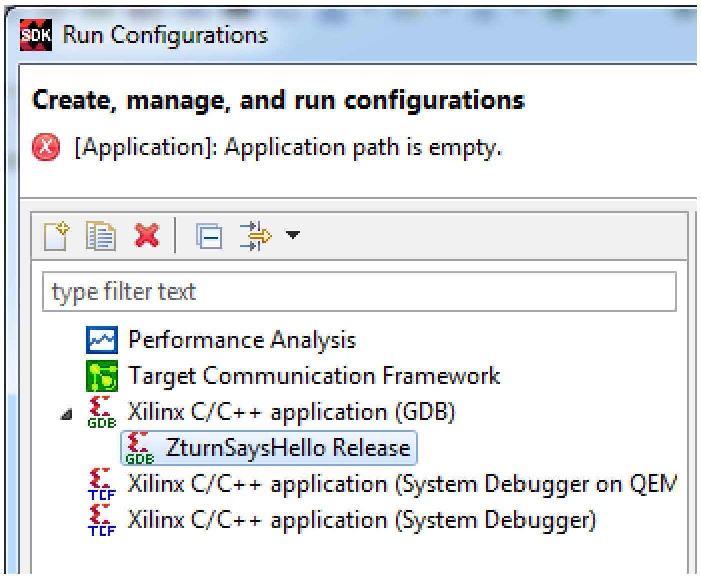
## 4. Test run the application

We are now about to run it! Power the Zturn board either with the USB cable or external supply (the blue LED should be ON) and connect the Xilinx programming cable to both the computer and the JTAG connector. The LED on the programmer should change from red to green.

Now click on the down arrow at the right of the Run button (like a play sign on a green circle) and select 'Run Configurations'



Double click on the 'Xilinx C/C++ application (GDB) and an entry will appear below wih the name taken from the project name.

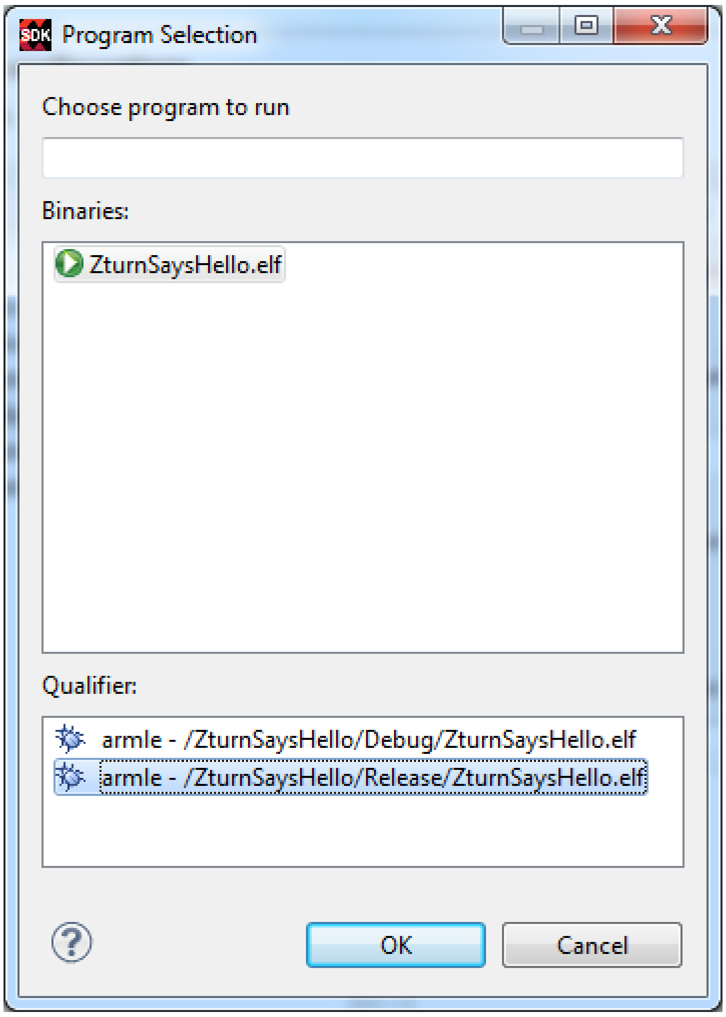

Run Configurations

Create, manage, and run configurations

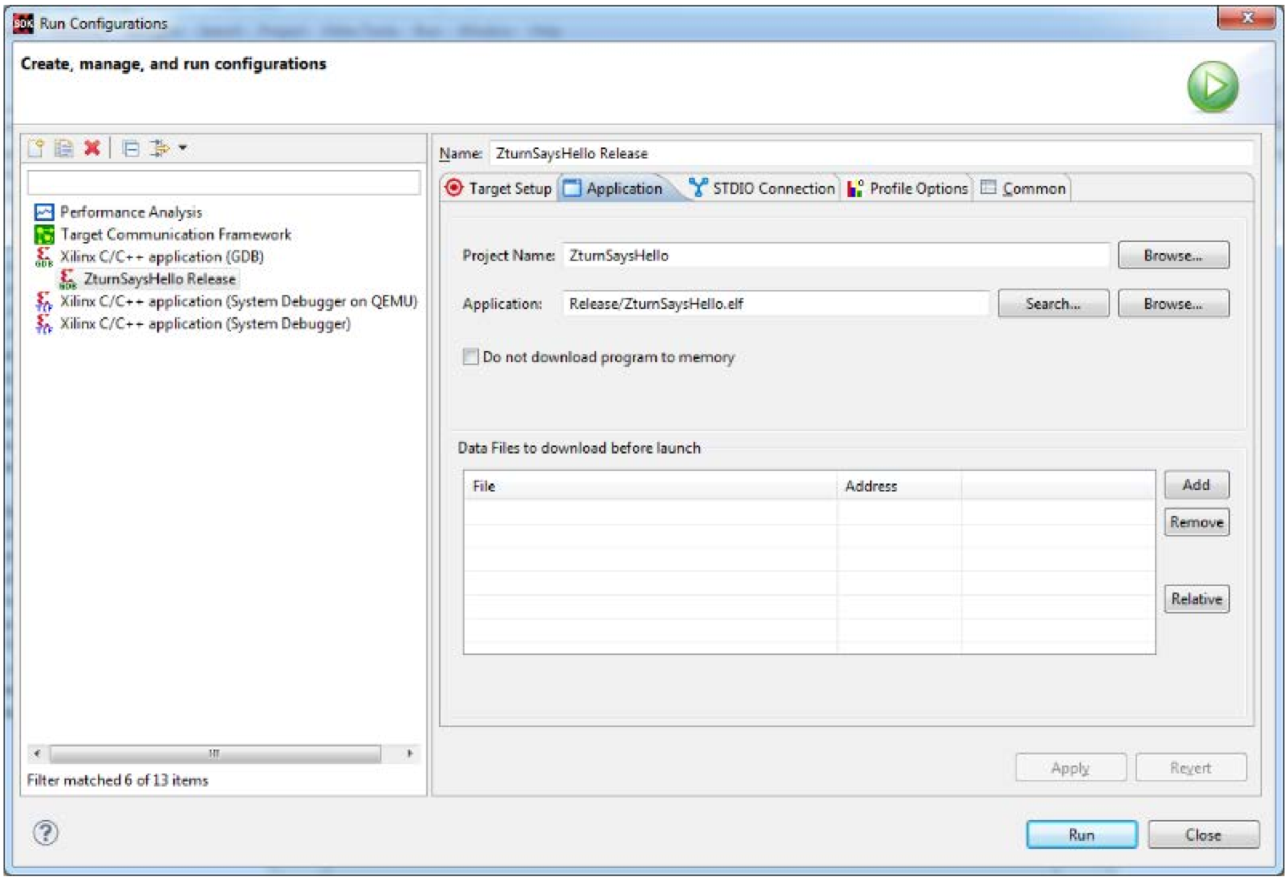
[Application]: Application path is empty.

type filter text
Performance Analysis
Target Communication Framework
Xilinx C/C++ application (GDB)
ZturnSaysHello Release
Xilinx C/C++ application (System Debugger on QEM
Xilinx C/C++ application (System Debugger)

On the right, click the 'Application' tab and click the Search button next to the empty field. On the pop up window, select the elf file as below and click OK


Program Selection
Choose program to run
Binaries:
ZturnSaysHello.elf
Qualifier:
armle - /ZturnSaysHello/Debug/ZturnSaysHello.elf
armle - /ZturnSaysHello/Release/ZturnSaysHello.elf
OK
Cancel

Back to the previous window click the Apply button and Close (not Run)


Run Configurations
Create, manage, and run configurations
Performance Analysis
Target Communication Framework
Xilinx C/C++ application (GDB)
ZturnSaysHello Release
Xilinx C/C++ application (System Debugger on QEMU)
Xilinx C/C++ application (System Debugger)
Filter matched 6 of 13 items
Target Setup
Application
STDIO Connection
Profile Options
Common
Browse...
Search...
Browse...
Add
Remove
Relative
Apply
Revert
Run
Close

Name: ZturnSaysHello Release

Project Name: ZturnSaysHello

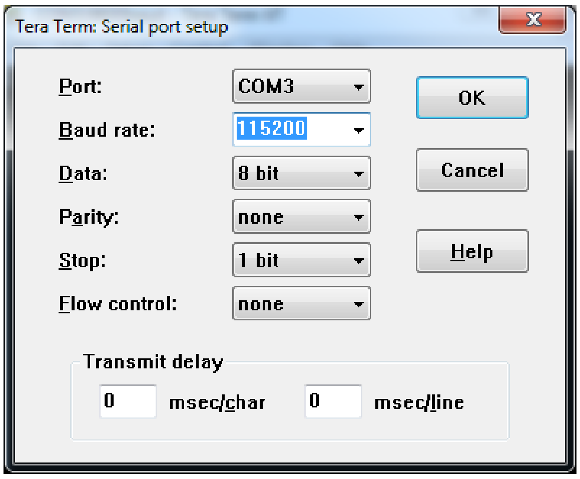
Application: Release/ZturnSaysHello.elf

Do not download program to memory

Data Files to download before launch

| File | Address |
|---|---|

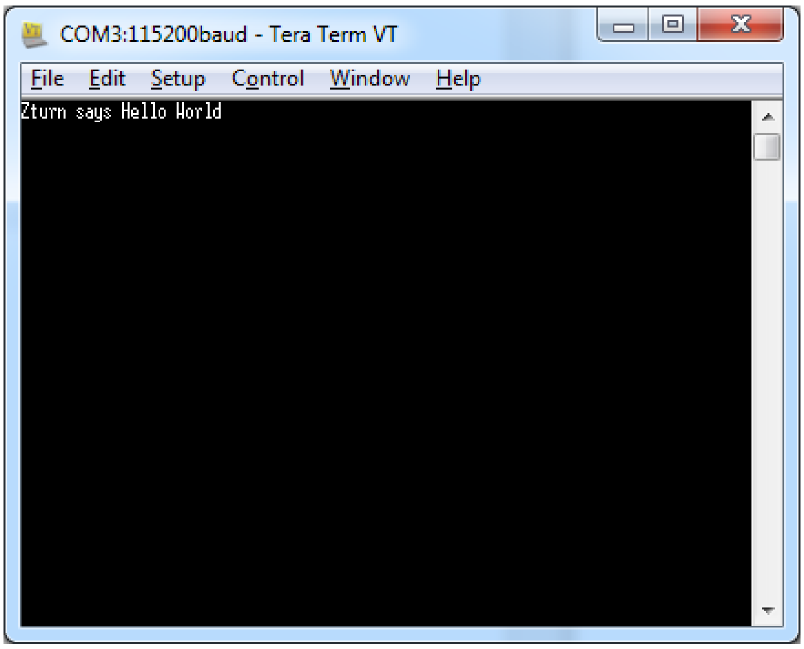
Now let's connect to the serial port to hopefully receive the message. Open your favourite console application (I use Tera Term) and select the por corresponding to the Zturn (in my case is 3 but that may vary). Select the speed as 115200 bps and leave other settings as below.


Tera Term: Serial port setup

Port: COM3
Baud rate: 115200
Data: 8 bit
Parity: none
Stop: 1 bit
Flow control: none

Transmit delay
0 msec/char 0 msec/line

Once the port is open, go back to SDK and click the Run button on the top bar. A window will pop up warning you the FPGA has not been configured. As we are now only using the processor that's fine, click OK and after a second, hopefully, you get this:


COM3:115200baud - Tera Term VT

```
Zturn says Hello World
```

## Congratulations!

If you have got the final result, you have learnt:

- How to create and configure a basic Processing System (PS) for the Zynq
- How to wrap it up into a VHDL/ Verilog file
- How to implement it and generate the bitstream
- Export the bitstream to the Software Development Kit (SDK)
- Create a software application from a template
- Make a Run configuration to run the app on the PS
- Run your app and connect to it via a Com port