Getting Started with Vivado and Vitis for Baremetal Software Projects

Overview

This guide will work you through the process of setting up a project in Vivado and Vitis. A simple hardware design including a processor with several AXI GPIO () peripherals connected to buttons and LEDs will be created. This design will then be exported to the Vitis IDE, and a baremetal software project will be created and run which polls the buttons and writes to the LEDs.

Note: *Screenshots presented in this guide may not have been taken with your version of the tools. The workflow presented here has been verified in Vivado and Vitis 2020.1.*

Inventory

- A Digilent FPGA Development Board
  - USB Programming cables, USB UART cables, and Power Supply, as required by the board.
- Vivado and Vitis installations
  - See Installing Vivado, Vitis, and Digilent Board Files (https://digilent.com/reference/programmable-logic/guides/installing-vivado-and-vitis) for instructions on how to install these tools.
  - You also need the board files for your board. This guide is intended for use with the board files available from Digilent's vivado-library repo on Github. You can get these files using the process described in the Installing Vivado, Vitis, and Digilent Board Files (https://digilent.com/reference/programmable-logic/guides/installing-vivado-and-vitis) guide. These files will also be available through the Vivado application itself in versions 2022.1 and newer.

Note: *If you are using a version of Vivado that includes Xilinx SDK (2019.1 or older), check out Getting Started with Vivado IP Integrator and Xilinx SDK (https://digilent.com/reference/vivado/getting-started-with-ipi/2018.2).*

Guide

Launch Vivado

Select the dropdown corresponding to your operating system, below.

Windows

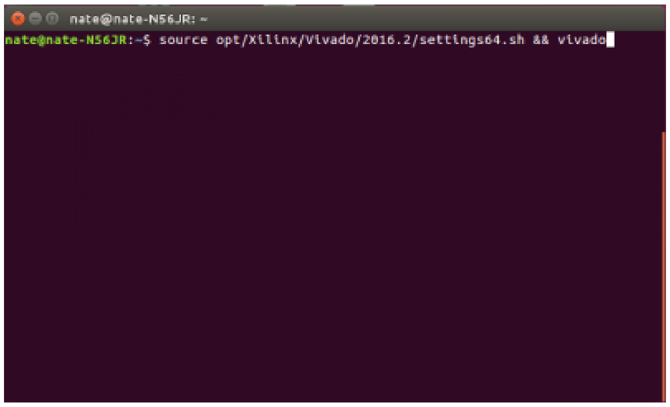
Open Vivado through the start menu or desktop shortcut created during the installation process.

(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/launch-vivado/open-vivado.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

Linux

Open a terminal, and change directory (cd) to a folder where log files for your Vivado session can be placed, then run the following commands:

```
source <install_path>/Vivado/<version>/settings64.sh
vivado
```



```
nate@nate-N56JR: ~
nate@nate-N56JR:~$ source opt/Xilinx/Vivado/2016.2/settings64.sh && vivado
```

(https://digilent.com/reference/_detail/vivado/getting_started/linux_start_vivado.
id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

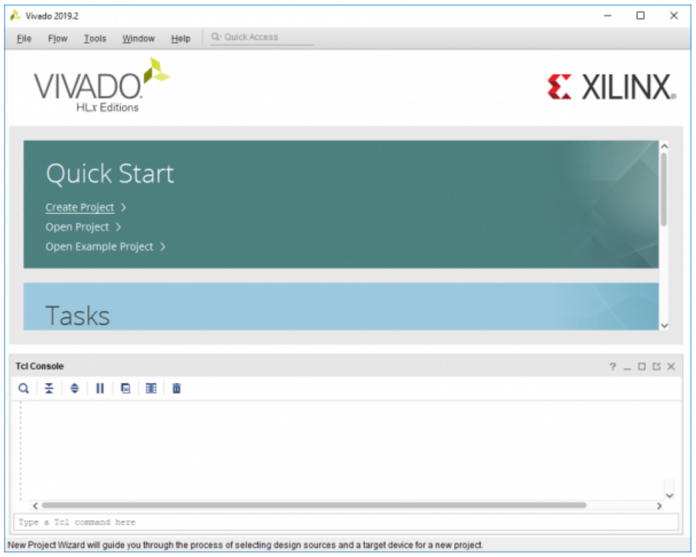
Create a Vivado Project

In Vivado's welcome screen, several options are presented:

- **Create Project**: Opens a wizard used to begin creating a Vivado project from scratch, which will be used here.
- **Open Project**: Can be used to open a Vivado project (defined by an XPR file) that has been previously created or downloaded from the internet.
- **Open Hardware Manager**: Can be used to program an FPGA development board with a bitstream, without opening the associated project.

Note: *Various other options are available, but are not described here.*

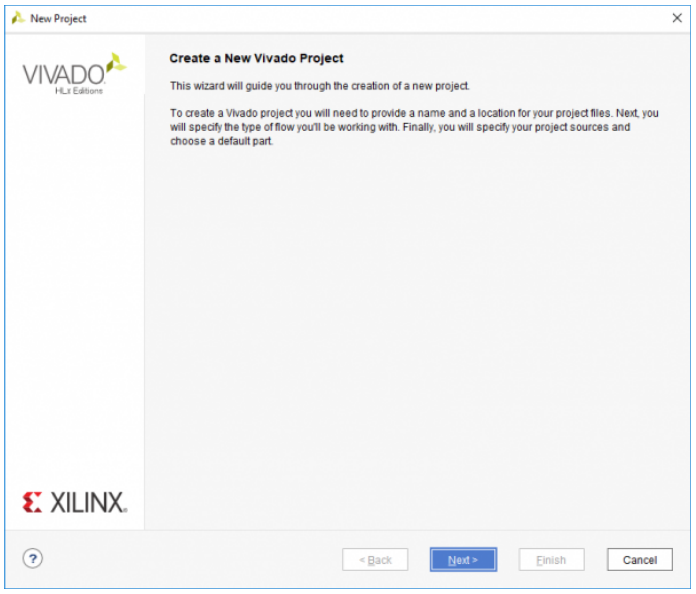
For the purposes of this guide, click the **Create Project** button.



New Project Wizard will guide you through the process of selecting design sources and a target device for a new project.

(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/create-vivado-project/create_project.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

The first page of the New Project wizard summarizes the steps involved in creating a project. Click **Next**.



Create a New Vivado Project

This wizard will guide you through the creation of a new project.

To create a Vivado project you will need to provide a name and a location for your project files. Next, you will specify the type of flow you'll be working with. Finally, you will specify your project sources and choose a default part.

(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/create-vivado-project/new-project-1.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

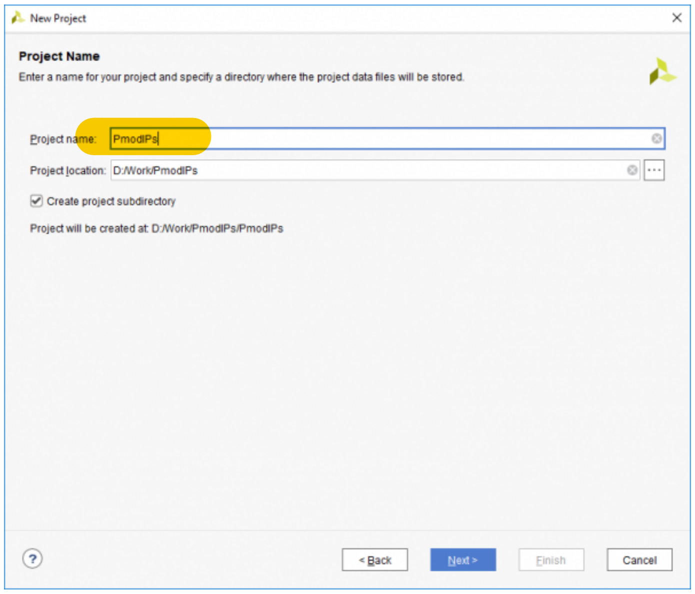
The first step is to set the *name* for the project. Vivado will use this name when generating its folder structure.

Important: *Do NOT use spaces in the project name or location path. This will cause problems with Vivado. Instead use an underscore, a dash, or CamelCase (https://en.wikipedia.org/wiki/CamelCase).*

Pick a memorable *location* in your filesystem to place the project.

Checking the **Create project subdirectory** box will create a new folder in the chosen location to store the project's files. This is recommended.

Click **Next** to continue.


New Project

Project Name

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

Project name: PmodIPs

Project location: D:/Work/PmodIPs

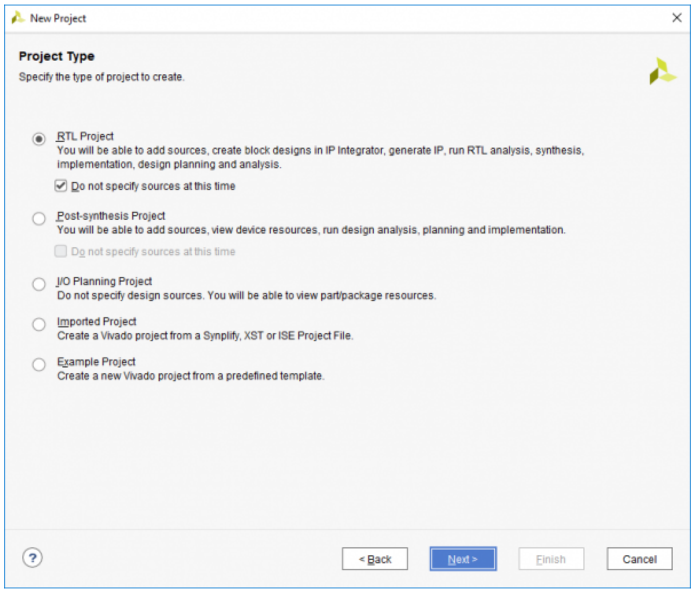
☑ Create project subdirectory

Project will be created at: D:/Work/PmodIPs/PmodIPs

(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/create-vivado-project/new-project-2.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

At the Select Project Type screen, choose **RTL Project** and check the *Do not specify sources at this time* box. Advanced users may want to use the other options on this screen, but they will not be covered here.

Click **Next** to continue.


New Project

Project Type

Specify the type of project to create.

[● RTL Project]
You will be able to add sources, create block designs in IP Integrator, generate IP, run RTL analysis, synthesis, implementation, design planning and analysis.

☑ Do not specify sources at this time

○ Post-synthesis Project
You will be able to add sources, view device resources, run design analysis, planning and implementation.

☐ Do not specify sources at this time

○ I/O Planning Project
Do not specify design sources. You will be able to view part/package resources.

○ Imported Project
Create a Vivado project from a Synplify, XST or ISE Project File.

○ Example Project
Create a new Vivado project from a predefined template.

(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/create-vivado-project/new-project-3.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

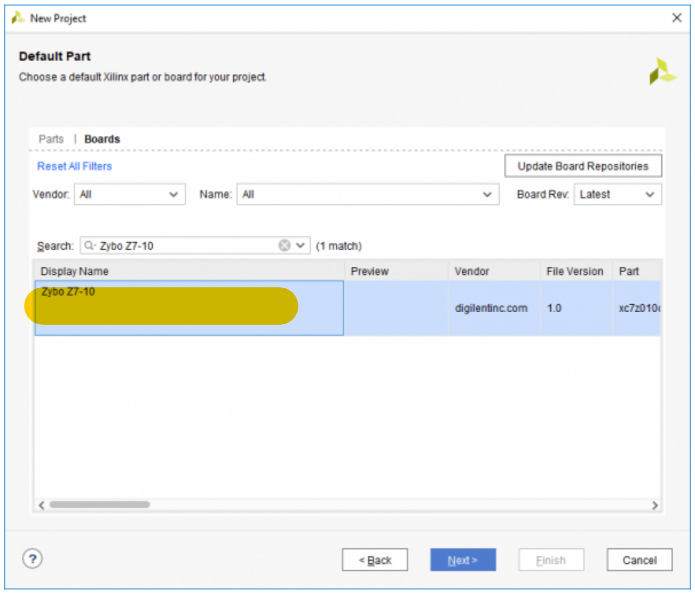
Next, a part or a board must be chosen for the project to target. The project will only be usable with the chosen device (though the selection can later be changed through the project's Settings).

Selecting a board over a part is recommended, as the board files provide additional configuration information for a variety of peripherals and components in a design. Click the **Board** button to open the board tab.

Search for your board and select it from the list.

Important: *If your board does not appear in the list, Digilent's board files have not been installed. Review Installing Vivado, Vitis, and Digilent Board Files (https://digilent.com/reference/programmable-logic/guides/installing-vivado-and-vitis) for instructions on installation of these files.*

Click **Next** to continue.



Default Part
Choose a default Xilinx part or board for your project.

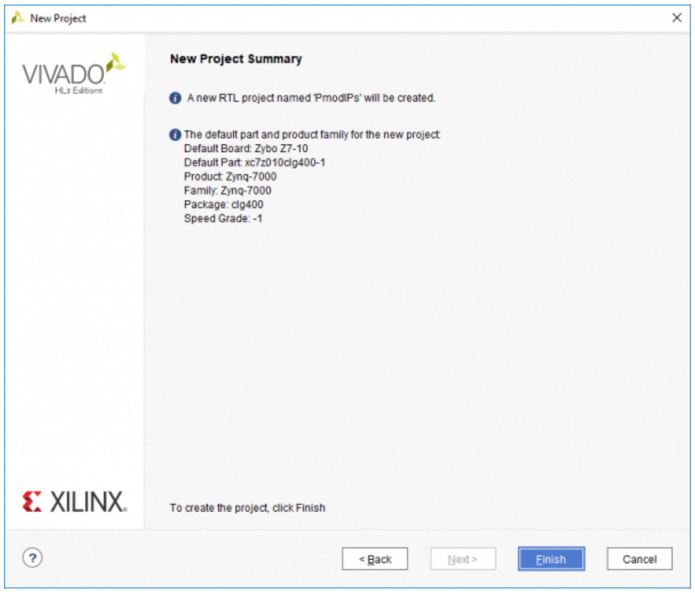
Vendor: All Name: All Board Rev: Latest

Search: Zybo Z7-10 (1 match)

Display Name	Preview	Vendor	File Version	Part
Zybo Z7-10		digilentinc.com	1.0	xc7z010

(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/create-vivado-project/new-project-4.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

The last screen of the New Project wizard summarizes what was chosen in the previous screens. Click **Finish** to open your project.



New Project Summary

A new RTL project named 'PmodIPs' will be created.

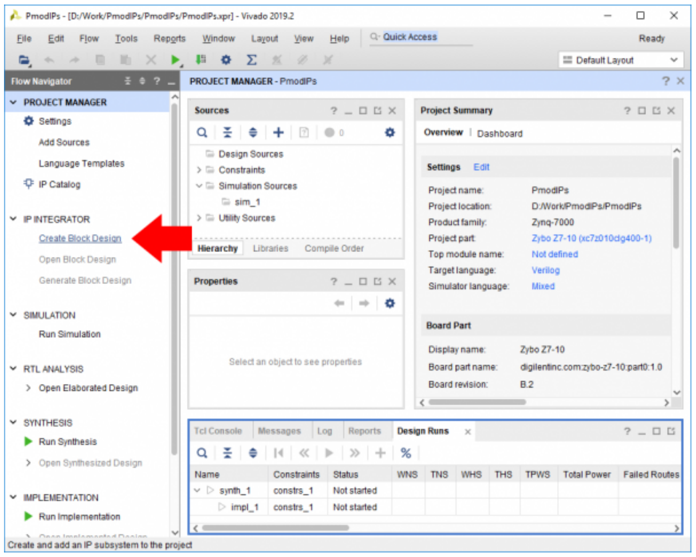
The default part and product family for the new project:
Default Board: Zybo Z7-10
Default Part: xc7z010clg400-1
Product: Zynq-7000
Family: Zynq-7000
Package: clg400
Speed Grade: -1

To create the project, click Finish

(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/create-vivado-project/new-project-5.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

Create a Block Design

Click the **Create Block Design** button in the *IP Integrator* dropdown of Vivado's *Flow Navigator* pane. A block design provides a visual representation of your hardware design, and can be used to easily connect and configure IP cores.



PmodIPs - [D:/Work/PmodIPs/PmodIPs/PmodIPs.xpr] - Vivado 2019.2

PROJECT MANAGER - PmodIPs

Settings Edit

Project name:	PmodIPs
Project location:	D:/Work/PmodIPs/PmodIPs
Product family:	Zynq-7000
Project part:	Zybo Z7-10 (xc7z010clg400-1)
Top module name:	Not defined
Target language:	Verilog
Simulator language:	Mixed

Board Part

Display name:	Zybo Z7-10
Board part name:	digilentinc.com:zybo-z7-10:part0:1.0
Board revision:	B.2

Name	Constraints	Status	WNS	TNS	WHS	THS	TPWS	Total Power	Failed Routes
synth_1	constrs_1	Not started							
impl_1	constrs_1	Not started							

Create and add an IP subsystem to the project

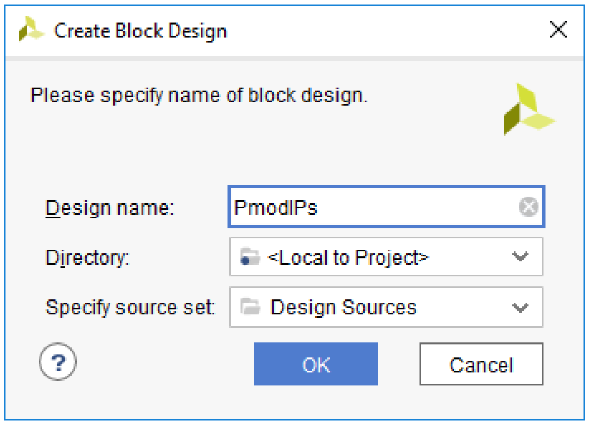
(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/create-block-design/create-block-design-1.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

In the dialog that pops up, give your block design a name (or use the default “design_1”).

Important: *Do NOT use spaces in the block design name. Instead use an underscore, a dash, or CamelCase (https://en.wikipedia.org/wiki/CamelCase).*

The other two fields should be left as defaults. *Directory* can be used to place the block design's source files outside of the project directory, and *Specify source set* can be used to create block designs that are not part of the normal Design Sources, which are used to build the project.

Click **OK** to continue.



Create Block Design

Please specify name of block design.

Design name:	PmodIPs
Directory:	<Local to Project>
Specify source set:	Design Sources

(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/create-block-design/create-block-design-2.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

Add a Processor to a Block Design

The process of adding a processor to your design is very different depending on the processor used. The Microblaze dropdown should be selected only if using a board that does not have a Zynq or Zynq UltraScale+ device.

Add a Microblaze Processor to a Block Design

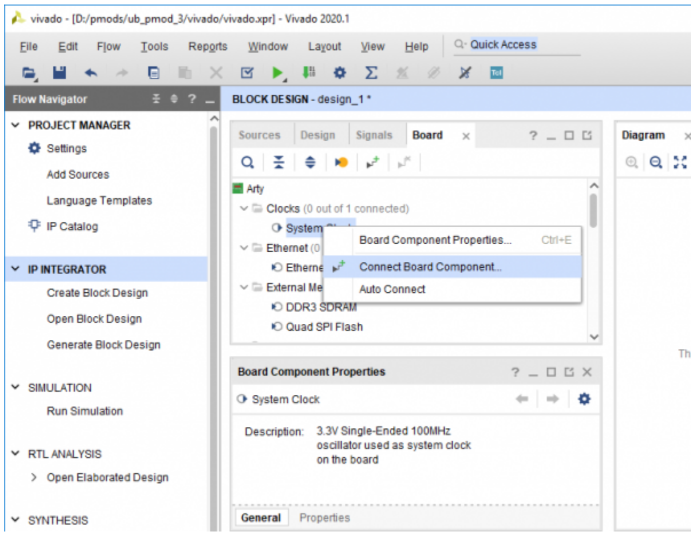
The Microblaze soft-core processor IP can be used to instantiate a processor within your FPGA design. This processor can be very useful for controlling and configuring hardware components. This section discusses how you can add a Microblaze processor and several useful components, including UART for standard output and DDR memory support, to your block design.

Note: *This section is intended for use with boards without a Zynq chip. For Zynq boards, the Zynq7 Processing System should be used instead.*

Note: *If you aren't sure whether your board has DDR memory, check the* Memory *column of the specification table on this site's Programmable Logic (https://digilent.com/reference/programmable-logic/start) page.*

Follow the steps in this dropdown for boards without DDR memory

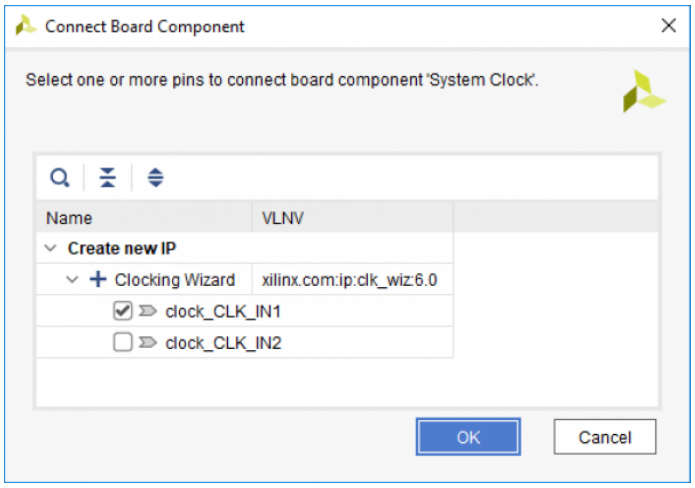
First, an external clock should be added to the block design, so that it can be used to generate any other clocks required by the design. Open the **Board** tab, and find the system clock. Right-click on it and select **Connect Board Component**.



Description: 3.3V Single-Ended 100MHz oscillator used as system clock on the board

(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/add-microblaze-processor/microblaze-1.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

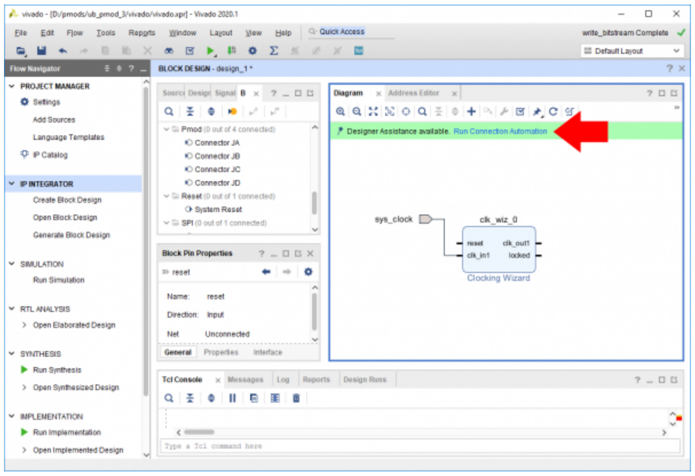
In the *Connect Board Component* dialog, select the CLK_IN1 of a new Clocking Wizard IP. This will add a clocking wizard to the design, which can be used to easily configure the board's MMCMs and PLLs to generate any required clocks. Click **OK** to continue.



Select one or more pins to connect board component 'System Clock'.

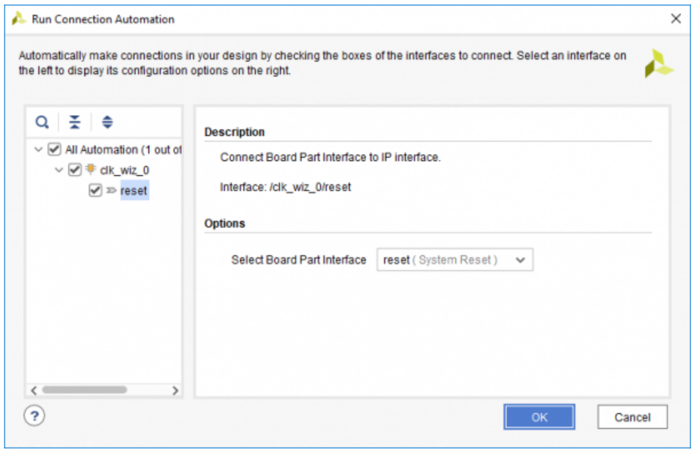
(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/add-microblaze-processor/microblaze-2.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

Next, an external reset port should be added to the design that can be used to reset the entire system. Click the **Run Connection Automation** button in the green *Designer Assistance* toolbar.


vivado - [D:/pmods/ub_pmod_3/vivado/vivado.xpr] - Vivado 2020.1
BLOCK DESIGN - design_1 *
Pmod (0 out of 4 connected)
Connector JA
Connector JB
Connector JC
Connector JD
Reset (0 out of 1 connected)
System Reset
SPI (0 out of 1 connected)
Block Pin Properties
reset
Designer Assistance available. Run Connection Automation
sys_clock
clk_wiz_0
reset
clk_in1
clk_out1
locked
Clocking Wizard
Tcl Console
Messages
Log
Reports
Design Runs
Name: reset
Direction: Input
Net: Unconnected

(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/add-microblaze-processor/microblaze-3.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

In the *Run Connection Automation* dialog, in the list on the left side of the dialog, make sure that the Clocking Wizard's reset box is checked. The *Select Board Part Interface* dropdown lists any options for resets that are specified in your board's board files. For most boards, only one option will be available. Click **OK** to continue.


Run Connection Automation

Automatically make connections in your design by checking the boxes of the interfaces to connect. Select an interface on the left to display its configuration options on the right.

Description

Connect Board Part Interface to IP interface.

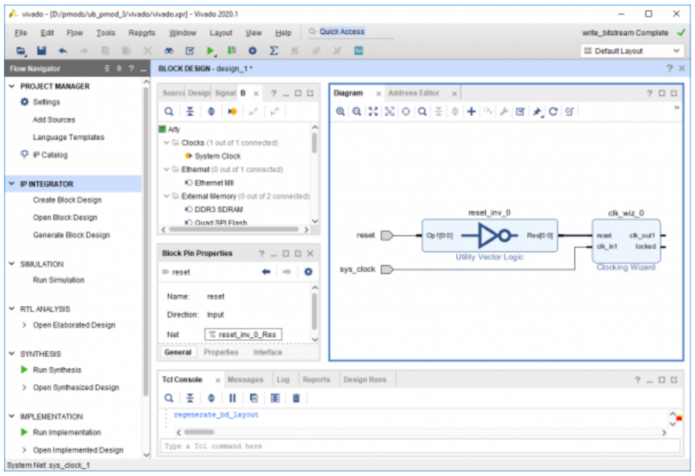
Interface: /clk_wiz_0/reset

Options

Select Board Part Interface: reset (System Reset)

(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/add-microblaze-processor/microblaze-4.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

Note: *Depending on the polarity of the reset button (active high or active low), a* Utility Vector Logic *IP may be inserted between the reset port and the Clocking Wizard. This is used to ensure that the active high reset pin of the IP is provided with the correct polarity of reset signal, and that the design will not be held in reset while the reset button is not pressed.*



Block Pin Properties

Name: reset

Direction: Input

Net: reset_inv_0_Res

```
regenerate_bd_layout
```

(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/add-microblaze-processor/microblaze-5.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

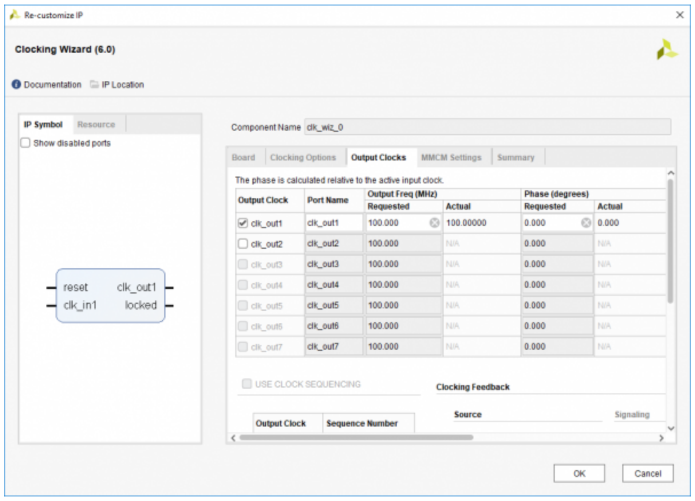
Your design will require at least one clock from the clocking wizard. If your design requires more clocks, then they must be added through the Clocking Wizard.

Double click on the Clocking Wizard IP core to edit its settings. Navigate to the *Output Clocks* tab. Additional clocks are added to the Clocking Wizard by checking a box in the *Output Clock* column, and specifying a *Requested Output Frequency*. Additionally, if desired, the ports can be named according to their intended purpose.

If your design requires additional clocks (such as for an ext_spi_clock pin), they should be added now.

Click **OK** to confirm your changes to the Clocking Wizard's settings.

If you aren't sure that you have all of the clocks you need, don't worry, you can always come back and add them by reconfiguring this IP. This task can be performed whenever in the design process it becomes necessary.

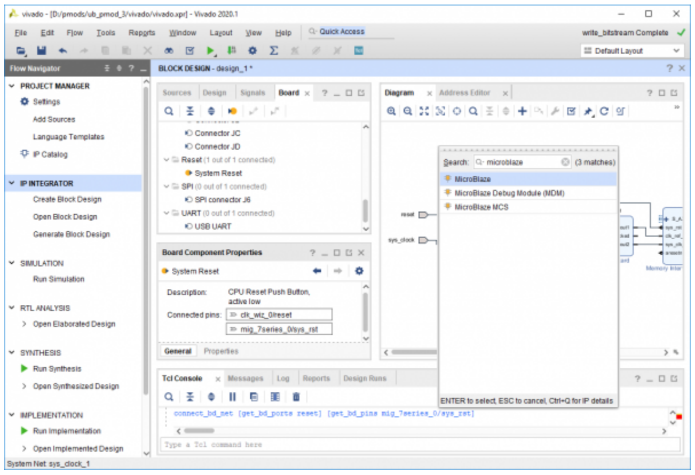


The phase is calculated relative to the active input clock.

Output Clock	Port Name	Output Freq (MHz) Requested	Actual	Phase (degrees) Requested	Actual
✓ clk_out1	clk_out1	100.000	100.00000	0.000	0.000
clk_out2	clk_out2	100.000	N/A	0.000	N/A
clk_out3	clk_out3	100.000	N/A	0.000	N/A
clk_out4	clk_out4	100.000	N/A	0.000	N/A
clk_out5	clk_out5	100.000	N/A	0.000	N/A
clk_out6	clk_out6	100.000	N/A	0.000	N/A
clk_out7	clk_out7	100.000	N/A	0.000	N/A

(https://digilent.com/reference/_detail/reference/programmable-logic/guides/microblaze-8.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

Next, use the **Add IP** button (+) to add the *MicroBlaze* IP to the design. The block that is added represents the core of the Microblaze processor.

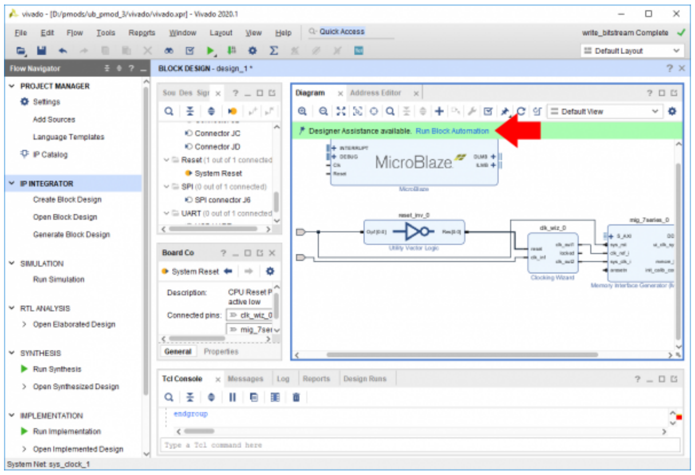


```
connect_bd_net [get_bd_ports reset] [get_bd_pins mig_7series_0/sys_rst]
```

(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/add-microblaze-processor/microblaze-12.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

Next, block automation will be run so that additional supporting infrastructure can be added to the design. Click **Run Block Automation** in the green *Designer Assistance* bar.

Note: *The screenshot to the right is not representative for a design not using DDR, as these designs will not contain a MIG IP core.*



```
endgroup
```

(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/add-microblaze-processor/microblaze-13.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

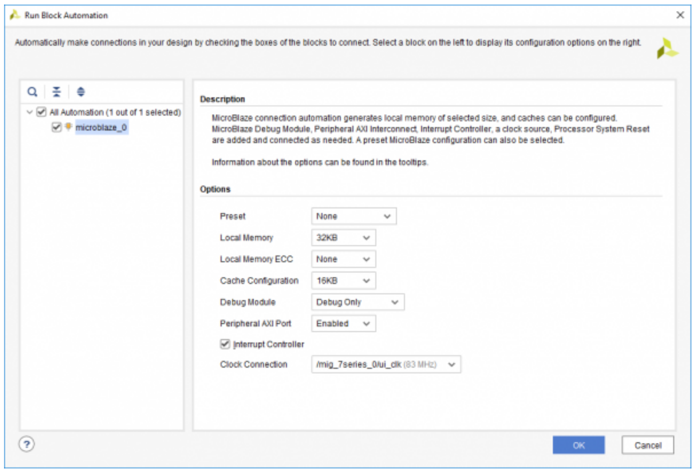
In the *Run Block Automation* dialog, several settings can be specified for how the Microblaze IP will be connected to the rest of the design:

- **Local Memory** specifies how much block RAM () memory will be dedicated to the processor. DDR-less designs require more memory, and the amount of memory necessary depends heavily on the size of the software application being run. 32KB is enough for many small applications.
- **Cache Configuration** can help the speed of designs using DDR memory. It should be enabled when using DDR and disabled otherwise.
- **Debug Module** allows you to specify the capabilities of the debugger. The default *Debug Only* option is recommended.
- **Peripheral AXI Port** enables or disables the AXI master interface of the processor. It must be enabled to allow the processor to be connected to hardware peripherals.
- **Interrupt Controller** specifies whether the processor can be interrupted by its peripherals. Whether or not it needs to be enabled depends on your design requirements. If any IP that you intend to connect to the processor must have interrupts to function correctly, the box must be checked.
- **Clock Connection** specifies the processor's clock. Designs using DDR should use the MIG's ui_clk pin, while designs without DDR should use the Clocking Wizards clk_out1 pin.

Note: *Settings not present in this list are out of the scope of this guide, and can safely be left as their default.*

Confirm that the settings meet your design requirements. It should be noted that while it is possible to change these settings manually later (for example, by manually adding an AXI INTC IP and connecting it to the processor), the easiest way to do so will be to clear the Microblaze processor out of your block design and restart the process of adding the processor. This is to say, the settings chosen here are important. Getting them right the first time will save you time in the long run.

Click **OK** to continue.



Automatically make connections in your design by checking the boxes of the blocks to connect. Select a block on the left to display its configuration options on the right.

All Automation (1 out of 1 selected)
microblaze_0

Description

MicroBlaze connection automation generates local memory of selected size, and caches can be configured. MicroBlaze Debug Module, Peripheral AXI Interconnect, Interrupt Controller, a clock source, Processor System Reset are added and connected as needed. A preset MicroBlaze configuration can also be selected.

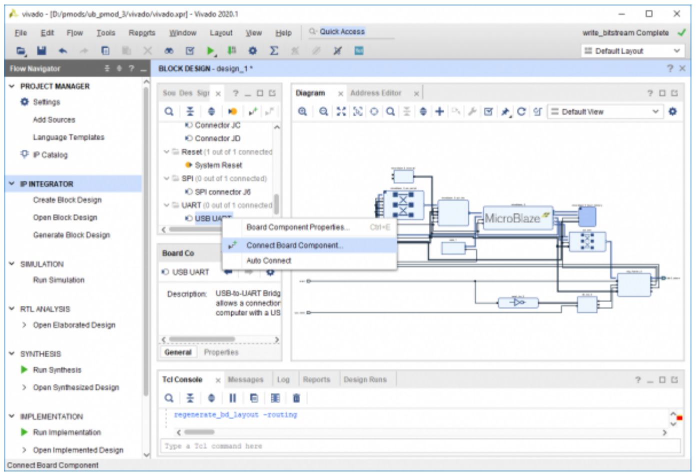
Information about the options can be found in the tooltips.

Options

Preset	None
Local Memory	32KB
Local Memory ECC	None
Cache Configuration	16KB
Debug Module	Debug Only
Peripheral AXI Port	Enabled
✓ Interrupt Controller	
Clock Connection	/mig_7series_0/ui_clk (83 MHz)

(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/add-microblaze-processor/microblaze-14.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

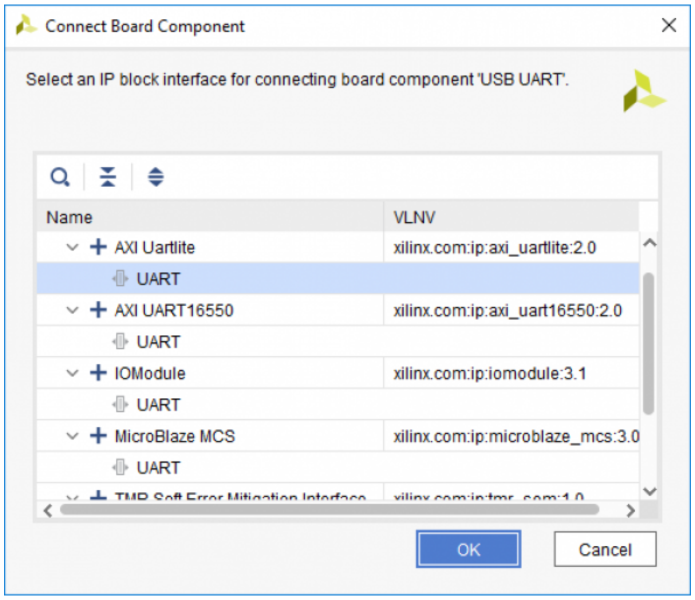
Next, in order for the software design to be able to print to a serial console on a host computer, a UART peripheral must be connected. Find the *USB UART* interface in the *Board* tab, right click on it, and select **Connect Board Component**.



```
regenerate_bd_layout -routing
```

(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/add-microblaze-processor/microblaze-18.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

In the dialog that pops up, select a new AXI Uartlite IP's UART interface, and click **OK** to add the block to the diagram.

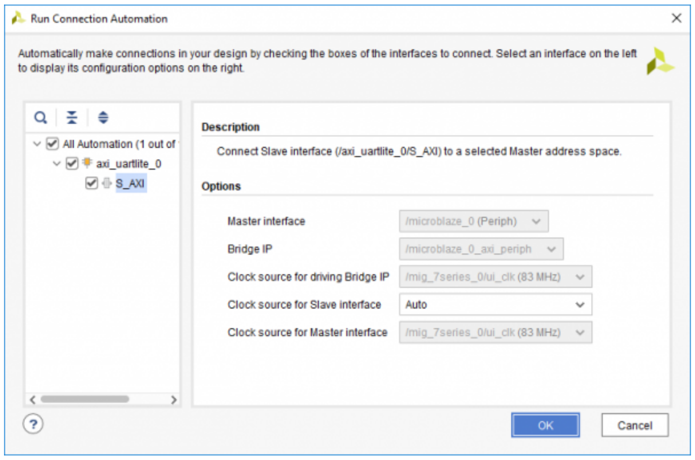


Select an IP block interface for connecting board component 'USB UART'.

Name	VLNV
AXI Uartlite	xilinx.com:ip:axi_uartlite:2.0
UART	
AXI UART16550	xilinx.com:ip:axi_uart16550:2.0
UART	
IOModule	xilinx.com:ip:iomodule:3.1
UART	
MicroBlaze MCS	xilinx.com:ip:microblaze_mcs:3.0
UART	
TMR Soft Error Mitigation Interface	xilinx.com:ip:tmr_sem:1.0

(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/add-microblaze-processor/microblaze-19.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

To connect all remaining AXI cores that have not yet been connected to the processor, click the **Run Connection Automation** button in the green *Designer Assistance* bar. Check the *All Automation* box in the list on the left side of the window to select all of the remaining AXI cores. The dropdown settings available for each core can safely be left as their default values. Click **OK** to automatically connect them to the processor.



Automatically make connections in your design by checking the boxes of the interfaces to connect. Select an interface on the left to display its configuration options on the right.

Description

Connect Slave interface (/axi_uartlite_0/S_AXI) to a selected Master address space.

Options

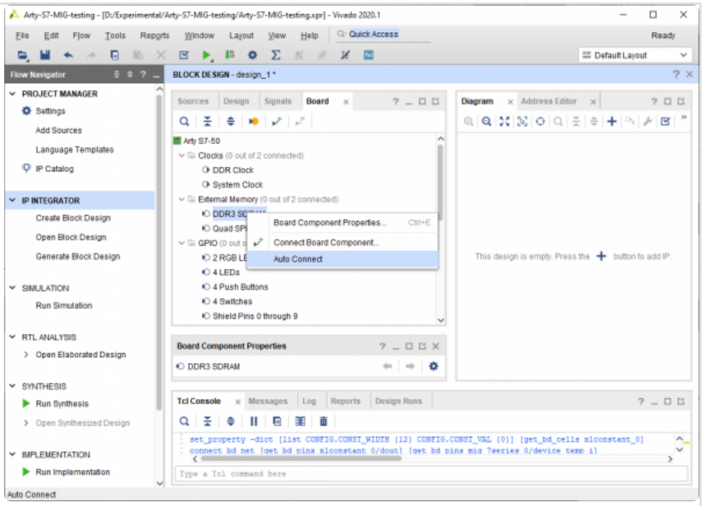
Master interface	/microblaze_0 (Periph)
Bridge IP	/microblaze_0_axi_periph
Clock source for driving Bridge IP	/mig_7series_0/ui_clk (83 MHz)
Clock source for Slave interface	Auto
Clock source for Master interface	/mig_7series_0/ui_clk (83 MHz)

(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/add-microblaze-processor/microblaze-20.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

Follow the steps in this dropdown for boards with DDR memory

When creating a design with DDR, it's best to add the DDR interface first, as it is typically also used to generate the clock or clocks that will be used by the rest of your design.

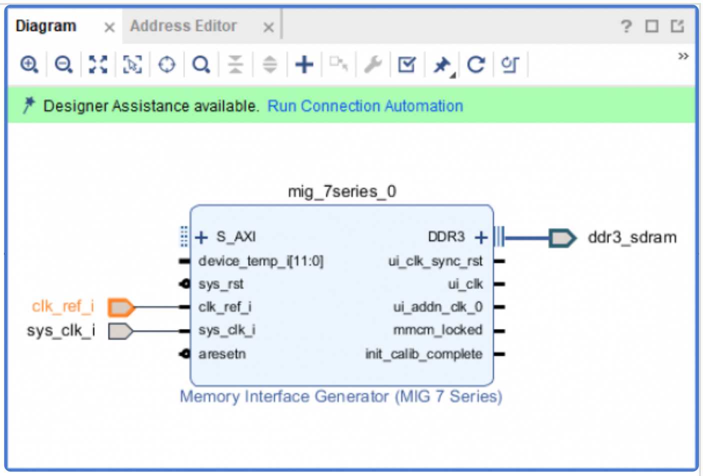
In the Board tab, right click on the DDR interface and select "Auto Connect". This process will add a MIG and the external DDR interface to the design. Two clock pins are also created, which will need to be modified.


Board Component Properties
Connect Board Component...
Auto Connect
This design is empty. Press the + button to add IP.

```
set_property -dict [list CONFIG.CONST_WIDTH {12} CONFIG.CONST_VAL {0}] [get_bd_cells xlconstant_0]
connect_bd_net [get_bd_pins xlconstant_0/dout] [get_bd_pins mig_7series_0/device_temp_i]
```

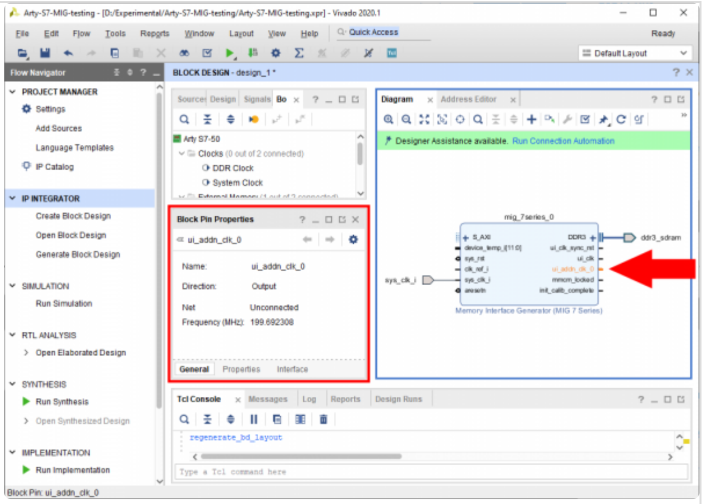
(https://digilent.com/reference/_detail/programmable-logic/guides/auto-connect-ddr.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

Delete the "clk_ref_i" pin. This can be accomplished either by right-clicking on the pin and selecting delete or by selecting and pressing the delete key.



(https://digilent.com/reference/_detail/programmable-logic/guides/delete-clk-ref.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

Verify that the "ui_addn_clk_0" pin has a frequency near 200 MHz () by selecting it and looking at the "Block Pin Properties" pane.

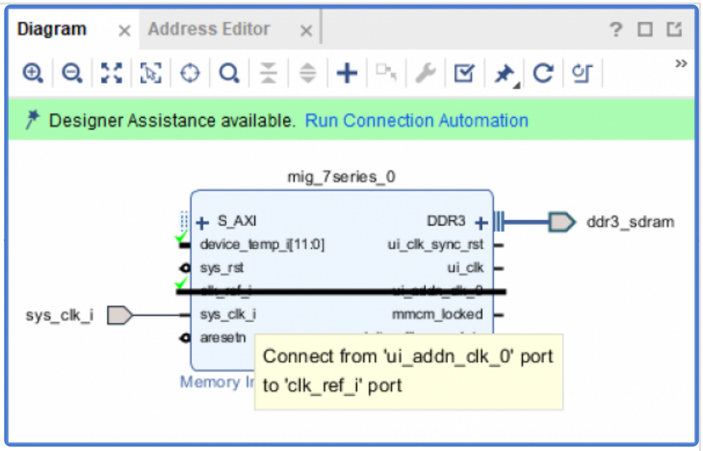


Name: ui_addn_clk_0
Direction: Output
Net: Unconnected
Frequency (MHz): 199.692308

```
regenerate_bd_layout
```

(https://digilent.com/reference/_detail/programmable-logic/guides/check-ui-addn-clk-properties.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

Manually connect the “ui_addn_clk_0” pin to the “clk_ref_i” pin by clicking and dragging from one to the other.



Connect from 'ui_addn_clk_0' port to 'clk_ref_i' port

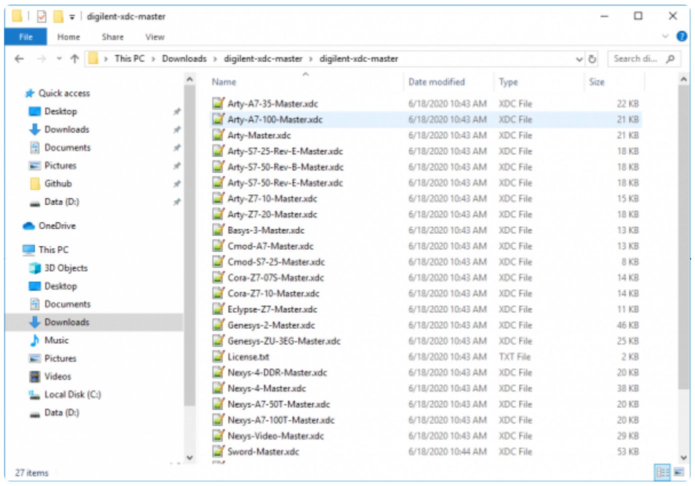
(https://digilent.com/reference/_detail/programmable-logic/guides/create-clk-loopback.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

It's important to note that the “sys_clk_i” pin will not be constrained by the board files, and you will need to add a constraint file to map it to the corresponding pin location on the FPGA.

If your project doesn't contain the master Xilinx Design Constraint (XDC) file for your board, the dropdown below details how to add it. This file contains the constraints that your board places on designs using it - specific interfaces wired up to specific pins, clock frequencies, and FPGA bank voltages, for some examples. Click the dropdown below for a walkthrough of how to add this file to your project.

Add a Master XDC File to a Vivado Project

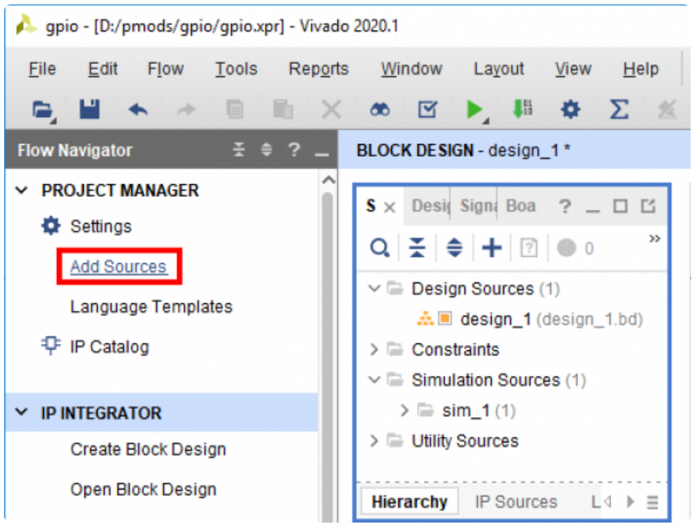
Download and extract digilent-xdc-master.zip (https://digilent.com/reference/lib/exe/fetch.php?tok=a49a20&media=https%3A%2F%2Fgithub.com%2FDigilent%2Fdigilent-xdc%2Farchive%2Fmaster.zip). This file includes all of the latest template XDC files released for Digilent's boards, which are available on Github in the digilent-xdc (https://github.com/Digilent/digilent-xdc) repository.



Name	Date modified	Type	Size
Arty-A7-35-Master.xdc	6/18/2020 10:43 AM	XDC File	22 KB
Arty-A7-100-Master.xdc	6/18/2020 10:43 AM	XDC File	21 KB
Arty-Master.xdc	6/18/2020 10:43 AM	XDC File	21 KB
Arty-S7-25-Rev-E-Master.xdc	6/18/2020 10:43 AM	XDC File	18 KB
Arty-S7-50-Rev-B-Master.xdc	6/18/2020 10:43 AM	XDC File	18 KB
Arty-S7-50-Rev-E-Master.xdc	6/18/2020 10:43 AM	XDC File	18 KB
Arty-Z7-10-Master.xdc	6/18/2020 10:43 AM	XDC File	15 KB
Arty-Z7-20-Master.xdc	6/18/2020 10:43 AM	XDC File	18 KB
Basys-3-Master.xdc	6/18/2020 10:43 AM	XDC File	13 KB
Cmod-A7-Master.xdc	6/18/2020 10:43 AM	XDC File	13 KB
Cmod-S7-25-Master.xdc	6/18/2020 10:43 AM	XDC File	8 KB
Cora-Z7-07S-Master.xdc	6/18/2020 10:43 AM	XDC File	14 KB
Cora-Z7-10-Master.xdc	6/18/2020 10:43 AM	XDC File	14 KB
Eclypse-Z7-Master.xdc	6/18/2020 10:43 AM	XDC File	11 KB
Genesys-2-Master.xdc	6/18/2020 10:43 AM	XDC File	46 KB
Genesys-ZU-3EG-Master.xdc	6/18/2020 10:43 AM	XDC File	25 KB
License.txt	6/18/2020 10:43 AM	TXT File	2 KB
Nexys-4-DDR-Master.xdc	6/18/2020 10:43 AM	XDC File	20 KB
Nexys-4-Master.xdc	6/18/2020 10:43 AM	XDC File	38 KB
Nexys-A7-50T-Master.xdc	6/18/2020 10:43 AM	XDC File	20 KB
Nexys-A7-100T-Master.xdc	6/18/2020 10:43 AM	XDC File	20 KB
Nexys-Video-Master.xdc	6/18/2020 10:43 AM	XDC File	29 KB
Sword-Master.xdc	6/18/2020 10:44 AM	XDC File	53 KB

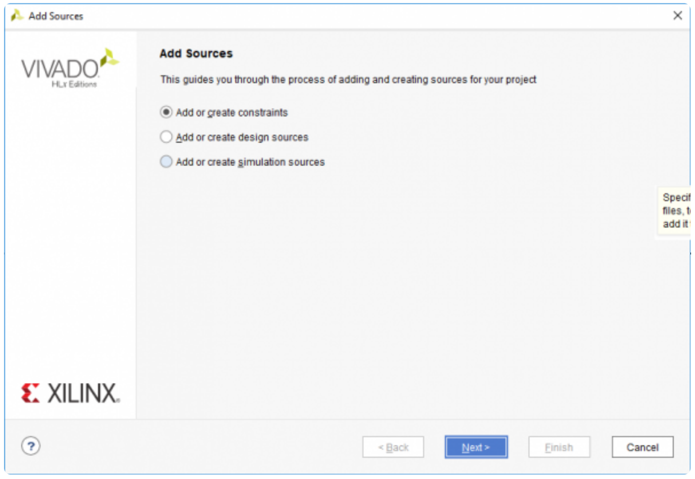
(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/add-gpio-to-block-design/extracted-xdc-folder.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

Returning to Vivado, click the **Add Sources** button in the *Project Manager* section of the *Flow Navigator* pane. This will launch a dialog that you can use to add a variety of types of source files to the project (or create new ones).



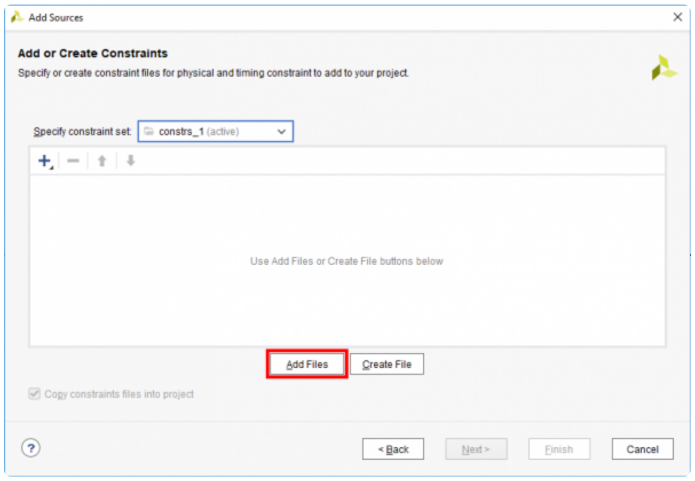
(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/add-gpio-to-block-design/add-sources.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

On the first screen, select *Add or create constraints*. Click **Next** to continue.


Add Sources
VIVADO
Add Sources
This guides you through the process of adding and creating sources for your project
Add or create constraints
Add or create design sources
Add or create simulation sources
XILINX
< Back
Next >
Finish
Cancel

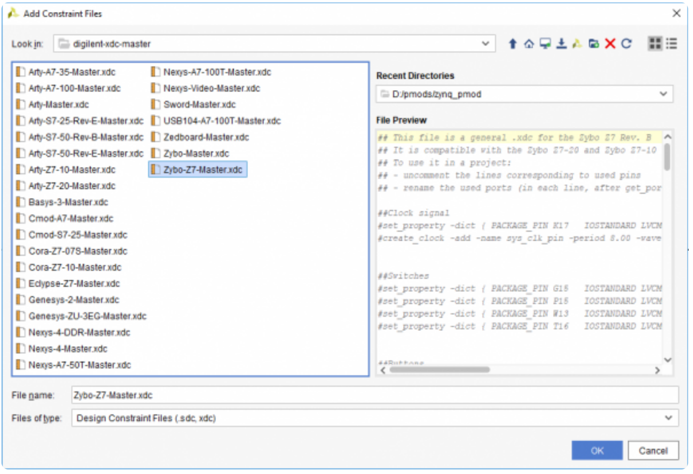
(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/add-gpio-to-block-design/add-constraints.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

In the next screen, make sure that the constraint set specified (the one that the master XDC will be added to) is set to *constrs_1*, and that it is the *active* set. Click the **Add Files** button.


Add Sources
Add or Create Constraints
Specify or create constraint files for physical and timing constraint to add to your project.
Specify constraint set: constrs_1 (active)
Use Add Files or Create File buttons below
Add Files
Create File
Copy constraints files into project
< Back
Next >
Finish
Cancel

(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/add-gpio-to-block-design/add-files.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

In the dialog that pops up, navigate to the folder that the *digilent-xdc-master.zip* file was extracted into. Highlight the XDC file for your board, then click **OK** to continue.



```
## This file is a general .xdc for the Zybo Z7 Rev. B
## It is compatible with the Zybo Z7-20 and Zybo Z7-10
## To use it in a project:
## - uncomment the lines corresponding to used pins
## - rename the used ports (in each line, after get_por

##Clock signal
#set_property -dict { PACKAGE_PIN K17   IOSTANDARD LVCM
#create_clock -add -name sys_clk_pin -period 8.00 -wave


##Switches
#set_property -dict { PACKAGE_PIN G15   IOSTANDARD LVCM
#set_property -dict { PACKAGE_PIN P15   IOSTANDARD LVCM
#set_property -dict { PACKAGE_PIN W13   IOSTANDARD LVCM
#set_property -dict { PACKAGE_PIN T16   IOSTANDARD LVCM
```

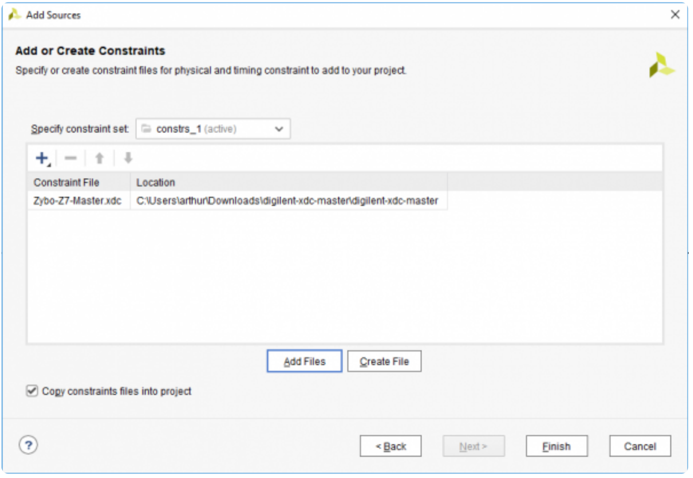
File name: Zybo-Z7-Master.xdc

Files of type: Design Constraint Files (.sdc, xdc)

(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/add-gpio-to-block-design/find-xdc.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

Back in the *Add Sources* dialog, make sure that your chosen constraint file appears in the table. Also, make sure that the *Copy constraint files into project* box is checked. If this box is unchecked, the file will be linked by your project, and any modifications made within the project will affect the version you downloaded. Since you may need to use this file again in other projects, copying the constraint file is recommended, so that you can always work from a fresh copy.

Click **Finish** to add the constraint file to your project.



Add or Create Constraints

Specify or create constraint files for physical and timing constraint to add to your project.

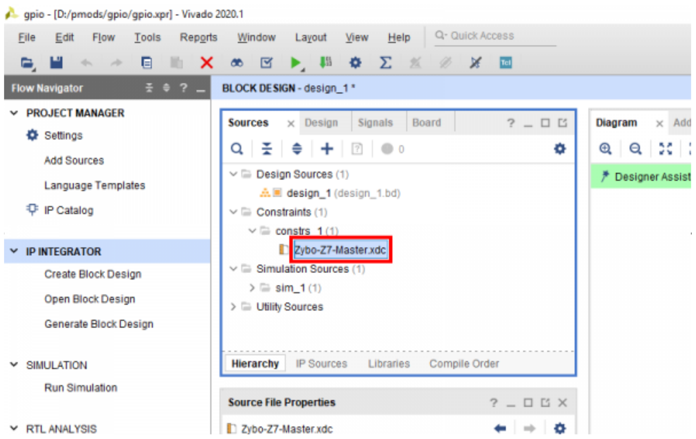
Specify constraint set: constrs_1 (active)

Constraint File	Location
Zybo-Z7-Master.xdc	C:\Users\arthur\Downloads\digilent-xdc-master\digilent-xdc-master

[✓] Copy constraints files into project

(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/add-gpio-to-block-design/import-constraint-file.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

Once added, the XDC file will appear in the *Sources* tab (in the same pane as the *Board* tab). Double click it to open the file.

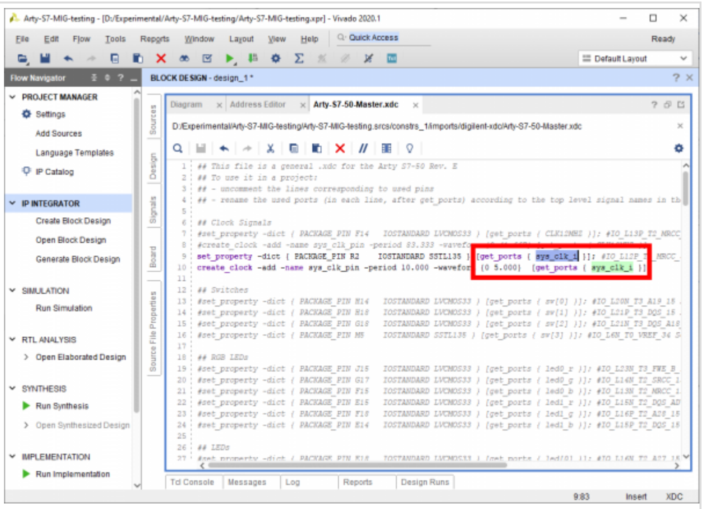


(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/add-gpio-to-block-design/open-constraint.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

Find the set_property and create_clock constraints for your board's 100 MHz () input clock, uncomment them by removing the `#` at the start of each line, and change the name of the *port* that they are constraining to `sys_clk_i`, to match the name of the port in the block design.

If your board has multiple clocks, the 100 MHz () one can be determined by observing that the create_clock constraint specifies a `10.000` ns period, as seen in the screenshot to the right.

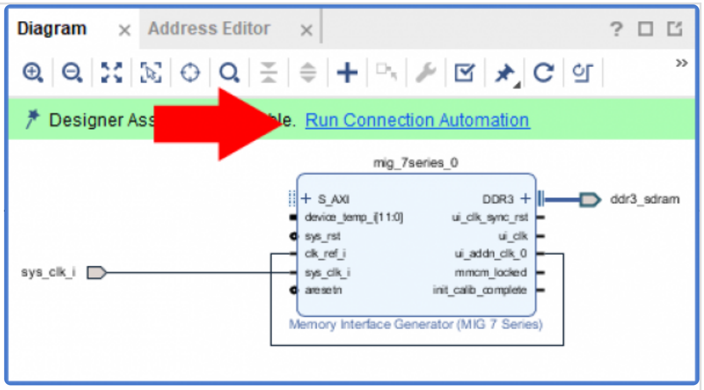
Make sure to save your changes.



```
## This file is a general .xdc for the Arty S7-50 Rev. E
## To use it in a project:
## - uncomment the lines corresponding to used pins
## - rename the used ports (in each line, after get_ports) according to the top level signal names in th

## Clock Signals
#set_property -dict { PACKAGE_PIN F14   IOSTANDARD LVCMOS33 } [get_ports { CLK12MHZ }]; #IO_L13P_T2_MRCC_
#create_clock -add -name sys_clk_pin -period 83.333 -wavefo
set_property -dict { PACKAGE_PIN R2    IOSTANDARD SSTL135 } [get_ports { sys_clk_i }]; #IO_L12P_T1_MRCC_
create_clock -add -name sys_clk_pin -period 10.000 -wavefor {0 5.000}  [get_ports { sys_clk_i }]

## Switches
#set_property -dict { PACKAGE_PIN H14   IOSTANDARD LVCMOS33 } [get_ports { sw[0] }]; #IO_L20N_T3_A19_15
#set_property -dict { PACKAGE_PIN H18   IOSTANDARD LVCMOS33 } [get_ports { sw[1] }]; #IO_L21P_T3_DQS_15
#set_property -dict { PACKAGE_PIN G18   IOSTANDARD LVCMOS33 } [get_ports { sw[2] }]; #IO_L21N_T3_DQS_A18
#set_property -dict { PACKAGE_PIN M5    IOSTANDARD SSTL135 } [get_ports { sw[3] }]; #IO_L6N_T0_VREF_34 S

## RGB LEDs
#set_property -dict { PACKAGE_PIN J15   IOSTANDARD LVCMOS33 } [get_ports { led0_r }]; #IO_L23N_T3_FWE_B_
#set_property -dict { PACKAGE_PIN G17   IOSTANDARD LVCMOS33 } [get_ports { led0_g }]; #IO_L14N_T2_SRCC_1
#set_property -dict { PACKAGE_PIN F15   IOSTANDARD LVCMOS33 } [get_ports { led0_b }]; #IO_L13N_T2_MRCC_1
#set_property -dict { PACKAGE_PIN E15   IOSTANDARD LVCMOS33 } [get_ports { led1_r }]; #IO_L15N_T2_DQS_AD
#set_property -dict { PACKAGE_PIN F18   IOSTANDARD LVCMOS33 } [get_ports { led1_g }]; #IO_L16P_T2_A28_15
#set_property -dict { PACKAGE_PIN E14   IOSTANDARD LVCMOS33 } [get_ports { led1_b }]; #IO_L15P_T2_DQS_15

## LEDs
#set_property -dict { PACKAGE_PIN E18   IOSTANDARD LVCMOS33 } [get_ports { led[0] }]; #IO_L16N_T2_A27_15
```

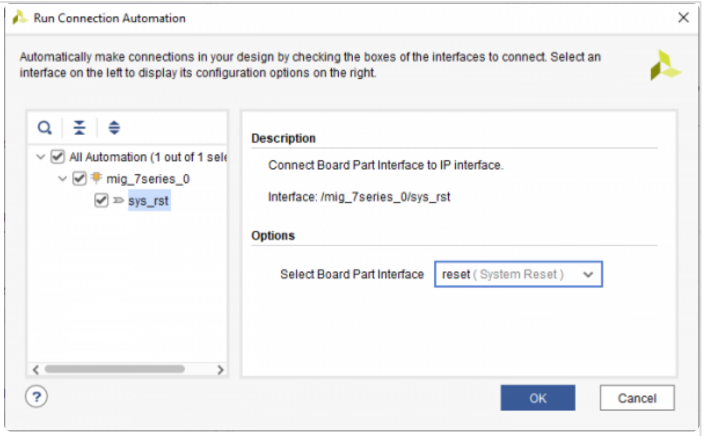
(https://digilent.com/reference/_detail/programmable-logic/guides/change-port-name.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

Next, the MIG's reset pin will be connected to the board's reset button. Click "Run Connection Automation" in the green bar at the top of the window.



(https://digilent.com/reference/_detail/programmable-logic/guides/run-reset-automation.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

In the list to the left side of the dialog that pops up, make sure that the "sys_rst" box is checked. Click **OK** to connect the reset to the corresponding board part interface.



Run Connection Automation

Automatically make connections in your design by checking the boxes of the interfaces to connect. Select an interface on the left to display its configuration options on the right.

- [✓] All Automation (1 out of 1 sele
  - [✓] mig_7series_0
    - [✓] sys_rst

Description

Connect Board Part Interface to IP interface.

Interface: /mig_7series_0/sys_rst

Options

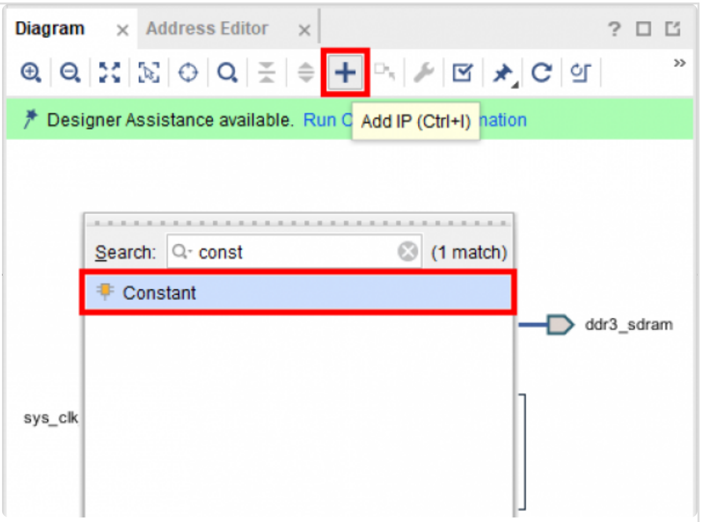
Select Board Part Interface: reset (System Reset)

OK Cancel

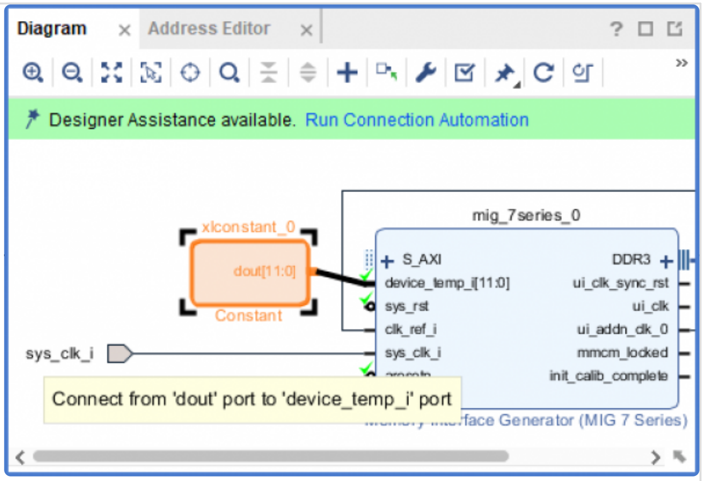
(https://digilent.com/reference/_detail/programmable-logic/guides/run-reset-automation-wizard.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

The MIG block may have other ports which will need to be connected to ensure that it works correctly. This section discusses each of those ports.

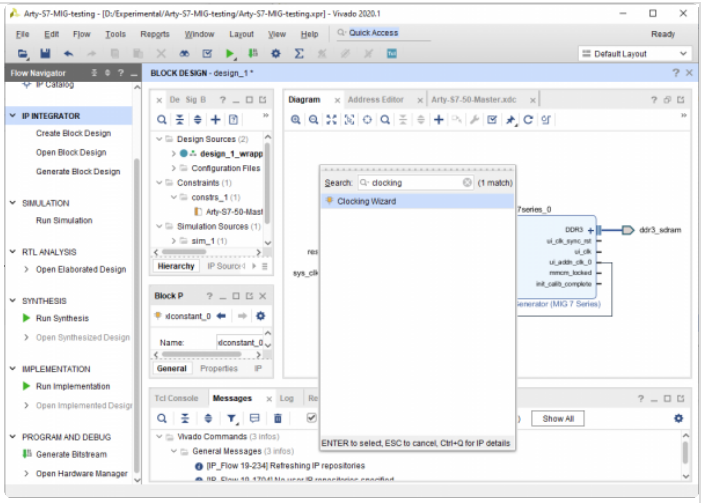
Open this dropdown if your MIG has a "device_temp_i" port

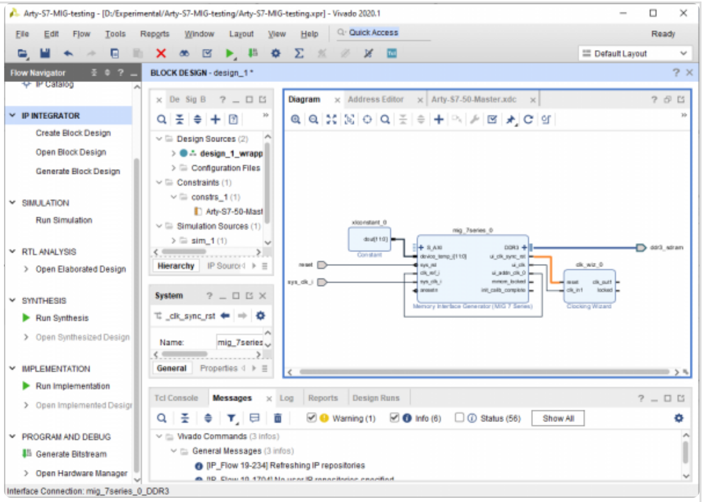
If your MIG block has a “device_temp_i” port, it means that the MIG is not using the chip's XADC analog-to-digital converter feature. We'll ground this port to prevent any warnings that it may throw. Add a “Constant” IP to the design.	 Diagram Address Editor Designer Assistance available. Run C Add IP (Ctrl+I) nation Search: const (1 match) Constant ddr3_sdram sys_clk (https://digilent.com/reference/_detail/programmable-logic/guides/add-const.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

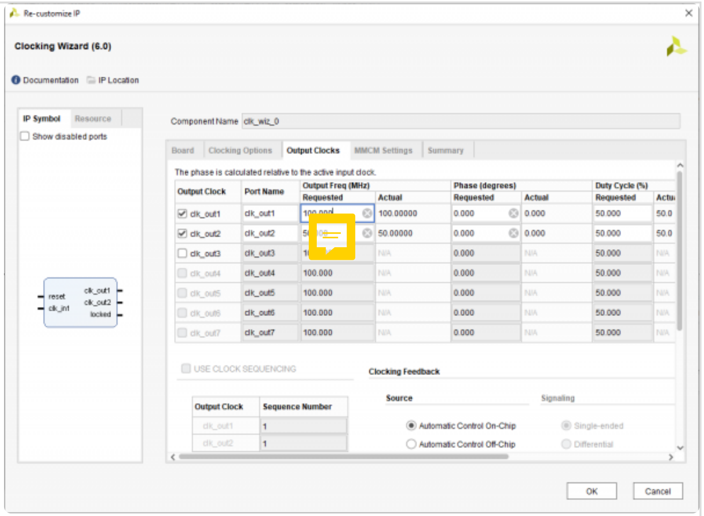
Double click on the block to open it's configuration wizard and modify it to have a Value of “0” and a Width of “12”.	(https://digilent.com/reference/_detail/programmable-logic/guides/configure-const.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

Connect its output port to the “device_temp_i” port on the MIG.	 Diagram Address Editor Designer Assistance available. Run Connection Automation mig_7series_0 xlconstant_0 dout[11:0] Constant S_AXI device_temp_i[11:0] sys_rst clk_ref_i sys_clk_i DDR3 ui_clk_sync_rst ui_clk ui_addn_clk_0 mmcm_locked init_calib_complete sys_clk_i Connect from 'dout' port to 'device_temp_i' port face Generator (MIG 7 Series) (https://digilent.com/reference/_detail/programmable-logic/guides/connect-const-to-temp.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

If your design requires more clocks than just the ui_clk provided by the MIG, you will need to add a clocking wizard IP that is driven by the MIG's ui_clk.

Use the Add IP button to search for and add a Clocking Wizard to the design.	 Search: clocking (1 match) Clocking Wizard ENTER to select, ESC to cancel, Ctrl+Q for IP details (https://digilent.com/reference/_detail/programmable-logic/guides/add-clocking-wizard-ip.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

Manually connect the Clocking Wizard's clk_in1 and reset ports to the MIG's ui_clk and ui_clk_sync_rst ports, respectively.	 BLOCK DESIGN - design_1 * Diagram xlconstant_0 mig_7series_0 clk_wiz_0 Memory Interface Generator (MIG 7 Series) Clocking Wizard ddr3_sdram reset sys_clk_i (https://digilent.com/reference/_detail/programmable-logic/guides/connect-wizard.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

Finally, double-click on the Clocking Wizard to open and configure it. The third tab, *Output Clocks* contains all of the settings required to specify how many clocks you need, and of what frequencies. The screenshot to the right shows the wizard configured to create a 100 MHz () clk_out1 and a 50 MHz () clk_out2. If your design requires additional clocks (such as for an ext_spi_clock pin), they should be added here.	 Re-customize IP Clocking Wizard (6.0) Documentation IP Location IP Symbol Resource Show disabled ports reset clk_in1 clk_out1 clk_out2 locked Board Clocking Options Output Clocks MMCM Settings Summary OK Cancel (https://digilent.com/reference/_detail/programmable-logic/guides/configure-clocks.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

Component Name clk_wiz_0

The phase is calculated relative to the active input clock.

Output Clock	Port Name	Output Freq (MHz)		Phase (degrees)		Duty Cycle (%)	
		Requested	Actual	Requested	Actual	Requested	Actu
✓ clk_out1	clk_out1	100.000	100.00000	0.000	0.000	50.000	50.0
✓ clk_out2	clk_out2	50	50.00000	0.000	0.000	50.000	50.0
clk_out3	clk_out3	10	N/A	0.000	N/A	50.000	N/A
clk_out4	clk_out4	100.000	N/A	0.000	N/A	50.000	N/A
clk_out5	clk_out5	100.000	N/A	0.000	N/A	50.000	N/A
clk_out6	clk_out6	100.000	N/A	0.000	N/A	50.000	N/A
clk_out7	clk_out7	100.000	N/A	0.000	N/A	50.000	N/A

USE CLOCK SEQUENCING

Output Clock	Sequence Number
clk_out1	1
clk_out2	1

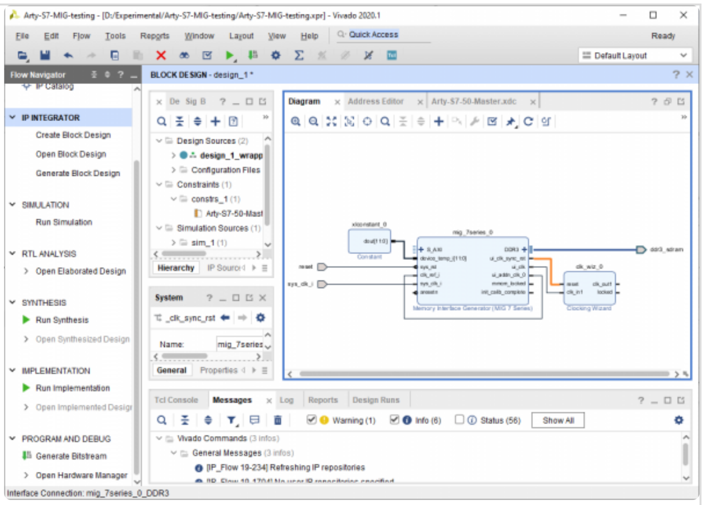
Clocking Feedback

Source: [✓ Automatic Control On-Chip] Automatic Control Off-Chip

Signaling: [✓ Single-ended] Differential

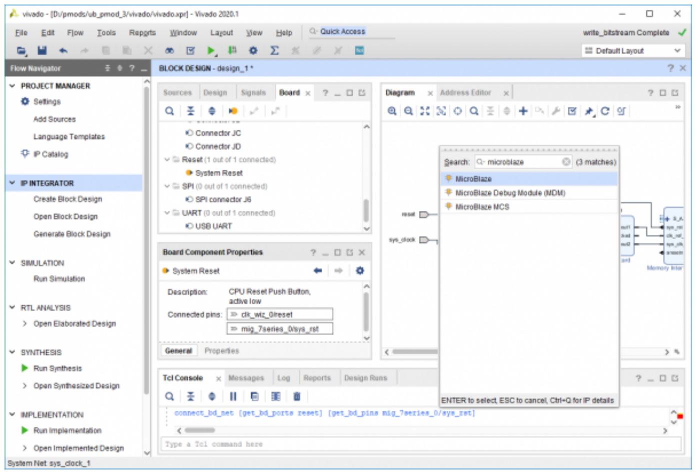
Note: *If you aren't sure that you have all of the clocks you need, don't worry, you can always come back and add them by reconfiguring this IP. This task can be performed whenever in the design process it becomes necessary.*

At the end of this process, the block design will look something like this:



(https://digilent.com/reference/_detail/programmable-logic/guides/connect-wizard.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

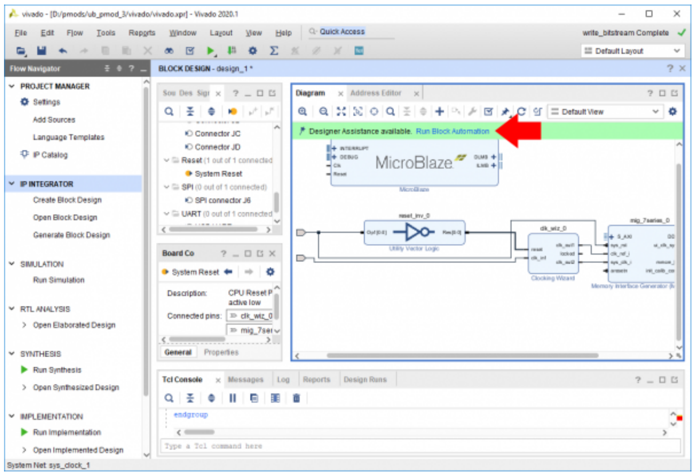
Next, use the **Add IP** button (+) to add the *MicroBlaze* IP to the design. The block that is added represents the core of the Microblaze processor.



(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/add-microblaze-processor/microblaze-12.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

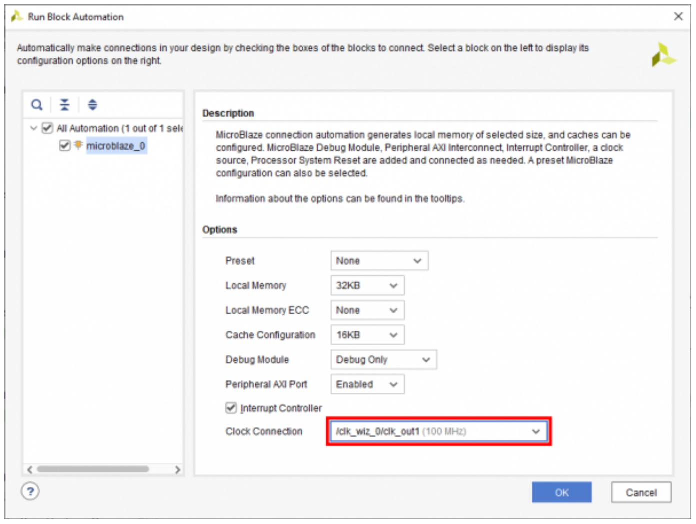
Next, block automation will be run so that additional supporting infrastructure can be added to the design. Click **Run Block Automation** in the green *Designer Assistance* bar.

Note: *The screenshot to the right is not representative for a design not using DDR, as these designs will not contain a MIG IP core.*


Designer Assistance available. Run Block Automation
MicroBlaze
BLOCK DESIGN - design_1 *
Flow Navigator

(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/add-microblaze-processor/microblaze-13.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

In the *Run Block Automation* dialog, several settings can be specified for how the Microblaze IP will be connected to the rest of the design:



Run Block Automation

Automatically make connections in your design by checking the boxes of the blocks to connect. Select a block on the left to display its configuration options on the right.

All Automation (1 out of 1 sele
microblaze_0

Description

MicroBlaze connection automation generates local memory of selected size, and caches can be configured. MicroBlaze Debug Module, Peripheral AXI Interconnect, Interrupt Controller, a clock source, Processor System Reset are added and connected as needed. A preset MicroBlaze configuration can also be selected.

Information about the options can be found in the tooltips.

Options

Preset	None
Local Memory	32KB
Local Memory ECC	None
Cache Configuration	16KB
Debug Module	Debug Only
Peripheral AXI Port	Enabled
✓ Interrupt Controller	
Clock Connection	/clk_wiz_0/clk_out1 (100 MHz)

OK Cancel

(https://digilent.com/reference/_detail/programmable-logic/guides/block-automation.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

- **Local Memory** specifies how much block RAM () memory will be dedicated to the processor. DDR-less designs require more memory, and the amount of memory necessary depends heavily on the size of the software application being run. 32KB is enough for many small applications.
- **Cache Configuration** can help the speed of designs using DDR memory. It should be enabled when using DDR, and disabled otherwise.
- **Debug Module** allows you to specify the capabilities of the debugger. The default *Debug Only* option is recommended.
- **Peripheral AXI Port** enables or disables the AXI master interface of the processor. It must be enabled to allow the processor to be connected to hardware peripherals.
- **Interrupt Controller** specifies whether the processor can be interrupted by its peripherals. Whether or not it needs to be enabled depends on your design requirements. If any IP that you intend to connect to the processor must have interrupts to function correctly, the box must be checked.
- **Clock Connection** specifies the processor's clock. Designs using DDR should use the MIG's ui_clk pin, while designs without DDR should use the Clocking Wizards clk_out1 pin.

Note: *Settings not present in this list are out of the scope of this guide, and can safely be left as their default.*

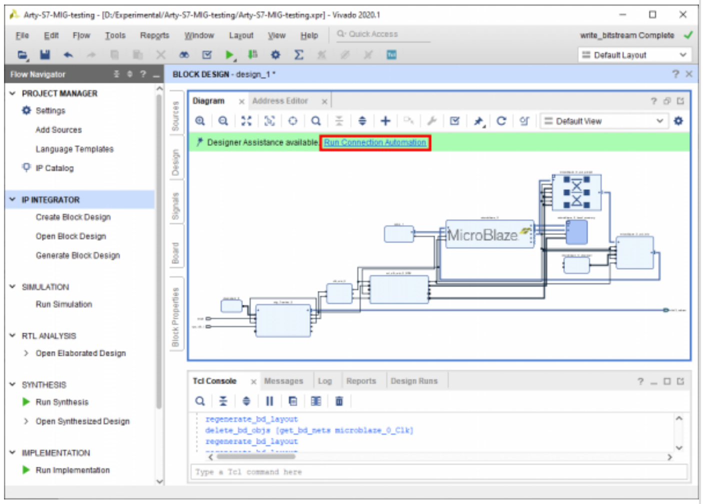
Confirm that the settings meet your design requirements. It should be noted that while it is possible to change these settings manually later (for example, by manually adding an AXI INTC IP and connecting it to the processor), the easiest way to do so will be to clear the Microblaze processor out of your block design and restart the process of adding the processor. This is to say, the settings chosen here are important. Getting them right the first time will save you time in the long run.

Important! When working with multiple clocks in a design (as happens to always be the case when working with the MIG) it is important to verify that you are picking the correct *Clock Connection* from the dropdown. In the screenshot to the right, the 100 MHz () clk_out1 from a clocking wizard is chosen. You may instead wish to run your design off of ui_clk itself. Do not select the system clock input to the MIG.

Click **OK** to continue.

Next, the MIG's AXI interface must be connected to Microblaze's cache ports, in order to allow data to move back and forth between processor and DDR memory.

To connect the MIG's AXI interface to the processor, click the **Run Connection Automation** button in the green banner at the top of the block design.

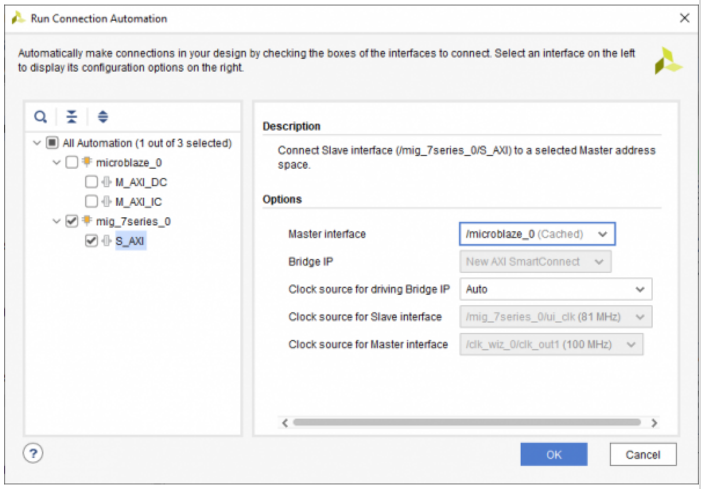


```
regenerate_bd_layout
delete_bd_objs [get_bd_nets microblaze_0_Clk]
regenerate_bd_layout
```

Type a Tcl command here

(https://digilent.com/reference/_detail/programmable-logic/guides/run-connection-automation.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

In the dialog that pops up, you will be presented with several options for interfaces that can be connected to other interfaces. Both the Microblaze's IC and DC ports, as well as the MIG's AXI port will appear. You should only run connection automation for one side of the connection - the S_AXI port, as shown in the screenshot to the right. Make sure that its box is checked. Check that the *Master interface* is set to "/microblaze_0 (Cached)", indicating that the microblaze local memory will act as a cache for the DDR memory, then click **OK** to make the connections.



Automatically make connections in your design by checking the boxes of the interfaces to connect. Select an interface on the left to display its configuration options on the right.

All Automation (1 out of 3 selected)
- ☐ microblaze_0
  - ☐ M_AXI_DC
  - ☐ M_AXI_IC
- ☑ mig_7series_0
  - ☑ S_AXI

Description

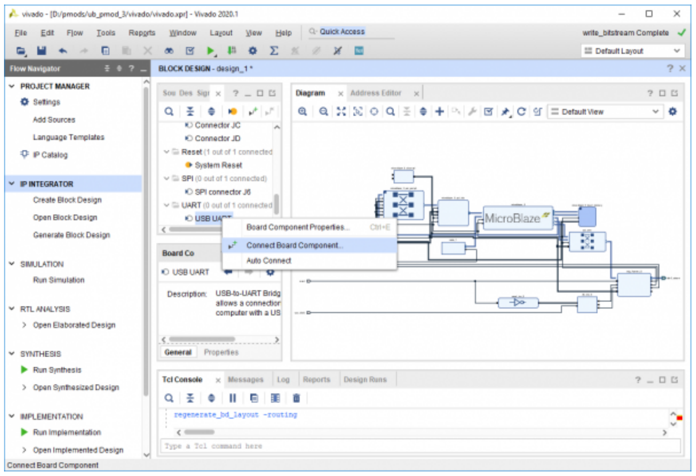
Connect Slave interface (/mig_7series_0/S_AXI) to a selected Master address space.

Options

Master interface	/microblaze_0 (Cached)
Bridge IP	New AXI SmartConnect
Clock source for driving Bridge IP	Auto
Clock source for Slave interface	/mig_7series_0/ui_clk (81 MHz)
Clock source for Master interface	/clk_wiz_0/clk_out1 (100 MHz)

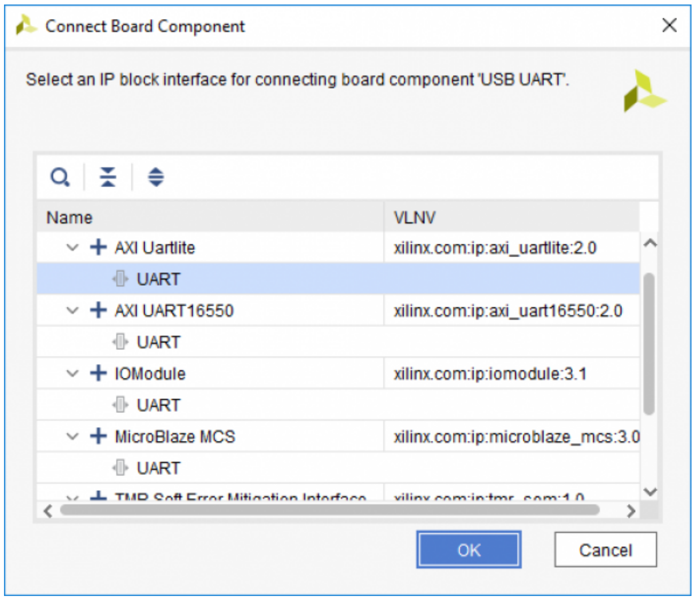
(https://digilent.com/reference/_detail/programmable-logic/guides/connection-automation.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

Next, in order for the software design to be able to print to a serial console on a host computer, a UART peripheral must be connected. Find the *USB UART* interface in the *Board* tab, right click on it, and select **Connect Board Component**.



(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/add-microblaze-processor/microblaze-18.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

In the dialog that pops up, select a new AXI Uartlite IP's UART interface, and click **OK** to add the block to the diagram.

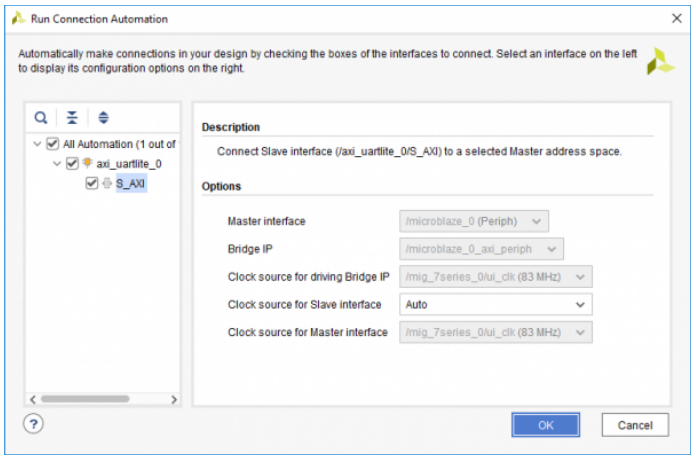

Connect Board Component

Select an IP block interface for connecting board component 'USB UART'.

Name	VLNV
AXI Uartlite	xilinx.com:ip:axi_uartlite:2.0
UART	
AXI UART16550	xilinx.com:ip:axi_uart16550:2.0
UART	
IOModule	xilinx.com:ip:iomodule:3.1
UART	
MicroBlaze MCS	xilinx.com:ip:microblaze_mcs:3.0
UART	
TMR Soft Error Mitigation Interface	xilinx.com:ip:tmr_sem:1.0

(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/add-microblaze-processor/microblaze-19.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

To connect all remaining AXI cores that have not yet been connected to the processor, click the **Run Connection Automation** button in the green *Designer Assistance* bar. Check the *All Automation* box in the list on the left side of the window to select all of the remaining AXI cores. The dropdown settings available for each core can safely be left as their default values. Click **OK** to automatically connect them to the processor.


Run Connection Automation

Automatically make connections in your design by checking the boxes of the interfaces to connect. Select an interface on the left to display its configuration options on the right.

- [✓] All Automation (1 out of
  - [✓] axi_uartlite_0
    - [✓] S_AXI

Description

Connect Slave interface (/axi_uartlite_0/S_AXI) to a selected Master address space.

Options

Master interface	/microblaze_0 (Periph)
Bridge IP	/microblaze_0_axi_periph
Clock source for driving Bridge IP	/mig_7series_0/ui_clk (83 MHz)
Clock source for Slave interface	Auto
Clock source for Master interface	/mig_7series_0/ui_clk (83 MHz)

(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/add-microblaze-processor/microblaze-20.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

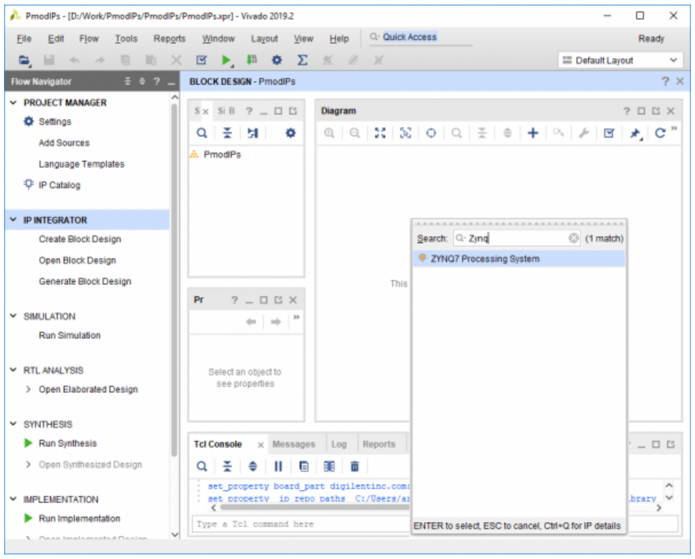
Add a Zynq Processor to a Block Design

The Zynq7 Processing System IP represents the non-FPGA components of a Zynq chip, referred to as the Processing System, or PS. It must be used in a block design that wants to connect anything to the processor, and to configure PS-side peripherals, clocks, and other settings.

Note: *This section only applies to boards with a 7-series Zynq chip.*

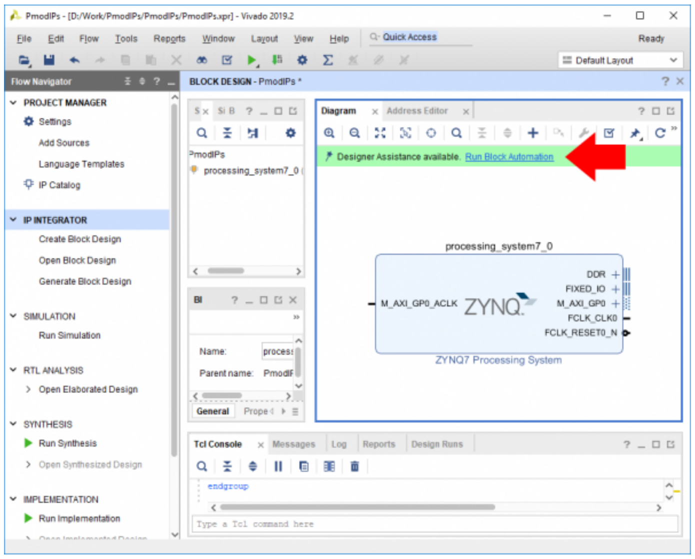
In the block diagram pane's toolbar, click the **Add IP** button (➕).

In the pop up, search for and double click on **ZYNQ7 Processing System**.



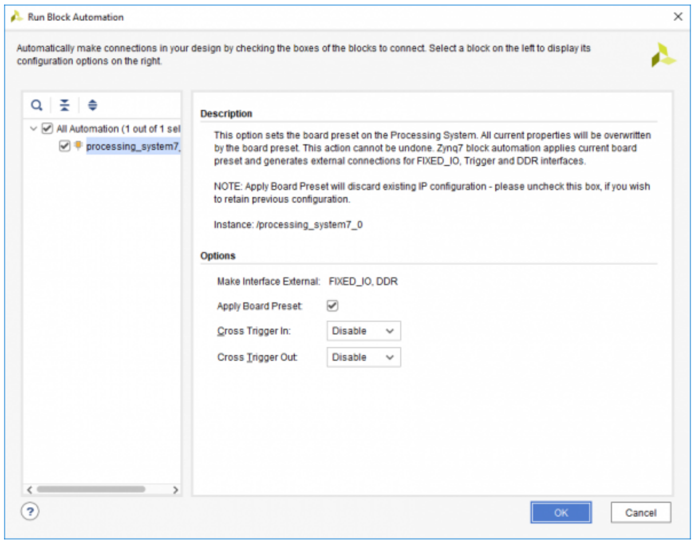
(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/add-zynq-processor/add-zynq.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

Click **Run Block Automation** in the Design Assistance banner (the green bar).



(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/add-zynq-processor/run-block-automation-1.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

In the dialog that pops up, leave all settings as defaults. *Apply Board Preset* should be checked.



Automatically make connections in your design by checking the boxes of the blocks to connect. Select a block on the left to display its configuration options on the right.

Description

This option sets the board preset on the Processing System. All current properties will be overwritten by the board preset. This action cannot be undone. Zynq7 block automation applies current board preset and generates external connections for FIXED_IO, Trigger and DDR interfaces.

NOTE: Apply Board Preset will discard existing IP configuration - please uncheck this box, if you wish to retain previous configuration.

Instance: /processing_system7_0

Options

Make Interface External:	FIXED_IO, DDR
Apply Board Preset	✓
Cross Trigger In:	Disable
Cross Trigger Out	Disable

(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/add-zynq-processor/run-block-automation-2.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

The needs of your project may require that you change some of the default settings of the Zynq PS. To edit its settings, double click on it to open the configuration wizard.

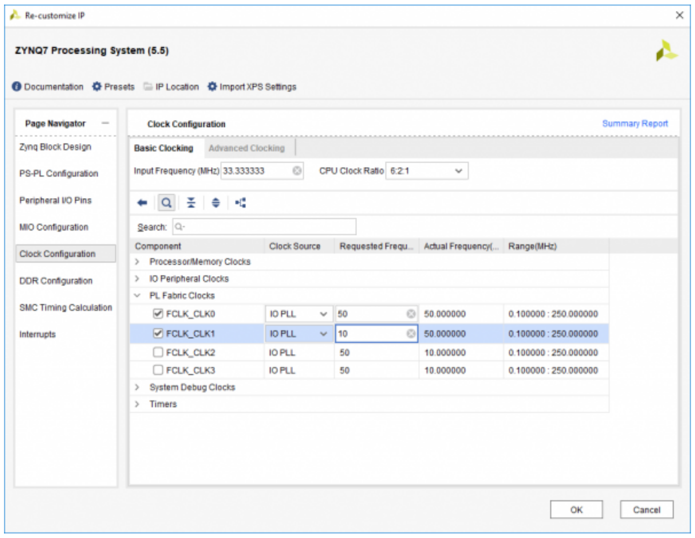
Two specific cases are highlighted below:

The Zynq PS can generate multiple clocks that are then provided to the FPGA fabric. These clocks are referred to as FCLKs, and can be found in the **Clock Configuration** tab of the Zynq PS configuration wizard. They are located under the *PL Fabric Clocks* dropdown. They can be enabled (or disabled) with a checkbox, the hardware used to drive the clock can be changed, and the frequency can be modified.

All board files for Digilent Zynq boards enable a single Zynq PL clock by default, which is intended to be used with peripherals connected to the Zynq's M_AXI_GP0 port.

Some designs may require additional clocks of specific frequencies be added to your design. In these cases, enable a second clock and specify the needed frequency, as seen in the image to the right.

Note: *This section can always be returned to later, as the addition of an additional clock can be performed any time before the hardware is built.*



Clock Configuration

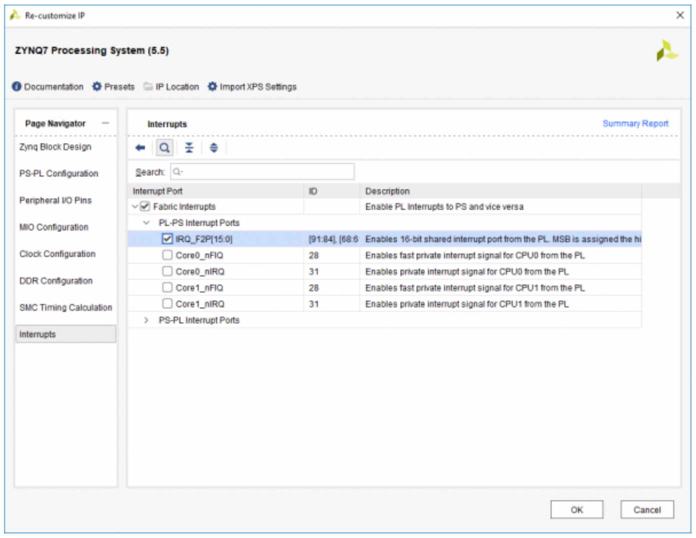
Input Frequency (MHz) 33.333333 CPU Clock Ratio 6:2:1

Search:

Component	Clock Source	Requested Frequ...	Actual Frequency(...	Range(MHz)
Processor/Memory Clocks				
IO Peripheral Clocks				
PL Fabric Clocks				
✓ FCLK_CLK0	IO PLL	50	50.000000	0.100000 : 250.000000
✓ FCLK_CLK1	IO PLL	10	50.000000	0.100000 : 250.000000
FCLK_CLK2	IO PLL	50	10.000000	0.100000 : 250.000000
FCLK_CLK3	IO PLL	50	10.000000	0.100000 : 250.000000
System Debug Clocks				
Timers				

(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/add-zynq-processor/add-additional-clock.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

Zynq devices can also use interrupts generated in FPGA fabric to trigger interrupts within the Processing System. Interrupt-related settings can be changed within the configuration wizard's interrupts tab. These interrupts typically use the IRQ_F2P port, which can be found under the Fabric Interrupts → IRQ_F2P dropdown. To enable this port, both the Fabric Interrupts and IRQ_F2P ports must be enabled.

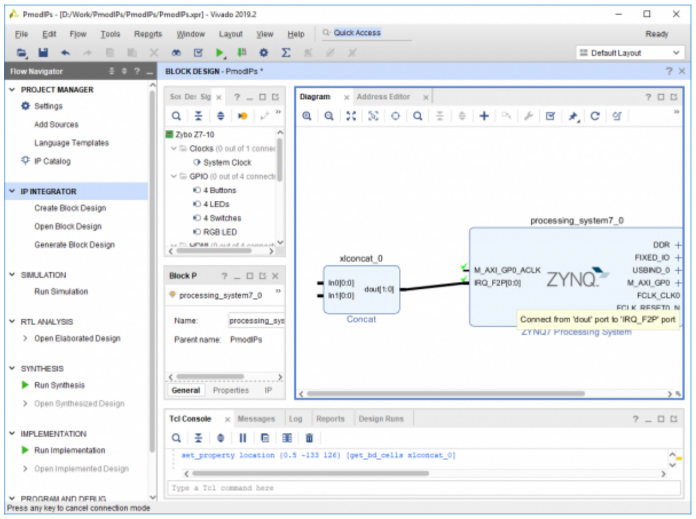


Interrupts

Interrupt Port	ID	Description
✓ Fabric Interrupts		Enable PL Interrupts to PS and vice versa
PL-PS Interrupt Ports		
✓ IRQ_F2P[15:0]	[91:84], [68:6	Enables 16-bit shared interrupt port from the PL. MSB is assigned the hi
☐ Core0_nFIQ	28	Enables fast private interrupt signal for CPU0 from the PL
☐ Core0_nIRQ	31	Enables private interrupt signal for CPU0 from the PL
☐ Core1_nFIQ	28	Enables fast private interrupt signal for CPU1 from the PL
☐ Core1_nIRQ	31	Enables private interrupt signal for CPU1 from the PL
PS-PL Interrupt Ports		

(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/add-zynq-processor/add-zynq-interrupt.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

While interrupts can be directly connected to the IRQ_F2P port (by clicking and dragging from one port to another), some designs may require multiple interrupt sources. In these cases, add a **Concat** IP to your block design, and manually connect it to the IRQ_F2P port. Additional input ports can be added to a Concat block through its configuration wizard (opened by double clicking on the IP).



```
set_property location {0.5 -133 126} [get_bd_cells xlconcat_0]
```

(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/add-zynq-processor/add-zynq-interrupt-concat.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

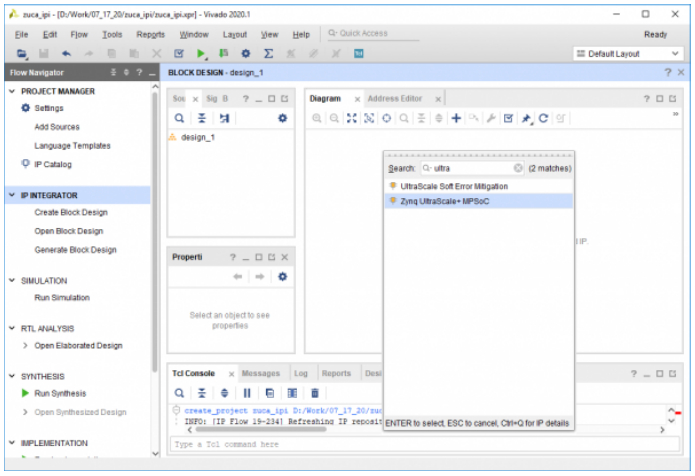
Add a Zynq UltraScale+ Processor to a Block Design

The Zynq UltraScale+ MPSoC IP represents the non-FPGA components of a Zynq UltraScale chip, referred to as the Processing System, or PS. It must be used in a block design that wants to connect anything to the processor, and to configure PS-side peripherals, clocks, and other settings.

Note: *This section only applies to boards with a Zynq UltraScale+ chip.*

In the block diagram pane's toolbar, click the **Add IP** button (+).

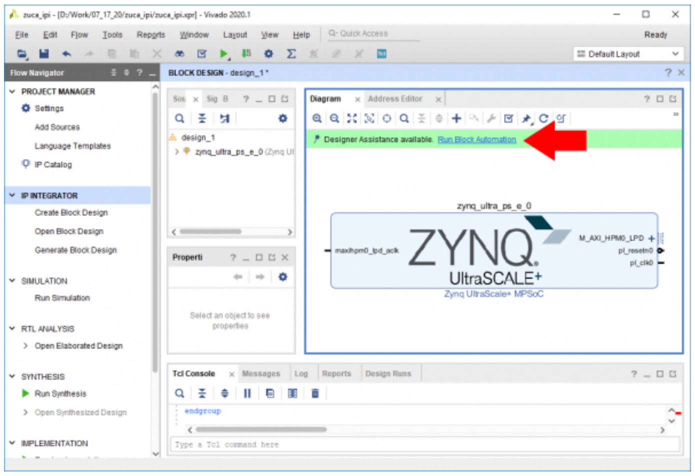
In the pop up, search for and double click on **Zynq UltraScale+ MPSoC**.


Search: ultra (2 matches)
UltraScale Soft Error Mitigation
Zynq UltraScale+ MPSoC
ENTER to select, ESC to cancel, Ctrl+Q for IP details

```
create_project zuca_ipi D:/Work/07_17_20/zuc
INFO: [IP Flow 19-234] Refreshing IP reposit
```

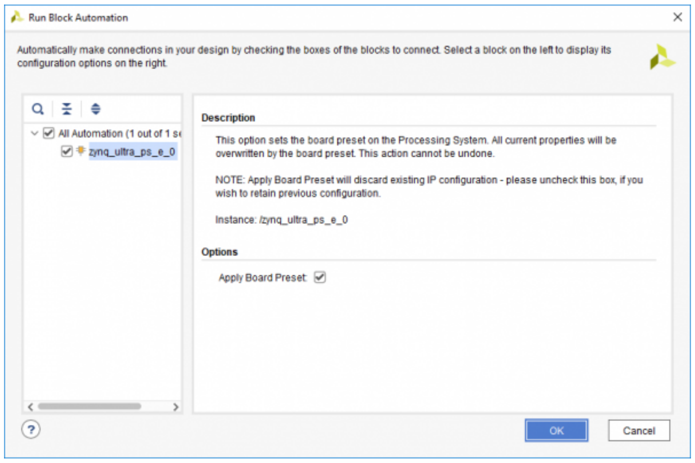
(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/add-ultrascale-zynq-processor/add-zynq-ultra.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

Click **Run Block Automation** in the Design Assistance banner (the green bar).



(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/add-ultrascale-zynq-processor/run-block-automation-1.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

In the dialog that pops up make sure *Apply Board Preset* is checked. This will apply the preset configuration from the board files to the IP, which saves a lot of time and prevents potential issues with doing the configuration entirely manually. Click **OK** to continue.


Run Block Automation

Automatically make connections in your design by checking the boxes of the blocks to connect. Select a block on the left to display its configuration options on the right.

All Automation (1 out of 1 se
zynq_ultra_ps_e_0

Description

This option sets the board preset on the Processing System. All current properties will be overwritten by the board preset. This action cannot be undone.

NOTE: Apply Board Preset will discard existing IP configuration - please uncheck this box, if you wish to retain previous configuration.

Instance: /zynq_ultra_ps_e_0

Options

Apply Board Preset ✓

OK Cancel

(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/add-ultrascale-zynq-processor/run-block-automation-2.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

The needs of your project may require that you change some of the default settings of the PS. To edit its settings, double click on it to open the configuration wizard.

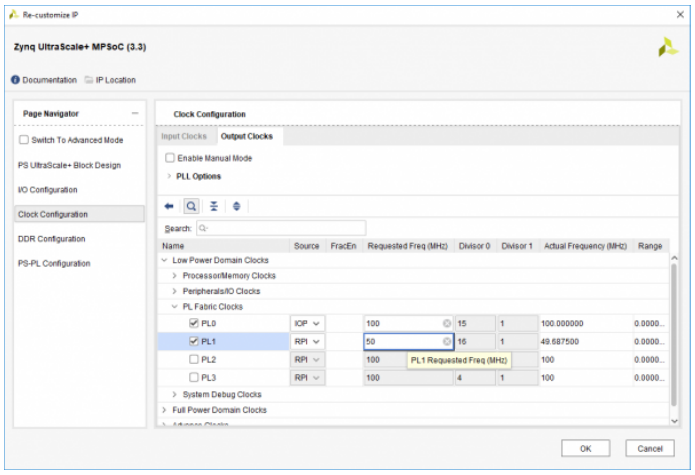
Two specific cases are highlighted below:

The PS can generate multiple clocks that are then provided to the FPGA fabric. These clocks are referred to as PL clocks, and can be found in the **Clock Configuration** tab of the MPSoC configuration wizard. They are located under the *Low Power Domain Clocks → PL Fabric Clocks* dropdowns. They can be enabled (or disabled) with a checkbox, the hardware source used to drive the clock can be changed, and the frequency can be modified.

Board files for Digilent Zynq UltraScale boards enable at least one low power domain PL clock by default, which is intended to be used with peripherals connected to the MPSoC's M_AXI_HPM0_LPD port.

Some designs may require additional clocks of specific frequencies be added to your design. In these cases, enable a second clock and specify the needed frequency, as seen in the image to the right.

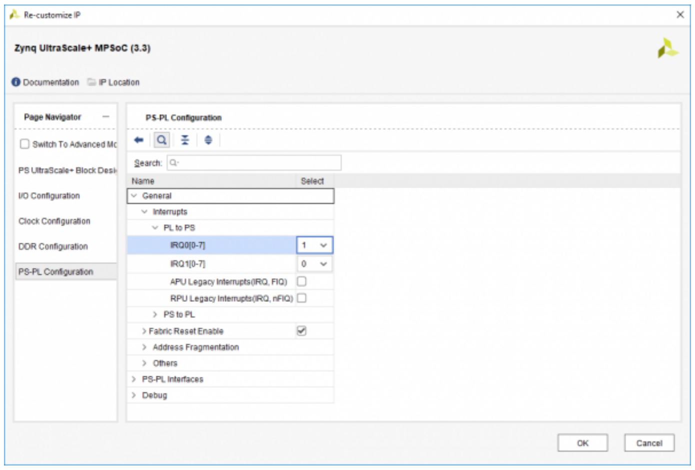
Note: *This section can always be returned to later, as the addition of an additional clock can be performed any time before the hardware is built.*



Name	Source	FracEn	Requested Freq (MHz)	Divisor 0	Divisor 1	Actual Frequency (MHz)	Range
Low Power Domain Clocks							
Processor/Memory Clocks							
Peripherals/IO Clocks							
PL Fabric Clocks							
✓ PL0	IOP		100	15	1	100.000000	0.0000...
✓ PL1	RPI		50	16	1	49.687500	0.0000...
PL2	RPI		100	PL1 Requested Freq (MHz)		100	0.0000...
PL3	RPI		100	4	1	100	0.0000...
System Debug Clocks							
Full Power Domain Clocks							

(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/add-ultrascale-zynq-processor/add-additional-clock.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

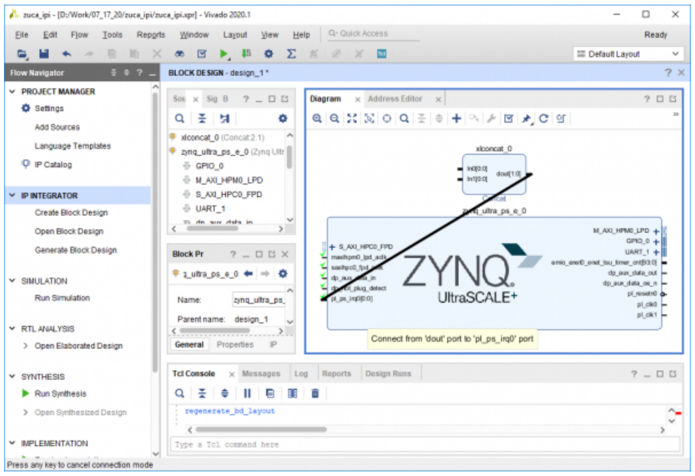
UltraScale devices can also use interrupts generated in FPGA fabric to trigger interrupts within the Processing System. Interrupt-related settings can be changed within the configuration wizard's **PS-PL Configuration** tab. These interrupts can use the IRQ0 port, which can be found under the *General → Interrupts → PL to PS* dropdowns. To enable this port, the IRQ0 dropdown should be set to "1".



Name	Select
General	
Interrupts	
PL to PS	
IRQ0[0-7]	1
IRQ1[0-7]	0
APU Legacy Interrupts(IRQ, FIQ)	
RPU Legacy Interrupts(IRQ, nFIQ)	
PS to PL	
Fabric Reset Enable	✓
Address Fragmentation	
Others	
PS-PL Interfaces	
Debug	

(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/add-ultrascale-zynq-processor/add-interrupt.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

While interrupts can be directly connected to the pl_ps_irq0 (IRQ0) port by clicking and dragging from one port to another, some designs may require multiple interrupt sources. In these cases, add a **Concat** IP to your block design, and manually connect it to the pl_ps_irq0 port. Additional input ports can be added to a Concat block through its configuration wizard (opened by double clicking on the IP).



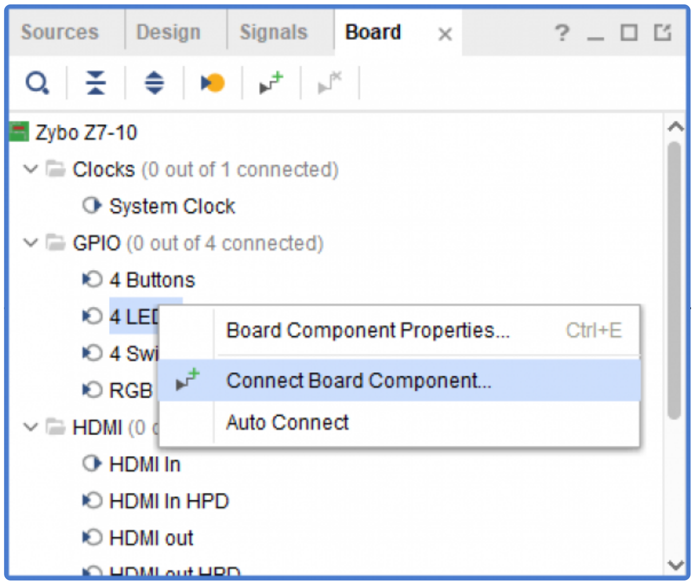
(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/add-ultrascale-zynq-processor/add-interrupt-concat.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

Add GPIO Peripherals to a Block Design

This section covers the steps involved in adding a GPIO () peripheral to a block design. While an AXI GPIO () IP is used, other IPs and interfaces can potentially be added to your design in the same ways. Two methods are presented here, one for each of the two AXI GPIO () peripherals that will be connected. The first takes advantage of board files to automatically generate constraints, the second presents how an IP interface can be connected to chosen pins manually.

Interfaces for Digilent boards supported by the board files can be found in the *Board* tab. For the purposes of this guide, find the GPIO () section of the list, right click on an LED () interface, and select the **Connect Board Interface** option.

Note: *If your board does not have single-color LEDs, you can use it's RGB LEDs instead. Note that these interfaces have three pins for each LED (), for the R, G, and B components.*

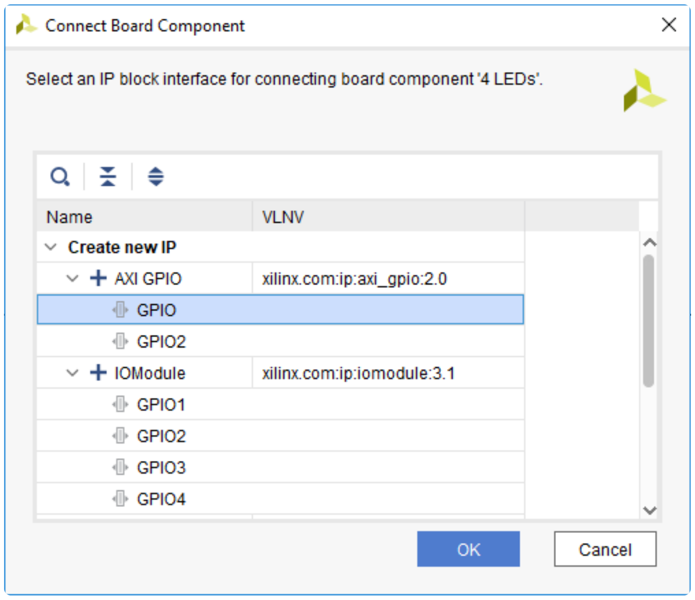


(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/add-gpio-to-block-design/add-ip-from-board.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

In the dialog that pops up, choose the "GPIO ()" interface (not GPIO2) of a new AXI GPIO () IP.

Some boards use one of their user buttons as reset sources. In these cases, make sure to choose the *Component mode* that does not include the reset button.

Click **OK** to continue. This will add the IP to your design, and connect it to an external port, which will not require any further work to constrain.



Connect Board Component

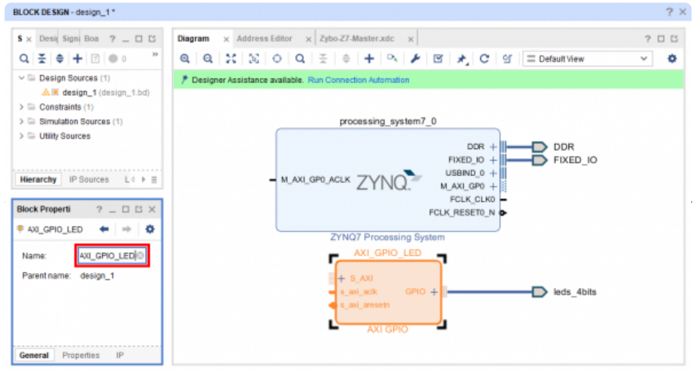
Select an IP block interface for connecting board component '4 LEDs'.

Name	VLNV
Create new IP	
AXI GPIO	xilinx.com:ip:axi_gpio:2.0
GPIO	
GPIO2	
IOModule	xilinx.com:ip:iomodule:3.1
GPIO1	
GPIO2	
GPIO3	
GPIO4	

OK Cancel

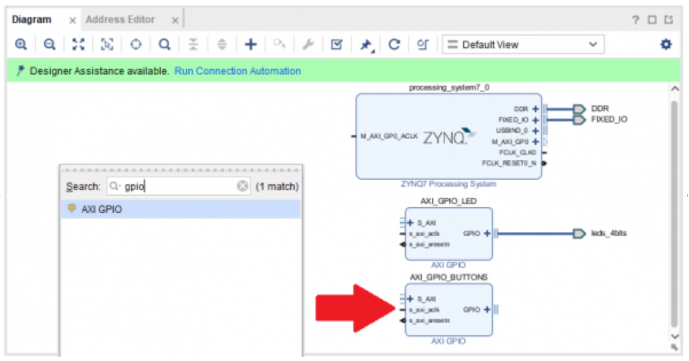
(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/add-gpio-to-block-design/connect-board-component.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

Next, select the axi_gpio_0 block. The *Block Properties* pane to the left of the Diagram and below the Board tab will allow you to view some information about the block, and modify it in some ways, without running through its customization wizard. For now, just change its name to "AXI_GPIO ()_LED ()" by typing in the *Name* field. Pressing enter or clicking out of the text box confirms the change. Using memorable names in your block design makes it easier to remember which IP does what when you are later writing software in Vitis.



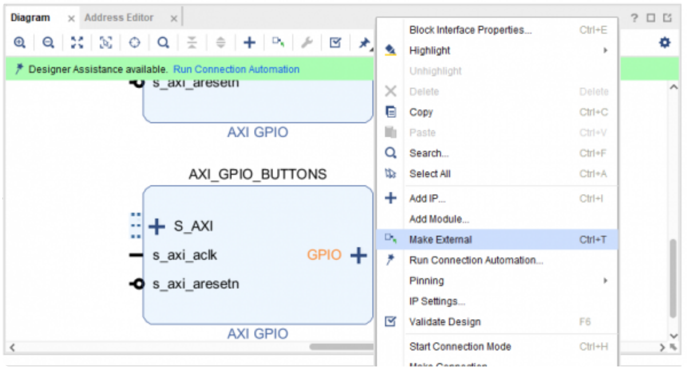
(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/add-gpio-to-block-design/rename-ip.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

Next, a second AXI GPIO () IP will be manually added to the block diagram, and manually constrained with an XDC file. Click the **Add IP** button (+) and search for "AXI GPIO ()". Double click on the only result to add the second AXI GPIO () block to the design. Once added, rename this IP "AXI_GPIO ()_BUTTONS"



(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/add-gpio-to-block-design/add-ip.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

Select the AXI_GPIO ()_BUTTONS IP's GPIO () interface by clicking on the text "GPIO ()", right click on the highlighted text, and select **Make External**. This option creates a new external interface port that does not rely on the board files. Because the board files are not used here, a Xilinx Design Constraint (XDC) file must be added to the project to tell Vivado which FPGA pins to connect the interface to.

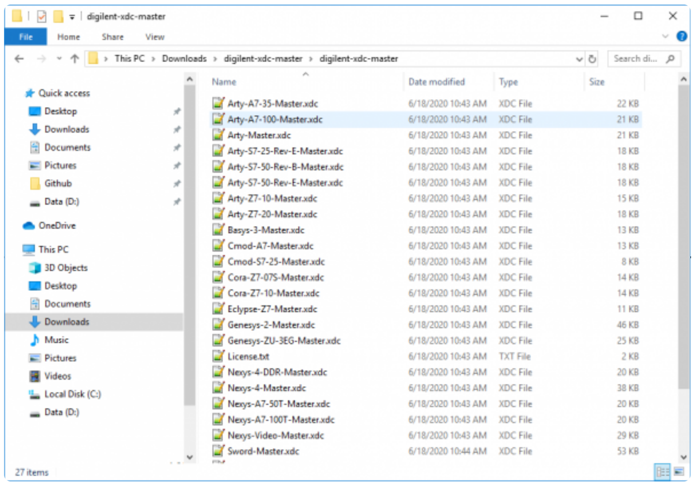


(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/add-gpio-to-block-design/make-external.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

If your project doesn't contain the master Xilinx Design Constraint (XDC) file for your board, the dropdown below details how to add it. This file contains the constraints that your board places on designs using it - specific interfaces wired up to specific pins, clock frequencies, and FPGA bank voltages, for some examples. Click the dropdown below for a walkthrough of how to add this file to your project.

Add a Master XDC File to a Vivado Project

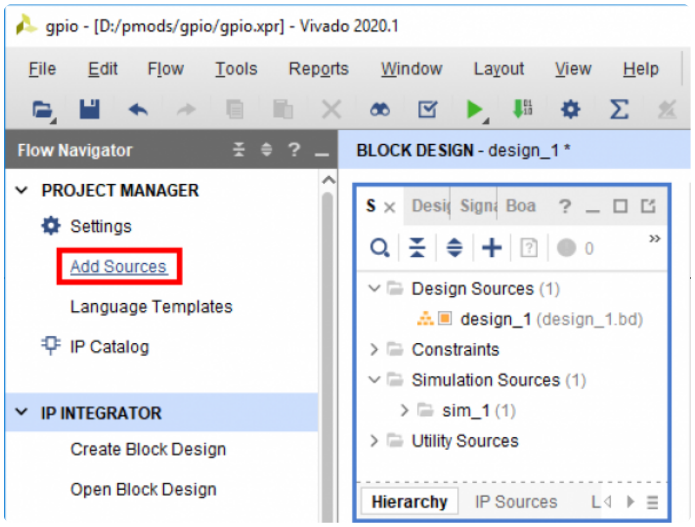
Download and extract digilent-xdc-master.zip (https://digilent.com/reference/lib/exe/fetch.php?tok=a49a20&media=https%3A%2F%2Fgithub.com%2FDigilent%2Fdigilent-xdc%2Farchive%2Fmaster.zip). This file includes all of the latest template XDC files released for Digilent's boards, which are available on Github in the digilent-xdc (https://github.com/Digilent/digilent-xdc) repository.



Name	Date modified	Type	Size
Arty-A7-35-Master.xdc	6/18/2020 10:43 AM	XDC File	22 KB
Arty-A7-100-Master.xdc	6/18/2020 10:43 AM	XDC File	21 KB
Arty-Master.xdc	6/18/2020 10:43 AM	XDC File	21 KB
Arty-S7-25-Rev-E-Master.xdc	6/18/2020 10:43 AM	XDC File	18 KB
Arty-S7-50-Rev-B-Master.xdc	6/18/2020 10:43 AM	XDC File	18 KB
Arty-S7-50-Rev-E-Master.xdc	6/18/2020 10:43 AM	XDC File	18 KB
Arty-Z7-10-Master.xdc	6/18/2020 10:43 AM	XDC File	15 KB
Arty-Z7-20-Master.xdc	6/18/2020 10:43 AM	XDC File	18 KB
Basys-3-Master.xdc	6/18/2020 10:43 AM	XDC File	13 KB
Cmod-A7-Master.xdc	6/18/2020 10:43 AM	XDC File	13 KB
Cmod-S7-25-Master.xdc	6/18/2020 10:43 AM	XDC File	8 KB
Cora-Z7-07S-Master.xdc	6/18/2020 10:43 AM	XDC File	14 KB
Cora-Z7-10-Master.xdc	6/18/2020 10:43 AM	XDC File	14 KB
Eclypse-Z7-Master.xdc	6/18/2020 10:43 AM	XDC File	11 KB
Genesys-2-Master.xdc	6/18/2020 10:43 AM	XDC File	46 KB
Genesys-ZU-3EG-Master.xdc	6/18/2020 10:43 AM	XDC File	25 KB
License.txt	6/18/2020 10:43 AM	TXT File	2 KB
Nexys-4-DDR-Master.xdc	6/18/2020 10:43 AM	XDC File	20 KB
Nexys-4-Master.xdc	6/18/2020 10:43 AM	XDC File	38 KB
Nexys-A7-50T-Master.xdc	6/18/2020 10:43 AM	XDC File	20 KB
Nexys-A7-100T-Master.xdc	6/18/2020 10:43 AM	XDC File	20 KB
Nexys-Video-Master.xdc	6/18/2020 10:43 AM	XDC File	29 KB
Sword-Master.xdc	6/18/2020 10:44 AM	XDC File	53 KB

(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/add-gpio-to-block-design/extracted-xdc-folder.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

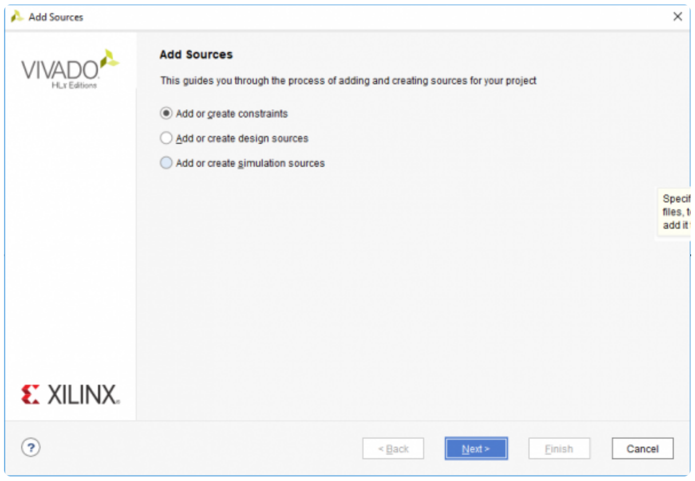
Returning to Vivado, click the **Add Sources** button in the *Project Manager* section of the *Flow Navigator* pane. This will launch a dialog that you can use to add a variety of types of source files to the project (or create new ones).



(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/add-gpio-to-block-design/add-sources.png?

id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

On the first screen, select *Add or create constraints*. Click **Next** to continue.



Add Sources

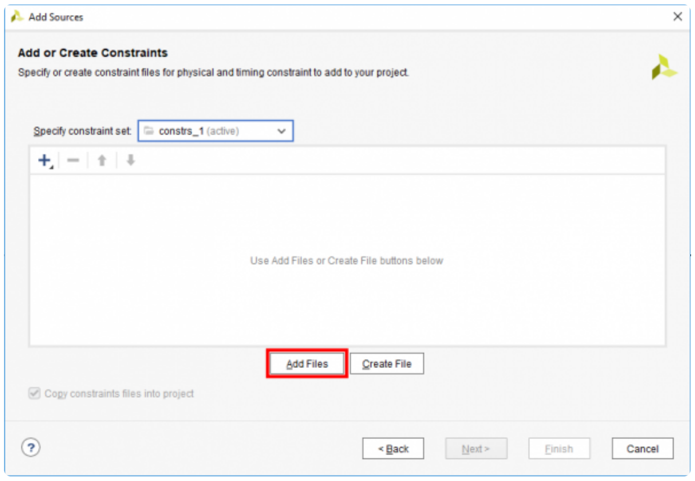
This guides you through the process of adding and creating sources for your project

(●) Add or create constraints
( ) Add or create design sources
( ) Add or create simulation sources

Specif
files, t
add it

(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/add-gpio-to-block-design/add-constraints.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

In the next screen, make sure that the constraint set specified (the one that the master XDC will be added to) is set to *constrs_1*, and that it is the *active* set. Click the **Add Files** button.



Add or Create Constraints

Specify or create constraint files for physical and timing constraint to add to your project.

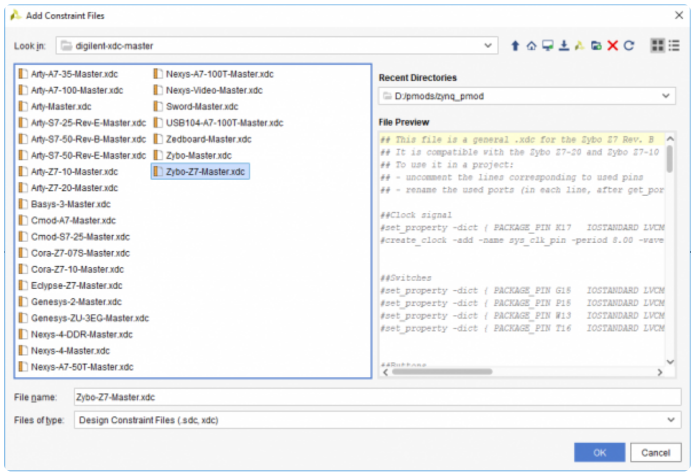
Specify constraint set: constrs_1 (active)

Use Add Files or Create File buttons below

[✓] Copy constraints files into project

(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/add-gpio-to-block-design/add-files.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

In the dialog that pops up, navigate to the folder that the *digilent-xdc-master.zip* file was extracted into. Highlight the XDC file for your board, then click **OK** to continue.



Arty-A7-35-Master.xdc	Nexys-A7-100T-Master.xdc
Arty-A7-100-Master.xdc	Nexys-Video-Master.xdc
Arty-Master.xdc	Sword-Master.xdc
Arty-S7-25-Rev-E-Master.xdc	USB104-A7-100T-Master.xdc
Arty-S7-50-Rev-B-Master.xdc	Zedboard-Master.xdc
Arty-S7-50-Rev-E-Master.xdc	Zybo-Master.xdc
Arty-Z7-10-Master.xdc	Zybo-Z7-Master.xdc
Arty-Z7-20-Master.xdc	
Basys-3-Master.xdc	
Cmod-A7-Master.xdc	
Cmod-S7-25-Master.xdc	
Cora-Z7-07S-Master.xdc	
Cora-Z7-10-Master.xdc	
Eclypse-Z7-Master.xdc	
Genesys-2-Master.xdc	
Genesys-ZU-3EG-Master.xdc	
Nexys-4-DDR-Master.xdc	
Nexys-4-Master.xdc	
Nexys-A7-50T-Master.xdc	

Recent Directories: D:/pmods/zynq_pmod

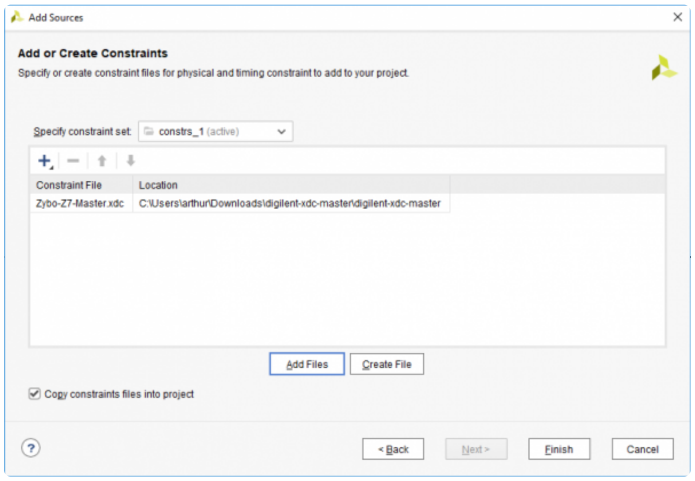
```
## This file is a general .xdc for the Zybo Z7 Rev. B
## It is compatible with the Zybo Z7-20 and Zybo Z7-10
## To use it in a project:
## - uncomment the lines corresponding to used pins
## - rename the used ports (in each line, after get_por

##Clock signal
#set_property -dict { PACKAGE_PIN K17   IOSTANDARD LVCM
#create_clock -add -name sys_clk_pin -period 8.00 -wave


##Switches
#set_property -dict { PACKAGE_PIN G15   IOSTANDARD LVCM
#set_property -dict { PACKAGE_PIN P15   IOSTANDARD LVCM
#set_property -dict { PACKAGE_PIN W13   IOSTANDARD LVCM
#set_property -dict { PACKAGE_PIN T16   IOSTANDARD LVCM
```

File name: Zybo-Z7-Master.xdc
Files of type: Design Constraint Files (.sdc, .xdc)

(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/add-gpio-to-block-design/find-xdc.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

Back in the *Add Sources* dialog, make sure that your chosen constraint file appears in the table. Also, make sure that the *Copy constraint files into project* box is checked. If this box is unchecked, the file will be linked by your project, and any modifications made within the project will affect the version you downloaded. Since you may need to use this file again in other projects, copying the constraint file is recommended, so that you can always work from a fresh copy.

Click **Finish** to add the constraint file to your project.



Add or Create Constraints
Specify or create constraint files for physical and timing constraint to add to your project.

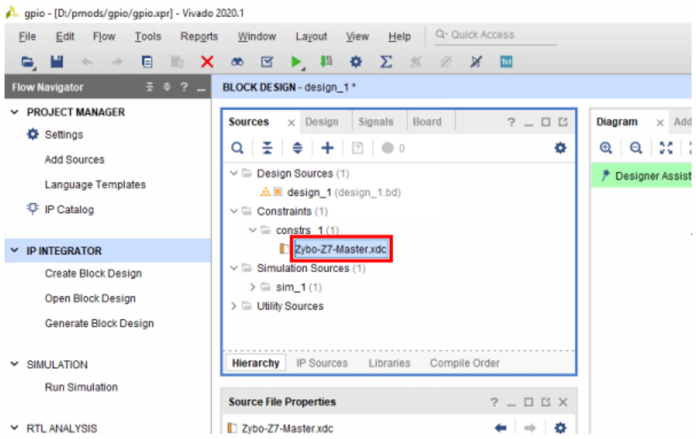
Specify constraint set: constrs_1 (active)

Constraint File	Location
Zybo-Z7-Master.xdc	C:\Users\arthur\Downloads\digilent-xdc-master\digilent-xdc-master

☑ Copy constraints files into project

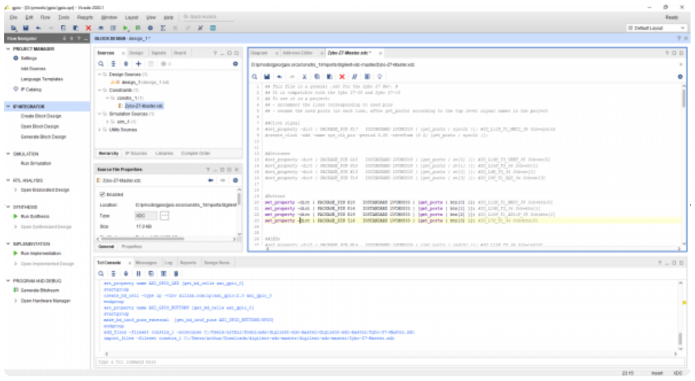
(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/add-gpio-to-block-design/import-constraint-file.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

Once added, the XDC file will appear in the *Sources* tab (in the same pane as the *Board* tab). Double click it to open the file.



(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/add-gpio-to-block-design/open-constraint.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

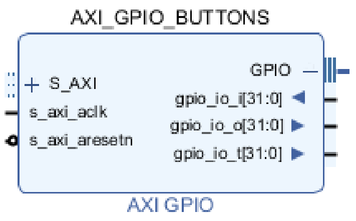
Master XDC files for Digilent boards contain pin constraints for I/O interfaces the board offers. These constraints are sorted by interface. Scroll down until you see constraints for the user buttons. These constraints typically are for a bus port named “btn”. Un-comment the button constraints by removing the single leading '#' character in each line corresponding to the buttons, as seen in the screenshot to the right.



(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/add-gpio-to-block-design/uncomment-constraints.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

Next, the names of the block design's GPIO () port for the buttons must be determined, so that the buttons can be properly constrained. Reopen the *Diagram* tab, and select the GPIO ()_0 external port that is connected to the AXI_GPIO ()_BUTTONS block. Change the name of the external interface to “btn” in the *Properties* pane.

The AXI GPIO () IP automatically uses tri-state buffers for the pins its interfaces are connected to. The individual I, O, and T buses can be seen when expanding the interface through the plus button (+) next to the interface name on the IP block. As can be seen, the individual ports that make up the interface are named <interface>_tri_i, <interface>_tri_o, and <interface>_tri_t. When constrained to tristate buffers, the bus that is connected to FPGA ports is named <interface>_tri_io.



(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/add-gpio-to-block-design/view-interface-ports.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

With this knowledge, return to the XDC file, and change the name of the button bus that is constrained. Specifically, change the text after the “get_ports” call on each line of the button interface to “btn_tri_io[#]”, where # is a decimal number, counting up from zero. When finished, save the file.

With the constraints for the port finished, the AXI GPIO () must be manually configured. In particular, the width of the GPIO () interface must match the number of buttons available on the board. Take note of how many buttons are constrained in the XDC.

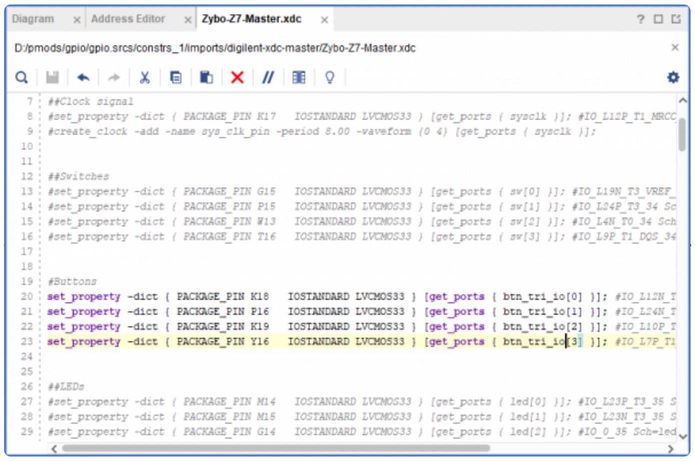


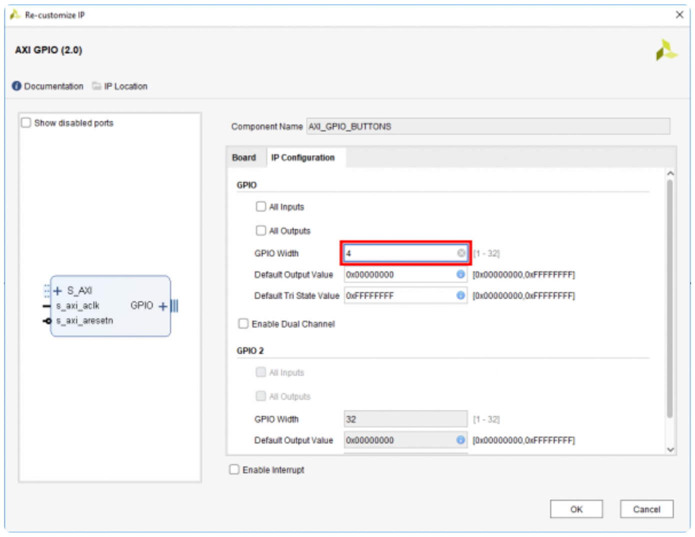
Diagram
Address Editor
Zybo-Z7-Master.xdc
D:/pmods/gpio/gpio.srcs/constrs_1/imports/digilent-xdc-master/Zybo-Z7-Master.xdc

```
##Clock signal
#set_property -dict { PACKAGE_PIN K17   IOSTANDARD LVCMOS33 } [get_ports { sysclk }]; #IO_L12P_T1_MRCC
#create_clock -add -name sys_clk_pin -period 8.00 -waveform {0 4} [get_ports { sysclk }];


##Switches
#set_property -dict { PACKAGE_PIN G15   IOSTANDARD LVCMOS33 } [get_ports { sw[0] }]; #IO_L19N_T3_VREF_
#set_property -dict { PACKAGE_PIN P15   IOSTANDARD LVCMOS33 } [get_ports { sw[1] }]; #IO_L24P_T3_34 Sc
#set_property -dict { PACKAGE_PIN W13   IOSTANDARD LVCMOS33 } [get_ports { sw[2] }]; #IO_L4N_T0_34 Sch
#set_property -dict { PACKAGE_PIN T16   IOSTANDARD LVCMOS33 } [get_ports { sw[3] }]; #IO_L9P_T1_DQS_34


#Buttons
set_property -dict { PACKAGE_PIN K18   IOSTANDARD LVCMOS33 } [get_ports { btn_tri_io[0] }]; #IO_L12N_T
set_property -dict { PACKAGE_PIN P16   IOSTANDARD LVCMOS33 } [get_ports { btn_tri_io[1] }]; #IO_L24N_T
set_property -dict { PACKAGE_PIN K19   IOSTANDARD LVCMOS33 } [get_ports { btn_tri_io[2] }]; #IO_L10P_T
set_property -dict { PACKAGE_PIN Y16   IOSTANDARD LVCMOS33 } [get_ports { btn_tri_io[3] }]; #IO_L7P_T1


##LEDs
#set_property -dict { PACKAGE_PIN M14   IOSTANDARD LVCMOS33 } [get_ports { led[0] }]; #IO_L23P_T3_35 S
#set_property -dict { PACKAGE_PIN M15   IOSTANDARD LVCMOS33 } [get_ports { led[1] }]; #IO_L23N_T3_35 S
#set_property -dict { PACKAGE_PIN G14   IOSTANDARD LVCMOS33 } [get_ports { led[2] }]; #IO_0_35 Sch=led
```

(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/add-gpio-to-block-design/change-port-name-constraints.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

Return to the *Diagram* tab, and double click on the *AXI_GPIO ()_BUTTONS* block. This will open a dialog that will allow you to configure the IP's settings. Switch to the configuration wizard's *IP Configuration* tab.

Only one setting need be changed for the purposes of this guide. Enter the number of buttons you constrained into the GPIO () interface's *GPIO () Width* field. When finished, click **OK** to save your changes.



Re-customize IP

AXI GPIO (2.0)

Documentation IP Location

Show disabled ports

Component Name AXI_GPIO_BUTTONS

Board | IP Configuration

GPIO

All Inputs

All Outputs

GPIO Width 4 [1 - 32]

Default Output Value 0x00000000 [0x00000000,0xFFFFFFFF]

Default Tri State Value 0xFFFFFFFF [0x00000000,0xFFFFFFFF]

Enable Dual Channel

GPIO 2

All Inputs

All Outputs

GPIO Width 32 [1 - 32]

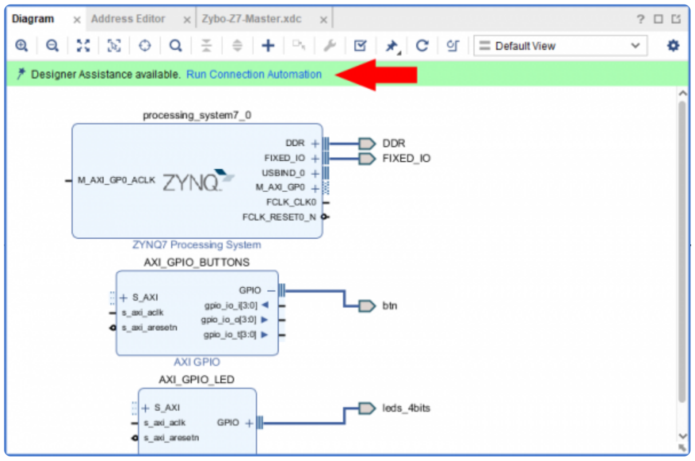
Default Output Value 0x00000000 [0x00000000,0xFFFFFFFF]

Enable Interrupt

OK Cancel

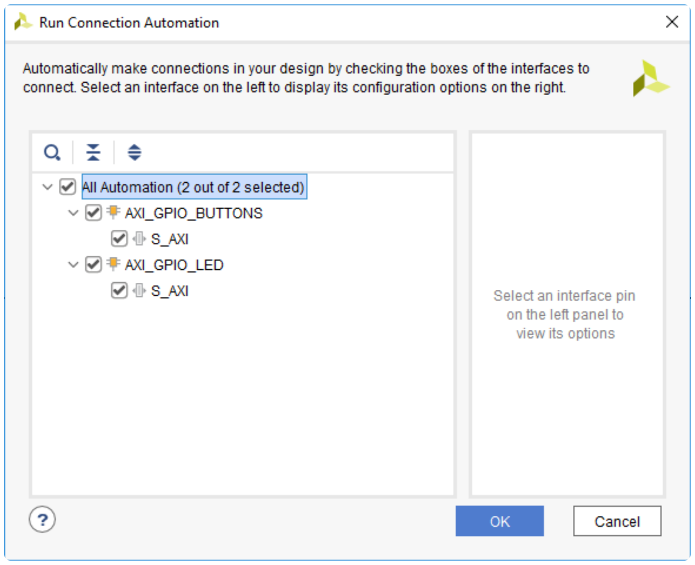
(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/add-gpio-to-block-design/change-gpio-width.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

Finally, the two AXI GPIO () IP blocks need to be connected to the processor in your design. Click the **Run Connection Automation** button in the green *Designer Assistance* bar.



(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/add-gpio-to-block-design/run-connection-automation.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

In the dialog that pops up, make sure that the boxes for the *S_AXI* interfaces for both of the AXI GPIO () IPs are checked. Click OK to run connection automation and connect the AXI GPIO () blocks to your processor.



Run Connection Automation

Automatically make connections in your design by checking the boxes of the interfaces to connect. Select an interface on the left to display its configuration options on the right.

- [✓] All Automation (2 out of 2 selected)
  - [✓] AXI_GPIO_BUTTONS
    - [✓] S_AXI
  - [✓] AXI_GPIO_LED
    - [✓] S_AXI

Select an interface pin on the left panel to view its options

(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/add-gpio-to-block-design/run-connection-automation-dialog.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

Edit the Address Map

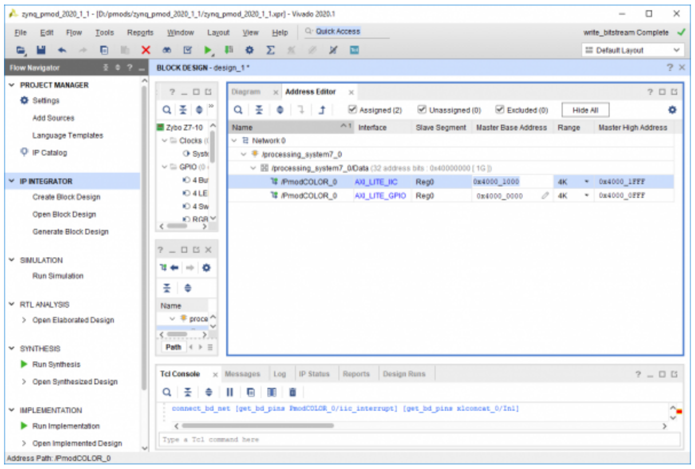
In the unlikely event that Vivado fails to correctly assign addresses to each AXI IP connected to your processor, you may need to manually set their addresses. If this occurs, errors will pop up during validation of the block design, and the bitstream will not be able to be generated.

The *Address Editor* can be accessed through its tab in the *Diagram* pane. Addresses can be assigned to unmapped peripherals by typing the desired address into the peripheral's *Master Base Address* column.

It should be noted that addresses must be aligned in the memory space - for instance, an address with a range of 4K (bytes) takes up a range of 0x1000 addresses, and must have three trailing zeros. Address ranges for different segments cannot overlap.

Assigning an segment to address 0 may result in assertions in some software drivers and should be avoided.

After manually assigning addresses, the block design should be re-validated.



zynq_pmod_2020_1_1 - [D:/pmods/zynq_pmod_2020_1_1/zynq_pmod_2020_1_1.xpr] - Vivado 2020.1

BLOCK DESIGN - design_1 *

Address Editor — Assigned (2) Unassigned (0) Excluded (0)

Name	Interface	Slave Segment	Master Base Address	Range	Master High Address
Network 0					
/processing_system7_0					
/processing_system7_0/Data (32 address bits : 0x40000000 [1G])					
/PmodCOLOR_0	AXI_LITE_IIC	Reg0	0x4000_1000	4K	0x4000_1FFF
/PmodCOLOR_0	AXI_LITE_GPIO	Reg0	0x4000_0000	4K	0x4000_0FFF

```
connect_bd_net [get_bd_pins PmodCOLOR_0/iic_interrupt] [get_bd_pins xlconcat_0/In1]
```

(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/address-editor/edit-address.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

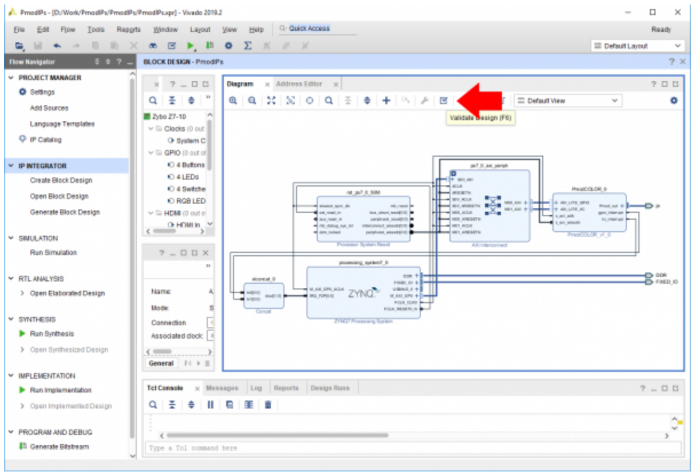
Validate a Block Design

Before the Vivado project can be built, the block design must be validated. This step runs an automatic check of the block design to see if there are any potential issues with it. Click the **Validate Design** button ((https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/validate-block-design/validate-button.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)) in the Diagram pane's toolbar (or press the F6 key).

If the design has issues, a dialog will pop up that lists them. It should be noted that most *Warnings* can be ignored, as can some *Critical Warnings*. These issues can also be viewed in the *Messages* tab of the pane at the bottom of the window.

If there are no issues, a dialog will pop up that will tell you so. Click **OK** to continue.

Note: *Some Zynq boards may produce critical warnings at this stage relating to PCW_UIPARAM_DDR_DQS_TO_CLK_DELAY parameters. These warnings are ignorable and will not affect the functionality of the project. See the Hardware Errata section of your board's reference manual for more information.*



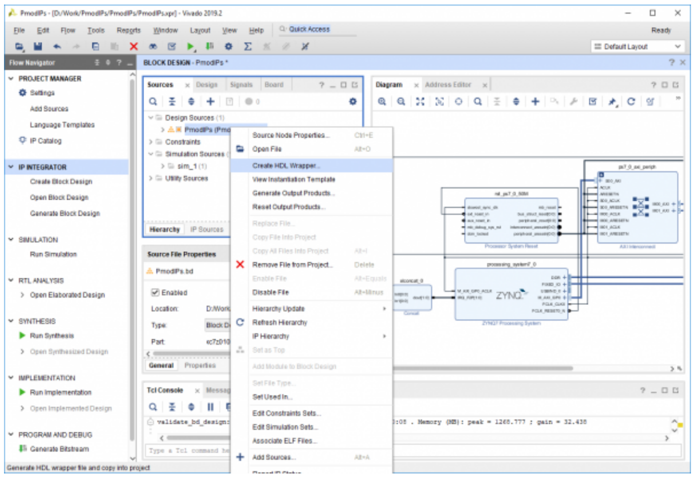
(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/validate-block-design/validate-design.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

Create an HDL Wrapper

Additionally, an HDL wrapper must be created for the block design. This process translates the block design into a source file that can be read by the Vivado tools, and is used to build the actual design.

Open the *Sources* pane and locate the block design file (.bd) under the *Design Sources* dropdown. Right click on it and select **Create HDL Wrapper**.

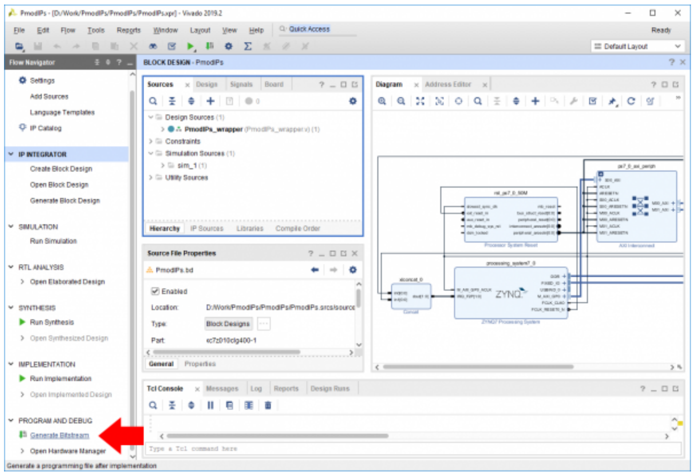
In the dialog that pops up, you can decide whether to let Vivado edit the wrapper file itself. *Let Vivado manage wrapper and auto-update* is recommended, as a user rarely needs to manually edit the wrapper file. Click **OK** to continue.



(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/validate-block-design/create-hdl-wrapper.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

Build a Vivado Project

At this point, the Vivado Project is ready to be built, by running it through Synthesis and Implementation, and finally generating a bitstream. Click the **Generate Bitstream** button in the *Program and Debug* section of the *Flow Navigator* pane at the left side of the window.

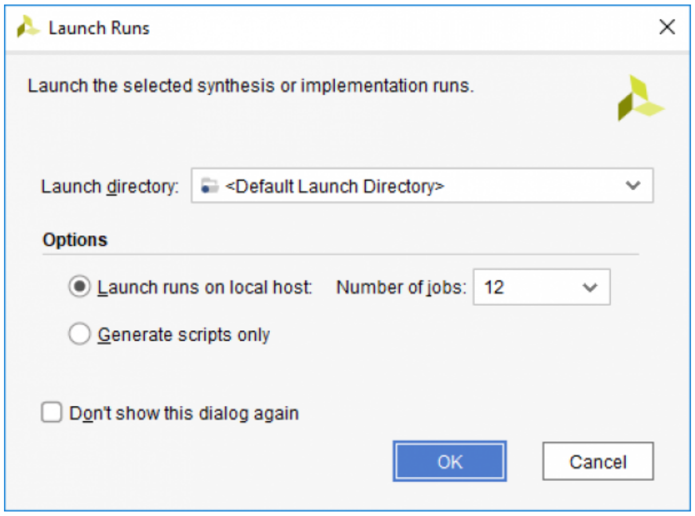


(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/generate-bitstream/generate-bitstream-1.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

A dialog will pop up with several options for how Synthesis and Implementation should be run. Most should be left as defaults. Of particular importance is the *Number of jobs* dropdown, which is used to specify how much of the resources of your computer should be dedicated to the build. A larger number of jobs will dedicate more resources, which will allow the build to be completed faster. It is recommended to choose the highest available number.

Note: *Critical warnings about how IPs included within another IP were packaged with a different board value can be safely ignored. The same is true for warnings related to negative CK-to-DQS delays seen on some Zynq boards.*

Depending on the complexity of the design, the board used, and the strength of your computer, the process of building the project can take between 5 and 60 minutes.


Launch Runs

Launch the selected synthesis or implementation runs.

Launch directory: <Default Launch Directory>

Options

[✓ Launch runs on local host] Number of jobs: 12

Generate scripts only

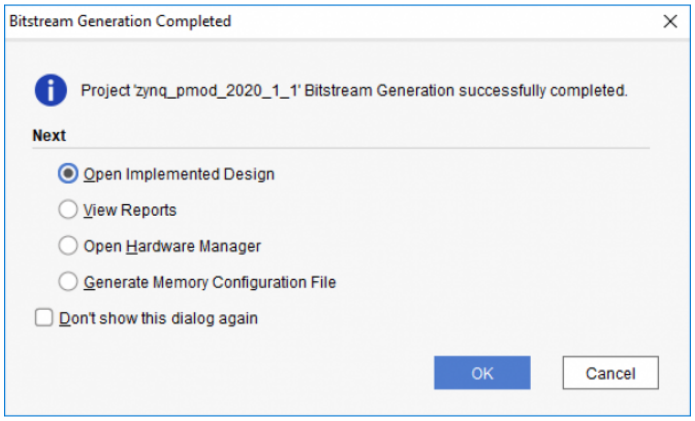
Don't show this dialog again

(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/generate-bitstream/generate-bitstream-2.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

When complete, a dialog will pop up that presents several options for what to do next:

- *Open Implemented Design* can be used to view the actual hardware design that has been implemented and will be placed onto the chip.
- *View Reports* can be used to view additional information about the design, including how much of the resources of the FPGA will be used by the design.
- *Open Hardware Manager* can be used to go directly to Vivado's Hardware Manager, which can be used to program a hardware design onto a board. This is typically used for designs that do not involve a software component.
- *Generate Memory Configuration File* can be used to create a file for programming an FPGA-only design into flash memory.

If none of these options are desired, click **Cancel** to continue.


Bitstream Generation Completed

Project 'zynq_pmod_2020_1_1' Bitstream Generation successfully completed.

Next

[✓ Open Implemented Design]

View Reports

Open Hardware Manager

Generate Memory Configuration File

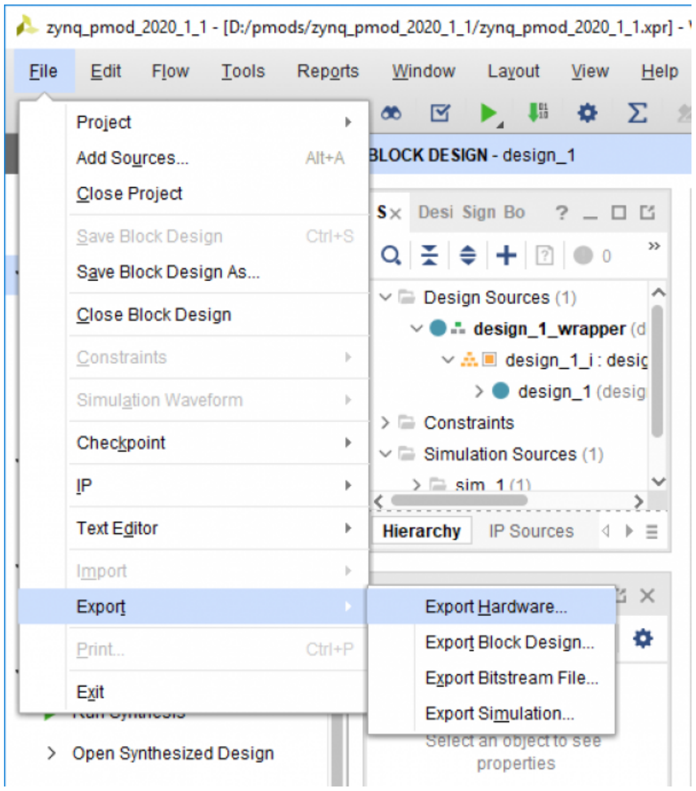
Don't show this dialog again

(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/generate-bitstream/generate-bitstream-3.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

Export a Fixed Post-Synthesis Hardware Platform

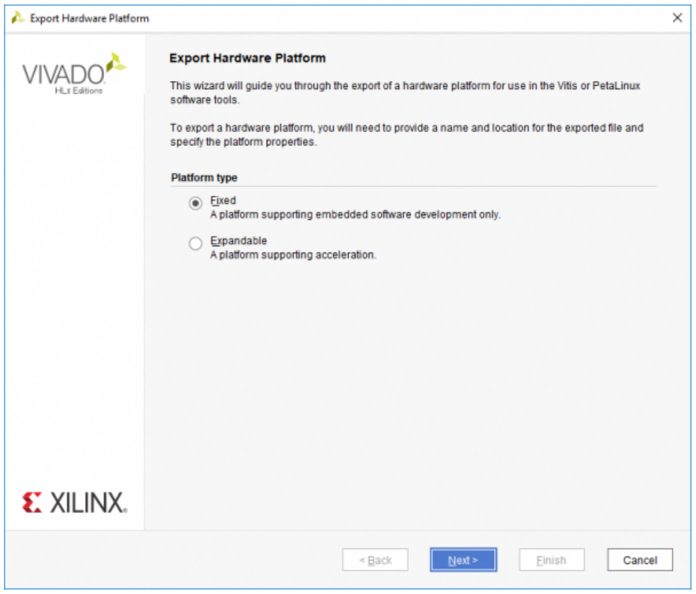
Once the project has been built, the design must be exported from Vivado so that Vitis has access to information about the hardware that a software application is being developed for. This includes the set of IP connected to the processor, their drivers, their addresses, and more. Exporting hardware after the bitstream has been generated allows you to program your board directly from within Vitis.

To export the hardware design, click **Export → Export Hardware** in the *File* menu.



(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/export-fixed-hardware/export-hardware.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

The wizard that pops up guides you through the options available for hardware export. The first screen allows you to select a *Fixed* or *Expandable* platform. In this case, choose a Fixed platform and click **Next** to continue.


Export Hardware Platform

Export Hardware Platform

This wizard will guide you through the export of a hardware platform for use in the Vitis or PetaLinux software tools.

To export a hardware platform, you will need to provide a name and location for the exported file and specify the platform properties.

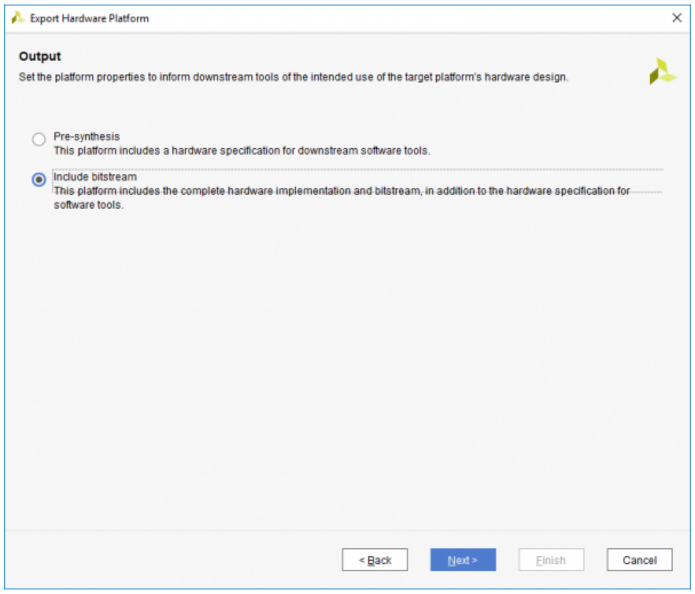
Platform type

[✓ Fixed] A platform supporting embedded software development only.

Expandable A platform supporting acceleration.

(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/export-fixed-hardware/export-hardware-fixed-1.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

The *Output* screen allows you to select whether only the hardware specification (*Pre-synthesis*) should be exported, or whether the bitstream should be included. Since the bitstream has already been generated, it should be included in the platform so that Vitis can automatically figure out where it is when programming a board. Select *Include bitstream* and click **Next** to continue.


Export Hardware Platform

Output

Set the platform properties to inform downstream tools of the intended use of the target platform's hardware design.

Pre-synthesis
This platform includes a hardware specification for downstream software tools.

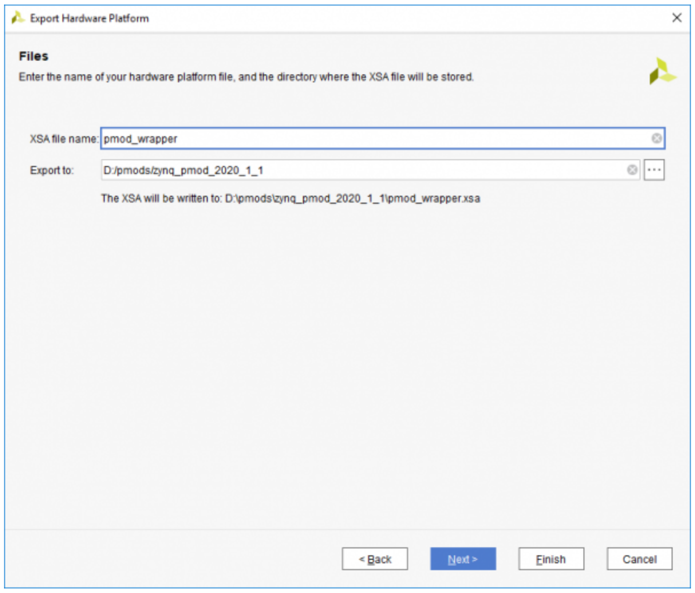
[✓ Include bitstream]
This platform includes the complete hardware implementation and bitstream, in addition to the hardware specification for software tools.

(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/export-fixed-hardware/export-hardware-fixed-2.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

The *Files* screen gives you the option to choose a name for the Xilinx Shell Architecture (XSA) file, and provide a path to a folder that the file will be placed within. Give your XSA file a name, and choose a memorable location to place it in. This file will later be imported into Vitis, so take a note of where it is placed and what it is called.

Important: *Do not use spaces in the file name or export path. Underscores or camelCase (https://en.wikipedia.org/wiki/Camel_case) are recommended instead.*

Click **Next** to continue.


Export Hardware Platform

Files

Enter the name of your hardware platform file, and the directory where the XSA file will be stored.

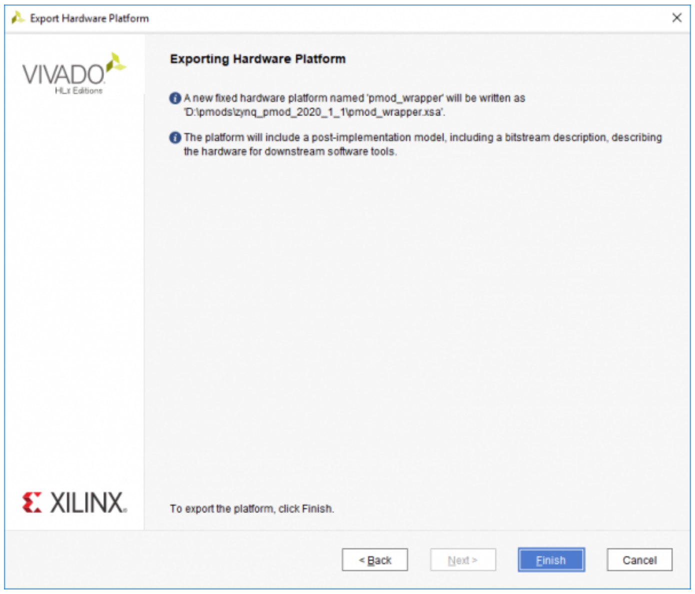
XSA file name: pmod_wrapper

Export to: D:/pmods/zynq_pmod_2020_1_1

The XSA will be written to: D:\pmods\zynq_pmod_2020_1_1\pmod_wrapper.xsa

(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/export-fixed-hardware/export-hardware-fixed-3.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

The final screen of the wizard summarizes the options you selected. Click **Finish**.



Export Hardware Platform

Exporting Hardware Platform

A new fixed hardware platform named 'pmod_wrapper' will be written as 'D:\pmods\zynq_pmod_2020_1_1\pmod_wrapper.xsa'.

The platform will include a post-implementation model, including a bitstream description, describing the hardware for downstream software tools.

To export the platform, click Finish.

(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/export-fixed-hardware/export-hardware-fixed-4.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

Launch Vitis

Select the dropdown corresponding to your operating system, below.

Windows

Open Vitis through the start menu or desktop shortcut created during the installation process.

(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/launch-vitis/windows.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

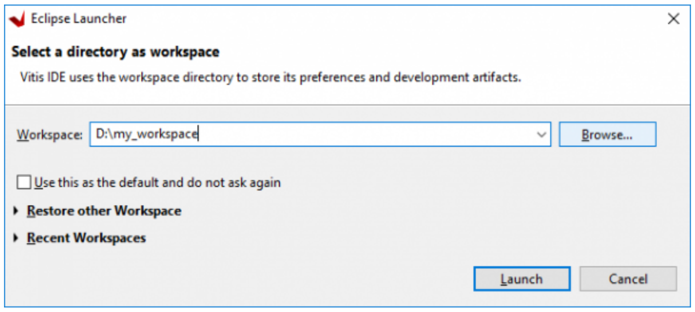
Linux

Open a terminal and run the following commands. The install path is /opt/Xilinx by default.

```
source <install_path>/Vitis/2020.1/settings64.sh
vitis
```

Note: *Regardless of OS (), if Vivado is open, Vitis can also be launched through the* Tools → Launch Vitis *toolbar option.*

Upon launching Vitis, a dialog will appear where a workspace must be chosen. The workspace is the directory where all of the projects and files for the application being developed will live. If a folder that does not currently exist is chosen, it will be created. Choose a workspace and click **Launch** to finish launching Vitis.



Eclipse Launcher

Select a directory as workspace

Vitis IDE uses the workspace directory to store its preferences and development artifacts.

Workspace: D:\my_workspace

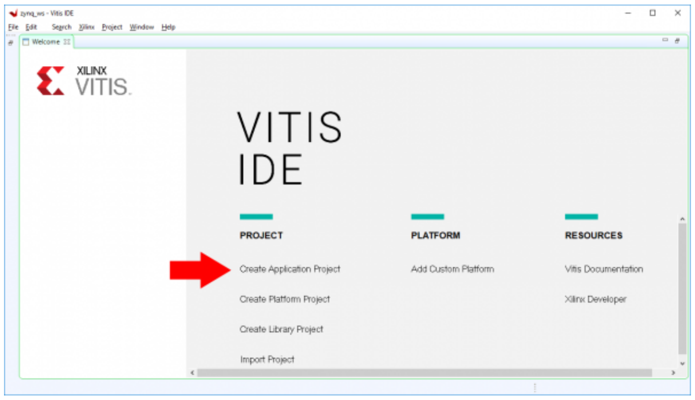
☐ Use this as the default and do not ask again

(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/launch-vitis/open-vitis-2.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

Create a New Application Project

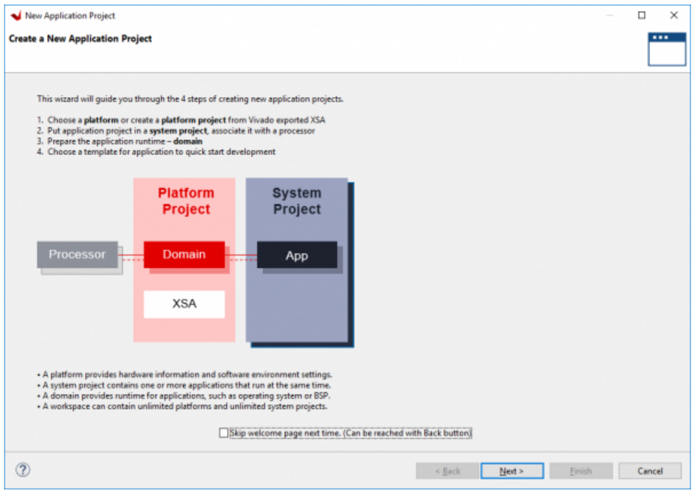
With Vitis open, an application project must be created to hold your source files. In creating an application project, a hardware platform will also be created from an XSA file previously exported from Vivado.

On Vitis' welcome screen, click **Create Application Project**. The wizard that launches will be used to create and configure a new application.



(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/create-application-project/vitis-new-application-project-1.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

The first screen of the wizard is a welcome page, which summarizes what each of the components of a software design are. Click **Next** to continue.



Create a New Application Project

This wizard will guide you through the 4 steps of creating new application projects.

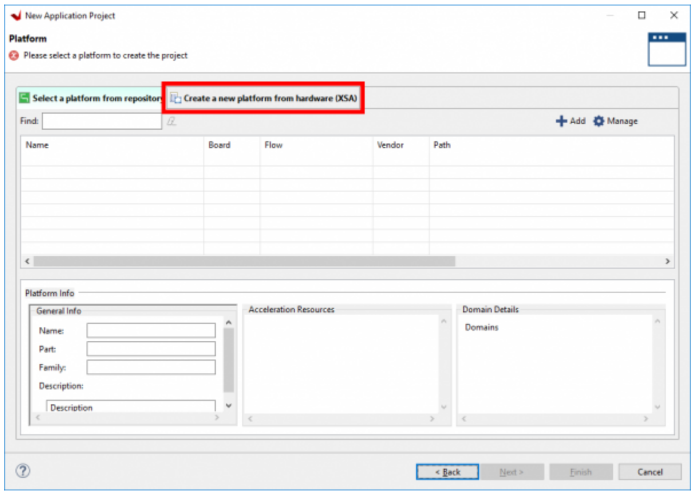
1. Choose a **platform** or create a **platform project** from Vivado exported XSA
2. Put application project in a **system project**, associate it with a processor
3. Prepare the application runtime - **domain**
4. Choose a template for application to quick start development

- A platform provides hardware information and software environment settings.
- A system project contains one or more applications that run at the same time.
- A domain provides runtime for applications, such as operating system or BSP.
- A workspace can contain unlimited platforms and unlimited system projects.

Skip welcome page next time. (Can be reached with Back button)

(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/create-application-project/vitis-new-app-summary.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

Next, the platform that the application targets must be created. Open the **Create a new platform…** tab.

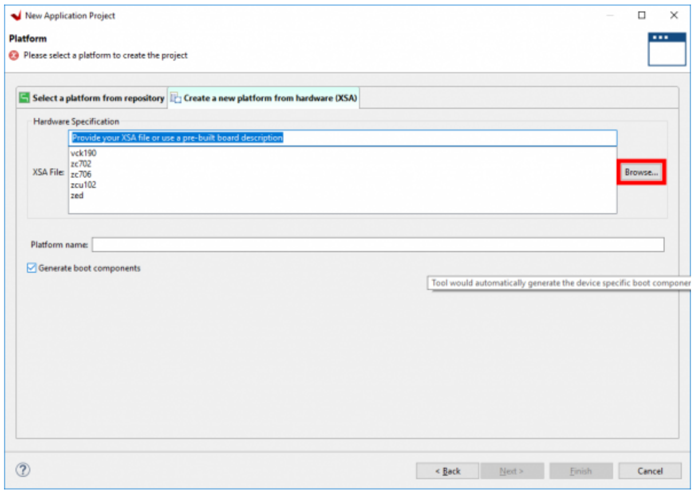


Platform

Please select a platform to create the project

(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/create-application-project/vitis-new-app-1.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

Browse your file system to find the Xilinx Shell Architecture previously exported from Vivado. With the XSA file highlighted, click **Open** to select it and return to the *Platform* screen of the wizard.



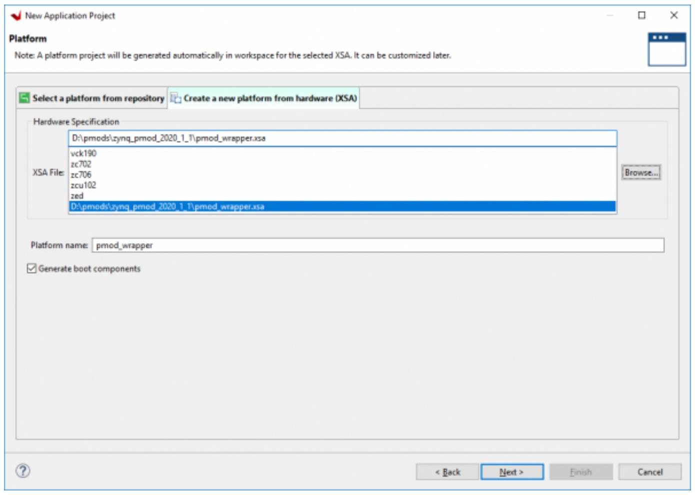
Please select a platform to create the project

Provide your XSA file or use a pre-built board description
vck190
zc702
zc706
zcu102
zed

Tool would automatically generate the device specific boot componer

(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/create-application-project/vitis-new-app-2.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

Once you have found the XSA file and opened it, make sure that it is selected in the *Hardware Specification* list. Give your platform a name (the default uses whatever the name of the XSA file is and will work fine). The *Generate boot components* box can be used to automatically build all of the additional components necessary to boot the application from flash memory or an SD card. Leaving this box checked is recommended. Click **Next** to continue.



Note: A platform project will be generated automatically in workspace for the selected XSA. It can be customized later.

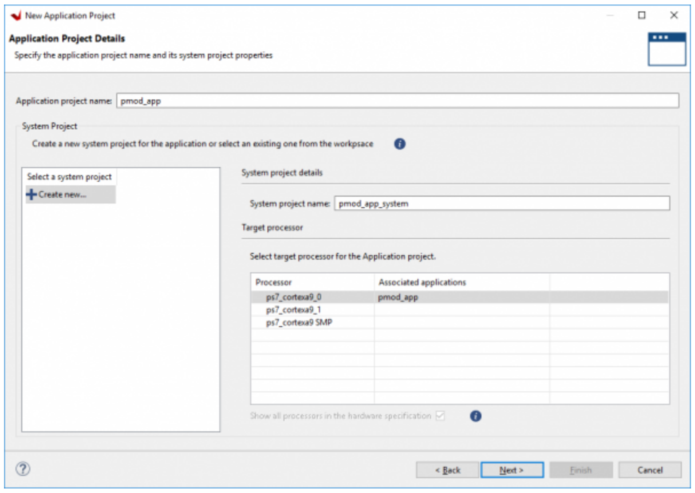
D:\pmods\zynq_pmod_2020_1_1\pmod_wrapper.xsa
vck190
zc702
zc706
zcu102
zed
D:\pmods\zynq_pmod_2020_1_1\pmod_wrapper.xsa

Platform name: pmod_wrapper

(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/create-application-project/vitis-new-app-4.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

The next screen is used to set some options for the application project and the system project. The names of both projects can be set, as well as which processor core will be used to run the application. All settings can be left as defaults. Click **Next** to continue.

Note: *A system project can contain multiple application projects, which can all be run at once.*



Specify the application project name and its system project properties

Application project name: pmod_app

Create a new system project for the application or select an existing one from the workspace

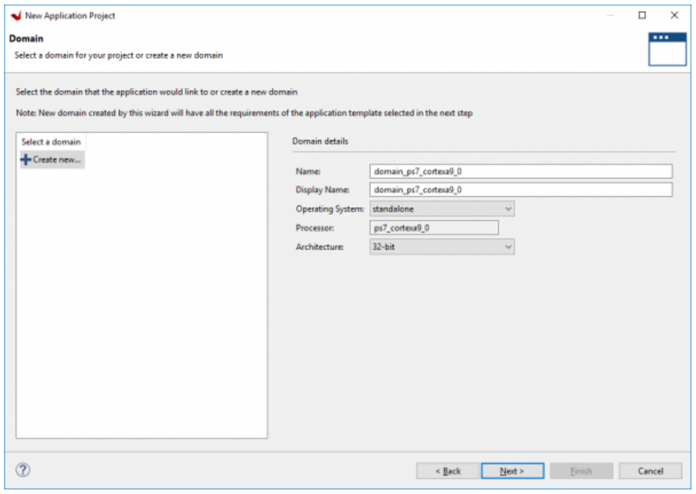
System project name: pmod_app_system

Select target processor for the Application project.

Processor	Associated applications
ps7_cortexa9_0	pmod_app
ps7_cortexa9_1	
ps7_cortexa9 SMP	

(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/create-application-project/vitis-new-app-5.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

Next, the domain that the application project operates in will be defined. In this case all default settings will be used. Click **Next** to continue.


New Application Project

Domain

Select a domain for your project or create a new domain

Select the domain that the application would link to or create a new domain

Note: New domain created by this wizard will have all the requirements of the application template selected in the next step

Select a domain

Create new...

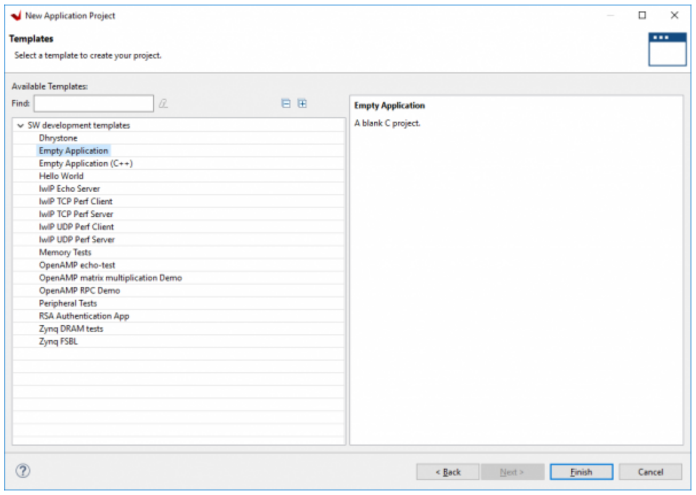
Domain details

Name:	domain_ps7_cortexa9_0
Display Name:	domain_ps7_cortexa9_0
Operating System:	standalone
Processor:	ps7_cortexa9_0
Architecture:	32-bit

(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/create-application-project/vitis-new-app-6.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

Lastly, a template project will be chosen. Each template pre-configures the application project for a different purpose. Depending on the whether your application will be written in C or C++, choose **Empty Application** or **Empty Application (C++)**. You will be adding an example main source file later, as opposed to working from and editing an example.

Click **Finish** to finish creating the project.


New Application Project

Templates

Select a template to create your project.

Available Templates:

Find:

SW development templates
- Dhrystone
- Empty Application
- Empty Application (C++)
- Hello World
- lwIP Echo Server
- lwIP TCP Perf Client
- lwIP TCP Perf Server
- lwIP UDP Perf Client
- lwIP UDP Perf Server
- Memory Tests
- OpenAMP echo-test
- OpenAMP matrix multiplication Demo
- OpenAMP RPC Demo
- Peripheral Tests
- RSA Authentication App
- Zynq DRAM tests
- Zynq FSBL

Empty Application

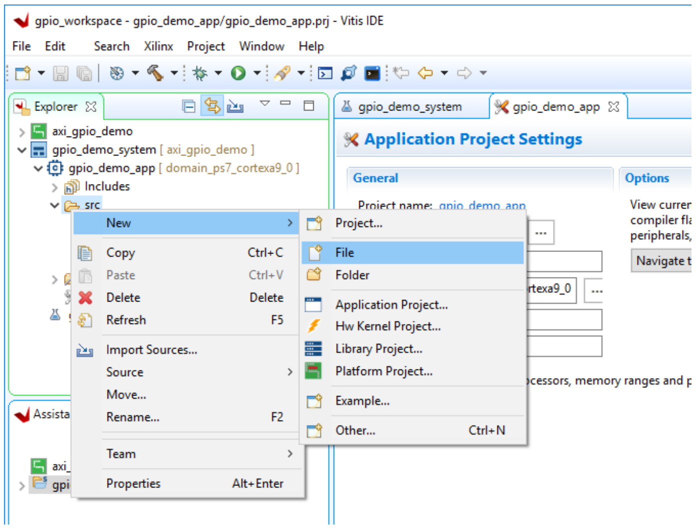
A blank C project.

(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/create-application-project/vitis-new-app-7.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

Create a Main C Source to Control AXI GPIO Peripherals

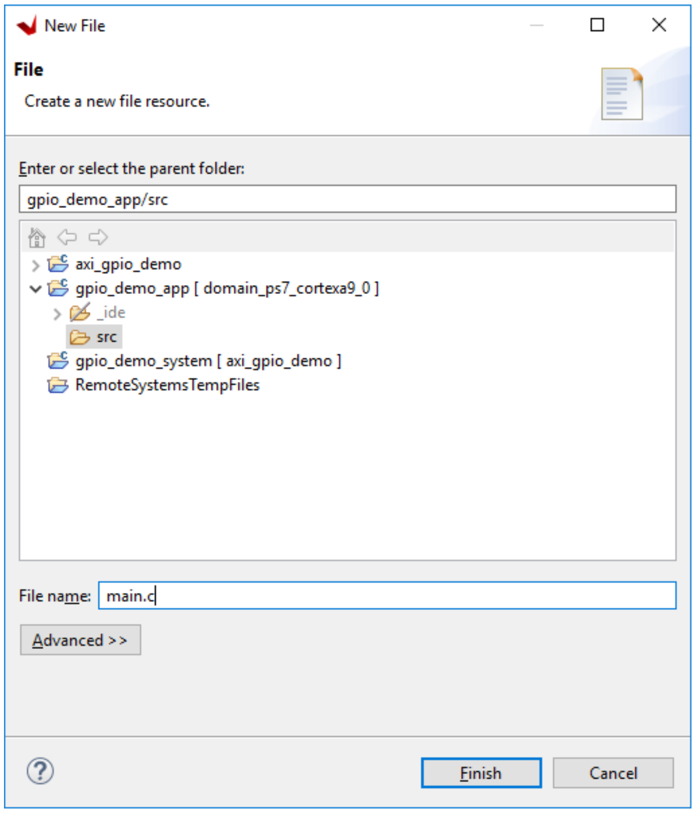
An application needs source files to define its behavior. This step will show how to create a new source file for the application, and provide some example code.

In Vitis' *Explorer* pane, find the application projects "src" directory. Right click on it and select *New → File.*



(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/create-blinky-software-example/software-new-source-file.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

In the dialog that pops up, name the file "main.c". The parent folder can be specified as well, but through the use of the right click in the previous step, the correct folder has already been chosen.



New File
File
Create a new file resource.
Enter or select the parent folder:
gpio_demo_app/src
axi_gpio_demo
gpio_demo_app [domain_ps7_cortexa9_0]
_ide
src
gpio_demo_system [axi_gpio_demo]
RemoteSystemsTempFiles
File name: main.c
Advanced >>
Finish
Cancel

(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/create-blinky-software-example/name-source-file.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

Copy and paste the code to the right into the empty main.c file that has now been opened. Change the BTN_MASK and LED ()_MASK macros so that they contain a number of '1's equal to the number of buttons and leds connected to the GPIO () peripherals in the hardware design.

Header Files - Additional Information

This code pulls in several headers that are automatically pulled into the Vitis workspace:

xparameters.h is a file generated during the process of exporting a platform from Vivado. It includes information on the hardware design, including addresses and some configuration parameters for AXI IPs. This is used by the example code to find the device IDs that must be passed to the GPIO () drivers, so that they can look up the driver configuration required to correctly initialize the GPIO () devices.

xil_printf.h gives access to the xil_printf function, which can be used to print to standard output, and requires less memory space than the stdio library.

xgpio.h gives access to the XGpio drivers, which are used to provide a standard API () for controlling AXI GPIO () peripherals. Several functions from this API () are used in the example, including the GPIO () reads, writes, and direction-setting calls.

xil_types.h contains a variety of different C types. In this case, it is only used to get access to the "u32" (unsigned 32-bit int) type, which is used in arguments to XGpio function calls.

What the Example Code Does

When the example is started, the message "Entered function main" is printed to a connected serial console. After that, the AXI GPIO () IPs and drivers are initialized, and the application constantly loops, checking whether any button is pressed, and if they are, setting the LEDs high. When no buttons are pressed, the LEDs are held low.

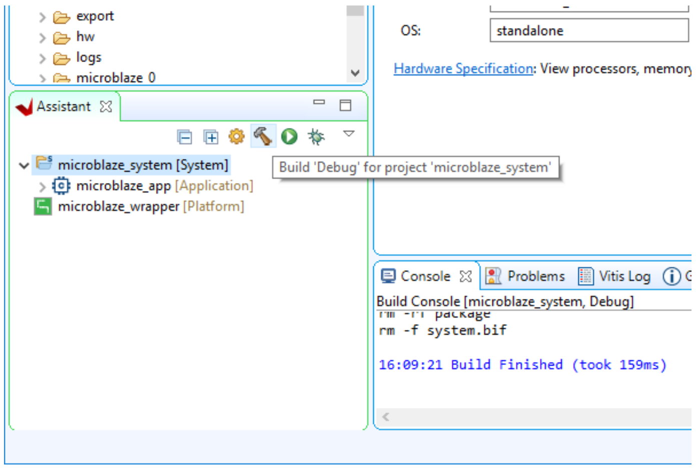
```
#include "xparameters.h"
#include "xil_printf.h"
#include "xgpio.h"
#include "xil_types.h"

// Get device IDs from xparameters.h
#define BTN_ID XPAR_AXI_GPIO_BUTTONS_DEVICE_ID
#define LED_ID XPAR_AXI_GPIO_LED_DEVICE_ID
#define BTN_CHANNEL 1
#define LED_CHANNEL 1
#define BTN_MASK 0b1111
#define LED_MASK 0b1111

int main() {
        XGpio_Config *cfg_ptr;
        XGpio led_device, btn_device;
        u32 data;

        xil_printf("Entered function main\r\n");

        // Initialize LED Device
        cfg_ptr = XGpio_LookupConfig(LED_ID);
        XGpio_CfgInitialize(&led_device, cfg_ptr, cfg_ptr

        // Initialize Button Device
        cfg_ptr = XGpio_LookupConfig(BTN_ID);
        XGpio_CfgInitialize(&btn_device, cfg_ptr, cfg_ptr

        // Set Button Tristate
        XGpio_SetDataDirection(&btn_device, BTN_CHANNEL,

        // Set Led Tristate
        XGpio_SetDataDirection(&led_device, LED_CHANNEL,

        while (1) {
                data = XGpio_DiscreteRead(&btn_device, BT
                data &= BTN_MASK;
                if (data != 0) {
                        data = LED_MASK;
                } else {
                        data = 0;
                }
                XGpio_DiscreteWrite(&led_device, LED_CHAN
        }
}
```

Build a Vitis Application

Once an application project has been set up and includes all necessary sources, it should be built. To build the project and all of its dependencies, select the system project in the *Assistant* pane, and either click the **Build** button (🔨), or press Ctrl-B on your keyboard.

Note: *There are three types of build targets in the Assistant pane, Platforms, Systems, and Applications. Building the application will not trigger any other applications in the system to be built, but will build the wrapper as a dependency. Building the platform will only build the platform, as it has no dependencies. Building the system causes each application in the system, as well as the platform, to be built.*

This process may take several minutes to complete. When done, the *Console* tab at the bottom of the window will display a "Build Finished" message.


export
hw
logs
microblaze_0
OS:
standalone
Hardware Specification: View processors, memory
Assistant
microblaze_system [System]
microblaze_app [Application]
microblaze_wrapper [Platform]
Build 'Debug' for project 'microblaze_system'
Console
Problems
Vitis Log
Build Console [microblaze_system, Debug]

```
rm -rf package
rm -f system.bif

16:09:21 Build Finished (took 159ms)
```

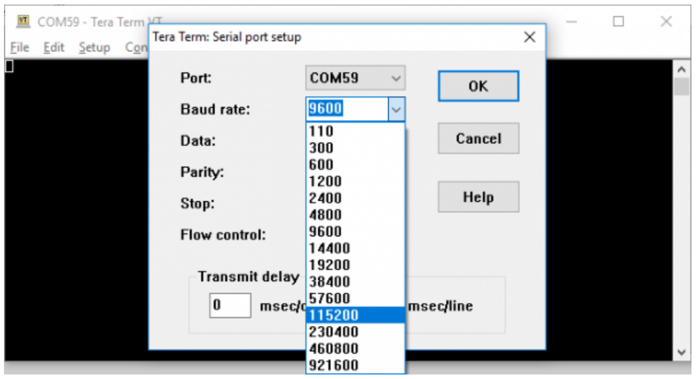
(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/build-software/build-system.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

It's time to program the application project onto your board! Plug your board into your computer through its USB programming and USB UART port/s, connect an external power supply (if necessary), and turn on the board.

Launch a Vitis Baremetal Software Application

First, many applications require that a serial console is connected to the board, so that standard output (from print statements) can be viewed. For this purpose, a serial terminal should be used. Use a serial terminal application to connect to the board's serial port. Unless otherwise stated, Zynq designs use a baud rate of 115200 and Microblaze designs with an AXI UART Lite IP use a baud rate of 9600.

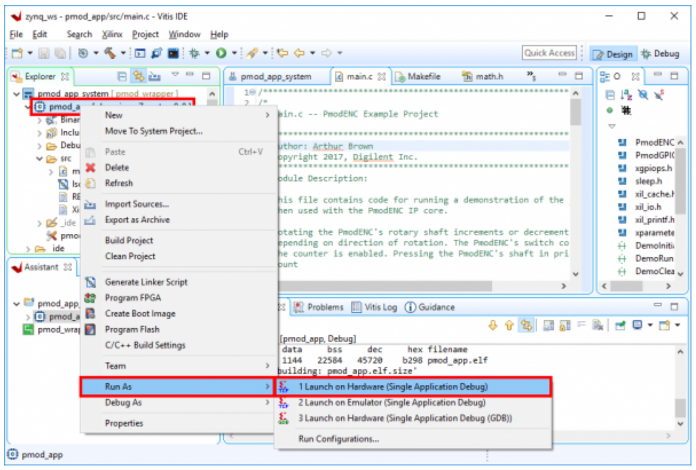
Note: *While Vitis has a built in serial terminal included in its Debug view, it sends characters to a board on a line-by-line basis. Some software examples require the use of character-by-character reception of data. Tera Term (https://ttssh2.osdn.jp/index.html.en) or PuTTY (https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html) are recommended if you are not sure what will work.*



(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/launch-vitis-application/set-baud.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

In the *Explorer* pane at the left side of the screen, right click on the application or system project that is to be run, and select *Run as → 1 Launch on Hardware (Single Application Debug)*. The FPGA will be programmed with the bitstream, the ELF file created by the software build is loaded into system memory, and the application project will begin to run. You will need to click back over to the *Vitis Serial Terminal* from the *Console* tab.

Note: *Once the project has been run at least once, you can use the green run button ( ) in the toolbar at the top of the screen to program the board instead.*



```
ain.c -- PmodENC Example Project
uthor: Arthur Brown
opyright 2017, Digilent Inc.
odule Description:
his file contains code for running a demonstration of the
hen used with the PmodENC IP core.
otating the PmodENC's rotary shaft increments or decrement
epending on direction of rotation. The PmodENC's switch co
he counter is enabled. Pressing the PmodENC's shaft in pri
ount
```

```
[pmod_app, Debug]
 data     bss     dec     hex filename
 1144   22584   45720    b298 pmod_app.elf
building: pmod_app.elf.size'
```

(https://digilent.com/reference/_detail/learn/programmable-logic/tutorials/2020.1/launch-vitis-application/launch-on-hardware.png?id=programmable-logic%3Aguides%3Agetting-started-with-ipi)

Next Steps

At this point, your application is running, and printed messages can be seen. Congratulations, you have finished this guide!!!

The hardware project and application created here can be used as a basis for future work. See instructions found in Update an Existing Vitis Platform's Hardware Specification (https://digilent.com/reference/programmable-logic/guides/vitis-update-hardware-specification) for additional information on how the hardware design can be switched out later.

For more guides and demos for your board, return to the device's resource center, linked from the Programmable Logic (https://digilent.com/reference/programmable-logic/start) page of this wiki.

For technical support, please visit the FPGA (https://forum.digilentinc.com/forum/4-fpga/) section of the Digilent Forums.