Implementing the ADC on Intel® MAX® 10-10M08 Evaluation Kit

By Sean D. Liming and John R. Malin
Annabooks, LLC. – www.annabooks.com

October 2022

This hands-on article will create a simple design for the ADC controller that is built into the MAX 10 FPGAs. The Max 10 is used in conjunction with the higher-end Intel FPGA to provide the ADC support that these higher-end FPGAs don't support. The MAX 10 -10M08 Evaluation Kit developed by Terasic, Inc, is a very simple development board to help make it easy to get started developing with the Quartus Prime software.

The Project Requirements:

- Intel Quartus Prime Lite Edition V21.0 – 7zip is required to extract the tar file.
- Intel® MAX® 10 – 10M08 Evaluation Kit – the evaluation kit and the schematic for the evaluation board are required. The schematic PDF file can be downloaded from the Intel FPGA website.
    - Optional: a populated 10KΩTrimmer pot for R94 on the schematic, part number 3362P-1-103TLF.
- Intel FPGA Programming cable – USB Blaster II or EthernetBlaster II. Unlike other FPGA boards, the MAX 10 -10M08 Evaluation Kit doesn't have a built-in USB Blaster solution; so a separate programming cable is required.
- A signal or function generator – some kind an AC signal generator, i.e., BK Precision 4012A or equivalent is required.

Note: There are equivalent MAX 10 development and evaluation boards available. These boards can also be used as the target, but you will have to adjust to the available features on the board. Please make sure that you have the board's schematic files as these will be needed to identify pins.

1.1 Download and Install the Quartus Prime Lite Edition Software

The Quartus can run on Windows and different Linux distributions. The installation of the software has many steps. Please see the article *Intel® Quartus® Prime Lite and NIOS® II SBT for Eclipse Installation Instructions* on Annabooks.com to install the software needed for this hands-on exercise. All you need for this hands-on paper is the installation of the Quartus Prime Lite software with Max 10 support.

1.2 ADC Project

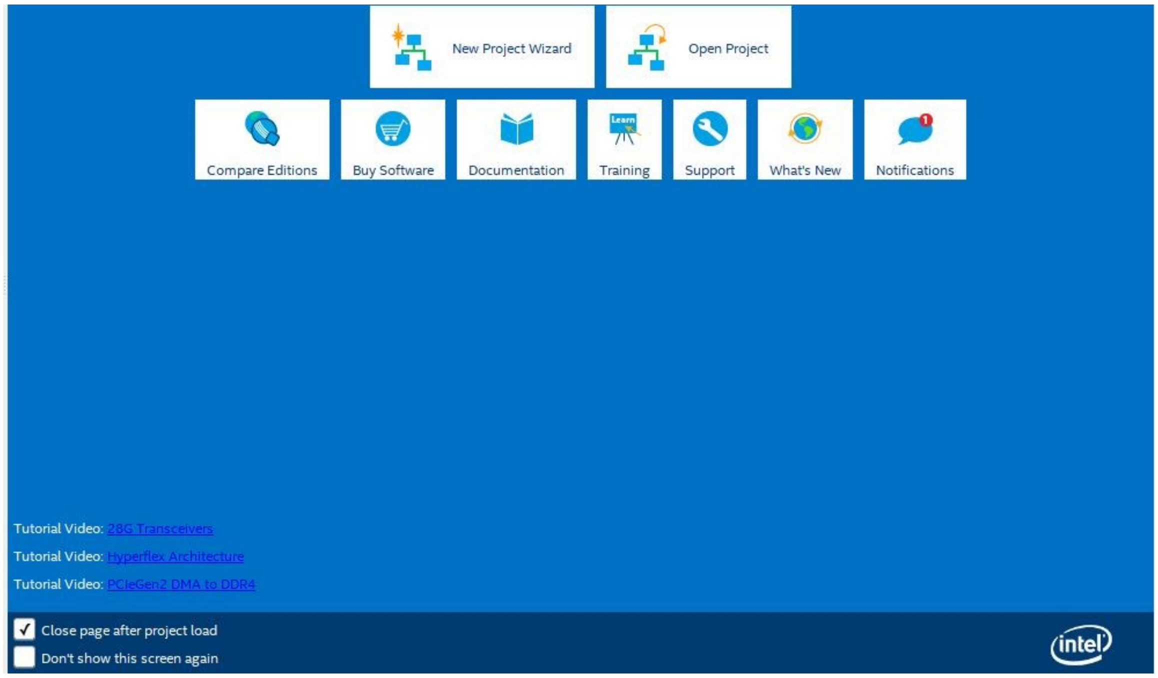
There are several parts in the MAX 10 family. The lower end of the family has 1 ADC block built in, and the upper end of the family has 2 ADC blocks.

The 10M08SAE144C8G that is on the evaluation kit board contains only 1 ADC block. To help create a design with the ADC, there is an ADC IP block that comes with the Quartus Prime software that can be used to create a design with the ADC. This IP block allows developers to quickly create a design without having to start from scratch. 10M08SAE144C8G's onboard 12-bit ADC has 8 input channels (CH1-CH8), a dedicated input channel 0 (ANIAN1), and a temperature sensing diode (TSD) channel as input. The 7 input channels are multiplexed with GPIOs. When the ADC is in the design, the ADC/GPIO multiplexed pins can only be used for ADC. A compilation error will occur if you try to make an assignment. The built-in temperature sensing diode (TSD) can measure the internal temperature of the MAX 10. The ADC is clocked using a phase-locked loop (PLL) The MAX 10 has 4 PLLs built in. IP blocks are used to implement the ADC and PLL into the design. For this hands-on example, we will create a design that uses two channels. The first is channel 1 which is attached to a DC or AC source and the second is channel 7 which is connected to the R94 10KΩ trimmer pot. The ADC Toolkit that comes with Quartus Prime will read the output from the ADC.

1.2.1 Create the Project

The first step is to create the design project.

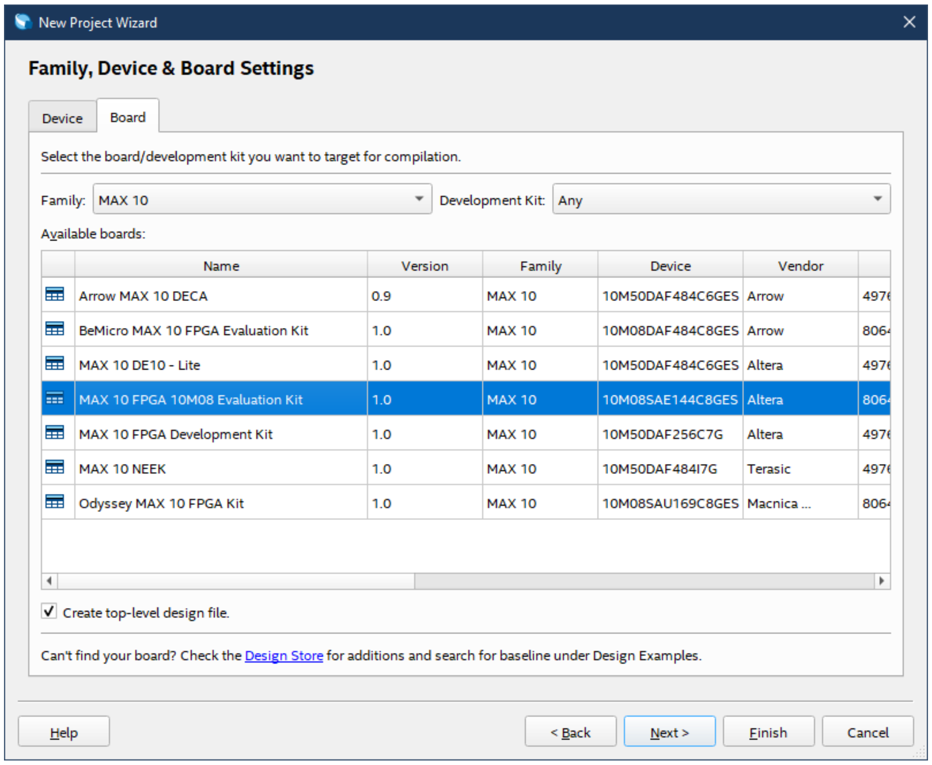
1. Open Quartus.
2. Click on the New Project Wizard.


New Project Wizard
Open Project
Compare Editions
Buy Software
Documentation
Training
Support
What's New
Notifications
Tutorial Video: 28G Transceivers
Tutorial Video: Hyperflex Architecture
Tutorial Video: PCIeGen2 DMA to DDR4
Close page after project load
Don't show this screen again
intel

3. Click Next to the Introduction dialog.
4. Select or create a project directory c:\FPGA\ADC_Example (Do not use the Quartus installation directory) and name the project “ADC0_Example”. Click Next.

Note: By default, the root directory is the Quartus installation directory. Make sure the root project directory is a separate path from the Quartus installation files.

5. Project Type: Empty project, click Next.
6. Add File, no files to add, click Next.
7. Family, Device & Board Settings, click the Board tab and select: MAX 10 FPGA 10M08 Evaluation Kit and click Next.


New Project Wizard

Family, Device & Board Settings

Select the board/development kit you want to target for compilation.

Family: MAX 10 Development Kit: Any

Available boards:

Name	Version	Family	Device	Vendor	
Arrow MAX 10 DECA	0.9	MAX 10	10M50DAF484C6GES	Arrow	497
BeMicro MAX 10 FPGA Evaluation Kit	1.0	MAX 10	10M08DAF484C8GES	Arrow	806
MAX 10 DE10 - Lite	1.0	MAX 10	10M50DAF484C6GES	Altera	497
MAX 10 FPGA 10M08 Evaluation Kit	1.0	MAX 10	10M08SAE144C8GES	Altera	806
MAX 10 FPGA Development Kit	1.0	MAX 10	10M50DAF256C7G	Altera	497
MAX 10 NEEK	1.0	MAX 10	10M50DAF484I7G	Terasic	497
Odyssey MAX 10 FPGA Kit	1.0	MAX 10	10M08SAU169C8GES	Macnica ...	806

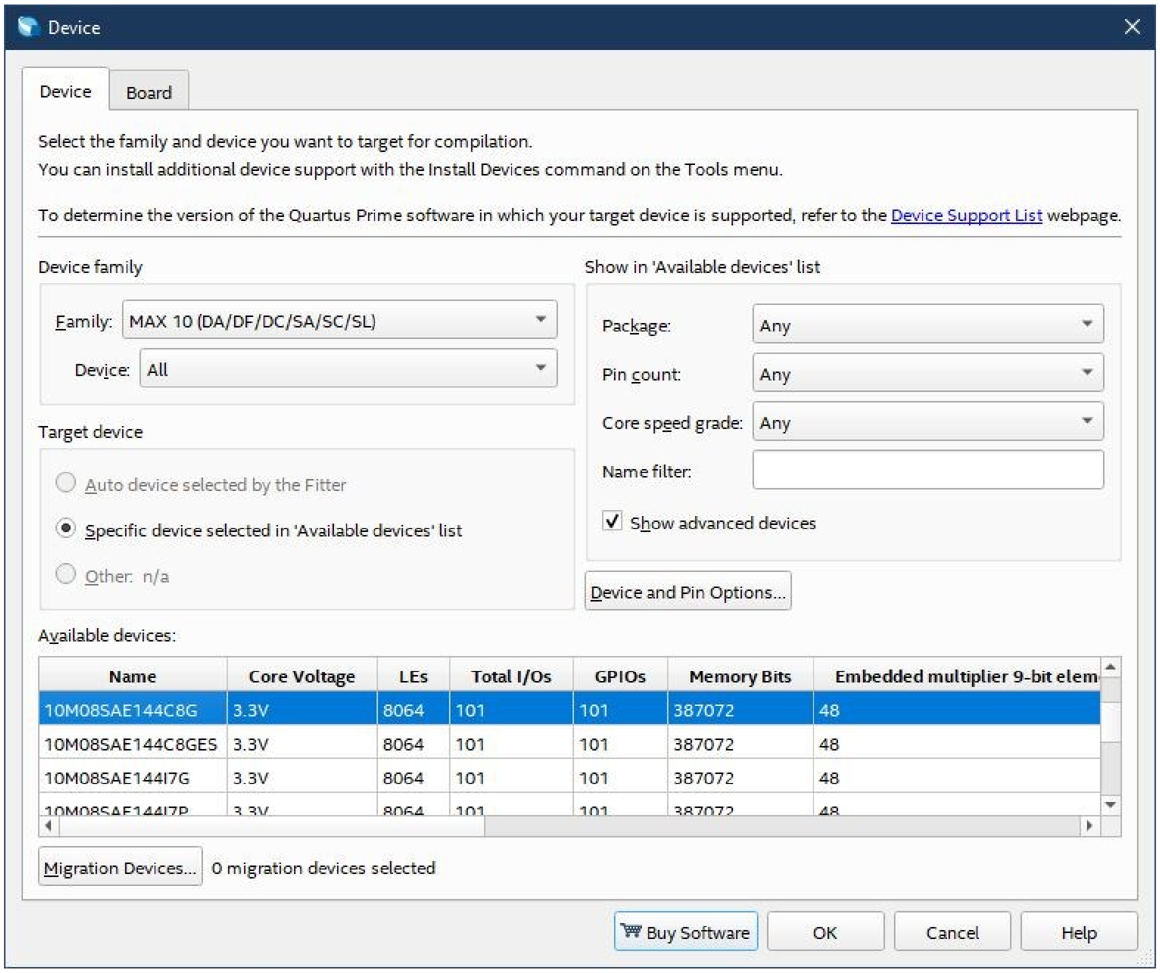
✓ Create top-level design file.

Can't find your board? Check the Design Store for additions and search for baseline under Design Examples.

8. EDA Tools, click Next.
9. Summary, click Finish.

Note: The actual MAX 10 on our board is the 10M08SAE144C8G, thus it is not an Engineering Sample (ES). The next two steps change the device to the production device. Your experience might be different. These next two optional steps change the device.

10. In the project navigation pane on the left, right-click on 10: 10M08SAE144C8GE, and select Device from the context menu.
11. In the Available devices, scroll down and select the 10M08SAE144C8G, click OK.



Select the family and device you want to target for compilation.
You can install additional device support with the Install Devices command on the Tools menu.

To determine the version of the Quartus Prime software in which your target device is supported, refer to the Device Support List webpage.

Device family

Family: MAX 10 (DA/DF/DC/SA/SC/SL)

Device: All

Target device

( ) Auto device selected by the Fitter

(●) Specific device selected in 'Available devices' list

( ) Other: n/a

Show in 'Available devices' list

Package: Any

Pin count: Any

Core speed grade: Any

Name filter:

✓ Show advanced devices

Available devices:

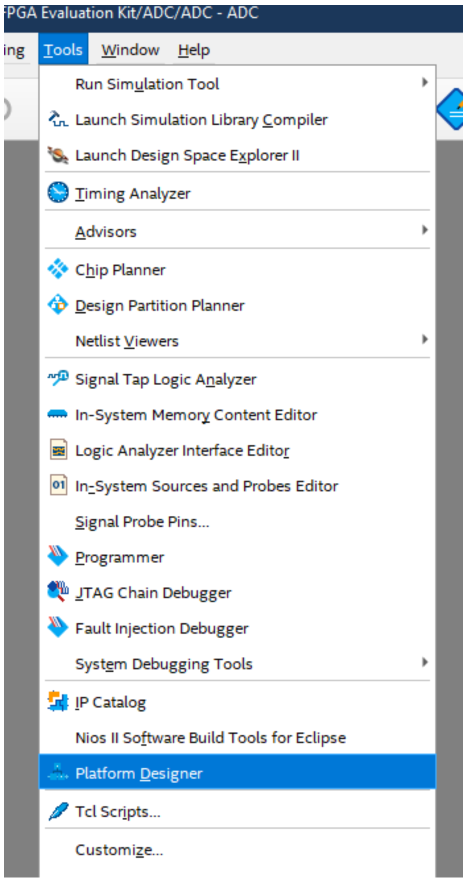
Name	Core Voltage	LEs	Total I/Os	GPIOs	Memory Bits	Embedded multiplier 9-bit elem
10M08SAE144C8G	3.3V	8064	101	101	387072	48
10M08SAE144C8GES	3.3V	8064	101	101	387072	48
10M08SAE144I7G	3.3V	8064	101	101	387072	48
10M08SAE144I7P	3.3V	8064	101	101	387072	48

0 migration devices selected

1.2.2 Create the Design Step 1: Platform Builder

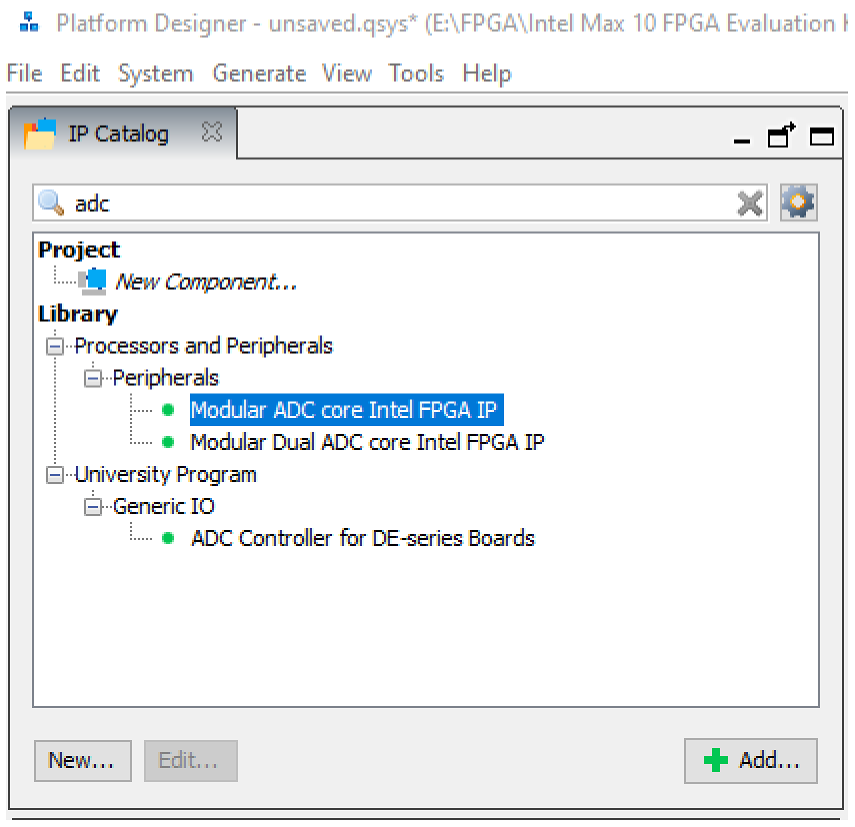
Quartus supports many design types to create an FPGA design. The Platform Designer tool will be used for this hands-on exercise. Platform Builder makes it easy to add already-built IP blocks and interconnect them.

1. From the menu, select Tools->Platform Designer, or the Platform Designer icon from the toolbar.

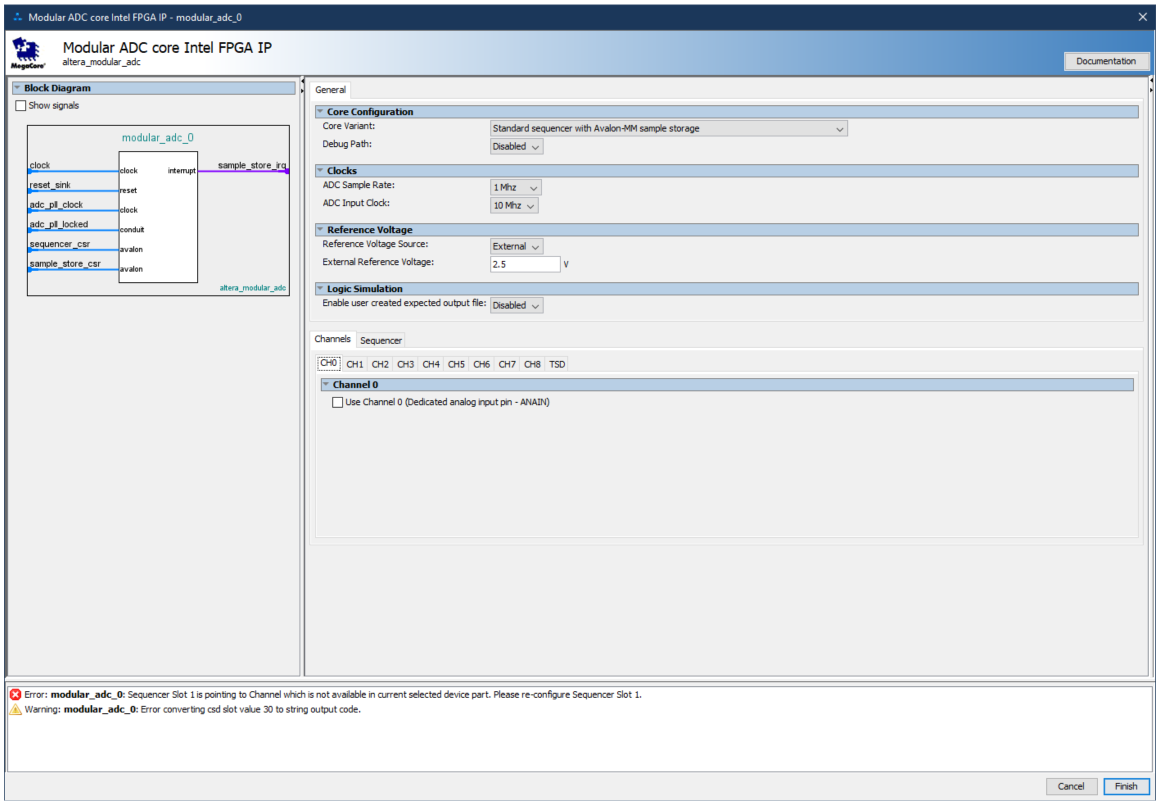

FPGA Evaluation Kit/ADC/ADC - ADC
ing
Tools
Window
Help
Run Simulation Tool
Launch Simulation Library Compiler
Launch Design Space Explorer II
Timing Analyzer
Advisors
Chip Planner
Design Partition Planner
Netlist Viewers
Signal Tap Logic Analyzer
In-System Memory Content Editor
Logic Analyzer Interface Editor
In-System Sources and Probes Editor
Signal Probe Pins...
Programmer
JTAG Chain Debugger
Fault Injection Debugger
System Debugging Tools
IP Catalog
Nios II Software Build Tools for Eclipse
Platform Designer
Tcl Scripts...
Customize...

The Platform Designer tool is launched. By default, a clock (clk_0) is added to the design. Platform Designer tool makes it easy to add IP blocks and make interconnections between the blocks.

2. The top, left pane contains the IP Catalog with all the available IP blocks that come with Quartus Prime. In the search box, type ADC.


Platform Designer - unsaved.qsys* (E:\FPGA\Intel Max 10 FPGA Evaluation
File Edit System Generate View Tools Help
IP Catalog
adc
Project
New Component...
Library
Processors and Peripherals
Peripherals
Modular ADC core Intel FPGA IP
Modular Dual ADC core Intel FPGA IP
University Program
Generic IO
ADC Controller for DE-series Boards
New...
Edit...
Add...

3. Expanding the branches reveals the available IP. Double-click on Modular ADC core Intel FPGA IP. This will add the ADC IP to the design and open the Modular ADC core Intel FPGA IP configuration page.



Core Configuration
Core Variant: Standard sequencer with Avalon-MM sample storage
Debug Path: Disabled

Clocks
ADC Sample Rate: 1 Mhz
ADC Input Clock: 10 Mhz

Reference Voltage
Reference Voltage Source: External
External Reference Voltage: 2.5 V

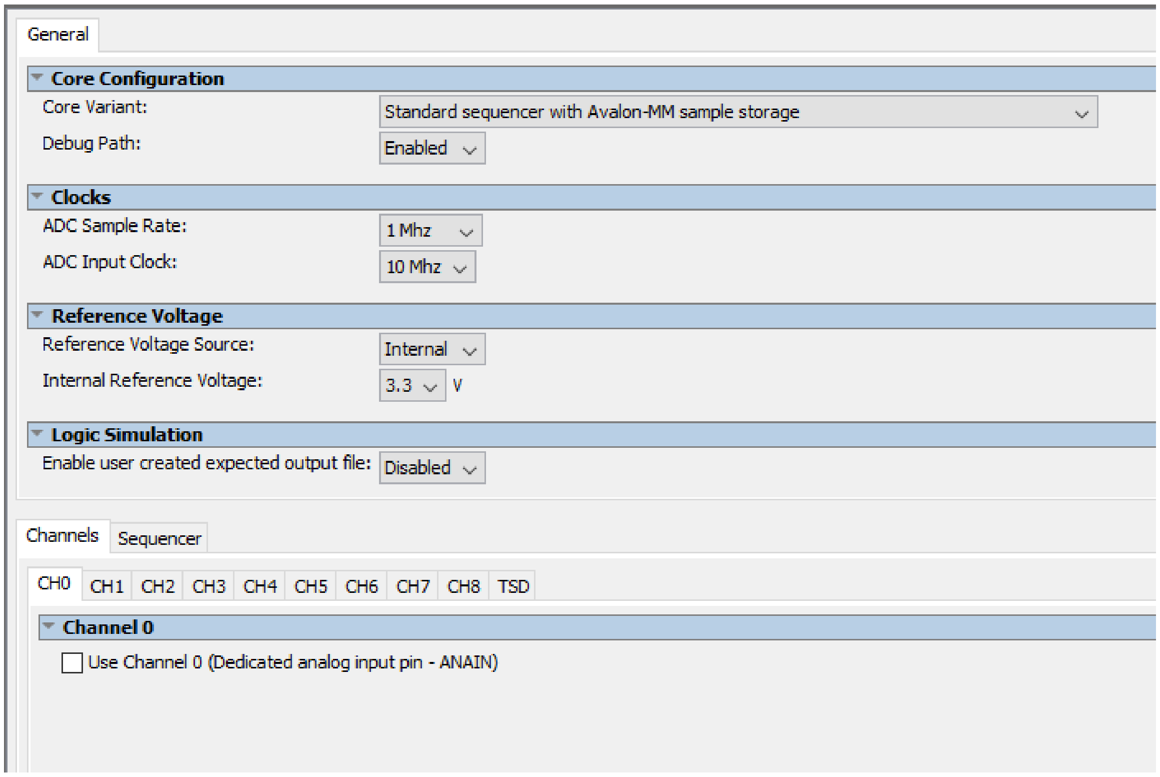
Logic Simulation
Enable user created expected output file: Disabled

Channel 0
☐ Use Channel 0 (Dedicated analog input pin - ANAIN)

Error: **modular_adc_0:** Sequencer Slot 1 is pointing to Channel which is not available in current selected device part. Please re-configure Sequencer Slot 1.
Warning: **modular_adc_0:** Error converting csd slot value 30 to string output code.

4. In the General tab, set the following:
    a. Core Variant: Standard sequencer with Avalon-MM sample storage.
    b. Debug Path: Enable.
    c. ADC Sample Rate: 1 MHz.
    d. ASC Input Clock: 10 MHz.
    e. Reference Voltage Source: External.
    f. Internal reference Voltage: 3.3V
    g. Enable user created expected output file: Disabled.

The ADC IP block supports several implementation variants. The one chosen will use the MAX 10's internal RAM to save the data. Since the ADC Toolkit will be used to monitor the ADC output, the Debug Path is set to Enabled. The evaluation kit has a 2.5 V reference voltage for the ADC; but since the 10K trimmer pot can supply 3.3 V to the channel, we will use the internal 3.3V.


General

Core Configuration
Core Variant: Standard sequencer with Avalon-MM sample storage
Debug Path: Enabled

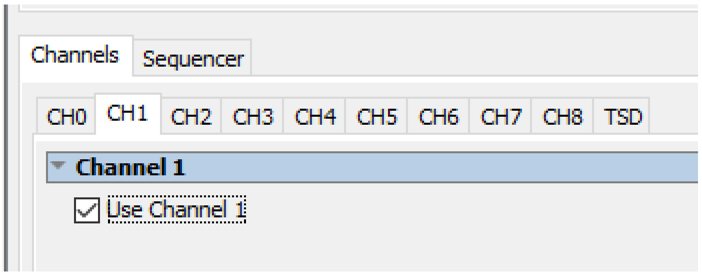
Clocks
ADC Sample Rate: 1 Mhz
ADC Input Clock: 10 Mhz

Reference Voltage
Reference Voltage Source: Internal
Internal Reference Voltage: 3.3 V

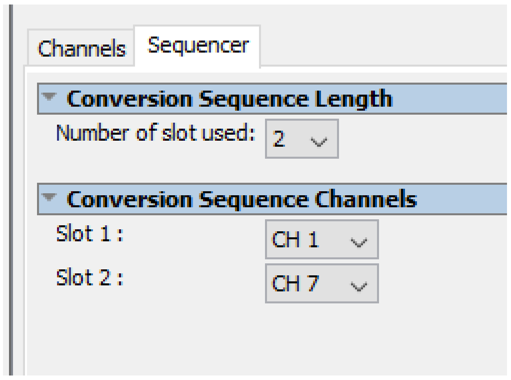
Logic Simulation
Enable user created expected output file: Disabled

Channels | Sequencer
CH0 | CH1 | CH2 | CH3 | CH4 | CH5 | CH6 | CH7 | CH8 | TSD
Channel 0
☐ Use Channel 0 (Dedicated analog input pin - ANAIN)

5. In the Channels tab, click on CH1, and check the “Use Channel 1” box.

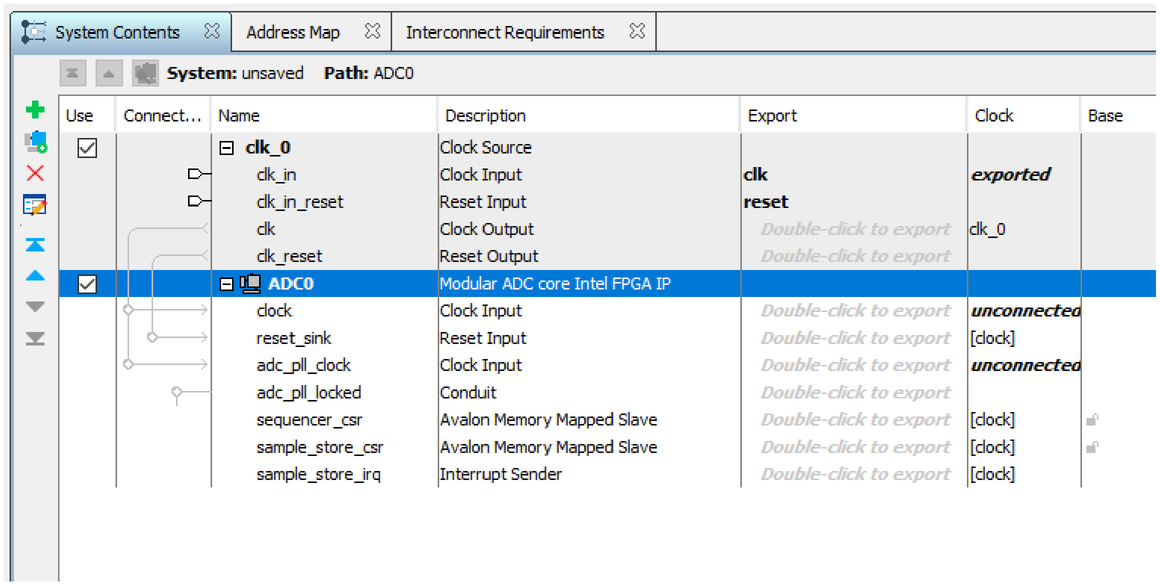

Channels | Sequencer
CH0 | CH1 | CH2 | CH3 | CH4 | CH5 | CH6 | CH7 | CH8 | TSD
Channel 1
☑ Use Channel 1

6. Click on CH7, and check the “Use Channel 7” box.
7. Click on the Sequencer tab.
8. Set the number of slots used to 3.
9. Set Slot 1: to CH1 and Slot 2: to CH 7.


Channels | Sequencer
Conversion Sequence Length
Number of slot used: 2

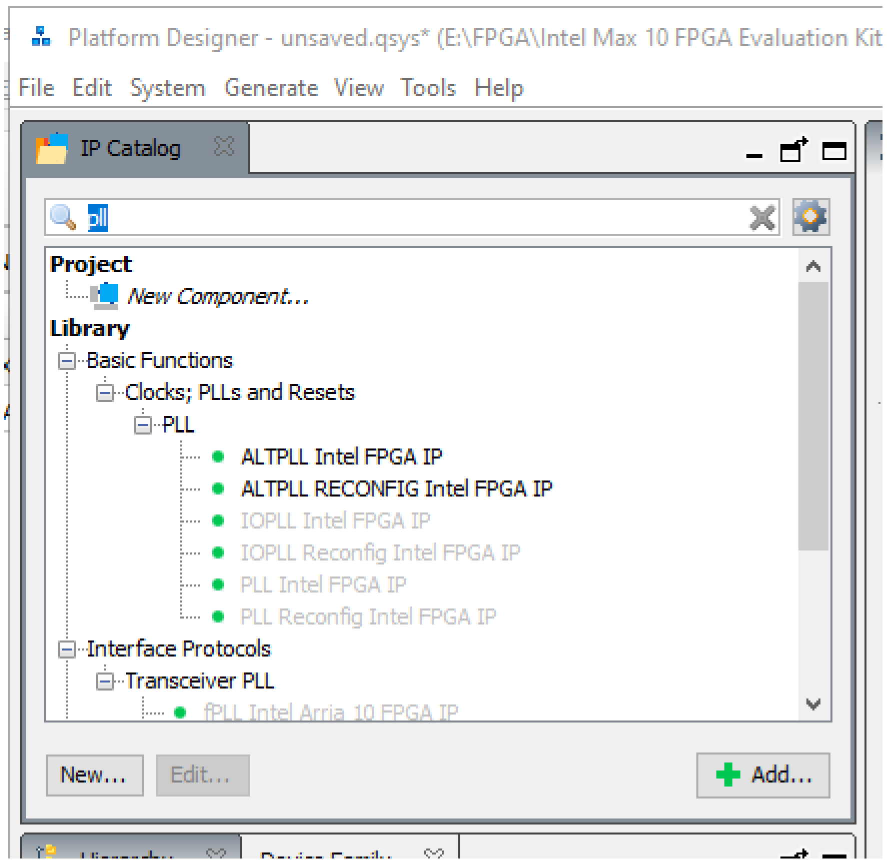
Conversion Sequence Channels
Slot 1 : CH 1
Slot 2 : CH 7

10. Click Finish.
11. The ADC will be added to the design. In the System Contents, you will see the ADC has been added to the list of devices to be interconnected. Right-click on the name and rename the device to ADC0.

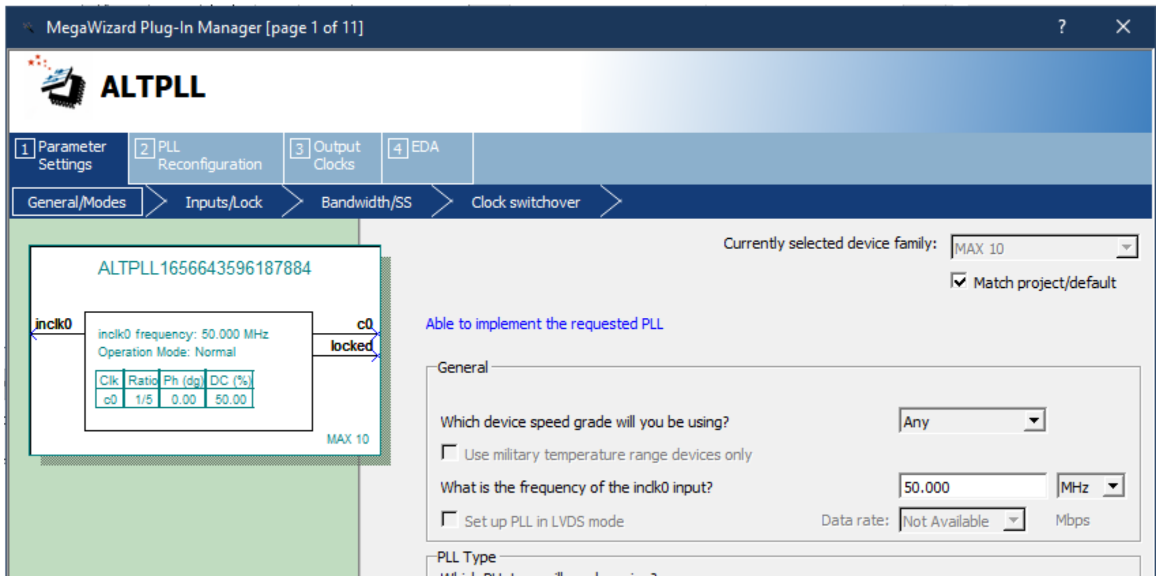


Use	Connect...	Name	Description	Export	Clock	Base
✓		clk_0	Clock Source			
		clk_in	Clock Input	clk	exported	
		clk_in_reset	Reset Input	reset		
		clk	Clock Output	Double-click to export	clk_0	
		clk_reset	Reset Output	Double-click to export		
✓		ADC0	Modular ADC core Intel FPGA IP			
		clock	Clock Input	Double-click to export	unconnected	
		reset_sink	Reset Input	Double-click to export	[clock]	
		adc_pll_clock	Clock Input	Double-click to export	unconnected	
		adc_pll_locked	Conduit	Double-click to export		
		sequencer_csr	Avalon Memory Mapped Slave	Double-click to export	[clock]	
		sample_store_csr	Avalon Memory Mapped Slave	Double-click to export	[clock]	
		sample_store_irq	Interrupt Sender	Double-click to export	[clock]	

12. Now we need to add the PLL. In the IP Catalog, type pll in the search.
13. A number of different PLLs appear in the branches, but only a few are available. Double-click on the ALTPLL Intel FPGA IP to add to the design.


Platform Designer - unsaved.qsys* (E:\FPGA\Intel Max 10 FPGA Evaluation Kit
File Edit System Generate View Tools Help
IP Catalog
pll
Project
New Component...
Library
Basic Functions
Clocks; PLLs and Resets
PLL
ALTPLL Intel FPGA IP
ALTPLL RECONFIG Intel FPGA IP
IOPLL Intel FPGA IP
IOPLL Reconfig Intel FPGA IP
PLL Intel FPGA IP
PLL Reconfig Intel FPGA IP
Interface Protocols
Transceiver PLL
fPLL Intel Arria 10 FPGA IP
New...
Edit...
Add...

14. The PLL is added and the ALTPLL Intel FPGA IP configuration page appears. The configuration page has a workflow-like presentation. Tab 1 Parameter Settings contains the general settings for the PLL. For the “What is the frequency of the inclk0 input?” set the value to 50.000 MHz. The evaluation kit has a 50 MHz oscillator.



Currently selected device family: MAX 10
[✓] Match project/default

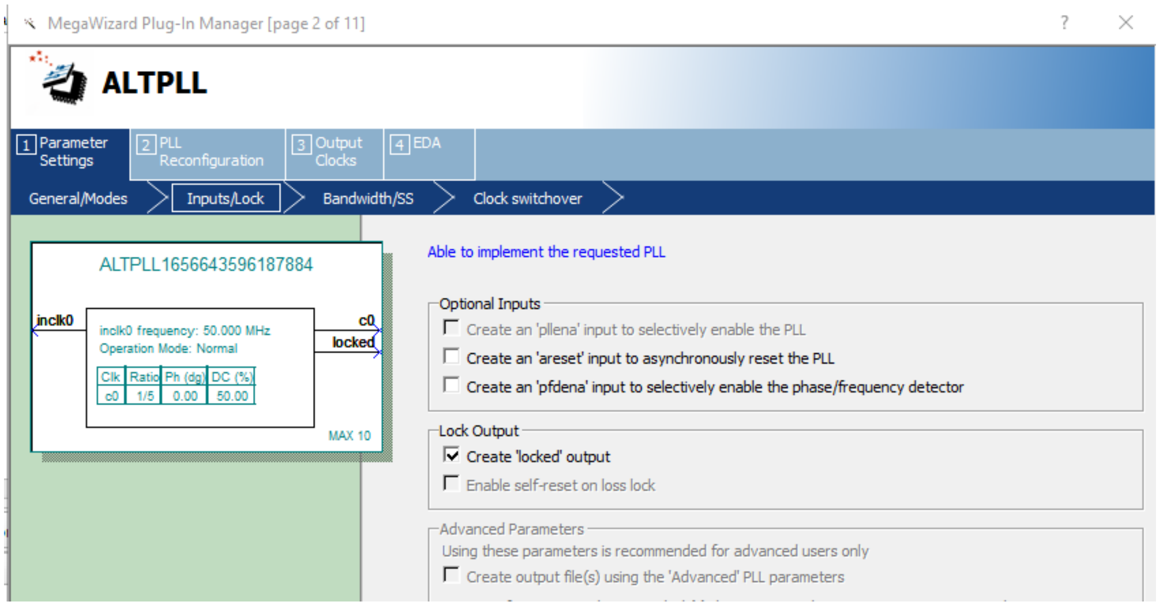
Able to implement the requested PLL

General

Which device speed grade will you be using? Any
Use military temperature range devices only
What is the frequency of the inclk0 input? 50.000 MHz
Set up PLL in LVDS mode   Data rate: Not Available Mbps

PLL Type

15. Click Next.
16. Optional: Uncheck the box next to “Create an ‘areset’ input to asynchronously reset the PLL”. This signal is not needed for this design, and this will remove one warning from the list. Leave Create ‘locked’ output checked.



inclk0 frequency: 50.000 MHz
Operation Mode: Normal

Clk	Ratio	Ph (dg)	DC (%)
c0	1/5	0.00	50.00

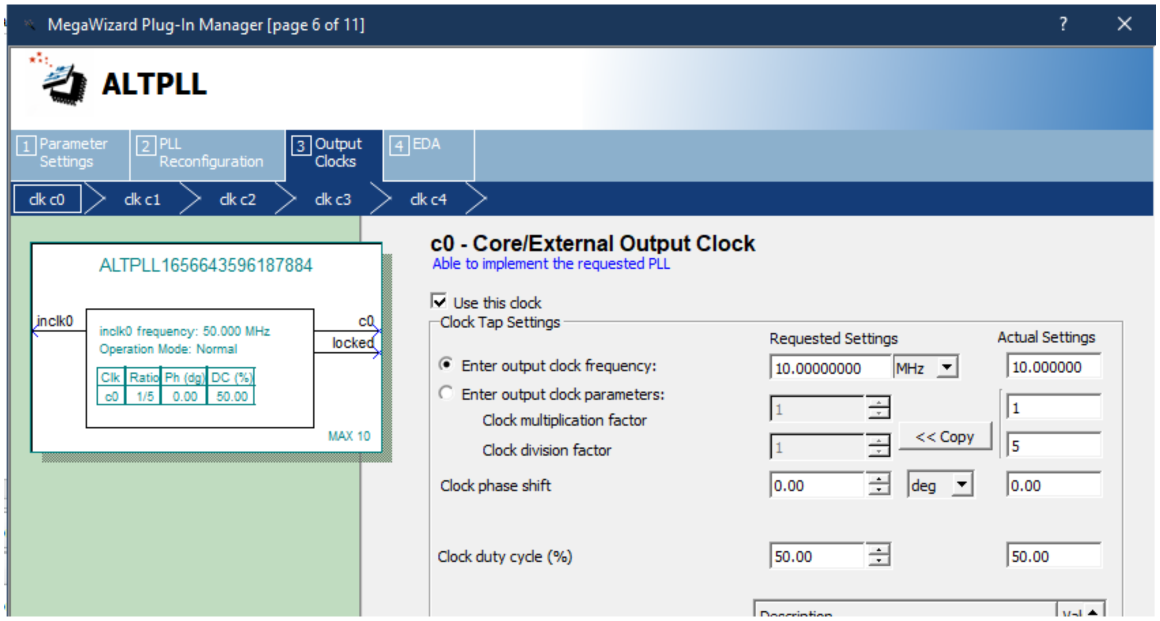
Able to implement the requested PLL

Optional Inputs
- [ ] Create an 'pllena' input to selectively enable the PLL
- [ ] Create an 'areset' input to asynchronously reset the PLL
- [ ] Create an 'pfdena' input to selectively enable the phase/frequency detector

Lock Output
- [x] Create 'locked' output
- [ ] Enable self-reset on loss lock

Advanced Parameters
Using these parameters is recommended for advanced users only
- [ ] Create output file(s) using the 'Advanced' PLL parameters

17. Click on 3. Output Clocks tab.
18. There are 5 output clock settings. All we need is clk c0. Under clk c0 click the radio button next to Enter output clock frequency.
19. Set the Requested Settings to 10.00 MHz. This is to match the input clock frequency of the ADC.



inclk0 frequency: 50.000 MHz
Operation Mode: Normal

Clk	Ratio	Ph (dg)	DC (%)
c0	1/5	0.00	50.00

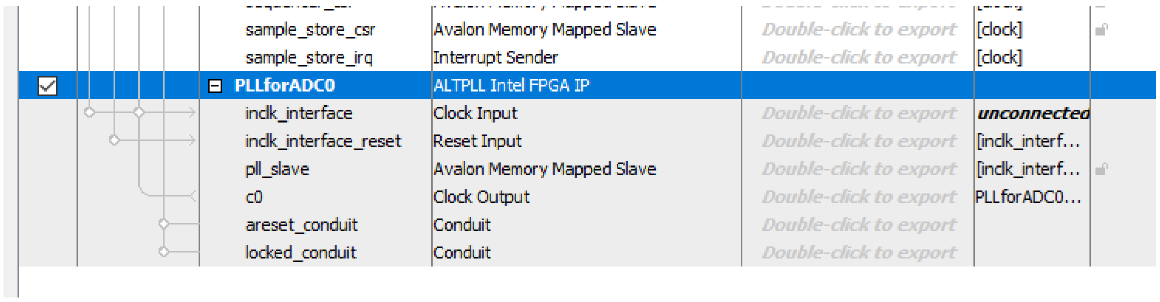
c0 - Core/External Output Clock
Able to implement the requested PLL

- [x] Use this clock

Clock Tap Settings

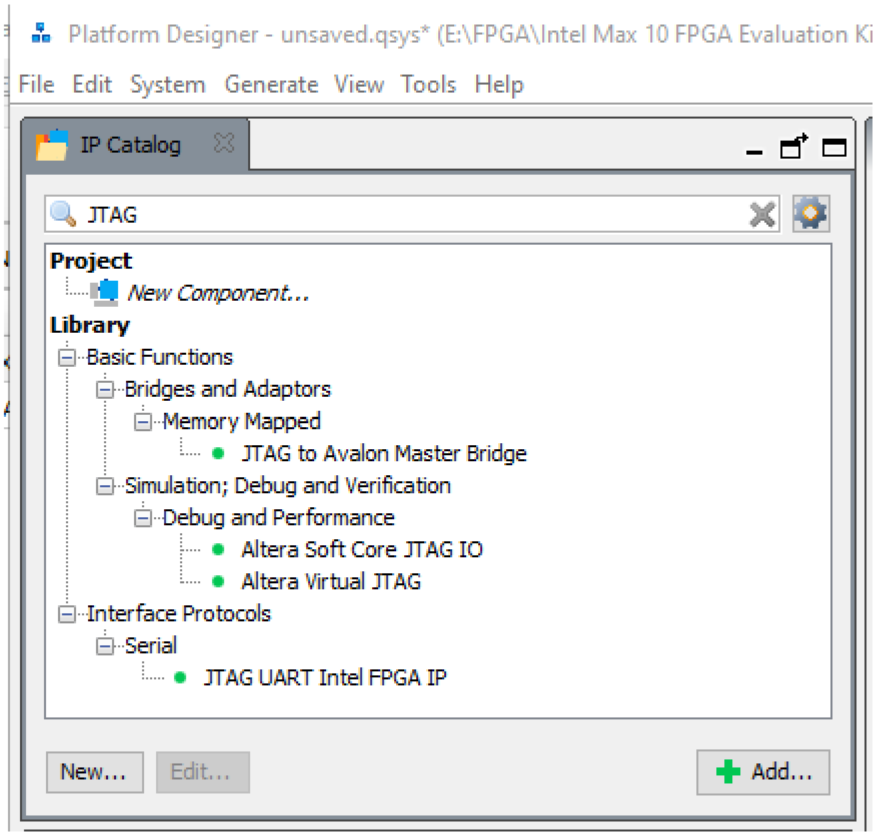
	Requested Settings		Actual Settings
[✓] Enter output clock frequency:	10.00000000	MHz	10.000000
Enter output clock parameters:			
Clock multiplication factor	1		1
Clock division factor	1		5
Clock phase shift	0.00	deg	0.00
Clock duty cycle (%)	50.00		50.00

20. Click Finish.
21. The PLL is added to the design. Rename the PLL to PLLforADC0. As there are 4 PLLs in the MAX 10, a good name helps show what the PLL is connected to.



sample_store_csr	Avalon Memory Mapped Slave	Double-click to export	[clock]	
sample_store_irq	Interrupt Sender	Double-click to export	[clock]	
☑ PLLforADC0	ALTPLL Intel FPGA IP			
inclk_interface	Clock Input	Double-click to export	unconnected	
inclk_interface_reset	Reset Input	Double-click to export	[inclk_interf...	
pll_slave	Avalon Memory Mapped Slave	Double-click to export	[inclk_interf...	
c0	Clock Output	Double-click to export	PLLforADC0...	
areset_conduit	Conduit	Double-click to export		
locked_conduit	Conduit	Double-click to export		

22. Last item to add is the JTAG bridge so we can interact with the ADC output using the ADC Tool. In the IP catalog, type JTAG in the search.



Platform Designer - unsaved.qsys* (E:\FPGA\Intel Max 10 FPGA Evaluation Ki

File Edit System Generate View Tools Help

IP Catalog

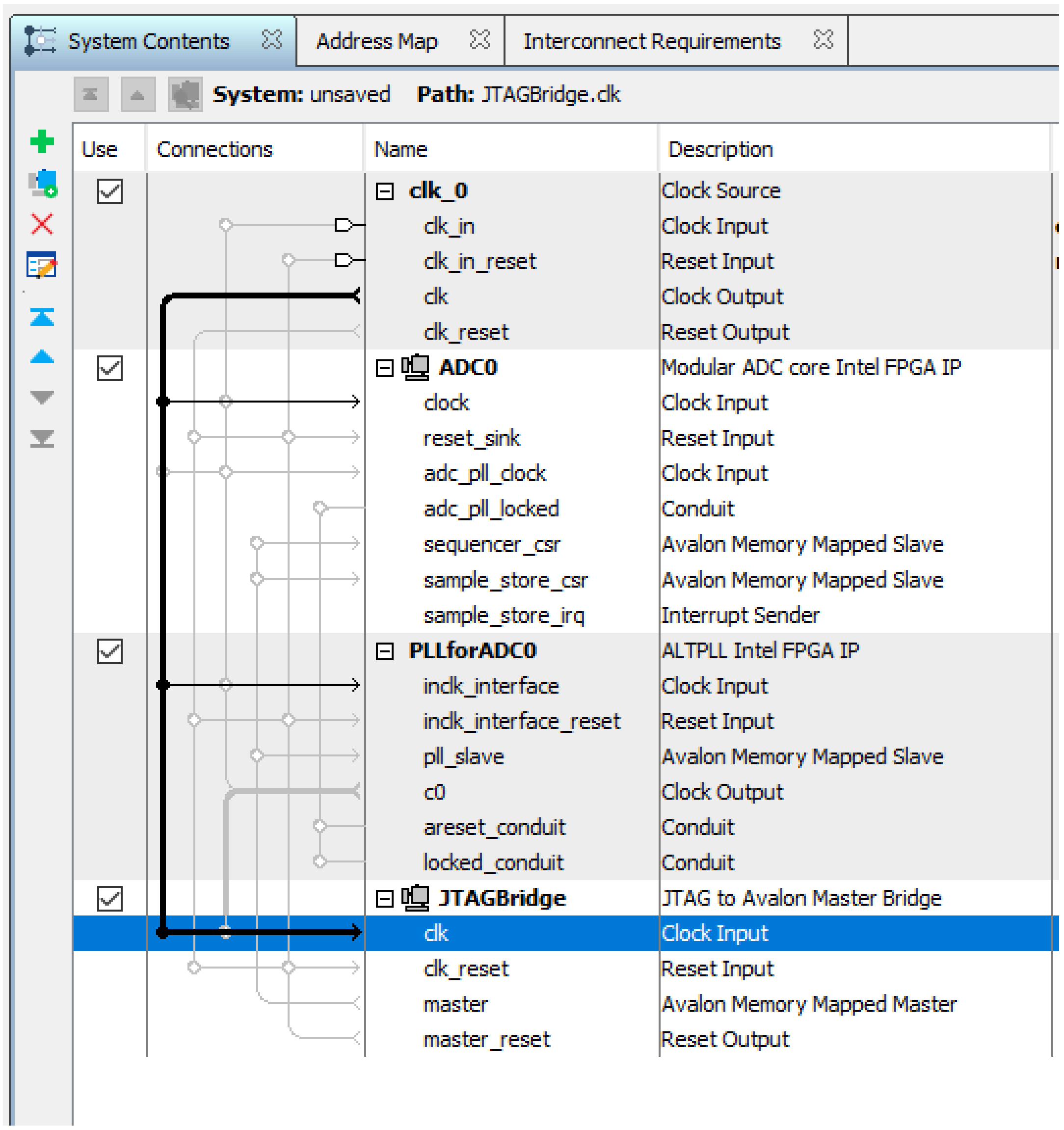
JTAG

Project
- *New Component...*

Library
- Basic Functions
  - Bridges and Adaptors
    - Memory Mapped
      - ● JTAG to Avalon Master Bridge
  - Simulation; Debug and Verification
    - Debug and Performance
      - ● Altera Soft Core JTAG IO
      - ● Altera Virtual JTAG
- Interface Protocols
  - Serial
    - ● JTAG UART Intel FPGA IP

New... Edit... Add...

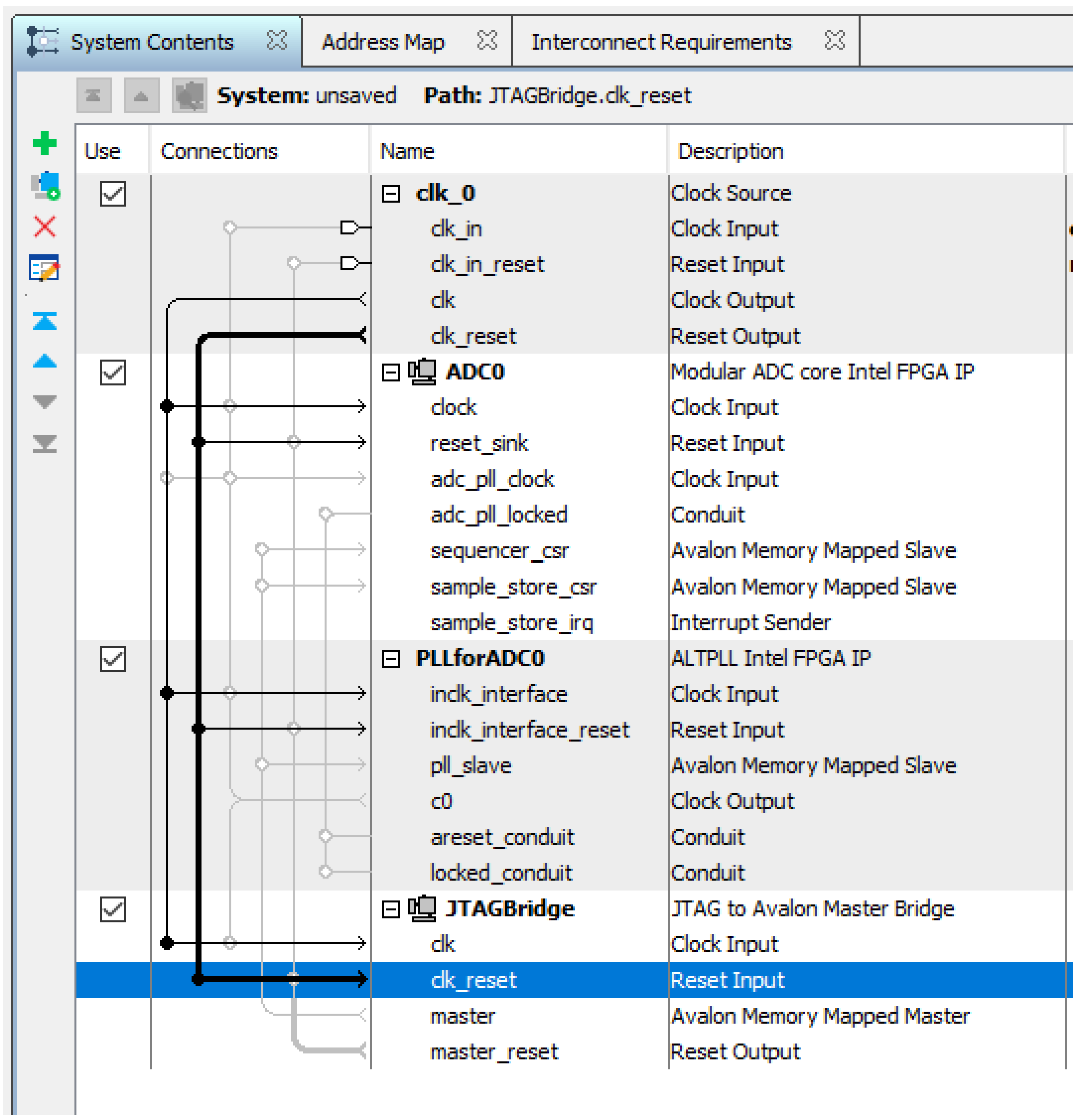
23. Double-click on the JTAG to Avalon Master Bridge to add it to the design.
24. A configuration page will appear. Click Finish.
25. Rename the JTAG to JTAGBridge.
26. Now we need to connect everything together. The first step is to connect clk_0 output to the 3 other devices. In the System Contents tab, click on the dots that connect clk_0 Clock Output to the clock inputs for all three devices. Don't connect the adc_pll_clock to this line.


System Contents | Address Map | Interconnect Requirements

System: unsaved **Path:** JTAGBridge.clk

Use	Connections	Name	Description
✓		⊟ **clk_0**	Clock Source
		clk_in	Clock Input
		clk_in_reset	Reset Input
		clk	Clock Output
		clk_reset	Reset Output
✓		⊟ **ADC0**	Modular ADC core Intel FPGA IP
		clock	Clock Input
		reset_sink	Reset Input
		adc_pll_clock	Clock Input
		adc_pll_locked	Conduit
		sequencer_csr	Avalon Memory Mapped Slave
		sample_store_csr	Avalon Memory Mapped Slave
		sample_store_irq	Interrupt Sender
✓		⊟ **PLLforADC0**	ALTPLL Intel FPGA IP
		inclk_interface	Clock Input
		inclk_interface_reset	Reset Input
		pll_slave	Avalon Memory Mapped Slave
		c0	Clock Output
		areset_conduit	Conduit
		locked_conduit	Conduit
✓		⊟ **JTAGBridge**	JTAG to Avalon Master Bridge
		clk	Clock Input
		clk_reset	Reset Input
		master	Avalon Memory Mapped Master
		master_reset	Reset Output

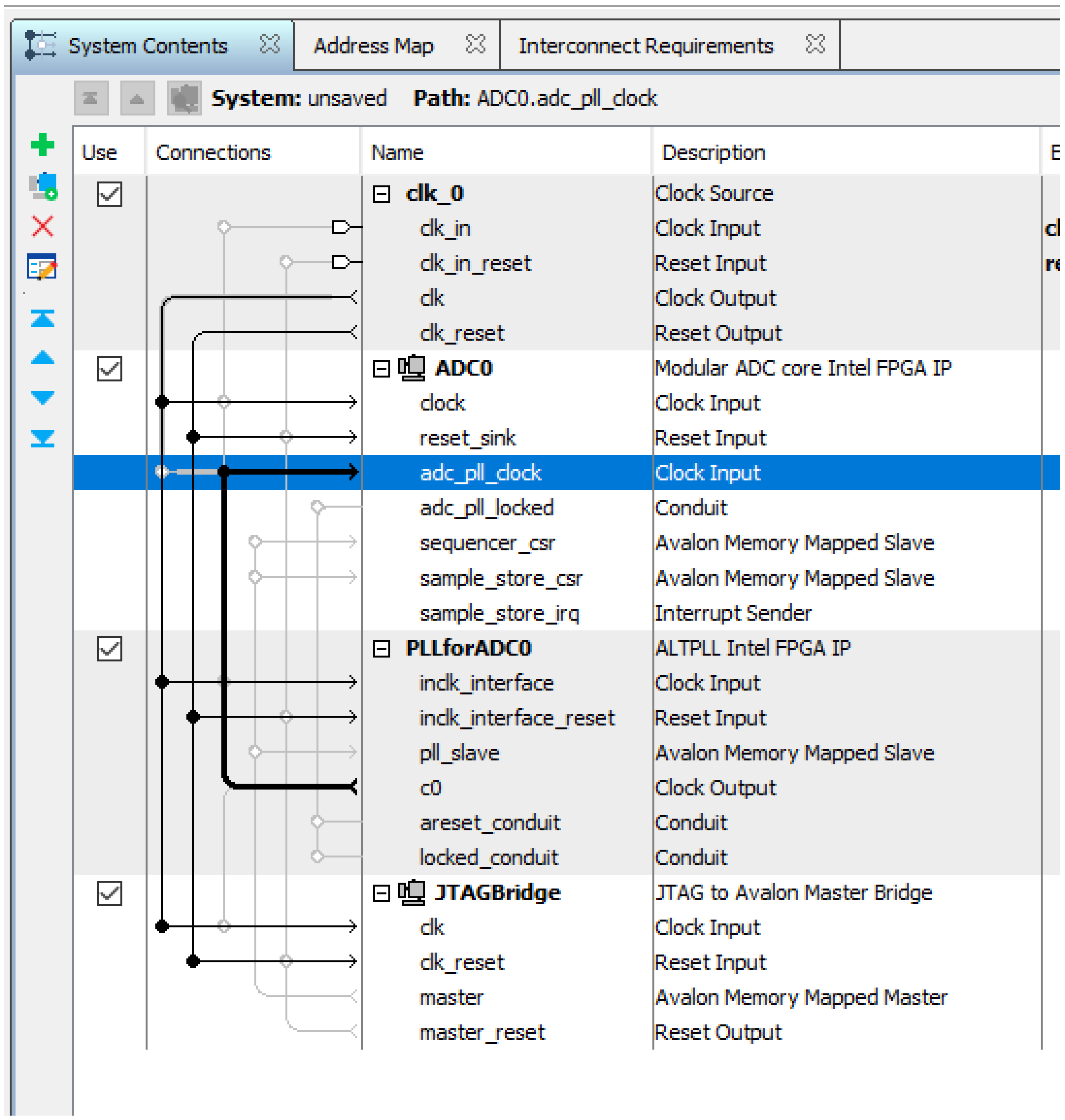
27. Now connect clk_0’s clk_reset to the 3 devices' Reset Input pins.



System: unsaved **Path:** JTAGBridge.clk_reset

Use	Connections	Name	Description
☑		⊟ **clk_0**	Clock Source
		clk_in	Clock Input
		clk_in_reset	Reset Input
		clk	Clock Output
		clk_reset	Reset Output
☑		⊟ **ADC0**	Modular ADC core Intel FPGA IP
		clock	Clock Input
		reset_sink	Reset Input
		adc_pll_clock	Clock Input
		adc_pll_locked	Conduit
		sequencer_csr	Avalon Memory Mapped Slave
		sample_store_csr	Avalon Memory Mapped Slave
		sample_store_irq	Interrupt Sender
☑		⊟ **PLLforADC0**	ALTPLL Intel FPGA IP
		inclk_interface	Clock Input
		inclk_interface_reset	Reset Input
		pll_slave	Avalon Memory Mapped Slave
		c0	Clock Output
		areset_conduit	Conduit
		locked_conduit	Conduit
☑		⊟ **JTAGBridge**	JTAG to Avalon Master Bridge
		clk	Clock Input
		clk_reset	Reset Input
		master	Avalon Memory Mapped Master
		master_reset	Reset Output

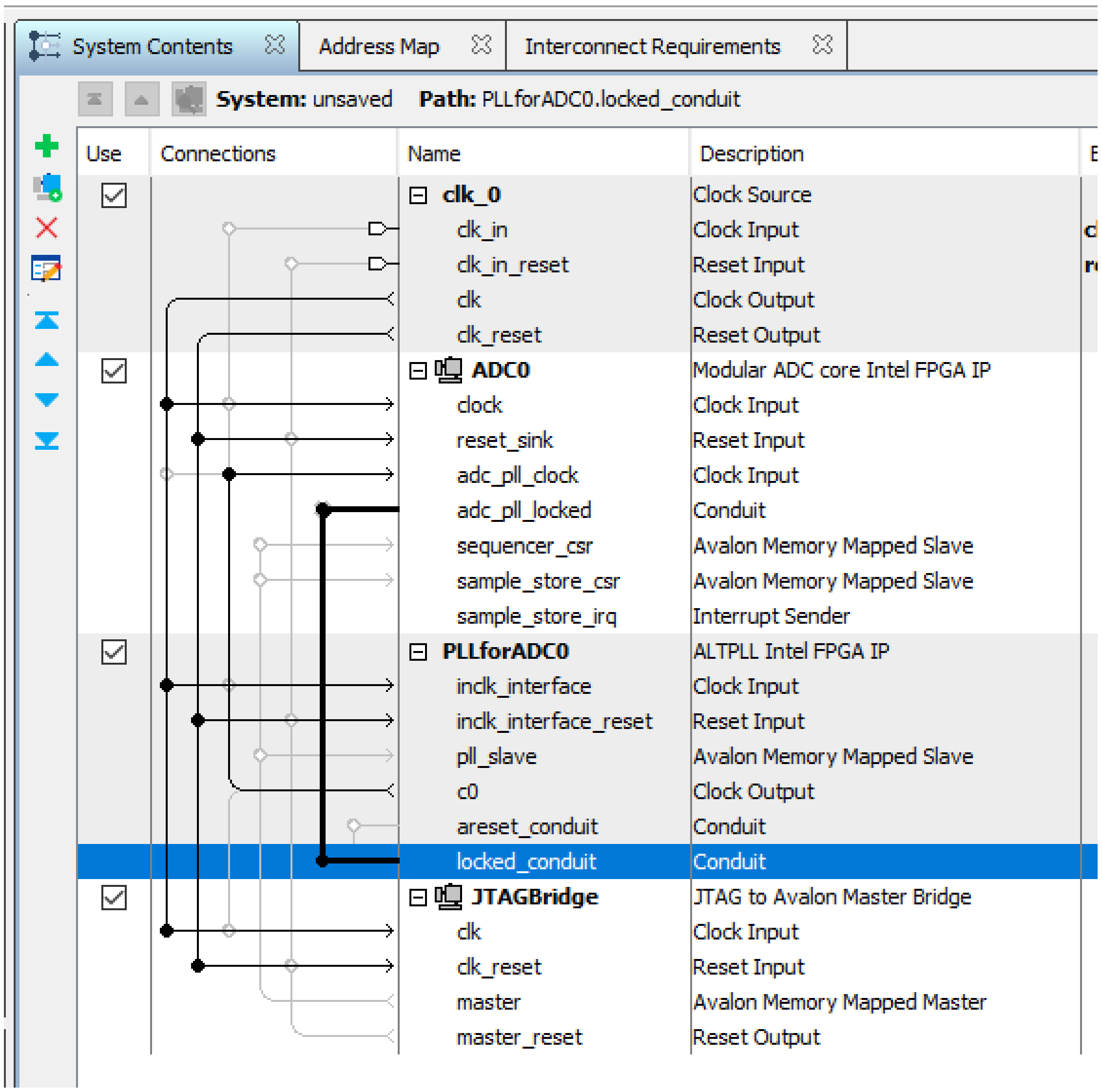
28. Next, we connect the PLL output clock (c0) to the ADC PLL clock input (adc_pll_clock).



System: unsaved Path: ADC0.adc_pll_clock

Use	Connections	Name	Description
✓		clk_0	Clock Source
		clk_in	Clock Input
		clk_in_reset	Reset Input
		clk	Clock Output
		clk_reset	Reset Output
✓		ADC0	Modular ADC core Intel FPGA IP
		clock	Clock Input
		reset_sink	Reset Input
		adc_pll_clock	Clock Input
		adc_pll_locked	Conduit
		sequencer_csr	Avalon Memory Mapped Slave
		sample_store_csr	Avalon Memory Mapped Slave
		sample_store_irq	Interrupt Sender
✓		PLLforADC0	ALTPLL Intel FPGA IP
		inclk_interface	Clock Input
		inclk_interface_reset	Reset Input
		pll_slave	Avalon Memory Mapped Slave
		c0	Clock Output
		areset_conduit	Conduit
		locked_conduit	Conduit
✓		JTAGBridge	JTAG to Avalon Master Bridge
		clk	Clock Input
		clk_reset	Reset Input
		master	Avalon Memory Mapped Master
		master_reset	Reset Output

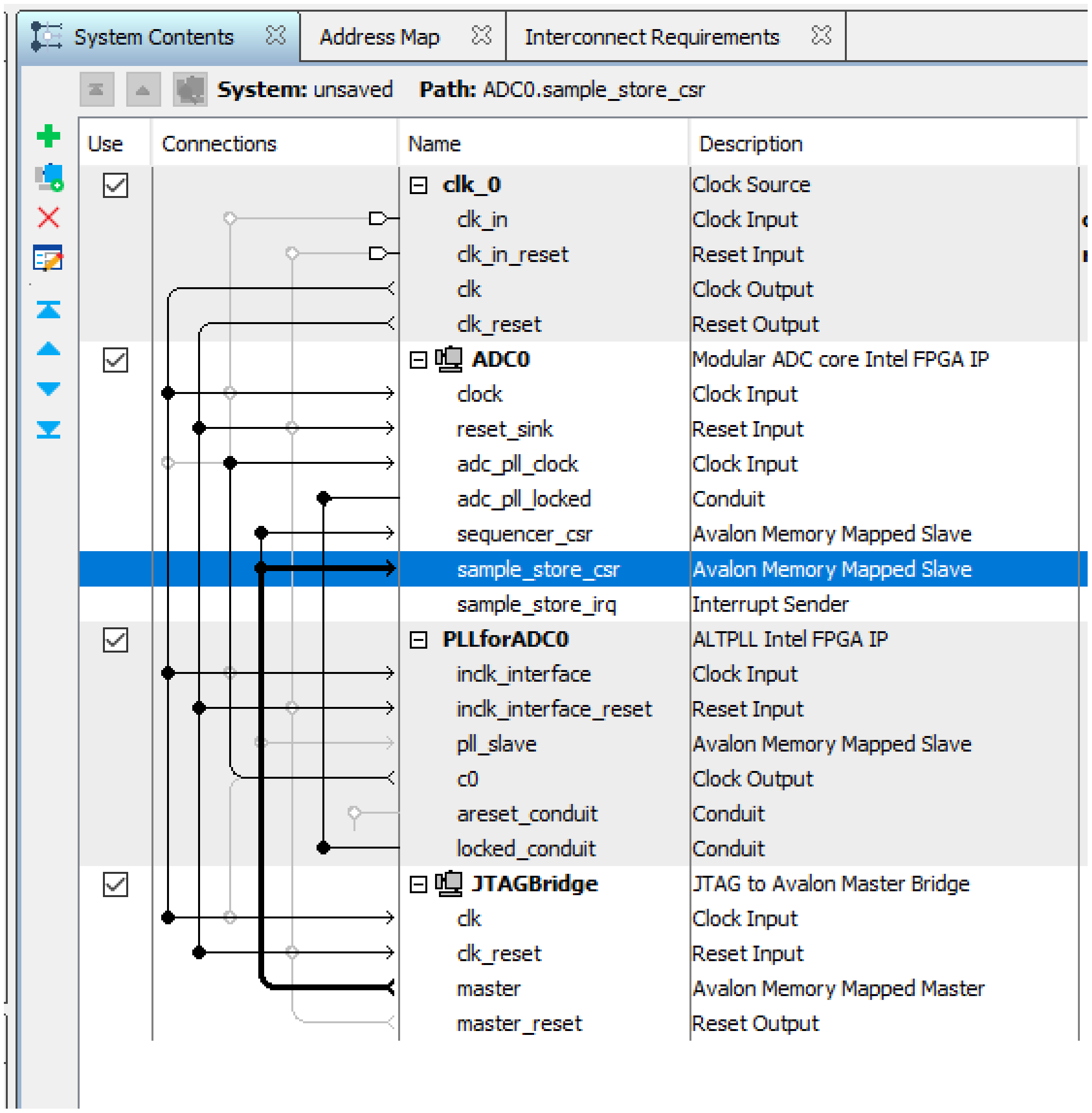
29. Connect the PLL’s locked_conduit to the ADC’s asc_pll_locked pin.



System: unsaved **Path:** PLLforADC0.locked_conduit

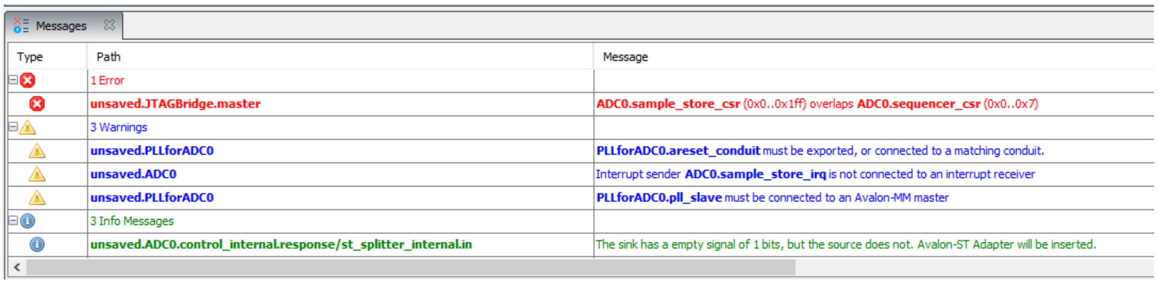
Use	Connections	Name	Description	E
✓		**clk_0**	Clock Source	
		clk_in	Clock Input	c
		clk_in_reset	Reset Input	r
		clk	Clock Output	
		clk_reset	Reset Output	
✓		**ADC0**	Modular ADC core Intel FPGA IP	
		clock	Clock Input	
		reset_sink	Reset Input	
		adc_pll_clock	Clock Input	
		adc_pll_locked	Conduit	
		sequencer_csr	Avalon Memory Mapped Slave	
		sample_store_csr	Avalon Memory Mapped Slave	
		sample_store_irq	Interrupt Sender	
✓		**PLLforADC0**	ALTPLL Intel FPGA IP	
		inclk_interface	Clock Input	
		inclk_interface_reset	Reset Input	
		pll_slave	Avalon Memory Mapped Slave	
		c0	Clock Output	
		areset_conduit	Conduit	
		locked_conduit	Conduit	
✓		**JTAGBridge**	JTAG to Avalon Master Bridge	
		clk	Clock Input	
		clk_reset	Reset Input	
		master	Avalon Memory Mapped Master	
		master_reset	Reset Output	

30. We need the JTAG to be able to access the data. Connect the JTAGBridge's master to ADC0's sequence_csr and sample_store_csr.



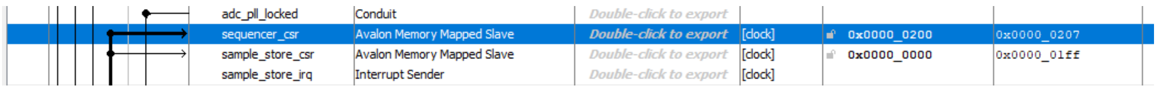
Use	Name	Description
✓	clk_0	Clock Source
	clk_in	Clock Input
	clk_in_reset	Reset Input
	clk	Clock Output
	clk_reset	Reset Output
✓	ADC0	Modular ADC core Intel FPGA IP
	clock	Clock Input
	reset_sink	Reset Input
	adc_pll_clock	Clock Input
	adc_pll_locked	Conduit
	sequencer_csr	Avalon Memory Mapped Slave
	sample_store_csr	Avalon Memory Mapped Slave
	sample_store_irq	Interrupt Sender
✓	PLLforADC0	ALTPLL Intel FPGA IP
	inclk_interface	Clock Input
	inclk_interface_reset	Reset Input
	pll_slave	Avalon Memory Mapped Slave
	c0	Clock Output
	areset_conduit	Conduit
	locked_conduit	Conduit
✓	JTAGBridge	JTAG to Avalon Master Bridge
	clk	Clock Input
	clk_reset	Reset Input
	master	Avalon Memory Mapped Master
	master_reset	Reset Output

There will be an error after these connections are made since ADC lines have the same address.



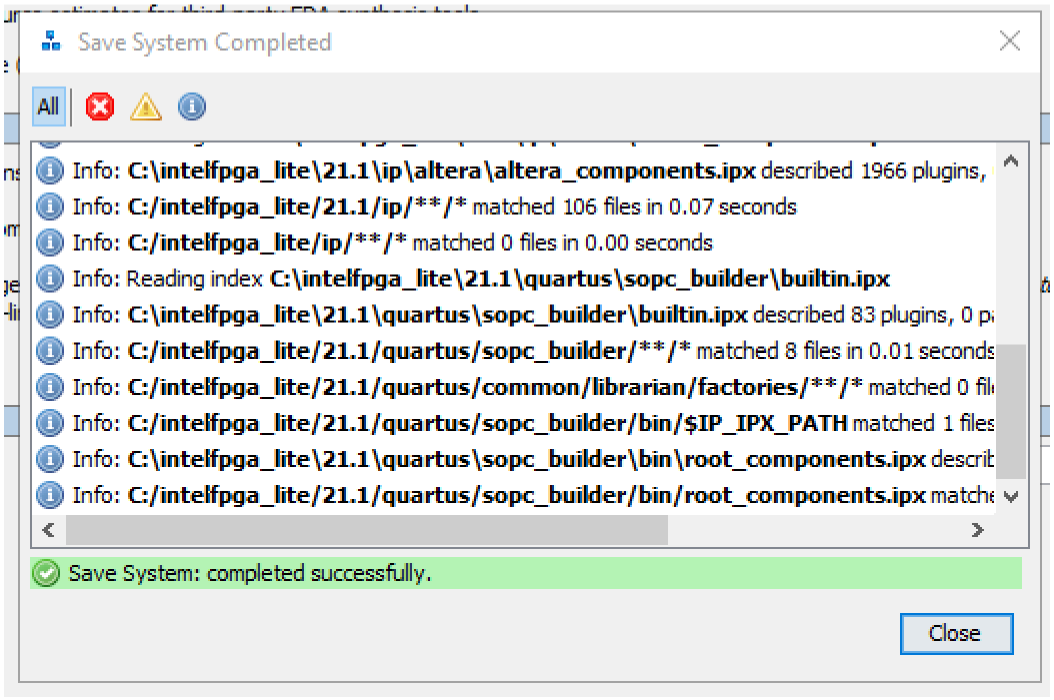
Type	Path	Message
	1 Error	
	unsaved.JTAGBridge.master	ADC0.sample_store_csr (0x0..0x1ff) overlaps ADC0.sequencer_csr (0x0..0x7)
	3 Warnings	
	unsaved.PLLforADC0	PLLforADC0.areset_conduit must be exported, or connected to a matching conduit.
	unsaved.ADC0	Interrupt sender ADC0.sample_store_irq is not connected to an interrupt receiver
	unsaved.PLLforADC0	PLLforADC0.pll_slave must be connected to an Avalon-MM master
	3 Info Messages	
	unsaved.ADC0.control_internal.response/st_splitter_internal.in	The sink has a empty signal of 1 bits, but the source does not. Avalon-ST Adapter will be inserted.

31. To fix this error, change the sequencer_ser base address to 200. Click on the base value of 0x000_0000 and change the value to 0x0000_0200. The error will go away.



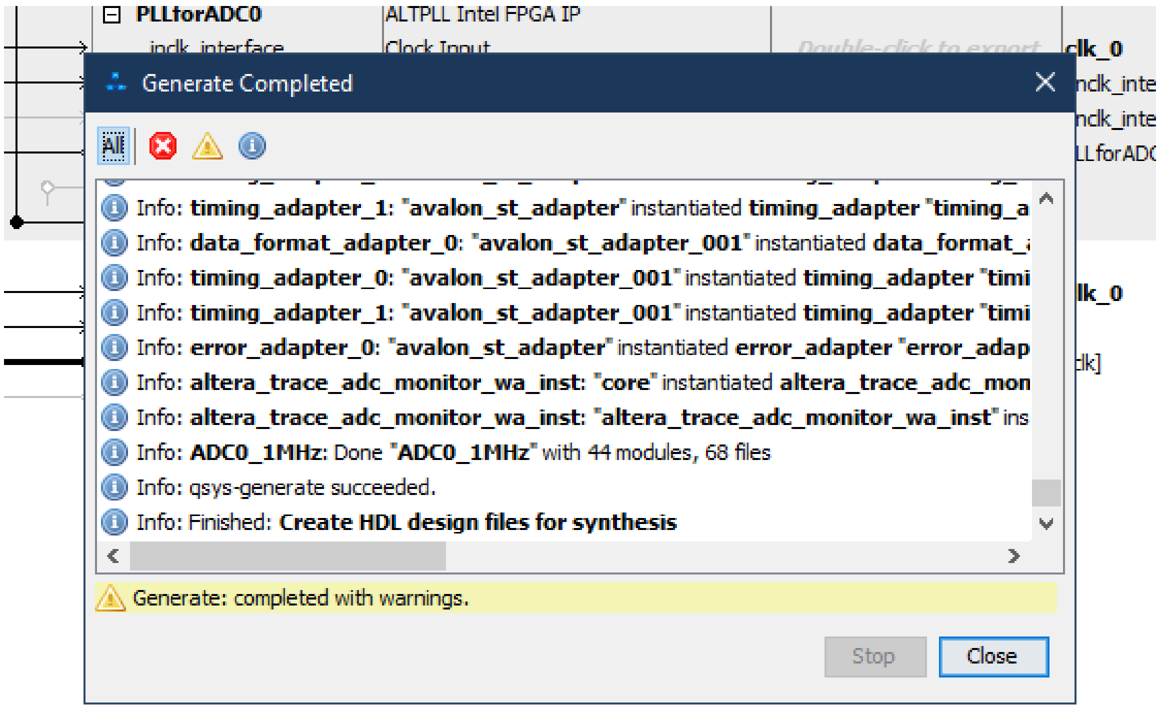
adc_pll_locked	Conduit	Double-click to export			
sequencer_csr	Avalon Memory Mapped Slave	Double-click to export	[clock]	0x0000_0200	0x0000_0207
sample_store_csr	Avalon Memory Mapped Slave	Double-click to export	[clock]	0x0000_0000	0x0000_01ff
sample_store_irq	Interrupt Sender	Double-click to export	[clock]		

32. The design needs a base address to make the memory-mapped components unique. From the menu, select System->Assign Base Address.
33. We can now generate the HDL code. Click on the Generate HDL… button.
34. A Generation dialog appears. This will generate a block symbol file (.bsf) that we can use in the Block Diagram designer of Quartus. Keep the output Path the same as the project folder and click the Generate button.
35. A dialog will appear asking to save the system. Click save.
36. Name the system ADC0_1MHz.qsys and click save.
37. The save process should complete successfully. Click close.



```
Info: C:\intelfpga_lite\21.1\ip\altera\altera_components.ipx described 1966 plugins,
Info: C:/intelfpga_lite/21.1/ip/**/* matched 106 files in 0.07 seconds
Info: C:/intelfpga_lite/ip/**/* matched 0 files in 0.00 seconds
Info: Reading index C:\intelfpga_lite\21.1\quartus\sopc_builder\builtin.ipx
Info: C:\intelfpga_lite\21.1\quartus\sopc_builder\builtin.ipx described 83 plugins, 0 p
Info: C:/intelfpga_lite/21.1/quartus/sopc_builder/**/* matched 8 files in 0.01 seconds
Info: C:/intelfpga_lite/21.1/quartus/common/librarian/factories/**/* matched 0 fil
Info: C:/intelfpga_lite/21.1/quartus/sopc_builder/bin/$IP_IPX_PATH matched 1 files
Info: C:\intelfpga_lite\21.1\quartus\sopc_builder\bin\root_components.ipx describ
Info: C:/intelfpga_lite/21.1/quartus/sopc_builder/bin/root_components.ipx matche
Save System: completed successfully.
```

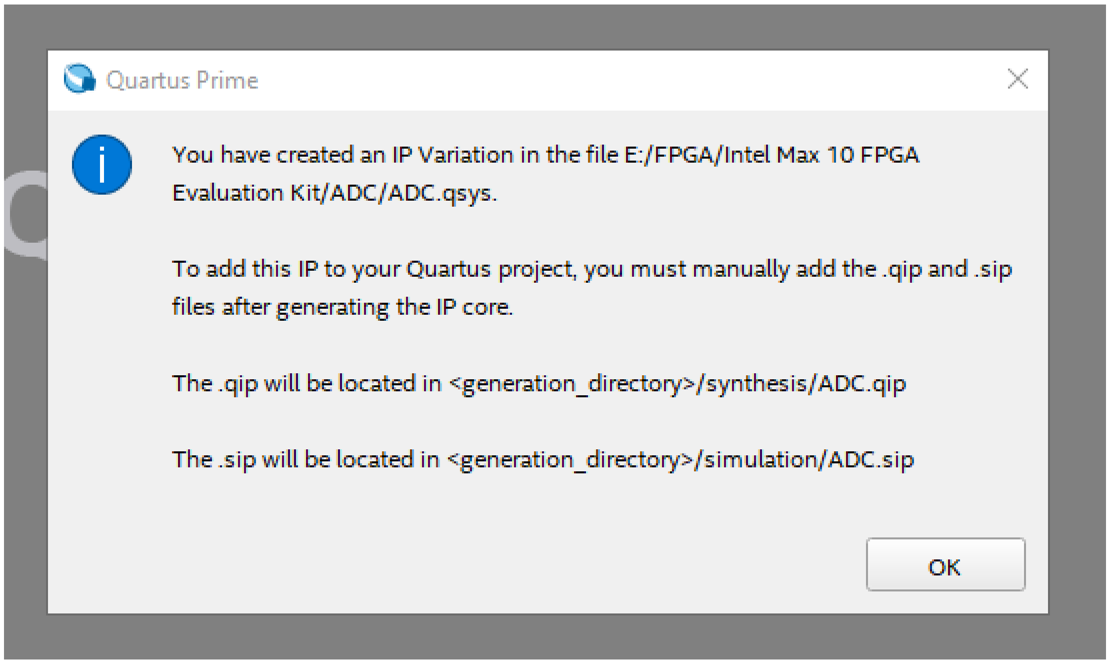
38. The Generate HDL process will run and should finish successfully. There will be warnings for items not connected, but for this exercise, they will not be needed. Click Close.



```
Info: timing_adapter_1: "avalon_st_adapter" instantiated timing_adapter "timing_a
Info: data_format_adapter_0: "avalon_st_adapter_001" instantiated data_format_
Info: timing_adapter_0: "avalon_st_adapter_001" instantiated timing_adapter "timi
Info: timing_adapter_1: "avalon_st_adapter_001" instantiated timing_adapter "timi
Info: error_adapter_0: "avalon_st_adapter" instantiated error_adapter "error_adap
Info: altera_trace_adc_monitor_wa_inst: "core" instantiated altera_trace_adc_mon
Info: altera_trace_adc_monitor_wa_inst: "altera_trace_adc_monitor_wa_inst" ins
Info: ADC0_1MHz: Done "ADC0_1MHz" with 44 modules, 68 files
Info: qsys-generate succeeded.
Info: Finished: Create HDL design files for synthesis
```

Generate: completed with warnings.

39. Click Finish to close the Platform Designer.



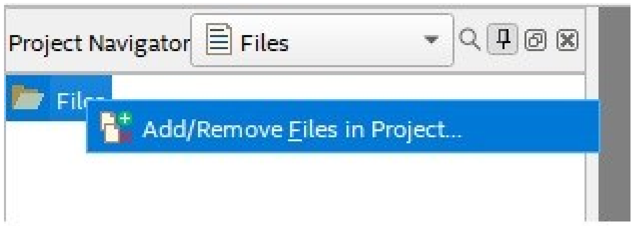
You have created an IP Variation in the file E:/FPGA/Intel Max 10 FPGA Evaluation Kit/ADC/ADC.qsys.

To add this IP to your Quartus project, you must manually add the .qip and .sip files after generating the IP core.

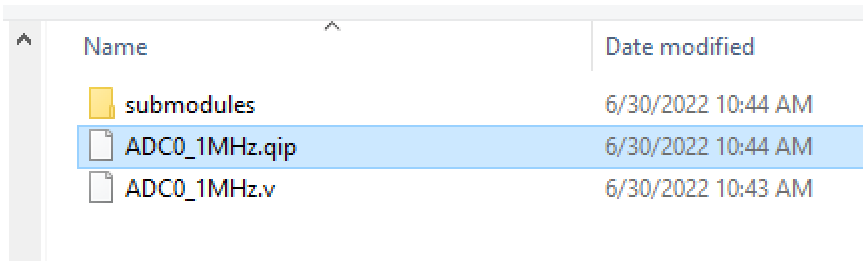
The .qip will be located in <generation_directory>/synthesis/ADC.qip

The .sip will be located in <generation_directory>/simulation/ADC.sip

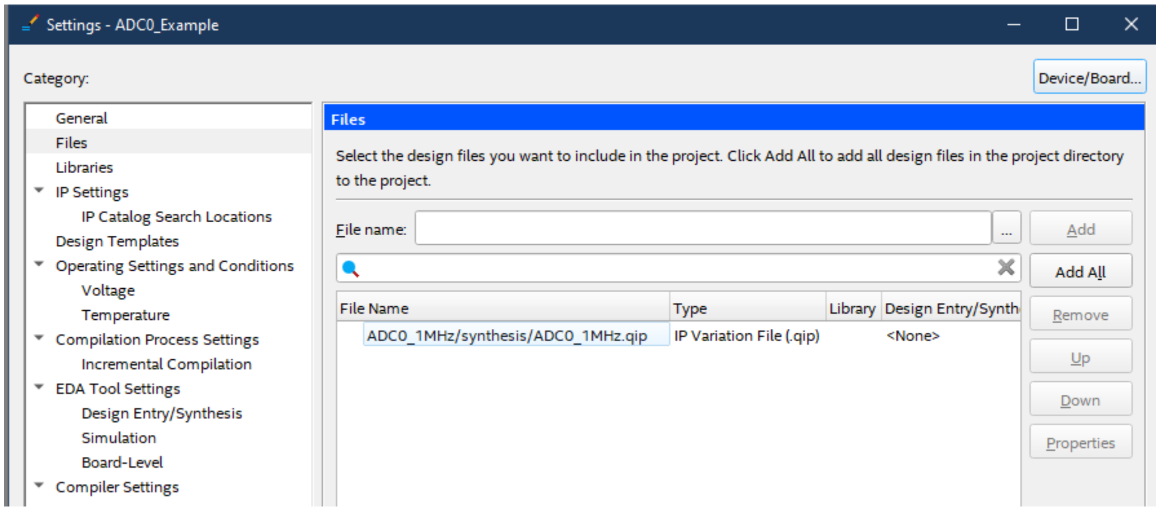
40. Once the ADC0_1MHz design has been created, Quartus automatically reminds you to add the ADC0_1MHz IP to the ADC design project. Click Ok.
41. First, we need to add the newly create ADC block to the project. In the Project Navigator, click on the drop-down and select Files.
42. Right-Click on Files and select Add/Remote Files in Project.



43. A Settings-ADC page appears with Files on the left highlighted. Click the three dots browse button for File name, and navigate to ADC_Example\ADC0_1MHz\synthesis folder.
44. Click on ADC0_1Mhz.qip file and click open.



Name	Date modified
submodules	6/30/2022 10:44 AM
ADC0_1MHz.qip	6/30/2022 10:44 AM
ADC0_1MHz.v	6/30/2022 10:43 AM



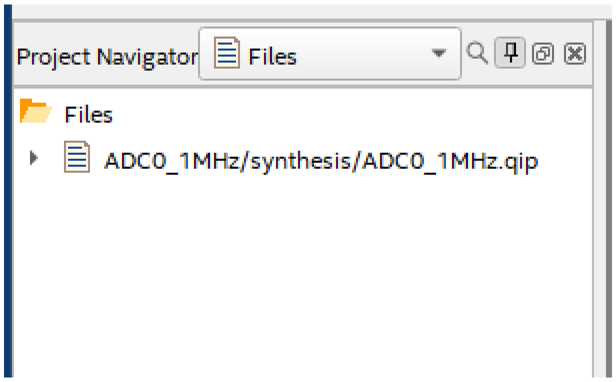
Files

Select the design files you want to include in the project. Click Add All to add all design files in the project directory to the project.

File name:

File Name	Type	Library	Design Entry/Synth
ADC0_1MHz/synthesis/ADC0_1MHz.qip	IP Variation File (.qip)		<None>

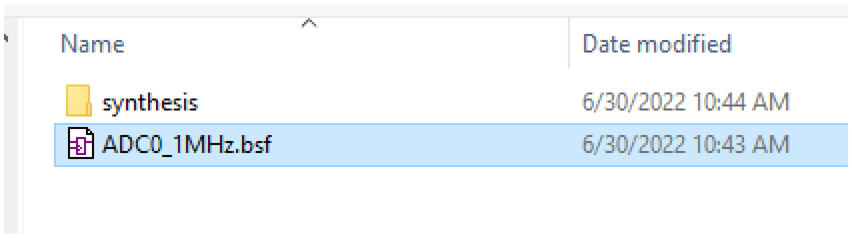
45. Click OK to close the Settings-ADC0_Example page. The qip file is added to the Project navigator list. Underneath are all the Verilog files that were generated by Platform Builder.



1.2.3 Create the Design Step 2: Block Diagram

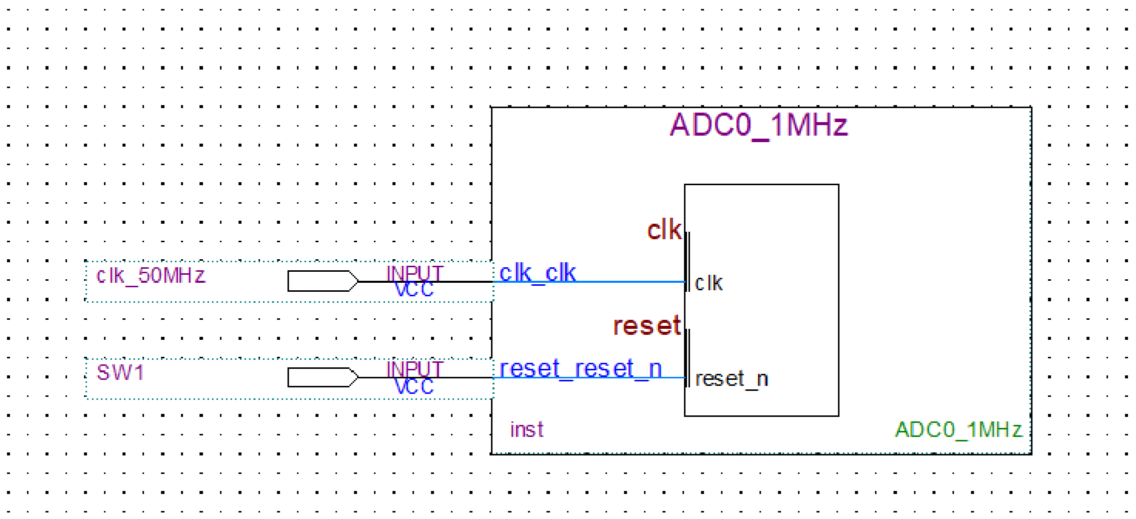
With the ADC qip file added to the project, let's create the block diagram and complete the design.

1. From the menu, select New->Block Diagram/Schematic File or the icon from the toolbar. Click Ok.
2. The symbol window appears. Double-click on the symbol windows and the symbol dialog appears.
3. Click on the 3 dots to open the file browser.
4. Browse to \ADC_Example\ADC0_1MHz folder and open the ADC0_1MHz.bsf file.



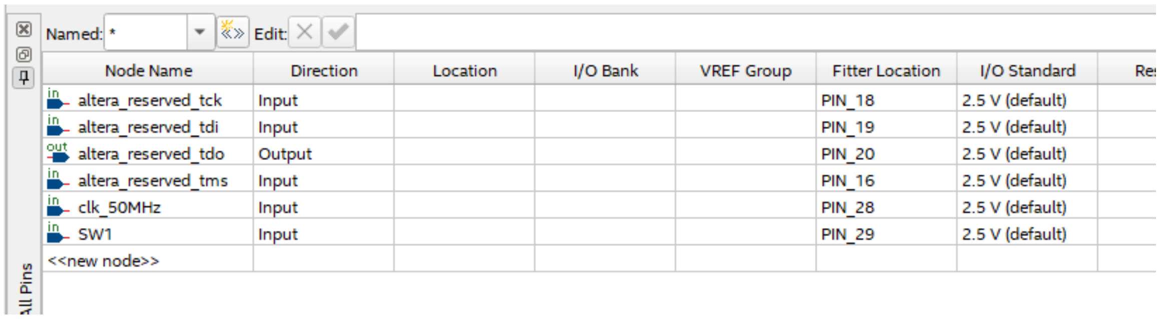
Name	Date modified
synthesis	6/30/2022 10:44 AM
ADC0_1MHz.bsf	6/30/2022 10:43 AM

5. The symbol for the ADC0_1Mhz appears. Click OK to add the symbol to the schematic.
6. Drag the mouse with the ADC0_1MHz symbol to a location on the diagram and then left-click to drop it in place.
7. Right-click on the ADC0_1MHz symbol and select Generate Pins for Symbol Ports.
8. Change the name of the clk_clk pin to clk_50MHz.
9. Change the name of the reset_reset_n to SW1. The SW1 switch on the evaluation kit is connected to the FPGA DEV_CLRN pin. The circuit for SW1 is logic 1 on startup and logic0 when pressed.



10. Save the schematic as ADC0_Example.bdf.
11. In the Task pane on the left, double-click on Fitter (Place & Route) to start the task. The analysis will take some time, and it should succeed in the end. This step helps to diagnose any errors and finds the Node Names for the pin assignments in the next step.

12. Once the process completes, the pin assignments need to be set. From the menu, select Assignments->Pin Planner or click on the icon from the toolbar. The analysis that was just run populated the Node Name list at the bottom of the Pin Planner dialog.

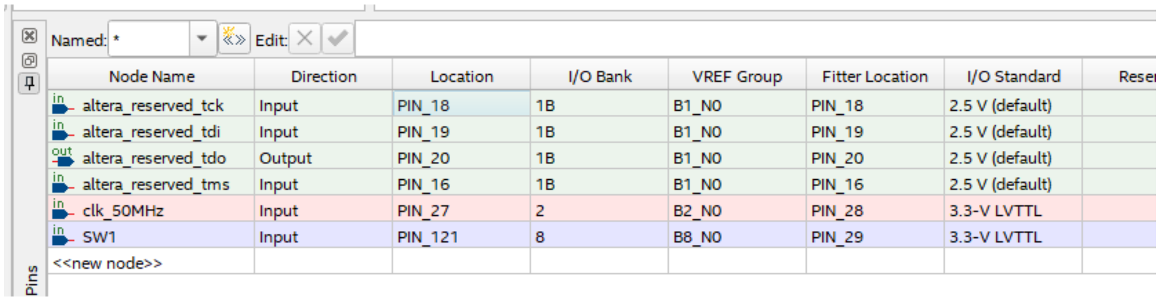


Node Name	Direction	Location	I/O Bank	VREF Group	Fitter Location	I/O Standard	Res
altera_reserved_tck	Input				PIN_18	2.5 V (default)	
altera_reserved_tdi	Input				PIN_19	2.5 V (default)	
altera_reserved_tdo	Output				PIN_20	2.5 V (default)	
altera_reserved_tms	Input				PIN_16	2.5 V (default)	
clk_50MHz	Input				PIN_28	2.5 V (default)	
SW1	Input				PIN_29	2.5 V (default)	
<<new node>>							

13. Using the board schematic, locate the pins for the SW1 and the 50MHz clock. Set the Location values for both node names. For the MAX 10 – 10M08 Evaluation Board, these values are as follows:

Node Name	Location
SW1	PIN_121
Clk_50MHz:	PIN_27
altera_reserved_tck	PIN_18
altera_reserved_tdi	PIN_19
altera_reserved_tdo	PIN_20
altera_reserved_tms	PIN_16

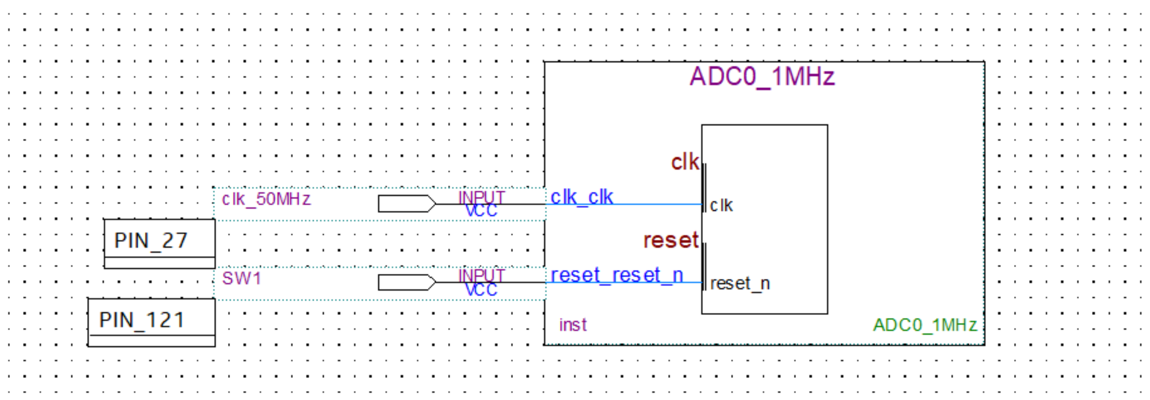
14. Set the I/O Standard to 3.3V-LVTTL for both pins. You can see from the schematic that the I/O are all tied to 3.3V.



Node Name	Direction	Location	I/O Bank	VREF Group	Fitter Location	I/O Standard	Rese
altera_reserved_tck	Input	PIN_18	1B	B1_N0	PIN_18	2.5 V (default)	
altera_reserved_tdi	Input	PIN_19	1B	B1_N0	PIN_19	2.5 V (default)	
altera_reserved_tdo	Output	PIN_20	1B	B1_N0	PIN_20	2.5 V (default)	
altera_reserved_tms	Input	PIN_16	1B	B1_N0	PIN_16	2.5 V (default)	
clk_50MHz	Input	PIN_27	2	B2_N0	PIN_28	3.3-V LVTTL	
SW1	Input	PIN_121	8	B8_N0	PIN_29	3.3-V LVTTL	
<<new node>>							

The ADC channel pins don't need to be assigned, since the ADC IP Block takes care of this already.

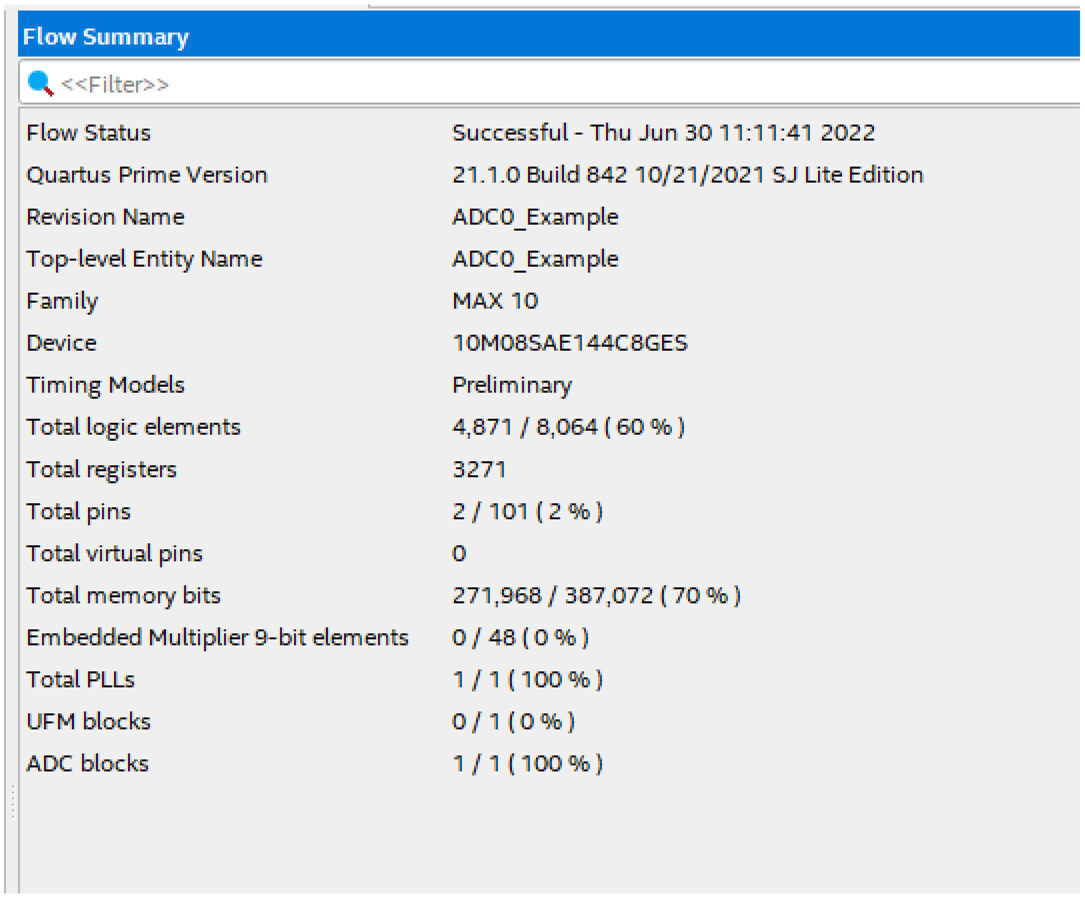
15. Close the Pin Planner when finished. The diagram gets updated with the pin numbers.



16. Save the project.

Note: A best practice at this point would be to make a backup of the project folder. Quartus can crash unexpectedly, since it appears to be written in Java.

17. Finally, compile the design. In the Task pane, right-click on Compile and Design and select Start from the context menu, or you can click on the ▶ symbol in the toolbar. Make sure the project compiles successfully.


Flow Summary

<<Filter>>

Flow Status	Successful - Thu Jun 30 11:11:41 2022
Quartus Prime Version	21.1.0 Build 842 10/21/2021 SJ Lite Edition
Revision Name	ADC0_Example
Top-level Entity Name	ADC0_Example
Family	MAX 10
Device	10M08SAE144C8GES
Timing Models	Preliminary
Total logic elements	4,871 / 8,064 (60 %)
Total registers	3271
Total pins	2 / 101 (2 %)
Total virtual pins	0
Total memory bits	271,968 / 387,072 (70 %)
Embedded Multiplier 9-bit elements	0 / 48 (0 %)
Total PLLs	1 / 1 (100 %)
UFM blocks	0 / 1 (0 %)
ADC blocks	1 / 1 (100 %)

1.2.4 Program the Board

With the design compiled, we can now test it on the board.

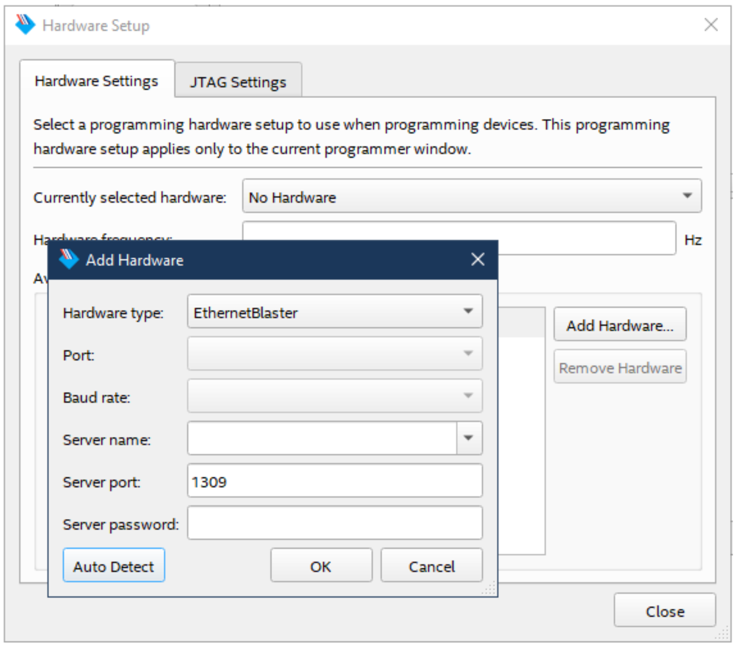
1. Connect the board and the programming cable together per the cable instructions.

Note: The MAX 10 – 10M08 Evaluation Kit doesn't come with a programming cable or built-in JTAG USB Blaster II. You will have to use either the USB Blaster II or EthernetBlaster II external cables. The EthernetBlaster II was used. DHCP setup was not working, so a direct Ethernet cable connection was made between a PC and the EthernetBlaster II. Set the static IP for the PC network card to 198.162.0.1. Access the EthernetBlaster II via a browser and then change the IP to a static IP that matches the network. The new IP address was used as the Server name. Your experience might be different.

2. Connect a signal generator to the board as follows:

   - Signal out lead to J4-1 (Arduino connector Annalog_IN0).
   - Signal ground to J2-6 (Arduino connector GND).

3. Set the function generator to 1Khz 2VPP with a DC offset enabled and set for 1V positive.
4. Power on the board and the programming cable box.
5. In Quartus Prime, from the Task pane, right-click on Program Device (Open Programmer) and select Open from the context menu or click on the icon on the toolbar.
6. When the Programmer dialog appears, click on the "Hardware Setup" button.
7. Click the Add hardware button, select the Hardware type, fill in any remaining information, and click OK.



Hardware Setup

Hardware Settings | JTAG Settings

Select a programming hardware setup to use when programming devices. This programming hardware setup applies only to the current programmer window.

Currently selected hardware: No Hardware

Add Hardware

Hardware type: EthernetBlaster

Port:

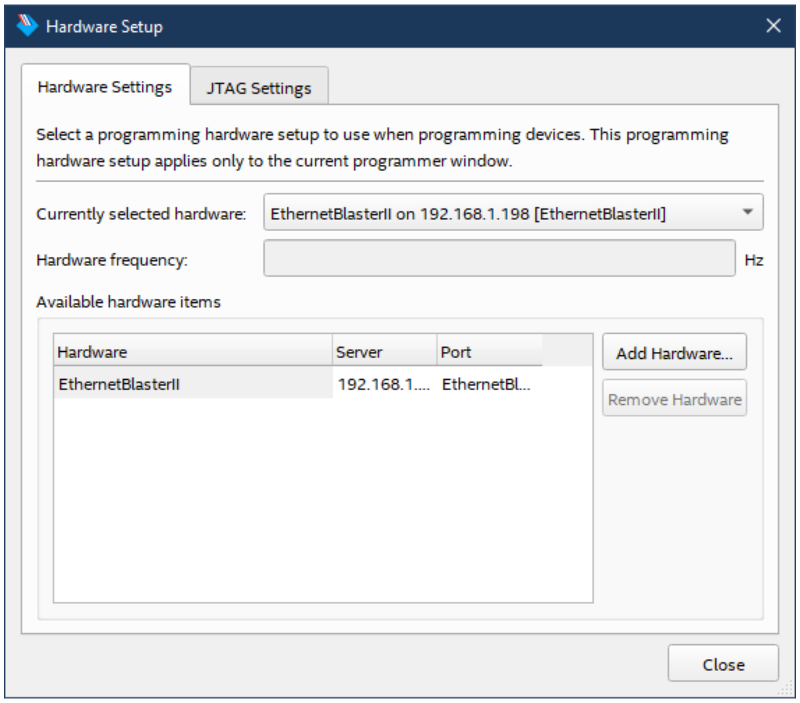
Baud rate:

Server name:

Server port: 1309

Server password:

8. The tool allows you to connect to a number of programming cables. We need to select the one for our board. In the "Currently selected hardware", click the drop-down, select the hardware cable for the board, and click Close when finished


Hardware Setup

Hardware Settings | JTAG Settings

Select a programming hardware setup to use when programming devices. This programming hardware setup applies only to the current programmer window.

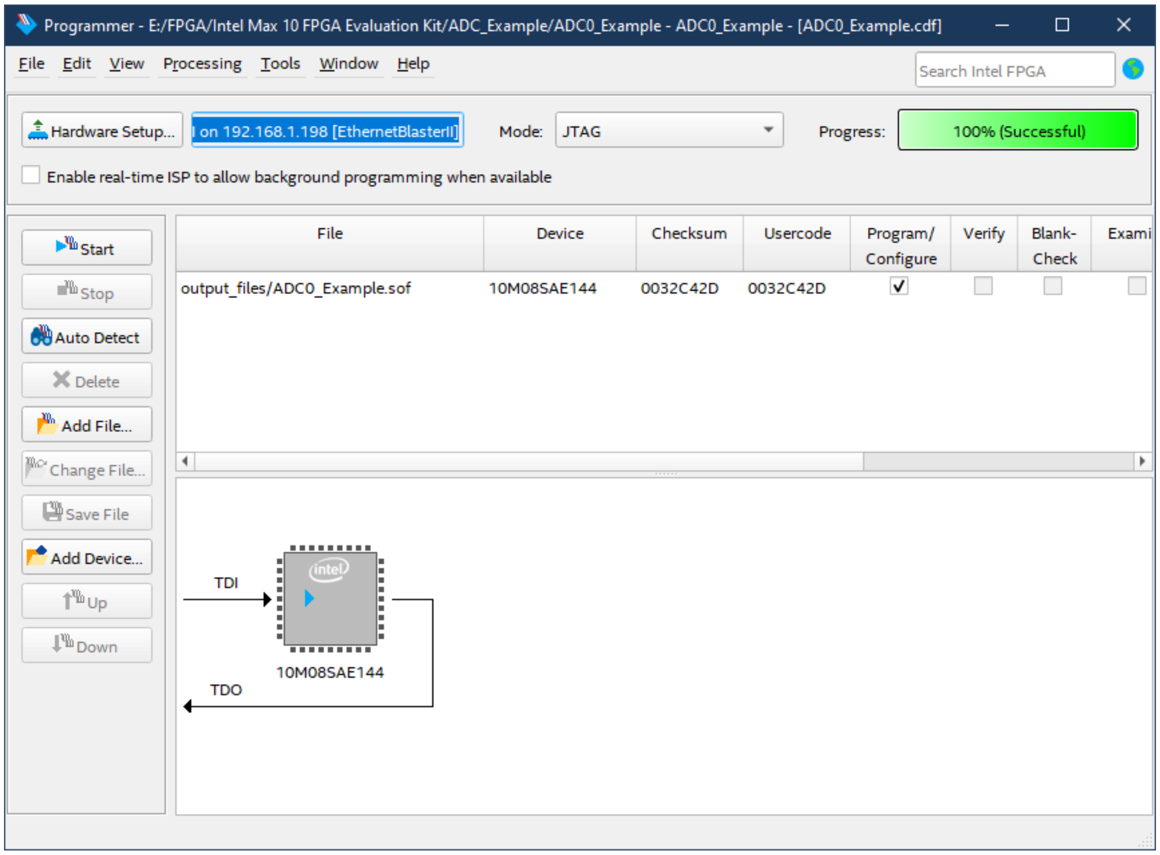
Currently selected hardware: EthernetBlasterII on 192.168.1.198 [EthernetBlasterII]

Hardware frequency: Hz

Available hardware items

Hardware	Server	Port
EthernetBlasterII	192.168.1....	EthernetBl...

9. An ADC0_Example.sof file gets created during the Compile Design flow. The file is automatically filled in. There is only one FPGA on the board and in the JTAG chain, so the file already has the Program/Configure checkbox checked. Click the Start button to program the board. The process takes a few seconds and shows that the task completed successfully.


Programmer - E:/FPGA/Intel Max 10 FPGA Evaluation Kit/ADC_Example/ADC0_Example - ADC0_Example - [ADC0_Example.cdf]
File Edit View Processing Tools Window Help
Search Intel FPGA
Hardware Setup...
Start
Stop
Auto Detect
Delete
Add File...
Change File...
Save File
Add Device...
Up
Down
TDI
TDO
10M08SAE144

l on 192.168.1.198 [EthernetBlasterII] Mode: JTAG Progress: 100% (Successful)

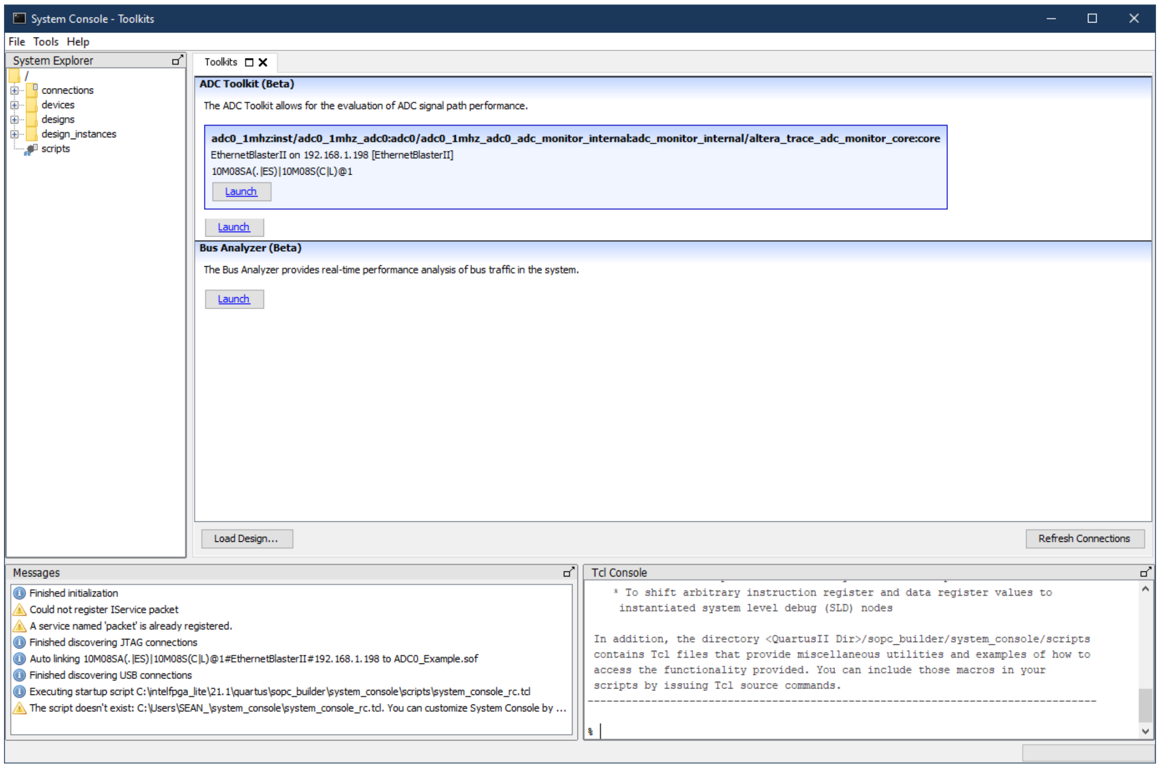
Enable real-time ISP to allow background programming when available

File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank-Check	Exami
output_files/ADC0_Example.sof	10M08SAE144	0032C42D	0032C42D	✓			

1.2.5 Test the Design with the ADC Toolkit

The board now has the system loaded in SRAM. The ADC Toolkit can view the output from the ADC.

1. Open Platform Builder
2. As Platform Builder launches, the pp[en file dialog opens asking to open the ADC0_1MHz.qsys file. Open the ADC0_1MHz.qsys file and the design will load.
3. From the menu, select Tools->System Console.



ADC Toolkit (Beta)

The ADC Toolkit allows for the evaluation of ADC signal path performance.

adc0_1mhz:inst/adc0_1mhz_adc0:adc0/adc0_1mhz_adc0_adc_monitor_internal:adc_monitor_internal/altera_trace_adc_monitor_core:core
EthernetBlasterII on 192.168.1.198 [EthernetBlasterII]
10M08SA(.|ES)|10M08S(C|L)@1

Bus Analyzer (Beta)

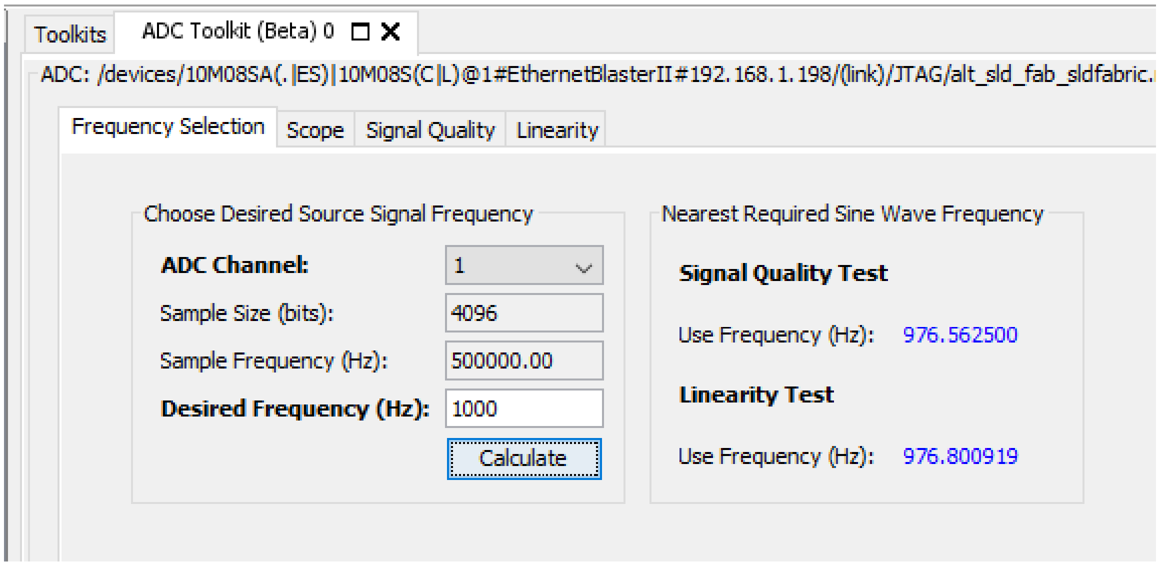
The Bus Analyzer provides real-time performance analysis of bus traffic in the system.

```
Finished initialization
Could not register IService packet
A service named 'packet' is already registered.
Finished discovering JTAG connections
Auto linking 10M08SA(.|ES)|10M08S(C|L)@1#EthernetBlasterII#192.168.1.198 to ADC0_Example.sof
Finished discovering USB connections
Executing startup script C:\intelfpga_lite\21.1\quartus\sopc_builder\system_console\scripts\system_console_rc.tcl
The script doesn't exist: C:\Users\SEAN_\system_console\system_console_rc.tcl. You can customize System Console by ...
```

```
   * To shift arbitrary instruction register and data register values to
    instantiated system level debug (SLD) nodes

 In addition, the directory <QuartusII Dir>/sopc_builder/system_console/scripts
 contains Tcl files that provide miscellaneous utilities and examples of how to
 access the functionality provided. You can include those macros in your
 scripts by issuing Tcl source commands.
--------------------------------------------------------------------------------
% 
```

4. The console will read the JTAG and enable the ADC Toolkit. Click the Launch button to launch the ADC Toolkit.
5. A new tab appears. Under Frequency Selection, click Calculate.



ADC: /devices/10M08SA(.|ES)|10M08S(C|L)@1#EthernetBlasterII#192.168.1.198/(link)/JTAG/alt_sld_fab_sldfabric.

Choose Desired Source Signal Frequency

ADC Channel:	1
Sample Size (bits):	4096
Sample Frequency (Hz):	500000.00
Desired Frequency (Hz):	1000

Nearest Required Sine Wave Frequency

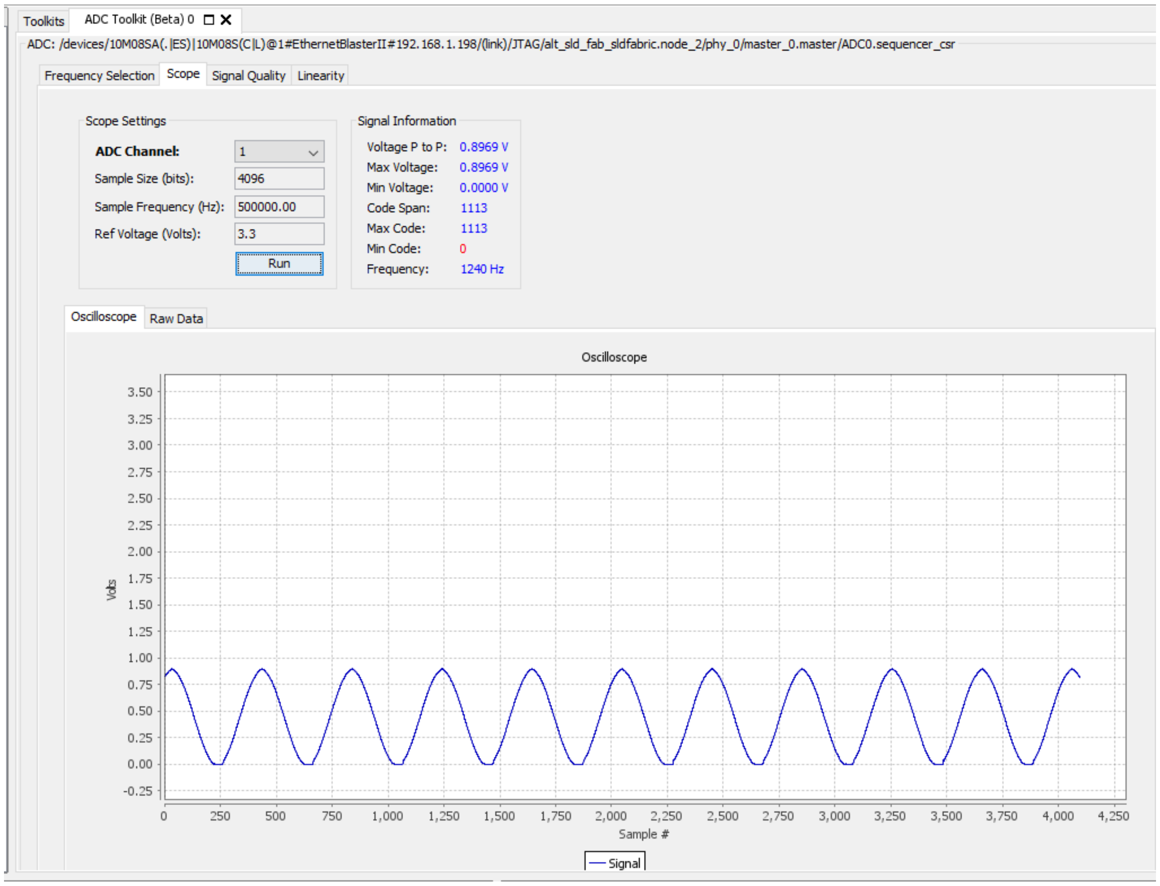
Signal Quality Test

Use Frequency (Hz): 976.562500

Linearity Test

Use Frequency (Hz): 976.800919

6. Click on the Scope tab.
7. ADC_Channel 1 should already be selected. The Scope tab looks like the screen of an Oscilloscope. Click the Run button, and you should see the sine wave appear.



ADC: /devices/10M08SA(.|ES)|10M08S(C|L)@1#EthernetBlasterII#192.168.1.198/(link)/JTAG/alt_sld_fab_sldfabric.node_2/phy_0/master_0.master/ADC0.sequencer_csr

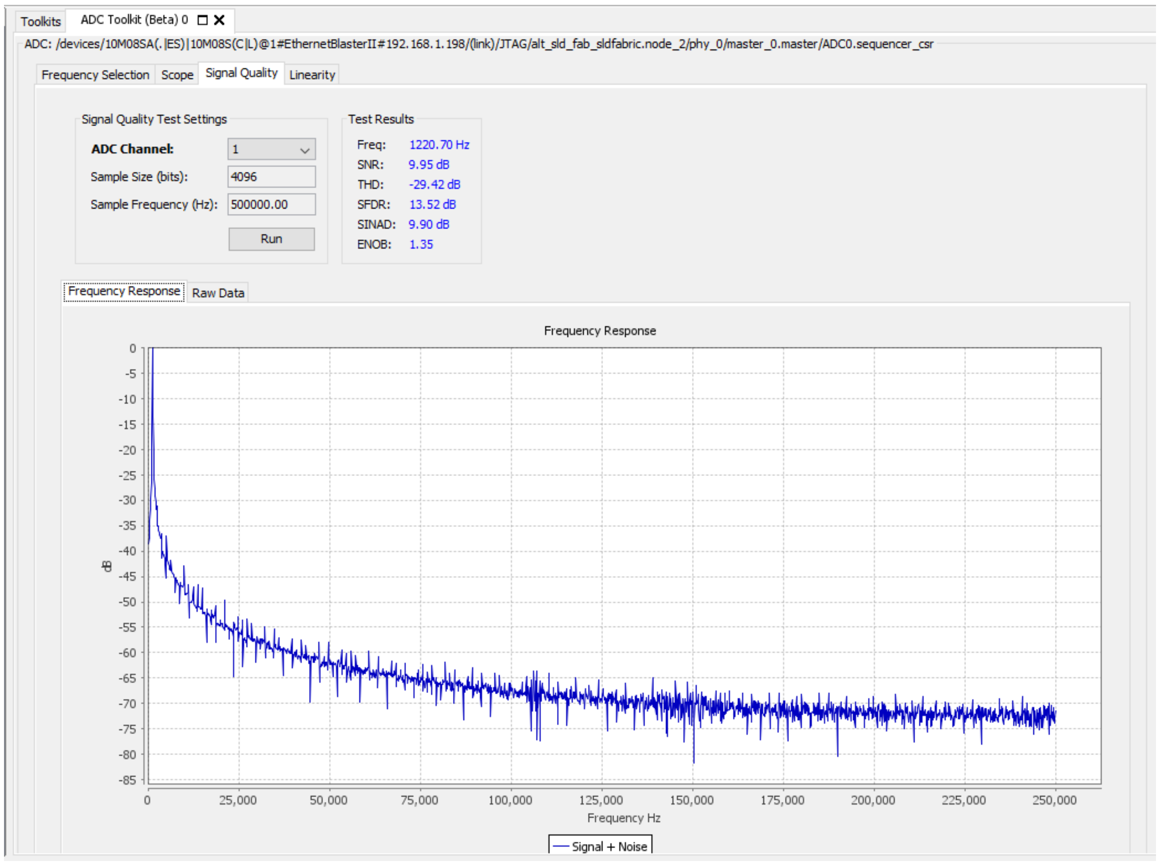
Scope Settings

ADC Channel: 1
Sample Size (bits): 4096
Sample Frequency (Hz): 500000.00
Ref Voltage (Volts): 3.3

Signal Information

Voltage P to P: 0.8969 V
Max Voltage: 0.8969 V
Min Voltage: 0.0000 V
Code Span: 1113
Max Code: 1113
Min Code: 0
Frequency: 1240 Hz

8. Click on the Raw Data tab to see the actual values coming out of the ADC.
9. Click on Stop to stop gathering data.
10. Click on the Signal Quality tab and then click the Run button. The command only runs for a few seconds and stops.



Toolkits | ADC Toolkit (Beta) 0

ADC: /devices/10M08SA(.|ES)|10M08S(C|L)@1#EthernetBlasterII#192.168.1.198/(link)/JTAG/alt_sld_fab_sldfabric.node_2/phy_0/master_0.master/ADC0.sequencer_csr

Frequency Selection | Scope | Signal Quality | Linearity

Signal Quality Test Settings

ADC Channel: 1
Sample Size (bits): 4096
Sample Frequency (Hz): 500000.00

Test Results

Freq: 1220.70 Hz
SNR: 9.95 dB
THD: -29.42 dB
SFDR: 13.52 dB
SINAD: 9.90 dB
ENOB: 1.35

11. Click on the Raw Data tab to view the actual values.

The Linearity tab has several scope views: Histogram, DNL, INL, and Raw Data.

12. Go back to Scope, and change the ADC_Channel to 7.
13. Click the Run button and adjust the 10KΩ trimmer pot in either direction to watch the voltage level change.

Note: TSD was left out of the original design, since there appears to be an issue enabling the TSD with the other two channels. This issue was not indicated in any of the documentation that we found. The old training examples stayed away from enabling the TSD. If you want to view the TSD, you can edit ADC0 in the Platform build. Deselect CH1 and CH7, and then enable TSD. Save and regenerate the system file, and click Finish to close Platform Builder. In Quartus, recompile the design and reprogram the board with the .sof file. Use the Raw Data in the ADC tool kit Scope tab to see the raw output files. Compare the HEX values to the table found in the Intel® MAX® 10 Analog to Digital Converter User Guide.

1.3 Summary: All the Little Steps

Platform Builder and the IP Blocks make it easy to create an FPGA design. The not-so-easy part was all of the little steps that are different from the older training material. Quartus even crashed in one iteration, and naming everything ADC in the first test crashed the compiler with name conflicts. Once the little issues and the TSD issue were resolved, designing and testing flows was very simple.

1.4 References

The following references were used for this article:

- Intel® MAX® 10 Analog to Digital Converter User Guide - https://www.intel.com/content/www/us/en/docs/programmable/683596/20-1/analog-to-digital-converter-overview.html
- Introduction to Analog to Digital Conversion in Intel® MAX® 10 Devices Parts 1 and 2 - Intel FPGA training site Intel® FPGA Technical Training
- Using the ADC Toolkit in Intel® MAX® 10 Devices - Intel FPGA training site Intel® FPGA Technical Training
- How to Create ADC Design in MAX 10 Device Using Qsys Tool - https://cdrdv2.intel.com/v1/dl/getContent/649255?explicitVersion=true / https://www.youtube.com/watch?v=0oO1RFa-4Xk
- Intel® MAX® 10-10M08 Evaluation Kit schematic file. Altera_10M08S_E144_eval_schematic_REV_1_0.pdf.