



Max10 ADC Data Capture Lab

For the MAX® 10 DECA FPGA Evaluation Kit

Version 15.0

6/07/2015

TABLE OF CONTENTS

LAB 5.	MAX10 ADC DATA CAPTURE	202
5.1	Getting Started	203
5.1	.1 Prepare the analog Signal Source	203
5.2	ADC Core overview	203
5.3	Create a Simple ADC system; no Nios required	205
5.3	.1 Open the Quartus Project	205
5.3	.2 Build the Qsys system	206
5.3	.3 Complete the Qsys system	213
5.3	.4 Verify your system	213
5.3	.5 Generate the Qsys System	214
5.3	.6 Compile the design in Quartus	216
5.3	.7 Connect the Hardware	216
5.3	.8 Test the Hardware	217
5.3	.9 Verify operation using the ADC Toolkit	219
5.3	.10 LED Volume Meter design	223
5.3	.11 Debug the design using Signal Tap II	223
5.4	Create a software acquisition system, using Nios	226
5.4	.1 Open the ADC Nios Lab project	227
5.4	.2 Modify the Qsys system	227
5.4	.3 Compile the project in Quartus	230
5.4	.4 Configure the FPGA	230
5.4	.5 Create the Nios software application	230
5.4	.6 Configure the target hardware	235
5.4	.7 Debug the application	236
5.4	.8 Other [possibly] helpful information	239

LAB 5. MAX10 ADC DATA CAPTURE

Overview: The purpose of this lab is to learn about the basic architecture and configuration options for the MAX 10 ADC.

In this lab you will implement the MAX 10 ADC hard IP in a design. In the first part of the lab, you will implement a simple hardware-only streaming design. This design samples an incoming analog waveform on the LINE_IN connector, and displays the amplitude of that signal using the LEDs on the DECA kit. You will also observe the data with the ADC toolkit and with the SignalTap Logic analyzer.

For the second part of the lab, you will replace the simple hardware driver with a Nios processor system. This Nios based system will perform the same function as the hardware-only lab, but does it with software running in the Nios processor. SignalTap is also used to observe the system operation.

Simple Hardware-Only Streaming Lab Block Diagram::



Nios Processor Lab Block Diagram::



202







5.1 Getting Started

Overview: The first objective is to ensure that you have all of the necessary hardware items and software installed so that the lab can be completed successfully.

Below is a list of items required to complete this lab:

- Arrow DECA Evaluation Kit
- USB cable
- 2.5mm Audio cable
- Analog Signal Generator (Smart phone with Waveform App will work)
- Lab files
- Quartus II 15.0 Design Software
- Personal computer or laptop running Windows 7 with at least an Intel i3 core (or equivalent), 4 GB of RAM, and 12 GB of free hard disk space
- A desire to learn

If you are missing one of these items, please your instructor know. Instructions for how to download Quartus can be found in the Appendix.

5.1.1 Prepare the analog Signal Source

This lab requires an analog signal source. A smart phone with a Waveform generator app will work, although you should not expect good results for the linearity measurement portion of the Lab. If you are using a smart phone for this lab, please prepare it now by downloading and installing an appropriate application. For best results, choose an app that can generate sinewaves, and permits manually entering the desired frequency. **Waveform Generator Lite** and **Frequency Sound Generator** are two possible options.





If your laptop has a headphone jack, you can use it to generate a sine wave from: http://onlinetonegenerator.com/

5.2 ADC Core overview

Altera supports four configuration variants of the ADC core:









• Configuration 1: Standard Sequencer with Avalon-MM Sample Storage

In this configuration variant, you can use the standard sequencer micro core with internal on-chip RAM for storing ADC samples.



• **Configuration 2**: Standard Sequencer with Avalon-MM Sample Storage and Threshold Violation Detection

In this configuration variant, you can use the standard sequencer micro core with internal on-chip RAM for storing ADC samples with the additional capability of detecting threshold violation.









• Configuration 3: Standard Sequencer with External Sample Storage

In this configuration variant, you can use the standard sequencer micro core and store the ADC samples in external storage.



• Configuration 4: ADC Control Core Only

In this configuration variant, the Altera Modular ADC generates only the ADC control core.



5.3 Create a Simple ADC system; no Nios required

For the first part of this lab, we will be building a **Configuration 3** variant of the ADC core. In this variant, the data streams out of the ADC, and the user application handles the storing of data samples. For this lab, we will simply reformat the data with a simple Verilog application, and display it on the LEDs in real time to make a digital volume meter, without buffering.

5.3.1 Open the Quartus Project

In this section you will use an existing Quartus project, and build a Qsys system that contains a Modular ADC. The top level Verilog design file along with pin assignments has been constructed for you.









- 5.3.1.1 Launch Quartus, version 15.0
- 5.3.1.2 Open the Quartus project for this lab: File → Open Project
- 5.3.1.3 Browse to <location of lab>/5_ADC_Lab/ADC_Streaming_Lab/ADC_Streaming_Lab.qpf and Click Open

5.3.2 Build the Qsys system

5.3.2.1 Launch Qsys: Click **Tools** \rightarrow **Qsys** (or use the Qsys tool bar icon 4)

When Qsys opens, you should have an almost empty system, with the exception of one **Clock Source** component. Next, you will add several more components to this system.

- 5.3.2.2 Save this Qsys file as adc_qsys.qsys
- 5.3.2.3 Add Clock Source Component

This component should already be present as a default in all new systems. If not, you can add it using the IP catalog by searching for Clock Source. The input frequency to the system is 50MHz

5.3.2.4 Add Avalon ALTPLL

Add an Avalon PLL. This PLL will receive the board clock at 50 MHZ, and generate a 10MHZ clock for the ADC, and a 50MHz clock for the user logic.

IP Catalog \rightarrow Basic Functions \rightarrow Clocks; Plls and Resets \rightarrow PLL \rightarrow Avalon ALTPLL

Set the input frequency to 50MHz (MegaWizard Plug-In Manager [Page 1 of 11])

WIDWN

206









Uncheck the box for Create an 'areset' input (MegaWizard Plug-In Manager [Page 2 of 11])









MegaWizard Plug-In Manager [page 2 of 11]	- P
Parameter 2 PLL 3 Output 4 E Settings Reconfiguration Clocks 4	DA
General/Modes Inputs/Lock Bandwidth/SS	Clock switchover
ALTPLL1429314183041669	Able to implement the requested PLL
inclk0 inclk0 frequency: 50.000 MHz C0 Operation Mode: Normal locked	Optional Inputs Create an 'pliena' input to selectively enable the PLL
Clk Ratio Ph (dg) DC (%) c0 1/1 0.00 50.00	Create an 'pfdena' input to selectively enable the phase/frequency detector
MAX 10	Lock Output Image: 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
	 Configurations with output clock(s) that use cascade counters are not supported
	Avalon Bus connectivity Use a separate clock input for Avalon bus connections
	<u></u> 2
	Cancel < Back Next > Finish

Set the output frequency of clk c0 to 10 MHz (MegaWizard Plug-In Manager [page 6 of 11])

208







Max10 DECA Workshop Manual













MegaWizard Plug-In Manager [page 7 of 11]			?
1 Parameter 2 PLL 3 Output 41	EDA		
dk c0 > dk c1 > dk c2 > dk c3 > c	ik c4 >		
ALTPLL1429360720578760	c1 - Core/External Output Cloc Able to implement the requested PLL	ck	
jncik0 c0	Use this clock Clock Tap Settings		
Operation Mode: Normal c1	Enter output dock frequency:	Requested Settings 50 MHz 👻	Actual Settings 50.000000
c) 1/5 0.00 50.00 c1 1/1 0.00 50.00	Enter output clock parameters: Clock multiplication factor	1	1
MAX 10	Clock division factor	1 × Copy	1
	Clock phase shift	0.00 🔹 deg 🔻	0.00
	Clock duty cycle (%)	50.00	50.00
		Description	Valı 🔺
	Note: The displayed internal settings of the	Primary clock VCO frequency (MHz)	6
	PLL is recommended for use by advanced users only		* *
		Per Clock Feasibility In	dicators
			al 61
	10	Cancel < <u>B</u> ack	Next > Finish

Set the output frequency of clk c1to 50 MHz (MegaWizard Plug-In Manager [page 7 of 11])









Click Finish, and then Finish again to complete the PLL configuration.

5.3.2.5 Add the Modular ADC Core from the IP Catalog.

Locate the ADC core under Processors and **Peripherals** → **Peripherals** → **Altera Modular ADC Core**, or simply type ADC in the search box. Click Add

- Set the Core Variant to Standard sequencer with external sample storage
- Set the IP for **2nd ADC**. The line input is connected to Channel 5 of ADC 2. Note all other inputs on the DECA board are fed to ADC 1
- On the Channels Tab, check the boxes to use **Channel 0**, **Channel 1**, **and Channel 5**. The Audio signal is connected to ADC 2, Channel 5. To demonstrate the sequencer, we are also enabling channel 0 and 1, but there will be no input signal on these channels.

Block Diagram Show signals	Ceneral Core Configuration Core Variant: Debug Path: PIP Generation Generate IP for which ADCs of this device Clocks ADC Input Clock: Reference Voltage Reference Voltage External Reference Voltage: Channels Sequencer CH0 CH1 CH2 CH3 CH4 CH5 C CH0 CH1 CH2 CH3 CH4 CH5 C Channel 0	Standard sequencer with external sample storage Enabled 10 Mhz External 2.5 V H6 CH7 CH8 No TSD	
Error: modular_adc_0: Sequencer Slot 1 is pointing to Channel white Warning: modular_adc_0: Error converting csd slot value 30 to strict Warning: modular_adc_0.adc monitor internal.core: Unknow	Use Channel 0 (Dedicated analog i definition of the selected device paragram output code. n dock rate; 10.0 ms startup delay will be com	III III art. Please re-configure Sequencer Slot 1. Iputed assuming a clock rate of 50MHz.	, ,

Version 15.0







211

- On the Sequencer Tab, select the number of slots to use as 4
- On the Sequencer tab, set the conversion sequence as follows:

Slot 1	CH 0
Slot 2	CH 5
Slot 3	CH 1
Slot 4	CH 5

• Verify your configuration matches the screenshot below, and click Finish

ock Diagram	IP Generation		
how signals	Generate IP for which ADCs of this d	levice': 2nd ADC 👻	
modular adc C	* Clocks		
	AUC Input Clock:	1) Mhz 👻	
ock elock avalan_streaming respons	 Reference Voltage 		
reset_sink reset	Reference Voltage Source:	External 👻	
dc_pli_plockclock	External Reference Voltage:	2.5 V	
dc pli ocked			
equencer csr	Channels Sequencer		
avalon			
altera_modular_ad	Conversion Sequence Length		
	Number of side used: 4		
	Conversion Sequence Channel	s	
	Slot 1:	Ť.	
	Slot 2 :		
	Clat 2 1		
	Slot 4 : CH 5 👻		
		m	

212





Max10 DECA Workshop Manual



5.3.2.6 Add the LED_Streaming_Driver.

This is a custom component that is be located in the DECA Labs folder of the IP Catalog. There are no configurable options for this component, so just click Finish

If you want to inspect the verilog code for this component, it is located in the IP directory. In summary, this component has three interfaces:

- An Avalon memory mapped master for writing commands to the ADC sequencer control and status register. After coming out of reset, the LED Driver performs a single write to the ADC sequencer to put the ADC in continuous acquisition run mode.
- An Avalon streaming sink to receive the data from the ADC "response" streaming source. The LED driver receives the data and channel information, so it can ignore other channels, if they are enabled. The Audio LINE_IN signal is on channel 5.
- An exported conduit that drives the board LEDs. The value output is based on the analog level received from the ADC

5.3.3 Complete the Qsys system

At this time, you should have four components in the system, with 7 Errors in the messages tab. Now we are going to connect the components together.

5.3.3.1 Connect the clock, data, and IRQ connections.

Using the screenshot below, connect the Qsys components. Pay careful attention to the clock connection to the ADC core. Note in the screen shot, the "Show clock domains in the system table" is enabled. This option displays each clock domain in a different color, and can be helpful in identifying unintentional clock domain crossings.

5.3.3.2 Name the Qsys components.

Right click on the component name (or use keyboard shortcut Ctrl-R). The names of the components must be entered exactly as shown if you want to use the SignalTap portion of this lab. The names should be:

- clk_in
- system_clk
- ADC
- LED_Streaming_Driver

5.3.3.3 Export the LED bus.

Double click in the export column to export the conduit out of the LED driver. This will connect to the LED pins at the top level. Name this conduit led_out

5.3.4 Verify your system

Version 15.0









213

When complete, your system should look like the screenshot below: You will still have 3 warnings about ports that have been left unconnected. You can safely ignore these warnings.



5.3.5 Generate the Qsys System

If your system matches, click Generate HDL to generate the system. Choose Verilog as the language for synthesis.

www

214





Max10 DECA Workshop Manual



pile the system in a Quartus II project.
thesis: Verilog 👻
e estimates for third-party EDA synthesis tools.
əsf)
generated HDL files for the simulator, and may include simulation-only features.
None 👻
lation
and a final features for the
oports mixed-language simulation.
C:/DECA/ADC_Streaming_Lab/adc_qsys

Set the output Directory path to <project Directory>/adc_qsys and click generate.

The Generation process window will appear, and the progress will be displayed; generation takes a few minutes for this system. When complete, click Close.









5.3.6 Compile the design in Quartus

5.3.6.1 Add the Quartus IP file (.qip) file to the project: **Project** -> Add/Remove Files to Project

Note: When you finished generating the Qsys system, you may have seen a dialog box that reminded you to add the Quartus IP file to the project. That is what we are doing now.

Click on the <u>un</u> button to browse to the <u>adc_gsys/synthesis/adc_gsys.gip</u> file. **Don't** forget to click Add after selecting the file. Click OK.

ile name: adc_qsys/s	ynthesis/adc_qsys.qip				Add
File Name	Туре	Library	Design Entry/Synthesis Tool		Add All
DECA.SDC Deca_tap.stp	Synopsys Design Constraints File stp SignalTap II Logic Analyzer File mp.rdb Other	<none> <none> <none></none></none></none>	<none> <none></none></none>		Remove
db/becA.dhp.rdb					Up
					Down
				Properties	
4				F	

5.3.6.2 Start the Compile: **Processing** \rightarrow **Start Compilation** (or click the compile icon \bowtie on the toolbar)

5.3.7 Connect the Hardware

5.3.7.1 Connect the Audio Cable

Connect the 2.5mm cable from the signal generator (smart phone) to the Line in jack of the DECA board. This is J2 on the DECA board (blue jack).

5.3.7.2 Connect the USB Cable

Plug the USB cable into the on-board USB blaster II. This is J10 on the board

216







Max10 DECA Workshop Manual





5.3.8 Test the Hardware

5.3.8.1 Open the Quartus Programmer: **Tools** -> **Programmer**, or click the programmer icon in the toolbar

Confirm the Programing hardware: Arrow MAX 10 DECA should appear in the programming hardware window. If it says "No Hardware", the programmer hardware will need to be configured. See the Appendix, **Chapter 9.3 Configure the Quartus the programmer** for detailed instructions.

🔔 Hardware Setup	Arrow MAX 10 DECA [USB-1]	
Enable real-time ISP t	o allow background programming when available	

5.3.8.2 Add the programming file: click Add file, and browse to the **ADC_Streaming_Lab.**sof file in the **output_files** directory.

5.3.8.3 Verify the Program / Configure box is checked.

If your screen matches the one below, **press the Start button**. Programming is very fast (~2 seconds). Confirm the Progress is 100% (Successful)











218





5.3.8.4 Save the programming configuration: **File** → **Save As**. If you name the configuration file with the same name as the project, it will automatically open when you open the programmer. Browse to the project directory, name the file: **ADC_Streaming_Lab.cdf**, and click Save to continue.

Save in	ADC_Stream	ming_Lab	▼ 🔄 🖆 💽 ▼		
C.	Name	~	Date modified	Туре	
Recent Places Desktop Libraries LE9794	.qsys_edit adc_qsys db increment ip output_file	al_db •s	4/17/2015 10:25 PM 4/18/2015 8:46 AM 4/22/2015 10:32 PM 4/17/2015 11:51 PM 4/20/2015 2:11 AM 4/18/2015 10:10 PM	File folder File folder File folder File folder File folder	
	•				
	File name:	ADC_Streaming_Lab.cdf		-	Save
	Save as type:	Chain Description Files (*	cdf)	-	Cancel

5.3.9 Verify operation using the ADC Toolkit

- 5.3.9.1 Open system console: in Quartus, Tools → System Debugging →System Console. Or alternatively, from Qsys: Tools →System Console
- 5.3.9.2 Launch the ADC tool kit by pressing the Launch button.



NOTE: if Launch does not become available by default you will need to click "Load Design" and point to the SOF file that is configured into the device. System Console will use SOF file data to match with JTAG debug target in the FPGA

ADC Toolkit	(Beta)
ADDERA	The ADC Toolkit allows for the evaluation of ADC signal path performance.
MAX 10	adc_qsys:u0/adc_qsys_adc:adc/adc_qsys_adc_adc_monitor_internal:adc_monitor_internal/altera_trace_adc_monitor_core:core Arrow MAX 10 DECA on localhost [USB-1] 10M50DA(. [ES]]10M50D(C F)@1









5.3.9.3 In the Frequency Select tab, select the ADC channel to evaluate. The Line-In jack is connected to channel 5, so select channel 5. Press Calculate. This will tell you the frequencies required for the Signal Quality Test and the Linearity Test.

requency Selection	Scope Signal (Quality	Linearity		
Choose Desire	d Source <mark>Signal</mark> I	=reque	ncy	Nearest Required Sine V	Wave Frequency
ADC Chann	ADC Channel:		•	Signal Quality Test	:
Sample Size ((bits):	4096	00.00	Use Frequency (Hz):	Enter Desired Freq
Desired Fre	equency (Hz):	1000	0.00	Linearity Test	
		0		Lise Frequency (Hz):	Enter Desired Fred

220







Max10 DECA Workshop Manual



- 5.3.9.4 Go to the **Scope** tab.
- 5.3.9.5 Confirm that ADC Channel 5 is selected. Click Run to begin acquiring data from the ADC
- 5.3.9.6 Start the Signal Generator (or Waveform Generator app on your smart phone). Select a waveform to output, and you should see the waveform appear on the Oscilloscope window.









- 5.3.9.7 Click Stop to move to the next section
- 5.3.9.8 Switch to the Signal Quality tab. Click Run to perform a Frequency Response Measurement: To zoom, click and drag a portion of the graph, i.e. grab the top left and drag to the region edge on the right to zoom. If you want to see the raw data that was measured for this plot, click on the Raw Data tab. Note for these measurements, it is critical that the frequency output from the signal source matches exactly the frequency calculated by the ADC toolkit. Since this is difficult without lab quality equipment, datasheet specifications should not be expected.



TEXAS INSTRUMENTS



5.3.9.9 Switch to the Linearity tab. If you have a sine wave running, you will see a bathtub curve, showing the number of samples collected at each ADC code. For an ideal sinewave, this will be a smooth curve. The smart phone is not an ideal source.



5.3.10 LED Volume Meter design

The design we built contains a streaming LED driver, which decodes the ADC value into an LED bar graph in real time. As you adjust the amplitude on your waveform generator, you may have noticed the LED bar graph changing in amplitude. Adjust the volume now to confirm the functionality. In the design, we have two LEDs on as the "off" condition, so even with no sound, you should see two LEDs.

5.3.10.1 If you are using a smart phone as your signal source, switch to t a music playing app, and you will see the LEDs respond with a simple "VU meter" display.

5.3.11 Debug the design using Signal Tap II

Max 10 is the first CPLD from Altera to support SignalTap, Altera's embedded logic Analyzer. As part of the lab you opened, a completed SignalTap system was included. In this section we will explore some of the basic SignalTap capabilities.

Version 15.0









223

5.3.11.1 Open SignalTap: Tools → SignalTap II Logic Analyzer

If you have already programmed the device, the system status should be "Ready to Acquire". If not, program the device using the Quartus programmer, or use the programmer built into SignalTap.



NOTE: In some cases, Quartus II Web-Edition will throw an error when launching SignalTap. Altera requires that you enable the TalkBack feature in order to use the SignalTap II Logic Analyzer.

If so, follow these steps:

Go to the menu: **Tools** \rightarrow **Options** and select the Category: Internet Connectivity. In the right-hand pane, click the TalkBack Options... button then check the **Enable sending TalkBack data to Altera** checkbox.

Click OK twice to close both the Quartus II TalkBack dialog box as well as the Options dialog box.

5.3.11.2 Start the acquisition: **Processing → Run Analysis**, or by clicking on the Run Analysis icon in the toolbar.

The SignalTap system included with the lab has been configured to trigger on a valid Channel 5 data response from the ADC. If your ADC is running correctly, the SignalTap system will trigger instantly, and display the captured data. The result should look something like this.

Edil Vew Proje	Lt Processing Tools <u>M</u>	indow <u>H</u> elp	7									Se	sarch alter	a.com	
N 19 19 19	🚓 🕨 🖏 🔞														
tance Manager: 📉	🔊 🔳 🛃 Ready to a	cquire		×	JTAG Chair	n Configur	ation: JTA	G ready							
ance	Status	Enabled	LEs: 1126	Memory	Undunes	Arrow M	AX 10 DECA	DISB 1							
🚯 auto_signaltap_0	Not running	V	1126 cells	802810	- Charaman er										astin
					Device:	@1:10	450DA(, [ES]	/10M5CD((C F) (0x0310	SCDD)				Scan	Chai
					>> SO=	Manager:	1	sture/Ma	×10/DECA/AD	C_Streamin	g_Lab/outp	ut_files/A	DC_Stream	ning_Lab.	sof
	πı			Þ											
og: Trig @ 2015/04/19	10:21:28 (0 0:0.2 elapsed)								dick to inser	t tima bar					
ype Alias		Name			-128	- 37 - I	0	128	256	384	. 5	12	640	- 20	/68
► KEY[1	.0]					_			3h						_
adc_qsys	u0 adc_qsys_ADC:adc respo	nse_valid	3/2								1			┶┶┥	_
adc_qsys	u0 adc_csys_ADC:adclrespo	nse_startofpac	ket					2						-	-
adc_qsys	adc_qsys:ub[adc_qsys_AUC:adc]response_endorpscket					-							I and I a		
> ⊡…adc_q					00h 00h	00h	00h 00h	00h	00h 00h	00h	00h 00h	00h	001 00	Jh 00	h
Sector and a s	sys utiade gsys ADC:addre	sponse data[1]	01		000h 000h	00010001	000h000h	000h]COC	h 000h 000h	000h[000h]	000h[000h	0001000	h]COCh]00	0h 000h	000
adc_qsys	uuladc_dsys_AUC.addisedue	encer_csr_read	5 X				1								
> adc_qsys	uojado_qsys_ADC.addiseque	ancer_csr_write													
adc_qsys	utilate_dsys_Abc.addsedue	streaming drive	adverset o												_
adc_qsys	util ED_Streaming_Driver:led	streamno driv	erlavalon waitreoi	uest			1								
adc gsvs	UNILED Streaming Driver:led	streamno driv	erlavalon write												_
adc gaya	uOILED Streaming Driver:led	streamna driv	criavalon read	1			1		_						
adc_g	sys u(ILED_Streaming_Driver	led_streaming	driverireg_linein_d	ata[110]	92Ch		1	92Bh	0		92	Ch	92Bh	92	Ch
y ⊡ adc_q	sys utilED_Streaming_Driver	led_streaming_	driver[vol[70]		03h							_			
						III									,
Data Setu						(and a second									
			-												_
erarchy Display:	×	Data Log:													
V 🕈 ADC_Streamin	lg_Lab	auto_sig	naltap_0												
adu_qsys	u0														
IVI ≫ adc I	ISVS ADC:adc														
M ~ [[D]	ou canning_Universica_s														
auto signaltar 0															

Max10 DECA Workshop Manual









5.3.11.3 Observe the ADC sequence: response_startofpacket is asserted to indicate the start of a sequence, and the response_channel bus provides the channel number. Insert a time bar by clicking above the timeline to see the data values on the bus without having to zoom in.

1210	115/04/19 10:36:37 (0:0:1.7 clapsed)	-							Tim	e Bar			200
	Name	200	Value	201	Q	32	6	1	96	128	160	19	2 224
ade	ade_qays:u0jade_qays_ADC:adejresponse_atemofpacket		0										A
ade	adc_qsys:u0 adc_qsys_ADC:adc response_endofpacket		1				V						
E	= adc_gsys:u0[adc_gsys_ADC:adc]response_channe[40]	\square	05h	$ \rightarrow $	00h		0Ch			OOh	L	00h	001
	adc_qsys:u0 adc_qsys_ADC:adc response_data[110]	(900h	>	000h	000h		000h		000h	R	000h	00Ch
adc	adc_qeye:u0]adc_qeye_ADC:adc eequencer_csr_read		0										
adr.	adc_qsystu0jadc_qsys_ALIC:adc sequencer_csr_wrte		0					1		2		1	-
adc	ado gsys.u0jado gsys ADC.ado/sequencer osr address		0					CHU		CHO		CH 1	CHO
adc	adc_qsys:u0 _ED_Streamng_Driver:icd_streaming_driver reset_n		1										
ade	adc_qsys:u0 _ED_Streaming_Driver:led_streaming_driver avalon_waitrequest		11										3
adc,	adc_qsys:u0j_ED_Streaming_Driver:ied_streaming_driverjavalon_write		0										
adc	adc_qsys:u0j_ED_Streaming_Driver:ied_streaming_driver avalon_read		0										
.	⊡··adc_qsys:uQLED_Streaming_Driver:led_streaming_driver(reg_linein_data[110]		8FFH	$ \rightarrow $	900h		901h				8FFh		90Ch
F··	Finado_qsystut(LEL)_Streaming_Driverled_streaming_driver(vol(7-0)		01h	_			I3h		1		dr.F		Ceh
-	Pilado_qsysiuQEHD_Streaming_DriverTed_streaming_driveryon(7-0)		01h		4		1	03h	03h	03h	03h	03h 04h	03h 04h

5.3.11.4 Change the trigger condition. Most changes to SignalTap require a full recompile of the system. Some changes, like the trigger condition, can be made without recompiling. Let's change the trigger to look for a specific data value. Go to the Setup Tab, and enter E00 as the trigger condition for the reg_linein_data.

If you want to ensure you don't accidentally make a change that requires a full recompile, set the Lock mode to "Allow trigger condition changes only".

		Node	Data Enable	Trigger Enable	Trigger Condition
Туре	Alias	Name	49	49	1 V Basic AND
*		adc_qsys:u0 adc_q.Node Name_response_startofpacket	V		
*		adc_qsys:u0 adc_qsys_ADC:adc response_endofpacket		V	
5			V		05h
5			V	V	XXXh
*		adc_qsys:u0 adc_qsys_ADC:adc sequencer_csr_read	V	V	
*		adc_qsys:u0 adc_qsys_ADC:adc sequencer_csr_write	V	V	
*		adc_qsys:u0 adc_qsys_ADC:adc sequencer_csr_address	V		
*		adc_qsys:u0 LED_Streaming_Driver:led_streaming_driver reset_n	V	V	
*	, î	adc_qsys:u0 LED_Streaming_Driver:led_streaming_driver avalon_waitrequest	V		
*		adc_qsys:u0 LED_Streaming_Driver:led_streaming_driver avalon_write	V	V	
*.		adc_qsys:u0 LED_Streaming_Driver:led_streaming_driver avalon_read	V	V	
R			V	V	E00h
R			V		XXh

5.3.11.5 Run Analysis again 🕅



NOTE: if the Run Analysis option is greyed out, confirm that you have an instance selected in the Instance manager window.







This time, the system may not trigger immediately, depending on the input level from your signal source. Turn the volume up, and the system should trigger. If not, try a lower trigger value, like 900h

That completes this portion of the Lab.

5.4 Create a software acquisition system, using Nios

In this section of the lab, you will use a Nios processor to read data from the modular ADC, and send that data to a Parallel IO peripheral that drives the LEDs. The overall function of this system is similar to the streaming system, but uses a Nios instead of dedicated hardware.

To convert the system, you will remove the Streaming LED controller and replace it with a complete Nios processor Subsystem. Since Nios will require a memory mapped interface vs. a Streaming interface, you will also reconfigure the Modular ADC core to output the data in a memory mapped format.



A second set of files have been provided as a starting point for this section. You could start with the files created during the streaming lab, but the SignalTap signals are different, so it will be much easier to start with the provided lab files.

As a reminder, here is what the system looked like in the first section of this lab:

lise	Connections	Name	Description	Export	Clock	Base	Fnd
V		🛢 clk_in	Clock Source			1	
		- dk_in - dk_in_reset - dk	Clock Input Reset Input Clock Output	clk reset	exported		
		dk_reset	Reset Output	Double-click to export	1.K_11		
		system_clk inck_nterface inck_nterface inck interface reset pl_alave c0 c1 areset_conduit	Avalon ALTPLL Cluck Input Reset Input Avalon Memory Mapped Slave Clock Output Clock Output Corduit	Double-click to export Double-click to export Double-click to export Double-click to export Double-click to export Double-click to export	clk_in [ndk_interf [ndk_interf systen_dk_c0 systen_dk_c1	ŵ	
		phasedone_condut FI 1 ADC dock reset sink adc_pll_clock adc_nll locked	Conduit Altera Modular ADC core Clock Input Clock Input Clock Input Clock Input	Double-click to export Double-click to export Double-click to export Double-click to export Double-click to export	system_cl [dock] system_cl		
100		cequencer_csr	Avalon Memory Mapped Slave Avalon Streaming Source	Double-click to export Double-click to export	[clock] [clock]	# 0x0000	02:0007
×		avalon_streaming_onv_ avalon_streaming_sink avalon_master dock reset	Avalon Streaming Lift/er Avalon Streaming Sink Avalon Nemory Mapped Master Clock Input Reset Input	Bouble-click to export Double-click to export Double-click to export Double-click to export	[clock] [rlock] system_tl [clock]	}-	Remove this component, and replace with a NIO SubSystem









5.4.1 Open the ADC Nios Lab project

- 5.4.1.1 Close the Streaming Lab Project (if necessary), closing all open Quartus windows, including Qsys, SignalTap, programmer, etc.
- 5.4.1.2 Open the ADC Nios Lab Project: **File → Open Project** and browse to <location of lab>\workshop_labs\5_ADC_Lab\ADC_Nios_Lab\ADC_Nios_Lab.qpf. Click Open

5.4.2 Modify the Qsys system

- 5.4.2.1 Launch Qsys: Click **Tools** \rightarrow **Qsys** (or use the Qsys tool bar icon 4)
- 5.4.2.2 Open the Qsys system <project directory>\adc_qsys.qsys
- 5.4.2.3 Delete the LED_Streaming_Driver. Click on the module and press Delete (Or **alternatively**, right click, and select Delete)
- 5.4.2.4 Add a Parallel Input/output component: **Processor and Peripherals** → **Peripherals** (or search for PIO in the IP catalog). This component will connect to the LEDs, so they can be written to from Nios

The default settings are correct. Click Finish

🚣 PIO (Parallel I/O) - pio_0			×
Megorerer PIO (Parallel I/O) altera_avalon_pio		Documentati	on
altera_avalon_pio Block Diagram Show signals clk clk clk clk external_connection conduit attera_avalon_pio	Basic Settings Width (1-32 bits): Direction: Output Port Reset Value: Output Port Reset Value: Cutput Register Enable individual bit s Cutput Register Synchronously capture Edge Type: Enable bit-clearing fo Interrupt Generate IRQ IRQ Type: Level: Interrupt CPU wh Edge Synchrony CPU wh Edge Synchro	Bidir Bidir Input Inout Output Outpu	
< >	register is logic true. Avai Test bench wiring Hardwire PIO inputs i Drive inputs to:	lable when synchronous capture is enabled in test bench 0x00000000000000000000000000000000000]









5.4.2.5 Add the Nios Subsystem: This is a custom system created for you, for this lab. It should be located in the DECA_Labs folder. There are no configuration options, so just click Finish to add it to the system. If you want to inspect this subsystem, right click on the module after it has been added, and select Drill into subsystem. You can also use the System navigation buttons to move up and down the hierarchy of a Qsys system. Return to the top level of the hierarchy when complete.



- 5.4.2.6 Rename the Nios subsystem to NiosSubsystem
- 5.4.2.7 Reconfigure the ADC Core. Double click on the Altera ADC Modular Core to open the Parameters Tab. Change the Core Variant to **Standard sequencer with Avalon-MM sample storage**.

Changing to this variant has several effects. One of which is that the streaming interface is no longer available. Instead, the streaming interface is connected internally inside the ADC core to an on-chip RAM. This RAM is accessible through an Avalon memory mapped interface that now appears on the ADC core in Qsys.

System Contents 🛛 Address Map	🛛 🔀 🍇 Parameters 🛛	Interconnect Requirements 🛛	- 🗗 🗖
System: adc_qsys Path: ADC		••	
Altera Modular ADC core altera_modular_adc	•		Details
General			^ ^
 Core Configuration 			
Core Variant:	Standard sequencer w	ith Avalon-MM sample storage	-
Debug Path:	Enabled 👻		
• IP Generation			
Generate IP for which ADCs of this	device?: 2nd ADC 👻		
* Clocks			
ADC Input Clock:	10 Mhz 👻		E
* Reference Voltage			
Reference Voltage Source:	External 👻		
External Reference Voltage:	2.5 V		
Channels Sequencer			
CH0 CH1 CH2 CH3 CH4 CH5	CH6 CH7 CH8 No TSD		
Thannel 0			
Vise Channel 0 (Dedicated ana	log input pin - ANAIN)		

228

᠕ᡢᡢ᠕







- 5.4.2.8 Connect the Qsys components. Connect the clock, reset, and memory mapped interfaces as shown in the screenshot below:
- 5.4.2.9 Fix the Address conflict. There will be a memory conflict in your system (the sample_store_csr, the sequencer_csr, and the LED_pio are all at address 0x000). Fix this manually by setting the sequencer_csr Base Address to 0x0200, and the LED_pio to 0x0210. Alternatively, you can let Qsys fix it automatically: System→Assign Base Addresses.

Use	Connections	Name	Description	Export	Clock	Base	En
		⊟ clk in	Clock Source			I.	
	<u> </u>	dk_in	Clock Input	clk	exported	[
	D		Reset Input	reset	1000000		
		dk	Clock Output	Double-click to export	dk_in		
		dk_reset	Reset Output	Double-click to export			
V		☐ system_clk	Avalon ALTPLL				
	• • • • • • • • • • • • • • • • • • •	inclk_interface	Clock Input	Double-click to export	clk_in		
	$ \downarrow $	inclk_interface_reset	Reset Input	Double-click to export	[inclk_interface]		
		pll_slave	Avalon Memory Mapped Slave	Double-click to export	[inclk_interface]	1	
		c0	Clock Output	Double-click to export	system_clk_c0		
		c1	Clock Output	Double-click to export	system_clk_c1		
	Q	areset_conduit	Conduit	Double-click to export	0.000		
	•	locked_conduit	Conduit	Double-click to export			
	¢—	phasedone_conduit	Conduit	Double-click to export			
V		모 변 ADC	Altera Modular ADC core				
		dock	Clock Input	Double-click to export	system_clk_c1		
	$ \downarrow \downarrow \rightarrow$	reset_sink	Reset Input	Double-click to export	[clock]		
		adc_pll_clock	Clock Input	Double-click to export	system_clk_c0		
		adc_pll_locked	Conduit	Double-click to export	1000000000		
	│	sequencer_csr	Avalon Memory Mapped Slave	Double-click to export	[clock]	0x0200	0x
		sample_store_csr	Avalon Memory Mapped Slave	Double-click to export	[clock]	0x0000	0.8
		sample_store_irq	Interrupt Sender	Double-click to export	[clock]		
-		🗆 🛄 NiosSubsystem	NiosSubsystem				
	♦ ○ ○	clk	Clock Input	Double-click to export	clk_in		
	$ \downarrow \rightarrow$	nios_irq	Interrupt Receiver	Double-click to export	[dk]	IRQ (
		nios_mm_master	Avalon Memory Mapped Master	Double-click to export	[clk]		T
	│ <mark>↓</mark> →	reset	Reset Input	Double-click to export			
		LED_pio	PIO (Parallel I/O)				
	• • • • • •	clk	Clock Input	Double-click to export	clk_in		
	• • •	reset	Reset Input	Double-click to export	[dk]		
	│	s1	Avalon Memory Mapped Slave	Double-click to export	[dk]	≝ 0x0210	0x
	0-0	external_connection	Conduit	led_out			
				1			







- 5.4.2.10 Generate the system. You should have 3 warnings; these can be ignored. Click Generate HDL to open the Generate Dialog box, and click Generate to begin the generation process.
- 5.4.2.11 If there are no errors, click Close to close the Qsys Generation status dialog box.
- 5.4.2.12 Close Qsys and return to Quartus. If you get a message reminding you to add the qip, you can ignore this message, as the qip was added for you in this project.

5.4.3 Compile the project in Quartus

5.4.3.1 Start the Compile: **Processing** \rightarrow **Start Compilation** (or click the compile icon \checkmark on the toolbar) Confirm the compile completed successfully.

5.4.4 Configure the FPGA

- 5.4.4.1 Open the Quartus II Programmer via **Tools** → **Programmer**
- 5.4.4.2 The Programmer window will open with a predefined Chain-Description File (ADC_Nios_Lab.cdf) that specifies the programming file output_files/ADC_Nios_Lab.sof
- 5.4.4.3 Click Start to begin configure the FPGA.

5.4.5 Create the Nios software application

- 5.4.5.1 Open Eclipse: From Quartus, Tools → Nios II Software Build Tools for Eclipse
- 5.4.5.2 Select a workspace: It is recommended that you create a workspace named eclipse_workspace in the same directory as the Quartus project files. Browse to the project folder and click Make New Folder. Rename the folder to eclipse_workspace. Click OK to continue

230





Max10 DECA Workshop Manual



A 📕 DEC	CA	^
4 💼 W	2 Onus Intro Lab	
N 1	2_Qsys_Intro_Lab	
N III	A Gesture Sensor Lab	E
	5 ADC Lab	
	ADC Nios Lab	
242 4	asys edit	
	db	
Þ	eclipse_workspace	
	ip ip	
	Prewritten_source	-

5.4.5.3 Create the project: File → New → Nios II Application and BSP from Template

<u>File</u> <u>E</u> dit <u>S</u> ource	Refac <u>t</u> or <u>N</u> avigate Se <u>a</u> rch	<u>P</u> roject Ni <u>o</u> s II <u>R</u> un <u>W</u> indow <u>H</u> elp
<u>N</u> ew	Shift+Alt+N 🕨	📴 Nios II Application and BSP from Template
Open File <u>.</u>		🕅 Nios II Application
<u>C</u> lose C <u>l</u> ose All	Ctrl+W Shift+Ctrl+W	 ➢ Nios II Board Support Package ➢ Nios II Library [™] P<u>r</u>oject
Save As	Ctri+5	[[•]] <u>O</u> ther Ctrl+N







5.4.5.4 Configure the new project. When the Template dialog appears, enter these values:

- Browse to <project directory>\ADC_Nios_Lab\adc_qsys.sopcinfo
- Name the project adc_demo
- Select Hello World Small as the Project Template
- Click Finish

arget hardware information	
SOPC Information File name:	\ADC_Nios_Lab\adc_qsys.sopcinfo
CPU name:	NiosSubsystem_nios2_gen2
pplication project	
Project name: adc demo	
Lasterna	
Use default location	
Vse default location	TA\Altera\Literature\Max10\DECA\ADC_Nios_Lab\software\adc_dem
Use default location Project location: C:\DA	TA\Altera\Literature\Max10\DECA\ADC_Nios_Lab\software\adc_dem
Use default location Project location: C:\DA Project template	TA\Altera\Literature\Max10\DECA\ADC_Nios_Lab\software\adc_dem
Use default location Project location: C:\DA Project template Templates	TA\Altera\Literature\Max10\DECA\ADC_Nios_Lab\software\adc_dem
Use default location Project location: C:\DA Project template Templates Blank Project	TA\Altera\Literature\Max10\DECA\ADC_Nios_Lab\software\adc_dem
 Use default location Project location: C:\DA[*] Project template Templates Blank Project Board Diagnostics 	TA\Altera\Literature\Max10\DECA\ADC_Nios_Lab\software\adc_dem Template description Hello World Small prints 'Hello from Nios II' to STDOUT. The project occupies the smallest memory footprint
Use default location Project location: C:\DA Project template Templates Blank Project Board Diagnostics Count Binary Else D Servet ine Etc.	TA\Altera\Literature\Max10\DECA\ADC_Nios_Lab\software\adc_dem Template description Hello World Small prints 'Hello from Nios II' to STDOUT. The project occupies the smallest memory footprint possible for a hello world application.
 ✓ Use default location Project location: C:\DA Project template Templates Blank Project Board Diagnostics Count Binary Float2 Functionality Eloat2 GCC 	TA\Altera\Literature\Max10\DECA\ADC_Nios_Lab\software\adc_dem
 ✓ Use default location Project location: C:\DA[*] Project template Templates Blank Project Board Diagnostics Count Binary Float2 Functionality Float2 GCC Float2 Performance 	TA\Altera\Literature\Max10\DECA\ADC_Nios_Lab\software\adc_dem Template description Hello World Small prints 'Hello from Nios II' to STDOUT. The project occupies the smallest memory footprint possible for a hello world application. Image: Template description Image
 ✓ Use default location Project location: C:\DA Project template Templates Blank Project Board Diagnostics Count Binary Float2 Functionality Float2 GCC Float2 Performance Hello Freestanding 	TA\Altera\Literature\Max10\DECA\ADC_Nios_Lab\software\adc_dem Template description Hello World Small prints 'Hello from Nios II' to STDOUT. The project occupies the smallest memory footprint possible for a hello world application. Image: This example runs with or without the MicroC/OS-II RTOS and requires an STDOUT device in your system's hardware. Image: This example runs with or without the MicroC/OS-II RTOS
 ✓ Use default location Project location: C:\DA Project template Templates Blank Project Board Diagnostics Count Binary Float2 Functionality Float2 GCC Float2 Performance Hello Freestanding Hello MicroC/OS-II 	TA\Altera\Literature\Max10\DECA\ADC_Nios_Lab\software\adc_dem Template description Hello World Small prints 'Hello from Nios II' to STDOUT. The project occupies the smallest memory footprint possible for a hello world application. This example runs with or without the MicroC/OS-II RTOS and requires an STDOUT device in your system's hardware. For details, click Finish to create the project and refer to the
 ✓ Use default location Project location: C:\DA Project template Templates Blank Project Board Diagnostics Count Binary Float2 Functionality Float2 GCC Float2 Performance Hello Freestanding Hello MicroC/OS-II Hello World 	TA\Altera\Literature\Max10\DECA\ADC_Nios_Lab\software\adc_dem Template description Hello World Small prints 'Hello from Nios II' to STDOUT. The project occupies the smallest memory footprint possible for a hello world application. This example runs with or without the MicroC/OS-II RTOS and requires an STDOUT device in your system's hardware. For details, click Finish to create the project and refer to the readme.txt file in the project directory.
 ✓ Use default location Project location: C:\DA Project template Templates Blank Project Board Diagnostics Count Binary Float2 Functionality Float2 GCC Float2 Performance Hello Freestanding Hello MicroC/OS-II Hello World Hello World Small Memory Test 	TA\Altera\Literature\Max10\DECA\ADC_Nios_Lab\software\adc_dem Template description Hello World Small prints 'Hello from Nios II' to STDOUT. The project occupies the smallest memory footprint possible for a hello world application. This example runs with or without the MicroC/OS-II RTOS and requires an STDOUT device in your system's hardware. For details, click Finish to create the project and refer to the readme.txt file in the project directory. The BSP for this template is based on the Altera HAL
 ✓ Use default location Project location: C:\DA Project template Templates Blank Project Board Diagnostics Count Binary Float2 Functionality Float2 Functionality Float2 GCC Float2 Performance Hello Freestanding Hello MicroC/OS-II Hello World Hello World Small Memory Test Memory Test Small 	TA\Altera\Literature\Max10\DECA\ADC_Nios_Lab\software\adc_dem Template description Hello World Small prints 'Hello from Nios II' to STDOUT. The project occupies the smallest memory footprint possible for a hello world application. This example runs with or without the MicroC/OS-II RTOS and requires an STDOUT device in your system's hardware. For details, click Finish to create the project and refer to the readme.txt file in the project directory. The BSP for this template is based on the Altera HAL operating system with reduced code footprint.

WDW

232







5.4.5.5 Import the pre-written C-Code for this lab: File → Import → General → File System. Click Next

> For the "From directory", Browse to <project directory>ADC_Nios_Lab\Prewritten_source and check on main.c file

Browse to adc_demo for the "Into Folder".

Click Finish.

File system Import resources from the local file system. From directory: I\DECA\ADC_Nios_Lab\Prewritten_source Prewritten_source Filter Types Select All Deselect All Into folder: adc_demo Options Overwrite existing resources without warning Create top-level folder	Browse
From directory: I\DECA\ADC_Nios_Lab\Prewritten_source Prewritten_source Filter Types Select All Deselect All Into folder: adc_demo Options Options Overwrite existing resources without warning Create top-level folder	Browse
Prewritten_source Filter Types Select All Deselect All Into folder: adc_demo Options Overwrite existing resources without warning Create top-level folder	
Filter Types Select All Deselect All Into folder: adc_demo Options Overwrite existing resources without warning Create top-level folder	
Into folder: adc_demo Options Overwrite existing resources without warning Create top-level folder	
Options Overwrite existing resources without warning Create top-level folder	Browse
Create top-level folder	
Advanced >>	
(?) < Back Next > Finish	

Alternatively, to import a file into the workspace, you can simply drag it in from Windows Explorer.







Delete the hello_world_small.c from Project 5.4.5.6



234





Max10 DECA Workshop Manual



5.4.5.7 Build the project : **Project** \rightarrow **Build All** (or click the Build All icon $\boxed{100}$ on the toolbar

5.4.6 Configure the target hardware

- 5.4.6.1 Open the debug configuration dialog: **Run → Debug Configurations**
- 5.4.6.2 Double Click on Nios II Hardware to create a new Nios II launch configuration
- 5.4.6.3 Name this configuration: DECA_hardware
- 5.4.6.4 Select adc_demo as the Project name. The ELF should automatically populate if the compile was successful.
- 5.4.6.5 The result should look like this:

Create, manage, and run configur Nios II Hardware Tab Group	tions	Ť.
Image: Second Secon	Name DECA_Hardware Project Target Connection Debug Project name: adc_demo Project ELF file name: C:\DATA\Altera\Literatur C:\DATA\Altera\Literatur File system ELF file name:	ger Common Source
∢ ₩ Filter matched 8 of 8 items		Apply Revert
?		Debug Close







5.4.6.6 Click Debug to begin debugging the application.

5.4.7 **Debug the application**

When you begin the debug session, the debugger downloads the code to the on-chip RAM, and begins executing the code. A breakpoint is automatically set at the entry point to main(), so the processor should be paused, waiting for you to resume.

5.4.7.1 Set a breakpoint at line 45 by double clicking in the blue margin to the left of the code.

This will stop the processor just before configuring the ADC core.

```
C
      main.c 🔀
                c altera_modular_adc.c
    #include "sys/alt_irq.h"
    #include "sys/alt_alarm.h"
  int main()
    {
      printf("adc_example.c\n\n");
      11*
      int i;
      int loop count;
      int led_output;
      float scale factor;
      int line in binned;
      alt_u32 *adc_data;
      alt u32 line in data;
      printf("********** System Information **********\n");
      printf("ADC SAMPLE STORE CSR BASE: 0x%08X\n",ADC SAMPLE STORE CSR BASE);
      printf("ADC SEQUENCER CSR BASE:
                                             0x%08X\n",ADC_SEQUENCER_CSR_BASE);
      printf("Number of Slots in Sequencer: %d\n\n",ADC SEQUENCER CSR CSD LENGTH);
      printf("Sequencer Slot 0: %s\n",ADC_SAMPLE_STORE_CSR_CSD_SLOT_0);
printf("Sequencer Slot 1: %s\n",ADC_SAMPLE_STORE_CSR_CSD_SLOT_1);
      printf("Sequencer Slot 2: %s\n",ADC_SAMPLE_STORE_CSR_CSD_SLOT_2);
      printf("Sequencer Slot 3: %s\n\n", ADC_SAMPLE_STORE_CSR_CSD_SLOT_3);

    Set a breakpoint here

      printf("*** Configure and start the ADC sequencer ***\n\n");
      adc stop(ADC SEQUENCER CSR BASE);
                                                                 // ADC must be stoppe
      adc_interrupt_disable(ADC_SEQUENCER_CSR_BASE);
                                                                 // Disable interrupts
      adc_clear_interrupt_status(ADC_SEQUENCER_CSR_BASE);
      adc_set_mode_run_continuously(ADC_SEQUENCER_CSR_BASE);
```

236

WOW







- Resume the processor: Run -> Resume, or press the Resume icon 5.4.7.2
- 5.4.7.3 Observe the console output. Notice the slot information matches the configuration we entered in the modular ADC core. This information was automatically passed from the hardware configuration in Qsys to the software via the .sopcinfo file that was used to create the application.

```
~ - 8
                                                             • 루 🖡
📃 Console 🎣 Tasks  Nios II Console 🔀
DECA_Hardware - cable: Arrow MAX 10 DECA on localhost [USB-1] device ID: 1 instance ID: 0 name: jtaguart_0
adc example.c
ADC SAMPLE STORE CSR BASE: 0x00040000
ADC_SEQUENCER_CSR_BASE: 0x00040200
Number of Slots in Sequencer: 4
Sequencer Slot 0: CH0
Sequencer Slot 1: CH5
Sequencer Slot 2: CH1
Sequencer Slot 3: CH5
******
*** Configure and start the ADC sequencer ***
```







5.4.7.4 Open SignalTap Analyzer: From Quartus, Tools → SignalTap II Logic Analyzer.

The Deca_tap_nios.stp system should open.

- Start the Analyzer in repetitive trigger mode: Processing -> Autorun Analysis, or press the autorun icon 5.4.7.5 2. The trigger condition has been configured to trigger on reads OR writes to the ADC (see this on the Setup tab), so every time Nios accesses the ADC, the Logic Analyzer should trigger, making it easy to understand how the system functions.
- 3 In the debugger, step through the code: $Run \rightarrow Step Over$, or by pressing the Step Over icon . After 5.4.7.6 each step, the SignalTap system should automatically trigger, and display the waveforms. See the screenshot below for the result of stepping over the adc_start(ADC_SEQUENCER_CSR_BASE) instruction.
- Resume the code by pressing the resume icon . This will cause the program to run freely until the 5.4.7.7 next breakpoint. Since there are no more breakpoints set, the system will perform 100,000 reads, and then complete.

SignalTap	I Logic Analyzer - C:/D	DATA/Altera/Literat	ure/Max10/	DECA/ADC Nios	Lab b139/ADC Ni	os Lab - Al	OC Nios Lab - 'Deca tap	nios.stpl*				6			
Eine gen groper regoessing toos window hep 🐨															
	のの推躍▼	R\$ 0													
Instance Manager: 🔊 🔊 🔳 🖄 Acquisition in progress								×	X JTAG Chain Configurator: JTAG ready X						
Instance		Status	Enabled	LEs: 2238	Memory: 311295	5 Small: (0/0 Medium: 115/	132 Large: 0,	Hardware	Arrow MA	X 10 DECA [U	ISB-1] *	Setup		
R auto_sgnaltap_0		Waitng for trigger	1	2238 cells	311296 bits	0 block	s 38 blocks	0 blocks	Devices	Lett 1045	ODA/ JES)/10	M500(CIE) -	Cons Chain		
									Device:	1071. 10413		Magaz(ch) +	Scarrenain		
									>> SOF	Manager:	<u>ж</u>] [<u>((</u>) ф	ut_fies/ADC_N	los_Lab.sof		
									1						
Ing: Trg @	2015/04/22 10:13:53 (0:	0:3.9 elapsed)			🦛 I 🛛 🔹			23							
Type Alia	s	Name			23 Value	24 -	128 -64	Q	64	128	192	256	320		
1	1 HKEY[10] 3h						3h								
3	adc_qsys:u0 adc_qsys	adc_qsys:u0 adc_qsys_ADC:adc sample_store_csr_read					Nrite of								
10	adc_qsys:u0 adc_qsy	adc_qsys:u0jadc_qsys_ADC:adcjsample_store_csr_write					0x00000001								
5	the adc_qsys:u0 adc_	+ adc_qsys:u0 adc_qsys_ADC:adc sample_store_csr_address[60]					00h to CSR			00h					
8	I adc_qsys:u0 adc_	adc_qsys:u0jadc_qsys_ADC:adcjsample_store_csr_readdata[010]					0000000h			0000000h					
5		Finadc_gsystu0jadc_gsys_ADC:adcjsample_store_csr_writedata[31_0]					0000000h			00000000					
3	adc_qsys:u0 adc_qsy	adc_qsys:u0 adc_qsys_ADC:adc sequencer_csr_read													
3	adc_qsys:u0 adc_qsy	adc_qsys:u0 adc_qsys_ADC:adc sequencer_csr_write								Sequencer started,					
2	adc_qsys100adc_qsys	adc_qsystii0[adc_qsys_ADC:adc]sequencer_csr_address						and storing samples							
5	terr adc_qsys:u0 adc_	adc_qsys:u0 adc_qsys_ADC:adc sequencer_csr_readdata[310]						0000000h							
8	adc_qsys:u0 adc_qsys_ADC:adc sequencer_csr_writedata[310]				0000001h	0000000h				0	00,0001h				
3				00h				00h							
10	C:adc altera_modula	e_internal(rsp_valid	0												
-	adu_qsys.u0 adb_qsy	s_ADC.adc sample_s	store_iq_irq		0	_									
						*							F		
Data 🔊	Satup														
Hierarchy Dis	solav:	×	Data Loc: 1	สเ									×		
1 1 -	ADC Nios Lab		auto so	iraltan 0											
▲ 👽 🦈 adc_qsys_ADC:adc															
1	📝 🌥 altera modula	ar adc 💌													
🖁 auto_sig	jnal-ap_0														
												0%	00:00:00		

238

ЛПЛЛ





Max10 DECA Workshop Manual



Rerun the application, if desired. To restart the code, click on the DECA_Hardware option under the under the Debug Icon:

Nios II Debug - adc_demo/adc_demo.c - Eclipse										
File	Edit	Source	Refactor	Navigate	Search	Project	Nios II	Run Window Help		
5	- 13	隠 色	1	$\mathbf{x} \mid \mathbf{x} \mid$	0 2	IP 11		🕸 • Q • Q • 😕 🗁 🔗 • [
蓉	Debug	x						1 DECA_Hardware		
								Debug As Debug Configurations Organize Favorites		

5.4.7.9 **Congratulations. You have completed the ADC capture lab!** Feel free to experiment with changes to the c-code and the SignalTap analysis system as time permits. See the following section for additional reference information.

5.4.8 Other [possibly] helpful information

5.4.8.1 How big is my code?

If you plan to run Nios from a Max10 device without external RAM, on-chip RAM will often be the limiting factor in your design. So the question arises, how can I easily find out the size of my code? Use the nios2-stackreport command, from a Nios command shell.











5.4.8.2 What other signals are connected to the ADC on the DECA board?

The DECA schematics are included in the lab files. For convenience, here is a screenshot of the relevant section. ANAIN1 and ANAIN2 come from the SMA connectors, Line_IN_L is connected to the left channel of the Line In jack, and the rest are connected to the 46 pin headers.



That completes this lab section.



240





Max10 DECA Workshop Manual

