Integrating Arm Cortex-M Processors into Xilinx FPGAs

Course Workbook

Copyright 2020 Adiuvo Engineering & Training, Ltd.

Table of Contents

About this Workbook	Page 3
Pre-Lab: Workshop Pre-requisites	Page 4
Lab 1: Exploring the Architecture in Vivado	Page 7
Lab 2: Saying Hello World	Page 25
Lab 3: Completing a Simple Application	Page 47

This workbook is designed to be used in conjunction with the Integrating Arm Cortex-M processors in Xilinx FPGAs course.

The contents of this workbook are created by Adiuvo Engineering & Training, Ltd.

If you have any questions about the contents, or need assistance, please contact Adam Taylor at adam@adiuvoengineering.com.

Pre-Lab Workshop Pre-requisites

Required Hardware

The hands-on labs in this course use the following hardware:

- Digilent Arty S7-50 development board
- Digilent PmodNAV
- Digilent PmodHYGRO
- USB-A to Micro-B cable

Downloads and Installations

Step 1 – Download and install the following at least 1 day prior to the workshop. This may take a significant amount of time and drive space.

Vivado 2018.2 **Please check that it is this version and not a newer version**	Download
Digilent Board Files	Download
Arm Design Start FPGA IP **Download Cortex-M1 for Xilinx**	Download
Arm Keil MDK Tools	Download
Arm Keil Licensing Tool	Download
Digilent Vivado IP – Master Branch	Download

Step 1 - Ensure you have the following environment path variables set:

C:\Keil_v5\ARM\ARMCC\bin

C:\Xilinx\Vivado\2018.2\bin

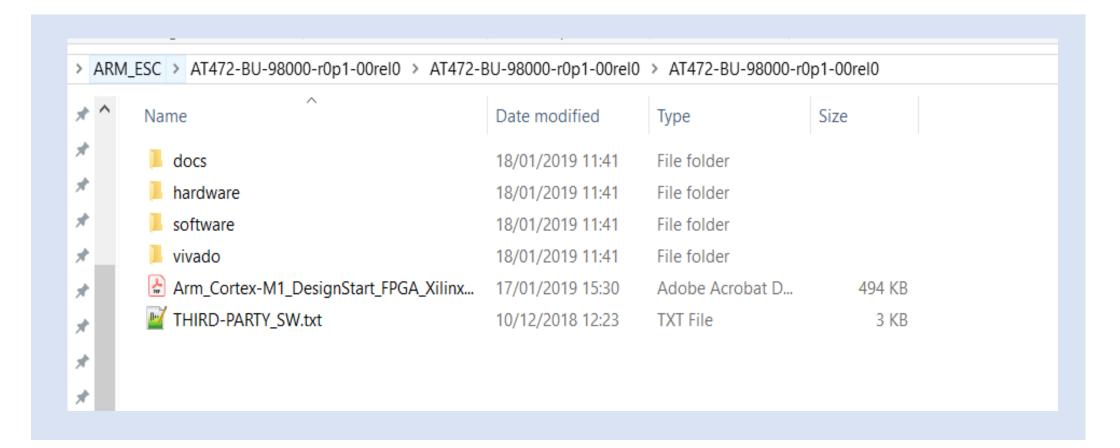
%SystemRoot%	^ <u>N</u> ew
%SystemRoot%\System32\Wbem	
%SYSTEMROOT%\System32\WindowsPowerShell\v1.0\	Edit
C:\Users\aptay\AppData\Local\Programs\Python\Python	_
C:\Aldec\Riviera-PRO-2016.10-x64\bin	Browse
C:\Xilinx\Vivado\2016.2\bin	2.0000
C:\Python27\python.exe	Delete
C:\Program Files\PuTTY\	Delete
C:\Program Files\Git\cmd	
C:\Aldec\Riviera-PRO-2017.02-x64\bin	Move Up
C:\Program Files (x86)\Pico Technology\PicoScope6\	wove <u>op</u>
C:\Program Files (x86)\IVI Foundation\VISA\WinNT\Bin\	Maus Davis
C:\Program Files\IVI Foundation\VISA\Win64\Bin\	Move Down
C:\Program Files (x86)\IVI Foundation\VISA\WinNT\Bin	
%SYSTEMROOT%\System32\OpenSSH\	
Gil Program Files (Gil) bin	Edit <u>t</u> ext
C:\Xilinx\Vivado\2018.2\bin	
C.(Frogram Files (Amazon (AwSeer, bin)	
C:\Keil_v5\ARM\ARMCC\bin	
C:\HashiCorp\vagrant\bin	
C:\Program Files\TortoiseSVN\bin	~

Step 2 – Navigate to the directory containing the downloaded Arm M1 Core. This will be called **AT472-BU-98000-r0p1-00rel0.tgz.**

Step 3 – Using a compression program (e.g., seven zip) unzip everything, including the inner directory.

> OS (C:) > ARM	ESC			
* ^	Name	Date modified	Туре	Size
*	AT472-BU-98000-r0p1-00rel0.tgz	18/04/2019 13:30	TGZ File	6,699 KB
*				
*				
*				

Step 4 – When extracted you should see the directory structure below.



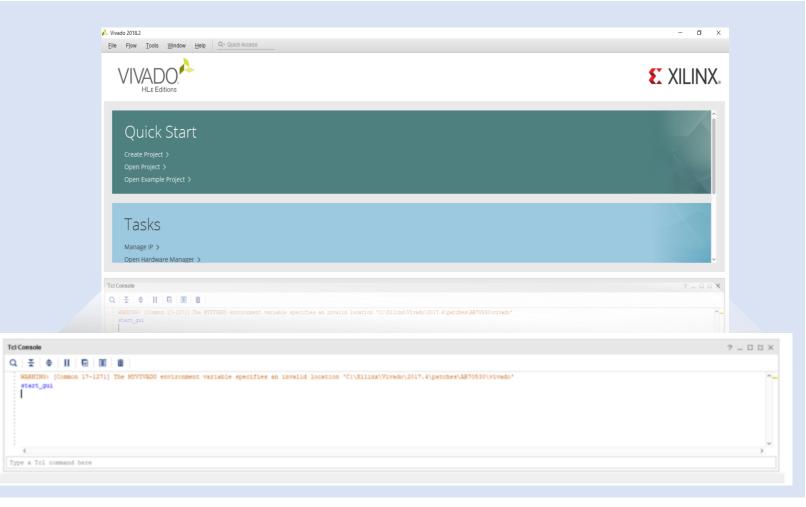
Step 5 – If you have not installed the Digilent Board files already, copy the Digilent boards contained in the directory below into <**Vivado Install path>\Vivado\2018.2\data\boards\board_files**

 ✓ Type File folder File folder File folder 	Compressed size	Password p	Size	Ratio	Date modified 13/12/2018 20:55
File folder					13/12/2018 20:55
File folder					13/12/2018 20:55
					13/12/2018 20:55
File folder					13/12/2018 20:55
File folder					13/12/2018 20:55
File folder					13/12/2018 20:55
File folder					13/12/2018 20:55
File folder					13/12/2018 20:55
File folder					13/12/2018 20:55
File folder					13/12/2018 20:55
File folder					13/12/2018 20:55
File folder					13/12/2018 20:55
	File folder File folder File folder File folder File folder File folder				

Step 6 – Open Vivado 2018.2. If you are familiar with Vivado <u>do not</u> do this by opening the project file.

Step 7 – Change the directory using the TCL console to:

cd C:\\<YOUR DIRECTORY>\\AT472-BU-98000-r0p1-00rel0



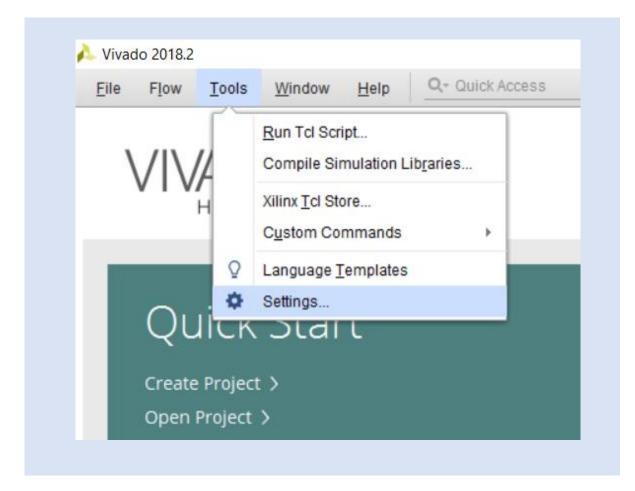
Step 8 – In the TCL Console enter the following command exec subst V: .

Make sure there is a space between ":" and "."

This gets around the character limit on file paths in Windows by mapping the PWD to the V directory. Using a file browser to explore the V directory should show this file structure:

	^			
* ^	Name	Date modified	Туре	Size
*	📙 docs	18/01/2019 11:41	File folder	
*	📕 hardware	18/01/2019 11:41	File folder	
*	🦲 software	18/01/2019 11:41	File folder	
*	📕 vivado	18/01/2019 11:41	File folder	
*	🛃 Arm_Cortex-M1_DesignStart_FPGA_Xilinx	17/01/2019 15:30	Adobe Acrobat D	494 KB

Step 9 – The next thing we need to do is map in the IP repository. In Vivado click on **Tools** \rightarrow **Settings**



Step 10 – In the dialog box select **IP Defaults**.

Q-	IP Defaults	
Tool Settings Project	Specify default IP example directory and IP repository search paths.	e 1997 (* 1997)
IP Defaults	Default IP Example Project Directory	
Source File Display WebTalk Help	Last project's directory Specify project directory:	
> Text Editor	IP Catalog	
3rd Party Simulators > Colors Selection Rules Shortcuts > Strategies	Default IP Repository Search Paths	
> Window Behavior	No content	
	Note: Default IP repository search paths will be applied only at new project creation.	
?		

Lab 1

Step 11 – Select the **Add IP** repository button (shown as a +).

Navigate to V:\Vivado\Arm_IPI_Repository

Q		
Tool Settings	IP Defaults Specify default IP example directory and IP repository search paths.	- 🖊
Project		
IP Defaults	Default IP Example Project Directory	
Source File	Delault in Example Project Directory	
Display	Last project's directory	
WebTalk	Specify project directory:	
Help		
> Text Editor	IP Catalog	
3rd Party Simulators		
> Colors	Default IP Repository Search Paths	
Selection Rules	$+ - \uparrow \downarrow $	
Shortcuts		
> Strategies	V:/vivado/Arm_ipi_repository	
> Window Behavior		
	1 Note: Default IP repository search paths will be applied only at new	
	project creation.	
?)	OK Cancel Apply F	Restore.

Step 12 – We are now ready to open the project. Select **File**→**Project**→**Open**. Navigate to **V:/hardware/m1_for_arty_s7/m1_for_arty_s7** Select the project file **m1_for_arty_s7.xpr**

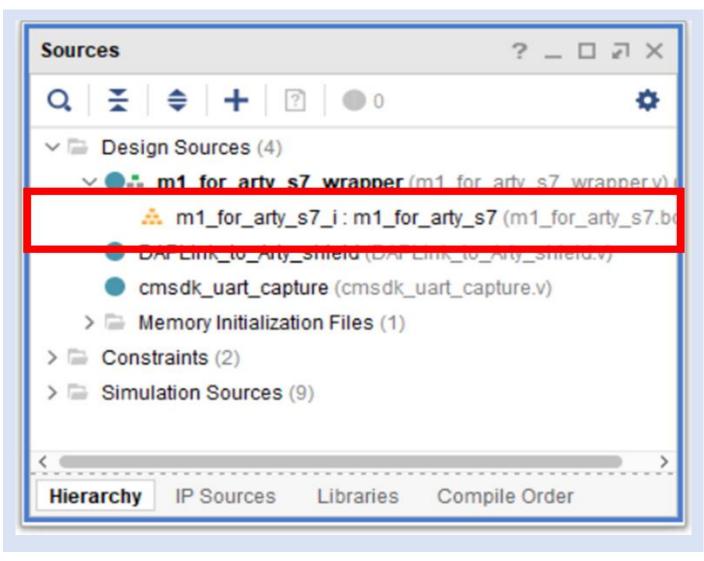
A Open Project	×
Look in: im1_for_arty_s7	✓ ↑ ☆ 및 ± A ☎ X C Ⅲ Ⅲ
m1_for_arty_s7.xpr	Recent Directories V:/hardware/m1_for_arty_s7 File Preview File: m1_for_arty_s7.xpr Directory: V:/hardware/m1_for_arty_s7/m1_for_arty_s7 Created: Today at 12:07 PM Accessed: Today at 12:07 PM Modified: Thursday 13.12.18 03:15 PM Size: 19 KB Type: Vivado project
	Version: Vivado v2018.2 Owner: DESKTOP-HQKVQ13\aptay
File name: m1_for_arty_s7.xpr	
Files of type: Vivado, PlanAhead, and ISE Project Files (xpr, ppr, xise)	×
	OK Cancel

Step 13 – If you see a critical message about simulation do not worry. Click on **Continue / OK.**

To open the block diagram, select on the > arrow next to the top level.

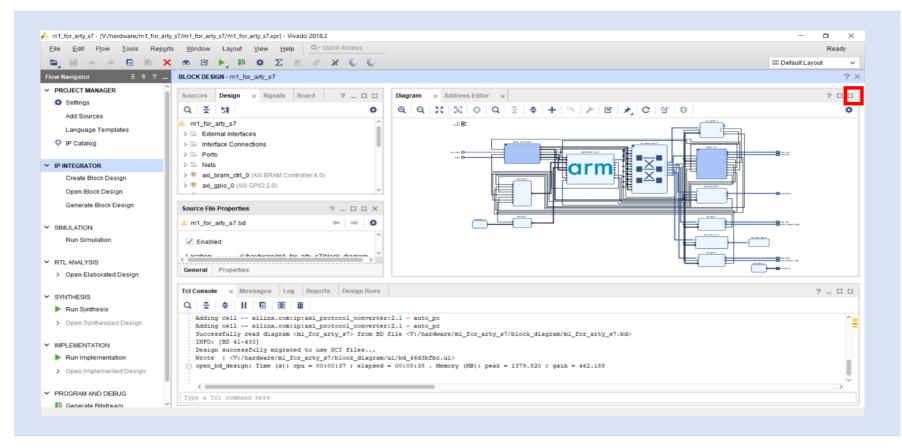
Sources	? _ D @ X
Q ¥ \$ + ? ● 0	\$
V Design Sources (4)	
> 📑 m1_for_arty_s7_wrappe	r (m1_for_arty_s7_wrapper.v)
DAPLINK_to_Arty_shield (D)	APLINK_to_Arty_shield.v)
cmsdk_uart_capture (cmsd	lk_uart_capture.v)
> 🗁 Memory Initialization Files (1)
> Constraints (2)	
> Simulation Sources (9)	
<	>
Hierarchy IP Sources Libraries	s Compile Order

Step 14 – Double click on the highlighted file. This will open the block diagram.



Step 15 – If you want to pop out and explore the block diagram click on the button highlighted below.

You can examine the settings of the IP cores by double clicking on them, however, do not change anything.



Step 16 – The next step is to set up the software environment launch SDK by selecting **File** \rightarrow Launch.

This will open a dialog, set the dialog as below in the diagram.

🝌 Launch SDK		×
Launch software development tool.		4
Exported location: V:/software		~
Workspace: V:/software/m1_for_arty_s7/sdk_workspace		~
?	ОК	Cancel

Step 17 – If you see a warning click **Yes**, SDK will then open

3 - 🗟 🔯 🗟 💌 🗞 - 1 🎋 - 0 - 1 🔌 🗖 🏭 🖬 🚳	A • \$ + + + +								Quick Access	B
🗅 Project Explorer 🛛 📄 😫 🛛 🔻 🗢	🗖 🗖 🗊 system.hdf 🛙						- [🗖 🗄 Outline 🛛 🔁 Docume	e Make Tar	
v / m1_for_arty_s7_wrapper_hw_platform_0 in system.hdf	m1_for_arty_s7_wrappe	m1_for_arty_s7_wrapper_hw_platform_0 Hardware Platform Specification								
	Design Information	Design Information								
	Created With: Vivado 2	Target FPGA Device: 7s50 Part: xc7s50csga324-1 Created With: Vivado 2018.2 Created On: Thu Sep 20 14:20:31 2018								
	Address Map for processor Co	rtex_M1_0								
	Cell	Base Addr	High Add	Ir Slave I/f	Mem/Reg					
	axi_gpio_1	0x40120000	0x4012fff	-	REGISTER					
	daplink_if_0_axi_single_spi_0		0x4003fff	-	REGISTER					
	daplink_if_0_axi_xip_quad_s		0x4000fff	-	REGISTER					
	daplink_if_0_axi_xip_quad_s		0x000ffff		REGISTER					
	daplink_if_0_axi_gpio_0	0x40010000	0x4001fff	-	REGISTER					
	axi_gpio_0	0x40110000	0x4011fff	f S_AXI	REGISTER					
	axi_uartlite_0	0x40100000	0x4010fff	f S_AXI	REGISTER					
	axi_bram_ctrl_0	0x60000000	0x60001f	ff S_AXI	MEMORY					
	daplink_if_0_axi_quad_spi_0	0x40020000	0x4002fff	f AXI_LITE	REGISTER					
	axi_quad_spi_0	0x40130000	0x4013fff	f AXI_LITE	REGISTER					
	IP blocks present in the design Clocks_and_Resets_i_const_1		constant	1.1						
	daplink if 0 DAPLink to Arty s Overview	hield 0 D	IAPLink to A	rtv.shield 1.0				×		
a Target Connections 🛛 🕹 🚄		sole 🔲 Prope	erties 📮 SD	K Terminal		~ - 8	BSDK Log ⊠			
> 😂 Hardware Server > 😂 Linux TCF Agent > 🗠 QEMU TcfGdbClient	0 items Description			Resource	Path	Loc	12:44:03 INFO : XSCT server has st 12:44:03 INFO : Successfully done 12:44:05 INFO : Successfully done 12:44:05 INFO : Restoring global r V:\vivado\Arm_sw_repository 12:44:05 INFO : Processing command 12:44:05 INFO : ***** MYVIVADO is	setting XSCT server co setting SDK workspace repository preferences: d line option -hwspec V	/:/software/m1_	_fo

Step 18 – We need to add in the SW IP so that we can build the SW application correctly.

Select Xilinx→Repositories.

sdk_workspace - C/C++ - m1_for_arty_s7_w	rapper_hw_platform_0/system.hdf - Xilinx SD	Ж
File Edit Navigate Search Project Run	Xilinx Window Help	
	Generate linker script Board Support Package Settings	
ြဲ Project Explorer 🛛	Repositories	
w Use m1_for_arty_s7_wrapper_hw_platform_C system.hdf	Dump/Restore Da	Preferences P
	XSCT Console	evice: 7s50 Part: xc7s50c: With: Vivado 2
	Created	d On: Thu Sep

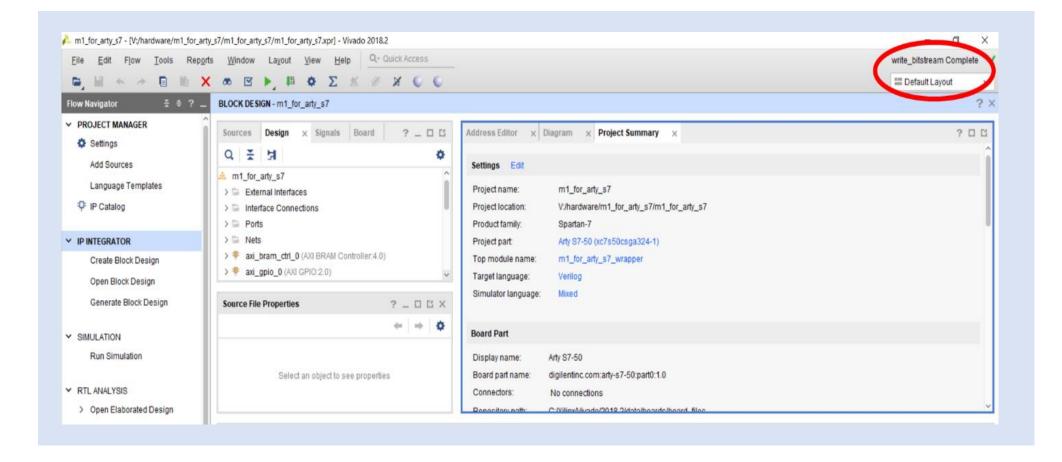
Step 19 – In the resultant dialog select the following directory:

V:\Vivado\Arm_SW_repository

We are now ready to start using our Arm Cortex-M1 system, this completes Lab 1!

General C(-++) Help Local Repositories (available to the current we limit of the drivers', 'bap' or 'sw_services' sub-directory by locally contains the 'drivers', 'bap' or 'sw_services' sub-directory by locally contains the 'drivers', 'bap' or 'sw_services' sub-directory by locally contains the 'drivers', 'bap' or 'sw_services' sub-directory by locally contains the 'drivers', 'bap' or 'sw_services' sub-directory by locally contains the 'drivers', 'bap' or 'sw_services' sub-directory by locally contains the 'drivers', 'bap' or 'sw_services' sub-directory by locally contains the 'drivers', 'bap' or 'sw_services' sub-directory by locally contains the 'drivers', 'bap' or 'sw_services' sub-directory by locally contains the 'drivers', 'bap' or 'sw_services' sub-directory by locally contains the 'drivers', 'bap' or 'sw_services' sub-directory by locally contains the 'drivers', 'bap' or 'sw_services' sub-directory by locally contains the 'drivers', 'bap' or 'sw_services' sub-directory by locally contains the 'drivers', 'bap' or 'sw_services' sub-directory by locally contains the 'drivers', 'bap' or 'sw_services' sub-directory by locally contains the 'drivers', 'bap' or 'sw_services' sub-directory 's is Downloads > Remote Systemis Clobal Repositories (available across workspant's by Downloads > is Downloads > Wiwado Varm_sw_repository SDK Installation Repositories > is Downloads SDK Capabilities SDK Installation Repositories > is Of Ware C/X/limx/SDK/2018.2/data/embeddedsw > is Software > is ortware > is ortware > is ortware > is ortware > is ortware > is ortware </th <th>Preferences</th> <th></th> <th></th> <th></th>	Preferences			
 C/C++ Help Install/Update Remote Developmen Remote Systems Ram/Oebug Tacing Kilinx SDK Global Repositories (available across workspath SDK Imstallation Repositories SDK Capabilities Toolchain Preferences Flash Programming Hardware Specific C/Xilinx/SDK/2018.2/data/embeddedsw Rescan Repositories Note: Local repository settings take preceder Rescan Repository settings take preceder Model Amplitude across take preceder 		Add, remove or change the order of S	DK's software repositories.	
 Help Help Histal/Update Remote Developmen Remote Systems Run/Debug Tracing Stand Boot Image Boot Image Boot Image Boot Image SDK Imstallation Repositories (available across workspace Vivivado/xrm_sw_repository Downloads Downloads Downloads Wideos Wideos		Local Repositories (available to the current wo	Browse For Folder	
Note: Local repository settings take precedenc My Passport (D:)	Help Install/Update Remote Developmen Remote Systems Run/Debug Team Terminal Tracing Xilinx SDK Boot Image BSP Preferences Flash Programming Hardware Specific Log Information Le Repositories SDK Capabilities	V:\vivado\Arm_sw_repository SDK Installation Repositories C:/Xilinx/SDK/2018.2/data/embeddedsw	 This PC 3D Objects Desktop Downloads Music Downloads Music Pictures Videos Videos CS (C:) My Passport (D:) Local Disk (L:) Cos (V:) Local Disk (L:) OS (V:) I.Xil docs Inardware precompiled_sim_libs software vivado Arm_ipi_repository CortexM 	_services' sub-director
		Note: Local repository settings take precedence		
Arm av specificat			> _ My Passport (D:)	
> Folder: Ann_sw_repository	>		Folder: Arm_sw_repository	
				OK Cancel

Step 1 – Wait for Vivado to complete the bitstream generation.



Step 2 – Export the hardware definition select

File→Export→Export Hardware

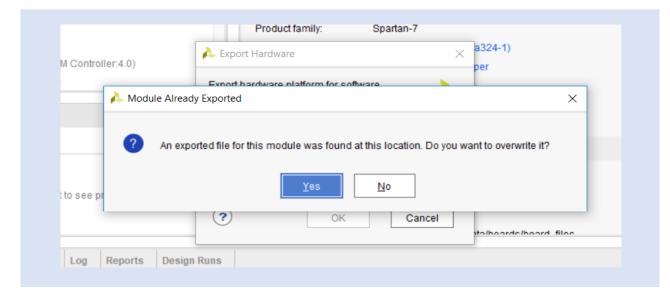
<u>F</u> ile	Edit Flow Tools	Rep <u>o</u> rts	Window Layout View
1	Project	•	× 🛛 🕨 👬 🔅
8	Add Sources	Alt+A	LOCK DE SIGN - m1_for_arty_
•	Close Project		Sources Design × Si
	Save Block Design	Ctrl+S	Q 꽃 뇌
	<u>C</u> lose Block Design		 m1_for_arty_s7 External Interfaces
	Constraints	•	> 🖹 Interface Connections
	Simulation Waveform	•	> 🚍 Ports
1	Checkpoint	•	Nets # axi_bram_ctrl_0 (AXI B)
	Īb	•	> 🌻 axi_gpio_0 (AXI GPIO:2
	Text E <u>d</u> itor	Þ	Pausa File Descartion
	Import	Þ	Source File Properties
	Export	•	Export <u>H</u> ardware
	Launch SDK		Export Block Design
	Print	Ctrl+P	Export Bitstream File
1	Exit		Export Simulation

Step 3 – This will create a pop up. Select the export location as **V:\software**

You do not need to include the bit stream

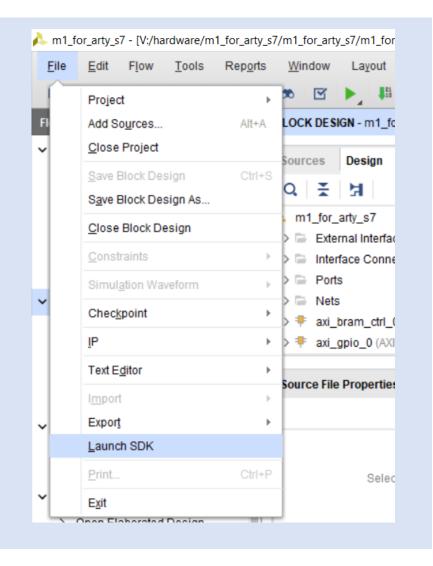
If you see the warning below click Yes.

À Export Hardwa	re	×				
Export hardware platform for software development tools.						
Include bits	tream					
Export to:	V:/software	~				
?	ОК	Cancel				



Step 4 – Next, we need to open XDSK. Select

File→Launch SDK

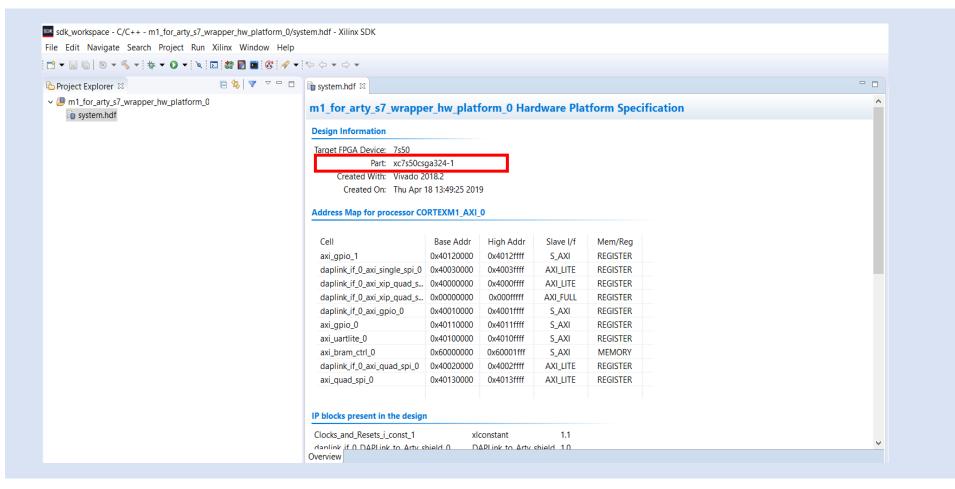


Step 5 – In the pop up box, for the export location select the location we just exported the hardware definition to (**V:\Software**) and select the workspace located at:

V:\software\m1_for_arty_s7\sdk_workspace

aunch software develo	pment tool.	
Exported location:	DV:/software	 ~
Workspace: 🔁 V	/software/m1_for_arty_s7/sdk_workspace	~

Step 6 – Once SDK is opened, we will see the hardware definition imported into SDK. Note, the part should show **xc7S50**. You will also see the address spaces for all of the added peripherals.



Lab 2

Step 7 – The next step in SDK is to create a BSP, this will create APIs which enable us to work with the peripherals included in the design. Select **File** → **New** → **Board Support Package**.

_		kspace - C/ Navigate						stem.hdf - X	ilinx SDK	
	New Open I Open I	File Projects fro	m File Sy	stem	Alt+	-Shift+N	>	Applicatio SPM Project Board Sup	-	
	Close Close	AII			Ctrl+	Ctrl+W Shift+W	59	Project Source Fol Folder	der	
	Save Save A Save A Revert	.11			Ctrl	Ctrl+S +Shift+S	Ċ	Source File Header File File from T	e	
1	Move Renam					F2	 ⊡ 	Class Other	Man for pro	Ctrl+N

Lab 2

Step 8 – In the next dialog leave the name as the automatically generated one and click **Finish**.

www Board Suppo	rt Package Project —		×
Xilinx Board Supp	ort Package Project		пЪ
Create a Board Supp	ort Package.		
Project name: stand	alone bsp 0		
Use <u>d</u> efault locati			
Location: V:\softwar	e\m1_for_arty_s7\sdk_workspace\standalone_bsp_0	Browse	e
Choose file	system: default \vee		
Hardware Platform:	m1_for_arty_s7_wrapper_hw_platform_0	\sim	N <u>e</u> w
CPU:	CORTEXM1_AXI_0	\sim	
Compiler:	32-bit \vee		
standalone	Standalone is a simple, low-level software layer. It provides access to basic features such as caches, interrupts and exceptions as well as the basic feature environment, such as standard input and output, profiling, abort and exit.		
?	<u> </u>	Cance	el

Step 9 – This will open the BSP configuration page. Explore around and ensure the **STDIN STDOUT** is set to the **AXI UART**. Once you have finished exploring click **Finish** and the BSP will be generated.

Board Support Package Set Board Support Package S Control various settings of you	Settings	kage.		×	Board Support Package Se Board Support Package S Control various settings of yo	Settings					×
 Overview standalone drivers CORTEXM1_AXI_0 	Processor:	ation: V:\softv CORTE) ct to the librar	Standalone is a simple, low-level software I processor features such as caches, interrupts features of a hosted environment, such as st abort and exit. vare\m1_for_arty_s7\sdk_workspace\m1_for_ (M1_AXI_0 ies you want included in your Board Support Description	and exceptions as well as the basic andard input and output, profiling, arty_s7_wrapper_hw_platform_0\system	 Overview standalone drivers CORTEXM1_AXI_0 	Configuration for OS: sta Name hypervisor_guest lockstep_mode_debug sleep_timer stdin stdout ttc_select_cntr zynqmp_fsbl_bsp > microblaze_exceptions > enable_sw_intrusive_profited 	Value false false axi_uartlite_0 axi_uartlite_0 2 false iii false iiii false iiii jalse jalse	Default false false none none 2 false false false false i alse alse alse alse alse alse alse alse	Type boolean boolean peripheral peripheral enum boolean boolean boolean	Description Enable hypervisor guest support for A53 6 Enable debug logic in non-JTAG boot mod This parameter is used to select specific til stdin peripheral stdout peripheral Selects the counter to be used in the repec Disable or Enable Optimization for FSBL's Enable MicroBlaze Exceptions Enable S/W Intrusive Profiling on Hardwar	ide, ime ectiv s BS
0				OK Cancel	0					OK Cancel	

Lab 2

Lab 2

Lab 2: Saying Hello World

Step 10 – Once the BSP has finished generating you will notice a new MSS file opens in SDK, along with a new project under the project explorer. This is the board support package. You can see the BSP files and the APIs included in the design.

📑 🕶 🔚 🐚 🗞 🕶 🗞 🕶 🔅 🕶 🔕 📼	
	E S V V C C System.hdf System.mss 82
<u> m1_for_arty_s7_wrapper_hw_platform_0 m1_for_brance m1_for_branc</u>	standalone_bsp_0 Board Support Package
system.hdf setandalone_bsp_0 i BSP Documentation	Modify this BSP's Settings Re-generate BSP Sources
✓ ➢ CORTEXM1_AXI_0	Target Information
code include	This Board Support Package is compiled to run on the following target.
> 🗁 lib	Hardware Specification: V:\software\m1_for_arty_s7\sdk_workspace\m1_for_arty_s7_wrapper_hw_platform_0\system.hdf
✓ 🍃 libsrc	Target Processor: CORTEXM1_AXI_0
> 🗁 bram_v4_2 > 🗁 cpu cortexm1 v1 1	Operating System
> > p gpio_v4_3	Board Support Package OS.
> 🗁 spi_v4_4	Name: standalone
> 😂 standalone_v6_7	Version: 6.7
> 🗁 uartlite_v3_2	Description: Standalone is a simple, low-level software layer. It provides access to basic processor features such as caches, interrupts and exceptions
🚡 Makefile	as well as the basic features of a hosted environment, such as standard input and output, profiling, abort and exit.
system.mss	Documentation: Not found
	Peripheral Drivers
	Drivers present in the Board Support Package.
	axi_bram_ctrl_0 bram Documentation Import Examples
	axi_gpio_0 gpio Documentation Import Examples
	axi_gpio_1 gpio Documentation Import Examples
	axi_quad_spi_0 spi <u>Documentation Import Examples</u>
	axi_uartlite_0 uartlite Documentation Import Examples
	daplink_if_0_axi_gpio_0 gpio <u>Documentation Import Examples</u>

Step 11 – Copy the files

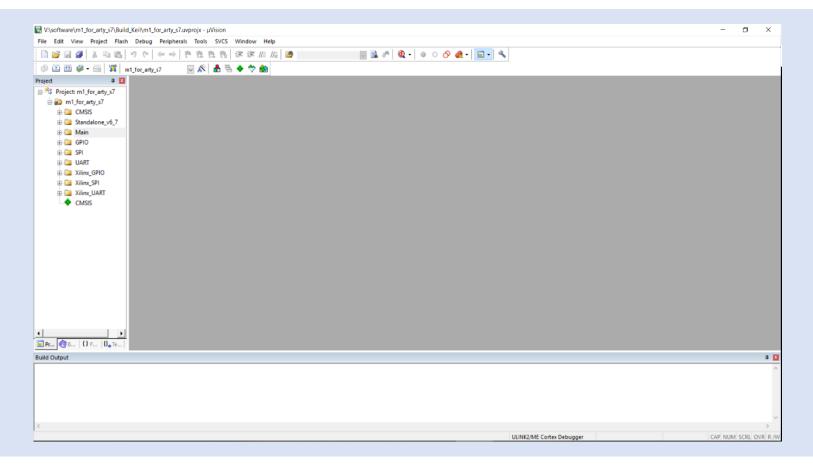
Xpsuedo_asm_rvct.c

Xpseudo_asm_rvct.h

From V:\vivado\Arm_sw_repository\CortexM\bsp\standalone_v6_7\src\arm\cortexm1\armcc

Into V:\software\m1_for_arty_s7\sdk_workspace\standalone_bsp_0\CORTEXM1_AXI_0\include

Step 12 – We now need to develop our SW application in Arm KEIL. Navigate to V:/software/m1_for_arty_s7/Build_Keil/. Click on the file m1_for_arty_s7.uvprojx This will open the Arm Keil project.



Step 13 – Make sure the target is set to **m1_for_arty_s7**.

Open the **Main** directory under the project, and double click on **main.c.** This will open the file for editing.

Step 14 – Scroll down to line 149 and change the output string from

print ("Example design for Digilent S7 board\r\n");

to

print ("Adiuvo Tutorial Example design for Digilent S7 board \r\n");

Image:	File	Edit	View	Project	t Fla	sh C	lebu
toject		2 14	ø	X 0	0.18	1 -7	(°
Project: m1_for_arty_s7 m1_for_arty_s7 m1_for_arty_s7 m2_CMSIS m2_Standalone_v6_7 m3_GPIO m3_SPI m3_UART m3_Xiinx_GPIO m3_Xiinx_SPI m3_Xiinx_UART m3_Xiinx_UART m3_Xiinx_UAR	٩	(1) (E		• 68	11	nt)	or_ar
m1_for_wty_s7 GMSIS GM	Project					1	
CMSIS CMS CMSIS C	8 1	Projec	ct: m1	for_arty	1.57		
Standalone_v6_7 Main GPIO GPIO JART	ė	🔊 m	(Jer.)	arty_s7			
Main GPIO GPIO SPI GPIO Art Art Xiinx_GPIO Xiinx_SPI Xiinx_LART		- (e) 🗀	CMS	35			
GPIO SPI GPIO SPI GPIO		(e) C	Stan	dalone,	y6,7		
SPI UART Jinx_GPIO Zilinx_SPI Zilinx_LART		÷-	Mair	8			
				>			
 a Xiinx_GPIO a Xiinx_SPI a Xiinx_UART 				_			
 A Siless SPI A Siless UART 							
🗉 🦢 Xilinx UART				-			
- CN00							
		•	CMS	15			

Step 15 – Before we can click on Build, we need to check the name of the Arm Cortex processor generated by SDK. Use a file explorer and navigate to

V:\software\m1_for_arty_s7\sdk_workspace\standalone_bsp_0

Ensure the folder is named **CORTEX_M1_0** and not **CORTEXM1_AXI_0** or similar, if they are, correct the naming to **CORTEX_M1_0**.

Lab 2

Lab 2: Saying Hello World

Step 16– Click on Build and you should see the project compile and produce necessary output files.

📄 pi 🛄 🍠 🕺 🐚 🕵 🗌		
	り (*) ← ⇒ № 魯 魯 優 罪 罪 / [/ [/]] / / [/] / [/]] / [/] / [/] / [/]] / (*)	
🈂 🖉 🛗 🌶 - 📖 🗱 m	1_for_arty_s7 🛛 🖂 🛣 📥 🗢 🐡 🏟	
roject 📮 🔀	main.c	•
Project: m1_for_arty_s7	139 case 0xC23 : strcpy(CPU_name, "Cortex-M3"); break;	
🖻 🚂 m1_for_arty_s7	<pre>140 default : sprintf(CPU_name, "Unknown %x", CPU_part);</pre>	
E CMSIS	141) 142 -	
🖃 🦾 Standalone_v6_7	143 sprintf (debugStr, "Arm %s Revision %i Variant %i\r\n\n", CPU name, CPU rev, CPU var);	
e-line outbyte.c	144	
	145 bifndef SIM_BUILD	
	146 // Use Xilinx version print command	
. xil_printf.c	147 print (""""""""""""""""""""""""""""""""""""	
🕀 🛄 Main	<pre>148 print (debugStr); 149 print ("Example design for Digilent S7 board\r\n");</pre>	
🕀 🧰 GPIO	150 if (DAPLinkFittedn)	
e 🧰 SPI	<pre>151 print ("\nV2C-DAPLink board not detected\r\n");</pre>	
	152 else	
E DAKI	153 print ("\nV2C-DAPLink board detected\r\n");	
	<pre>154 print ("Use DIP switches and push buttons to\r\ncontrol LEDS\r\n"); 155 print ("Version 1.1\r\n");</pre>	
Xilinx_UART	<pre>155 print ("version 1:1/(n"); 156 print ("************************************</pre>	
	157 felse	
CMSIS	158 print (debugStr);	
	159 #endif	
	162 // Test the BRAM	
	163 // ******	
	164	
	165 // Write to BRAM	
	<pre>166 for(i=0; i< (sizeof(bram_data)/sizeof(u32)); i++) 167 *pBRAMmemory++ = bram_data[i];</pre>	
· · ·	167 *pokwimenory++ = brain_data[1];	
🖬 Pr 🌏 B {} F 🛛 🗛 Te	¢	>
uild Output		¢
:\software\ml for arty	s7\Build Keil>copy bram s7.*\\hardware\ml_for_arty s7\ml_for_arty s7	
ram_s7.elf		
ram_s7.hex		
2 file(s) copied		
:\software\ml_for_arty_ l_file(s)_copied	<pre>37\Build_Keil>copy qspi_s7.hex\\hardware\ml_for_arty_s7\testbench</pre>	
	7.axf" - 0 Error(s), 26 Warning(s).	
uild Time Elapsed: 00:		

Lab 2

Lab 2: Saying Hello World

Step 17– Use a file browser to navigate to V:\hardware\m1_for_arty_s7\m1_for_arty_s7. You will see a files named **bram_s7.elf** and **bram_s7.hex** (these should have today's time stamp as you just generated them).

Name	Date modified	Туре	Size
m1_for_arty_s7.cache	18/04/2019 13:35	File folder	
m1_for_arty_s7.hw	18/04/2019 13:35	File folder	
m1_for_arty_s7.runs	18/04/2019 13:49	File folder	
m1_for_arty_s7.sdk	18/04/2019 14:12	File folder	
m1_for_arty_s7.sim	18/04/2019 13:35	File folder	
bram_s7.elf	18/04/2019 14:49	ELF File	137 KB
bram_s7.hex	18/04/2019 14:49	HEX File	29 KE
🗋 m1.mmi	13/12/2018 15:15	MMI File	3 KB
🝌 m1_for_arty_s7.xpr	18/04/2019 14:02	Vivado Project File	19 KB
m1_for_arty_s7_reference.bit	13/12/2018 15:15	BIT File	2,141 KB
m1_for_arty_s7_reference.mcs	13/12/2018 15:15	MCS File	6,022 KB
🞯 make_mmi_file.tcl	17/01/2019 15:30	TCL File	8 KB
make_prog_files.bat	17/01/2019 15:30	Windows Batch File	2 KB
make_prog_files.tcl	17/01/2019 15:30	TCL File	4 KB

Step 18 – Double click on the **make_prog_files.bat** this will create the bit file needed with the Cortex-M1 and application we just created to download into the Arty S7-50 board.

Step 19 – In Vivado open Hardware Manager:

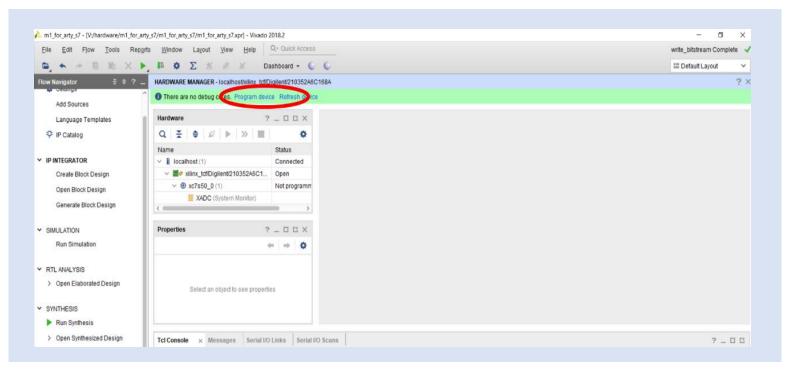
e Edit Flow Tools Repor	rts Window Layout View Help Q- Quick Access		write_bitstream Complete
	(∞ 🖻), 🖩 Φ Σ % 🖉 Χ 🕻 🤇		III Default Layout
v Navigator 🗄 🔍 🗕	BLOCK DESIGN - m1_for_arty_s7		?
Add Sources	Sources Design × Signals Board ? _ 🗆 🖸	Address Editor × Diagram × Project Summary ×	? 🗆
Language Templates	Q ¥ 31 0	Settings Edit	
P: IP Catalog	M m1_for_arty_s7	Project name: m1_for_arty_s7	
P INTEGRATOR	> Interface Connections	Project location: V:/hardware/m1_for_arty_s7/m1_for_arty_s7	
Create Block Design	> Ports > Ports	Product family: Spartan-7	
Open Block Design	> # axi_bram_ctrl_0 (AXI BRAM Controller:4.0)	Project part Arty S7-50 (xc7s50csga324-1) Top module name: m1_for_arty_s7_wrapper	
Generate Block Design	> 👎 axi_gpio_0 (AXI GPI0:2.0)	Target language: Verilog	
		Simulator language: Mixed	
SIMULATION	Source File Properties ? _ Source File Properties		
Run Simulation	← → Q	Board Part	
RTL ANALYSIS		Display name: Arty S7-50	
> Open Elaborated Design	Select an object to see properties	Board part name: digilentinc.com.arty-s7-50:part0:1.0	
		Connectors: No connections	
SYNTHESIS		Repository path: C. Willing Alivado (20.19.2) data (board: filos.	
Run Synthesis	Tcl Console × Messages Log Reports Design Run	5	? _ 🗆
> Open Synthesized Design	Q∰ ♦ II 🖻 🖩 🛍		
		= 00:00:48 . Memory (MB): peak = 1401.281 ; gain = 79.230	
MPLEMENTATION	<pre>file mkdir V:/hardware/ml_for_arty_s7/ml_for_arty write_hwdef -force -file V:/software/ml_for_arty</pre>		
Run Implementation		/sdk_workspace -hwspec V:/software/ml_for_arty_s7_wrapper.hdf	
> Open Implemented Design		/software/ml_for_arty_s7/sdk_workspace -hwspec V:/software/ml_for_arty_s7_wrapper.hdf	
ROGRAM AND DEBUG	INFO: [Vivado 12-3157] SDK launch initiated. Please	ase check console for any further messages.	
Generate Bitstream			
Open Hardware Manager	Type a Tol command here		

Step 20 – Once Hardware Manager is open select **Open Target** and auto connect. This will open the JTAG link with the target board.

m1_for_arty_s7 - [V:/hardware/m1_for_ar			- 5 ×
Eile Edit Flow Iools Repo	rts Window Layout View H	p Q+ Quick Access	write_bitstream Complete
🖻, 🖘 🖉 🗎 🗙 🕨	μο Σ % % Χ	Dashboard 🧯 🌜	🚍 Default Layout 🗸 🗸
Flow Navigator 🗧 🏺 ? 🚊	HARDWARE MANAGER - unconnecto		?
Add Sources	1 No hardware target open. Ope	target	
Language Templates	Hardware	7 _ D G X	
	Q		
P IP Catalog			
IP INTEGRATOR			
Create Block Design	No content		
Open Block Design			
Generate Block Design			
SIMULATION	Source File Properties	? _ 🗆 🖒 ×	
Run Simulation		+ + 0	
RTL ANALYSIS			
> Open Elaborated Design			
7 Open Liaborated Design	Select an object to see	roperties	
SYNTHESIS			
Run Synthesis			

Step 21 – We are now connected to the Arty S7-50 board and can program it. Before we do this we should open a terminal program, such as terra term or PuTTY, so we can see the output from the serial link. **Set it for 115200, 1 Start, 1 Stop no Parity.**

Step 22 –To program the FPGA in Hardware Manager we need to select the correct bit file, select **Program Device**.

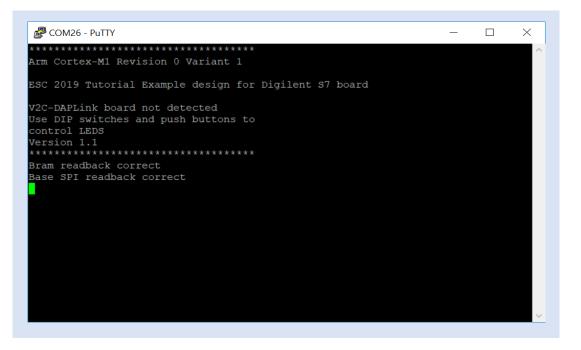


Step 23 – When the dialog appears select the file

V:/hardware/m1_for_arty_s7/m1_for_arty_s7/m1_for_arty_s7.bit

À Program Device	×
	gramming file and download it to your hardware device. You can optionally file that corresponds to the debug cores contained in the bitstream
Bitstre <u>a</u> m file:	V:/hardware/m1_for_arty_s7/m1_for_arty_s7/m1_for_arty_s7.bit 🛛 💿 🚥
Debu <u>a</u> probes file:	
✓ Enable end of s	startup check
?	Program Cancel

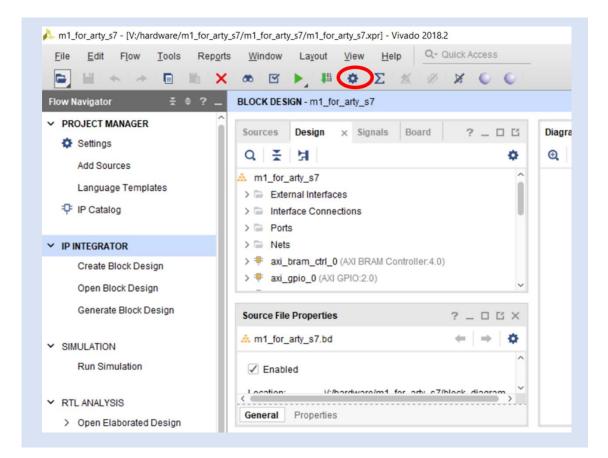
Step 24 – Check the output in your terminal, it should look similar to below:



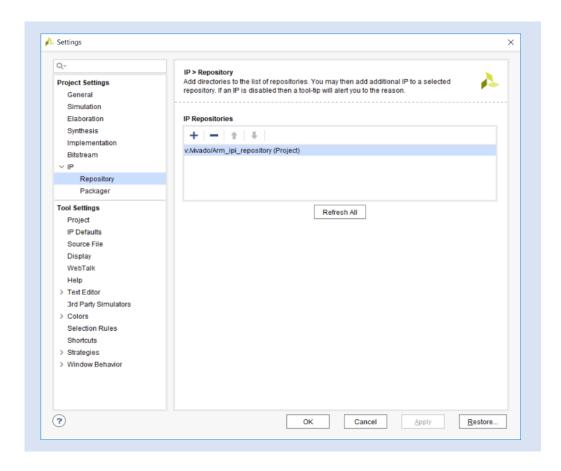
As we build on top of the existing reference design we can also press **BTN1** and **BTN0** on the Arty S7-50 to see LD0 and LD1 cycle through colors with each press.

This concludes Lab 2!

Step 1 – In Vivado we need to add in IP cores from the Digilent IP library. This requires that we first add in the IP repository. To do this click on the **Project Settings**.

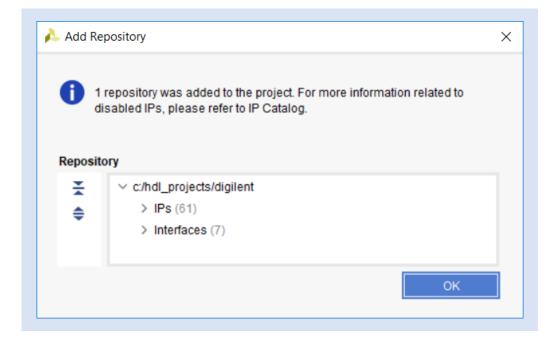


Step 2 – This will open the project settings dialog box. Select $IP \rightarrow Repository$ and click on the + button to add a IP repository.

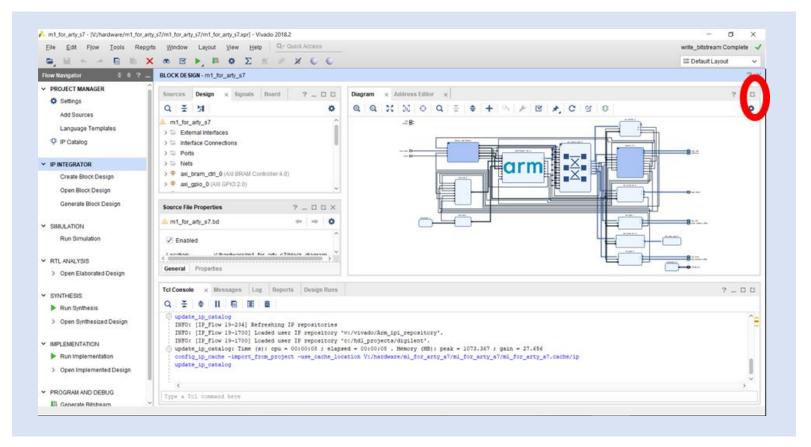


Step 3 – Navigate to the location of the Digilent IP library on your system (downloaded during the Pre-Lab). If you do not have them they can be downloaded from <u>https://github.com/Digilent/vivado-library.</u>

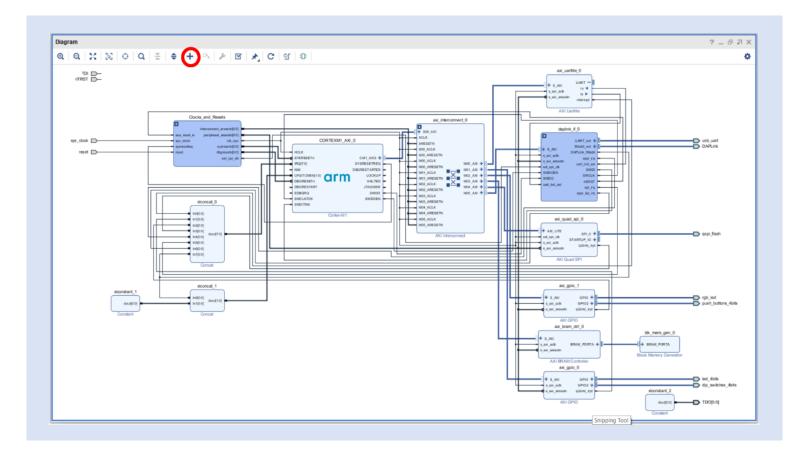
Step 4 – When you see the dialog below click **OK**.



Step 5 – We are now able to begin adding in the IP from this library into our Vivado design. The first stage is to close the IP settings and open the **block diagram**. Float the block diagram so we can make it fit the screen by selecting the button highlighted below.

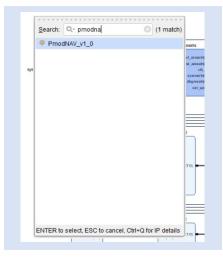


Step 6 – We are now going to add in the **Pmod Nav** and **Pmod HYGRO.** In the block diagram select the **+ button**. This will bring up a dialog which we can use to add in the necessary IP.



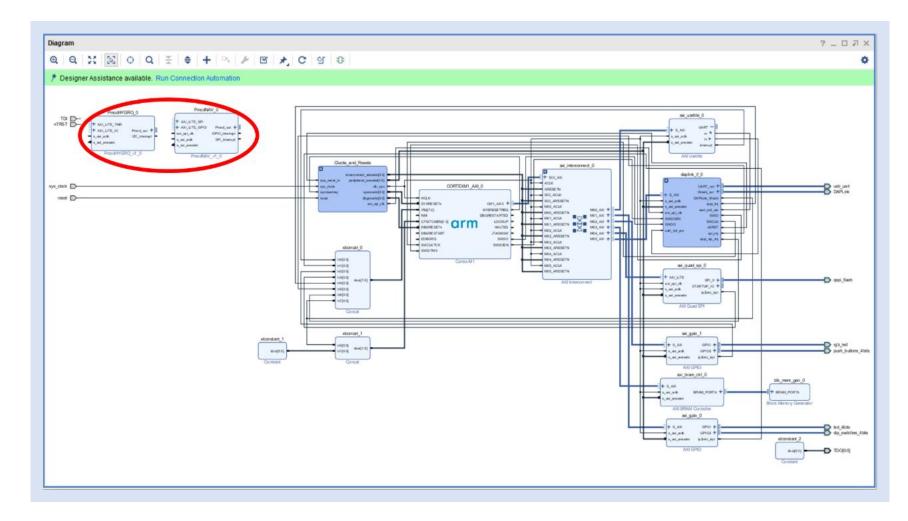
Step 7 – To add **Pmod Nav**, in the search bar type "**pmodnav**" and double click on the IP to add it into the block diagram.

Step 8 – To add in the **Pmod HYGRO**, repeat steps 6-7 by clicking the **+ button** again and in the search bar type **"pmodhygro"** double click on the IP to add it into the block diagram.

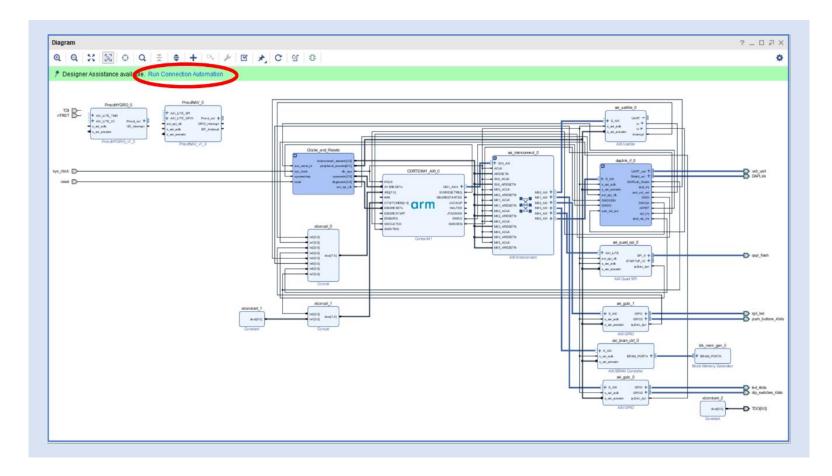




Step 9 – Check that both the IP blocks have now been added in the block diagram.



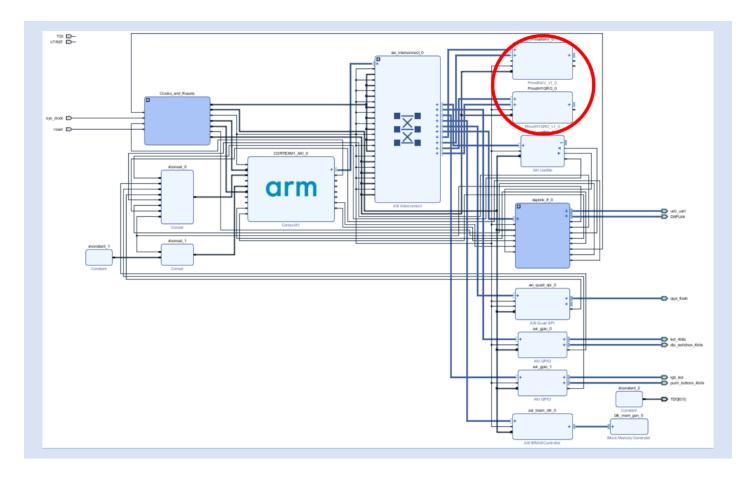
Step 10 – Click on the **Run Connection Automation** option. This will open a dialog to connect in the newly added IP.



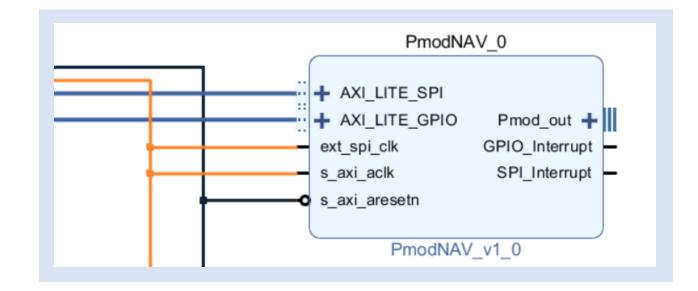
Step 11 – In the dialog check **All Automation** and click **OK**.

Connect Slave interface (/PmodNAV_0/AXI	LITE_SPI) to a selected Master address space.
Options	
Master	/CORTEXM1_AXI_0/CM1_AXI3 V
Bridge IP	/axi_interconnect_0 🗸
Clock source for driving Interconnect IP	/Clocks_and_Resets/clk_wiz_0/clk_out1 (100 MHz) 🗸
Clock source for Master interface	/Clocks_and_Resets/clk_wiz_0/clk_out1 (100 MHz) 🗸
Clock source for Slave interface	Auto ~
	1840
	Options Master Bridge IP Clock source for driving Interconnect IP Clock source for Master interface

Step 12 – You will see the updated block diagram with the **PmodNAV** and **PmodHYGRO** connected into the AXI system and with the Cortex-M1 processor.

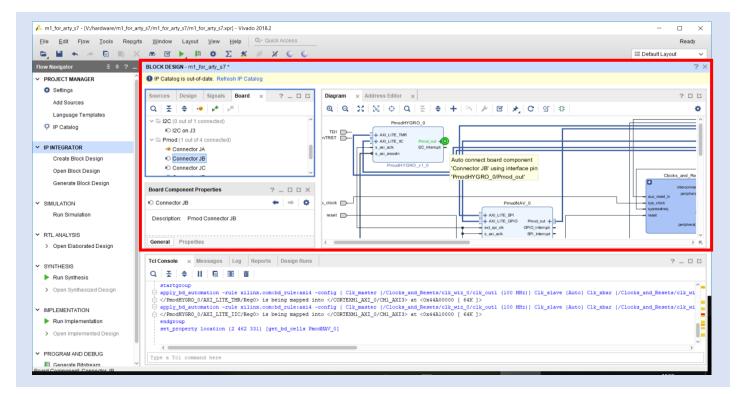


Step 12 (cont'd) – Make sure you connect the ext_spi_clk to the s_axi_clk

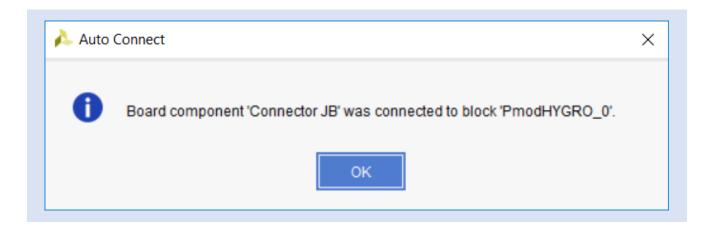


Step 13 – At this point we have not added any IO to these IP blocks connecting them to the Pmod ports on the Arty S7-50.

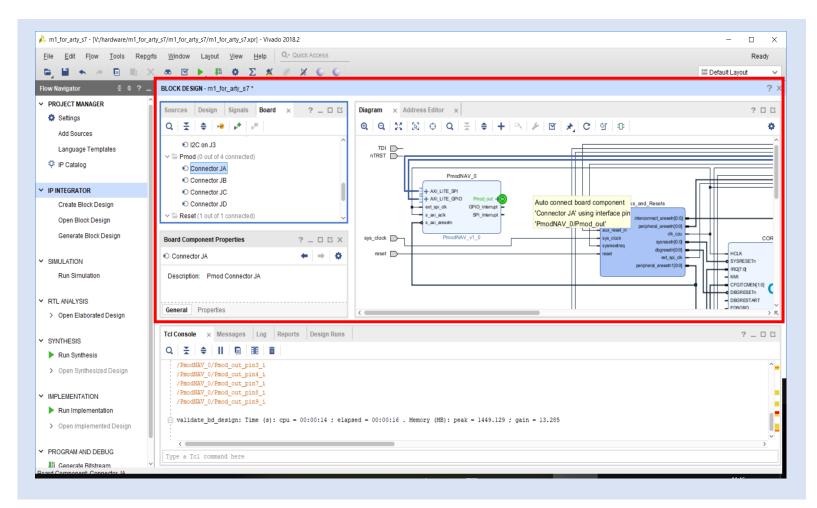
The next thing to do is select the **Connector JB** under the board tab and drag and drop it on the **PmodHYGRO**.



Step 14 – This should show the connection to the PmodNAV correctly.



Step 15 – Repeat the same thing for **PmodNAV** and **Connector JA**.



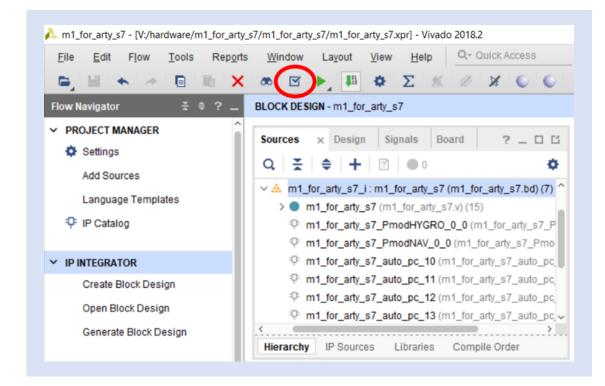
Step 16 – Again, you will see a confirmation of the connection.

🝌 Auto	Connect	×
0	Board component 'Connector JA' was connected to block 'PmodNAV_0'.	
	ОК	

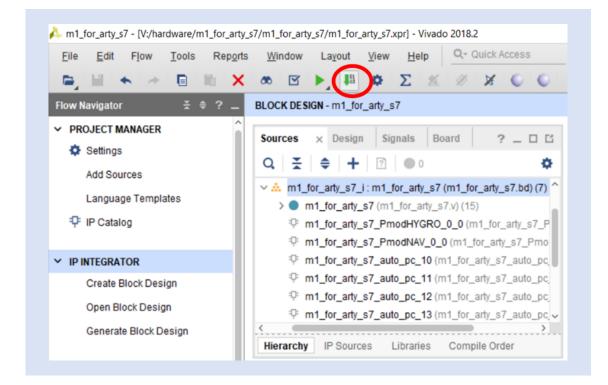
Step 17 – We now check assigned memory addresses. Save and close the block diagram and double check in the **Address Editor** that all of the Pmods are in the **0x4xxx_xxx Range**.

Sources Design Signals Board x ? _ []	Diagram	× AddressEditor ×		Address Editor × IP Catalog ×						
Q 😤 单 📌 🖉	0, 0	10 N O Q ± 0	+ ~ & B *,	Q, X ≑ ₫						
Similar Si		PmodHYGRO_0		Cell	Slave Interface	Base Name	Offset Address	Range		High Address
C 12C on J3 ⇒ Prood (1 out of 4 connected)		+ AXU_LITE_TMR + AXU_LITE_IC Proof and +								
-* Connector JA		s_al_ack DC_Hanut		CM1_AXI3 (32 address bits : 0x0000)	0000 [1M] ,0x4000	0000[512M],0	x60000000 [1G]	0xA0000] 000	1G])
Connector JB		PhodHYGRO_s1_0	Auto connect board comport		S_AXI	Mem0	0x6000_0000	8K	٠	0x6000_1FF
C Connector JC		Printerrund_si_o	'Connector JB' using interface 'PmodHYGRO 0/Pmod out	axi dbio u	S_AXI	Reg	0x4011_0000	64K	•	0x4011_FFF
				🚥 daplink_if_0/axi_gpio_0	S_AXI	Reg	0x4001_0000	64K	*	0x4001_FFF
oard Component Properties ? _ D C ×				📼 axi_gpio_1	S_AXI	Reg	0x4012_0000	64K	•	0x4012_FFF
Connector JB 🔶 🔶 🔿	Lolock D-	·	PmodNA	🚥 axi_quad_spi_0	AXI_LITE	Reg	0x4013_0000	64K	٠	0x4013_FFF
Description: Prood Connector JB	nuset D-		+ AXI_LITE_SPI + AXI_LITE_SPID	daplink_if_0/axi_quad_spi_0	AXI_LITE	Reg	0x4002_0000	64K	•	0x4002_FFE
			est_spi_dk s_asi_ach	daplink_if_0/axi_single_spi_0	AXI_LITE	Reg	0x4003_0000	64K	٠	0x4003_FFF
Seneral Properties	<	1	11	∞ axi_uartlite_0	S_AXI	Reg	0x4010_0000	64K	٠	0x4010_FFF
All and the second seco				aplink_if_0/axi_xip_qued_spi_u	AXI_FULL	MEM0	0x0000_0000	414	٠	0x000F_FFF
clConsole × Messages Log Reports Design Runs				🚥 daplipk_11_0/axi_xip_quad_spi_0	AXI_LITE	Reg	0x4000_0000	64K	*	OX4000 FFF
				PmodNAV_0	AXI_LITE_SPI	Reg0	0x4004_0000	64K	٠	0x4004_FFF
				m PmodNAV_0	AXI_LITE_GPIO	Reg0	0x4005_0000	4K	•	0x4005_0FF
				m PmodHYGRO_0	AXI_LITE_TMR	Reg0	0x44A0_0000	64K	*	0x44A0_FFF
				PmodHYGRO_0	AXI_LITE_IIC	Reg0	0x44A1_0000	64K	*	0x44A1_FFE

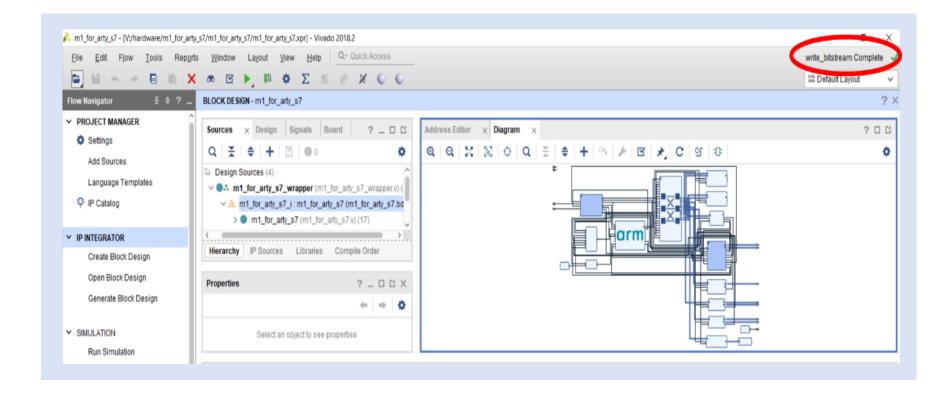
Step 18 – We are now ready to build the design. Click on the **validate button** to prove we have no errors or critical warnings in the design.



Step 19 – Click on the generate bit stream button.



Step 20 – This may take a little while but once the bit stream is completed you will see the message write_bitstream_complete in the upper right hand corner of Vivado.



Step 21 – Now we need to update the MMI file. To do this we need to open the implemented design.

There is a script we need to run to generate this, available under:

V:\hardware\m1_for_arty_s7\m1_for_arty_s7

In the **TCL console**, change the working directory to the one above using:

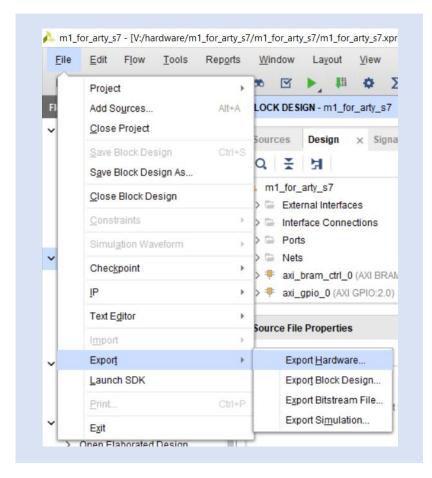
cd V:\\hardware\\m1_for_arty_s7\\m1_for_arty_s7

Then enter the command source make_mmi_file.tcl

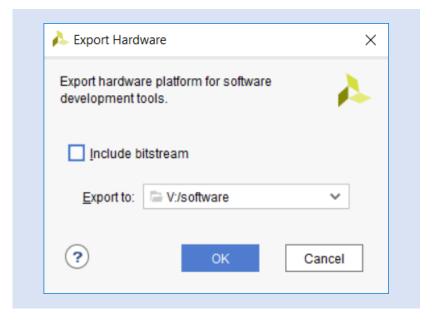
This will create the updated MMI file.

Note, if you see any warning when you open the implemented design, click on **OK**.

Step 22 – Next, export the hardware definition. Select **File** \rightarrow **Export** \rightarrow **Export Hardware**.



Step 23 – In the pop up, set the export location as V:\software



Step 24 – In the pop up, set the export location as **V:\software.**

If you see a warning asking if you want to overwrite the existing file, click **Yes**.

🝌 Export Hardy	ware	×
Export hardwar development to	re platform for software pols.	4
Include b	itstream	
<u>E</u> xport to:	🗎 V:/software	~
?	ОК	Cancel

I Controller:4.0)	🔥 Export Ha	rdware		× ^{a324-1)} per	
	Export bardy	ware platform for so	offware		
🝌 Module	Already Exported				×
A (2)	An exported file for this			, ea	
to see pr		<u>Y</u> es	No		
to see pr	(3)	Yes OK	No Cance	lta/baarda/ba	and files

Step 25 – Next, we need to open XDSK. Select **File** \rightarrow Launch SDK.

<u>F</u> ile	<u>E</u> dit Flow <u>T</u> ools	Rep <u>o</u> rts	Window Layout
ſ	Project	•	× 🗹 🕨 👫
FI	Add Sources	Alt+A	LOCK DE SIGN - m1_f
~	Close Project		
	Save Block Design	Ctrl+S	Sources Design
	Save Block Design As.		Q 포 네
	<u>C</u> lose Block Design		m1_for_arty_s7 External Interfa
	Constraints	Þ	> Interface Conn
	Simulation Waveform	Þ	> 🗁 Ports
~	Checkpoint	•	> 🚍 Nets
	IP	•	# axi_bram_ctrl_ # axi_gpio_0 (AX)
	-		> + axi_gpio_0 (~
	Text E <u>d</u> itor	•	Source File Propertie
	Import	+	
~	Export	×	
	Launch SDK		
	Print	Ctrl+P	Sele
~	Exit		1

Step 26 – In the pop up box, for the export location, select the location we just exported the hardware definition to **(V:\Software)**.

Select the workspace located at V:\software\m1_for_arty_s7\sdk_workspace.

Note: Use **File Explorer** to confirm there is only one exported hardware definition under V:\software. If there is a HDF file for the Arty A7, please delete it.

×

Step 27 – In SDK you will see the hardware definition and BSP being updated from what we used in Lab 2. Under the BSP delete the directory **cortexm1_0**.

In the hardware definition you will see the **PmodHYGRO** and **PmodNAV**.

In the BSP MSS file you will also see the drivers have been included with the BSP.

arget FPGA Device: 7s50				
Part: xc7s50cs	ga324-1			
Created With: Vivado 2	018.2			
Created On: Thu Apr	18 16:21:09 201	9		
deleges Mars for several CC	DTEVA4 AVI	•		
ddress Map for processor CC	RTEXIVIT_AXI	U		
Cell	Base Addr	High Addr	Slave I/f	Mem/Reg
daplink_if_0_axi_single_spi_0	0x40030000	0x4003ffff	AXI LITE	REGISTER
daplink_if_0_axi_gpio_0	0x40010000	0x4001ffff	S_AXI	REGISTER
axi_gpio_0	0x40110000	0x4011ffff	S_AXI	REGISTER
axi_bram_ctrl_0	0x60000000	0x60001fff	S_AXI	MEMORY
axi_quad_spi_0	0x40130000	0x4013ffff	AXI_LITE	REGISTER
axi_gpio_1	0x40120000	0x4012ffff	S_AXI	REGISTER
PmodHYGRO_0	0x44a10000	0x44a1ffff	AXI_LITE_IIC	REGISTER
daplink_if_0_axi_xip_quad_s	0x40000000	0x4000ffff	AXI_LITE	REGISTER
daplink_if_0_axi_xip_quad_s	0x0000000	0x000fffff	AXI_FULL	REGISTER
PmodNAV_0	0x40050000	0x40050fff	AXI_LITE_G	REGISTER
axi_uartlite_0	0x40100000	0x4010ffff	S_AXI	REGISTER
PmodHYGRO_0	0x44a00000	0x44a0ffff	AXI_LITE_T	REGISTER
daplink_if_0_axi_quad_spi_0	0x40020000	0x4002ffff	AXI_LITE	REGISTER
PmodNAV 0	0x40040000	0x4004ffff	AXI LITE SPI	REGISTER

Peripheral Drivers	
Drivers present in the Board Support	t Package.
PmodHYGRO_0 Pmo	odHYGRO
PmodNAV_0 Pmo	odNAV
axi_bram_ctrl_0 bra	m <u>Documentation</u> Import Examples
axi_gpio_0 gpi	o <u>Documentation</u> Import Examples
axi_gpio_1 gpi	o <u>Documentation</u> Import Examples
axi_quad_spi_0 spi	Documentation Import Examples
axi_uartlite_0 uart	tlite Documentation Import Examples
daplink_if_0_axi_gpio_0 gpi	o <u>Documentation</u> Import Examples
daplink_if_0_axi_quad_spi_0 spi	Documentation Import Examples
daplink_if_0_axi_single_spi_0 spi	Documentation Import Examples
daplink_if_0_axi_xip_quad_spi_0 spi	Documentation Import Examples

Step 28 – Next, we need to copy the files **Xpsuedo_asm_rvct.c** and **Xpseudo_asm_rvct.h**

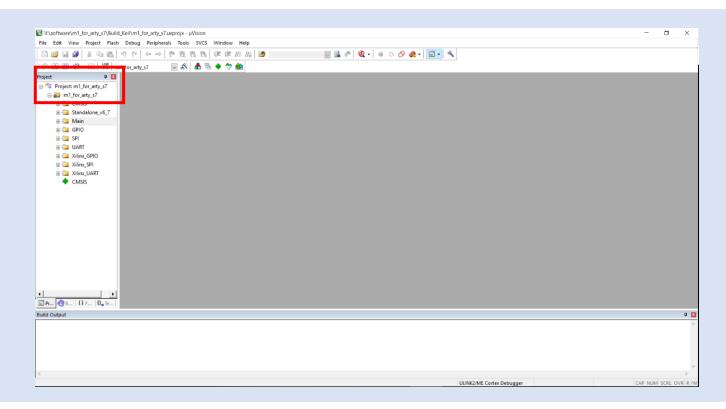
From V:\vivado\Arm_sw_repository\CortexM\bsp\standalone_v6_7\src\arm\cortexm1\armcc to V:\software\m1_for_arty_s7\sdk_workspace\standalone_bsp_0\CORTEXM1_AXI_0\include

Step 29 – Check the name of the Cortex-M processor generated by SDK. Use File Explorer and navigate to V:\software\m1_for_arty_s7\sdk_workspace\standalone_bsp_0.

Ensure the folder is named **CORTEX_M1_0** and not **CORTEXM1_AXI_0** or similar, if so correct the naming to **CORTEX_M1_0**.

Step 30 – Now, let's develop our SW application in Arm KEIL. Navigate to V:/software/m1_for_arty_s7/Build_Keil/. Click on the file m1_for_arty_s7.uvprojx. This will open the Arm Keil project.

Make sure the target is set to m1_for_arty_s7.



Lab 3

Step 31 – The next thing we need to do is ensure the compiler directives are set correctly for both our application and the drivers we will be using. To open the options, click on the **highlighted button**.

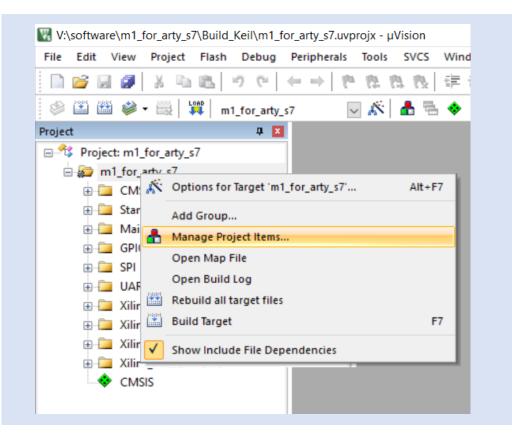
Ensure Optimization is set to **Level-1 (O-1), C99** and **GNU extensions** are enabled, if you want you can also select the optimization level.

File Edit V	iew Proje	ct Flash	Debug	Peripherals	Tools	SVCS	Window	Help
🗋 💕 🗔	🍠 🐰	6 B	2 64	$\leftrightarrow \Rightarrow \uparrow^{p}$	12.1	19.1		//≣ // _R
🖉 🏥 🕮	🧼 • 📖	斗 m1	_for_arty_	\$7	×	📥 🕾	۰ 🔶	
Project			д 🗙		\smile			
🖃 🍄 Project	m1_for_art	y_s7						
🖮 🔛 m1	_for_arty_s7							
Đ - 🚞	CMSIS							
🕢 🚞	Standalone	_v6_7						
Đ - 🚞	Main							
Đ - 🚞	GPIO							
Đ- 🚞	SPI							
Đ	UART							
Đ 🚞	Xilinx_GPIO							
Đ - 🧰	Xilinx_SPI							
· - 🧰	Xilinx UAR	r						

Options for Target 'm1_for_arty_s7'	×
Device Target Output Listing User C/C++ Asm Linker Debug Utilities	
Preprocessor Symbols Define: CORTEX_M1 ARTY_CM1 Undefine:	1
Language / Code Generation Execute-only Code Optimization: Level 1 (O1) Coptimization: Plain Charis Signed No Auto Includes No Auto Includes Split Load and Store Multiple Read-Only Position Independent One ELF Section per Function Read-Write Position Independent Paths cmsis\CMSIS\Include:\cmsis\Device\Include:\cmsis\Device\Include\ARTY_CM1;\sdk_works Misc Controls	
Compiler control string	
OK Cancel Defaults Help	

Step 32 – To work with the **PmodHYGRO** and the **PmodNAV** we need to add in the source code.

To do this we right click on the project and select Manage Project Items.



Step 33 – Under **Groups** we can add in the **PModHYGRO** and the **PModNAV** as new groups.

Then add in the source code which can be found under the **PmodXXXX drivers directory.**

Find the drivers under:

V:\software\m1_for_arty_s7\sdk_workspace\ standalone_bsp_0\CORTEX_M1_0\libsrc

For the PmodNAV do not include the SPI source files. These are already included under the Xilinx_SPI group

Manage Project Items			×
Project Items Folders/Extensions Books		Files: PmodHYGRO.c xiic.c xiic_dyn_master.c xiic_intr.c xiic_matter.c xiic_multi_master.c xiic_options.c xiic_slave.c xiic_slave.c xiic_stats.c xtmrctr_i.c xtmrctr_l.c xtmrctr_options.c xtmrctr_sinit.c xtmrctr_stats.c	× + +
<u>S</u> et as Current Target	,	, <u>A</u> dd Files	
	OK Cancel		Help

Step 34 – Open the **PmodNAV.c** and comment out usleep() calls.

Open **xtmrctr_sinit.c** under the Xilinx PmodHYGRO and edit the line to be *XTmrCtr_Config XTmrCtr_ConfigTable[1]*;

```
xiic_l.c distriction and a state with a state of the stat
main.c
                                                                                                                                                                                                                                                                      xtmrctr_sinit.c
                       #define XPAR XTMRCTR NUM INSTANCES 0
58
                       #endif
59
60
                                                                                                             ************ Variable Declarations *******************************/
61
62
63 = /* A table of configuration structures containing the configuration information
                          * for each timer core in the system. */
64 -
65
                            XTmrCtr Config XTmrCtr ConfigTable[1];
66
                                                                                                                                              ******* Function Definitions *******
67
68
69
                                                                                                                                               ***********
                                                                                                                                                                                                                                                                                          **********************************
```

Lab 3

Step 35 – We can then use the Pmods and create applications. In this example, let's create a simple application which reads the Pmods and Temperature, Humidity, X,Y & Z information over the terminal by adding the following into our code in main.c:

#include "PmodHYGRO.h"
#include "PmodNAV.h"
#define TIMER_FREQ_HZ 10000000
PmodHYGRO myDevice;
PmodNAV nav;
bool sample;
float temp_degc, hum_perrh, temp_degf;

The above should be declared as global variables

Step 36 – To provide a timing reference you might want to use the system tick. This can be accessed by

#define STCTRL (*((volatile unsigned long *) 0xE000E010))
#define STRELOAD (*((volatile unsigned long *) 0xE000E014))
#define STCURR (*((volatile unsigned long *) 0xE000E018))
#define SBIT_ENABLE 0
#define SBIT_TICKINT 1
#define SBIT_CLKSOURCE 2
#define RELOAD_VALUE 98999999

The ISR can be accessed using

```
void SysTick_Handler(void)
{
  sample=TRUE;
}
```

This is already mapped into the vector. Check out **startup_arty_cm1.s**.

Step 37 – In the main loop add in the following code

HYGRO_begin(&myDevice,XPAR_PMODHYGRO_0_AXI_LITE_IIC_BASEADDR,0x40,XPAR_PMODHYGRO_0_AXI_LITE_TMR_BASEADDR,1,TIMER_FREQ_HZ);

```
NAV begin
(&nav,XPAR_PMODNAV_0_AXI_LITE_GPIO_BASEADDR,XPAR_PMODNAV_0_AXI_LITE_SPI_BASEADDR);
NAV Init(&nav);
STRELOAD = RELOAD VALUE:
STCTRL = (1<<SBIT_ENABLE) | (1<<SBIT_TICKINT) | (1<<SBIT_CLKSOURCE);
 while (1)
  if (sample == TRUE){
    temp_degc = HYGRO_getTemperature(&myDevice);
          hum perrh = HYGRO getHumidity(&myDevice);
          sprintf (debugStr, "Temp is %f Humidity is %f\r\n\n", temp_degc, hum_perrh );
          print ( debugStr );
          NAV_GetData(&nav);
    sprintf (debugStr, "X is %f Y is %f Z is %f\r\n\n", nav.acclData.X, nav.acclData.Y, nav.acclData.Z);
          print ( debugStr );
```

```
sample = FALSE;
```

}

Step 38 – The expected output should produce something like below

COM26 - PuTTY -	×
Temp is 24.003906 Humidity is 34.295654	^
X is 0.148779 Y is -0.164944 Z is	
Temp is 23.984375 Humidity is 34.295654	
X is 0.148535 Y is -0.164212 Z is	
Temp is 23.984375 Humidity is 34.295654	
X is 0.148901 Y is -0.164090 Z is	
Temp is 24.013672 Humidity is 34.295654	
X is 0.148047 Y is -0.163541 Z is	
Temp is 23.984375 Humidity is 34.295654	
X is 0.148657 Y is -0.164151 Z is	
Temp is 24.003906 Humidity is 34.197998	
	~

Step 39 – Now, let's test our application on the hardware.

Use a file browser navigate to V:\hardware\m1_for_arty_s7\m1_for_arty_s7.

You will see a file named **bram_s7.elf** and **bram_s7.hex.** These should have today's time stamp as you just generated them.

Name	Date modified	Туре	Size
m1_for_arty_s7.cache	18/04/2019 13:35	File folder	
m1_for_arty_s7.hw	18/04/2019 13:35	File folder	
m1_for_arty_s7.runs	18/04/2019 13:49	File folder	
m1_for_arty_s7.sdk	18/04/2019 14:12	File folder	
m1_for_arty_s7.sim	18/04/2019 13:35	File folder	
bram_s7.elf	18/04/2019 14:49	ELF File	137 Ki
bram_s7.hex	18/04/2019 14:49	HEX File	29 KE
🗋 m1.mmi	13/12/2018 15:15	MMI File	3 KI
🍌 m1_for_arty_s7.xpr	18/04/2019 14:02	Vivado Project File	19 KE
m1_for_arty_s7_reference.bit	13/12/2018 15:15	BIT File	2,141 KE
m1_for_arty_s7_reference.mcs	13/12/2018 15:15	MCS File	6,022 KE
😵 make_mmi_file.tcl	17/01/2019 15:30	TCL File	8 KE
make_prog_files.bat	17/01/2019 15:30	Windows Batch File	2 KE
make_prog_files.tcl	17/01/2019 15:30	TCL File	4 KE

Step 40 – Double click on make_prog_files.bat

This will create the bit file needed with the Cortex-M1 and application we just created to download into the Arty S7-50 board.

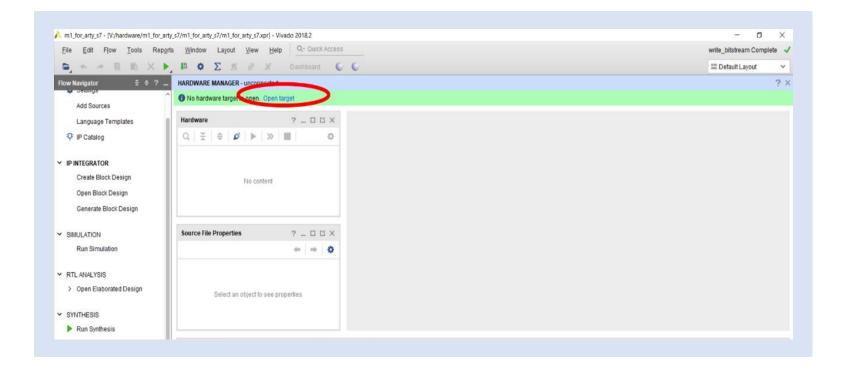
Step 41 – Power off the Arty S7 board and connect the PmodNAV to PmodA and PmodHYGRO to PmodB.



Step 42 – Power on the Arty Board and In Vivado open the **Hardware Manager**.

	_arty_s7/m1_for_arty_s7/m1_for_arty_s7.xpr] - Vivado 2018.2		- 0
	ports Window Layout View Help Qr Quick Access		write_bitstream Complete
「国を子田田	× ∞ 🖻 🕨 🏙 🏟 Σ 🕷 🖉 🖉 🤅		III Default Layout
low Navigator 😤 🔍 ?	BLOCK DESIGN - m1_for_arty_s7		
Add Sources	Sources Design × Signals Board ? _ C	Address Editor × Diagram × Project Summary ×	? 🗆
Language Templates	Q X X Q	Settings Edit	
P IP Catalog	M m1_for_arty_s7	Projectname: m1_for_arty_s7	
IP INTEGRATOR	> Connections	Project location: V:/hardware/m1_for_arty_s7/m1_for_arty_s7	
Create Block Design	> Ports	Product family: Spartan-7	
Open Block Design	Nets # axi bram.ctrl 0 (AXI BRAM Controller:4.0)	Project part: Arty 37-50 (xc7s50csga324-1)	
Generate Block Design	> ♥ axi_gpio_0 (AXI GPI0:2.0)	raigerianguage. Veriog	
SIMULATION	Source File Properties ? _ 🗆 🗅 🗠 🗙		
Run Simulation	+ + 0	Board Part	
RTL ANALYSIS		Display name: Arty S7-50	
> Open Elaborated Design	Select an object to see properties	Board part name: digilentinc.com:arty-s7-50;part0:1.0	
		Connectors: No connections	
SYNTHESIS		Repacitory path: C-Willow/Aliade/2019.2/data/beards/beard_files.	
Run Synthesis	Tcl Console × Messages Log Reports Design Run	5	? _ 🗆
> Open Synthesized Design	Q		
MPLEMENTATION	launch_runs: Time (s): cpu = 00:00:32 ; elapsed = ; file mkdir V:/hardware/ml for arty s7/ml for arty	<pre>00:00:48 . Memory (MB): peak = 1401.281 ; gain = 79.230 v 37/ml for arty 37 mlk</pre>	
Run Implementation	write_hwdef -force -file V:/software/ml_for_arty	r_s7_wrapper.hdf	
> Open Implemented Design	Iaunch_sdk -workspace V:/software/ml_for_arty_s7/ INFO: [Vivado 12-393] Launching SDK	<pre>'sdk_workspace -hwspec V:/software/ml_for_arty_s7_wrapper.hdf</pre>	
	INFO: [Vivado 12-417] Running xsdk -workspace V:/	<pre>/software/ml_for_arty_s7/sdk_workspace -hwspec V:/software/ml_for_arty_s7_wrapper.hdf /so check_console_for_ary_further_massaces</pre>	
PROGRAM AND DEBUG	i into, [vivado iz-olor] ook inunci interaced, rice	neer condore for and forenet messades.	
III Generate Bitstream	1		>
> Open Hardware Manager	Type a Tcl command here		

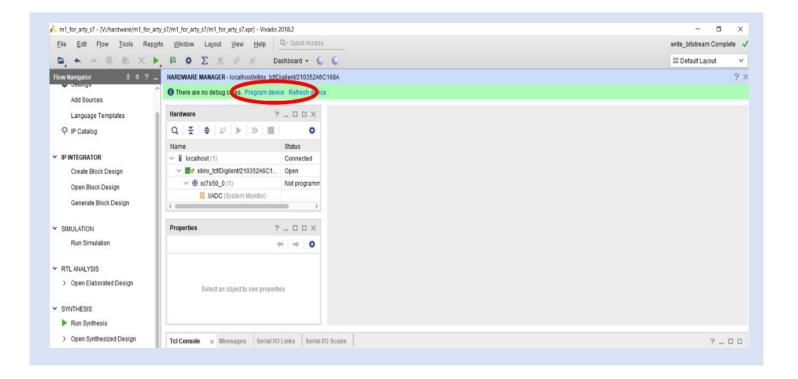
Step 43 – Once Hardware Manager is open, select **Open Target** and auto connect. This will open the JTAG link with the target board.



Step 44 – This will connect to the Arty S7-50 so that we can program it. Before we do this we should open a terminal program (such as terra term or PuTTY) so that we can see the output from the serial link.

Set it for 115200, 1 Start, 1 Stop no Parity.

Step 45 – To program the FPGA in Hardware Manager we need to select the correct bit file. Select **Program Device**.



Step 46 – When the dialog appears, select the file V:/hardware/m1_for_arty_s7/m1_for_arty_s7/m1_for_arty_s7.bit

	amming file and download it to your hardware device ile that corresponds to the debug cores contained in		
Bitstre <u>a</u> m file:	V:/hardware/m1_for_arty_s7/m1_for_arty_s7/m1_fo	r_arty_s7.bit 🛛 😒]
Debug probes file:]
✓ Enable end of st	artup check		
(?)	Р	rogram Cancel	

Step 47 – Check the output in your terminal.

Congratulations! You have finished Lab 3 and completed your simple application.