

CoDeveloper Platform Support Package Pico Computing M501 PSP User Guide – Linux Version 1.1.1

Impulse Accelerated Technologies, Inc.

www.ImpulseAccelerated.com

1.0 Table of Contents

1.0	TABLE OF CONTENTS	2
2.0	TABLE OF FIGURES	4
3.0	REVISION HISTORY	6
4.0	OVERVIEW	7
4.1. 4.	Hardware Block Diagram 1.1. Pico Computing Block Diagram	
5.0	BEFORE GETTING STARTED: READ THIS FIRST	9
5.1. 5.2. 5.3. 5.4.	Required Software Tools: Additional Required Files	9 10
6.0	HOST SYSTEM SETUP (LINUX)	11
6.3.	Running ISE on Linux	11 11 11
7.0	DEVELOPMENT SYSTEM SETUP (WINDOWS)	12
7.		12 12 12
8.0	PASSTHROUGH EXAMPLE AND TUTORIAL	13
	CoDeveloper Project Files Opening Project Building Desktop Simulation Executable Running Desktop Simulation Executable Project Setup Before Hardware/Software Generation and Export Generating Hardware Exporting Hardware Compiling FPGA in Xilinx ISE 13.4 9.1. Building bitfile under Windows 9.2. Building bitfile under Linux 0. Exporting Software 1. Programming the FPGA	15 17 18 21 22 24 26 27 33 39 41
9.0	MEMTEST EXAMPLE AND TUTORIAL	44
9.1. 9.2. 9.	I	45

9.2.	2. Overview of memtest_hw.c	45
9.2.	3. Overview of memtest_sw.c	47
9.3.	CoDeveloper Project Files	50
9.4.	Opening Project	
9.5.	Building Desktop Simulation Executable	53
9.6.	Running Desktop Simulation Executable	
9.7.	Project Setup Before Hardware/Software Generation and Export	55
9.8.	Generating Hardware	56
9.9.	Exporting Hardware	58
9.10.	Compiling FPGA in Xilinx ISE 13.4	
9.10	0.1. Building bitfile under Windows	
9.10	0.2. Building bitfile under Linux	63
9.11.	Exporting Software	64
9.12.	Programming the FPGA	
9.13.	Running Target Executable on the Host System	

2.0 Table Of Figures

Figure 1 – Firmware Architecture (Pico M50X Series Platform Support Package Users Guide	\ Q
Figure 2 - Impulse C Header File with 64 bit co_stream	
Figure 3 - ImpulseC Hardware File with 64 bit co_stream	
Figure 4 - Opening a project in CoDeveloper	
Figure 5 - Build Simulation Desktop in CoDeveloper using pull-down menu	
Figure 6 - Build Simulation Desktop in CoDeveloper using toolbar icon	
Figure 7 - Output within the CoDeveloper IDE build window	
Figure 8 - Launch software simulation window using pull-down menu	
Figure 9 - Launch software simulation using toolbar icon	
Figure 10 - Pop-up window during desktop simulation	
Figure 11 - Project setup to pick Platform Support Package	
Figure 12 - Generate HDL using pull-down menu	
Figure 13 - Generate HDL using toolbar icon	
Figure 14 - Build window output	
Figure 15 - Export Generated Hardware (HDL) using pull-down menu	
Figure 16 - Export Generated Hardware (HDL) using toolbar icon	
Figure 17 - Build window output	
Figure 18 - Compiling FPGA in exported ISE directory structure	
Figure 19 - Generate ISE project files	27
Figure 20 - Expected gen50x_xise.log report	
Figure 21 - Initial Xilinx ISE GUI screen	
Figure 22 - Xilinx ISE regenerate IP core fifo128x512	29
Figure 23 - Xilinx ISE regenerate IP Core coregen_fifo_32x128	29
Figure 24 - Xilinx ISE 13.4 with timing score = 0	30
Figure 25 - Build bitfile in ISE 13.4	
Figure 26 – Xilinx ISE 13.4 compile log file – timing score	32
Figure 27 - Generate ISE project files	
Figure 28 - Expected gen50x_xise.log report	33
Figure 29 - Select the ISE project	34
Figure 30 - Initial project status after loading project	35
Figure 31 - Regenerate fifo128x512	
Figure 32 - Regenerate coregen_fifo_32x128	36
Figure 33 - Place and Route and bitfile generation with timing score equal to zero	
Figure 34 - Ubuntu Xilinx ISE 13.4 Command Line output	
Figure 35 - Xilinx ISE 13.4 results file with timing score equal to zero	38
Figure 36 - Export Generated Software	39
Figure 37 - Build window output	
Figure 38 - Exported software directory	
Figure 39 - Files & Directories to be copied to the Host System	
Figure 40 - Exported SW executed on target platforn	
Figure 41 - memtest.h	
Figure 42 - User defined funtion in memtest_hw.c	
Figure 43 - Configuration in memtest_hw.c	
Figure 44 - User defined stimulus in memtest_sw.c	
Figure 45 - Main program in memtest_sw.c	
Figure 46 - Impulse C Header File	
Figure 47 - ImpulseC Hardware File	
Figure 48 - Opening a project in CoDeveloper	
Figure 49 - Build Simulation Desktop in CoDeveloper using pull-down menu	

Figure 50 - Build Simulation Desktop in CoDeveloper using toolbar icon	53
Figure 51 - Output within the CoDeveloper IDE build window	53
Figure 52 - Launch software simulation window using pull-down menu	
Figure 53 - Launch software simulation using toolbar icon	
Figure 54 - Pop-up window during desktop simulation	54
Figure 55 - Project setup to pick Platform Support Package	
Figure 56 - Generate HDL using pull-down menu	
Figure 57 - Generate HDL using toolbar icon	56
Figure 58 - Build window output	57
Figure 59 - Export Generated Hardware (HDL) using pull-down menu	
Figure 60 - Export Generated Hardware (HDL) using toolbar icon	58
Figure 61 - Build window output	58
Figure 62 - Compiling FPGA in Quartus directory structure	59
Figure 63 - Generate ISE project files	60
Figure 64 - Expected Gen_Ise_File log report	60
Figure 65 - Initial Xilinx ISE 13.4 GUI screen	61
Figure 66 - Xilinx ISE 13.4 regenerate IP core fifo128x512com	61
Figure 67 - Xilinx ISE 13.4 regenerate IP Core fifo128x512	62
Figure 68 - Xilinx ISE 13.4 regenerate IP Core coregen_fifo_32x128	62
Figure 69 - Xilinx ISE 13.4 with timing score = 0	62
Figure 70 - Export Generated Software	64
Figure 71 - Build window output	64
Figure 72 - Exported software directory	65
Figure 73 - Files & Directories to be copied to the Host System	67
Figure 74 - Exported SW executed on target platforn	68

Date	Version	Description	Author
11/3/2012	1.0	Initial Creation	Shaumil Dave
11/10/2012	1.1	Added Memtest description	Shaumil Dave
2/5/2013	1.1.1	Added no-EOS support to limitations, EOS has never been supported	Ed Trexel

3.0 Revision History

4.0 Overview

This user guide covers the CoDeveloper Platform Support Package (PSP) for the Pico Computing M501 module (referred hereon simply as "M501"). Highlights for this PSP include:

- Automatic creation of a complete ready-to-build Xilinx ISE project upon exporting hardware for creating FPGA binary '.bit' file referred to as the "bitfile". The bitfile may be built under Windows or Linux and either via the ISE GUI or from command line using a build script.
- Application executable built via Makefile on a Linux platform.
- Loading of the FPGA on the M501 via host CPU over PCIe.

4.1. Hardware Block Diagram

4.1.1. Pico Computing Block Diagram

Below is a diagram of the M50X platform:

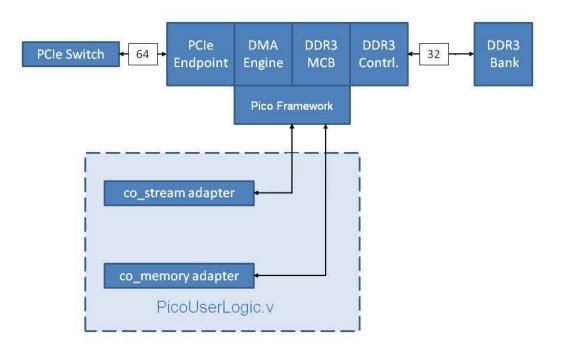


Figure 1 – Firmware Architecture (Pico M50X Series Platform Support Package Users Guide)

The only elements of the architecture the user has control of are those contained inside the Verilog module "PicoUserLogic.v". All of the logic generated by Impulse CoDeveloper and any external HDL modules must be instantiated inside this top level user module

5.0 Before Getting Started: Read This First

Before getting started, please ensure that you have obtained and installed all the necessary software tools, additional files, and hardware as described below.

5.1. Hardware limitations:

- 1. co_streams require all data transfers to be a multiple of 128-bits/16bytes
- co_streams to/from host do not support End-Of-Stream (EOS). Using co_stream_close() on a stream to/from host will either cause extra data being sent to the host or the hardware process to lockup.
- 3. co_memory (not used in Passthrough example) use limitations:
 - a. The memory interface requires that memory transfers begin at an address with 256-byte alignment
 - b. The memory interface assumes that the transfer size in bytes is a multiple of 16

5.2. Required Software Tools:

- Impulse CoDeveloper v3.70.e.6 or newer for the Development System.
- NOTE: The use of floating point for Virtex-6 and newer devices requires the use of Xilinx's v5.0 CORE Generator cores which is supported via a patch to CoDeveloper made available via the 'XilinxFPv5BetaPatch' link under the supplied Pico PSP link.
- Xilinx ISE 13.4 for Windows (exactly, not newer) for the Development System.
- Windows 7 for Impulse CoDeveloper and Xilinx ISE 13.4 (the Development System).
- Ubuntu (Debian) 12.04 LTS for the Host System (With EX500 PCIe card and either M501 or M503 FPGA modules)
- Pico Linux installer package version 5.0.6.1 or newer for the Host System:
 - o Go to http://picocomputing.com/support/software
 - Select Linux Installer (M-501, M-503, M-505)
 - Linux_5.0.6.1_all.deb

5.3. Additional Required Files

The following Examples files are not included with the installation of the software tools and are required for development using this PSP. They include the CoDeveloper project file (*.icProj), associated design files (*_hw.c, *_sw.c, *.h), and data files (*.dat) used as stimulus. A link to download the M5xx PSP and Examples files should have already been provided, if not please email support@impulsec.com to request one.

Examples

M5XX

Passthrough

filter_in.dat passthrough.icProj passthrough_hw.c passthrough_sw.c passthrough.h

CoDeveloper will use the project file, "passthrough.icProj", to generate HDL as well as a software application to load the FPGA image.

5.4. Required Hardware

The following hardware is required for development using this PSP:

- EX500 x16 full length full height PCIe FPGA Development Board.
- M501 or M503 FPGA module.
- Host System with Ubuntu 12.04 LTS 64-bit OS based development PC for running all tools – Recommended: 100GB disk space available for tools installation and 12GB RAM.
- Host System that has an Intel Motherboard with x58 chipset and available x16 (physical) Gen 2 PCIe express slot. Consideration for your PCIe video card must be given if it is a x16 video card. The motherboard must accommodate (not a shared resource) independent x16 Gen 2 PCIe slots.
- Available 12 volt PCIe power supply for the M501 FPGA development board in the Host System. Consideration for your video card must be given if it also requires a separate 12 volt power connection.
- Development System for installation of CoDeveloper and Xilinx ISE 13.4.
 - Impulse CoDeveloper and M50x PSP: Windows only (32 or 64-bit)
 - Xilinx ISE 13.4: Windows 64-bit or Linux 64-bit

6.0 Host System Setup (Linux)

6.1. Install Pico M501 driver

After the Pico M501 and EX-xxx carrier are installed, the user must install the drivers.

 After downloading the Pico Linux installer package (Linux_*.deb), follow the "Linux_PicoGettingStarted.pdf" to install Linux software, firmware, and the driver for the M50x.

6.2. Running ISE on Linux

6.2.1. Install Xilinx ISE 13.4

- 1. Please see vendor supplied documentation for installation.
- 2. Please see vendor supplied documentation for licensing.

6.3. Running ISE on Windows

6.3.1. Copy Pico Installer source from Host PC to Development PC

If Xilinx ISE will be run on Windows, prepare to copy the directory "/usr/src/picocomputing-5.0.6.1" from the Linux Host PC to the Windows Development system. For example, save the directory to a USB flash drive.

7.0 Development System Setup (Windows)

7.1. Install Impulse CoDeveloper v3.70.e.6 or newer

NOTE: The use of floating point for Virtex-6 and newer devices requires the use of Xilinx's v5.0 CORE Generator cores which is supported via a patch to CoDeveloper made available via the 'XilinxFPv5BetaPatch' link under the supplied Pico PSP link

- 1. Add the Pico M5XX PSP to the CoDeveloper installation.
 - a. Copy the supplied "Architectures" directory to "Impulse\CoDeveloper3\".

Architectures

Pico

pico_m501_linux_vhdl.xml

pico_m503_linux_vhdl.xml

- 2. Copy the supplied "**Examples**" directory to a working directory on the development PC for access to the pre-built example files.
- 3. Download the latest version 3.x and installation note
 - a. http://www.impulseaccelerated.com/ReleaseFiles/
 - b. (optional) Installation of floating point support is via unzipping the .zip file (password: impulsefpv5beta) into 'Impulse' after each CoDeveloper installation. Please see enclosed README file for specific notes on the patch.

7.2. Running ISE on Windows

7.2.1. Install Xilinx ISE 13.4

- 1. Please see vendor supplied documentation for installation.
- 2. Please see vendor supplied documentation for licensing.

7.2.2. Copy the Linux Pico Installer

1. If Xilinx ISE is to be run on Windows: From the Host PC, copy "/usr/src/ picocomputing-5.0.6.1" to "c:\usr\src\picocomputing-5.0.6.1"

7.2.3. Configure Environment Variables

- 1. Add environment variable PICOBASE = "c:\usr\src\picocomputing-5.0.6.1"
- 2. Add environment variable XILINX = "c:\Xilinx\13.4\ISE_DS\ISE"
- 3. Add environment variable XILINX_BASE = "c:\Xilinx\13.4"

8.0 Passthrough Example and Tutorial

"Passthrough" is provided as an example that may be used for quickly creating user applications and for the purpose of a tutorial showing the steps involved to go from an Impulse C application in CoDeveloper all the way through to a Xilinx ISE 13.4-compiled FPGA binary and target application executable. The base files required for recreating the example using this tutorial are provided within the Impulse supplied examples which needs to be copied to a working directory on the development PC in order to run the tutorial.

NOTE: Ensure there are no spaces (' ') in the directory path chosen to avoid potential path issues with any of the tools.

The Passthrough example's hardware process is "Passthrough()", located in the source file "Passthrough_hw.c". It performs the following operations:

- 1) Read value from co_stream "input_stream"
- 2) Write value to co_stream to "output_stream"

NOTE: The hardware code runs continuously

8.1. Prerequisites

The tutorial in this Platform Support Package assumes that you have read and understand the introductory sections of the CoDeveloper User's Guide, installed with CoDeveloper and accessed from the Help menu. In particular, you should take the time to go through the tutorials provided with CoDeveloper so you have a good understanding of the front-end design flow including both desktop software simulation and hardware compilation.

8.2. CoDeveloper Project Files

The Passthrough example CoDeveloper project is made up of the following files:

- Passthrough.icProj CoDeveloper project file
- Passthrough_hw.c Source code for hardware process
- Passthrough_sw.c Source code for software processes
- Passthrough.h Header file that defines the width of the stream

When you define the width of the steam, you must make the changes in the header file as well as the in the passthrough_hw.c file. The default example defines the steam to be the maximum width of 64 bit data bus.

S 1	FextPad - [D:\ImpulseC_Pico\trunk\Examples\W5XX\Passthrough_Linux\passthrough.h]	
÷ Fi	le Edit Search View Tools Macros Configure Window Help	₽×
Ľ) 🛎 🖬 🛢 🚭 🐧 🗐 🐰 🖻 🏝 ! 그 오ㅣ ቒ 🎫 프 ୩ 🎯 🎔 🐓 🚱 ⊄ 🖓 🕨 • 🚥 🕨 🖕 📒 🗖 🖶 🕾 🖉 🖕	
1	///passthrough example.	~
345	// // Copyright(c) 2003-2009 Impulse Accelerated Technologies, Inc. //	
5 7 8 9	<pre>#define INPUT_FILE "filter_in_dat" #define OUTPUT_FILE "filter_out.dat"//all software simulation will place output file in C:\\Impulse\\trunk' //software companion to firmware implementation will place output file</pre>	
11	#define BUFSIZE 2 /* buffer size for FIFO in hardware */	
12 13	#define IMPULSEC_GLOBAL_RST_ADDR 0x08000000	
14	<pre>//128-bit math is impossible in C without function calls, so when running simulation we will use 64-bit math</pre>	h
16 17 18 19 20	<pre>#if defined(IMPUISE_C_SYNTHESIS) defined(IMPUISE_C_TARGET) #define picobusWidth 64 #define picobusDataType uint64 #else #else</pre>	
21	#define picobusWidth 64 #define picobusDataType uint64	
23 24	#endif	~
<		>
	2 4 Read Ovr Block Sync Rec C	aps 🔡

Figure 2 - Impulse C Header File with 64 bit co_stream

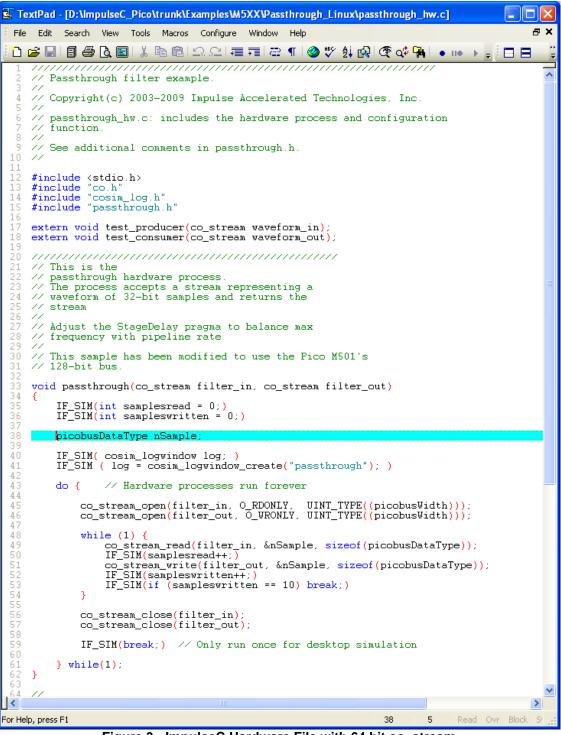


Figure 3 - ImpulseC Hardware File with 64 bit co_stream

8.3. Opening Project

Open the CoDeveloper project file 'Passthrough.icProj' by selecting and pressing 'Enter' or by double-clicking it:

Look in: Bassthroug	h_Linux 🔄	+ 🗈 💣 📰 +		
Name	*	Date modified	Туре	Si
🜛 ReadmeFiles		11/4/2012 8:04 AM	File folder	
🛃 passthrough.icProj		11/4/2012 8:04 AM	Impulse C Project	
∢ [III.			•
∢ File name:passthroug			Open	,

Figure 4 - Opening a project in CoDeveloper

8.4. Building Desktop Simulation Executable

Build the desktop software simulation executable via the "Project" menu:

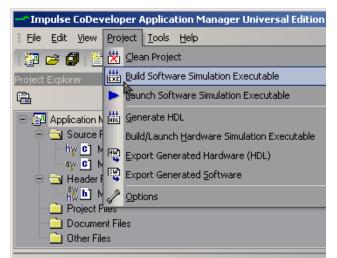


Figure 5 - Build Simulation Desktop in CoDeveloper using pull-down menu

Or via toolbar:

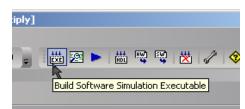


Figure 6 - Build Simulation Desktop in CoDeveloper using toolbar icon

Note the compiler output in the CoDeveloper IDE "Build" window:



Figure 7 - Output within the CoDeveloper IDE build window

8.5. Running Desktop Simulation Executable

Launch the desktop software simulation executable via "Project" menu:



Figure 8 - Launch software simulation window using pull-down menu

Or via toolbar:



Figure 9 - Launch software simulation using toolbar icon

A command window will pop up in which the desktop simulation executable runs. Press "Enter" to exit:

C:\Windows\system32\cmd.exe	
'C:\ImpulseC_Pico\Examples\M5XX\Passthrough_Linux\passthrough.exe" Impulse C is Copyright 2012 Impulse Accelerated Technologies, Inc.	-
Consumer reading data	
Sending waveform	_
cest_producer:Writing value 1	
cest_producer:Writing value 2	
cest_producer:Writing value 3	
iltered value: 1	
Filtered value: 2	
cest_producer:Writing value 4	
est_producer:Writing value 5	
Filtered value: 3	
iltered value: 4	
est_producer:Writing value 6	
est_producer:Writing value 7	
est_producer:Writing value 8	
'iltered value: 5	
est_producer:Writing value 9	
liltered value: 6	
est_producer:Writing value a	
iltered value: 7	
iltered value: 8	
inished writing waveform.	
iltered value: 9	
'iltered value: a	
onsumer read 10 waveform datapoints	
pplication complete. Press the Enter key to continue.	
pprication complete. ress the Enter Key to continue.	

Figure 10 - Pop-up window during desktop simulation

8.6. Project Setup Before Hardware/Software Generation and Export

Settings within the CoDeveloper IDE necessary for generating and exporting both hardware and software using this PSP are summarized below:

- Platform Support Package: "Pico M-501 Linux (VHDL)"
- Hardware export directory: <user hardware export directory>
- Software export directory: <user software export directory>
- Unsupported settings include:
 - Generate dual clocks (must be unchecked)
 - Active-low reset (must be unchecked)
 - Include floating point library (must be unchecked)

An example of these settings as it appears in the Passthrough example:

te dual clocks
ite active-low reset
d_logic types for VHDL interfaces
include co_ports in bus interface
ibrary options:
ctories
build directory:
uild directory:
export directory:
xport directory:

Figure 11 - Project setup to pick Platform Support Package

8.7. Generating Hardware

Generate hardware via "Project" menu:

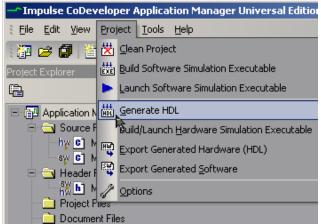


Figure 12 - Generate HDL using pull-down menu

Or via toolbar:

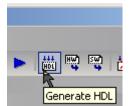


Figure 13 - Generate HDL using toolbar icon

Final results will appear in the directory specified during project setup in "Hardware build directory". Note the final output in the CoDeveloper IDE's "Build" window:

Impulse CoDeveloper Application	Manager Universal Edition = [passthrough] = [Readme.htm]	
Ele Edit View Project Tools		
🗄 🚰 🧀 🕼 🔛 🖄 Clean Proje	ct 👘 超 🕨 📓 🐂 👹 🦉 🕲 🖉 🗇 🖕	
	are Simulation Executable	4 b 🛡 🗙
9 m	tware Simulation Executable	// Open stream for read
😑 📳 Application pas 🛗 Generate H		a magnetication (a b)
Hy C pas K Export Ger	h Hardware Simulation Executable	1 (1)[1][1]] 5= co err ecol (1, "Nio ", Kelsī(K)ī;
sw i pas Export Ger	rented hardware (HUL)	, TNIS T, REDITION
Header File Coptions	enere Vortuale	
Project Files		
Document Files Other Files	Summary	
	This sample projects demonstrates the basics of streams-based communications.	
		E
	SW HW SW	
	(sim) (sim)	
	Character Pass-through Test	
	generator filter consumer	
	Buld Lacho C. Annuka Co Davalonar-Varchtach uaz /Bcn /MS/V. Lin vz Au #S/V. vnl	# ×
	Loading C./mpulse.CoDevelope%/Architectures/Pico/MS/k_Linux/busSikxmi Loading C./mpulse.CoDevelope%/Architectures/Pico/MS/k_Linux/aci_dotum Loading C./mpulse.CoDevelope%/Architectures/PiCoLinux/aci_dotum	
	Loading C // Impuber CoDeveloper3 // chitectures // HDL/Sint/ vMech.xml Loading C // Impuber CoDeveloper3 // chitectures // HDL/Sint/ vMech.xml Loading C // Impuber CoDeveloper3 // chitectures // HDL/Genetic //system.xml	
	Loading pasthrough xie Epoding pasthrough xie pspTeirgion = 0001 linux vhd ximboard=m501	
	connections2=stream in p_producer_process_waveform_in 64 picobus sp () waveform_in ()	
	joornedions2-stream out p_consumer_process_waveform_out 6 & picobus sp () waveform_out () jurning picoCutions2-stream in producer_process_waveform_in 6 bichoobus sp () waveform_in () 1) {stream out p_consumer_process_waveform_out 64 picobus sp () waveform_out () 1)	
	icon = stream in p. producer, process, waveform, in 6 fancobus go () waveform, in () 1, 1 icon = stream out, p. consumer, process, waveform, aux 6 d A choices so () waveform, aux 6 d 1, 1	
	Design generation complete chmod R + w hw	
	inkde sw "C:/moules/CoDeveloper3/bin/mpulse.jb" "=C/Impulse/CoDeveloper3/Architectures/pico_m501_jinux_vhd1xml" -hwdritw -files "passthrough_sw.c" passthrough_sic sw/co_init.c	
	Impulse C Software Interface Generator Cockerted Technologies, Inc.	
	All rights reserved, Loading C:/houles/CDeveloper3//richtectures/pico_m501_linux_vhdi xmi	
	Lading C/Impuler,CoDevelope3/Architecturer/Nor/MS/GLInux/cpu30x.ml Lading C/Impuler,CoDevelope3/Architecturer/Nor/MS/GLInux/cpu30x.ml	
	Loading pasthough vic for in passhough zer, ci do cp \$i sw; done	
	for in passinough ;; do cp si sw.; done for in passinough ;; do cp Si sw.; done	E
	crmoo -x -w/ sw ======= Buid of target build' complete =======	
	====== Bull of target own compare ======	-
	础Buld 국 Find in Files	
Generate HDL for platform target		OVR NUM

Figure 14 - Build window output

8.8. Exporting Hardware

Export hardware via "Project" menu:

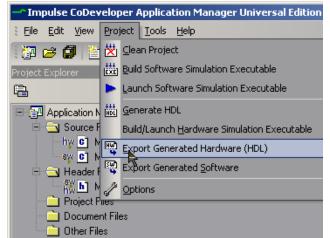


Figure 15 - Export Generated Hardware (HDL) using pull-down menu

Or via toolbar:



Figure 16 - Export Generated Hardware (HDL) using toolbar icon

Final results will appear in the directory specified during project setup in "Hardware export directory". Note the final output in the CoDeveloper IDE's "Build" window:

Image: COncertiger Application Manager Universal Solition - [passthrough] [Readmention] Image: Difference of the project
Image: Source Smulton Executable Image
Big id Software Smulton Executable Image: Software Smulton Executable Image: Software Smulton Executable Image: Software Smulton Executable
exact hardware (Ht) Junch Software Smulton Executable eract hardware (Ht) Benerate HoL Junch Software Smulton Executable Benerate HoL Hold File Doort Generated Hardware (HOL) Prove File Doort Generated Hardware (HOL) Prove File Summary This sample projects demonstrates the basics of streams-based communications. SW (sim) Character Pass-through This sample projects demonstrates the basics of streams-based communications. SW (sim) Character Pass-through Tori in pasting topot, Jundware in fig. Juladefig exerces Model Registrong in x:: do op 8 ar. done Model Registrong in x:: do op 8 ar. done Model Registrong in x:: do op 8 ar. done Model Registrong in x:: do op 8 ar. done Model Registrong in x:: do op 8 ar. done Model Registrong in x:: do op 8 ar. done Model Registrong in x:: do op 8 ar. done
<pre>entable hadware (HD) entable had ware (</pre>
erected hardware (HD) Bud/Lanch ljadware Smulation Executable Image: Second contracted software
With and the part demonstrates should be used to be an end of the second state of the second st
Image: Second
RBL res Summary Document Files Summary This sample projects demonstrates the basics of streams-based communications. SW HW SW (sim) Character Pass-through Test generator filter consumer Md ******** Min Ray for topol, hadrage in file_Madetie ******* Month Ray for topol, hadrage in file_Madetie
Proved Real Deber Files Summary This sample projects demonstrates the basics of streams-based communications. SW (sim) HW SW (sim) Character Pass-through Test generator Filter consumer
Documer Fie Summary This sample projects demonstrates the basics of streams-based communications. SW HW SW (sim) Character Pass-through Test consumer Bidd memory in flag Jub/office With Restinging w.c. do g B ar. done bar. done Month R wy fin bar. done Mont
This sample projects demonstrates the basics of streams-based communications. SW (sim) HW SW (sim) Character Pass-through Test generator filter consumer
SW HW SW (sim) HW SW Character Pass-through Test generator filter consumer Bid consumer filter consumer filter filter
Side Side Side Side Character Pass-through Test generator filter consumer divid for in pasting troop, jund, do p & m. done divided fly way for in pasting troop, jund, do p & m. done divided fly way
Side Side Side Side Character Particular Side Side <
Side Side Side Side Character Particular Side Side <
Side Side Side Side Character Particular Side Side <
Side Side Side Side Character Particular Side Side <
Character Pass-through Test generator filter consumer
Build wind ********* Building target loopot_hardware in file_Midelife ******** chronod R. write for in pastfrough year, do op B sw. done of write if file write of write if file write
Build wind ********* Building target loopot_hardware in file_Midelife ******** chronod R. write for in pastfrough year, do op B sw. done of write if file write of write if file write
Build wind ********* Building target loopot_hardware in file_Midelife ******** chronod R. write for in pastfrough year, do op B sw. done of write if file write of write if file write
Build Swide ******** chronod R-wink for in pastfrough w.r.: do op Silver. done for in pastfrough w.r.: do op Silver. done by in pastfrough b.v.: do op Silver. done
for in passifrough gw c; do cp S aw; done for in passifrough to do cp S aw; done I chrond -R +w aw
for in passitivough h, to do p Si sw; done china do p Si sw; done china do p Si sw; done china d A + w sw
"C:/Impulse/CoDeveloper3/bin/impulse_export" +hardware -srcdirtw "-aC:/Impulse/CoDeveloper3/Architectures/pico_m501_linux_vhdl xml" passthrough xic "export_hw"
Impute C Design Exporter Copyright 2022/07. Impute Accelerated Technologies. Inc.
All rights reserved. Loadin C. ¹ /moule-CoDeveloper3/r/chtectures/bio m501 linux vhd.xml
Loading C/Impute/CoDeveloper3/Architectures/Pico/M50x_Linux/cpu50x.xml
portFree too, and U and what and board and the second and the seco
board-mb01 Architecture passthrough, arch
======= Build of target 'export_hardware' complete =======
i Buld 马 Find in Files D System
Export generated hardware (HDL)

Figure 17 - Build window output

8.9. Compiling FPGA in Xilinx ISE 13.4

After exporting hardware, under the specified hardware export directory will be a directory structure that includes all necessary files for building the FPGA binary.

🔾 🚽 🚽 🖉 Impu	IseC_Pico + Examples + M5XX + Passthrough_L	.inux ▶ export_hw ▶	▼ 4 9	Search export_hw	_
Organize 🔻 🛛 Incluc	de in library 🔻 Share with 👻 Burn New	w folder			
🔆 Favorites	Name	Date modified	Туре	Size	
	🕌 hdl	11/3/2012 8:47 PM	File folder		
词 Libraries	build_bit_file.tcl	11/2/2012 7:25 PM	TCL File	4 KB	
Documents	🚳 build_m501lx240_passthrough_arch.bat	11/3/2012 8:47 PM	Windows Batch File	1 KB	
🖻 🌙 Music	build_m501lx240_passthrough_arch.sh	11/3/2012 8:47 PM	SH File	1 KB	
▷ E Pictures	customizeM501xise.tcl	11/3/2012 8:47 PM	TCL File	3 KB	
Subversion	gen_ise_files.bat	11/3/2012 8:47 PM	Windows Batch File	1 KB	
Videos	gen_ise_files.sh	11/3/2012 8:47 PM	SH File	1 KB	
	gen50x_xise.tcl	11/2/2012 7:25 PM	TCL File	27 KB	

Figure 18 - Compiling FPGA in exported ISE directory structure

At this point either Windows or Linux may be used to produce the bitfile running Xilinx ISE either through the GUI or from command line. Windows and Linux both use the following similar steps. Choose the one that suits your needs.

Windows:

- 1. Generate the ISE project files by running "gen_ise_files.bat"
- 2. Build the bit file using Xilinx ISE either by:
 - a. Running the generated "build_m50*.bat" file
 - b. Launching the Xilinx ISE GUI opening the newly built "m50*.xise" file

Linux:

- 3. Generate the ISE project files by running "gen_ise_files.sh"
- 4. Build the bit file using Xilinx ISE either by:
 - a. Running the generated "build_m50*.sh" file
 - b. Launching the Xilinx ISE GUI opening the newly built "m50*.xise" file

These methods will generate the necessary bitfile to be loaded on the Host System and are outlined in the sections that follow.

8.9.1. Building bitfile under Windows

8.9.1.1. Generate Xilinx ISE Project Files

First generate the ISE project by executing "gen_ise_files.bat". This process will create an ISE project file as well as copy all the necessary HDL files and CORE Generator components from the Pico installation needed by ISE to generate the FPGA bitfile.

- 1. Open a command window
- 2. Change directories to "export_hw"
- 3. Execute "gen_ise_files.bat" as shown in Figure 19.
- 4. Verify that the files were created successfully by viewing the end of log file, as shown in Figure 20. "Successful!" should be the last line in the log file (use "type gen50x_xise.log" to display log).

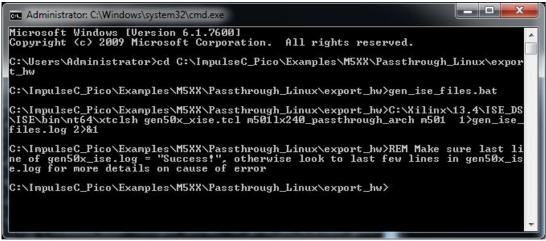


Figure 19 - Generate ISE project files

Administrator: C:\Windows\system32\cmd.exe	
source/axi_basic_rx_pipeline.v	4 01
xfile_add: add glob file=firmware/m501/coregen-LX240T/v6_pcie_v2 source/axi_basic_top.v	_4_81ane_gen2/
xfile_add: add_glob_file=firmware/m501/coregen-LX240T/v6_pcie_v2	_4_81ane_gen2/
source/axi_basic_tx.v xfile_add:	_4_81ane_gen2/
source/axi_basic_tx_pipeline.v xfile_add: add glob file=firmware/m501/coregen-LX240T/v6_pcie_v2	4 81ane (en2/
source/axi_basic_tx_thrtl_ctl.v	
xfile_add:	_4_81ane_gen2/
xfile_add:	_4_81ane_gen2/
source/gtx_rx_valid_filter_v6.v xfile_add:	
source/gtx_tx_sync_rate_v6.v xfile_add: add glob file=firmware/m501/coregen-LX240T/v6_pcie_v2	4 81ape (ep2/
source/gtx_wrapper_v6.v	an area and areas
xfile_add:	_4_81ane_gen2/
xfile_add:	_4_81ane_gen2/
source/pcie_2_0_v6.v xfile_add: add glob file=firmware/m501/coregen-LX240T/v6_pcie_v2	_4_81ane_gen2/
source/pcie_brams_v6.v xfile_add:	4 81ape (ep2/
source/pcie_bram_top_v6.v	
xfile_add: add glob file=firmware/m501/coregen-LX240T/v6_pcie_v2 source/pcie_bram_v6.v	_4_81ane_gen2/
xfile_add: add_glob_file=firmware/m501/coregen-LX240T/v6_pcie_v2	_4_81ane_gen2/
source/pcie_cfg_128.v xfile_add:	_4_81ane_gen2/
source/pcie_clocking_v6.v xfile_add:	4 8lane gen2/
source/pcie_gtx_v6.v	an anns anns anns a
xfile_add:	_4_81ane_gen2/
xfile_add: add glob file=firmware/m501/coregen-LX240T/v6_pcie_v2	_4_81ane_gen2/
source/pcie_pipe_misc_v6.v xfile_add:	_4_81ane_gen2/
source/pcie_pipe_v6.v xfile_add: add glob file=firmware/m501/coregen-LX240T/v6_pcie_v2	4 8lane gen2/
source/pcie_reset_delay_v6.v	the street st
xfile_add: add glob file=firmware/m501/coregen-LX240T/v6_pcie_v2 source/pcie_trn_128.v	_4_81ane_gen2/
xfile_add: add glob file=firmware/m501/coregen-LX240T/v6_pcie_v2 source/pcie_upconfig_fix_3451_v6.v	_4_81ane_gen2/
xfile_add:add_glob_file=firmware/m501/coregen-LX240T/v6_pcie_v2	_4_81ane_gen2/
source/sync_fifo.v xfile_add: add glob file=firmware/m501/coregen-LX240T/v6_pcie_v2	_4_81ane_gen2/
source/trn_rx_128.v xfile_add:add glob_file=firmware/m501/coregen-LX240T/v6_pcie_v2	19. 17.55 (19.17.5
source/trn_tx_128.v	
xfile_add: add glob file=firmware/m501/coregen-LX240T/v6_pcie_v2 source/v6_pcie_v2_4_8lane_gen2.v	_4_81ane_gen2/
xfile_add: add file=firmware/m501/src/M501_LX240T_DDR3.ucf	
xfile_add: add file=firmware/m501/src/M501_LX240T_PCIe.ucf xfile_add:End	
Configuring project properties closing project	
Success!	
C:\ImpulseC_Pico\Examples\M5XX\Passthrough_Linux\export_hw>	*

Figure 20 - Expected gen50x_xise.log report

8.9.1.2. Compiling FPGA in Xilinx ISE GUI

Launch the Xilinx ISE GUI and open the ISE project located in the **export_hw** directory. Select Pico_Toplevel in the *Hierarchy* window and double-click "generate programming file" in the *Processes* window.

NOTE: Xilinx ISE will ask the user to regenerate two fifos (a one time process). The following figures illustrate initial project launch, regenerate the two corgen fifos, and final output with timing score equal to zero.

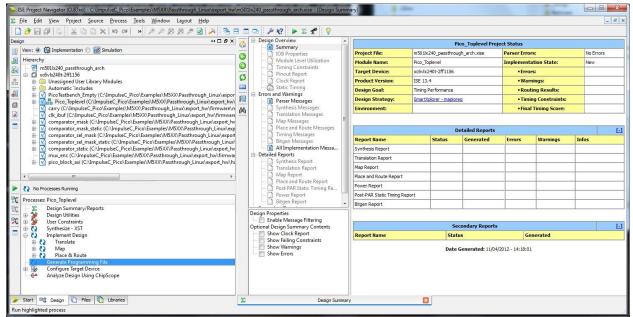


Figure 21 - Initial Xilinx ISE GUI screen

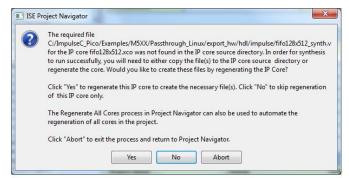


Figure 22 - Xilinx ISE regenerate IP core fifo128x512

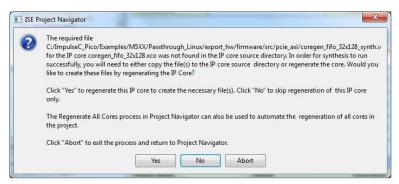


Figure 23 - Xilinx ISE regenerate IP Core coregen_fifo_32x128

sign ↔ □ & ×	0	e	Desi	ign Overview	-		Pico_Toplevel Project St	atus (11/04/2	012 - 14:45:3	7)	
View: 💿 🏨 Implementation 🔘 🧱 Simulation	-			Summary IOB Properties		Project File:	m501lx240_passthrough_arch.xise	Parse	er Errors:	1	No Errors	
Hierarchy ├────────────────────────────────────				Module Level Utilization		Module Name:	Pico_Toplevel	Imple	ementati	on State:	Programmin	g File Generated
				 Timing Constraints Pinout Report 		Target Device:	xc6vlx240t-2ff1156		•Errors:	5	No Errors	
🗄 🛅 Unassigned User Library Modules	ç			Clock Report		Product Version:	ISE 13.4		• Warnin	igs:	8090 Warni	ngs (8089 new)
🐵 🧰 Automatic includes	FILE	11		Static Timing rs and Warnings		Design Goal:	Timing Performance		• Routin	g Results:	All Signals C	ompletely Route
V PicoTestbench_Empty (C:\ImpulseC_Pico\Examples\M5XX\Passthrough_Linux\export V Pico_Toplevel (C:\ImpulseC_Pico\Examples\M5XX\Passthrough_Linux\export_hw)		F		Parser Messages		Design Strategy:	SmartXplorer - mapioreg	• Timing Constraint		Constraints:	s: All Constraints Met	
carry (C:\ImpulseC_Pico\Examples\M5XX\Passthrough_Linux\export_hw\firmware\r	10			Synthesis Messages		Environment:	System Settings		• Final Timing Score:		0 (Timing Report)	
				 Translation Messages Map Messages 		-						
Comparator_mask (Cumpulse_)rico txamples/WXAVastmrough_inux.epport, Comparator_mask, tacking (CUmpulse_)rico(txamples/MXAVAstmrough_inux.epport V comparator_sel_mask (CAImpulseC, Pico(txamples/MXAVAstmrough_inux) v comparator_sel_mask(txtic (CulmpulseC, Pico(txamples/MXAVA) v comparator_static (CUmpulseC, Pico(txamples/MXAVA) v comparator_static (CUmpulseC, Pico(txamples/MXAVA) mux_enc (CUmpulseC, Pico(txamples/MXAVA) mux_enc (CUmpulseC, Pico(txamples/MXAVA) v mux_enc (SUMmulseC, Pico(txamples/MXAVA) v mux_enc (SUMMulseC) v mux_enc (SUMMulseC)	ė.			Place and Route Messages		Device Utilization Summa						
				Timing Messages		Slice Logic Utilizati	on		Used	Available	Utilization	Note(s)
	ł.		Bitgen Messages		Number of Slice Registers			26,283	301,440	8%		
		0	Post-PAR Static Tin		8	Number used as Flip	Number used as Flip Flops		26,280			
				Translation Report		Number used as Lat	tches		3			
				Place and Route Report Post-PAR Static Timing Re Power Report		Number used as Lat	tch-thrus		0			
No Processes Running						Number used as AN	ID/OR logics		0			
Processes: Pico Toplevel	Ě					Number of Slice LUTs			16,908	150,720	11%	
Design Summary/Reports				Bitgen Report	-	Number used as log	ic		10,599	150,720	7%	
🕀 😼 Design Utilities				Properties		Number using O6	output only)	7,967			
🐵 🌠 User Constraints		Optional Design Summary Contents		Number using O5 output only			601					
Constant Synthesize - XST Design			- 🗐 :	Show Clock Report		Number using O5 and O6			2,031			
🕀 🔁 🔔 Translate		H		Show Failing Constraints Show Warnings		Number used as ROM			0			
⊕ ₹2.1 Map ⊕ ₹3.1 Place & Route		L		Show Errors		Number used as Memory			2,533	58,400	4%	
- 🖒 🛕 Generate Programming File		L .	-			Number used as Dual Port RAM			1,156			
Configure Target Device Analyze Design Using ChipScope		1		Number using O6 output only			320					
Analyze Design Using ChipScope						Number using	O5 output only		9			
						Number using	05 and 06		827			

Figure 24 - Xilinx ISE 13.4 with timing score = 0

Once ISE 13.4 has completed, a bitfile "pico_toplevel.bit" will be present in the **export_hw** directory.

Note: The bitfile will need to be renamed to "Pico_Toplevel.bit" when copying to Linux to run with the software in the example.

8.9.1.3. Compiling FPGA in Xilinx ISE using Command Line

In the top directory there will be the batch file "build_passthrough_arch.bat" used to automatically run Xilinx ISE to create the necessary .bit file used to program the M501 FPGA.

- 1. Open a command window
- 2. Execute "build_m501lx240_passthrough_arch.bat" as shown in Figure 25.

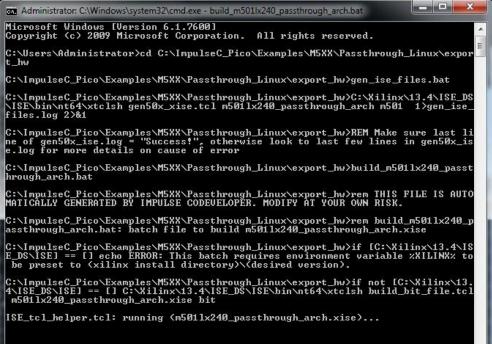


Figure 25 - Build bitfile in ISE 13.4

A command window will appear showing the FPGA build process (primarily made up of many, many info and warning messages). Compile time will vary by machine depending upon project size. When completed successfully, something similar to the following will appear and a bitfile "pico_toplevel.bit" will be present in the **export_hw** directory.

Note: The bitfile will need to be renamed to "Pico_Toplevel.bit" when copying to Linux to run with the software in the example.

C:\Impulse	C_Pico\Examples\M5XX\Passthrough_Linux\export_hw\I	Pico_Toplevel_summary.html	▼ 49	X Funmoods	,			
🖥 Favorites 🛛 🍰 🔽 Suga	gested Sites 👻 🙋 Web Slice Gallery 👻 🔏 Building Ap	plications that 🔏 How to C	Configure Visual 🛛 🔏 Ho	ow to Enable a 64-Bit Vis				
🗿 Xilinx Design Summary				• 🖾 • 🖃 🖶 • Pa	ge 🔻 Safety 👻 Tools 👻 🌘			
To help protect your secur	ity, Internet Explorer has restricted this webpage from rur	nning scripts or ActiveX contro	Is that could access your co	mputer. Click here for optio	ns			
	D:-	Taulanal Duria at Stat						
During Film		o_Toplevel Project State	IS	No Errors				
Project File: Module Name:	m501lx240_passthrough_arch.xise	Parser Errors: Implementation	6 4 - 4		. 1			
Module Name:	Pico_Toplevel		State:	Programming File Ger	nerated			
Target Device:	xc6vlx240t-2ff1156	• Errors:		No Errors				
Product Version:	ISE 13.4	• Warnings		8090 Warnings (8089	9 new)			
Design Goal:	Timing Performance	Routing F	Lesults:	All Signals Completel	y Routed			
Design Strategy:	SmartXplorer - mapioreg	• Timing Co	onstraints:	All Constraints Met				
Environment:	System Settings	• Final Timi	ing Score:	0 (Timing Report)				
	Device Ut	ilization Summary						
Slice Logic Utilization	Device Of	Used	Available	Utilization	[-] Note(s)			
Number of Slice Registe		26,283	301.440	8%	rote(s)			
Number used as Flip		26,283	501,440	070				
Number used as Late		3						
Number used as Late		0						
Number used as ANI		0						
Number of Slice LUTs	, or logics	16,908	150,720	11%				
Number used as logic		10,500	150,720	7%				
Number using O6	U.	7.967	150,720					
Number using O5 o		601						
Number using O5 a		2.031						
Number used as R		0						
Number used as Men		2,533	58,400	4%				
Number used as D		1,156						
Number using O		320						
Number using O		9						
Number using O		827						
Number used as Si		0	~ 2					
Number used as SI	uft Register	1,377						

Figure 26 – Xilinx ISE 13.4 compile log file – timing score

8.9.2. Building bitfile under Linux

8.9.2.1. Generate Xilinx ISE Project Files

First generate the ISE project by executing "gen_ise_files.sh". This process will create an ISE project file as well as copy all the necessary HDL files and CORE Generator components from the Pico installation needed by ISE to generate the FPGA bitfile.

- 1. Copy the **export_hw** directory to a location on your Host System that has the Linux operating system. (ie Pico/Passthrough/export_hw).
- 2. Open a command console
- 3. Change directories to "export_hw"
- 4. Execute "source ./gen_ise_files.sh" as shown in Figure 27.
- 5. Verify that the files were created successfully by viewing the end of log file, as shown in Figure 28. "Successful!" should be the last line in the log file (use "cat gen50x_xise.log" to display log).

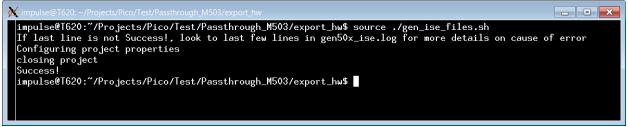


Figure 27 - Generate ISE project files

impulse@T620: ~/Projects/Pico/Test/Passthrough_M503/export_hw
xfile_add: add glob file=firmware/m503/coregen-LX240T/v6_pcie_v2_4_8lane_gen2/source/pcie_pipe_v6.v
xfile_add: add glob file=firmware/m503/coregen-LX240T/v6_pcie_v2_4_8lane_gen2/source/pcie_brams_v6.v
xfile_add: add glob file=firmware/m503/coregen=LX240T/v6_pcie_v2_4_81ane_gen2/source/pcie_pipe_misc_v6.v
xfile_add: add glob file=firmware/m503/coregen-LX240T/v6_pcie_v2_4_81ane_gen2/source/axi_basic_rx.v
xfile_add: add glob file=firmware/m503/coregen-LX240T/v6_pcie_v2_4_8lane_gen2/source/pcie_clocking_v6.v
xfile_add: add glob file=firmware/m503/coregen=LX240T/v6_pcie_v2_4_81ane_gen2/source/trn_rx_128.v
xfile_add: add glob file=firmware/m503/coregen=LX240T/v6_pcie_v2_4_81ane_gen2/source/axi_basic_tx_thrt1_ct1.v
xfile_add: add glob file=firmware/m503/coregen-LX240T/v6_pcie_v2_4_81ane_gen2/source/pcie_reset_delay_v6.v
xfile_add: add glob file=firmware/m503/coregen-LX240T/v6_pcie_v2_4_8lane_gen2/source/axi_basic_rx_null_gen.v
xfile_add: add glob file=firmware/m503/coregen-LX240T/v6_pcie_v2_4_81ane_gen2/source/axi_basic_tx.v
xfile_add: add glob file=firmware/m503/coregen-LX240T/v6_pcie_v2_4_8lane_gen2/source/sync_fifo.v
xfile_add: add file=firmware/m503/src/M503_LX240T_DDR3_0.ucf
xfile_add: add file=firmware/m503/src/M503_LX240T_DDR3_1.ucf
xfile_add: add file=firmware/m503/src/M503_LX240T_PCIe.ucf
xfile_add:End
Configuring project properties
closing project
Success!
impulse@T620:~/Projects/Pico/Test/Passthrough_M503/export_hw\$

Figure 28 - Expected gen50x_xise.log report

8.9.2.2. Compiling FPGA in Xilinx ISE GUI

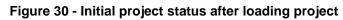
- 1. If you already haven't done so, launch or verify that the license manager is operational and that your license is valid. Contact Xilinx for more information on licensing related to a Linux installation.
- 2. Open a terminal window and set environment variables by executing the "settings64.sh" script as well as the following variables:
 - a. source /opt/Xilinx/13.4/ISE_DS/setting64.sh
 - b. export PICOBASE=/usr/src/picocomputing-5.0.6.1
 - c. export XILINX=/opt/Xilinx/13.4/ISE_DS/ISE
- Go to the export_hw directory and begin executing the shell scripts to generate the Xilinx ISE project.
 - a. cd /Pico/Passthrough/export_hw
 - b. source gen_ise_files.sh
- 4. Launch the Xilinx ISE 13.4 GUI, select the project, regenerate coregent files, and generate the bitfile.

🛞 Open Pr	oject					
Look in:	home/administratorssthrough/export_hw	\$ G	0	0	G	
Computer	firmware.					
File <u>n</u> ame:	m501lx240_passthrough_arch.xise	 				<u>O</u> pen
Files of type:	ISE Project Files (*.xise)			\$][]	<u>Cancel</u>

a. /opt/Xilinx/13.4/ISE_DS/ISE/bin/lin64/ise &

Figure 29 - Select the ISE project

Des	ign Giù@@	0	E Design Overview	1		Pico 1	loplevel Projec	t Status				
UI.	View® 🛱 Implemental 🔿 🗱 Simulat	-	Summary	Project File: m501ix240 passthrough arch.xise			ugh_arch_xise	Parser Er	No B	Errors		
51	Hierarchy	0	Moour Lavel Utilization	Module Name:	Pico_Toplevel xc6vix240t-2ff1156			Implemen	New	New		
ñ.	m501lx240_passthrough_arch ac6vlx240t 2ff1156 ac6vlx240	0	Timing Constraints	Target Device:				+ Erro				
		0	Clock Report	Product Version:	ISE 13.4			+ War	mings:			
Ē.	Automatic `includes PicoTestbench Empty (Pico`)	-	Errors and Warnings	Design Goal:	Timing Pe	erformance		+ Rou	ting Results:			
Ġ.	r Rico Toplevel (Pice Tople	-	Parser Messages	Design Strategy:	Xilinx Del	fault (unlock	ec]	+ Tim	ing Constraint	s:		
2	carry (carry.v)	錮	Synthesis Messages	Environment:				+ Fina				
	🗉 🖸 comparator_mask (compar-	-		Lafter officer of the second				11/101000	Participant and and an and an			
9	 comparator_mask_static (c- comparator_sel_mask (com 	60		E.		-						
	🗉 🖸 comparator sel mask stati			Report Name		Status	Generated	Errors		Infos	ŝ	
	comparator_static (compare mux enc (mux enc.v)		All Implementation Mes	Synthesis Report		Status	Generated	Errors	Warnings	Intos		
	Inde Circ Inde Circ V pice block, and pice block v No Processes Running Processes: Pice Toplevel Design Utilities * 2 Design Utilities		Detailed Reports Synthesiz Report Tarniation Report Add Report	Translation Report		-				-	_	
				Map Report		-	-		-	-		
			Flace and Route Seport	Place and Boute Rep						-		
-			Peet PAR Stelle Timing	Power Report	na.	-	-			-		
e			B Power Report	Post-PAR Static Timing Report						-		
21				Bitgen Report	3.00			-		-		
E.			Design Properties				1					
ų	🗷 🔁 Synthesize - XST		Optional Design Summary Contents									
	E () Implement Design E () Translate				10.000	dary Reports	-					
	æ €≩ Map		Show Warnings	Report Name	Status			Generated				
	E: Q Place & Route Generate Programming file Generate Programming file Generate Programming file Generate Design Using Ch		1 I Show Errors			Date Gene	arated: 11/07/2	012 - 21:29:5	6			
	Start 🗠 Design 🚺 Files 👔	T	Par Design Summ	ary	×							
Lon DI DI	sole NFO:ProjectHgmt - Include file	foun foun n hi	d: '/home/administrator/Pico/Pas d: '/home/administrator/Pico/Pas erarchy completed successfully.	sthrough/expert_hw/	hdl/impul:					/Pics/Pa		



🛞 ISE Project Navigator										
The required file /home/administrator/Pico/Passthrough/export_hw/hdl/impulse/fifo128x512_ for the IP core fifo128x512.xco was not found in the IP core source director order for synthesis to run successfully, you will need to either copy the file the IP core source directory or regenerate the core. Would you like to creat these files by regenerating the IP Core?	y. In e(s) to									
Click "Yes" to regenerate this IP core to create the necessary file(s). Click " skip regeneration of this IP core only.	Click "Yes" to regenerate this IP core to create the necessary file(s). Click "No" to skip regeneration of this IP core only.									
The Regenerate All Cores process in Project Navigator can also be used to automate the regeneration of all cores in the project.										
Click "Abort" to exit the process and return to Project Navigator.										
🗶 Abort 🖉 🖉	Yes									
Figure 24 Degenerate file420vE42										

Figure 31 - Regenerate fifo128x512

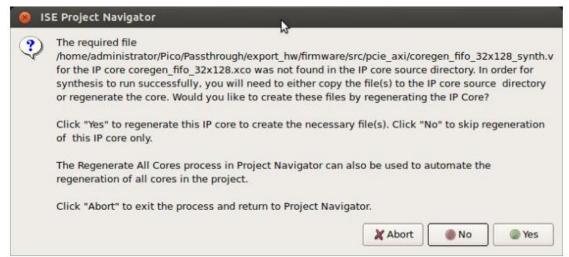


Figure 32 - Regenerate coregen_fifo_32x128

Des	ign eigele	in.	E Design Overview		Pico Toplevel Project Status	(11/07/20	12 - 21:57:26)					
1	View 🖲 🎆 Implementat 🗇 🚟 Simulat	00	 Summary IOB Properties 	Project File:	Parser E	and the local design of th	No Errors					
	Hierarchy	Module Level Utilization		Module Level Utilization Timing Constraints	Module Lavel Utilization	Module Lavel Utilization	Module Lavel Utilization Module Name: Pico Toplevel	- And	Implementation State:		Programming File Generated	
4	 ac6vix240t 2ff1156 ac6vix240t 2ff1156 ac6vix240t 2ff1156 		 Pinout Report Clock Report 	Target Device:	xc6vlx240t-2ff1156	+ Erre	ors:	No En	rors			
-	Automatic `includes PicoTestbench_Empty (Pico`	Q III	Static Timing	Product Version:	ISE 13.4	+ Wa	mings:		8090 Warnings (8089 new)			
1	Pico_Taplevel (Pico_Tople carry (carry.v) clk_ibuf (clk_ibuf.v)	(P) S Place and Route Messa	Synthesis Messages	Synthesis Messages	Design Goal:	Timing Performance	+ Rot	rting Results:	All Signals Completely Routed			
a	comparator_mask (compare comparator_mask_static (comparator_mask_static (comparator_sel_mask (comparator_sel_mask (comparator))))))))))))))))))))))))))))))))))		 Map Messages Place and Route Messa 	Design Strategy:	Xilinx Default (unlocked)	+ Timing Constraints:		All Constraints Met				
3	comparator_sel_mask (com comparator_sel_mask_stati comparator_sel_mask_stati comparator_static (comparator_static (comparator_sta	Bitgen Messages		Environment:	ironment: System Settings		• Final Timing Score:		0 (Timing Report)			
	F g pico_block_axi (pico_block_		Detailed Reports Synthesis Report Translation Report		Device Utilization Su	mmary			Ŀ			
	4 II F		Map Report Place and Route Report	Slice Logic Utiliza	tion	Used	Available Utili	zation	Note(s)			
	No Processes Running	Processes Running	Post PAR Static Timing	Number of Slice Re	gisters	26,283	301,440	8%				
	Processes: Pico Toplevel		Bitgen Report	Number used as Flip Flops								
	Design Summary/Reports Sign Utilities		E Secondary Reports	Number used as	Latches	3						
1		Design Properties Enable Message Filtering Optional Design Summary Contents		Number used as Latch-thrus		0						
	😤 🎽 User Constraints 🗷 🔁 🔥 Synthesize - XST		Number used as	AND/OR logics	0							
	E 🔁 🚹 Implement Design		Show Clock Report	how Clock Report Number of Slice LUTs Number used as logic how Warnings		16,908	150,720	11%				
1	王 (2) A Translate 王 (2) A Map		Show Failing Constraints			10,599	150,720	7%,				
	🖭 🔁 🛕 Place & Route		Show Warnings			7,967						
	Generate Programming File Source Target Device					601	0.8					
	Analyze Design Using Ch					2.031						
	The second s					a						
			Number used as	2.533	58,400	4%						
	Start Cosign Start Files	g	Design Summary (Programm	ing Hile Generated)	×	1			80			
	727 R.		d. Routing from the different bu	ffer types					62144			

Figure 33 - Place and Route and bitfile generation with timing score equal to zero

8.9.2.3. Compiling FPGA in Xilinx ISE using Command Line

1. Copy the **export_hw** directory to a location on your Host System that has the Ubuntu operating system. (ie Pico/Passthrough/export_hw).

- 2. If you already haven't done so, launch or verify that the license manager is operational and that your license is valid. Contact Xilinx for more information on licensing related to a Linux installation.
- 3. Open a terminal window and set environment variables by executing the "settings64.sh" script as well as the following variables:
 - c. cd /opt/Xilinx/13.4/ISE_DS
 - d. sudo ./setting64.sh
 - e. export PICOBASE=/usr/src/picocomputing-5.0.6.1
 - f. export XILINX=/opt/Xilinx/13.4/ISE_DS/IS
- 4. Go to the **export_hw** directory and begin executing the shell scripts to generate then build the FPGA bitfile.
 - g. cd /Pico/Passthrough/export_hw
 - h. source gen_ise_files.sh
 - i. source build_m501lx240_passthrough_arch.sh
- 5. Verify that the bitfile was generated and that the timing score equals zero.
 - j. Il *.bit (should see Pico_Toplevel.bit)
 - k. grep -- i score *.par

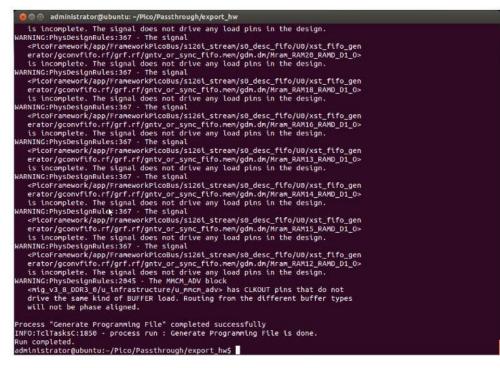


Figure 34 - Ubuntu Xilinx ISE 13.4 Command Line output

] 🧰 Open 🔹 💆 Save 🚦 🦛 Undo 🦽	× 1	1	2		*		
🖉 go2psp.sh 🕱 📄 Pico_Toplevel.par 🕱							
MMCM_PHASE_CALIBRAII	2	8.000	0.746		Q 1	iming score	^ v
MMCM_PHASE_CALIBRATI		8.000	0.323				
MMCM_PHASE_CALIBRATI		0.000	0.293				
* The fanout is the number of component pin: for example SLICE loads not FF loads. TURING SCORE: 0 (Setup: 0, Hold: 0, Component INF0:Timing:3386 - Intersecting Constraints Information, see the TSI report. Please Tools User Guide for information on geneu Number of Timing Constraints that were not a Asterisk (*) preceding a constraint indicate This may be due to a setup or hold violational	nt Switchin found and consult th rating a TS applied: 3 es it was n	g Limit: resolvec e Xilin I report	: 0) d. For mor « Command L	e			
Constraint	Check	We	orst Case			Timing	
			Slack	Achievable	Errors	Score	
TS_CLK_250 = PERIOD TIMEGRP "CLK_250" TS_ SYSCLK HIGH 50% PRIORITY 10	SETUP HOLD MINPERIO		0.005ns 0.000ns 0.000ns		0 0	0 0 8	
TS_CLK_500 = PERIOD TIMEGRP "CLK_500" TS_ SYSCLK * 2 HIGH 50% PRIORITY 1	SETUP HOLD MINPERIO	D	0.005ns 0.000ns 0.000ns	i	0 0 0	0 0 0	
TF	1. #19195195		0.0001	4.0201			

Figure 35 - Xilinx ISE 13.4 results file with timing score equal to zero

8.10. Exporting Software

The software application to be run on the host computer (with the M501 installed with it's drivers) can be exported in CoDeveloper.

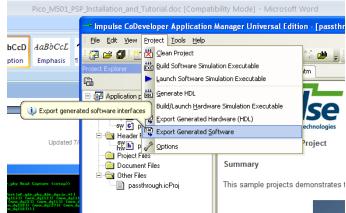


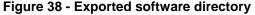
Figure 36 - Export Generated Software

Once completed without error in the Build window, it should be noted that the software code will be written to a newly created directory. The user can modify the target directory name. In this example, export_sw contains the exported software files.

Impulse CoDeveloper Application	Manager Universal Edition - [passthroug	h] - [Readme.htm]				
Eile Edit View Project Tools	Help					
🗄 🚰 😅 🕼 🎦 💥 Clean Projec		📑 🛗 🗷 🕨 👹 🗣	题 概 2 🕎 🛢			
	are Simulation Executable					4 Þ 🛡 🗙
	tware Simulation Executable					4 P • X
E Application pat	DI					s.open(ks_is, 0_RDOWIX); // Open stream for read A pdfadne_fwrits(ing, "SDES toyschedule:\n");
	h Hardware Simulation Executable					- 6; 1 < 2; k++) (Equilates forstalling, "Reading Edge Adva",k);
Leu sontware internaces	ambed Upedware (UDL)					L = 0: L = HE DEFT(L_L++ } [De BELGARI H==B(KE LA, AKEL[[k]] != SO @EF BOO[[HE LOGWINDOW PWITE(LOG, "%14 ", KE[1][k]])
sw C pas Export Gen	arstad Coffuera					in_logwindow_fwrite(log, "Wig ", %21)
Header File Coptions	eraced <u>s</u> ortware					
hw pas / Options		1				
Document Files	Summary					
💼 Other Files						
1	This sample projects demonstrates	the basics of streams-basi	ed communications.			
			and the second			=
					B	
			SW	LBAZ	SW	
			(sim)	HW	(sim)	
			()		()	
			Character	Pass-through	Test	
			generator	filter	consumer	
			generator	men	consumor	
	Build					1
	======= Building target 'export_software' in t	file Makefile =======				
	for i in passthrough sw.c; do cp \$i sw; done					
	for i in passthrough h; do cp \$i sw; done chmod -R +nv sw					
	"C:/Impulse/CoDeveloper3/bin/impulse_expo Impulse C Design Exporter	.rt" -software -srcdirsw "-aC:/Impl	llse/CoDeveloper3/Architectu	res/pico_m501_linux_vhdl.xml" pa	ssthrough xic "export_sw"	
	Copyright 2002-2007, Impulse Accelerated Te	chnologies, Inc.				
	All rights reserved. Loading C:/Impulse/CoDeveloper3/Architectu	ures/pico m501 linux vhdl.xml				
	Loading C:/Impulse/CoDeveloper3/Architectu Loading passthrough xic	.res/Pico/M50x_Linux/cpu50x.xr	nl			
	Build of target 'export_software' cor	nplete ======				
						-
	🛗 Build 🔫 Find in Files 🖸 System					
Export generated software interface						OVR NUM .:
Export generated software interface	15		a concernante			OVR NUM

Figure 37 - Build window output

Organize 👻 Include in lib	orary 🕶 Share with 🖛 🛛 Burn 🛛 New fo	lder			· 🗖 🌘
🔆 Favorites	^ Name	Date modified	Туре	Size	
	👔 lib	11/2/2012 9:44 PM	File folder		
词 Libraries	co_init.c	11/2/2012 9:44 PM	C Source	2 KB	
Documents	Makefile	11/2/2012 9:44 PM	File	1 KB	
🖻 🌙 Music	h passthrough.h	11/2/2012 9:44 PM	C/C++ Header	1 KB	
 Pictures Subversion Videos 	c passthrough_sw.c	11/2/2012 9:44 PM	C Source	5 KB	



8.11. Programming the FPGA

The compiled software application in the "export_sw" directory will program the FPGA on the M501. Please continue to the next section.

8.12. Running Target Executable on the Host System

The Host System will need the following files:

- 1. Compiled bitfile output from ISE 13.4
- 2. Input data file
- 3. Application software source code
- 4. Makefile to compile the software

Please copy the following files and directory over to the Host System (figure 30). Create a folder (ie Pico) and copy the following to that folder.

- 1. **export_hw** directory which includes "pico_toplevel.bit" bitfile
- 2. **export_sw** directory which includes the Makefile and software application source code
- 3. filter_in.dat file which will be used to input stimulus to the FPGA via the software application.
- 4. Once the files are copied, please copy filter_in.dat to the **export_sw** directory. The application software will expect the input data file to be present in that directory.

Irganize	e 🔻 🔚 Open Burn New fol	der			
*	Name	Date modified	Туре	Size	
	👩 export_hw	11/4/2012 8:20 AM	File folder		
3	👩 export_sw	11/4/2012 8:20 AM	File folder		
	👩 hw	11/4/2012 8:19 AM	File folder		
GE	illes ReadmeFiles	11/4/2012 8:04 AM	File folder		
E	👩 sw	11/4/2012 8:19 AM	File folder		
E	🚳 _Make_passthrough.bat	11/4/2012 8:20 AM	Windows Batch File	1 KB	
	Q _Makefile	11/4/2012 8:20 AM	File	1 KB	
	_Makefile.defs	11/4/2012 8:19 AM	DEFS File	1 KB	
8	🔊 filter_in.dat	11/4/2012 8:04 AM	DAT File	1 KB	
	assthrough.h	11/4/2012 8:04 AM	C/C++ Header	1 KB	
	of passthrough.i	11/4/2012 8:19 AM	Preprocessed C/C	16 KB	
6	🕢 passthrough.icProj	11/4/2012 8:04 AM	Impulse C Project	2 KB	
	passthrough.pk0	11/4/2012 8:19 AM	PK0 File	27 KB	
	passthrough.pk1	11/4/2012 8:19 AM	PK1 File	14 KB	
	passthrough.pky	11/4/2012 8:19 AM	PKY File	14 KB	
	passthrough.sic	11/4/2012 8:19 AM	SIC File	13 KB	
	assthrough.smd	11/4/2012 8:19 AM	SMD File	5 KB	
	o passthrough.snt	11/4/2012 8:19 AM	SNT File	27 KB	
	assthrough.xhw	11/4/2012 8:19 AM	XHW File	3 KB	
	passthrough.xic	11/4/2012 8:19 AM	XIC File	4 KB	
	assthrough_hw.c	11/4/2012 8:04 AM	C Source	3 KB	
	assthrough_sw.c	11/4/2012 8:04 AM	C Source	5 KB	
	Readme.htm	11/4/2012 8:04 AM	HTML Document	2 KB	

Figure 39 - Files & Directories to be copied to the Host System

The software application will load the FPGA every time it is executed and report the results upon completion. Please note that the process of loading the FPGA may take up to 20 seconds due to the driver re-start sequence after the FPGA bitfile has been transferred to the card.

- 1. Open a terminal window.
- 2. Navigate to the location of your copied files and directories:

(ie. cd Pico/export_sw/software).

- 3. Run "make"
- 4. Execute the generated application: "./passthrough_arch"

Image: Second
ALL AD USE SUITE AND DITUUY DEACTY T Thus less interesting F A 6 thereft
g++ -O3 -Wno-multichar -DLINUX -DPOSIX -II/usr/src/picocomputing-5.0.6.1/soft
ware/include/linux -I/usr/src/picocomputing-5.0.6.1/software/include -fmessage-l
ength=0 -fdiagnostics-show-option -fms-extensions -Wno-write-strings -DOSNAME=
"Linux" -I lib -o passthrough_arch passthrough_sw.c co_init.c lib/co_memory.c li
<pre>b/co_process.c lib/co_stream.c lib/co_type.c lib/pico_wrapper.cpp /usr/src/picoc</pre>
omputing-5.0.6.1/software/source/GString.cpp /usr/src/picocomputing-5.0.6.1/soft
ware/source/pico_drv.cpp /usr/src/picocomputing-5.0.6.1/software/source/pico_err
ors.cpp /usr/src/picocomputing-5.0.6.1/software/source/linux/linux.cpp /usr/src/
picocomputing-5.0.6.1/software/source/pico_drv_linux.cpp
administrator@ubuntu:~/Pico/export_sw/software\$
administrator@ubuntu:~/Pico/export_sw/software\$
administrator@ubuntu:~/Pico/export_sw/software\$
administrator@ubuntu:~/Pico/export_sw/software\$
administrator@ubuntu:~/Pico/export_sw/software\$./passthrough_arch
Pico_LoadFPGA:Loading FPGA with '//export_hw/pico_toplevel.bit'
Impulse C is Copyright 2012 Impulse Accelerated Technologies, Inc.
Sending waveform
test_producer:Writing value 1
test_producer:Writing value 2
test_producer:Writing value 3
test_producer:Writing value 4
test_producer:Writing value 5
test_producer:Writing value 6
test_producer:Writing value 7
test_producer:Writing value 8
test_producer:Writing value 9
test_producer:Writing value a
Finished writing waveform.
Consumer reading data
Filtered value: 1
Filtered value: 2
Filtered value: 3
Filtered value: 4
Filtered value: 5
Filtered value: 6
Filtered value: 7
Filtered value: 8
Filtered value: 9
Filtered value: a
Consumer read 10 waveform datapoints
Application complete. Press the Enter key to continue.
^C

Figure 40 - Exported SW executed on target platforn

9.0 Memtest Example and Tutorial

9.1. Prerequisites

The tutorial in this Platform Support Package assumes that you have read and understand the introductory sections of the CoDeveloper User's Guide, installed with CoDeveloper and accessed from the Help menu. In particular, you should take the time to go through the tutorials provided with CoDeveloper so you have a good understanding of the front-end design flow including both desktop software simulation and hardware compilation.

9.2. Memtest Example Description

The Memtest example is a CoDeveloper project that includes three source files. A brief description of the files and their functions is outlined in the following sections. This example targets existing external DDR3 memory available on the M501 FPGA module.

- 1. memtest.h
- 2. memtest_hw.c
- 3. memtest_sc.c

The Memtest example allows a stream to write to DDR3 memory. The Impulse User module will read from DDR3, increment the data by one, and write it back out to a stream. The software application generates the data and checks the data returned from the FPGA.

9.2.1. Overview of memtest.h

The figure below is a screen shot of header file. It defines the stream widths (STREAMWIDTH) as 128 bits. It also defines the size of the target memory (MEMORY_SIZE) that will be used by the co_memory API calls.

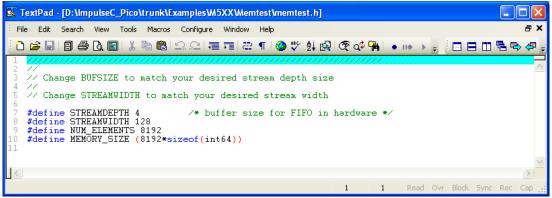


Figure 41 - memtest.h

9.2.2. Overview of memtest_hw.c

The memtest_hw.c file is the user defined function that accesses external DDR3 memory. The figure below shows the user defined function, *pass*, that will be implemented on the FPGA. On line 18, the ports are defined as input stream *idata*, output stream *odata*, and memory port *mem*.

In this example, the function **pass** will receive, on stream **idata**, the number of words to read from external DDR3. The "do" loop on line 29 indicates that this hardware process run continuously. Input and output streams, **idata** and **odata**, are open as shown on lines 32 & 33. Line 36 is an Impulse API to read in data from the stream. Line 39 shows the Impulse API to read a block of data from external memory. The "for" loop on line 41 operates on each word as it is received from external memory and is also incremented by one. Once all data had been read and incremented, it is written back to external memory as shown on line 47. The data is also written back out on the stream, **odata**, as shown on line 49. Once all data and processing is complete, the input and output streams are closed.

TextPad - [D:\ImpulseC_Pico\trunk\Examples\M5XX\Memtest\memtest_hw.c] 8 × File Edit Search View Tools Macros Configure Window Help 🗅 🚅 🖫 🛯 🚭 🐧 🗐 🗼 🖻 🛍 으 오너 = 🗐 🗁 ୩ 🎯 ザ 🛃 🐼 👁 🖓 🖡 🔹 🚥 void pass(co_stream idata, co_stream odata, co_memory mem) ~ 19 1 m $-\mathbf{n}$ co_uint128 data128; 21 22 23 int64 data, inc. int64 work[NUM ELEMENTS]; 24 25 // EdT: co_banks not available until CoDeveloper v3.70.d.13 or later // Configure array to be two banks of 64 bits to maximize throughput from 128
// co_array_config(work,co_banks,"2"); 26 27 28 29 do £ 31 // Hardware processes run forever co_stream_open(idata, O_RDONLY, INT_TYPE(STREAMWIDTH)); co_stream_open(odata, O_WRONLY, INT_TYPE(STREAMWIDTH)); 34 35 // Read values from the stream 36 while (co_stream_read(idata, &data128, sizeof(data128)) == co_err_none) 37 n = (int) data128 IF_SIM([0]); 39 co_memory_readblock(mem, 0, work, n * sizeof(int64)); 40data = 0; for (i = 0; i < n; i++)</pre> 4142 £ 43 inc = work[i]; 44 work[i] = data; 45 data += inc; 46 47 co_memory_writeblock(mem, 0, work, n * sizeof(int64)); data128 IF_SIM([0]) = data; 48 co_stream_write(odata, &data128, sizeof(data128)); 49 50 51 52 co_stream_close(idata); co_stream_close(odata); 53 54IF_SIM (break;) // Only run once for desktop simulation 55 } while(1); 56 } 57 < > For Help, press F1 20 14

Figure 42 - User defined funtion in memtest_hw.c

The configuration process defines the processes and ports that can be accessed by the user defined function **pass**. Line 58 is the configuration **config_memtest**. Line 60 thru 62 defined the available ports while line 64 & 65 define processes. Lines 67 shows the Impulse defined co_memory with type "mc0" and size MEMORY_SIZE. Lines 68 & 69 shows the Impulse defined co_streams of widths STREAMWIDTH and depth STREAMDEPTH (both defined in the header file). The configuration statement also defines the testbench process (line 71) as well as the user process (line 77). Line 86 is necessary when using co_memory.

💁 TextPad - [D:\ImpulseC_Pico\trunk\Examples\M5XX\Memtest\memtest_hw.c] Search View Tools Macros Configure Window Help Β× File Edit 🗅 🚄 🔚 🗐 🖨 🐧 国 ୍ଥ 🖻 🛍 🗅 🗠 | ቒ 📰 🗁 ୩ | 🏈 🎌 斜 🚱 | ଫୁ 🐢 🗛 | • 🗤 🕨 void config_memtest(void *arg) 59 £ 60 co_stream ipassdata; 61 co_stream opassdata; 62 co_memory aemem; 63 64 co_process pass_process; 65 co_process testbench_process; 66 aemem = co_memory_create("workmem", "mc0", MEMORY_SIZE); ipassdata = co_stream_create("idata", INT_TYPE(STREAMWIDTH), STREAMDEPTH); opassdata = co_stream_create("odata", INT_TYPE(STREAMWIDTH), STREAMDEPTH); 67 68 69 70 71 72 testbench_process = co_process_create("testbench", (co_function) Testbench, З. 73 74 75 ipassdata, opassdata, aemem); 76 77 pass_process = co_process_create("pass", (co_function) pass, 78 79 З. ipassdata, 80 opassdata, 81 aemem): 82 83 co_process_config(pass_process, co_loc, "PEO"); 84 } 85 86 co_architecture co_initialize(int param) 87 £ 88 return (co_architecture_create("memtest_arch", "Generic", config_memtest, (voi 89 } 90 < > 49 52

Figure 43 - Configuration in memtest_hw.c

9.2.3. Overview of memtest_sw.c

The software application provides stimulus and receives return data for post processing of the user defined functions. Line 20 defines a process "Testbench" which also has ports **ipassdata**, **opassdata** and **mem**. The port **opassdata** will send data to the user defined function. In this example, line 31 generates data to be written to external memory. Similar to the hardware process, streams must be open in order to read or write to them. Lines 38 & 39 open streams. Line 42 calls the Impulse API co_memory to write a block of data to external memory. Line 46 sends the number of bytes to be read by the hardware process while line 49 waits for data to be return by the hardware process. Line 55 reads data from external memory and that data is used to verify the data received on the input stream as shown in the "for" loop on line 58. Finally, the streams are closed as shown on lines 66 & 67.

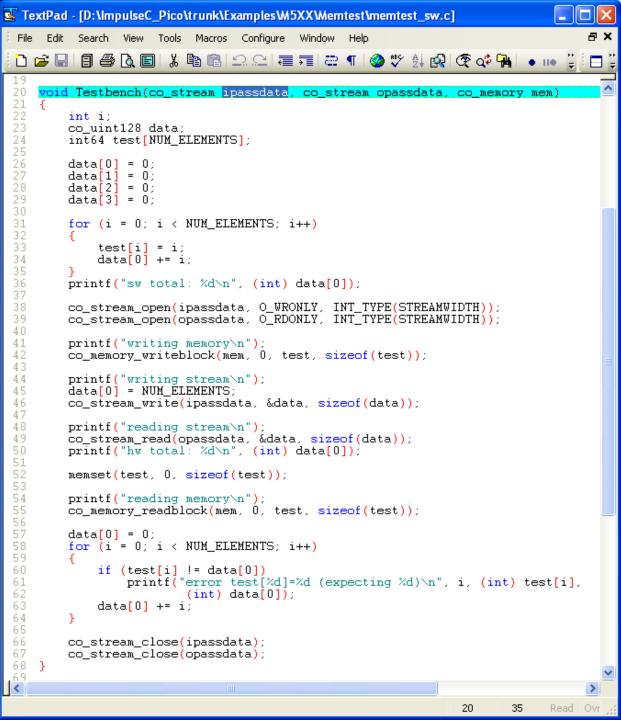


Figure 44 - User defined stimulus in memtest_sw.c

The main function is shown below. The first step for the software application is to load the FPGA bitfile using the Pico API Pico_LoadFPGA as shown on line 78. If the file does not exist or an error is encountered, then an error is reported. Otherwise the testbench is executed.

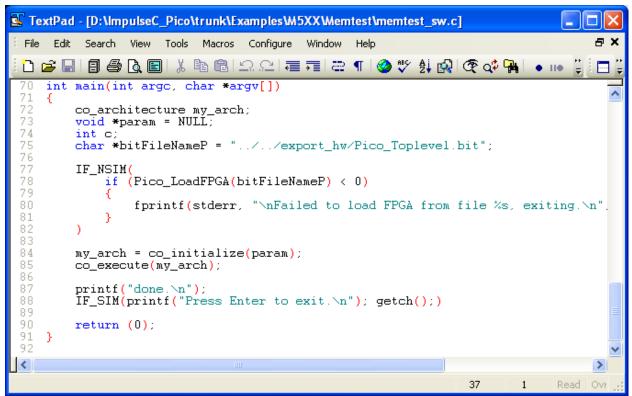


Figure 45 - Main program in memtest_sw.c

9.3. CoDeveloper Project Files

The Memtest example CoDeveloper project is made up of the following files:

- memtest.icProj CoDeveloper project file
- memtest_hw.c Source code for hardware process
- memtest_sw.c Source code for software processes
- memtest.h Header file that defines the width of the stream

When you define the width of the steam, you must make the changes in the header file as well as the in the memtest_hw.c file. The default example defines the steam to be the width of 128 bit data bus.

🗾 C:\	JmpulseC_Pico\Examples\M5XX\Memtest\memtest.h - Notepad++	_ D X
File	Edit Search View Encoding Language Settings Macro Run Plugins Window ?	X
	🔒 🖶 🗞 💫 🕹 🐇 🐚 🖍 ⊃ C # 🍇 🔍 🥄 🖾 🔤 💷 ୩ 🌉 🖉 🔸	
🔚 m	emtest h 📄 memtest_hw.c	
1		
2	11	
3	// Change BUFSIZE to match your desired stream depth size	
4	11	
5	// Change STREAMWIDTH to match your desired stream width	
6		
7	<pre>#define STREAMDEPTH 4 /* buffer size for FIFO in hardware */</pre>	
8	#define STREAMWIDTH 128	
9	#define NUM_ELEMENTS 8192	
10	<pre>#define MEMORY_SIZE (8192*sizeof(int64))</pre>	
11		
, C++ si	ource file length : 350 lines : 11 Ln : 1 Col : 1 Sel : 0 UNIX ANSI	INS

Figure 46 - Impulse C Header File

C:\Ir	npulseC_Pico\Examples\M5XX\Memtest\memtest_hw.c - Notepad++	X	
File E	idit Search View Encoding Language Settings Macro Run Plugins Window ?	X	
	9 🗄 🗣 🕉 🕼 🕼 ∦ 🕅 🕲 ⊃ ⊂ # 🦕 🔍 🥞 🖫 🕼 🗐 🚍 🗐 🗐 🗐 🖉 🖉 🖉		
2 minutes			
	ntest h 📄 memtest_hw.c	10	-1
1	//////////////////////////////////////	- f	i.
3	// function.		1
4	11		1
5	// See additional comments in memtest.h.		1
6			1
7			1
8	<pre>#include <stdio.h></stdio.h></pre>		1
9	#include "co.h"		1
10	<pre>#include "cosim_log.h" #include "moments b"</pre>		1
11	<pre>#include "memtest.h"</pre>		1
13	extern void Testbench(co stream ipassdata, co stream opassdata, co memory mem);		1
14			
15	///////////////////////////////////////		J
16	// This is the hardware process.	-	1
17	11		1
18	void pass(co_stream idata, co_stream odata, co_memory mem)		1
19	曱 (1
20	int i, n;		1
21	co_uint128 data128;		1
22	<pre>int64 data, inc; int64 work[NUM ELEMENTS];</pre>		1
23	Incor work[Non_Elements].		1
25	// EdT: co banks not available until CoDeveloper v3.70.d.13 or later		1
26	// Configure array to be two banks of 64 bits to maximize throughput from 128 bit memory.		1
27	<pre>// co array config(work,co banks,"2");</pre>		1
28			1
29	do		
30			1
31	// Hardware processes run forever		I
32	co_stream_open(idata, O_RDONLY, INT_TYPE(STREAMWIDTH));		I
33 34	<pre>co_stream_open(odata, 0_WRONLY, INT_TYPE(STREAMWIDTH));</pre>		I
35	// Read values from the stream		l
36	while (co_stream_read(idata, &data128, sizeof(data128)) == co_err_none)		I
37			
38	n = (int) data128 IF_SIM([0]);		
39	<pre>co_memory_readblock(mem, 0, work, n * sizeof(int64));</pre>		
40	data = 0;		I
41	for $(i = 0; i < n; i++)$		
42			
43	<pre>inc = work[i];</pre>		
44 45	work[i] = data;		
45	data += inc;		
47	<pre>co memory writeblock(mem, 0, work, n * sizeof(int64));</pre>		
48	<pre>data128 IF SIM([0]) = data;</pre>		
49	co stream write (odata, &data128, sizeof(data128));		
50	3 The second s		
51	<pre>co_stream_close(idata);</pre>		
52	co stream close(odata):	-	1
C sourc	e file length: 2317 lines: 90 Ln: 1 Col: 1 Sel: 0 UNIX ANSI	INS	

Figure 47 - ImpulseC Hardware File

9.4. Opening Project

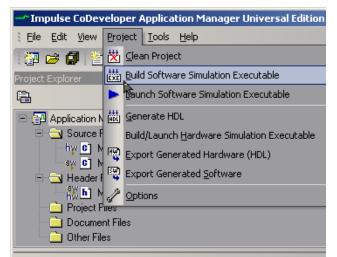
Open the CoDeveloper project file 'Memtest.icProj' by selecting and pressing 'Enter' or by double-clicking it:

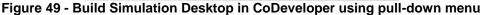
Look in: 🚺	Memtest	•	+ 🗈 💣 📰 +		
Name	*		Date modified	Туре	Si
📝 memtes	t.icProj		11/1/2012 8:58 AM	Impulse C Project	
< [m,			•
ìle name:	memtest.icProj			Open	
iles of type:	ImpulseC Project Files (*.io	- Desil		▼ Cance	. 1

Figure 48 - Opening a project in CoDeveloper

9.5. Building Desktop Simulation Executable

Build the desktop software simulation executable via the "Project" menu:





Or via toolbar:

iply]						
			E I	ri 🕰	1	3
	K	ftware Sim				

Figure 50 - Build Simulation Desktop in CoDeveloper using toolbar icon

Note the compiler output in the CoDeveloper IDE "Build" window:



Figure 51 - Output within the CoDeveloper IDE build window

9.6. Running Desktop Simulation Executable

Launch the desktop software simulation executable via "Project" menu:

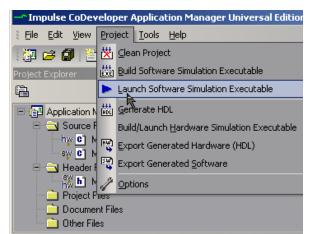


Figure 52 - Launch software simulation window using pull-down menu

Or via toolbar:



Figure 53 - Launch software simulation using toolbar icon

A command window will pop up in which the desktop simulation executable runs. Press "Enter" to exit:



Figure 54 - Pop-up window during desktop simulation

9.7. Project Setup Before Hardware/Software Generation and Export

Settings within the CoDeveloper IDE necessary for generating and exporting both hardware and software using this PSP are summarized below:

- Platform Support Package: "Pico M-501 Linux (VHDL)"
- Hardware export directory: <user hardware export directory>
- Software export directory: <user software export directory>
- Unsupported settings include:
 - Generate dual clocks (must be unchecked)
 - Active-low reset (must be unchecked)
 - Include floating point library (must be unchecked)

An example of these settings as it appears in the Memtest example:

Generate dual clocks
Generate active-low reset
Use std_logic types for VHDL interfaces
✓ Do not include co_ports in bus interface
Additional library options:
Output Directories Hardware build directory:
hw
Software build directory:
sw
Hardware export directory:
export_hw
Software export directory:
export_sw

Figure 55 - Project setup to pick Platform Support Package

9.8. Generating Hardware

Generate hardware via "Project" menu:

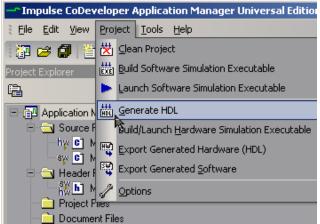


Figure 56 - Generate HDL using pull-down menu

Or via toolbar:

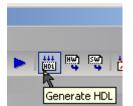


Figure 57 - Generate HDL using toolbar icon

Final results will appear in the directory specified during project setup in "Hardware build directory". Note the final output in the CoDeveloper IDE's "Build" window:

uild	
apyright 2002-2009, Impulse Accelerated Technologies, Inc.	
rights reserved.	
enerating pe0/pass	
omponent generation complete	
Software activated	
://mpulse/CoDeveloper3/bin/impulse_arch" "-aC:/Impulse/CoDeveloper3/Architectures/pico_m501_linux_vhdl.xml" -no_port_bus_connect -swdirsw -	
ss "memtest comp.vhd memtest top.vhd " srcs "memtest hw.c memtest h" memtest xic hw/memtest top.vhd	
pulse C HDL Design Generator	
pyright 2002-2012, Impulse Accelerated Technologies, Inc.	
ndhs reserved.	
ng is received. ading C:/Impulse/CoDeveloper3/Architectures/pico_m501_linux_vhdl.xml	
saling C:/Impulse/CoDeveloper3/Architectures/Pico/MS0x_Linux/bus50x.xml	
ading C/Impulse/CoDeveloper3/Architectures/Fico/MS0L Linux/asiddr.arml	
ading C:/mpulse/CoDeveloper3/Architectures/HOLAtaret.xml	
adang C./mpulse/Cobeveloper3/Admittatures/HDL/Agertanii	
adang C./mipulae/Cobevelopers/Achitectures/VHDL/Amitov4tech.xmi	
ading members xic	
pFile=pico_m501_linux_vhdl.xml	
vard=m501	
nnections2i=memory mc0 axiddr {} mc0 {}	
nnections2i=stream in p_testbench_ipassdata 128 picobus sp 🖁 idata 🕼	
nnections2i=stream out p_testbench_opassdata 128 picobus sp {} odata {}	
nnections2i=stream in config 32 picobus sp {} config {}	
nning picoCustomize Ucf	
nnections = {memory mc0 axiddr {} mc0 {} 0} {stream in p_testbench_ipassdata 128 picobus sp {} idata {} 1} {stream out p_testbench_opassdata 128	
cobus sp {} odata {} 1} {stream in config 32 picobus sp {} config {} 2}	
n = memory mc0 axiddr {} mc0 {} 0, 0	
n = stream in p_testbench_ipassdata 128 picobus sp {} idata {} 1, 1	
n = stream out p_testbench_opassdata 128 picobus sp (} odata {} 1, 1	
n = stream in config 32 picobus sp {} config {} 2, 2	
esign generation complete	
mod -R +rw hw	
cdir sw	
://mpulse/CoDeveloper3/bin/impulse lib" "-aC:/Impulse/CoDeveloper3/Architectures/pico m501 linux vhdlxml" -hwdirhw files "memtest sw.c"	
emtest xic sw/co init.c	
pulse C Software Interface Generator	
pyright 2002-2012. Impulse Accelerated Technologies. Inc.	
indits reserved.	
ng is occurred. Jading C:/Impulse/CoDeveloper3/Architectures/pico m501 linux vhdl.xml	
ading C://mpulse/CoDeveloper3/Architectures/Pico/M50x Linux/cpu50x.xml	
daming C:/Impulse/CoDeveloper/Architectures/VHDL/Generic/Generic/system.xml	
ading emittest xic	
adaing internets sw.c.; do cp \$i sw; done	
in members bir do c \$ sw: done	
nan menesian, do op ar sw. done	
====== Build of target 'build' complete =======	
build of talget build complete =======	
	-
Build 🖼 Find in Files 🗁 System	

Figure 58 - Build window output

9.9. Exporting Hardware

Export hardware via "Project" menu:

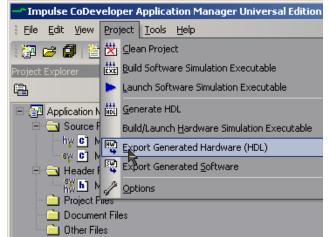


Figure 59 - Export Generated Hardware (HDL) using pull-down menu

Or via toolbar:



Figure 60 - Export Generated Hardware (HDL) using toolbar icon

Final results will appear in the directory specified during project setup in "Hardware export directory". Note the final output in the CoDeveloper IDE's "Build" window:

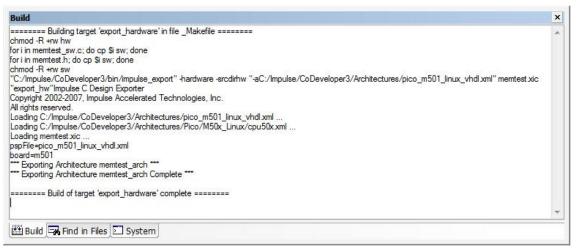


Figure 61 - Build window output

9.10. Compiling FPGA in Xilinx ISE 13.4

After exporting hardware, under the specified hardware export directory will be a directory structure that includes all necessary files for building the FPGA binary.

Organize 🔻	Include in library 🔻 Share with 🔻	Burn New folder		= 🕶 🗖 🌔
*	Name	Date modified	Туре	Size
	鷆 hdl	11/6/2012 6:35 PM	File folder	
	build_bit_file.tcl	11/2/2012 7:25 PM	TCL File	4 KB
	🚳 build_m501lx240_memtest_arch.bat	11/6/2012 6:35 PM	Windows Batch File	1 KB
	build_m501lx240_memtest_arch.sh	11/6/2012 6:35 PM	SH File	1 KB
=	customizeM501xise.tcl	11/6/2012 6:35 PM	TCL File	3 KB
	🚳 gen_ise_files.bat	11/6/2012 6:35 PM	Windows Batch File	1 KB
	gen_ise_files.sh	11/6/2012 6:35 PM	SH File	1 KB
	gen50x_xise.tcl	11/2/2012 7:25 PM	TCL File	27 KB

Figure 62 - Compiling FPGA in Quartus directory structure

At this point either Windows or Linux may be used to produce the bitfile running Xilinx ISE either through the GUI or from command line. Windows and Linux both use the following similar steps. Choose the one that suits your needs.

Windows:

- 1. Generate the ISE project files by running "gen_ise_files.bat"
- 2. Build the bit file using Xilinx ISE either by:
 - a. Running the generated "build_m50*.bat" file
 - b. Launching the Xilinx ISE GUI opening the newly built "m50*.xise" file

Linux:

- 1. Generate the ISE project files by running "gen_ise_files.sh"
- 2. Build the bit file using Xilinx ISE either by:
 - a. Running the generated "build_m50*.sh" file
 - b. Launching the Xilinx ISE GUI opening the newly built "m50*.xise" file

These methods will generate the necessary bitfile to be loaded on the Host System and are outlined in the sections that follow.

9.10.1. Building bitfile under Windows

9.10.1.1. Generate Xilinx ISE Project Files

First generate the ISE 13.4 project by executing "gen_ise_files.bat". This process will create an ISE project file as well as all the necessary HDL files and corgen components used to generate the bitfile when building in ISE.

- 1. Open a command window
- 2. Change directories to "export_hw"
- 3. Execute "gen_ise_files.bat" as shown in Figure 58.
- 4. Verify that the files were created successfully by viewing the end of log file, as shown in Figure 59. "Successful!" should be the last line in the log file.

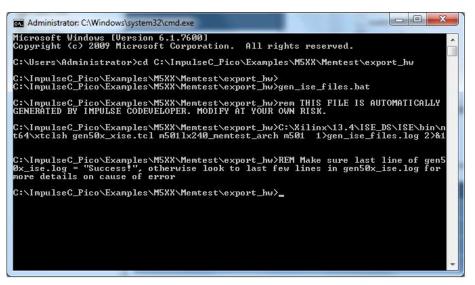


Figure 63 - Generate ISE project files

Administrator: C:\Windows\system32\cmd.exe
source/pcie_gtx_v6.v xfile_add: add glob file=firmware/m501/coregen-LX240T/v6_pcie_v2_4_8lane_gen2/ source/pcie_pipe_lane_v6.v
xfile_add: add glob file=firmware/m501/coregen-LX240T/v6_pcie_v2_4_8lane_gen2/ source/pcie_pipe_misc_v6.v
xfile_add:
xfile_add:
xfile_add: add glob file=firmware/m501/coregen-LX240T/v6_pcie_v2_4_8lane_gen2/ source/pcie_trn_128.v
xfile_add: add glob file=firmware/m501/coregen_LX240T/v6_pcie_v2_4_8lane_gen2/ source/pcie_upconfig_fix_3451_v6.v
xfile_add: add glob file=firmware/m501/coregen-LX240T/v6_pcie_v2_4_8lane_gen2/ source/sync_fifo.v
xfile_add: add glob file=firmware/m501/coregen-LX240T/v6_pcie_v2_4_8lane_gen2/ source/trn_rx_128.v xfile_add: add glob file=firmware/m501/coregen-LX240T/v6_pcie_v2_4_8lane_gen2/
source/trn_tx_128.v xfile_add: add glob file=firmware/m501/coregen-L%240T/v6_pcie_v2_4_8lane_gen2/
source/v6_pcie_v2_4_8lane_gen2.v xfile_add:add_file=firmware/m501/src/M501_LX240T_DDR3.ucf
xfile_add: add file=firmware/m501/src/M501_LX240T_PCIe.ucf xfile_add:End
Configuring project properties closing project
C:\ImpulseC_Pico\Examples\M5XX\Memtest\export_hw>

Figure 64 - Expected Gen_Ise_File log report

There are two method to produce the bitfile. The first is to launch the Xilinx ISE GUI and generate the bitfile. The second method it to execute the build script in the **export_hw** directory. Both method will generate the necessary bitfile to be loaded on the Host System.

9.10.1.2. Compiling FPGA in Xilinx ISE GUI

Launch the Xilinx ISE GUI and open the ISE project located in the **export_hw** directory. Select Pico_Toplevel in the *Hierarchy* window and double-click "generate programming file" in the *Processes* window.

NOTE: Xilinx ISE 13.4 will ask the user to regenerate three fifos (a one time process). The following figures illustrate initial project launch, regenerate the three corgen fifos, and final output with timing score equal to zero.

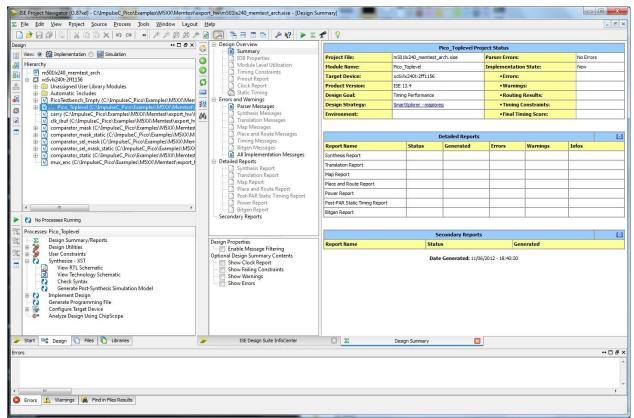


Figure 65 - Initial Xilinx ISE 13.4 GUI screen

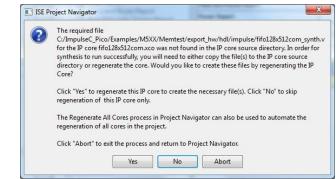


Figure 66 - Xilinx ISE 13.4 regenerate IP core fifo128x512com

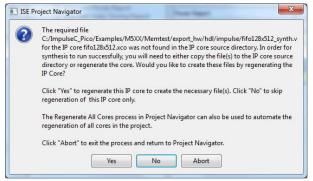


Figure 67 - Xilinx ISE 13.4 regenerate IP Core fifo128x512

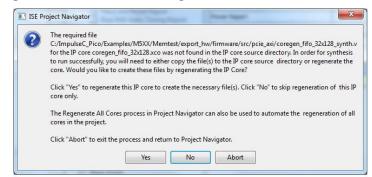


Figure 68 - Xilinx ISE 13.4 regenerate IP Core coregen_fifo_32x128

Design Overview Summary Summary Stoppenties Module Level Utilization Timing Constraints Pinout Report	Project File:		oject Status (1	11/05/2012 - 20:20:05)			
Module Level Utilization Timing Constraints Pinout Report		mS01ly240 membest arch vise						
Iming Constraints Pinout Report	Hodule Name:			Parser Errors:		No Errors Programming File Generated No Errors 8125 Warnings (8174 new)		
- Pinout Report	Hodule Name: Pico_Toplevel Target Device: xc6v/x240t-2ff1156 Product Version: 25E 13.4		Implementation State: • Errors: • Warnings:		Pro			
					No			
Clock Report					81			
Static Timing	Design Goal:	Timing Performance	Routing Results:		AL	Al Signals Completely Routed		
Parser Messages	Design Strategy:	SmartXplorer - mapioreg		Timing Constraints:	AL	Constraints Met		
	Environment:	System Settings	•1	Final Timing Score:	0	Timing Report)		
Place and Route Messages		Device Ullik	ration Summar	rv.				
	Slice Logic Utilization	U	sed	Available	Utilization	Note(s)		
All Implementation Messages			27.45		-			
raport, hvir firmværindi (2009) Tang (3, 15, 200 Marketti neport, hvir firmværindi (2009) Tang MSXX.Merketti neport, hvir firmværindi (2009) Tang MSXX.Merketti neport, hvir firmværindi (2009) Tang MSXX.Merketti neport, hvir firmværindi (2009) Tang Montesti neport, hvir firmværindi (2007) Tang Montest	Number used as Filo Floos							
	Number used as Latches			<u></u>	-			
Map Report	Number used as Latch-thrus			0	-			
	Number used as AND/OR logics							
- D Power Report					1	11%		
Bitgen Report								
Secondary Reports		,						
WebTalk Log File						-		
_								
	1							
			2,49	1 58.400	i i	4%		
Number used as Dual Port RAM 1,114								
	1							
Certional Decision Summary Contents								
F Show Clock Report					-			
		AM						
ET aven even						-		
		2000		50 L		-		
	Number using OS and O6							
	Number used exclusively as route-thrus							
					1			
	Number with same-slice carry load			· · · · · · · · · · · · · · · · · · ·				
		(0	1				
Design Summary (Programming File Generated)	8						
	Parer Meisage Price American Strength Price A	Dering Respect Dering Respect Respect	Porzer Missige Porzer M	Deckson Farborgen Deckson Farborgen	Decks Protection Decks Protection Training Constraints Training Constraints Protection Protection Protection State and protection Protection Protection	Branz Meinages System Research Sy	 Decise Storkboy: Storkboy: Storkboy:	

Figure 69 - Xilinx ISE 13.4 with timing score = 0

Once ISE 13.4 has completed, a bitfile "pico_toplevel.bit" will be present in the **export_hw** directory.

9.10.1.3. Compiling FPGA in Xilinx ISE using Command Line

In the top directory there will be the batch file "build_memtest_arch.bat" used to automatically run Xilinx ISE 13.4 to create the necessary .bit file used to program the M501 FPGA.

- 1. Open a command window
- 2. Execute "build_m501lx240_memtest_arch.bat".

A command window will appear showing the FPGA build process (primarily made up of many, many info and warning messages). Compile time will vary by machine depending upon project size. When completed successfully, something similar to the following will appear and a bitfile "pico_toplevel.bit" will be present in the **export_hw** directory.

9.10.2. Building bitfile under Linux

Please follow the steps outlined in the Passthrough example, **Sections 8.9.1 – Building bitfile under Linux**, in order to build the bitfile for the Memtest example under Linux. It is left to the user, as an exercise, to successfully build the bitfile.

9.11. Exporting Software

The software application to be run on the host computer (with the M501 installed with it's drivers) can be exported in CoDeveloper.

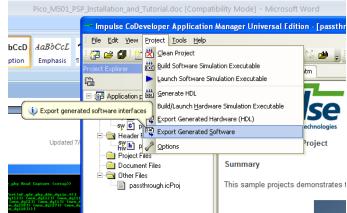


Figure 70 - Export Generated Software

Once completed without error in the Build window, it should be noted that the software code will be written to a newly created directory. The user can modify the target directory name. In this example, export_sw contains the exported software files.

Build	×
<pre>sersesses Building target 'export_software' in file _Makefile ====================================</pre>	*
🕮 Build 🖼 Find in Files 🗈 System	-

Figure 71 - Build window output

Organize 🔻	Include in library 🔻	Share with 🔻	Burn	New folder			•== •	
🚖 Fave	Name		Date	e modified	Туре	Size		
	🍶 lib		11/6	/2012 6:42 PM	File folder			
詞 Libr	co_init.c		11/6	i/2012 6:34 PM	C Source	2 KE	3	
De	🗋 Makefile		11/6	i/2012 6:34 PM	File	1 KE	3	
🚽 М	🔟 memtest.h		11/6	j/2012 6:42 PM	C/C++ Header	1 KE	3	
🔄 Pi	c memtest_sw.c		11/6	i/2012 6:42 PM	C Source	2 KE	3	

Figure 72 - Exported software directory

9.12. Programming the FPGA

The compiled software application in the "export_sw" directory will program the FPGA on the M501. Please continue to the next section.

9.13. Running Target Executable on the Host System

The Host System will need the following files:

- 1. Compiled bitfile output from ISE 13.4
- 2. Input data file
- 3. Application software source code
- 4. Makefile to compile the software

Please copy the following files and directory over to the Host System (figure 68). Create a folder (ie Pico/Memtest) and copy the following to that folder.

- 1. **export_hw** directory which includes "pico_toplevel.bit" bitfile
- 2. **export_sw** directory which includes the Makefile and software application source code

Organize	 Open Burn New folder 			·	1 (
@ ^	Name	Date modified	Туре	Size	
	👩 export_hw	11/6/2012 6:40 PM	File folder		
	👩 export_sw	11/6/2012 6:42 PM	File folder		
	👩 hw	11/6/2012 6:34 PM	File folder		
	Ø sw	11/6/2012 6:34 PM	File folder		
	🚯 _Exe_memtest.bat	11/6/2012 6:33 PM	Windows Batch File	1 KB	
	🚳 _Make_memtest.bat	11/6/2012 6:42 PM	Windows Batch File	1 KB	
	_Makefile	11/6/2012 6:42 PM	File	1 KB	
	[6] _Makefile.defs	11/6/2012 6:31 PM	DEFS File	1 KB	
=	👩 banktest.exe	11/6/2012 6:32 PM	Application	733 KB	
=	🔊 memtest.h	11/1/2012 8:58 AM	C/C++ Header	1 KB	
	👩 memtest.i	11/6/2012 6:34 PM	Preprocessed C/C	17 KB	
	🕢 memtest.icProj	11/1/2012 8:58 AM	Impulse C Project	2 KB	
	memtest.pk0	11/6/2012 6:34 PM	PK0 File	29 KB	
	memtest.pk1	11/6/2012 6:34 PM	PK1 File	17 KB	
	👩 memtest.pky	11/6/2012 6:34 PM	PKY File	17 KB	
	memtest.sic	11/6/2012 6:34 PM	SIC File	15 KB	
	memtest.smd	11/6/2012 6:34 PM	SMD File	11 KB	
	👩 memtest.snt	11/6/2012 6:34 PM	SNT File	29 KB	
	👩 memtest.xhw	11/6/2012 6:34 PM	XHW File	10 KB	
	memtest.xic	11/6/2012 6:34 PM	XIC File	9 KB	
	🔊 memtest_hw.c	11/1/2012 8:58 AM	C Source	3 KB	
	👌 memtest_hw.o	11/6/2012 6:32 PM	O File	368 KB	
	memtest_sw.c	11/4/2012 2:22 PM	C Source	2 KB	
	🔄 memtest_sw.o	11/6/2012 6:32 PM	O File	362 KB	
	b memtest_sw.o	11/6/2012 6:32 PM	O File	362 KB	

Figure 73 - Files & Directories to be copied to the Host System

The software application will load the FPGA every time it is executed and report the results upon completion. Please note that the process of loading the FPGA may take up

to 20 seconds due to the driver re-start sequence after the FPGA bitfile has been transferred to the card.

- 1. Open a terminal window.
- 2. Navigate to the location of your copied files and directories:

(ie. cd Pico/Memtest/export_sw/software).

- 3. Run "make"
- 4. Execute the generated application: "./memtest_arch"

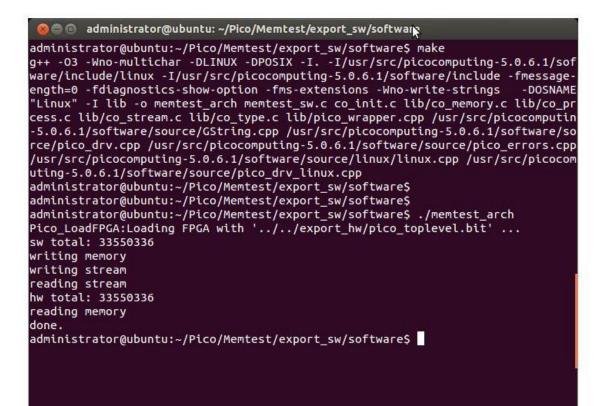


Figure 74 - Exported SW executed on target platforn