

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

Version 1.0.1 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	
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	54
9.7.	Project Setup Before Hardware/Software Generation and Export	55
9.8.	Generating Hardware	
9.9.	Exporting Hardware	
9.10.	Compiling FPGA in Xilinx ISE 13.4	
9.10	0.1. Building bitfile under Windows	59
9.10	0.2. Building bitfile under Linux	
9.11.	Exporting Software	
9.12.	Programming the FPGA	66
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	).8
Figure 2 - Impulse C Header File with 64 bit co_stream	
Figure 3 - ImpulseC Hardware File with 64 bit co_stream	16
Figure 4 - Opening a project in CoDeveloper	17
Figure 5 - Build Simulation Desktop in CoDeveloper using pull-down menu	18
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	
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	
Figure 24 - Xilinx ISE 13.4 with timing score = $0$	
Figure 25 - Build bitfile in ISE 13.4	
Figure 26 – Xilinx ISE 13.4 compile log file – timing score	
Figure 27 - Generate ISE project files	33
Figure 28 - Expected gen50x_xise.log report	
Figure 29 - Select the ISE project	
Figure 30 - Initial project status after loading project	
Figure 31 - Regenerate fifo128x512	
Figure 32 - Regenerate coregen_fifo_32x128	
Figure 33 - Place and Route and bitfile generation with timing score equal to zero	.36
Figure 34 - Ubuntu Xilinx ISE 13.4 Command Line output	
Figure 35 - Xilinx ISE 13.4 results file with timing score equal to zero	
Figure 36 - Export Generated Software	
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 40 - Impulse C fieader File	
Figure 48 - Opening a project in CoDeveloper	
Figure 49 - Build Simulation Desktop in CoDeveloper using pull-down menu	
righte re - Daild Cirraiditori Desittop in CoDeveloper dailig puil-down mend	

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	54
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	55
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	58
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	
Figure 68 - Xilinx ISE 13.4 regenerate IP Core coregen_fifo_32x128	62
Figure 69 - Xilinx ISE 13.4 with timing score = 0	
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

# 3.0 Revision History

Date	Version	Description	Author
11/3/2012	1.0	Initial Creation	Shaumil Dave
11/10/2012	1.1	Added Memtest description	Shaumil Dave

# 4.0 Overview

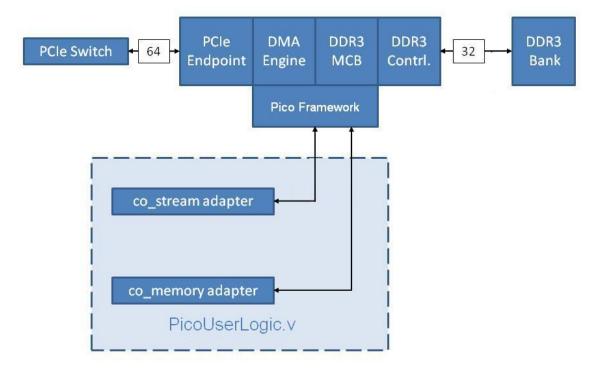
This user guide covers the CoDeveloper Platform Support Package (PSP) for the Pico Computing M501 module (referred hereon simply as "M501"). Highlights for this PSP include:

- 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
- 2. 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:
  - 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 <a href="mailto:support@impulsec.com">support@impulsec.com</a> 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.

<b>S</b> 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

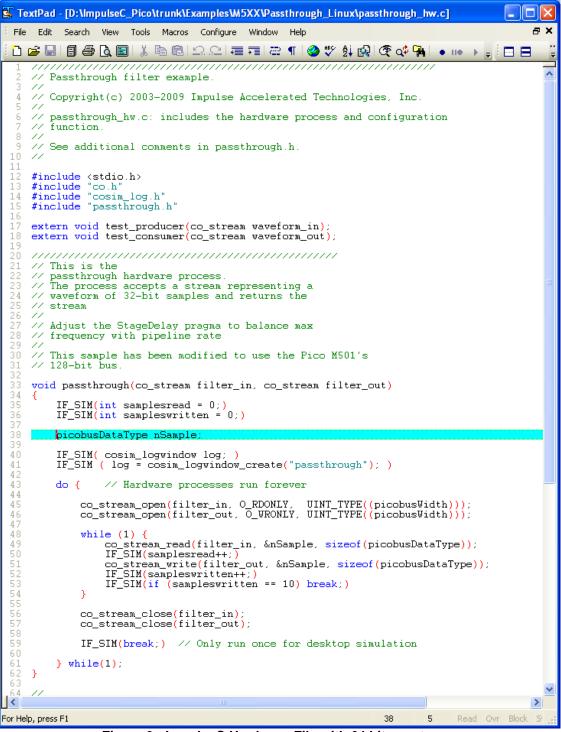


Figure 3 - ImpulseC Hardware File with 64 bit co\_stream

# 8.3. Opening Project

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

Look in:	Passthrough_Linux	-	← 🗈 💣 📰▼		
Name	^		Date modified	Туре	Si
🜛 Readme	Files		11/4/2012 8:04 AM	File folder	
🛃 passthrough.icProj		11/4/2012 8:04 AM	Impulse C Project		
•		m			,
∢ File name:	passthrough.icProj	m		0	lpen

Figure 4 - Opening a project in CoDeveloper

## 8.4. Building Desktop Simulation Executable

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

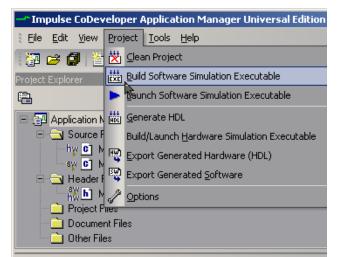


Figure 5 - Build Simulation Desktop in CoDeveloper using pull-down menu

Or via toolbar:



Figure 6 - Build Simulation Desktop in CoDeveloper using toolbar icon

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

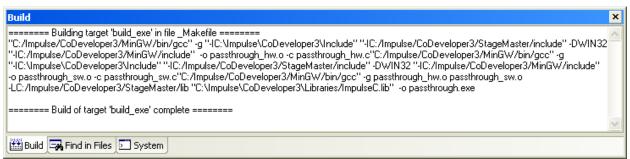


Figure 7 - Output within the CoDeveloper IDE build window

## 8.5. Running Desktop Simulation Executable

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

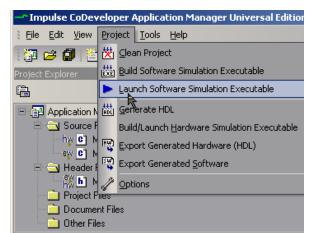


Figure 8 - Launch software simulation window using pull-down menu

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.	<u>^</u>
Consumer reading data	
Sending waveform	
test_producer:Writing value 1	
test_producer:Writing value 2 test_producer:Writing value 3	
Filtered value: 1	
Filtered value: 2	
test_producer:Writing value 4	
test_producer:Writing value 5	
Filtered value: 3	
Filtered value: 4	
test_producer:Writing value 6 test_producer:Writing value 7	
test_producer:Writing value 8	
Filtered value: 5	
test_producer:Writing value 9	
Filtered value: 6	
test_producer:Writing value a	
Filtered value: 7 Filtered value: 8	
Finished writing waveform.	
Filtered value: 9	
Filtered value: a	
Consumer read 10 waveform datapoints	
Application complete. Press the Enter key to continue.	
inprication complete. Hess the inter key to continue.	
Figure 40. Den um winden, during destates simulation	

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:

uild Simulate Generate System Registration	
Platform Support Package:	
Pico M-501 Linux (VHDL)	
Hardware Optimization and Generation	
Enable constant propagation	Generate dual clocks
✓ Scalarize array variables	Generate active-low reset
Relocate loop invariant expressions	Use std_logic types for VHDL interfaces
	Do not include co_ports in bus interface
Additional optimizer options:	Additional library options:
· · · · · ·	
Floating Point Options	Output Directories
Include floating point library	Hardware build directory:
Include co_math library	hw
Enable floating point optimization	Software build directory:
☐ Allow double-precision types and operators	sw
Use higher latency, faster clock operators	Hardware export directory:
	export_hw
Enable floating point accumulators	Software export directory:
Use extended precision accumulators	export_sw

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:

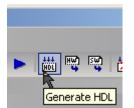


Figure 13 - Generate HDL using toolbar icon

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 - [passthro	Image:				
Ele Edit View Project Tools Help					
	🛃 🖂 🕨 🖏 🔍 🤁 🥒 🔷 💂				
e HDL for platform target					4 Þ 🖛 🗙 .
				a stream open to the O	ROOKLY[1 // Open stream for read
Generate HDL     Generate HDL     Generate HDL     Generate HDL	_				
-hw Close Robert and a state of the state of					1754 1+ )   (m in, iku[i](k]) 5= co err_ecal (
sw D pad					
high pad Options					
Project Files					
Other Files					
This sample projects demonstrat	es the basics of streams-based communications.				
					E
(sim) (sim) (sim)					
Image: Subject of the project demonstrates the basics of streams-based communications.         Image: Subject field         Image: Document field field         Image: Document field fie					
	Character	Pass-through	Test		
	generator	filter			
					-
Left relation tory biblis Software Smulton Executable unch Software Smulton Executable unch Software Smulton Executable bigeness HD. Doport Generated lativiane (HDL) popt Generated lativiane (HDL) Doport Generated lativiane (HDL) popt Generated lativiane (HDL) popt Generated lativiane (HDL) Doport Generated lativiane	# ×				
Loading C:/Impulse/CoDeveloper3/Archite	rctures/Pico/M50x_Linux/bus50x.xml				*
Bit       Summary         This sample projects demonstrates the basics of streams-based communications.         Summary         This sample projects demonstrates the basics of streams-based communications.         Stream         Bit         Locker Fie         Bit         Locker Collevelope: 3/Politecture: Phone MSDL Linux/wedghterd         Bit         Locker Collevelope: 3/Politecture: Phone MSDL Linux/wedghterd         Locker Collevelope: 3/Politecture: Phone MSDL Li					
pspFile=pico_m501_linux_vhdlxmboard=m	501				
connections2=stream out p consumer pro	cess waveform out 64 picobus sp {} waveform out {}				
con = stream in p producer process wave	eform in 64 picobus sp () waveform in () 1, 1	om_in () 1) (stream out p_consum	her_process_waveform_out t	4 picobus sp () waveform_out () 1)	
Design generation complete	.veform_out 64 picobus sp () waveform_out () 1, 1				
mkdir sw					
Impulse C Software Interface Generator		ndl.xml" -hwdirhw files "passthroug	h_sw.c" passthrough xic sw.	/co_init.c	
All rights reserved.	120.00				
Loading C:/Impulse/CoDeveloper3/Archite	ectures/Pico/M50x Linux/cpu50x.xml				
Loading C:/Impulse/CoDeveloper3/Archite Loading passthrough xic	ectures/VHDL/Generic/Generic/system.xml				
for i in passthrough_sw.c; do cp \$i sw; don	e				
Build of target 'build' complete -					-
Build - Find in Files - System					
Generate HDL for platform target	P				OVR NUM

Figure 14 - Build window output

## 8.8. Exporting Hardware

Export hardware via "Project" menu:

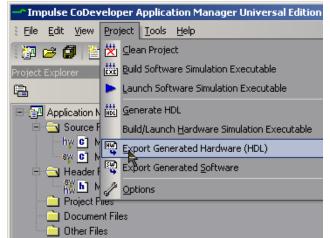


Figure 15 - Export Generated Hardware (HDL) using pull-down menu

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:

<pre>bit Strives Provid Lobe House Underset Educations - grantworks - grant works - grant - gr</pre>	- 0 <b>- X</b>	
Print Tools Heb         We find type		
	vare Simulation Executable	4 Þ 🔻 X
Effer Edit: Yiew       Project: Tools: Help         Image: Excore       Image: Excore         Ima	oftware Simulation Executable	
Pie Edit Ver       Protect Look Heb         We den Protect       We have Smulton Executable         We den Software Smulton Executable       We have Smulton Executable         We den Software Smulton Executable       Secret Below         We denote Software       Secret Below         We denot Software       Secret Below	<pre>0 RDMLP() // Open stream for read (log, "SDML Reynchedule:\n");</pre>	
Print Look Heb         We find type	tellog, "Awading idx: "sd\n",k);	
hy pas He Export Ger	nerated Hardware (HDL)	<pre>intering to the second se</pre>
	nerated Software	
bill pas / Options		
Project Files		
	Summary	
Coner mes	This sample projects demonstrates the basics of streams-based communications.	
		11 ( ) ( ) ( ) ( ) ( ) ( ) ( ) ( ) ( ) (
Pie Edit Ver       Preixt: Look jeb         With Settingers       With Settingers         With Settingers       Settingers         Settingers       Settingers         With Settingers<	SIM SIM	
	Character Pass-through Test	
Comparing Support Series	generator niter consumer	
	Will Runch Hardware (RUL)         Deport Generated Software         Proprior         Perior Generated Software         Proprior         Perior         It is sample projects demonstrates the basics of streams-based communications.         It is sample projects demonstrates the basics of streams-based communications.         It is sample projects demonstrates the basics of streams-based communications.         It is sample projects demonstrates the basics of streams-based communications.         It is sample projects demonstrates the basics of streams-based communications.         It is sample projects demonstrates the basics of streams-based communications.         It is sample projects demonstrates the basics of streams-based communications.         It is sample projects demonstrates the basics of streams-based communications.         It is sample projects demonstrates the basics of streams-based communications.         It is sample projects demonstrates the basics of streams-based communications.         It is sample projects demonstrates in the Makelle second- ter of the same streams-based communications.         It is sample projects demonstrates in the Makelle second- ter of the same streams-based communications.         It is patientingly. w. t. do s B sin . dore ter of the same streams-based communications.         It is patientingly. w. t. do s B sin . dore ter of the same streams-based communications.         It is pating to prove the concluster Provides. It. Mak	
		-
	Building target 'export hardware'in file Makefie	
	chmod -B +tw hw	
	for i in passthrough h; do cp \$ sw: done	
	"C:/Impulse/CoDeveloper3/bin/impulse export" hardware -srcdirhw "-aC:/Impulse/CoDeveloper3/Architectures/pico m501 linux vhdi.xml" passthrough xic "export hw"	
	Impulse C Design Exporter Considert 2002/2007. Impulse Accelerated Technologies. Inc.	
	All rights reserved	
	Loading C:/Impulse/CoDeveloper3/Architectures/Pico/M50x_Linux/cpu50x.xml	
	Loading passfireugh xic cost of the second se	
	board=m501	
	ardware Smulation Executable ad Hardware (HDD) ad Software ad Mardware (HDD) ad Software ad Mardware (HDD) ad Software ad Mardware (HDD) ad Software bis sample projects demonstrates the basics of streams-based communications.	
	Cook 操作D     Here Ran Project     With Software Strukton Executable     Work ware Strukton Executable     work Software Software     work Softwar	
		*
Encode and and have a second	j 🛄 Buid j=w Find in Files (= 1 System)	
Export generated hardware (HDL)		OVR NUM

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	lseC_Pico + Examples + M5XX + Passthrough_L	.inux ▶ export_hw ▶	<b>▼</b> 49	Search export_hw		
Organize 👻 Includ	le 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		
▷ 🔄 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).

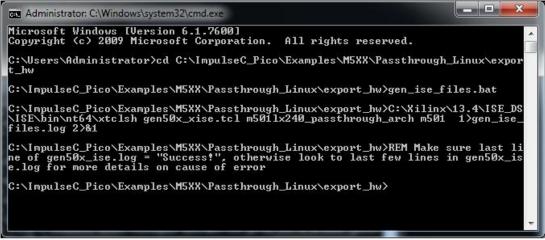


Figure 19 - Generate ISE project files

🔤 Administrator: C:\Windows\system32\cmd.exe
source/axi_basic_rx_pipeline.v
xfile_add:    add glob file=firmware/m501/coregen-LX240T/v6_pcie_v2_4_8lane_gen2/ source/axi_basic_top.v
xfile_add: add glob file=firmware/m501/coregen-LX240T/v6_pcie_v2_4_8lane_gen2/
source/axi_basic_tx.v xfile_add:    add glob file=firmware/m501/coregen=LX240T/v6_pcie_v2_4_8lane_gen2/
source/axi_basic_tx_pipeline.v xfile_add:
source/axi_basic_tx_thrtl_ctl.v
xfile_add:    add glob file=firmware/m501/coregen-LX240T/v6_pcie_v2_4_8lane_gen2/ source/gtx_drp_chanalign_fix_3752_v6.v
xfile_add:     add glob file=firmware/m501/coregen-LX240T/v6_pcie_v2_4_8lane_gen2/ source/gtx_rx_valid_filter_v6.v
xfile_add: add glob file=firmware/m501/coregen-LX240T/v6_pcie_v2_4_8lane_gen2/
source/gtx_tx_sync_rate_v6.v xfile_add:    add glob file=firmware/m501/coregen-LX240T/v6_pcie_v2_4_8lane_gen2/
source/gtx_wrapper_v6.v
xfile_add:    add glob file=firmware/m501/coregen=LX240T/v6_pcie_v2_4_8lane_gen2/ source/pcie_128_if.v
xfile_add:     add  glob  file=firmware/m501/coregen=LX240T/v6_pcie_v2_4_8lane_gen2/ source/pcie_2_0_v6.v
xfile_add: add glob file=firmware/m501/coregen-LX240T/v6_pcie_v2_4_8lane_gen2/
source/pcie_brams_v6.v xfile_add:    add glob file=firmware/m501/coregen-LX240T/v6_pcie_v2_4_8lane_gen2/
source/pcie_bram_top_v6.v xfile_add:    add glob file=firmware/m501/coregen-LX240T/v6_pcie_v2_4_81ane_gen2/
source/pcie_bram_v6.v
xfile_add:
xfile_add: add glob file=firmware/m501/coregen=LX2401/vb_pcie_v2_4_81ane_gen2/
source/pcie_clocking_v6.v xfile_add:    add glob file=firmware/m501/coregen-LX240T/v6_pcie_v2_4_8lane_gen2/
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:
xfile_add:    add glob file=firmware/m501/coregen=LX240T/v6_pcie_v2_4_8lane_gen2/ source/pcie_pipe_v6.v
xfile_add:
source/pcie_reset_delay_v6.v 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_81ane_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-LX240T/v6_pcie_v2_4_8lane_gen2/
source/v6_pcie_v2_4_8lane_gen2.v xfile_add:
xfile_add:add_file=firmware/m501/src/M501_LX240T_PCIe.ucf
xfile_add:End Configuring project properties
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.

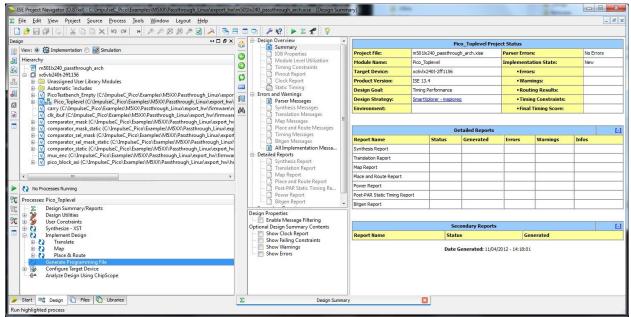


Figure 21 - Initial Xilinx ISE GUI screen

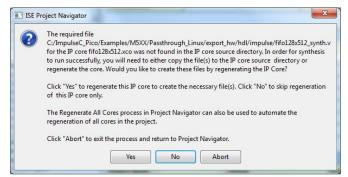


Figure 22 - Xilinx ISE regenerate IP core fifo128x512

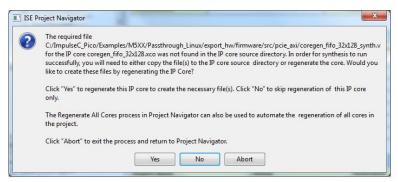


Figure 23 - Xilinx ISE regenerate IP Core coregen\_fifo\_32x128

♪ ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ●			Design Overview	Г		Pico_Toplevel Project St	atus (	11/04/2	012 - 14:45:3	7)		
Wew: ● ∰ Implementation ● I Simulation       Hierarchy       ● ☐ xx50h240 pasthrough arch       ● ☐ xx50h240t2ff115       ● ☐ Unassigned User Library Modules       ● ☐ Unassigned User Library Modules		IOB Properties	F	Project File:	m501lx240_passthrough_arch.xise	Parser Errors:			No Errors			
				1	Module Name:	Pico_Toplevel	Imple	ementati	on State:	Programmin	g File Generated	
	-		Timing Constraints     Pinout Report	1	Target Device:	xc6vlx240t-2ff1156		•Errors:	5	No Errors		
	ç		- 📑 Clock Report	F	Product Version:	ISE 13.4 •Wa		• Warnin	igs:	8090 Warni	8090 Warnings (8089 new)	
	1000		Frors and Warnings	C	Design Goal: Timing Performance			• Routin	g Results:	All Signals Completely Route		
VicoTestbench_Empty (C:\ImpulseC_Pico\Examples\M5XX\Passthrough_Linux\expor     DiscoToplevel (C:\ImpulseC_Pico\Examples\M5XX\Passthrough_Linux\export_hw\	38		Parser Messages	C	Design Strategy:	SmartXplorer - mapioreg	• Timing Constraints		All Constraints Met			
Volge rPC/opered (CumplesC) recovery and the second of the second o	A		Synthesis Messages	E	Environment:	System Settings		• Final Ti	ming Score:	0 (Timing R	eport)	
	-	Translation Messages     Map Messages     Place and Route Messages	- 📓 Map Messages									
Comparator_mask_static (C:\ImpulseC_Pico\Examples\M5XX\Passthrough_Linux\exp     Comparator_sel_mask (C:\ImpulseC_Pico\Examples\M5XX\Passthrough_Linux\exp			Place and Route Messages           Timing Messages	Device Utilization S				mary				
Comparator_sel_mask (c:\impulsec_Pico\Examples\W5XX\Passtrough_Linux\     V comparator_sel_mask_static (C:\ImpulseC_Pico\Examples\M5XX\Passthrough_Linux\			Bitgen Messages	9	Slice Logic Utilizatio	n		Used	Available	Utilization	Note(s)	
comparator_static (C:\ImpulseC_Pico\Examples\M5XX\Passthrough_Linux\export_hv     mux enc (C:\ImpulseC Pico\Examples\M5XX\Passthrough Linux\export_hv\firmwar		1	All Implementation Messa     Detailed Reports	N	Number of Slice Registers			26,283	301,440	8%		
mux_enc (C:\impulseC_Pico\Examples\W5AX\Passthrough_Linux\export_nw\inmwal     joco_block_axi (C:\ImpulseC_Pico\Examples\M5XX\Passthrough_Linux\export_hw\he		Synthesis Report		Number used as Flip Flops			26,280					
			Translation Report		Number used as Lat	ches		3				
K M			Map Report     Place and Route Report		Number used as Lat			0				
No Processes Running			Post-PAR Static Timing Re		Number used as AND/OR logics Number of Slice LUTs			0				
Processes: Pico Toplevel			Power Report     Bitgen Report	N				16,908	150,720	11%		
E Design Summary/Reports			Number used as logic			10,599	150,720	7%				
🕀 🎽 Design Utilities			esign Properties		Number using O6 output only Number using O5 output only Number using O5 and O6			7,967				
User Constraints     Synthesize - XST			ptional Design Summary Contents					601		<u> </u>		
Contract Address		1 F	Show Clock Report					2,031				
🗈 🔃 🔔 Translate		t	Show Failing Constraints		Number used as F	used as ROM		0				
⊕ C2 Map ⊕ C2 Place & Route		L.,	Show Errors		Number used as Memory Number used as Dual Port RAM Number using O6 output only			2,533	58,400	4%		
- 🕐 🥼 Generate Programming File								1,156				
Configure Target Device								320				
Analyze Design Using ChipScope					Number using C	D5 output only		9				
				II.	Number using C	05 and O6		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.

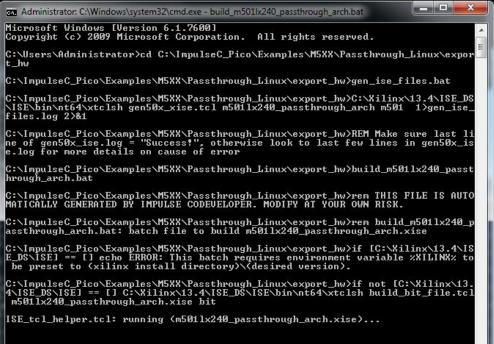


Figure 25 - Build bitfile in ISE 13.4

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\P	vico_Toplevel_summary.html	▼ 49	× Funmoods	\$			
Favorites 🛛 👍 💽 Sugg	gested Sites 👻 👩 Web Slice Gallery 👻 🔏 Building Ap	plications that 🔏 How to C	onfigure Visual 🔏 Ho	ow to Enable a 64-Bit Vis				
Xilinx Design Summary				• 🔊 • 🖃 🖶 • Pag	ge 🕶 Safety 🕶 Tools 🕶 🌘			
To help protect your securi	ity, Internet Explorer has restricted this webpage from run	ning scripts or ActiveX control	s that could access your co	mputer. Click here for optior	15			
	21 	o Toplevel Project Statu						
Project File:	m5011x240_passthrough_arch.xise	Parser Errors:	20101	No Errors				
Module Name:	Pico_Toplevel	Implementation	State:	Programming File Ger	ierated			
Target Device:	xc6vlx240t-2ff1156	• Errors:		No Errors				
Product Version:	ISE 13.4	Warnings		8090 Warnings (8089	new)			
Design Goal:	Timing Performance	Routing R	lesults:	All Signals Completely Routed All Constraints Met 0 (Timing Report)				
Design Strategy:	SmartXplorer - mapioreg	• Timing Co	onstraints:					
Environment:	System Settings	• Final Timi	ng Score:					
	Device Uti	ilization Summary			[-]			
Slice Logic Utilization		Used	Available	Utilization	Note(s)			
Number of Slice Register		26,283	301,440	8%				
Number used as Flip I		26,280						
Number used as Latel		3						
Number used as Latch		0						
Number used as ANI	D/OR logics	0						
Number of Slice LUTs		16,908	150,720	11%				
Number used as logic	<i>v</i>	10,599	150,720	7%				
Number using O6 o		7,967						
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 Du		1,156						
Number using O		320						
Number using O		9						
Number using O		827						
Number used as Si		0						
Number used as Sh	50	1,377						
Number using O	6 output only	1,237						

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).

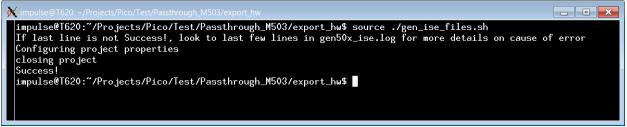


Figure 27 - Generate ISE project files

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

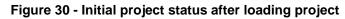
- 1. If you already haven't done so, launch or verify that the license manager is operational and that your license is valid. Contact Xilinx for more information on licensing related to a Linux installation.
- 2. Open a terminal window and set environment variables by executing the "settings64.sh" script 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 Pro	oject 🛌						
Look in:	home/administratorssthrough/export_hw	G	0	0	a		D
Computer administ	firmware						
File <u>n</u> ame:	m501lx240_passthrough_arch.xise					<u>O</u> pen	
Files of type:	ISE Project Files (*.xise)			\$		<u>C</u> ancel	

a. /opt/Xilinx/13.4/ISE\_DS/ISE/bin/lin64/ise &

Figure 29 - Select the ISE project

)es	sign Giù@@	v.	E Design Overview	1		Pico 1	loplevel Projec	t Status				
IJ.	View 🖲 🋱 Implemental 🔿 🞆 Simulat	1 00	Summary	Project File:	m501lx240_passthrough_arch.xise Pico_Toplevel xc6vix240t-2ff1156 ISF 13.4			Parser Errors: Implementation State:			No Errors New	
C)	Hierarchy	0	D Monte Lavel Utilization	Module Name:								
ē.	Orassigned User Library Mc	0	- D Final Report	Target Device: Product Version: Design Goal: Design Strategy: Environment:				+ Erro	ors:		-0.1	
		0	- D elock Report					• Warnings:				
FL.	Automatic 'includes     PicoTestbench Empty (Pico'	100	E Errors and Warnings		Timing Pe	erformance	9	+ Rou	ting Results:			
Ċ.	Pico_Toplevel (Pico_Tople	-	Partser Messages     Syntheers Messages     Translation Messages     Non Messages		Xilinx Default (unlocked)			+ Tim	ing Constraint	s:		
2	carry (carry.v)	錮						• Final Timing Score:				
	🗉 💽 comparator_mask (compare	-06						The second s				
ने	comparator_mask_static (c) comparator_sel_mask (com	60	Place and Routs Messa     Timing Messages	T:		0.00	led Reports					
	🗉 🔽 comparator_sel_mask_stati		Bitgen Messages	Report Name		Status	Generated	Errors	Warnings	Infos	1.4	
	E comparator_static (comparator_static (comparator_static ))		All Implementation Mes     Detailed Reports	Synthesis Report		status	Generated	Errors	warnings	intos		
	🖻 🔽 pico_block_axi (pico_block_		Synthesiz Poport	Translation Report						-		
		8	Translation Report	Map Report					-	-		
	С. II Б.		Place and Boste Report	Place and Boute Rep	nt.	-		-		-		
-	No Processes Running Processes: Pico_Toplevel     Design Summary/Reports		Foot PAB Stock Timing     Power Report     Brigen Report	Power Report				-		-		
e:				Post-PAR Static Timir	ng Report							
1			Secondary Reports	Bitgen Report								
94	Design Utilities     User Constraints		Design Properties	Lastration						-		
間	🗷 🔁 Synthesize - XST	c	Optional Design Summary Contents Show Clock Report Show Failing Constraints Show Warnings	The second se								
	E () Implement Design E () Translate				Secondary Reports		Generated					
	E-C2 Map			Report Name		Statu	•	Gei	nerated			
	Place 6 Route     Generate Programming File     Gongenerate Programming File     Gongene Target Device     Analyze Design Using Ch	Show Errors		Date Generated: 11/07/2012 - 21:29:56								
	And the second se		\$									
	Start Oesign Files	g	Design Summ	ary	×							
		foun n hi										



(?) The required file									
<ul> <li>/home/administrator/Pico/Passthrough/export_hw/hdl/impulse/fifo128x512_synth.v for the IP core fifo128x512.xco was not found in the IP core source directory. In order for synthesis to run successfully, you will need to either copy the file(s) to the IP core source directory or regenerate the core. Would you like to create these files by regenerating the IP Core?</li> <li>Click "Yes" to regenerate this IP core to create the necessary file(s). Click "No" to skip regeneration of this IP core only.</li> </ul>									
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.									

Figure 31 - Regenerate fifo128x512



Figure 32 - Regenerate coregen\_fifo\_32x128

sign	HUCK	a	E Design Overview		Pico Toplevel Project Status	(11/07/20	12 - 21:57:26)	8	
viev	🔹 🛱 Implemental 🔿 🚟 Simulat	Summary     IOB Properties		Project File:	Parser E	and the second second second	No Errors		
	archy m501lx240_passthrough_arch	00	Module Lavel Utilization Timing Constraints	Module Name:	Pico_Toplevel		ntation State:	Progra	amming File rated
	xc6vix240t 2ff1156 In Unassigned User Library Mc		Pinout Report Clock Report	Target Device:	arget Device: xc6vix240t-2ff1156		ors:	No Errors	
1	Automatic includes	0 日 組	Static Timing	Product Version:	ISE 13.4	• Wa	mings:	8090 Warnings (8089 new)	
	<ul> <li>carry (carry.v)</li> <li>clk ibuf (clk ibuf.v)</li> </ul>		Parser Messages     Synthesis Messages     Translation Messages	Design Goal:	Design Goal: Timing Performance		rting Results:	All Signals Completely Routed All Constraints Mot 0 (Timing Report)	
		06	<ul> <li>Map Messages</li> <li>Place and Route Messa</li> </ul>	Design Strategy:	Xilinx Default (unlocked)	• Timing Constraints: • Final Timing Score:			
	<ul> <li>comparator_sel_mask (com</li> <li>comparator_sel_mask_stati</li> <li>comparator_static (comparator_static (comparator_static</li></ul>		Timing Messages     Bitgen Messages     All Implementation Mes	Environment:	System Settings				
1	pico_block_axi (pico_block_		Detailed Reports     Synthesis Report     Translation Report				I		
		Map Report	Slice Logic Utilization			Used Available Utili		ization Note(s)	
53	No Processes Running		Post PAR Static Timing	Number of Slice Re	Number of Slice Registers		301,440	8%	
Pres	esses: Pico Toplevel		Power Report     Bitgen Report	Number used as	Flip Flops	26,280			
11-1-2	Design Summary/Reports     Design Utilities     Design Utilities     Design Utilities     Design Utilities     Design Utilities     Design Utilities     Design     Design Utilities     Design		E Secondary Reports	Number used as	Latches	3			
		Design Properties	Number used as Latch-thrus		0				
			Optional Design Summary Contents	Number used as	Number used as AND/OR logics				
			Show Clock Report			16,908	150,720	11%	
			Show Failing Constraints Show Warnings Show Errors	Number used as logic Number using O6 output only			150,720	736	
11									
				Number using O5 output only Number using O5 and O6		601			
						2.031			
			Number used as ROM Number used as Memory						
						58.400	4%		
onsole di	t Design S Files I I	25 104	Design Summary (Programm d. Routing from the different bu		×				ec

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

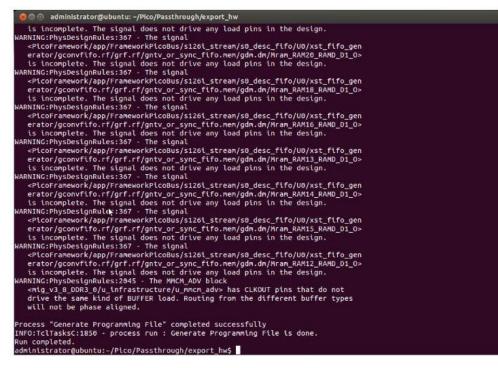


Figure 34 - Ubuntu Xilinx ISE 13.4 Command Line output

🧧 🗇 Pico_Toplevel.par (~/Pic 📄 🚞 Open 🔸 🌌 Save	o/Passthrough/expo		i Ma	2		Þ		
		× 100 °B						
go2psp.sh % Pico_Topleve MMCM_PHASE_CALIBRATI   ON_ML_LUT2_161_ML_NE   W_CLK	Local	2	0.000	0.746		Q	timing score	
MMCM_PHASE_CALIBRATI   ON_ML_LUT2_153_ML_NE   W_CLK	Local	2	8.000	6.323				
MMCM_PHASE_CALIBRATI   ON_ML_LUT2_169_ML_NE   W_CLK	Local		0.000	0.293				
Net Skew is the difference nly delays for the net. No s reported in TRCE timing he minimum and maximum pat The fanout is the number or example SLICE loads not	ote this is diff report. Clock S th delays which of component pi	erent from kew is the includes lo	Clock Ske differend gic delay	w which e between ys.	s,			
<pre>ining Score: 0 (Setup: 0, NFO:Timing:3386 - Intersec Information, see the TSJ Tools User Guide for ini umber of Timing Constraint sterisk (*) preceding a cc</pre>	ting Constraint I report, Pleas formation on gen ts that were not	s found and e consult t erating a T applied: 3	resolved he Xilinx SI report	. For mor Command L				
This may be due to a set								
Constraint		Chec	k   Wo	st Case   Slack	Best Case   Achievable		Timing Score	
TS_CLK_250 = PERIOD TIMEC SYSCLK HIGH 50% PRIORITY		_   SETUP   HOLD   MINPERI	     ao	0.005ns  0.000ns  0.000ns	3.990ns    4.000ns	0  0  0	0 0 0	
TS_CLK_500 = PERIOD TIME( SYSCLK * 2 HIGH 50% PRIOF		_   SETUP   HOLD   MINPERI		0.005ns  0.000ns  0.000ns		0  0  0	0 0 0	

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.

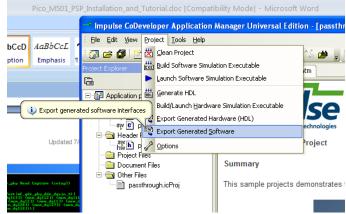


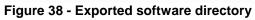
Figure 36 - Export Generated Software

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	nj - [Keadme.htm]	
<u> </u>	Help		
i 🛺 🧀 🕼 🎦 💥 Clean Proje		💡 磁 曖 曖 因 🖌 🗇 💂	
	are Simulation Executable		4 Þ 🖛 🗙
	tware Simulation Executable		N P • A
- Application pat 🛗 Generate H	IDL		(HLY)) // Open stream for read
	h Hardware Simulation Executable		og. "Reading ide: Ad\n".x1;
Export Gen	erated Hardware (HDL)		<pre>in, sks[1][k]: 1= co err ecs] { terlog, "%1u ", ks[1][k][]</pre>
sw C pas C Export Gen	erated Software		
SW h pas / Options			
Project Files			
Document Files	Summary		
Cther Files	This sample projects demonstrates	the basics of streams-based communications.	
	This sample projects demonstrates	the basics of allearns-based communications.	
			E
		SW HW SW	
		(sim) (sim)	
		Character Pass-through Test	
		generator filter consumer	
			*
	Build		$0.\times$
	====== Building target 'export_software' in for i in passthrough_sw.c; do cp \$i sw; done	e_Makefie =======	~
	for i in passthrough h; do cp \$i sw; done		
	chmod -R +nv sw "C://mpu/se/CoDeveloper3/bin/mpu/se_expo	t" =software =srcdirsw "-=aC//Impulse/CoDeveloper3/Architectures/pico_m501_linux_vhdl.xml" passthrough.xic "export_sw"	
	Impulse C Design Exporter		
	Copyright 2002-2007, Impulse Accelerated Te All rights reserved.		
	Loading C:/Impulse/CoDeveloper3/Architectu Loading C:/Impulse/CoDeveloper3/Architectu	es/pico_m501_linuxvhdlxml	
	Loading passthrough xic		
	====== Build of target 'export_software' cor	plete =======	
			*
	🔠 Build 🕞 Find in Files 🖸 System		
Export generated software interface	2S		OVR NUM .::

Figure 37 - Build window output

Organize 🔻 🛛 Include in lik	orary 🕶 Share with 🖛 Burn New fol	lder		· · · · · · · · · · · · · · · · · · ·
🔶 🔶 Favorites	^ Name	Date modified	Туре	Size
	ib 🔒	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
E 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.

rganize	and the second sec	der			853 •	
2	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			
	Ø hw	11/4/2012 8:19 AM	File folder			
	illes 🕺 ReadmeFiles	11/4/2012 8:04 AM	File folder			
	ø sw	11/4/2012 8:19 AM	File folder			
E	🚳 _Make_passthrough.bat	11/4/2012 8:20 AM	Windows Batch File	1 KB		
E	O _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		
	passthrough.h	11/4/2012 8:04 AM	C/C++ Header	1 KB		
	👩 passthrough.i	11/4/2012 8:19 AM	Preprocessed C/C	16 KB		
0	🕢 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		
	passthrough.smd	11/4/2012 8:19 AM	SMD File	5 KB		
	👩 passthrough.snt	11/4/2012 8:19 AM	SNT File	27 KB		
	👩 passthrough.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"

😣 🖨 🗉 administrator@ubuntu: ~/Pico/export_sw/software
administrator@ubuntu:~/Pico/export_sw/software\$ make 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.

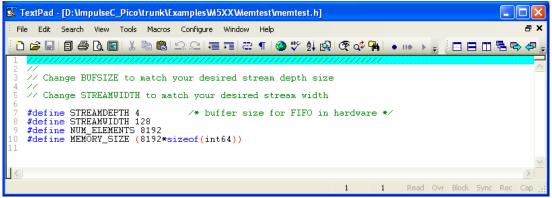


Figure 41 - memtest.h

#### 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 37 while (co\_stream\_read(idata, &data128, sizeof(data128)) == co\_err\_none) 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 | 사 🖻 🕄 으 으 ( = 〒 ) 곧 ¶ | 🥝 🎌 👭 🚱 ( 🏵 🐢 🗛 ) • 🗤 🕨 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.

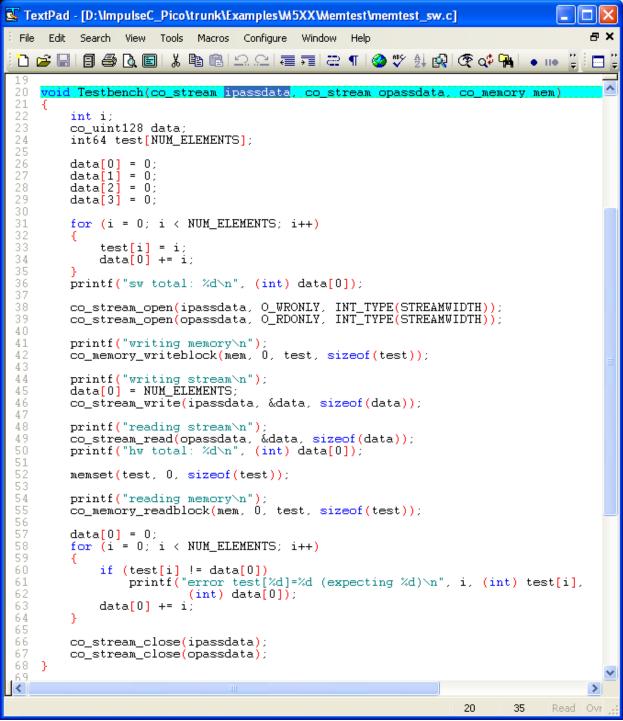


Figure 44 - User defined stimulus in memtest\_sw.c

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.

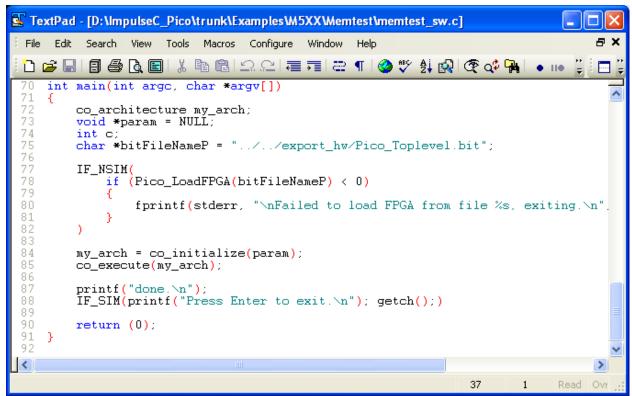


Figure 45 - Main program in memtest\_sw.c

## 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:\ImpulseC_Pico\Examples\M5XX\Memtest\memtest.h - Notepad++	- <b>-</b> X
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?	Х
▶ = = = = > > + = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = > < = = = =	
memtest.h	
1 /////////////////////////////////////	
2 //	
3 // Change BUFSIZE to match your desired stream depth size	
4 //	
5 // Change STREAMWIDTH to match your desired stream width	
6	
7 #define STREAMDEPTH 4 /* buffer size for FIFO in hardware */	
8 #define STREAMWIDTH 128	
9 #define NUM_ELEMENTS 8192	
10 #define MEMORY_SIZE (8192*sizeof(int64))	
11	I
C++ source file length : 350 lines : 11 Ln : 1 Col : 1 Sel : 0 UNIX ANSI	INS
C++ source file         length : 350         lines : 11         Ln : 1         Col : 1         Sel : 0         UNIX         ANSI	INS

Figure 46 - Impulse C Header File

C:\Im	npulseC_Pico\Examples\M5XX\Memtest\memtest_hw.c - Notepad++	×
File Ed	dit Search View Encoding Language Settings Macro Run Plugins Window ?	Х
	) 🗄 🖻 💫 🖓 👘 👘 🌶 d 🖷 🦕 🤏 🖓 🔂 🖬 🕇 🗐 🖾 🖬 🗐 🖉 🖉	
-		
_	test.h 📄 memtest_hw.c	
1		<u>^</u>
2	// memtest_hw.c: includes the hardware process and configuration	
3	// function.	
4		
6	<pre>// See additional comments in memtest.h. //</pre>	
7		
8	<pre>#include <stdio.h></stdio.h></pre>	
9	<pre>#include "co.h"</pre>	
10	#include "cosim log.h"	
11	finclude "memtest.h"	
12		
13	extern void Testbench(co stream ipassdata, co stream opassdata, co memory mem);	
14		
15		
16	// This is the hardware process.	-
17	11	
18	void pass(co_stream idata, co_stream odata, co_memory mem)	
	〒 (	
20	int i, n;	
21	co_uint128 data128;	
22	int64 data, inc;	
23	int64 work[NUM_ELEMENTS];	
24 25	// EdT: co banks not available until CoDeveloper v3.70.d.13 or later	
26	// Configure array to be two banks of 64 bits to maximize throughput from 128 bit memory.	
20	<pre>// configure array co be two banks of 64 bits to maximize throughput from 128 bit memory. // co array config (work, co banks, "2");</pre>	
28	// Co_array_conrig(work, co_banko, 2 /,	
29	do	
31	// Hardware processes run forever	
32	co stream open(idata, O RDONLY, INT TYPE(STREAMWIDTH));	
33	co_stream_open(odata, O_WRONLY, INT_TYPE(STREAMWIDTH));	
34		
35	// Read values from the stream	
36	<pre>while (co_stream_read(idata, &amp;data128, sizeof(data128)) == co_err_none)</pre>	
37		
38	<pre>n = (int) data128 IF_SIM([0]);</pre>	
39	<pre>co_memory_readblock(mem, 0, work, n * sizeof(int64));</pre>	
40	data = 0;	
41	for $(i = 0; i < n; i++)$	
42		
43	<pre>inc = work[i]; unpb[i] = data :</pre>	
44 45	<pre>work[i] = data; data t= inc;</pre>	
45	data += inc;	
47	<pre>co memory writeblock(mem, 0, work, n * sizeof(int64));</pre>	
48	<pre>data128 IF_SIM([0]) = data;</pre>	
49	<pre>co_stream_write(odata, &amp;data128, sizeof(data128));</pre>	
50		
51	, co stream_close(idata);	
52	Gostream close (odata):	•
C source	file length:2317 lines:90 Ln:1 Col:1 Sel:0 UNIX ANSI	INS

Figure 47 - ImpulseC Hardware File

# 9.4. Opening Project

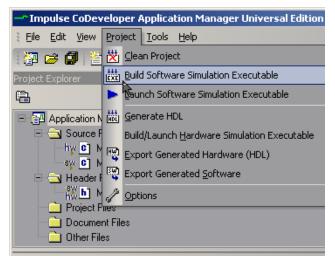
Open the CoDeveloper project file 'Memtest.icProj' by selecting and pressing 'Enter' or by double-clicking it:

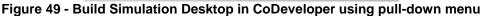
.ook in: 🔒	Memtest	-	+ 🗈 💣 📰 +		
Name	^		Date modified	Туре	Si
📝 memtes	t.icProj		11/1/2012 8:58 AM	Impulse C Project	-
1		m			,
ile name:	memtest.icProj			Open	

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:

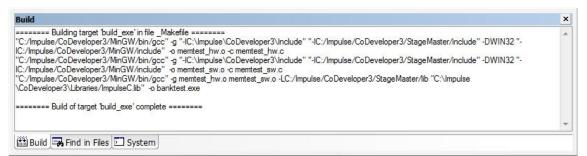


Figure 51 - Output within the CoDeveloper IDE build window

## 9.6. Running Desktop Simulation Executable

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

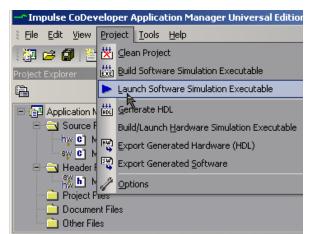


Figure 52 - Launch software simulation window using pull-down menu

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)
  - o Include floating point library (must be unchecked)

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

uild Simulate Generate System Registration	
Platform Support Package: Pico M-501 Linux (VHDL)	
Hardware Optimization and Generation	
Enable constant propagation	Generate dual clocks
Scalarize array variables	Generate active-low reset
Relocate loop invariant expressions	Use std_logic types for VHDL interfaces
	I ■ Do not include co_ports in bus interface
Additional optimizer options:	Additional library options:
Floating Point Options	- Output Directories
Include floating point library	Hardware build directory:
☐ Include co math library	hw
Enable floating point optimization	Software build directory:
Allow double-precision types and operators	sw
└ Use higher latency, faster clock operators	Hardware export directory:
Enable floating point accumulators	export_hw
Use extended precision accumulators	Software export directory:
Lise evienced precision accumulators	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:

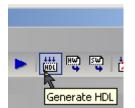


Figure 57 - Generate HDL using toolbar icon

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:

×
02-2009, Impulse Accelerated Technologies, Inc.
erved.
e0/pass
generation complete
CoDeveloper3/bin/impulse arch" "-aC:/Impulse/CoDeveloper3/Architectures/pico m501 linux vhdl.xml"-no port bus connect -swdirsw -
st_comp.vhd memtest_top.vhd "-srcs "memtest_hw.c memtest.h" memtest xic hw/memtest_top.vhd
DL Design Generator
02-2012, Impulse Accelerated Technologies, Inc.
erved.
mpulse/CoDeveloper3/Architectures/pico_m501_linux_vhdl.xml
mpulse/CoDeveloper3/Architectures/Pico/M50x Linux/bus50x.xml
mpulse/CoDeveloper3/Architectures/Pico/M50x Linux/axi ddr.xml
mpulse/CoDeveloper3/Architectures/VHDL/target.xml
mpulse/CoDeveloper3/Architectures/VHDL/Xiinx/v4tech.xml
mpulse/CoDeveloper3/Architectures/VHDL/Generic/Generic/system.xml
ntest xic
_m501_linux_vhdl.xml
2i=memory mc0 axiddr {} mc0 {}
2i=stream in p testbench ipassdata 128 picobus sp {} idata {}
2i=stream out p testbench opassdata 128 picobus sp () odata {}
2i=stream in config 32 picobus sp {} config {}
a (memory mc0 axiddr {) mc0 {} 0} (stream in p_testbench_ipassdata 128 picobus sp {} idata {} 1} (stream out p_testbench_opassdata 128
<ul> <li>memory micro andor y micro y or scream in p_escoence _possocia izo piccobus sp y idata y in scream out p_escoence_opassocia izo</li> <li>odata () 1) (stream in config 32 piccobus sp () config () 2)</li> </ul>
y mol avid t mol 0 0,0
in Diestench ipassdata 128 picobus sp () idata () 1, 1
i out p_testbench_opassdata 128 picobus sp () odata () 1, 1
i in config 32 picobus sp {} config {} 2, 2
ration complete
w hw
/CoDeveloper3/bin/impulse_lib" "-aC:/Impulse/CoDeveloper3/Architectures/pico_m501_linux_vhdl.xml" -hwdirhw files "memtest_sw.c" sw/co init.c
offware Interface Generator
Javare interace derivation 02-2012, Impulse Accelerated Technologies, Inc.
arved.
mpulse/CoDeveloper3/Architectures/pico_m501_linux_vhdl xml
mpulse/CoDeveloper3/Architectures/Pico/M50x_Linux/cpu50x.xml
mpulse/CoDeveloper3/Architectures/VHDL/Generic/Generic/system.xml
ntest xic
est_sw.c; do cp \$i sw; done
est h: do cp Si sw: done
w sw
uild of target build' complete ======
Find in Files 🔄 System



## 9.9. Exporting Hardware

Export hardware via "Project" menu:

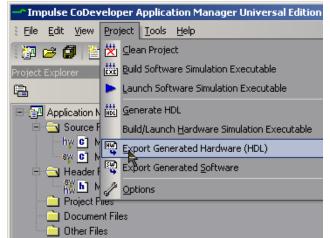


Figure 59 - Export Generated Hardware (HDL) using pull-down menu

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:

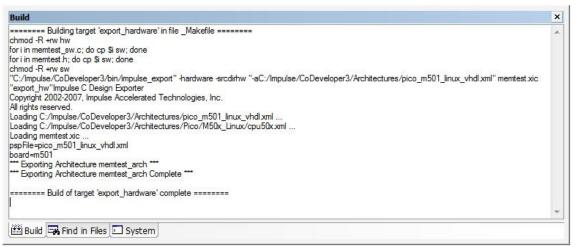


Figure 61 - Build window output

# 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.

rganize 🔻	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_m5011x240_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
0	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.

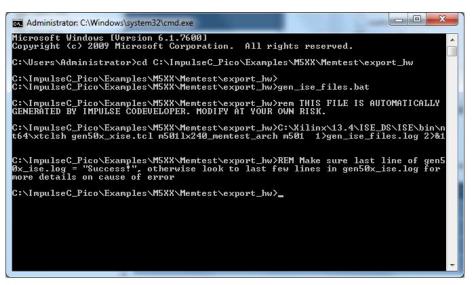


Figure 63 - Generate ISE project files

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:
xfile_add:
xfile_add: add glob file=firmware/m501/coregen_L%240T/v6_pcie_v2_4_8lane_gen2/ source/pcie_reset_delay_v6.v
xfile_add: add glob file=firmware/m501/coregen=L%240T/v6_pcie_v2_4_8lane_gen2/ source/pcie_trn_128.v
<pre>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/</pre>
source/sync_fifo.v xfile_add: add glob file=firmware/m501/coregen-L%240T/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-LX240T/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_PCle.ucf
xfile_add:End Configuring project properties
closing project Success!
C:\ImpulseC_Pico\Examples\M5XX\Memtest\export_hw>

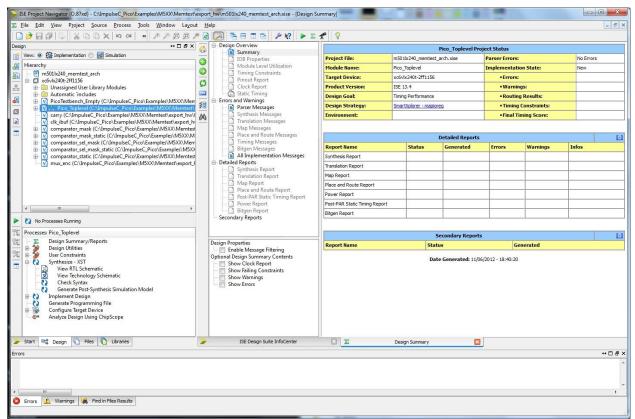
Figure 64 - Expected Gen\_Ise\_File log report

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

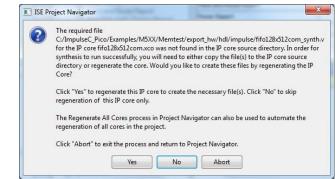
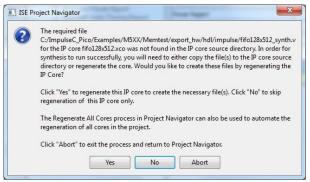


Figure 66 - Xilinx ISE 13.4 regenerate IP core fifo128x512com



#### Figure 67 - Xilinx ISE 13.4 regenerate IP Core fifo128x512

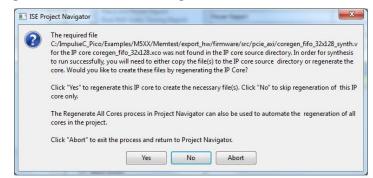


Figure 68 - Xilinx ISE 13.4 regenerate IP Core coregen\_fifo\_32x128

ming Messages is Messages essages essages Messages Messages versages mentation Messages s s s s s port port	Project File: Hodule Hame: Target Device: Product Version: Design Goal: Design Strategy: Environment: Silice Logic Utilization Nuriber of Sice Registers "Subper und an Pio Filos	n50 bi240 mentest archivise Pio_Topkinet uctricit.240 2011 156 158 13.4 Thining Performance Star Datorer - machines Sustem Sattinas	Parser En Implemen • En • W/ • Re • Th • Fir	/05/2012 - 20:20:05 rors: intation State: rors: amings: buting Results: ming Constraints: nal Timing Score:	No 1 Pro No 1 812 AL S AL S		2				
Level Unlikation Constraints Peport peport Ining Messages Sis Messages Sissages di Roste Messages Messages enementation Messages 5 5 5 is Report port	Hodule Rame: Target Device Product Version: Design Goal: Design Strategy: Environment: Skee Logic Utilization Tumber of Ske Registers	Pea_Toplanel Pea_Toplanel Pea_Toplanel Pea_Toplanel Pea_Toplanel Pea_Toplanel Peartmance	Implemen • En • Wi • Re • Th • Fir	ntation State: rors: amings: outing Results: ming Constraints:	Pro No I 812 ALS ALS	gramming File Generate irrors 5 Warnings (8174 new ignals Completely Rout ionstraints Met	2				
Constraints Peport export ming dessages in Messages in Messages sisage d fourte Messages d fourte Messages exemption Messages s is Report port	Target Device: Product Version: Design Strategy: Environment: Silce Logic Utilization Number of Sice Registers	volsko 2401-211136 252 13-4 Timing performance Startbolver - masorea System Sattinas Device Utilizo	• En • Wi • Re • The • Fir	rors: amings: outing Results: ming Constraints:	190 1 812 ALS ALS	irrors 5 Warnings (8174 new) Ignals Completely Roy Ionstraints Met	2				
Report export ings dessages is Messages ssages of Roste Messages Messages versages exementation Messages s is Report port	Product Version: Design Goal: Design Strategy: Environment: Slice Logic Utilization Number of Slice Registers	25: 13.4 Timing Performance <u>SmartOstoren - mackoren</u> Sustem Settings Device Utilizo	• Wi • Re • Th • Fir	amings: outing Results: ming Constraints:	812 ALS ALS	5 Warnings (8174 new Ignals Completely Rout Ionstraints Met					
ming Messages is Messages essages essages Messages Messages versages mentation Messages s s s s s port port	Design Goal: Design Strategy: Environment: Slice Logic Utilization Number of Slice Registers	Timing Performance Smart/Splorer - macioneg System Settings Device Utilizz	• Ro • Tir • Fir	outing Results: ming Constraints:	ALS ALC	ionals Completely Rout Ionstraints Met					
ings Acessages is Messages sasges sasges di Route Messages Messages rementation Messages s is Report ion Report port	Design Strategy: Environment: Slice Logic Utilization Number of Slice Registers	Smart/Spiorer - mapioreg System Settings Device Utiliza	• Tir • Fir	ming Constraints:	ALC	ionstraints Met	ted				
Aessages is Messages essages di Route Messages Messages Vessages ementation Messages s is Report ion Report port	Environment: Slice Logic Utilization Number of Sice Registers	System Settings Device Utiliza	• Fir								
ion Messages essages di Route Messages Messages ementation Messages is Report ion Report port	Slice Logic Utilization Number of Sice Registers	Device Utiliza		nal Timing Score:	0 0		All Constraints Met				
essages of Route Messages Messages vessages ementation Messages is Report ion Report port	Number of Slice Registers		ation Summary			Tming Report)					
nd Route Messages Messages Vessages ementation Messages is Report ion Report port	Number of Slice Registers		ation Summary								
Vessages ementation Messages s is Report ion Report port	Number of Slice Registers			Device Utilization Summary							
ementation Messages s is Report ion Report port	Number of Slice Registers		ed	Available	Utilization	Note(s)					
s is Report ion Report port			27,455 301,44		-	9%					
ion Report port			27,448	0012110							
port	Number used as Latches		7								
	Number used as Latch-thrus		0								
nd Route Report	Number used as AND/OR logics		0		-						
R Static Timing Report	Number of Slice LUTs		17,673	150,720	1	11%					
Report	Number used as logic		10.983	150,720		7%					
orts	Number using Q6 output only		8,206	1001/100							
c Report c Log File	Number using O5 output only		654		-	-					
					-						
	-		0		-	_					
			2.491	58.400	1	4%					
sage Filtering		· · · · · · · · · · · · · · · · · · ·									
Report											
		AM									
ngs						_					
	Number using O6 output only										
	Number using OS output only										
	Number using 05 and 06		140								
	Number used exclusively as route-thrus		4,199								
	Number with same-slice register load		4,152		-						
	Number with same-sice carry load		47								
	Number with other load		0								
Design Summary (Pro											
							_				
	age Filtering mrary Centents Prepert Constants gg	sge Filtering many Contrasts ges Contrasts g	Number und o 50 erd 05           Mumber und a ROH           Number und ROH NUM           Number und SOH 05           Number und SOH 050           Number Und S	Number using 0.5 mid 05         2.223           Mumber using 8.024         0           Number using 8.024         0           Number using 8.024         2.491           Number using 8.024         1.114           Number using 8.024         1.114           Number using 0.040x0 mty         2.292           Number using 0.040x0 mty         2.921           Number using 0.040x0 mty         0.92           Number using 0.040x0 mty         0.93           Number using 0.040x0 mty         0.93           Number using 0.040x0 mty         0.93           Number using 0.040x0 mty         1.337           Number using 0.040x0 mty         0.03           Number using 0.040x0 mty         1.337           Number using 0.040x0 mty         0.93           Number using 0.040x0 mty         0.90           Number using 0.040x0 mty         0.90 <td>Number uang 05 ind 09         2,23           Aunter uang an KOM         0           Number uang an KOM         0           Number uang an KOM         0,471           Stander uang an KOM         0,471           Number uang Coupturery         2,471           Number uang Coupturery         32,2           Number uang Coupturery         32,2           Number uang Coupturery         32           Number uang Coupturery         1,377           Number uang Coupturery         1,237           Number uang Coupturery         1,237           Number uang Coupturery         1,209           Number uang Coupturery         4,109           Number uang Coupturery         4,109           Number uang Coupturery         407           Number unit on tert load         407</td> <td>Number using 0.5 ind 0.6         2,122            Number usid as Rheiny         2,011         58,400           Number usid as Das/Ind 10,401         1,114            Application         1,222            Instruct usid as Das/Ind 11,401         1,114            Number usid as Das/Ind 11,401         1,114            Tuber usid as Das/Ind 11,401         1,114            Number usid as Das/Ind 11,401         1,217            Number usid as Grigh Data 0,61         7,80            Number usid as Grigh Data 0,61         1,207            Number usid as Grigh Data 0,61         1,401            Number usid as Grigh Data 0,61         1,401            Number usid as Grigh Data 0,61         1,401            Number usid nume Gright Mater Mail         4,132            Number usid nume Gright Mater Mail         4,132            Number usid nume Gright Mater Mail         4,132</td> <td>Handbr und p 05 and 06         2,133         Image         Image           Auber und a fX0M         0         0         0           Ruber und a fX0M         0,411         58,400         4%           Auber und a fX0M         1,114         0         0           Auber und a fX0M         1,114         0         0           Auber und a fX0M         1,114         0         0           Stander und a fX0M         1,114         0         0           Nubre ung 05 dupt only         32         0         0           Stander und a Staff R18M         0         0         0           Nubre ung 05 dupt only         1,217         0         0           Nubre ung 05</td>	Number uang 05 ind 09         2,23           Aunter uang an KOM         0           Number uang an KOM         0           Number uang an KOM         0,471           Stander uang an KOM         0,471           Number uang Coupturery         2,471           Number uang Coupturery         32,2           Number uang Coupturery         32,2           Number uang Coupturery         32           Number uang Coupturery         1,377           Number uang Coupturery         1,237           Number uang Coupturery         1,237           Number uang Coupturery         1,209           Number uang Coupturery         4,109           Number uang Coupturery         4,109           Number uang Coupturery         407           Number unit on tert load         407	Number using 0.5 ind 0.6         2,122            Number usid as Rheiny         2,011         58,400           Number usid as Das/Ind 10,401         1,114            Application         1,222            Instruct usid as Das/Ind 11,401         1,114            Number usid as Das/Ind 11,401         1,114            Tuber usid as Das/Ind 11,401         1,114            Number usid as Das/Ind 11,401         1,217            Number usid as Grigh Data 0,61         7,80            Number usid as Grigh Data 0,61         1,207            Number usid as Grigh Data 0,61         1,401            Number usid as Grigh Data 0,61         1,401            Number usid as Grigh Data 0,61         1,401            Number usid nume Gright Mater Mail         4,132            Number usid nume Gright Mater Mail         4,132            Number usid nume Gright Mater Mail         4,132	Handbr und p 05 and 06         2,133         Image         Image           Auber und a fX0M         0         0         0           Ruber und a fX0M         0,411         58,400         4%           Auber und a fX0M         1,114         0         0           Auber und a fX0M         1,114         0         0           Auber und a fX0M         1,114         0         0           Stander und a fX0M         1,114         0         0           Nubre ung 05 dupt only         32         0         0           Stander und a Staff R18M         0         0         0           Nubre ung 05 dupt only         1,217         0         0           Nubre ung 05				

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.

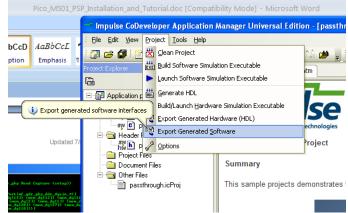


Figure 70 - Export Generated Software

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>======= Building target 'export_software' in file _Makefile ======== for i in memtest_sw.c; do cp \$i sw; done for i in memtest _sw.c; do cp \$i sw; done chmod -R +w sw "C:/Impulse/CoDeveloper3/bin/Impulse_export" -software -srcdirsw "-aC:/Impulse/CoDeveloper3/Architectures/pico_m501_linux_vhdl.xml" memtest xic "export_sw"Impulse C Design Exporter Copyright 2002-2007, Impulse Accelerated Technologies, Inc. All rights reserved. Loading C./Impulse/CoDeveloper3/Architectures/pico_m501_linux_vhdl.xml Loading C./Impulse/CoDeveloper3/Architectures/Pico/M50x_Linux_vhdl.xml Loading memtest xic ======== Build of target 'export_software' complete ===================================</pre>	*
Build Rind in Files System	*

Figure 71 - Build window output

Organize 🔻	Include in library 🔻	Share with $\bullet$	Burn	New folder			• == •	
🔆 Fave	Name		Date	modified	Туре	Size		
	鷆 lib		11/6	/2012 6:42 PM	File folder			
词 Libr	co_init.c		11/6	/2012 6:34 PM	C Source	2	KB	
De	📄 Makefile		11/6	/2012 6:34 PM	File	1	КВ	
🚽 м	🖻 memtest.h		11/6	/2012 6:42 PM	C/C++ Header	1	КВ	
Pi	imemtest_sw.c		11/6	/2012 6:42 PM	C Source	2	КВ	

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

rest.bat mtest.bat lefs re	Date modified 11/6/2012 6:40 PM 11/6/2012 6:42 PM 11/6/2012 6:34 PM 11/6/2012 6:33 PM 11/6/2012 6:33 PM 11/6/2012 6:42 PM 11/6/2012 6:31 PM 11/6/2012 6:32 PM 11/6/2012 6:32 PM 11/6/2012 6:32 PM	Type File folder File folder File folder Windows Batch File Windows Batch File File DEFS File Application	Size 1 KB 1 KB 1 KB 1 KB 1 KB	
mtest.bat lefs .e	11/6/2012 6:42 PM 11/6/2012 6:34 PM 11/6/2012 6:34 PM 11/6/2012 6:33 PM 11/6/2012 6:42 PM 11/6/2012 6:42 PM 11/6/2012 6:31 PM 11/6/2012 6:32 PM 11/6/2012 6:32 PM	File folder File folder File folder Windows Batch File Windows Batch File File DEFS File Application	1 KB 1 KB 1 KB	
mtest.bat lefs .e	11/6/2012 6:34 PM 11/6/2012 6:34 PM 11/6/2012 6:33 PM 11/6/2012 6:42 PM 11/6/2012 6:42 PM 11/6/2012 6:31 PM 11/6/2012 6:32 PM 11/1/2012 8:58 AM	File folder File folder Windows Batch File Windows Batch File File DEFS File Application	1 KB 1 KB 1 KB	
mtest.bat lefs .e	11/6/2012 6:34 PM 11/6/2012 6:33 PM 11/6/2012 6:42 PM 11/6/2012 6:42 PM 11/6/2012 6:31 PM 11/6/2012 6:32 PM 11/1/2012 8:58 AM	File folder Windows Batch File Windows Batch File File DEFS File Application	1 KB 1 KB 1 KB	
mtest.bat lefs .e	11/6/2012 6:33 PM 11/6/2012 6:42 PM 11/6/2012 6:42 PM 11/6/2012 6:31 PM 11/6/2012 6:32 PM 11/1/2012 8:58 AM	Windows Batch File Windows Batch File File DEFS File Application	1 KB 1 KB 1 KB	
mtest.bat lefs .e	11/6/2012 6:42 PM 11/6/2012 6:42 PM 11/6/2012 6:31 PM 11/6/2012 6:32 PM 11/1/2012 8:58 AM	Windows Batch File File DEFS File Application	1 KB 1 KB 1 KB	
lefs ce	11/6/2012 6:42 PM 11/6/2012 6:31 PM 11/6/2012 6:32 PM 11/1/2012 8:58 AM	File DEFS File Application	1 KB 1 KB	
e	11/6/2012 6:31 PM 11/6/2012 6:32 PM 11/1/2012 8:58 AM	DEFS File Application	1 KB	
e	11/6/2012 6:32 PM 11/1/2012 8:58 AM	Application		
	11/1/2012 8:58 AM		722 100	
			733 KB	
		C/C++ Header	1 KB	
	11/6/2012 6:34 PM	Preprocessed C/C	17 KB	
Proj	11/1/2012 8:58 AM	Impulse C Project	2 KB	
k0	11/6/2012 6:34 PM	PK0 File	29 KB	
k1	11/6/2012 6:34 PM	PK1 File	17 KB	
ky	11/6/2012 6:34 PM	PKY File	17 KB	
c	11/6/2012 6:34 PM	SIC File	15 KB	
md	11/6/2012 6:34 PM	SMD File	11 KB	
nt	11/6/2012 6:34 PM	SNT File	29 KB	
hw	11/6/2012 6:34 PM	XHW File	10 KB	
c	11/6/2012 6:34 PM	XIC File	9 KB	
iw.c	11/1/2012 8:58 AM	C Source	3 KB	
iw.o	11/6/2012 6:32 PM	O File	368 KB	
w.c	11/4/2012 2:22 PM	C Source	2 KB	
w.o	11/6/2012 6:32 PM	O File	362 KB	
IV IV	v.c v.o	۱1/6/2012 6:34 PM           v.c         11/1/2012 8:58 AM           v.o         11/6/2012 6:32 PM           /.c         11/4/2012 2:22 PM	11/6/2012 6:34 PM         XIC File           v.c         11/1/2012 8:58 AM         C Source           v.o         11/6/2012 6:32 PM         O File           r.c         11/4/2012 2:22 PM         C Source	11/6/2012 6:34 PM         XIC File         9 KB           v.c         11/1/2012 8:58 AM         C Source         3 KB           v.o         11/6/2012 6:32 PM         O File         368 KB           r.c         11/4/2012 2:22 PM         C Source         2 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"

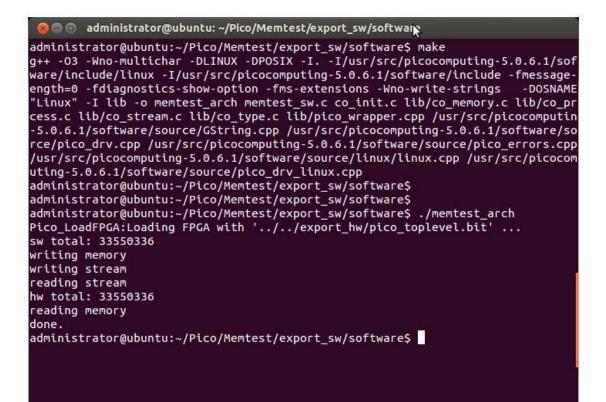


Figure 74 - Exported SW executed on target platforn