Simple Interface for Reconfigurable Computing (SIRC): PC⇔Xilinx V5/V6 Communication

Ken Eguro – 8/11, version 1.1

1. Introduction

The goal of the Simple Interface for Reconfigurable Computing (SIRC) project is to provide a fast and easy to use communication/control mechanism between C++ code running on a host PC and a hardware-based accelerator implemented on a FPGA. Our motivation is that the simple software and hardware interfaces of this API will lower the barrier to entry for reconfigurable hardware accelerators and attract new application developers. This system provides a complete and customizable interface solution that only requires users to have a basic knowledge of C++ and Verilog – only enough to develop their application-specific software and hardware computation. No in-depth knowledge of drivers, operating systems or communication protocols is necessary. Furthermore, the need for debugging is limited to users' computational cores.

This version of the system provides communication via a 1 Gb Ethernet link between a Windows machine and a Xilinx Virtex 5/Virtex 6 FPGA. In our testing, it achieves 50% of the maximum theoretical bandwidth with transfers of 8KB or larger and 95% with transfers of 128KB or larger.

Future releases will add the capability to use different communication methods (such as PCI Express, which is currently in the final stages of debugging), different operating systems and different FPGAs. Any updates will maintain the same software and hardware user interfaces. This will allow applications to easily migrate to new reconfigurable platforms with no changes to user code.

For the latest update, please visit the SIRC discussion forum at:

http://community.research.microsoft.com/forums/153.aspx

2. Setup & Installation

System Requirements

- 1) A network card capable of operating at 1 Gb. This can either be:
 - a. a dedicated card that can be connected directly to the FPGA via a crossover cable. This is the preferred method and will provide the best performance.
 - b. a card that shares a connection with normal PC network traffic and is connected to the FPGA and the upstream network link through a gigabit-capable switch.
- 2) A host PC with that meets the necessary requirements for either:
 - a. Virtual PC 2007 (XP, Vista, or Windows 7 machines)
 - b. Windows Virtual PC (Windows 7 machines)

Note that Virtual PC itself is not actually used. This project only uses the Virtual Machine Network Services driver, but this is not available as a stand-alone download. Also, as discussed later in this document, Virtual PC 2007 is strongly suggested, even on Windows 7 machines

- 3) For direct use with the default settings, a Digilent XUPV5, a Xilinx ML505/506/507 board, a BEE3 (either LX110T, LX155T or SX95T), or an ML605 board.
- 4) Xilinx ISE and a JTAG programming cable to create/compile user hardware-side applications and program the FPGA board (tested with ISE versions 10.1 to 13.2).
- 5) Visual Studio to create/compile user software-side applications (tested with VS2005 to VS2010).
- 6) Modelsim (if the user would like to simulate SIRC and their application in software before moving to hardware). We have tested with version 10.0a.

Host PC Installation Notes

Virtual PC 2007 SP1 is freely available for download from:

http://www.microsoft.com/downloads/details.aspx?familyid=28C97D22-6EB8-4A09-A7F7-F6C7A1F000B5&displaylang=en

Although SIRC supports the Virtual PC network driver included with Windows 7 XP Mode/Virtual PC, this driver is slower than the one included with Virtual PC 2007 (due to the fact the newer version offers less buffering). We encourage users to remove that version, if installed:

- 1) Go to "ControlPanel -> Programs -> Uninstall a Progam" and uninstall "Windows XP Mode"
- 2) After that is complete, in the same window click on "View Installed Updates".
- 3) Under Microsoft Windows, uninstall "Windows Virtual PC (KB95855)

After installation on the host PC, verify that the Virtual Machine Network Services driver was installed properly and enabled only for the network card connected to the FPGA. This can be done by opening "Control Panel->Network Connections" and right-clicking to select "Properties" on each network connection on the host PC. While this window is open, if the FPGA is connected to a dedicated network card, it is also best to unselect all other services for this connection. Lastly, although unlikely if the correct connections (and network switches) are used, it may also be necessary to configure the network card connected to the FPGA to explicitly operate at a speed of 1 Gbps. The method of setting this value differs among card manufacturers and drivers, but it is typically accessed from the "Properties->Configure->Advanced" window of the network connection.

At this point, the user can verify the proper setup and connectivity between the host PC and FPGA if they have a Digilent XUPV5/ML505/ML506/ML507/BEE3/ML605. We have included pre-compiled binaries for both the software and hardware portions of a simple example application in the "precompiledExampleBinaries" directory. These pre-compiled binaries can be used to follow the testing outlined in the "Programming & Execution of Example Program" section.

3. Software Interface

This goal of the following two sections is to provide a high-level view of the software and hardware-side APIs. This will enable users to understand the simple example application provided in "SW_Example" and "HWSrc" directories and build their own applications.

This API is intended to be used for batched execution on the FPGA in a supervisor/worker style mode. The expectation is that the user's program will:

- 1) send one or more pieces of data from the host PC to an input buffer connected to their circuit on the FPGA
- 2) signal the FPGA to start execution on that data
- 3) wait until the computation is done
- 4) retrieve the results from an output buffer also connected to the FPGA-based logic

All operations are initiated by the PC in a single threaded manner. Only one operation is performed at a time e.g. overlapped read and writes operations are not supported. Overlapped I/O can be supported by double buffering with multiple software and hardware APIs. Built-in support for such an interface will be introduced in future releases.

Although the API has the capability to configure the FPGA once both the software API on the PC and the hardware API controller on the FPGA are in communication with each other, it is expected that the hardware controller will be automatically bootstrapped onto the FPGA initially. As described in the "Programming FPGA & Execution of Example" section, this is typically performed either with iMPACT through a JTAG programmer or (preferably) via power-on initialization from flash memory.

As will be described in more detail in the "Hardware Interface" section, the user's circuit on the FPGA will communicate with the host PC through the hardware-side communication API. This hardware API has five primary features: an input memory, an output memory, a 255 x 32-bit parameter register file, a *userRun* signal, and a *userSideReset* signal. The *userRun* signal tells the user's circuit to begin execution. This signal is also used to indicate when the user's circuit has completed execution. The *userSideReset* signal tells the user's circuit that it should reset. These five parts of the hardware-side API are controlled in software via member functions of the *ETH_FPGA* class. The twelve functions within the *ETH_FPGA* class define the software-side API:

FTH FPGA(uint8 t *FPGA ID	Constructor for the API class EPGA ID is an array of 6 bytes that contain the
uint32 t driverVersion	MAC address of the target EPGA <i>driverVersion</i> is the version number of the
wchar t *nicName)	underlying network driver to use (in the current implementation between 1
wenal_t mertainey	and 3) If this value is 0, the system will attempt to connect with the all of the
	drivers from newest to oldest stanning when one is found <i>nicName</i> is a
	specific network adapter to use (e.g. "Microsoft Loophack Adapter"). If this
	value is NULL the system will attempt to connect with the first network
	adapter that has the given network driver version active. That said if this
	adapter that has the given network adapter on the machine chould have the
	Virtual DC driver active. This is because the enumeration of the network
	adapters is not necessarily static and can change for a variety of reasons
	This function only initializes the software side of the system. The EDCA is not
	This function only initializes the software side of the system. The FPGA is not
	All controller should already be beststranged onto the EDCA and ready to
	API controller should already be bootstrapped onto the FPGA and ready to
	run. For more information, regarding setting up the FPGA to automatically
	load the hardware API, see the "Programming FPGA & Execution of Example"
	section.
	If no errors are present after the function returns (check with the getLastError
	function), this means that the system is capable of communicating with the
	FPGA and ready for further commands.
~ ETH_FPGA()	Destructor for the API class. Similar to the constructor, this only tears down
	the software side of the system. Any user or configuration data left on the
	FPGA is not deleted.
int8_t getLastError()	If any of the commands below return false, the cause of the error can be
	checked with this function. A return of exactly zero indicates no error and any
	value less than zero indicates an error (see eth_FPGA.h for a list of errors).

	Although the constructor does not return a boolean value, the user should call this function after the creation of an ETH_FPGA object to make sure that the communication channel initialized successfully.
BOOL sendWrite(uint32_t startAddress, uint32_t length, uint8_t *buffer)	Send a block of data from the PC (pointed to by <i>buffer</i> and of length <i>length</i>) to an input buffer on the FPGA. The write will begin at a particular local byte address in the FPGA's input buffer (<i>startAddress</i>). Returns true if the write succeeds, else return false. Please note that "buffer" is a byte array. Thus, the endian-ness of any multi-byte data types used on the host machine need to be considered when copying values to the buffer. This may change if the user moves from a little endian machine to a big endian machine or vice versa.
BOOL sendRead(uint32_t startAddress, uint32_t length, uint8_t *buffer)	Retrieve a block of data (starting at the local address <i>startAddress</i> and of length <i>length</i>) from the output buffer on the FPGA. Place the data into a buffer on the PC (<i>buffer</i>). Returns true if the retrieval succeeds, else return false. Similar to <i>sendWrite</i> , the endian-ness of multi-byte data types on the host machine need to be considered when copying values out of the buffer.
BOOL sendParamRegisterWrite(uint8_t regNumber, uint32_t value)	Write a 32-bit unsigned integer (<i>value</i>) into one of the registers (address # <i>regNumber</i> between 0 and 254) within the parameter register file of the FPGA. Returns true if the write succeeds, else return false. Since the datatype of <i>value</i> is known, the system compensates for any change in endian-ness on the host machine.
BOOL sendParamRegisterRead(uint8_t regNumber, uint32_t *value)	Read the value of one of the registers (address # <i>regNumber</i> between 0 and 254) within the parameter register file of the FPGA. Put the read value into <i>value</i> . Returns true if the read succeeds, else return false. Similar to <i>sendParamRegisterWrite</i> , this function compensates for changes in endianness on the host machine.
BOOL sendRun()	Start execution on the FPGA by setting the signal <i>userRunValue</i> true. Returns true if the start command is accepted, else return false.
BOOL waitDone(uint8_t maxWaitTime)	Wait up to <i>maxWaitTime</i> seconds for execution on the FPGA to complete. Returns true if execution completes, else return false. Completion is indicated by the user circuit resetting the <i>userRunValue</i> via the <i>userRunClear</i> line. Further details regarding the hardware API are provided in the next section.
BOOL sendReset()	Abort execution of the user circuit and return control of the I/O buffers and parameter registers to the API controller by asserting the <i>userSideReset</i> line. While this also resets <i>userRunValue</i> , it does not change anything else on the FPGA (for example, the contents of I/O buffers and parameter registers are unchanged aside from any values that the user circuit might have modified while the <i>userRunValue</i> was true). Returns true if the reset command completes, else returns false.
BOOL sendWriteAndRun(uint32_t startAddress, uint32_t inLength, uint8_t *inData, uint8_t maxWaitTime, uint8_t *outData, uint32_t maxOutLength, uint32_t *outputLength)	Essentially a combination of the <i>sendWrite, sendRun, waitDone</i> and <i>sendRead</i> commands. Potentially harder to use than separate commands, but also likely much faster. Send a block of data from the PC (pointed to by <i>inData</i> of length <i>inLength</i>) to the input buffer of the FPGA (starting at <i>startAddress</i>). When the data has been sent to the FPGA, start execution and wait up to <i>maxWaitTime</i> seconds for execution to complete. When it is done, read up to <i>maxOutLength</i> bytes of results back from the output buffer, beginning at address 0. Put the results retrieved from the FPGA's output buffer into <i>outData</i> . The hardware-side API monitors writes made to the output buffer by the user's circuit during the execution phase of this command. The highest address written to the output buffer during a given execution complete is used to determine how many bytes.
	should be read back from the device. The number of bytes read back will be

	placed in <i>outputLength</i> . Returns true if the entire process succeeds, else returns false.
	If the function fails with a <i>FAILCAPACITY</i> error, the only problem was that the <i>outData</i> buffer was too small to copy all of the generated results from the FPGA back to the host. If this occurs, the <i>outputLength</i> variable will not contain the number of bytes returned, but rather the total number of bytes the execution phase wanted to return (the number of bytes actually returned will be <i>maxOutLength</i>). The user can then simply read the "overflow" bytes from addresses [maxOutLength] manually with a
	subsequent <i>sendRead</i> command.
	If the function fails with a <i>FAILREADACK</i> error, the device failed to fully respond during the readback phase of the command, despite retry attempts. The <i>outputLength</i> variable will not contain the number of bytes returned, but rather the number of bytes the execution phase wanted to return. The state of <i>outData</i> is unknown, but at least some output data has been partially written. In theory the user could elect to read the entire buffer from {0, <i>outputLength-1</i> } again with a separate <i>sendRead</i> command (despite the fact retries were already made and there is probably some other issue in the system). This option may be attractive if calling <i>sendWriteAndRun</i> is not easy.
	For example, if <i>inData</i> and <i>outData</i> point to overlapping addresses in the
	same array, it may be simpler to try and re-read <i>outData</i> rather than recreating <i>inData</i> so that execution can be attempted again.
BOOL sendConfiguration(char *path)	Configure the FPGA using iMPACT and a programming cable from the bitstream file at <i>path</i> . Return true if reconfiguration succeeded, else return false. Before using this option, please make certain to properly define the following constants in <i>eth_FPGA.h</i> : IMPACT PATHTOIMPACT PATHTOIMPACTTEMPLATEBATCHFILE PATHTOIMPACTPROGRAMMINGBATCHFILE PATHTOIMPACTPROGRAMMINGBATCHFILE IMPACTSUCCESSPHRASE
	Also, ensure that the batch command template file (PATHTOIMPACTTEMPLATEBATCHFILE) is appropriate for your setup. The included template file has been tested with the <i>Platform Cable USB II</i> device and the XUPV5/ML505/ML507 boards. See the "Programming FPGA & Execution of Example" section for more information.
	Future versions of this function will provide direct configuration over the Ethernet connection. This will eliminate the need for the host PC to have iMPACT and a programming cable installed. Furthermore, this will make reconfiguration much faster.

As will be discussed in more detail in the "Hardware Interface" section, the connections of the Verilog hardware API contain a little-endian bus structure. For example, a 4-byte bus would be indicated with *connection [31:0]*. In this case, the most significant byte would be held in *connection [31:24]* and the most significant bit of that byte be held in *connection [31]*. To maintain code portability with all host PCs, the software-side API will try to conserve the bit and byte ordering used on the host PC and adjust if possible. For example, when *sendParamRegisterWrite* is called on a big-endian byte-ordered PC, the software API will convert the big-endian integer parameter to little-endian byte ordering before transmission to the FPGA. Similarly, *sendParamRegisterRead* will convert the little-

endian integer response to a big-endian byte-ordered integer upon receipt. However, no such endian conversion is performed for the *sendWrite* and *sendRead* functions, as the *buffer* parameter is a pure byte-wise array.

4. Hardware Interface

As seen in Figure 1, the user's circuit on the FPGA primarily interacts with the host PC through three memories: an input buffer, and output buffer, and a set of parameter registers. In the current implementation, these memories are implemented with onboard BlockRAM and the size of the input and output buffers are customizable. The customization process is described in more detail in the "Compiling Hardware" section.

Since the system operates in a supervisor/worker style mode, a user interface signal, *userRunValue* also signifies which device has read/write access to the User device interface. The following specifies the behavior:

- Logic 0 the host has R/W control of all API logic state. All user logic writes will be ignored and all reads will return zero values.
- Logic 1 the user logic has R/W control of Input/Output BlockRAM and Parameter Register file. No host changes are possible. The user logic relinquishes control by clearing the *userRunValue* (user logic asserts *userRunClear*). Control can be returned to the host (in the event of a problem with the user logic) by calling the *sendReset* function.



FIGURE 1: USER HARDWARE INTERFACE

Notes:

- 1) Input Buffer is used by the *sendWrite* function e.g. data is moving from the PC to the FPGA
- 2) Output Buffer is used by the *sendRead* function e.g. data is moving from the FPGA to the PC.

Although the API controller itself requires a small handful of very specific clocks to communicate correctly with the board-level devices, the clock frequency of the interface between the API controller and the user's logic (*userInterfaceClk*) can operate at any frequency - arbitrarily low or high, subject to the compilation tools still being able to place and route within the proper timing constraints. The system has been successfully tested with *userInterfaceClock* set from 7 to 333 MHz, depending upon the target FPGA, desired size of I/O buffers and complexity of user logic. The default setting in the example code is 167 MHz.

The following table provides more specific details for the interface signals that the user's logic module should take as inputs and should provide as outputs.

userInterfaceClk	input	Clock to which all interface signals are aligned. Will function at any frequency (subject to ISE being able to meet timing constraints). User circuit can also use other clock signals if desired, but signals to hardware API must be aligned to this clock.
userLogicReset	input	User circuit should reset when this signal is asserted. Then this signal goes true, the hardware API will reset <i>userRunValue</i> automatically, reclaiming control over the I/O buffers and register file.
userRunValue	input	User circuit should begin execution when this signal is asserted. The hardware API gives control over the I/O buffers and parameter registers to the user circuit while this signal is asserted.
userRunClear	output	Asserting this signal will reset <i>userRunValue</i> . This indicates that the user circuit has completed computation and wants to give control over the I/O buffers and parameter registers back to the hardware API.
register32CmdReq	output	Asserting this line indicates that the user circuit would like to perform a read or write to the register file (depends upon <i>register32WriteEn</i>). The command is accepted by API controller when both <i>register32CmdReq</i> and <i>register32CmdAck</i> are true for 1 clock cycle.
register32CmdAck	input	This line is asserted when the API controller has accepted the read or write request.
register32WriteData	output[31:0]	Data to write to the parameter register file.
register32Address	output[7:0]	Address line to parameter register file.
register32WriteEn	output	Write enable to parameter register file.
register32ReadDataValid	input	This line is asserted when the register file has returned with data from a read request. The data will only be valid while this line is true.
register32ReadData	input[31:0]	Data read back from parameter register file. Will only be valid while <i>register32ReadDataValid</i> is true.
inputMemoryReadReq	output	Asserting this line indicates that the user circuit would like to perform a read from the input memory buffer. The read command is accepted by API controller when

		both <i>inputMemoryReadReq</i> and <i>inputMemoryReadAck</i> are true for 1 clock cycle.
inputMemoryReadAck	input	This line is asserted when the API controller has accepted a read request.
inputMemoryReadAdd	output[N:0]	Address of read request. Parameterized, more detail in the "Compiling Hardware" section
inputMemoryReadDataValid	input	This line is asserted when the input memory buffer has returned with data from a read request. The data will only be valid while this line is true.
inputMemoryReadData	input[M:0]	Data read back from the input memory buffer. Will only be valid while <i>inputMemoryReadDataValid</i> is true. Parameterized, more detail in the "Compiling Hardware" section
outputMemoryWriteReq	output	Asserting this line indicates that the user circuit would like to perform a write to the output memory buffer. The write command has only accepted by API controller when both <i>outputMemoryWriteReq</i> and <i>outputMemoryWriteAck</i> have been true for 1 clock cycle.
outputMemoryWriteAck	input	This line is asserted when the API controller has accepted a write request.
outputMemoryWriteAdd	output[N':0]	Address of write request. Parameterized, more detail in the "Compiling Hardware" section
outputMemoryWriteData	output[M':0]	Data of write request. Parameterized, more detail in the "Compiling Hardware" section
outputMemoryWriteByteMask	output[log2(M'/8):0]	Byte-wise write enable of write request. Parameterized, more detail in the "Compiling Hardware" section

Figure 2 describes the request/acknowledge logic of the user interface in more detail. In the top diagram of Figure 2, the user circuit would like to submit a read request (asserting *inputMemoryReadReq*). The user circuit must make the read address (*inputMemoryReadAdd*) valid the same cycle that the request line is asserted. If the acknowledge line (*inputMemoryReadAck*) is low, the request has not been accepted by the controller logic until both the request and acknowledge signals have been asserted for 1 clock cycle. Before this occurs, the user logic may lower the request line to cancel the request before it is accepted. After the read request is accepted, the controller will return with the read data (*inputMemoryReadDataValid* is asserted and the data is presented on *inputMemoryReadData*). As seen in the middle diagram of Figure 2, the acknowledge line may already be asserted before the user logic submits a request. In this case, the read request will be accepted the same cycle that *inputMemoryReadReq* is asserted. Extending this concept, as seen in the bottom diagram of Figure 2 multiple reads may be submitted in consecutive cycles if conditions allow – again, a request is accepted any cycle that both *inputMemoryReadReq* and *inputMemoryReadAck* are asserted.

The parameter registers and output memory follows a similar request/acknowledge scheme. In the case of a write, the address, data (and, in the case of the output memory byte mask) signals must begin to be valid the same cycle that the request line is asserted.



FIGURE 2: TIMING DIAGRAM FOR INPUT & OUTPUT MEMORY REQUEST/ACKNOWLEDGE SIGNALS

5. Compiling Hardware

The existing codebase should work with any "transceiver" Virtex 5 or Virtex 6 (LXT, SXT, TXT, FXT but not the baseline LX series). Compatibility through ISE 13.2 has been verified, and we do not expect any issues with newer versions of the tools. All sample screenshots depict ISE 13.2 and CORE Generator version 13.2. The user interface for new versions of the tool may change slightly, but all of the same options should be available.

If you are using ISE 13.2 or newer and targeting the ML50X/BEE3/ML605, you may use one of the example project folders (*HW_Example_13_2_X*) and skip to section III below (generating the needed CoreGen pieces). If using the ML50X/XUP or BEE3 example projects, please double-check that the device settings and the included .ucf match those needed for your board.

If you are not using ISE 13.2 or newer (or otherwise have trouble using the provided example projects), use the following directions to create a new project.

- I) Create a new project in Xilinx Project Navigator
 - a. Begin a new project from "File->New Project". Enter a name and location for the project, making sure to select "HDL" as the top-level source.

🍃 New Project	t Wizard	×
Create New Pro Specify project locatio Enter a name, locat	ject n and type. ons, and comment for the project	
Name:	HW_13_2_Example	
Location:	ht\SystemIntegration\EthernetAndRAM\sourceDepot\XUP_ethernet\HW_13_2_Example	
Working Directory: Description:	nt/SystemIntegration/EthernetAndRAM/sourceDepot(XUP_ethernet(+W_13_2_Example	
Select the type of to Top-level source ty HDL	op-level source for the project pe:	•
More Info	Next	ncel

b. Select the appropriate device, package and speed for your platform. ISE 13.2 has specific options for the ML505/506/507/605 boards, but just for completeness below are the parameters for the boards we directly support. If you port this system to any other platforms, please let me know.

Platform	Device	Package	Speed
ML505	XC5VLX50T	XC5VLX50T	
ML506	XC5VSX50T	EE1126	1
ML507	XC5VFX70T	FF1120	-1
XUPV5	XC5VLX110T		
	XC5VLX110T		
BEE3	XC5VLX155T	FF1136	-2
	XC5VSX95T		
ML605	XC6VLX240T	FF1156	-1

c. Ensure that "Verilog" is at least part of the parameter selected for the remaining project options and, if simulating is something that you would like to do (see Section 7 for more details), select the proper simulator. Finally, click "Next".

project	
Value	_
Virtex 6 ML605 Evaluation Platform	
All	-
Virtex6	-
XC6VLX240T	-
FF1156	-
-1	-
HDL	-
XST (VHDL/Verilog)	-
Modelsim-SE Verilog	-
Verilog	-
Store all values	-
VHDL-93	-
	Value Virtex 6 ML605 Evaluation Platform All Virtex6 XC6VL240T FF1156 -1 HDL XST (VHDL/Verilog) Modelsim-SE Verilog Verilog Store all values VHD1-93

d. We will be adding existing sources later, so click "Next" until prompted to "Finish".

158	New I	Project Wizard - Add Existing Sources		x
Ē	Add ex	isting sources		
		Source File	Copy to Project	Add Source
	1			Remove
Ac Sc	lding e ource''	xisting sources is optional. Additional sources can be or "Project->Add Copy of Source" commands.	added after the project is created usin	g the "Project->Add
	<u>M</u> ore	Info	< <u>B</u> ack <u>N</u> ext >	Cancel

- II) Add all of the appropriate source files to your project.
 - a. Right-click in the "Design->Hierarchy" window and select "Add Source..." and browse to the SIRC_INSTALL_PATH\HWSrc directory (if installed to the default location, it will be in

<u>C:\Program Files\Microsoft Research\Simple Interface for Reconfigurable Computing (SIRC)\HWSrc\</u>

or

<u>C:\Program Files (x86)\Microsoft Research\Simple Interface for Reconfigurable Computing (SIRC)\HWSrc</u>



- b. Add all of the .v files in the appropriate platform directory and <u>one</u> .ucf file for your platform (e.g. XUPV5system.ucf if targeting the Virtex 5 XUP board, ML505system.ucf if targeting the ML505, etc.).
- c. Double-check in the "Design->Hierarchy" window to ensure that the system module is the top-level module. This is indicated by a green icon to the left of the system module. If this module is not the top-level module, make it the top-level module by right-clicking the system module and selecting "Set as Top Module".



This project also relies on a few free cores generated by CORE Generator. We cannot re-distribute Xilinx's IP, so the user must generate those cores themselves. Below are instructions that describe how to generate the proper cores. Although some of the older versions of the CORE Generator modules are no longer available with newer versions of the tool, this does not seem to be an issue. Most of the differences in the CORE Generator versions are indicated below.

- III) EMAC Wrapper
 - a. Start CORE Generator from "Start→Programs→XILINX_DIRECTORY→ISE Design Tools-> 32/64-bit Tools →CORE Generator" (the exact menu path will differ slightly from machine to machine and version to version). CORE Generator can be started from within Project Navigator, but we have had problems with code generated when CORE Generator is started in this way.

1 Xilinx CORE Generator - No Project		
<u>File View Manage IP Help</u>		
🗋 ờ 🔒 🗐 🗣 📂 📓 K?		
If P Catalog View by Function View by Kame	ð×	logiCŘRE Xilinx CORE Generator
Image: Section 2014 Addenotive & Industrial Image: Section 2014 Image: Section 2014 Image: Section 2014 Image: Sectio	Statu	There is no project open. You may howse the IP Catalog but you will not be able to generate any cores until you open or create a project.
 Storage, NAS and SAN Video & Image Processing 		Canade et al. Nov. CORE Generator. Yelds protein installand. Oracefon has been installand. Concerlo has been configured at with the following Nam repositories: ~ C.VMIN.13.7USD_05155C vorgency [Samp existing al_mdex.xml] The JP classing has been included. New Project Cancelled.
Ceatlog: Gee Al IP versions Only IP compatible with chosen	n part	Search Cgroole Find Save Gear Information 🔬 Womings 📀 Errors
		Part: Unset Design Entry: Unset 🥥

- b. Create a new project within the ISE project directory created in step 1. A handy place to create this project is in YOUR_PROJECT_PATH\ipcore_dir
 - i. Select the proper part for your board (for example, the ML505 boards use a Virtex5 xc5vlx50t-ff1136-1 part and the ML605 board uses a Virtex 6 xc6vlx240t-ff1156-1 part)

V Project Options		? <mark>X</mark>
Project Options Part Generation Advanced	Part Select the part for your project: Fagily Virtex6 Degice xc6vk240t Package ff1156 Speed Grade -1	2 ×
	QK Cancel Apply) Heb

ii. Switch to the "Generation" Tab and change "Design Entry" to Verilog.

V Project Options	? ×
Generation Advanced	Flow Flow
	Flow Settings yendor Other Netlist Bus Format B <n:m></n:m>
	Simulation Files Preferred Language Image: Structural Image: Structural Image: Structural Image: Structural
	Unter Output Hooucts
	QK <u>C</u> ancel <u>Apply</u> <u>H</u> elp

- iii. The default options should be acceptable for other options.
- c. <u>If using a Virtex-5</u>, select "Communications & Networking→Ethernet→Virtex5 Embedded Trimode Ethernet MAC Wrapper", version 1.8.

<u>If using a Virtex-6</u>, select "Communications & Networking→Ethernet→Virtex-6 Embedded Trimode Ethernet MAC Wrapper", version 1.5

If only later versions of the core are shown (these were available as of CORE Generator 13.2.61xd), select "Show->All Versions" in the main window. Newer versions of the core may work, but Xilinx changes the interfaces that they use from time to time. Xilinx has been migrating away from the Local Link interface in lieu of the AXI interface (in versions 2.0+ of the wrappers). Future updates of SIRC will support wrappers that use the AXI interface.



- d. Double-click to select and customize
 - i. Name The existing codebase is expecting the module to be named "emac_single".
 - ii. <u>IF USING A VIRTEX-5</u>: Select only one EMAC. I have tested with EMAC0, but EMAC1 should work as well. This is not an option if using a Virtex-6 board.

Virtex-5 Embedded Tri-Mode Etherne	et MAC Wrapper		
View Documents			
IP Symbol	ē ×		1 - · · · · ·
RESET		Virtex-5 Embedde Ethernet MAC V Component Name i emac_single Shared bitraficos Host interface configuration. Host Type The host interface type is shared between the two EMACs. D DR Host None	d Tri-Mode Yrapper stim.com.ipvd_emec.1J DCR-specific options DCR Base Address (EMAC 0) 00 DCR Base Address (EMAC 1) 01
CLIENTEMACO_SPEEDIS10100	EMACOCLIENTERRETRANSMIT EMACOCLIENTEXSTATS EMACOCLIENTEXSTATSVLD EMACOCLIENTEXSTATSVLD EMACOCLIENTEXSTATSVEVLD EMACOCLIENTEXSMINICLAUDUT	Enable EMACs Enable EMAC 0 Enable EMAC 1	
CLENTEMACHROLENTOLIKI	→ BMACICLENTRICICENTCLADUT BMACICLENTRICIC) → BMACICLENTRICVLD → BMACICLENTRICVLD → BMACICLENTRICVLDADBW → BMACICLENTRICADPRAME → BMACICLENTRICADPRAME → BMACICLENTRICADPRAME → BMACICLENTRICADPRAME	Note that the provided demonstration testbanch will ove GUI via a DCR, Host or MDID interface. Please refer to th Started Guide.	write a subset of the emac configurations defined in the e "Demonstration Testbench Tasks" section in the Getting
	EMAC1CLIENTRISTATSVLD A EMAC1CLIENTRISTATSBYTEVI D	Qatasheet Sack Page :	L of 3 Next > Generate Gancel Help

iii. As of CORE Generator 13.2.61xd, the rest of the settings should remain at their default values. The critical values to double-check are Physical Interface: GMII and Speed: 1000 Mbps

iew Documents						
Symbol	ð ×	LOGICARE	/irt	ex-6 Emb	edded Tri-Mode	•
RESET				thernet	MAC wrapper	xilinx.com:ip:v6_emac:1
CLIENTEMACR/ICLIENTCLKIN	EMACCLIENTRAD[7:0]	Company of the		and a standa		
CLIENTEMACTICCENTCCKIN		Component Nar	me	emac_single		
CLENTEMACTION D					Inter	rface Configuration
CLENTERACTADVED	- FINACCUENTRADODERAME	Physical Inter	face		Client Interface	
CUENTEMOTYERSTOVE						
CUENTEMACTI INCERSI IN	DIACOURATE/STATE	PHY Interface	GMII	•	Data Width	
CUENTEMACTORODEL AVIT-D	A ENACCI ENTRYSTATION D	Speed		Overclocking	 Ø 8-bit 	
CLIENTEMACPHUSERED	EMACCUENTRISTITIBUTEVLD	() Tri-man		(B. None	() 16-bit	
CLIENTEWACEWUSEVALUS	-> EWACOLIENTTXOLIENTOLIKOUT	O mapee		@ none		
PHYEMACOTX CLK		1000 Mb	ops	2000 Mbps	Option to reduce BUFG use	
PHYEMACT/IGM/IMIICLKIN		10/100 f	Mbps	2500 Mbps	Clash Fashla	
EMAC_SPEEDIS10100					Clock Endore	
-						
		Host and Man	agemer	nt Interfaces		
		Host Type	DCR-	specific options		
		0.000	DCP.	On saddress	Panna: 00 EE	
HOSTOLK	HOSTWR0474(31.0)	Our buse rube tables Our buse rube ta				
HOSTOPCODE(1 c)						
HOSTADORIS IS-		A Mono	-			
HOSTRODATA(11 10-00		e none	1.16	nable MD10		
HOSTMIMSEL						
HOSTRED		SGMII Capabi	lities			
DOREMACCUX		@ 10/100/10	000 Mb/	s (clock tolerance cor	npliant with Ethernet specification)	
DOREMACABUB(1 U)		- 10/100/10	in the	- Frankrike ditektoren	for dealers on the most while	
DCREMACREAD	> EMACDORACK	10/100/10	runa mitaj	a (reaunced tolerand	e for Gookay on 100/1000 (MD/S	
DCREMICWRITE						
DCREMACDBUS(113)		🗌 Include SC	SMII / 1	000BASE-X mode swi	tching block in wrapper	
DOREMACENABLE						
DCRHOSTDONEIR		[Barbarbard]			The second second second	

- e. Select "Finish" or "Generate" and CORE Generator will create the core's logic. A new "emac_single" directory will be created within the Core Gen project directory.
- f. Add the following 7 files from the newly generated core to the ISE project. This can done by right-clicking in the "Sources" window and selecting "Add Source..."



- i. emac_single\example_design\client\fifo\eth_fifo_8.v
- ii. emac_single \example_design \client \fifo \rx_client_fifo_8.v
- iii. emac_single \example_design \client \fifo \tx_client_fifo_8.v
- iv. emac_single \example_design \physical \gmii_if.v

- v. emac_single \example_design \emac_single.v
- vi. emac_single \example_design \emac_block.v
- vii. emac_single \example_design\emac_locallink.v
- g. Double-check that the system/E2M/emac_ll module (and the modules used inside of this module) have been recognized by ISE. When browsing the design in the "Sources" window, there should no longer be a question mark inside the system/E2M/emac_ll module document icon (nor a question mark inside the icons of system/E2M/emac_ll/emac_block,, etc.).



- IV) Input/Output buffers and Parameter Register File
 - h. Using the same CORE Generator project as the EMAC wrapper, generate the logic for the parameter register file. Under the "View by Function" tab, select "Memory & Storage Elements→RAMs & ROMs→Block Memory Generator", version 6.2



- i. Double-click to select and customize
 - i. Generate core with the settings:
 - Block Me Documents ⊻iew IP Symbol e × LogiCXRE **Block Memory Generator** Com ent Na ne blk mem gen paramReg Interface Type Native O AXH Nat Mei of the LogiCORE Block tive Interface Block Me IG) are the original standard BMG DP), True Dual Port Single Port RAM (SP), Sir Native Port Ri based < Back Page 1 of 6 Next > Generate Gancel Help 🕴 IP Symbol 🍳 Power Estimatio Datasheet
 - 1. Name: "blk_mem_gen_paramReg" and select the "Native" interface type

2. Memory Type: True Dual Port RAM

Documents View			
P Symbol	# × LogiC [©] RE	Block Memory Generator	xilinx.com:ip:blk_mem_gen:6.2
	Memory Typ	True Dual Port RAM	
400843 0	Clocking O	Ions CAUTION (Potential Collision Issue): Refer to UG36. Avoidance' section, when using TDP Memory, with n Clock Read First in conjunction with asynchronous clocking	i3, 'Conflict Write Mode =
	ECC Option		3
REDCEA	вотеля ЕСС Туре	No ECC +	
RSTA	RDADDREDCI3-0	r Injection Pins Single Bit Error Injection +	
INJECTOBITERR	Use Byt	s Write Enable	
INJECTOB/TERR	Byte S	re 9 - bits	
ADDRB[1:0]	Algorithm		
ENS	Defines the more infor	algorithm used to concatenate the block RAM primitives. See the vation.	e datasheet for
WEB[0:0]	 Minimu 	n Area	
CLKB	Eow Po Fixed P	/er Imitives	
	Primiti	e (Write Port A) : Rio2 + Primitive(s) Used : 16io2, 8io2	

- 3. Port A and B Options
 - a. Memory Size Write Width: 32
 - b. Memory Size Write Depth: 255
 - c. Memory Size Read Width: 32
 - d. Operating Mode: Write First
 - e. Enable: Always Enabled

Slock Memory Generator	18 May 1998			
Documents View				
IP Symbol	#×	Port A Options Memory Size Write Width 32 Range: 11152 Res	enerator	xilinx.com:ip:blk_mem_gen:6.2
ADDRA/7 01	DOUTA(31.0)	Write Depth 255 Range: 29011200 Rea	id Depth: 255	
ENA	→ SB/TERR → DB/TERR → RDADDREC0[7:0]	Operating Mode Write First Read First No Change	Enable Always Enabled Use ENA Pin	
		Port 8 Options Memory Size	Road Weth- 22	
ADDRB[7:0]		Write Depth: 255	Read Depth: 255	
ENB		Operating Mode Write First Read First No Change	Enable Always Enabled Use ENB Pin	
IP Symbol Q Power Esti	mation	Detasheet	< Back Page 3 of 6 ∐ext > G	enerate <u>C</u> ancel <u>Help</u>

4. No registering for either port (double-check that none of the "Optional Output Registers" boxes are selected)

Bourners Jew Psymbol Image: Second part of Sec	Q Block Memory Generator		
Projection 0 x x ADDORF 09	Documents View		
ADOVED 00	IP Symbol	In the second se	xilinx.com:ip:blk_mem_gen:6.2
ACONUT 01 Solumers 0 CHARGE 03 Selecter Port A Output of Memory Core CHARGE 04 Selecter Port A Output of Memory Core REGRAM Selecter Port B Output of Memory Core Regram Port 8 Regram Port 8 Output of Memory Core Regram Port 8 Regram Port 8 Output of Memory Core Regram Port 8 Regram Port 8 Output of Memory Core Regram Port 8 Regram Port 8 Output of Memory Core Regram Port 8 Regram Port 8 Output of Memory Core Regram Port 8 Regram Port 8 Output of Memory Core Regram Port 8 Regram Port 8 Output of Memory Core Regram Port 8 Regram Port 8 Regram Port 9 Regram Port 9 Regram Port 9 Regram Port 9 Regram Port9 Regr		Optional Output Registers	
Aconcyr 19		Port A	
AcoMy Pi		Register Port A Output of Memory Primitives	
Image: solution in the second state	40084/201	Register Port A Output of Memory Core	
NU Sentral VMACTSHIRE Contrast Register Fort 8 Output of Memory Printitives Register Fort 8 Output of Memory Printitives Register Fort 8 Output of Sentral NACTSHIRE NACTSHIRE Content NACTSHIRE	DINA(31:0)	Register Port A Input of SoftECC logic	
HESCA	ENA	Use REGCEA Pin (separate enable pin for Port A output registers)	
wik489 → GUIDERCOTE wik489 → GUIDERCOTE uktoria → GUIDERCOTE	REGCEA		
CXX→ Register Fort 8 Output of Memory Vienthess INACCOMPRE Department of Software Vient 8 output registers) INACCOMPRE Department of Software Vient 8 output registers) INACCOMPRE Max Ster: 8 output registers) INACCOMPRE Image: I		Port B	
NullCitations Sourage reg NullCitations Sourage reg Accession for NullCitations Register Fort 8 Output of SoftECC logs Control reg Use Register Fort 8 Output of SoftECC logs Control reg Null SoftEr (Second to control to	сыха — э	Register Port B Output of Memory Primitives	
Accessor of multicides → booldiges → booldiges Accessor of multicides → booldiges → booldiges Accessor of multicides → booldiges		Register Port B Output of Memory Core	
Accestor in the solution of th	INJECTSBITERR	Register Port B Output of SoftECC logic	
Accesser ::::::::::::::::::::::::::::::::::		 Use REGCEB Pin (separate enable pin for Port B output registers) 	
Default of	ADDR8[7:0]	Simpling Change within Mur 0	
ton → Memory Initialization Weston → Remaining Memory Locations Remaining Memory Locations Remaining Memory Locations	DIN8(31:0)	Pipeline Stages within Mox 0 + Miux Size: 0x1	
I Load Stat Fife Coa Stat Fif	ENB	Memory Initialization	
non	WERGOLD	E Lord Talt File	
CLUE	R378		
Fill Remaining Memory Locations Remaining Memory Locations (Nex)	CLKB	Coe File no_coe_tile_loaded	Browse Show
Remaining Memory Locations (Hex) 0		Cill Camaining Mamory Locations	
Nemaning memory Lucasians (nex) (r		Description Memory Locations	
		Remaining Memory Locations (Hex)	
D Example D Prover Estimation Qatasheet Qatasheet ganerate ganerate ganerate ganerate ganerate ganerate https://www.estimation.com https://www.estimation.com https://www.estimation.com"/>https://www.estimation.com	the Symbol O Power Estimation	Datasheet < Back Page 4 of 6 Mer	d > Generate Gancel Help

5. Disable simulation warnings by checking both the "Disable Collision Warnings" and "Disable Out of Range Warnings" boxes.

Subclass and additional set of the set of t	Documents ⊻iew ₽ Symbol & 2	Block Memory Generator slin.com.ip.blic_mem_pen:6.
Address Width 2: 8 Color: 10 Co	АССЯ47 8 — → 20078/11 8] БИКИ 8 — → КПССА - → - 4 БИТИИ МИД 2 — → 5 БИТИИ МИД 2 — → - 5 БИТИИ КПЛ - →	Structural/Unism Simulation Model Options Defines the type of warnings and outputs are generated when a read-write or write-write collision occurs. All None Warning Only Generate X-only Behavioral Simulation Model Options Discussion Simulation Model Option Model Discussion Simulat
The Block Memory Generator core is not fully backward compatible with the Single Port and Dual Port Block Memory cores. Please see the datasheet for more information.	Contentioners → → 500/18(118) ACCRET () → Exact → Exa	Information Memory Type: True Dual Port RAM Block RAM resource(s) (IBK BRAM): 0 Block RAM resource(s) (IBK BRAM): 1 Total Furt A Read Latency (From Rusing Edge of Read Clock): 1 Clock Cycle(s) Total Port A Read Latency (From Rusing Edge of Read Clock): 1 Clock Cycle(s) Address Widb A: 8 Address Widb A: 8 The Block Memory Censer rore is not fully backward compatible with the Single Port and Dual Port Block Memory Cores: Rease see the distance for more information.

- j. Select "Finish" or "Generate" and CORE Generator will create the core's logic. A new "blk_mem_gen_paramReg.xco" file will be generated within the Core Gen project directory.
- k. With the same method as used for the Ethernet source files, add the "blk_mem_gen_ paramReg.xco" file to the ISE project and double-check that the system/E2M/ethernetController/blk_mem_gen_paramReg module has been recognized by ISE. Notice that the icon will not turn into a small document with a "V", but rather into a small CORE Generator lightbulb.



- I. Repeat the process to generate another block memory with the setting:
 - i. Name: "blk_mem_gen_inputMem", Interface Type: Native and Memory Type: Simple Dual Port RAM (Notice, this is not a True Dual Port RAM as with the parameter register file)

G Block Memory Generator	Percent copyright in more Reduction's one final parts	
Documents View		
Block Memory Generator Documents Documents View # Symbol Accoration Conversion Conversion	Block Memory Generator Memory Type Single Dual Pert RAM Clocking Options Cocking Options Coc Type Common Clock ECC Type BCC Type BCC Type BCC Type BCC Type BCC Common Clock BCC System BCC Structure BCC Book BCC System BCC Book BCC System BCC Book BCC System BCC Book BCC BCC BCC BOOK BCC BCC BOOK BCC BOOK BCC BOOK BCC BOOK BCC BOOK BCC BCC BOOK BCC BCC BOOK BCC BCC BCC BCC BOOK BCC BCC BCC BCC BCC BCC BOOK BCC BCC BCC BCC BOOK BCC BCC BCC BCC BOOK BCC BCC BCC BCC BCC BCC BOOK BCC BCC BCC BCC BCC BCC BOOK BCC	xilm.com;p;bli_mem_gen:6.2
P Symbol Power Estimation	Actual Primtive(s) Used : 1662, 862	Generate Cancel Help

- ii. Port A Options
 - 1. Memory Size Write Width: 8
 - 2. Memory Size Write Depth: <u>As per the user's requirements</u> (discussed in section v below, but <u>if you use the default settings in the "system.v" file you should use 131,072</u>).
 - 3. Enable: Always Enabled
- iii. Port B Options
 - Memory Size Read Width: <u>As per the user's requirements</u> (discussed in section vi below, but <u>if you use the default settings in the "system.v" file you should</u> <u>use 8</u>)
 - 2. Enable: Always Enabled

G Block Memory Generator	No on a sporter of some Walkelberg, our from per-	
Documents View		
IP Symbol & X	Ingicience Block Memory Generate	Dr xilins.com:ip:bil_mem_gen:6.2
_	Memory Size Write Width 8 Range: 11152 Write Depth 131072 Range: 2901200	
ADDR4/16 50→ DN4/7 30→ ENA → WE4D 01→ → SB/TENR → SB/TENR	Operating Mode Write First Read First No Change	Enable Alvaya Enabled Use ENA Pin
	Port & Options Memory Size Read Width (8 • Read Depth: 131072	
EN0 → REDCD → RET → CL0 →	Operating Mode Write First Read First No Change	Enable Alvays Enabled Use EN8 Pm
yr Symbol Q Power Estimation	Detacheet	lack Page 3 of 6 (Best >) Generate) Cancel (Belp)

- iv. Output Registers
 - 1. No registers on Port A
 - 2. Register Port B Output of Memory Core: <u>As per the user's requirements</u> (discussed in section vii below, but <u>if you use the default settings in the</u> <u>"system.v" file you should check the "Register Port B Output of Memory Core"</u> <u>option</u>).

Q Block Memory Generator	Per on congenitation range Madeulaurice and Haampark	
Documents View		
P Symbol 8 X	Coptional Output Registers Port A Biggister Port A Output of Memory Prenetives	xilinx.com:ip:blk_mem_gen:6.2
ADDRA(16.0)	Register Port A Output of Memory Core Register Port A Input of SoftECC logic Use REGCEA Pin (separate enable pin for Port A output registers) Port 8	
	Begister Port B Output of Memory Primitives Kegister Port B Output of Memory Core Begister Port B Output of SoftCo Expec Use RESCES Pin (separate enable pin for Port B output registers) Popeline Stages within Max 0 • Max Size: Ox1	
0.04 REDCS RT(10 CLVS>	Memory bitisitzation Load init File Core File [no_core_file_loaded EE Remote the entriese	Browse Show
P Symbol Q Power Estimation	Remaining Memory Locations (Hei) 0 Qatasheet < gack	te <u>Cancel</u> <u>Help</u>

- v. Simulation Model Options
 - 1. Disable simulation warnings by checking both the "Disable Collision Warnings" and "Disable Out of Range Warnings" boxes

Block Memory Generator	Parameter aggregation of managements and the state of the state of the	
Documents View		
P Symbol & ×	logi ^C [©] [₹] Block Memory Generator	xilinx.com:ip:blk_mem_gen:6.2
LODAVE 8 - СКА КАСТИВИТИЯ СКА КАСТИВИТИЯ КАСТИВИТИЯ АСОНЕНИИ КАСТИВИТИЯ СКА КАСТИВИТИЯ СКА СКА СКА СКА СКА СКА СКА СКА ССИ СС	Structural/Uniam Simulation Model Options Defines the type of warnings and outputs are generated when a read-write or write-write collision or	t Block Memory cores. Please see the

- vi. Regarding user customization, the default parameters at the top of the "system.v" file define a 2^17 = 131,072 = 128KB memory (INMEM USER ADDRESS WIDTH = 17) with a user-side interface of 1 byte wide (INMEM USER BYTE WIDTH = 1). The depth for both ports and the port B read width (as powers of 2 number of bytes) can be changed to fit the user's needs. For example, if we wanted to create a larger 512KB buffer with a user side interface of 32-bits wide, we would update the .xco in CORE Generator by:
 - 1. Leaving the port A write width at 8
 - 2. Changing the port A write depth to 524,288 (step ii.2 above)
 - 3. Changing the port B read width to 32 (step iii.3 above, which automatically updates the port B read depth to 131,072)
 - 4. Changing two parameters in the "system.v" file INMEM_USER_BYTE_WIDTH = 4 and INMEM_USER_ADDRESS_WIDTH = 17

Any time that the user wishes to change the size or arrangement of the input memory, the appropriate parameters in the "system.v" file should be updated and CORE Generator should be re-run on the blk_mem_gen_inputMem module with the appropriate arguments.

One note, if the user-side interface is larger than 1 byte wide, the *inputMemoryReadData* line will be organized as little endian. For example, if the interface is 32-bits wide,

inputMemoryReadData[31:0] = {byte3, byte2, byte1, byte0}

- vii. As for registering the port B output, this may be necessary to meet the desired timing if the input memory itself is large or if fanout of the input memory to the user's circuit is large. Register this output increases the read latency of the memory 1 or 2 clock cycles (although any changes in the latency should not require changes to the user's circuit due to the request/acknowledgement handshaking on the interface). The user may elect to select "Register Port B Output of Memory Primitives", "Register Port B Output of Memory Core" or both. If the core is generated with port B registering on, set the INMEM_USER_REGISTER parameter in the "system.v" file to 1 or 2. CORE Generator indicates the "Latency added by output register(s):" on page 4 of the customization process. The value reported should be used for the INMEM_USER_REGISTER parameter. The default setting at the top of the "system.v" file is 1 (i.e. check the <u>"Register Port B Output of Memory Core" option</u>).
- m. Select "Finish" or "Generate" and CORE Generator will create the core's logic. As before, a new "blk_mem_gen_inputMem.xco" file will be generated. Add the .xco file to the ISE project and double-check that the blk_mem_gen_inputMem module has been recognized by ISE.
- n. Generate one last block memory with the settings:
 - i. Name: "blk_mem_gen_outputMem", Interface Type: Native and Memory Type: Simple Dual Port RAM
 - ii. Port A Options
 - Memory Size Write Width: <u>As per the user's requirements</u> (discussed below in section vi, but <u>if you use the default settings in the "system.v" file you should</u> <u>use 8</u>)
 - Memory Size Write Depth: <u>As per the user's requirements</u> (discussed below in section vi, but <u>if you use the default settings in the "system.v" file you should</u> <u>use 8192</u>).
 - 3. Enable: Always Enabled
 - iii. Port B Options
 - 1. Memory Size Read Width: 8
 - 2. Enable: Always Enabled
 - iv. Output Registers
 - 1. No registering on either port A or port B

- v. Simulation Model Options
 - 1. Disable simulation warnings by checking both the "Disable Collision Warnings" and "Disable Out of Range Warnings" boxes
- vi. Regarding user customization, the default parameters at the top of the "system.v" file define a 2^13 = 8192 = 8KB memory (OUTMEM USER ADDRESS WIDTH) with a userside interface of 1 byte wide (OUTMEM USER BYTE WIDTH). The depth for both ports and the port A write width (as powers of 2) can be changed to fit the user's needs. The OUTMEM_USER_ADDRESS_WIDTH and OUTMEM_USER_BYTE_WIDTH parameters can be modified in a similar manner as the input memory values. As with the input memory, any time that the user wishes to change the size or arrangement of the input memory the appropriate parameters in the "system.v" file should be updated and CORE Generator should be re-run.
- o. Select "Finish" or "Generate" and CORE Generator will create the core's logic. Again, a new "blk_mem_gen_outputMem.xco" file will be generated. Add the .xco file to the ISE project and double-check that the blk_mem_gen_outputMem module has been recognized by ISE.

At this point, the user should be ready to compile the example circuit. This can be done by selecting the "system" module in the "Sources" window and double-clicking "Generate Programming File" in the "Processes" window.



When the compilation is complete, it will create a new "system.bit" file in the root project directory. The functionality of this new bitfile will be tested in the following "Programming FPGA & Execution of Example" section.

One note regarding MAC addresses. The user should customize is the MAC address of the FPGA (MAC_ADDRESS in system.v) so as to avoid duplicate MAC addresses on the same subnet. Furthermore, the user should also avoid broadcast or multi-cast MAC addresses (unless that is what the user truly desires). Please note that broadcast or multi-cast MAC addresses may cause odd behavior in the software elements of SIRC. If this parameter is updated, the Ethernet Wrapper does not need to be regenerated through CORE Generator. However, the circuit does need to be recompiled in ISE.

6. Programming FPGA & Execution of Example

Programming FPGA

Once the user has a valid bitstream, either generated from scratch using the instructions in Section 5 or from the "precompiledExampleBinaries" directory (double-checking to ensure that they select the correct bitstream file), they can use it to program their FPGA. There are three preferred ways to program the FPGA:

- 1) Using iMPACT through the GUI (in "Start \rightarrow Programs \rightarrow XILINX_DIRECTORY \rightarrow Accessories \rightarrow iMPACT)
- 2) Using iMPACT through the SIRC *sendConfiguration* command
- 3) Programming at power-on from on-board flash memory

Methods #1 and #3 are important because it is generally assumed that the FPGA will already be connected to the host PC and programmed with a circuit that has the SIRC hardware API before the constructor for the SIRC software API is called. For example, the ETH_SIRC constructor calls the *sendReset* function just before it returns. This validates that the software can communicate with the hardware API within the constructor. If a circuit with the SIRC hardware API is not already programmed onto the FPGA, the constructor will return with a "fatal" error code (in this particular case, *FAILINITIALCONTACT* = -7). The user may elect to ignore this error code and call a subsequent *sendConfiguration* command, but we discourage this type of use.

The simplest way to insure that the FPGA is always capable of being used with the SIRC API is to bootstrap at power-on from on-board flash memory. While the exact technique to do this can differ slightly from development board to development board and from flash technology to flash technology, performing this on the ML505/ML507 or Digilent XUPV5 boards is quite straightforward. Documentation regarding how to do this is can be found in the "ML505/ML507 Getting Started Tutorial" available for download at the Xilinx website.

IMPACT	Declares intention to use the <i>sendConfiguration</i> command with iMPACT
PATHTOIMPACT	Path to iMPACT executable
PATHTOIMPACTTEMPLATEBATCHFILE	Path to a template iMPACT batch file. An example template file (impactMatchTemplate.cmd) is provided in the "precompiled- ExampleBinaries" directory. The only difference from the standard batch commands described in the iMPACT documentation (provided with the iMPACT GUI program "Help→Help Topics→Software Help→ iMPACT Help→ Command Line and Batch Mode→Batch Mode") is the inclusion of the "BITSTREAMFILENAME" keyword. This keyword replaces the bitstream filename that would normally be used with the assignFile command.
PATHTOIMPACTPROGRAMMINGBATCHFILE	Path to which software API can write a temporary file. This will be the command batch file passed to iMPACT during execution of the <i>sendConfiguration</i> function. If iMPACT cannot successfully program the device, try to run this file from the command line with "iMPACT-batch filename.cmd"
PATHTOIMPACTPROGRAMMINGOUTPUTFILE	Path to which software API can redirect iMPACT output during programming. This file will be parsed by the software API to determine success or failure of programming attempt. This file can also be examined by the user if iMPACT is unable to successfully program the FPGA.
IMPACTSUCCESSPHRASE	When iMPACT successfully programs the FPGA, what is the reply? If unsure, the user can examine the file that is produced at

The *sendConfiguration* command requires a few compile-time constants to be defined before it can be used. As shown in eth_SIRC.h, six constants are needed:

PATHTOIMPACTPROGRAMMINGOUTPUTFILE.

Executing Example

The user can either compile the example software or use the pre-compiled executable in the "precompiledExampleBinaries" directory. The software binary should be run from a command line. For the default settings of the hardware example (also compiled into the provided FPGA bitstreams), no arguments are required. Any errors that are encountered will be reported back to the user and can be looked up in the eth_SIRC.h file. View the example software source code for more details.

The XUPV5/ML505/ML506/ML507 boards have a few configuration jumpers that need to be set to ensure correct operation. First, the default UCF file indicates some 2.5V I/O. J20 (on the top right of the board near the power switch) should have have both jumpers between pins 2 & 3 (to the right side) to select 2.5V I/O on two of the FPGA's pin banks (as set in the provided "system.ucf" file) Second, the current implementation uses a GMII interface to the PHY. J22 & J23 (bottom left near the DVI output) should have both jumpers between pins 1 & 2 (to the left side). Third, if you are trying to configure the FPGA with a Platform Cable JTAG programmer, you don't have to worry about the DIP switch settings of SW3 (upper left near the keyboard port). However, if you are trying to bring the configuration from the Compact Flash or other on-board Flash memory, pay attention to the settings of SW3. The "ML505/ML506/ML507 Evaluation Platform" document from the Xilinx website can help you figure out what is appropriate for your situation. Look at item #31 - "Configuration Address and Mode DIP Switches". Personally, I like to bootstrap the board from configuration #0 on the CF. This requires a setting of "00010101" reading from DIP 1 to DIP 8 left to right. See the "System ACE CompactFlash Solution" document from the Xilinx website for more information.

The BEE3 does not need any configuration jumpers to be set.

There are three jumpers that need to be set on the ML605. J66 and J67 should set such that the jumper is over pins 1 &2. J68 should be open. This is the default setting for these jumpers from the factory.

7. Simulating & Debugging with SIRC ⇔ Modelsim PLI

The hardware side of SIRC (along with the user application) can also be run completely in simulation with Modelsim. The software client does not require any changes and minimal changes are made to the Verilog. This makes the simulation as faithful as possible to the real hardware client. Simulation of SIRC has been tested with Modelsim SE and PE, version 10.0. Even the student version work, albeit slowly, since it does not optimize the execution of the simulation.

Loopback Setup

For performance reasons it might be a good idea to connect both the user software and the Modelsim simulation to a loopback network port. You can do this if you are:

- simulating the SIRC hardware & your application on the same machine as the host software and your software (this isn't necessarily the case, but likely unless you really want to use two completely separate machines – one for the software side and one for the hardware side of things)
- 2) not already using a dedicated network connection between the FPGA and the PC

While this is not strictly necessary, this will considerably to reduce the Ethernet traffic that must be processed by Modelsim. <u>Please note that due to a restriction in the Virtual PC driver code, the first time that the software client is run on a given machine and with a given Virtual PC driver, it must connect to a real network adapter (e.g. a wireless or wired NIC and not the loopback). Subsequent connections on the machine can attach to any network adapter, but it must connect to a physical NIC at least once. Thus, for example, it might be a good idea to first test</u>

with the example precompiled bitstreams on a real board before electing to do simulation. At the very least, you must run the example software client at least once trying to use a physical NIC (even if you don't have a board to test with, the software client will attempt to connect – this is sufficient and you only have to do it once per machine).

To add a loopback network port:

- I) Right-click on "Computer" (on the desktop or in the Start Menu) to get the "Manage" command.
- II) Then go to "Device Manager" and right-click on your machine. Select the "Add Legacy Hardware" option.
- III) Select the "Install the hardware that I manually select.." option.
- IV) Select the "Network adapters" category.
- V) Select "Microsoft -> Microsoft Loopback Adapter"
- VI) The loopback adapter will now appear in "Control Panel -> Network and Internet -> Network and Sharing Center => Change Adapter Settings". As described in Section 2, uncheck all other services on the loopback adapter besides the Virtual PC network driver. At the same time, for simplicity sake it may be best to temporarily uncheck the Virtual PC network service from all other network adapters.

Modelsim Setup

SIRC uses Xilinx IP such as the BlockRAM. Thus, to simulate we must export the library models into Modelsim. If you have not already done this, you can export the library in the following way:

- I) Start the Xilinx Simulation Library Compilation Wizard from
 "\$XILINXINSTALLROOT\XXX\ISE\bin\ntXXX\compxlibgui.exe", replacing the various terms as appropriate. On my 64-bit Windows 7 machine with 64-bit Modelsim installed, I run the executable here: "C:\Xilinx\13.2\ISE_DS\ISE\bin\nt64\compxlibgui.exe"
- II) The wizard will most likely find Modelsim installed on your machine and autofill the correct options and path. If not, fill these in appropriately.

🔞 Xilinx Simulation Library Compilation Wizard - Select Simulator
Select Simulator
O ModelSim PE
ModelSim SE
ModelSim DE
⊘ Active-HDL
Simulator Executable Location (The -p command-line option)
C:\modeltech64_10.0a\win64 Browse
Compxlib Configuration File (The -drg command-line option) compxlib .cfg Compxlib Log File (The -log command-line option)
compxib log Browse
Do not use this wizard For ISm or Modelsim Xilrx. Edition as they come with pre-compiled simulation libraries. Only specific versions of the simulators
are supported. Hease veny that the selected simulator version satisfies the following requirements. Modelsim/Questa/Sim 6.6d and later
Riviera 2010.10 or later
Active-HDL 2010.10 or later
More Info Kext > Cancel Cancel

III) Export either the Verilog models or both the Verilog and VHDL models.

A Xilinx Simulation Library Compilation Wizard - Indicate the HDLs supported by you	ır simulator	<u> </u>
Select HDL(s) used for simulation Both VHDL and Verlog VHDL Yenlog		
Please ensure that simulator is licensed for selected HDL(s).		
After complation, the following types of simulations: can be performed - Behavioral Smulation in Verlog - Structural Smulation in Verlog Timing Simulation in Verlog More Info	C Back Next 5	Cancel

IV) Export the models for the appropriate devices (Virtex 5 and/or Virtex 6)

🔁 Xilir	x Simulation Library Compilation Wizard - Select Device Families	
Sala	rt Devine Familee	
V	All EPGA Device Families	
	All <u>C</u> PLD Device Families	
	Adia 7	
H	Automative 9500YI	Â
H	Automotive CoolRunner?	
	Automotive Spartan3	
	Automotive Spartan3A	
	Automotive Spartan-3A DSP	
	Automotive Spartan3E	
✓	Automotive Spartan6	
	CoolRunner XPLA3 CPLDs	
	CoolRunner2 CPLDs	
	Defense-Grade Spartan-6Q	
	Defense-Grade Spartan-6Q Lower Power	
	Defense-Grade Virtex-4Q	=
	Defense-Grade Virtex-5Q	
	Defense-Grade Virtex-6Q	
	Defense-Grade Virtex-6Q Lower Power	
	Kintex7	
	Kintex7 Low Voltage	
	Space-Grade Virtex-4QV	
	Spartan3	
	Spartan 3A and Spartan 3AN	
	Spartan-3A DSP	
	Spartan3E	
	Spartano Lauras Paruas	
	Spanario Lower Power	
	Vitex5	
	Vitex6	
	Manuf Lauran Dauran	T
(
Mor	< Back Next >	Cancel

V) Select the libraries that need to be exported. For safety sake, I export all available libraries

	Xilinx Simulation Library Compilation Wizard - Select Simulation Libraries
ſ	Select libraries for Functional and Timing Simulation
	☑ All libraries
	CPLD designs (CPLD, UNI9000)
	CORE Generator (XilinxCoreLib)
	☑ Timing Simulation library for FPGA and CPLD designs (SIMPRIM)
	EDK Simulation Library
	Specify more library source path (The -source_lib command-line option):
	Specify more Compatib command-line option:
ſ	More Info

VI) Note the path of where the wizard will be placing the exported libraries. On my 64-bit Windows 7 machine with Modelsim SE installed, this is "C:\Xilinx\13.2\ISE_DS\ISE\verilog\mti_se\10.0a\nt64". You may need this path if Modelsim does not automatically import the libraries.

Xilinx Simulation Library Compilation Wizard - Output directory for compiled libraries	×
Output directory for compiled libraries	
C:\Vilinx\13.2\\SE_DS\\SE\danguage>\ <simulator>\<version>\<platform></platform></version></simulator>	Browse
	Def <u>a</u> ult
Map only to existing pre-compiled libraries (i.e. no -w overwrite command-line option)	
Exclude superseded (-exclude_superseded option for EDK only)	
Exclude sublib (-exclude_sublib option for EDK only)	
Verb <u>os</u> e (-verbose)	
Advanced options	
You need to remove those entries in angle brackets if you are changing the location to something other than the default.	
Choosing Map only option is same as not including -w argument in compxlib	
More Info	Cancel

Compiling PLI ⇔Ethernet DLL

The Modelsim simulation also sends and receives information using the Virtual PC network driver. This is accomplished using a DLL that bridges that gap. We provide code to accomplish this, but it uses library functions that are provided by Mentor. Thus, similar to the situation with Xilinx code, we cannot redistribute this library code. Thus, you must re-compile it yourself.

- Copy four files from your Modelsim directory into either the SIRC_INSTALL_PATH\PLI_Plugins\src32 or SIRC_INSTALL_PATH\PLI_Plugins\src64 folder (depending upon if you have a 32 or 64-bit version of Modelsim installed).
 - a. vpi_user.h, vpi_compatibility.h, veriuser.h (on my machine with a 64-bit version of Modelsim SE 10.0 installed, these files were in C:\modeltech64_10.0a\include)
 - b. mtipli.lib (on the same machine, this was in C:\modeltech64_10.0a\win64)
- II) Open the vpi2ether project in SIRC_INSTALL_PATH\PLI_Plugins\ether by double-clicking on the "vpi2ether.sln" file. Select the proper output target for your machine & version of Modelsim and build the solution with the "Build->Build Solution" command. Unless you want to debug the DLL itself, select a Release build.



III) Copy the resulting "vpi2ether.dll" file from the proper output directory (e.g. SIRC_INSTALL_PATH\ PLI_Plugins\ether\Release or SIRC_INSTALL_PATH\PLI_Plugins\ether\x64\Release) to the root directory of your ISE project. For example, if you are building from the example in the HW_Example_13_2_ML605 directory, put the .dll directly in that folder.

ISE Project Setup

The only real change to the hardware code that is necessary to simulate SRIC occurs in the top-level module (system.v). Our C++ \Leftrightarrow PLI software replaces the physical PHY and MAC. However, a few minor changes are needed to set the system up for simulation.

- I) Either when adding the files to the project initially or from the "Files" tab, change the association of:
 - a. the MAC library files (eth_fifo_8.v, rx_client_fifo_8.v, tx_client_fifo_8.v, gmii_if.v, emac_single.v, emac_block.v, emac_locallink.v)
 - b. and two SIRC files (system.v and ethernet2BlockMem.v)

to "Implementation".

> I	SE P	roject Navigator (0.61xd) - C:\eg	uro\SIF	RC\CurrentVersion\H	HW_Example_13	8_2_M	/IL605\HW_Example_13_2_ML605.xise - [Design S	ummary]
Σ	<u>F</u> ile	<u>E</u> dit <u>V</u> iew P <u>r</u> oject <u>S</u> ource	<u>P</u> ro	cess <u>T</u> ools <u>W</u> in	dow La <u>v</u> out	Hel	łp	
			< 0	(a) » 🏓	- B B -	2	🔀 🖻 🗉 🗣 🎤 😢 🖈	P 💡
Files					⇔⊡₽×	a	🖻 Design Overview 🔺	
F ¥	Na	me 🔺	Type	View Association	Source Library	W	- Summary	Project File
a	10	blk mem gen inputMem.xco	IP (ΔII	work	\bigcirc	Module Level Utilization	Trojecerne.
0	1 👌	blk mem gen inputMem.xise	IP (All		\bigcirc		Module Nam
d.	1	blk mem gen outputMem.xco	IP (All	work	-	- Pinout Report	Target Devi
	1	blk_mem_gen_outputMem.xise	IP (All		ç	- Clock Report	Product Ver
•	14	blk_mem_gen_paramReg.xco	IP (All	work	HTHE	- 👸 Static Timing	Design Conk
	1	blk_mem_gen_paramReg.xise	IP (All			Errors and Warnings	Design Goal
	V	emac_single.v	Veri	Implementation	work	30	Parser Messages	Design Strat
	I M	emac_single_block.v	Veri	Implementation	work	1	— 🗋 Synthesis Messages	Environmen
	I M	emac_single_locallink.v	Veri	Implementation	work		- 📄 Translation Messages	é
	<u> </u>	ethernet2BlockMem.v	Veri	Implementation	work		— 🛄 Map Messages	
	<u> </u>	ethernetController.v	Veri	All	work		 Place and Route Messages 	
	<u> </u>	eth_fifo_8.v	Veri	Implementation	work		— Timing Messages	
	<u> </u>	fifo36Wrapper.v	Veri	All	work		— 📑 Bitgen Messages	Report Nam
	<u> </u>	gmii_if.v	Veri	Implementation	work		 All Implementation Messages 	Synthesis Rep
		iobuf.v	Veri	All	work		Detailed Reports	Translation Re
	1	ML605.uct	Im	Implementation	work		- 🛄 Synthesis Report	(
	IЦ	rx_client_fifo_8.v	Veri	Implementation	work		- 📋 Translation Report	Map Report
	Ľ	simple lestModuleOne.v	Veri	All	work		Map Report -	Place and Rou
	H	system.v	Veri	Implementation	work		Design Properties	Power Report
	I K	systemper.v	Veri	Simulation	work		Enable Message Filtering	
	IШ	D_CHERC_HID_0.V	veri	implementation	WOIK		Optional Design Summary Contents	Post-PAR Stat
							- Show Clock Report	Bitgen Report
							- Show Failing Constraints	
							- Show Warnings	

- II) As also seen in the picture above, add the "systemPLI.v" file (found in <u>HWSrc\PLI</u> directory) to the project with a "Simulation" association. The association of all other files should remain "All".
- III) Select "Edit -> Preferences" and make sure that ISE knows the appropriate path to the Modelsim executable

Preferences - Integrated Toc	ols Options	×
<u>C</u> ategory	Set the paths for the integrated tools you have installed.	
Console	Model Tech Simulator:	
ISE General Design Goals & St	C:\modeltech64_10.0a\win64\modelsim.exe	Default
Editors	Synplify:	
Process Completi		Default
ISE Text Editor	Symplify Pro:	
RTL/Technology View Color Scheme	Precision	Default
New Object Color ≡	riguain.	Default
User Color Rules	PlanAhead:	
Check	C:\Vilinx\13.2\ISE_DS\PlanAhead\bin	Default
Colors Device Families		
Layout		
Sheet Sizes		
Check		
Timing Analyzer		
	OK Cancel A	pply Help

IV) Switch the Design view to "Simulation", select the "system" module, and right-click on "Simulate Behavior Model". Select "Process Properties".

> 1	SE Project Navigator (0.61xd) - C:\eguro\SIRC\CurrentVersion\HW_Example_1:	3_2_M	L605\HW_Example_13_2_ML605.xise - [Design	Sur	nmary]
Σ	File Edit View Project Source Process Tools Window Layout	Hel	p		
			🔊 🗟 🗆 🖻 🖻 🥓 🕅 🔊	*	v
Desi	n ⇔⊡∄×	0	Design Overview	^	
T#	View: 🔘 🎒 Implementation 💿 🙀 Simulation	-	Summary		Project File:
J.	Behavioral 💌	Θ	- Module Level Utilization		Module Name:
Rin I	Hierarchy	Θ	- 🗋 Timing Constraints		Target Device:
-	- IW_Example_13_2_ML605	Ç	Pinout Report Clack Panat		Product Vorcion
00	xc6vlx240t-1ff1156	HTP-5	Static Timing		Product version.
6	system (system PLLV) ec - ethernetController (ethernetController.v)	38	Errors and Warnings	=	Design Goal:
623	🗄 👿 tm - simpleTestModuleOne (simpleTestModuleOne.v)	20	Parser Messages		Design Strategy:
		(A)	Translation Messages		Environment:
-			🗋 Map Messages		
			Place and Route Messages		
			I Iming Messages		Report Name
_			All Implementation Messages		Synthesis Report
	No Processes Running		Detailed Reports		Translation Report
9	Processes: system		Translation Report		Map Report
91	🖻 🌠 ModelSim Simulator		🗋 Map Report	-	Place and Route Report
ETH-	Simulate Behavioral Model		Design Properties	٦	Power Report
~ 1,	Tių Kun Parus All		Enable Message Filtering		Post-PAR Static Timing Rep
	Refun All		Optional Design Summary Contents		Bitgen Report
	Star Mith Current Date		Show Failing Constraints		
	Run with Current Data		- Show Warnings		
	Process Properties		Show Errors		
					Report Name
>	Start Cesign C Files C Libraries	Σ	Design Summary		

- V) Change the "Property Display Level" to "Advanced" and make the following changes
 - a. Other VSIM Command Line Options: "-pli vpi2ether.dll"
 - b. Simulation Run Time: Ons

ategory	Switch Name	Property Name	Value
Simulation Properties	-dir	Compiled Library Directory	<pre>KILINX\<language>\<simulator></simulator></language></pre>
Display Properties		Ignore Pre-Compiled Library Warning Check	
		Generate Verbose Library Compilation Messages	
		Use Custom Do File	
		Custom Do File	
		Use Automatic Do File	
	1	Custom Compile File List	
		Other VSIM Command Line Options	-pli vpi2ether.dll
		Other VLOG Command Line Options	
		Other VCOM Command Line Options	
		Simulation Run Time	Ons
		Simulation Resolution	Default (1 ps)
		VHDL Syntax	93
	1	Use Explicit Declarations Only	
		Use Configuration Name	
		Configuration Name	Default
		Log All Signals In Simulation	
		Specify Top Level Instance Names	
		Property display level: Advanced 💌 🛽	Display switch names

VI) Select OK and go back to the main ISE window.

Execution of the Simulation

- After selecting the top-level "system" module in the "Design->Simulation" window in ISE, double-click "Simulate Behavior Model" in the "Design -> Processes" window. This will launch Modelsim. It should compile the necessary files and start the simulation (but not actually begin executing).
- II) If you built a Debug version of the PLI DLL above (as opposed to a Release version), the system will raise an exception when the simulator loads the DLL and you will have the chance to attach a debugger. Just select the "Debug" option and select an appropriate Visual Studio project.



III) You can now add signals that are of interest to the waveform viewer. When this is complete, begin the simulation with the command "run –all" in the "Transcript" console.



IV) You can now start the software example provided, just as you might when targeting a physical board. However, due to the much slower performance of the software simulation (e.g. rather than 400+Mbps bandwidth to the real hardware, simulation gives ~ 400Kbps), it is probably necessary to adjust the software example to perform smaller/fewer tests and with a greater timeout. I suggest running the example with the following arguments initially:

eth_sirc_lib_SW_Example.exe -waitTimeOut 30000 -bandwidthIter 20