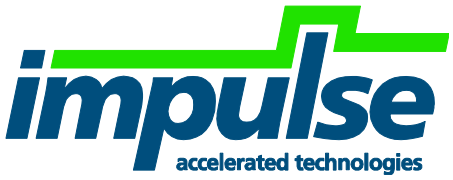


CoDeveloper Example and Tutorial

Solarflare AOE Enet MAC Counter Tutorial Version 1.0.3

Impulse Accelerated Technologies, Inc.

www.ImpulseAccelerated.com

1.0 Table of Contents

2.0 Overview

This tutorial covers the “Enet MAC Counter” example written in Impulse C and run on the FPGA within the Solarflare AOE low-latency programmable 10G NIC. This tutorial covers all the necessary steps for the supplied example to:

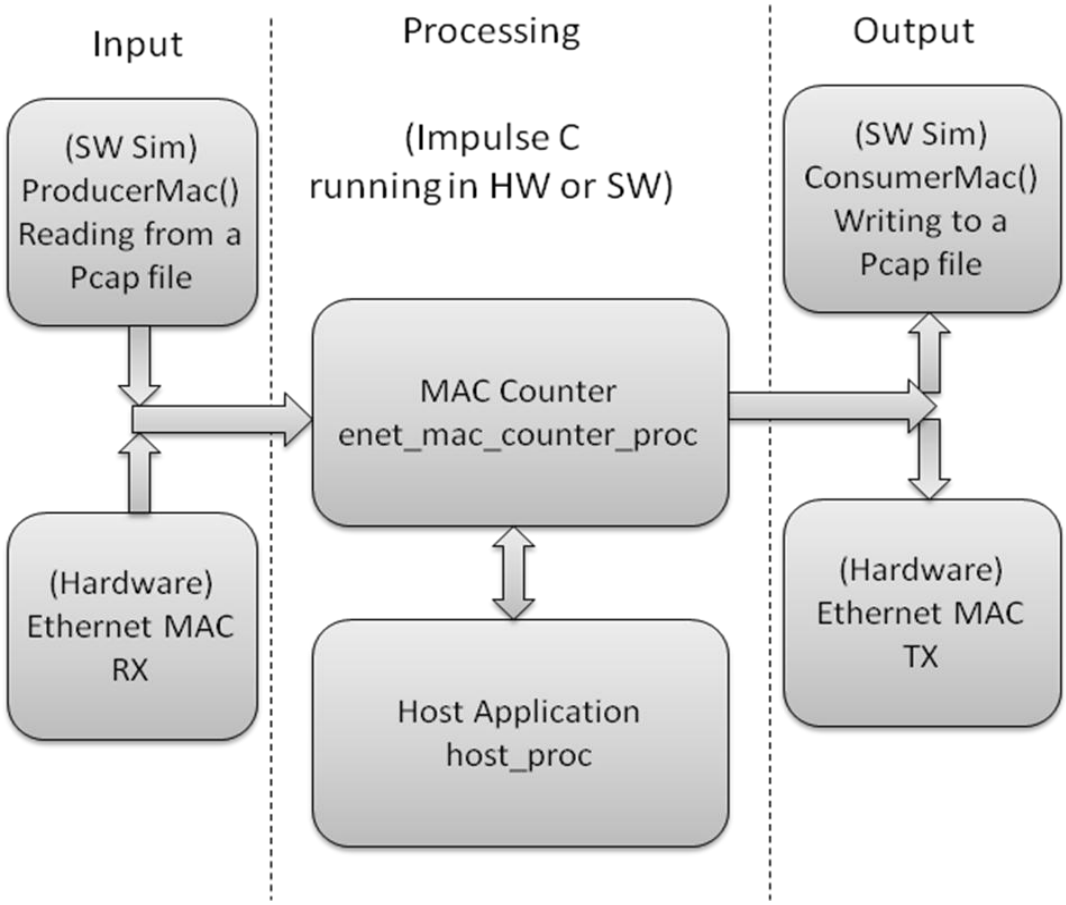
- Run in CoDeveloper:
  - Desktop software simulation
  - Generating and exporting the hardware module
- Integrate within the provided AOE framework:
  - Adding the hardware module to Qsys and connecting to MAC ports
- Run the example
  - Setup the test environment
  - Load and run the FPGA binary on the AOE
  - Run a test application on a host to verify the hardware module running in the AOE

Notes:

- This tutorial assumes that the user has already successfully setup and run the pass-through described in the document “Impulse_on_Solarflare_AOE_Getting_Started_Users_Guide.pdf”. The pass-through project and files serve as the base for adding the user’s application to the AOE.
- This tutorial shows the steps for running the example through Impulse C and CoDeveloper, however is not intended to completely teach Impulse C nor CoDeveloper. It is highly recommended that the user go through the “Hello World” tutorial installed with CoDeveloper and available from the “Impulse C User Guide”.

2.1. Enet MAC Counter Example Overview

The Enet MAC Counter simplified example of packet-in-packet-out written in Impulse C that is intended to help the user get started by showing the basic steps of handling raw Ethernet data using a single streaming processes. A block diagram of the design appears below:



Brief description of each block:

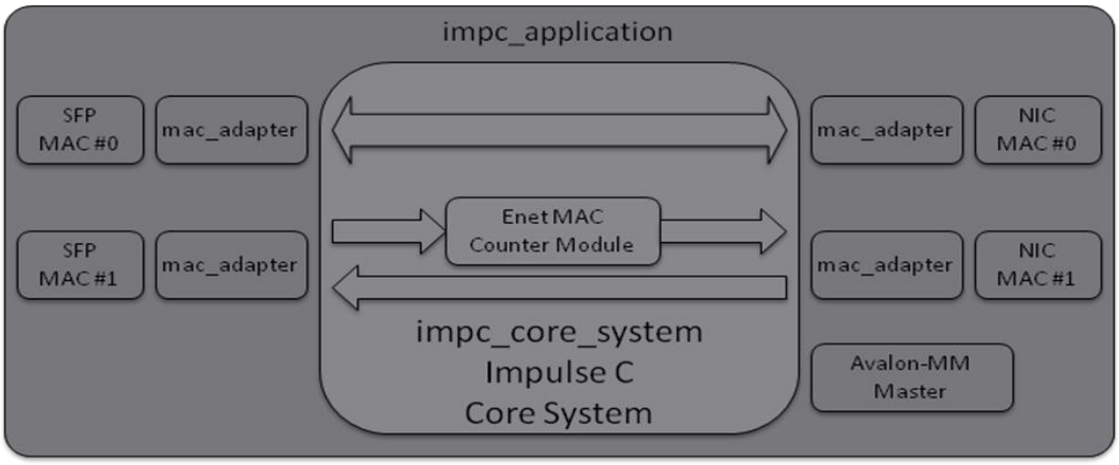
- Input:
  - In software simulation input is from the Impulse C ProducerMac() process that behaves as the incoming MAC passing Ethernet packets reading from a standard Pcap file.
  - In hardware the input is connected directly to the Ethernet MAC to receive packets from the network
- Processing:
  - MAC Counter: A process that receives data from the Ethernet MAC. Performs packet translation from external format (here Avalon-ST) to parse words to look for the specific MAC configured by the host. Data is treated pass-through and sent out as-is (in this case as Avalon-ST again).
  - Host Application: Application on host that configures the specific MAC to count and then periodically read the MAC counter.
- Output:
  - In software simulation output is to the Impulse C ConsumerMac() process that behaves as the outgoing MAC receiving Ethernet packets and writing

out to a standard Pcap file which may be readily viewed in a tool like Wireshark.

- In hardware the output is connected directly to the Ethernet MAC to transmit packets to the network

2.2. Enet MAC Counter Example Implementation on AOE

In this example, the Enet MAC Counter module will be implemented on the AOE by being inserted into the original pass-through base design between the incoming Ethernet stream of SFP #1 and the outgoing Ethernet stream to NIC MAC #1 as shown below. This will allow the Enet MAC Counter module to see all data coming in the AOE NIC on port#1. Note that this is only one of many possible configurations and chosen for convenience.



3.0 Setting up for the Enet MAC Counter Example

3.1. Additional Required Files

It is assumed that the user has successfully run through the pass-through base project and the following directory structure is already present:

fdk_release\sf_impc_base_project

The user should have received a link download and password to open the zipped file "<release date>_enet_mac_counter_Example.zip". If not, please contact support@impulsec.com to request access. The file is to be unzipped to a convenient location for development. When unzipped the "enet_mac_counter" directory will be present.

Notes:

- Be sure to use a directory path that does not contain spaces (' ') to avoid issues
- Note that these instructions are for running CoDeveloper on Windows and Quartus on Linux whereas Quartus could be run on Windows or Linux. If running Quartus on Windows, placing the example directory near the "sf_impc_base_project" Quartus directory is recommended so that the export directory setting in CoDeveloper may be set to avoid copying the exported HDL shown in a later step.

3.2. Network Setup

The example requires a network path that goes through the Enet MAC Counter module within the AOE FPGA. The setup described here makes use of an external loopback connection to do this.

To setup for the Enet MAC Counter example:

1) Connect the AOE's top (SFP #1) port using a 10G Ethernet cable to the bottom port (SAFP #0)
2) Double check using tcpdump or Wireshark that broadcast packets are received between the connections by monitoring SFP #1. For example using ping will generate ARP requests that are broadcast, assuming eth5=SFP #0 and configured as IP 192.168.5.62, then "ping 192.168.5.1" will cause ARP messages to be sent out SFP #0 which will be seen, though not responded to, by SFP #1.

4.0 Enet MAC Counter Example

4.1. Prerequisites

This tutorial for this example 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 generation.

This tutorial also assumes that the user has already successfully setup and run the pass-through described in the document "Impulse_on_Solarflare_AOE_Getting_Started_Users_Guide.pdf". The pass-through project and files serve as the base for adding the example application to the AOE as shown in the steps to follow.

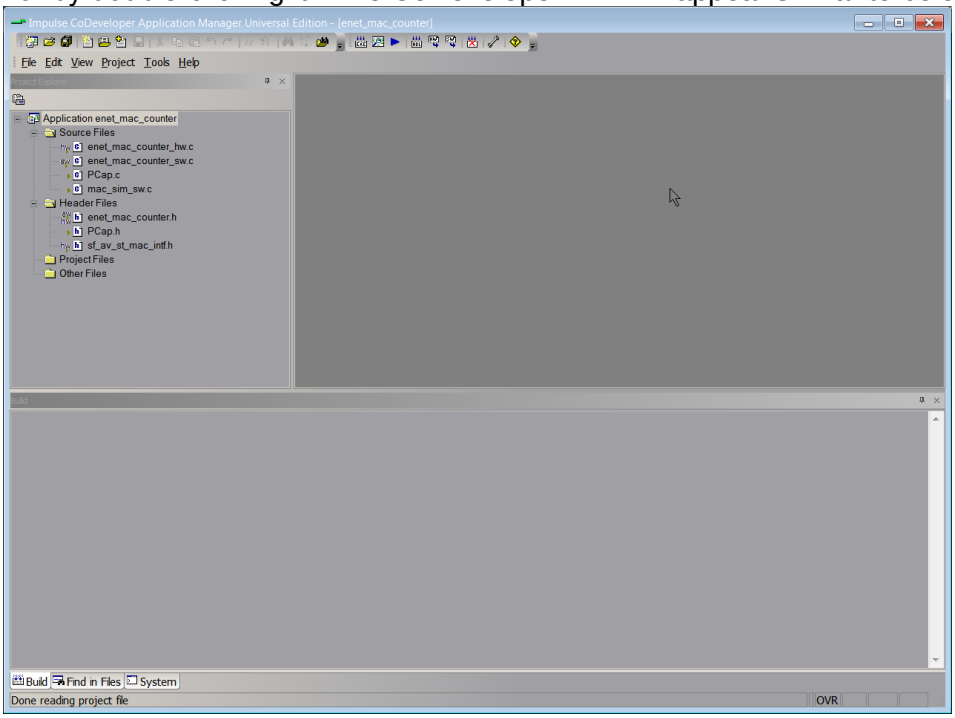
4.2. CoDeveloper Project Files

The Enet MAC Counter example CoDeveloper project is made up of the following files:

- enet_mac_counter.icProj – CoDeveloper project file
- enet_mac_counter_hw.c – Source code for application hardware process
- mac_sim_sw.c – Source code for MAC input and output simulation processes
- enet_mac_counter_sw.c – Source code for application software processes
- enet_mac_counter.h – Header file for application
- sf_av_st_mac_intf.h – Header file for the Avalon-ST interface to MACs
- PCap.c, PCap.h – Support file for reading and writing Pcap files
- multicastUdpFromOneIpAlignedMixed2.pcap – Input test file

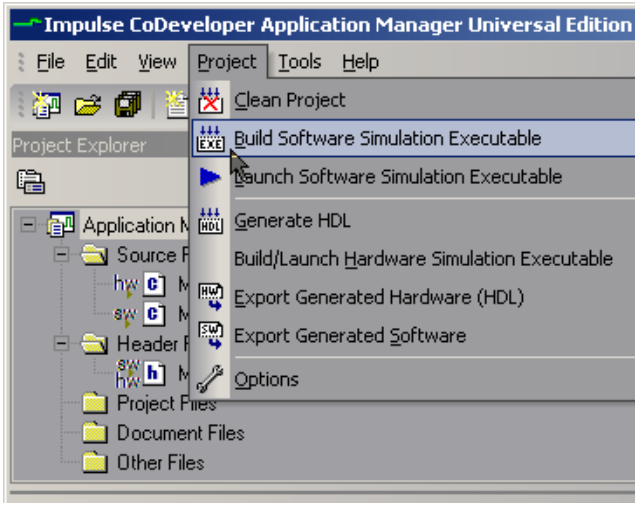
4.3. Opening Project

Open the CoDeveloper project file ‘enet_mac_counter.icProj’ by selecting and pressing ‘Enter’ or by double-clicking it. The CoDeveloper IDE will appear similar to below:

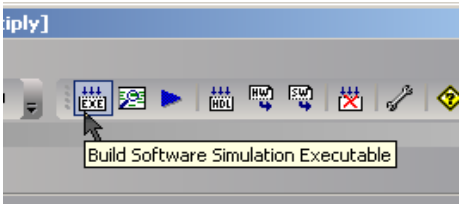


4.4. Building Desktop Simulation Executable

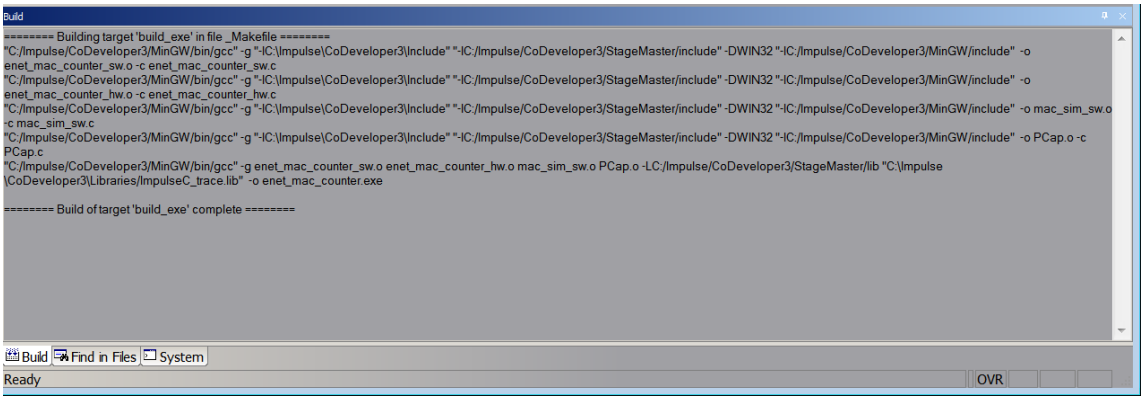
Build the desktop software simulation executable via the “Project” menu:



Or via toolbar:

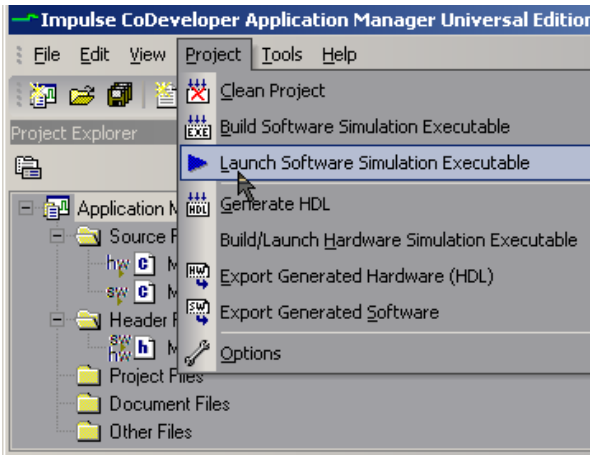


Note the compiler output in the CoDeveloper IDE “Build” window will appear similar to below:

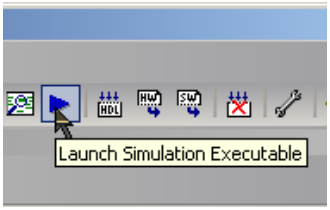


4.5. Running Desktop Simulation Executable

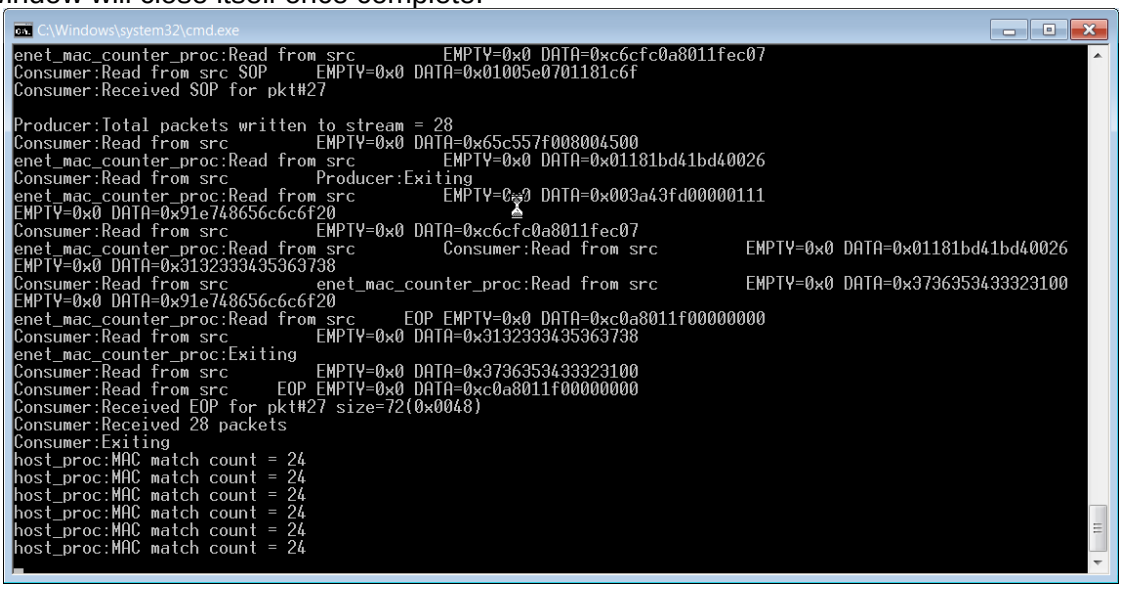
Launch the desktop software simulation executable via “Project” menu:



Or via toolbar:



A command window will pop up in which the desktop simulation executable runs. The MAC counter is being checked ten times once per second appearing as below. The window will close itself once complete.

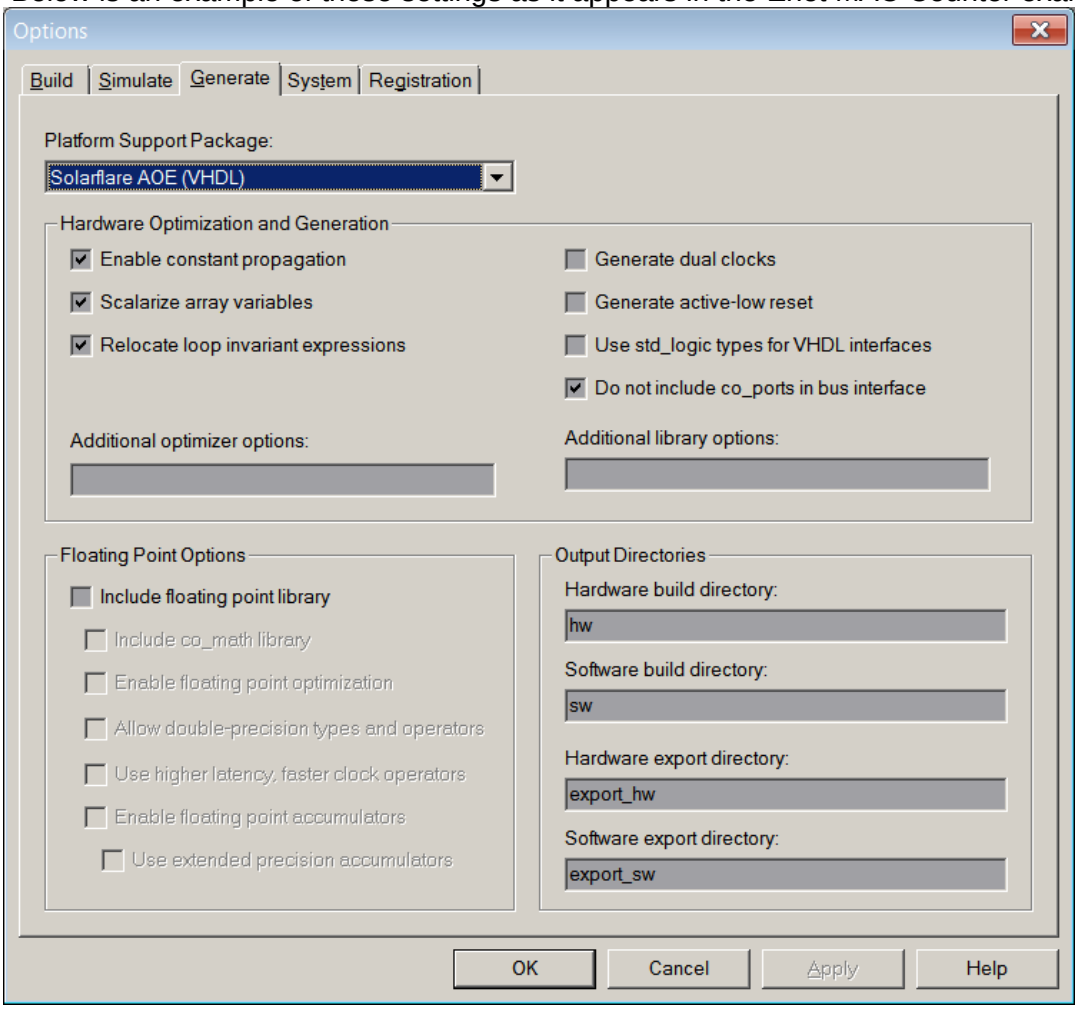


4.6. Project Setup Before Hardware Generation and Export

Settings within the CoDeveloper IDE necessary for generating and exporting hardware for the Solarflare AOE are set via the “Generate” tab under Project->Options and are summarized below:

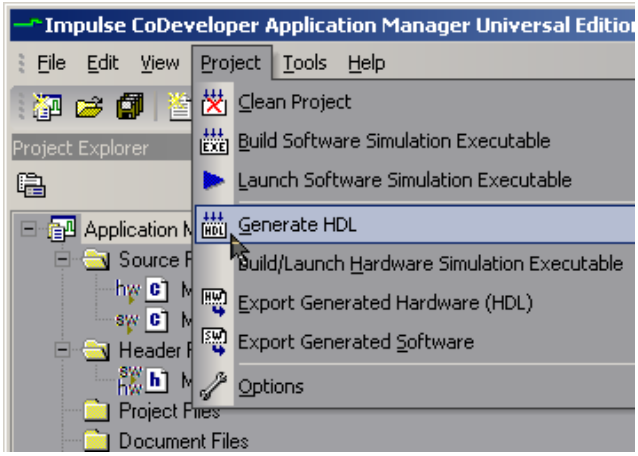
- Platform Support Package: “Solarflare AOE (VHDL)”
- Hardware export directory: export_hw
- Software export directory: export_sw
- Unsupported settings include:
    - Generate dual clocks (must be unchecked)
    - Active-low reset (must be unchecked)

Below is an example of these settings as it appears in the Enet MAC Counter example:

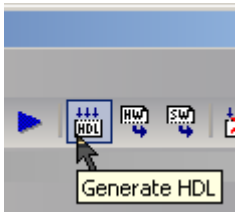


4.7. Generating Hardware

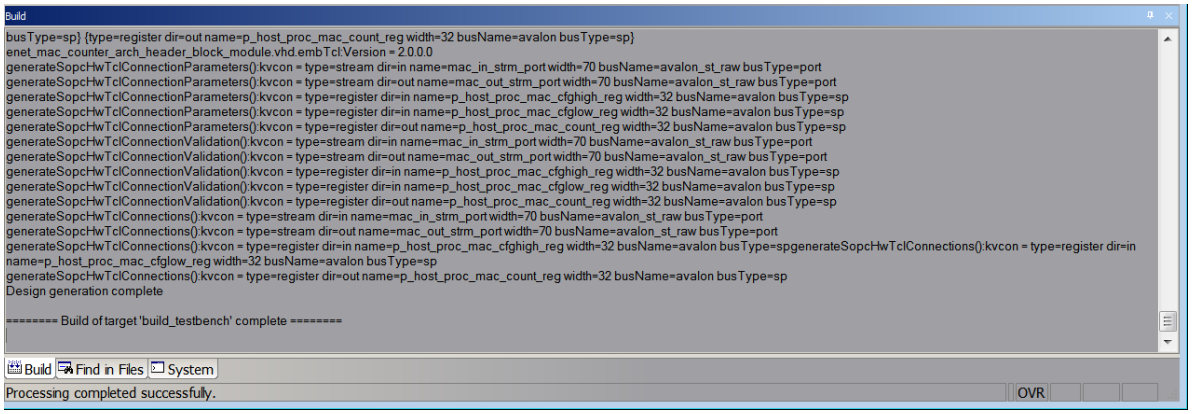
Generate hardware via “Project” menu:



Or via toolbar:

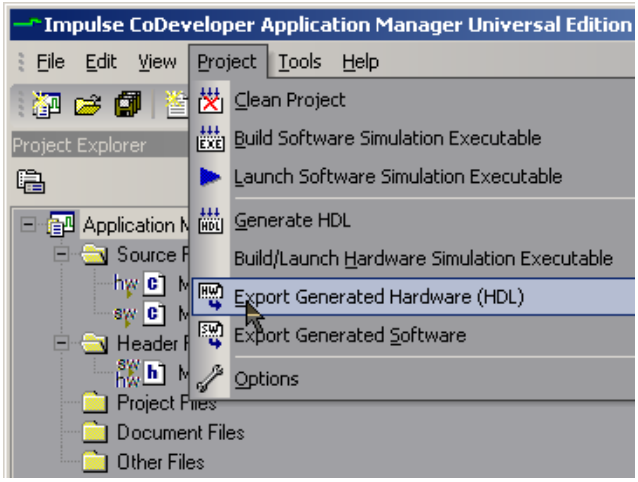


The generated HDL will appear in the directory specified during project setup in “hw” directory. Note the output in the CoDeveloper IDE’s “Build” window will appear similar to below:

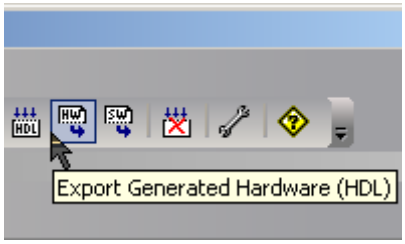


4.8. Exporting Hardware

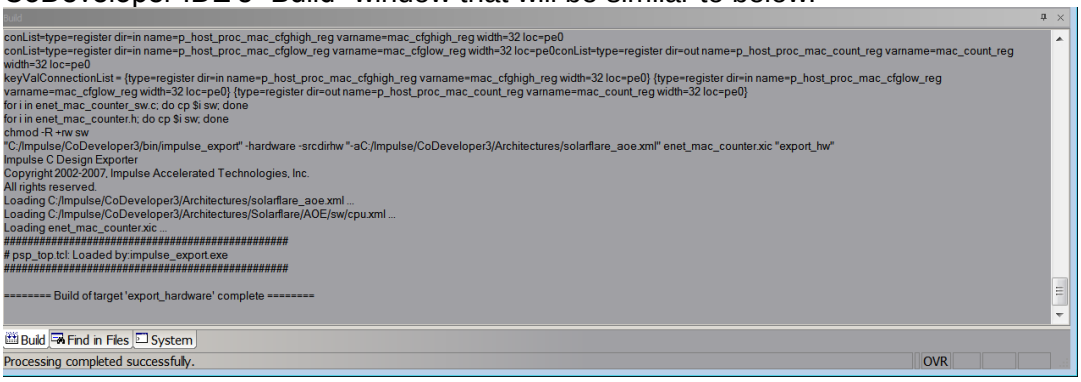
Export hardware via “Project” menu:



Or via toolbar:

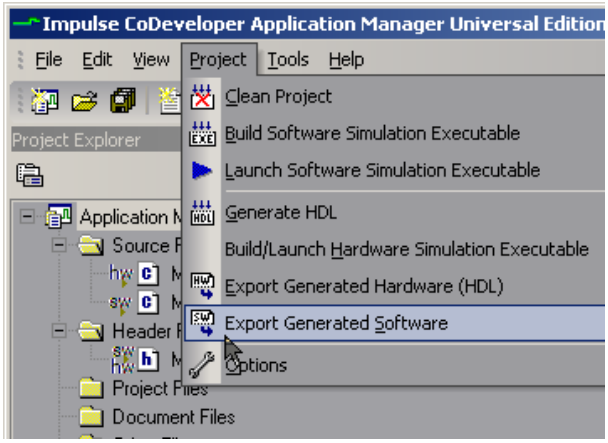


The complete exported HDL and Qsys module will appear in the “export_hw/ip/enet_mac_counter_arch_module” directory. Note the output in the CoDeveloper IDE’s “Build” window that will be similar to below:

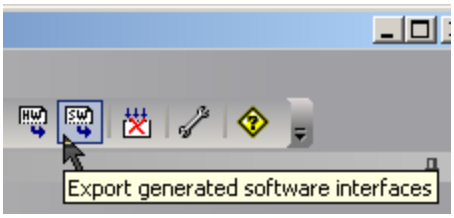


4.9. Exporting Software

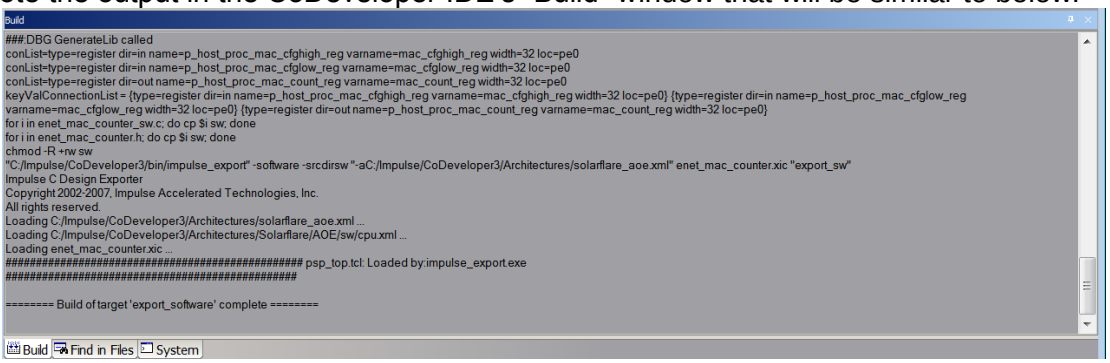
Export software via “Project” menu:



Or via toolbar:



The complete exported software and library will appear in the “export_sw” directory. Note the output in the CoDeveloper IDE’s “Build” window that will be similar to below:



The “export_sw” directory needs to be copied to the Linux host containing the AOE. Typically this would be the application directory somewhere under the user’s home directory, from here it will be simply referred to as “<application dir>”.

4.10. Adding the Module to the impc_core_system using Qsys

After exporting hardware, the “export_hw/ip” directory containing the Enet MAC Counter module must be copied to the Quartus directory. Note that the “sf_impc_base_project /ip” directory already exists, while copying be sure to allow files to be overwritten. The module may now be added to the impc_core_system of the original pass-through for the AOR using Qsys. The following steps walk through the process.

4.10.1.1. Open the Quartus II Project

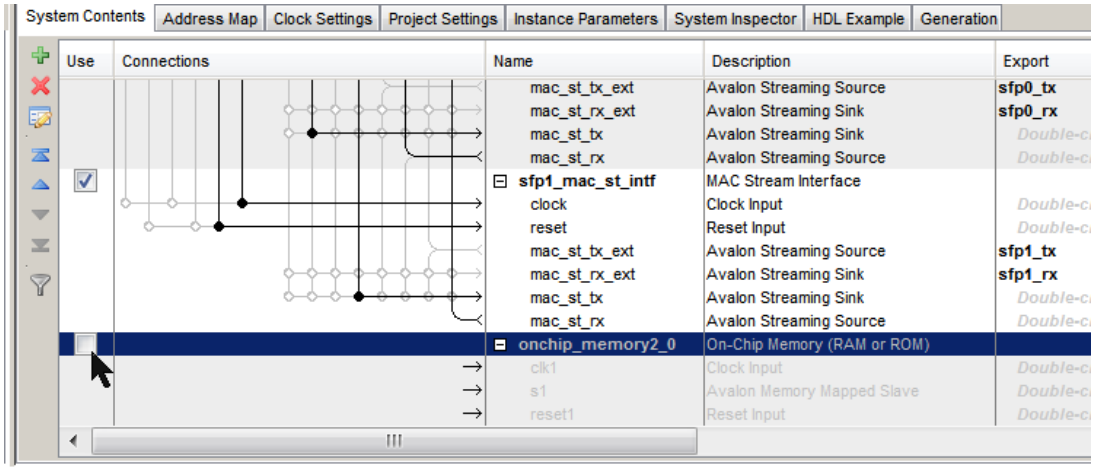
Start Quartus II 64-bit normally for the operating system being used and open the Quartus project file ‘sf_impc_base_project\sf_impc_base_project.qpf’.

4.10.1.2. Add the Module to the Core System

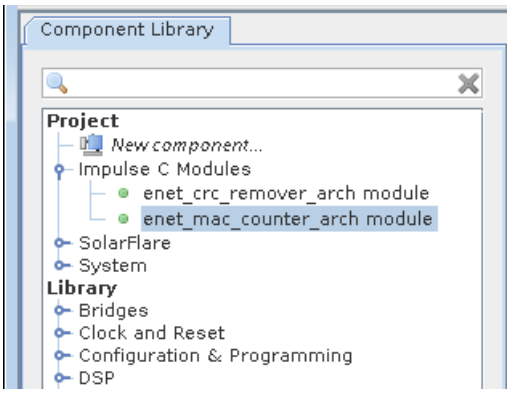
First open the impc_core_system Qsys Project by:

1. Start Qsys from Tools->Qsys
2. When the “Open” dialog appears, select the “impc_core_system.qsys” file and click the “Open” button. Once opened, no errors must be present.

The passthrough previously built has an onchip memory module present that is required in order for the impc_core_system to appear as a component to the AOE. The Impulse C module will replace this and the onchip memory must be first disabled. To disable the onchip memory, scroll down in Qsys to find the “onchip_memory2_0” and disable it by un-checking the associated box in the “Use” column as shown below:

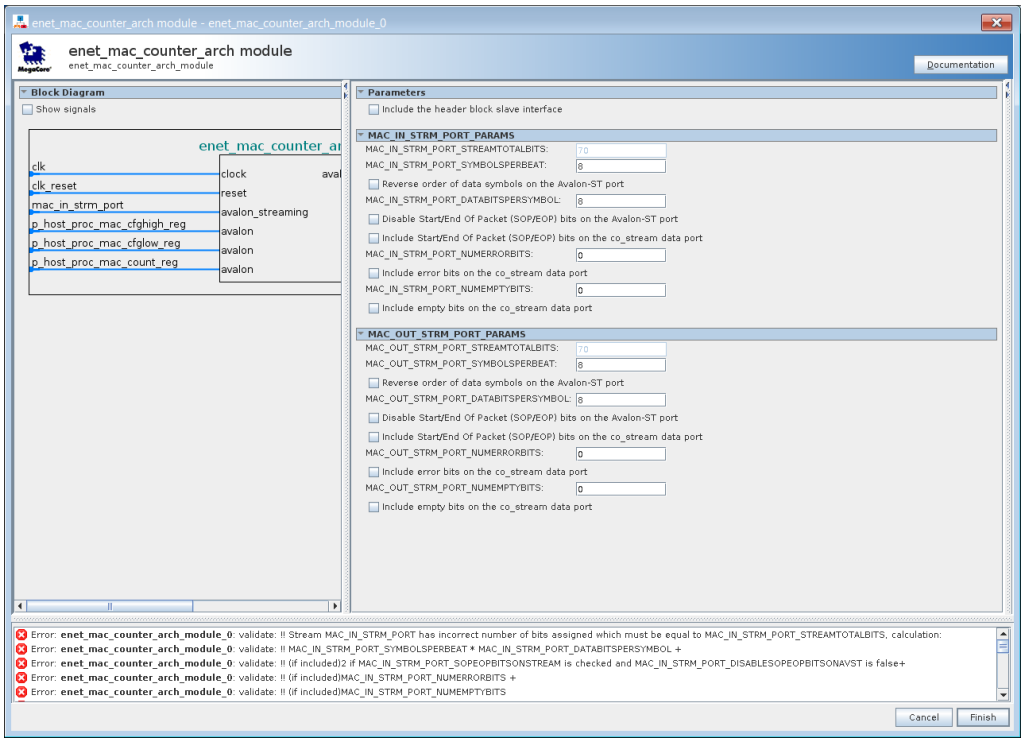


To add the Enet MAC Counter module that had been exported from CoDeveloper, locate the Qsys component named "enet_mac_counter_arch_module" under "Impulse C Modules" as shown below:



To add the module, double-click “enet_mac_counter_arch_module” and the module configuration window will appear similar to below:

Note: At this time errors are normal and will be corrected in the next steps.

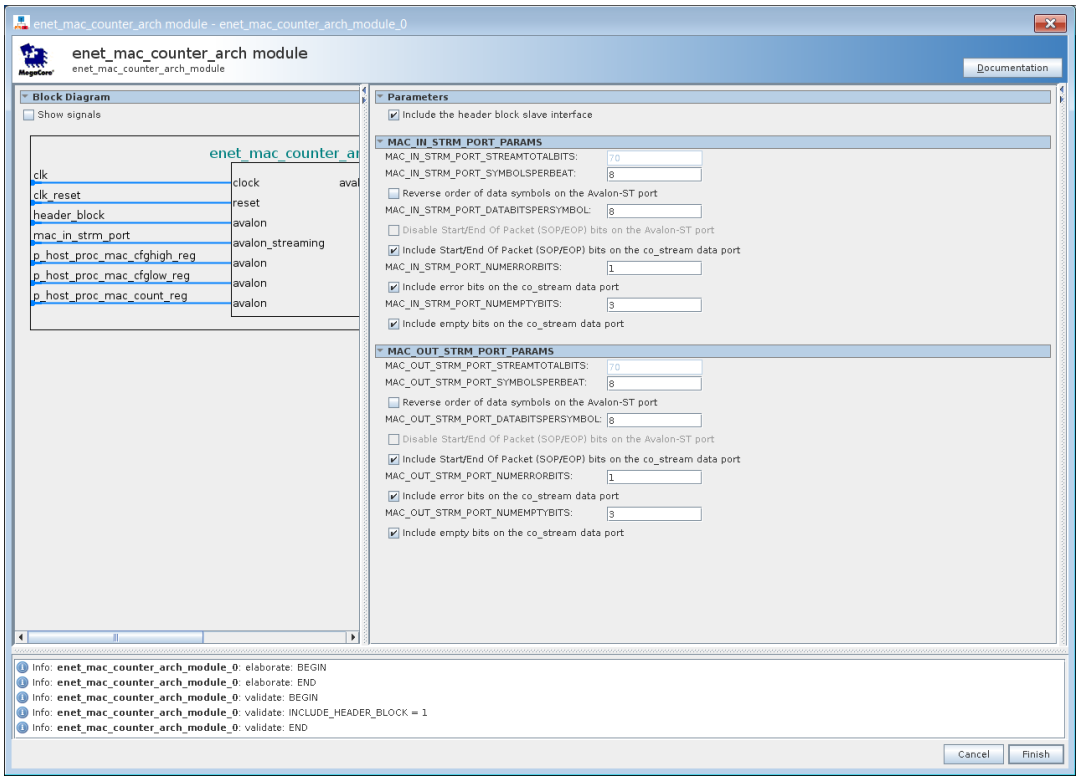


First enable the header block by checking the “Include the header block slave interface”

Now configure the Avalon-ST connection to match the Impulse C enet_mac_counter_proc process by making the following changes for the input and output streams:

- MAC_IN_STRM_PORT_PARAMS:
  - MAC_IN_STRM_PORT_SYMBOLSPERBEAT = 8
  - Check “Include Start/End Of Packet (SOP/EOP) bits on the co_stream data port
  - MAC_IN_STRM_PORT_NUMERRORBITS = 1
  - Check “Include error bits on the co_stream data port
  - MAC_IN_STRM_PORT_NUMEMPTYBITS = 3
  - Check “Include empty bits on the co_stream data port
- MAC_OUT_STRM_PORT_PARAMS:
  - MAC_OUT_STRM_PORT_SYMBOLSPERBEAT = 8
  - Check “Include Start/End Of Packet (SOP/EOP) bits on the co_stream data port
  - MAC_OUT_STRM_PORT_NUMERRORBITS = 1
  - Check “Include error bits on the co_stream data port
  - MAC_OUT_STRM_PORT_NUMEMPTYBITS = 3
  - Check “Include empty bits on the co_stream data port

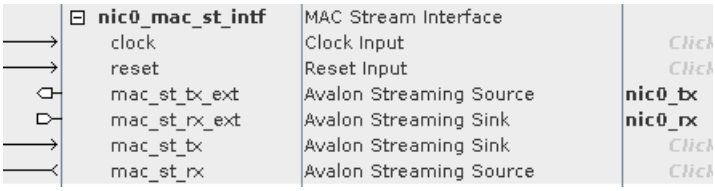
Upon successful settings, no error will be reported and the window will appear similar to below:

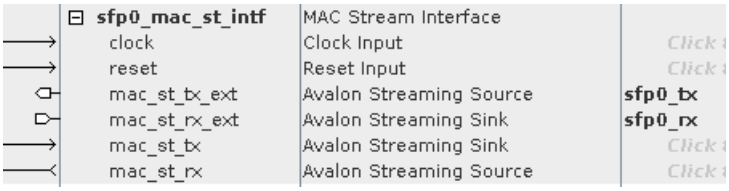


Click the lower-right “Finish” button to continue.

4.10.1.3. Connect the Module to the MACs within the impc_core_system

At this time please take a moment to note the connections available within the Core System, specifically that for each SFP and NIC MAC there are 2 instances numbered 0-1 of the below:

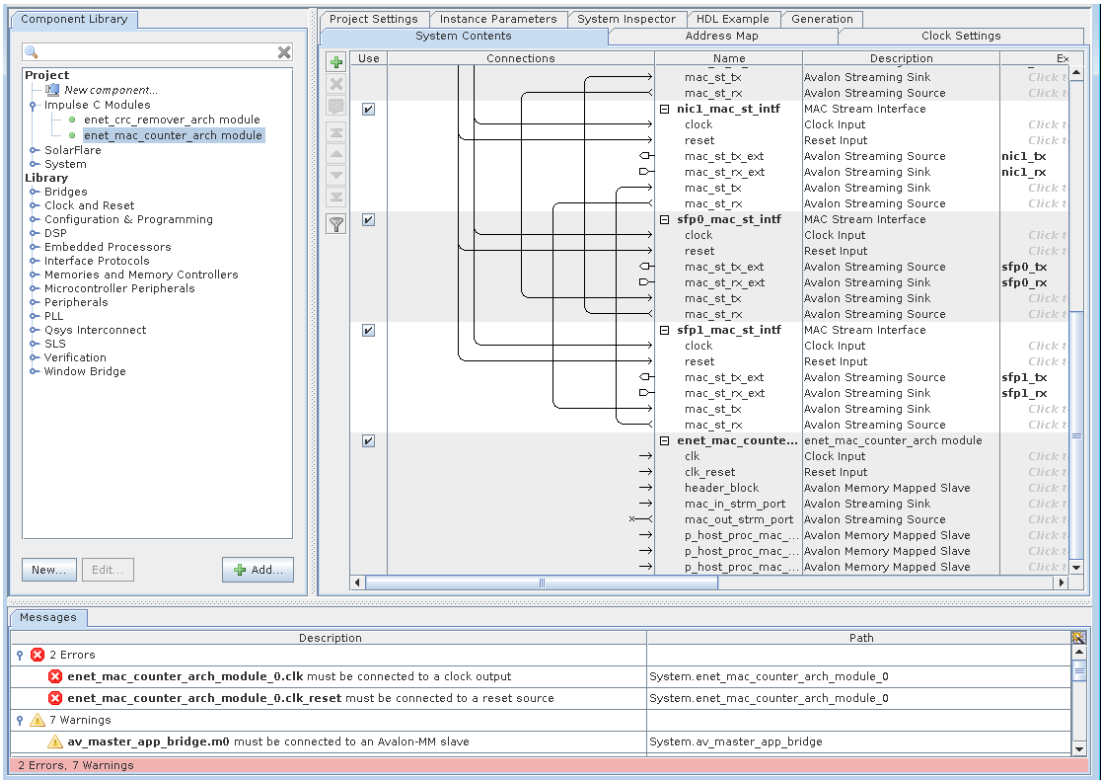




The “sfp[0-1]_st_intf” provide Avalon-ST connections to the MACs for the AOE’s SFP front panel ports #0-1, and the “nic[0-1]_st_intf” provide Avalon-ST connections to the MACs connected to the AOE’s internal NIC ports #0-1.

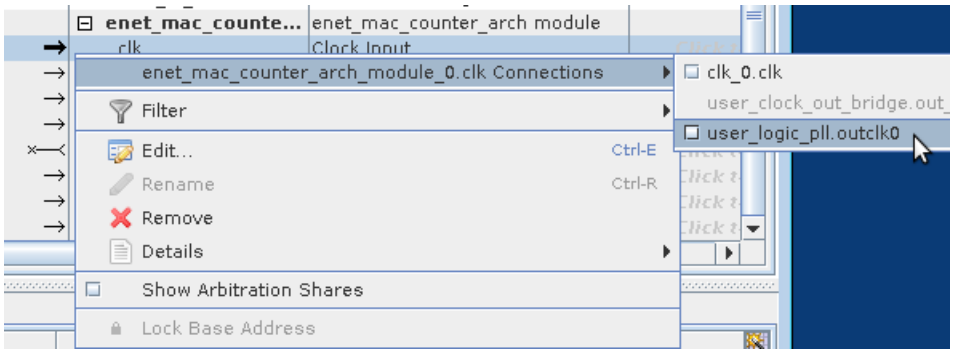
For this example the Enet MAC Counter will be connected between the “sfp1_st_intf” receive stream “mac_st_rx” and the “nic1_st_intf” transmit stream “mac_st_tx”. This will put the Enet MAC Counter module in a path to see all data coming from the network/SFP #1 and the AOE’s NIC port #1.

To see the new module, scroll down to the bottom which will appear similar to below:



Errors and warnings will be present as shown indicating that some connections have not been made which will be done in the following steps.

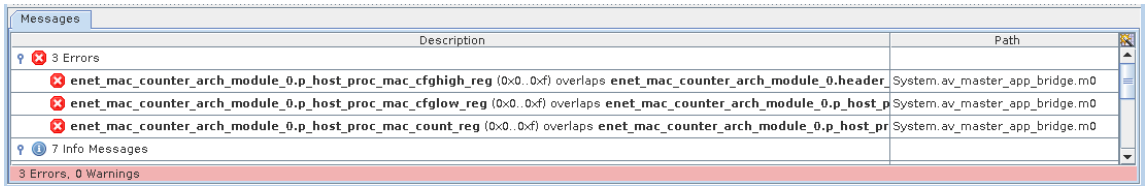
Making connections in Qsys may be done graphically or sometimes more conveniently by right-clicking a port and then selecting from the list provided for possible connections. This is shown below for the “clk” port of the module:



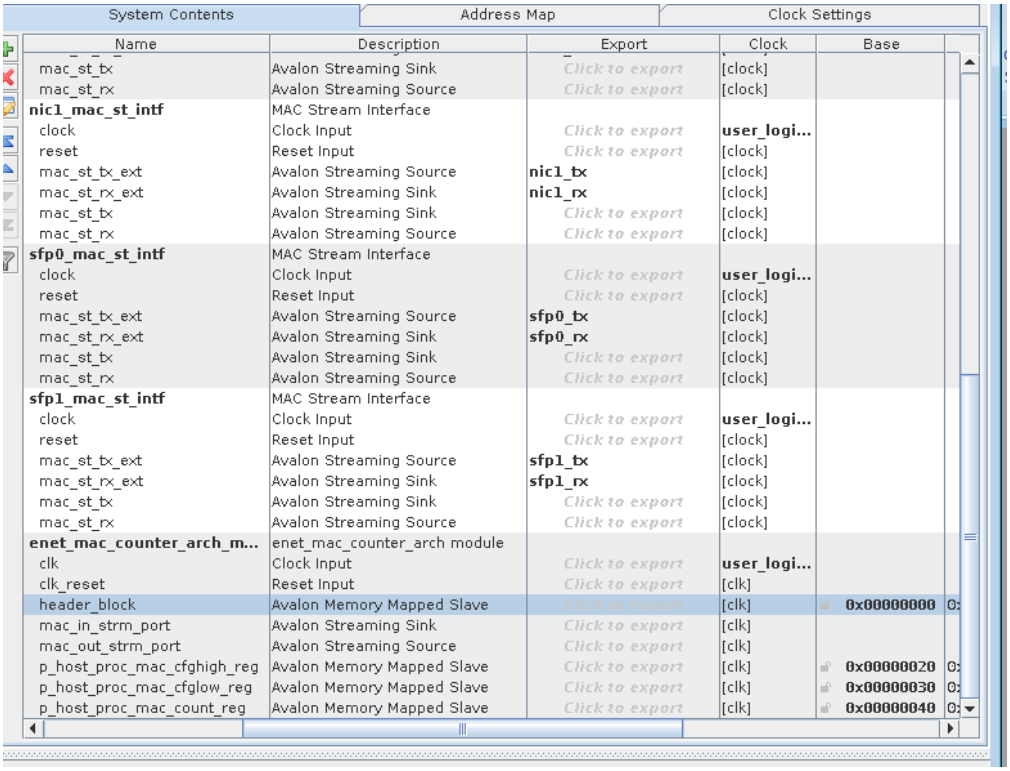
Using this method, make the following connections for each port:

- Connect “clk” to “user_logic_pll.out_clk0”
- Connect “clk_reset” to “user_reset_bridge.out_reset”
- Connect “header_block” to “av_master_app_bridge.m0”
- Connect “mac_in_strm_port” to “sfp1_st_intf.mac_st_rx”
- Connect “mac_out_strm_port” to “nic1_st_intf.mac_st_tx”
- Connect “p_host_proc_mac_cfghigh_reg” to “av_master_app_bridge.m0”
- Connect “p_host_proc_mac_cfglow_reg” to “av_master_app_bridge.m0”
- Connect “p_host_proc_mac_count_reg” to “av_master_app_bridge.m0”

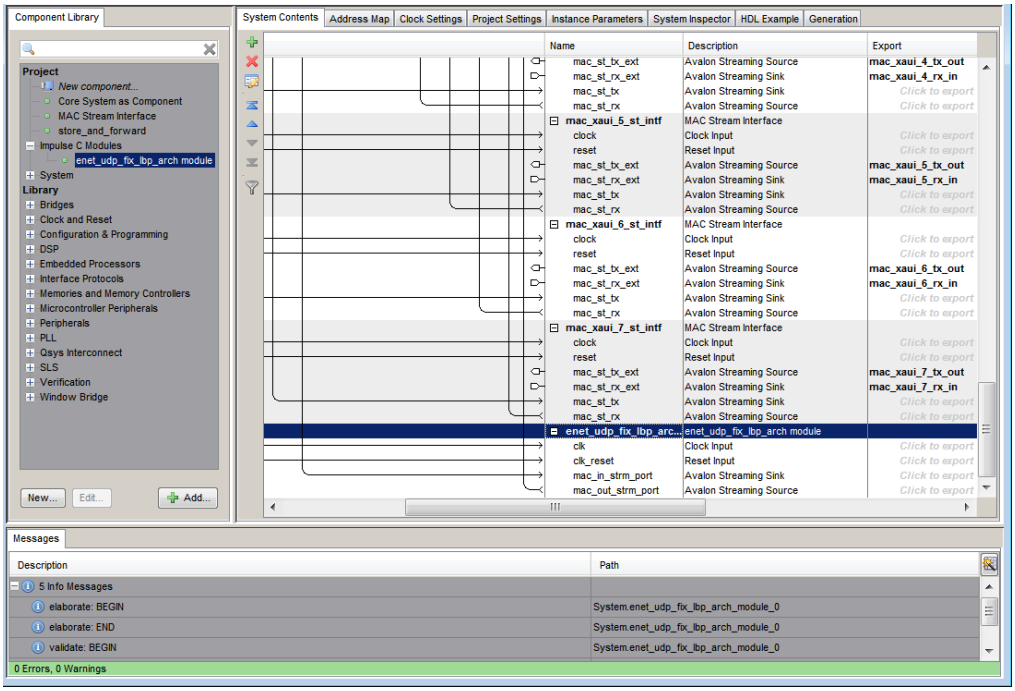
At this time only errors referencing memory overlaps must be present as shown below:



Correct these final errors automatically in Qsys by using System->"Assign Base Addresses". When complete, the "header_block" base address shown in the "Base" column must be 0x00000000 as shown:



Upon successful connections, at this time no error or warnings will be present and all ports of the module are connected similar to below:



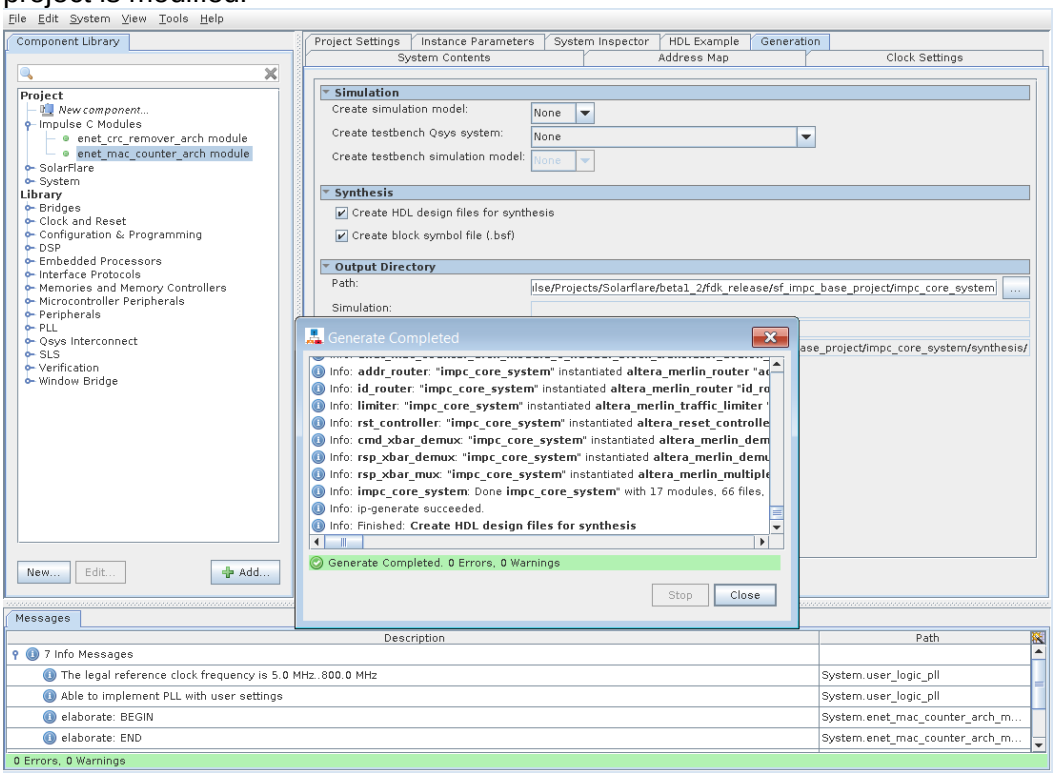
Save the project using Ctrl+S or File->Save

4.10.1.4. Generate the Core System

Now generate the HDL files for the impc_core_system by clicking on the top-right “Generation” tab, and then clicking the lower-left “Generate” button. Once complete, the impc_core_system HDL will have been generated for the Qsys system in a subdirectory of the same name and no errors must be present similar as shown below:

Notes:

- This step must be repeated each time the Enet MAC Counter module has been exported or when a change to the ‘impc_core_system.qsys’ Qsys project is modified.



4.11. Build the FPGA Binary

Start compilation of the FPGA from within Quartus by selecting Processing->"Start Compilation". Viewing the "Tasks" window will show the progress of building the FPGA through the "Analysis & Synthesis", "Fitter", "Assembler", and "TimeQuest Timing Analysis" phases. No errors should be reported at any time during compilation.

When complete, the FPGA binary will appear as 'sf_impc_base_project.sof' ready to be programmed into the FPGA using the Byte Blaster cable.

Notes:

- If an error does appear, it will typically appear during the first ~10 minutes of the "Analysis & Synthesis" phase and indicates a file was not found correctly. Please verify all the files were unzipped correctly into the correct location and try again.
- Timing errors will not prevent an FPGA binary from being created and will appear in the "Critical Warnings" window in Quartus. The use of partitions to save compile time sometimes causes unintentional timing errors to appear, to disable the partition: from the Assignments->"Design Partitions Window" change the setting for "board_services" from "Post-Fit" to "Source File".

4.12. Program the FPGA Binary

The FPGA binary may be programmed into the AOE in one of two ways: Command line utility to program the flash and then rebooting the system (see AOE documentation for more details) or using an Altera Byte Blaster programming cable which is recommended and described here.

4.12.1. Program the FPGA Binary Using the Programming Cable

Programming the FPGA on the AOE may be done using the Altera Byte Blaster USB cable as follows:

1) Connect the Byte Blaster USB cable from the host running Quartus to the AOE. The JTAG connector is located at the top of the card and requires an adapter to plug into the AOE.
2) Start the programmer from Quartus using Tools->Programmer
    a. "Hardware Setup" should already show the USB Byte Blaster cable
    b. "Auto Detect" will identify the FPGA, specify "5SGXMA5K" when prompted for the specific device found.
    c. Select the FPGA "5SGXMA5K" device:
        i. Right-click and select "Change File"
        ii. Browse to select either the 'sf_impc_base_project.sof' file
        iii. Check the box for "Program/Configure"
    d. Program device by clicking the "Start" button.
    e. Save programmer configuration using File->Save

4.13. Building Host Software

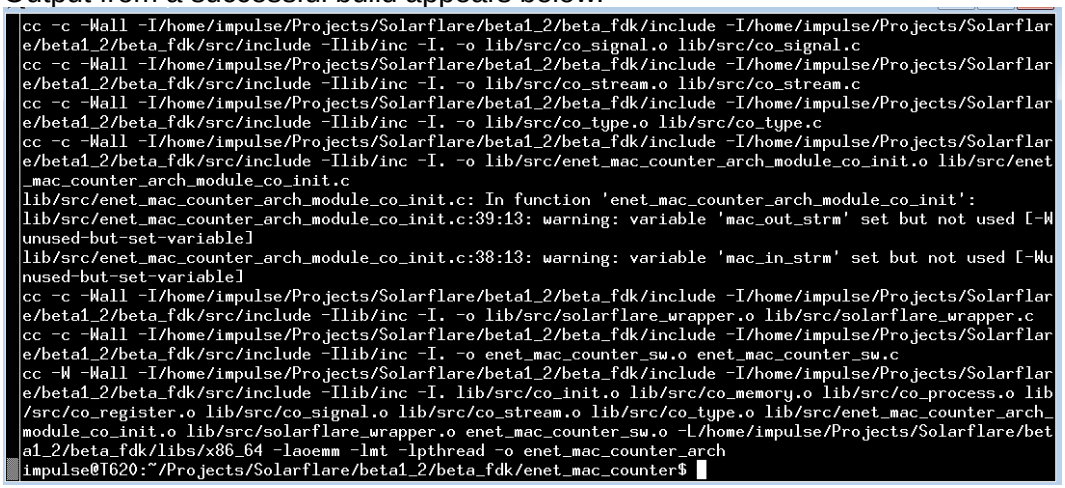
The original host software is set to look for a specific multicast MAC, here change it to look for a broadcast MAC by editing “enet_mac_counter_sw.c” and changing the following definitions at the top of the file to appear as:

```
#define DEST_MAC_LOW    0xFFFFFFFF  // lower 32-bits of MAC
#define DEST_MAC_HIGH   0xFFFF      // upper 16-bits of MAC
```

Save the file.

Build the host executable within the directory that “export_sw” had been copied to, such as “<application dir>/enet_mac_counter” using the following steps.

1) While generating HDL in Qsys, the file “impc_core_system.sopcinfo” is created which contains memory map information needed by the host software. Copy this file to the “<application dir>/enet_mac_counter” software directory.
2) From a shell, change directory into the “<application dir>/enet_mac_counter”
3) Build the host executable by running: make
4) Output from a successful build appears below:



Notes:

- The FDK_PATH variable must be set correctly in order to build
- impc_core_system.sopcinfo should be copied each time when generating HDL from Qsys, however it must be copied each time the memory map changes

4.14. Run and Test the FPGA Binary with Host Executable

Before continuing, the network must already be configured as described earlier in the "Network Setup" section.

When run, the host executable will perform the following actions:
1) Configure the destination MAC that is to be counted
2) Poll the FPGA register 10 times displaying the MAC count read
3) Exit

Run the host executable from the "<application dir>/enet_mac_counter" by typing: ./enet_mac_counter_arch

Without traffic, the MAC count should start and stay at zero. To change the count being read, use "ping" or another command to generate broadcast messages going out the AOE's SFP #0 which will then be counted as they come in the SFP #1.

For example using ping to generate ARP requests that are broadcast, assuming eth5=SFP #0 and configured as IP 192.168.5.62, then "ping 192.168.5.1" will cause ARP messages to be sent out SFP #0 which will be seen, though not responded to, by SFP #1. With the Enet MAC Counter FPGA image loaded, these broadcasts will now be counted.

Output while running the host executable (and ping simultaneously) will appear something similar to below:

```
host_proc:MAC match count = 0


Application complete.
impulse@T620:~/Projects/Solarflare/beta1_2/beta_fdk/enet_mac_counter$ newx
[4] 20056
impulse@T620:~/Projects/Solarflare/beta1_2/beta_fdk/enet_mac_counter$ Warn
brary, locale unchanged
./enet_mac_counter_arch
Impulse C is Copyright(c) Impulse Accelerated Technologies, Inc.
host_proc:MAC match count = 0
host_proc:MAC match count = 0
host_proc:MAC match count = 1
host_proc:MAC match count = 2
host_proc:MAC match count = 3
host_proc:MAC match count = 4
host_proc:MAC match count = 5
host_proc:MAC match count = 6
host_proc:MAC match count = 7
host_proc:MAC match count = 8


Application complete.
impulse@T620:~/Projects/Solarflare/beta1_2/beta_fdk/enet_mac_counter$
```