AXI4 Wrapper for MPTL Target Endpoint

Revision: 1.0
Date: 21/10/2012

	Head Office	US Office
Address	4 West Silvermills Lane, Edinburgh, EH3 5BD, UK	3507 Ringsby Court Suite 105 Denver, CO 80216
Telephone	+44 131 558 2600	(303) 954 8768
Fax	+44 131 558 2700	(866) 820 9956 - toll free
email	sales@alpha-data.com	sales@alpha-data.com
website	http://www.alpha-data.com	http://www.alpha-data.com

Table Of Contents

Tables

Figures

Page Intentionally left blank.

1 Introduction

This document decribes the operation of AXI4 wrapper IP for the Alpha Data MPTL endpoint. These cores communicate with the always connected PCIe Alpha Data ADB3 bridge devices on the following Alpha Data boards : ADMXRC6T1, ADMXRC6TGE, ADPEXRC6T, ADPEXRC6TL, ADMXRC7K1, ADMXRC7V1. The cores allow connection of up to 5 independent AXI-4 buses which can be accessed either as a memory mapped BAR using Direct Slave from the host processor or via the high performance DMA engines in the bridge.

These cores have been designed to be instantiated using Xilinx EDK Platform Studio (and successor tools), and hence contain both source VHDL and netlists as well as the appropriate IP description files for inclusion in a Xilinx Platform Studio project.

This document describes a range of cores suitable for different boards:

- mptl_target_v6x4_wrap - suitable for ADMXRC6T1, ADMXRC6TGE and ADPEXRC6T
- mptl_target_v6x1_wrap - suitable for ADPEXRC6TL
- mptl_target_k6x4_wrap - suitable for ADMXRC7K1
- mptl_target_v7xx4_wrap - suitable for ADMXRC7V1 boards with GTX tranceivers in the FPGA
- mptl_target_v7hx4_wrap - suitable for ADMXRC7V1 boards with GTH tranceivers in the FPGA

This document will first describe the cores operation, detailing the ports and generics that can be set. This will be followed by a tutorial which walks through the steps required to generate a basic example design.

2 MPTL_TARGET_WRAP AXI4 Cores

Figure 1 shows the structure of the MPTL Wrapper cores. All the cores share the same structure and only differ in their internal GTX/GTH configuration (to support V6,V7 and K7 devices), the number of DMA channels and the number of Serial Links to the bridge device (x4 or x1).

Within the wrapper the core functionality is contained in the MPTL-OCP endpoint netlist as supplied in all ADMXRC3 SDK examples. This core bridges between the packet based serial data transmitted over the GTX links between bridge and target FPGAs and independent OCP ports. This core handles the multiplexing of data between these channels allowing all the OCP ports to operate as independent streams. This core module runs in 2 clock domains, the GTX interfaces run at 125MHz, whereas the backend logic runs in the clock domain of a user supplied clock "ocp_clk". It is recommended that ocp_clk runs faster than 125MHz.

Each independent OCP channel is then converted to a full AXI-4 memory mapped interface. This is largely just a signal renaming exercise, however a few small state machines are required to implement the full protocol. Each AXI-4 interface can have its own independent clock domain, so the wrapper also contains OCP cross clock domain converter blocks as well to support this.

In this release, the AXI-4 ports do not use AXI-4 ID signals (due to an XPS GUI limitation) to tag requests, therefore it is recommended that multithreaded host access of the direct slave port be avoided where multiple AXI-4 Slave devices are attached which may return data out of order.

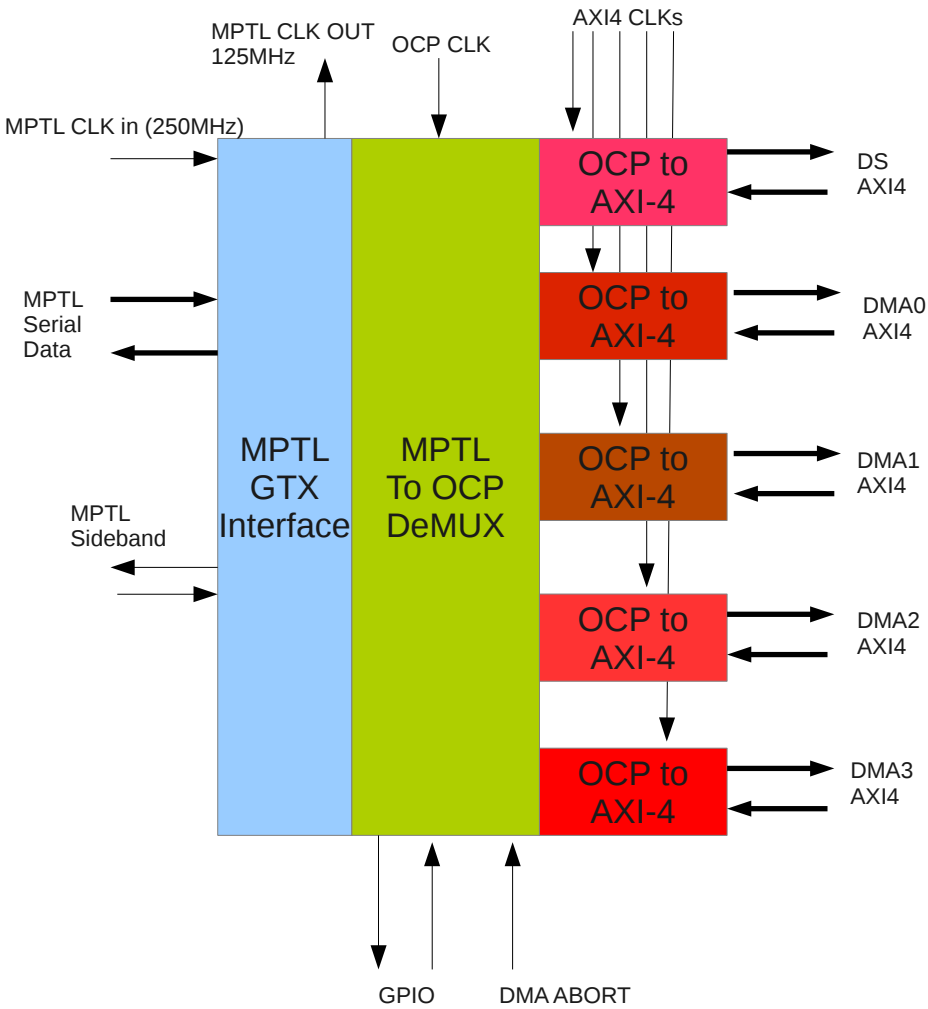

Figure 1: AXI-4 MPTL Target Wrapper

The wrappers contain a number of signals in addition to the AXI-4 memory mapped ports, some of these should be connected up to external IO pins, others should be connected up internally.

ocp_clk	Internal	user provided clock >125MHz
mptl_tx_p	External	Target to Bridge Serial Link(s)
mptl_tx_n	External	Target to Bridge Serial Link(s)
mptl_rx_p	External	Bridge to Target Serial Link(s)
mptl_rx_n	External	Bridge to Target Serial Link(s)
mptl_clk_p	External	Serial Link GTX Reference Clock
mptl_clk_n	External	Serial Link GTX Reference Clock
mptl_clk_out	Internal	125 MHz clock available to user
mptl_resetn_out	Internal	optional active low reset for user
mptl_target_configured_n	External	Link Sideband Signal
mptl_target_gtp_online_n	External	Link Sideband Signal
mptl_bridge_gtp_online_n	External	Link Sideband Signal
gpio_t2b	Internal	optional GPIO status signals
gpio_b2t	Internal	optional GPIO control signals
dma_abort	Internal	allows target FPGA to stop DMA

Table 1: MPTL Target Wrap signals (non-AXI4)

Note that the external signals must be connected up to the pins specified in the ADMXRC3 SDK UCF files. The names in the SDK may differ slightly from these e.g. tx and t2b are equivalent as are rx and b2t, and the SDK pin names may be significantly longer due to the use of VHDL records at the top level.

3 Tutorial

The following section documents how to create an EDK project from scratch using the mptl_target_wrapper blocks. The example is described for the ADM-XRC-7K1, however targetting a different board is simply a case of selecting the correct FPGA, mptl target wrapper IP core and ucf constraints. The example connects all the Master AXI-4 ports of the module to a single BRAM memory buffer. These instructions use XPS in ISE 14.2, other tools may require a slightly different approach.

In general it is not a good idea to map all your slave (AXI-4 peripherals) to all the Master Ports (effectively generating a single address space) as you will either have to implement a very large crossbar AXI-4 switch to keep your bandwidth or you will end up with a low performance shared bus. It is recommended that you connect the Direct Slave port to register type (low speed) Slave peripherals for control functions and connect individual DMA channels to specific high speed devices and large memory buffers that require high bandwidth host access.

3.1 Creating a Project

Launch XPS. From the Getting Started section choose Create New Blank Project.

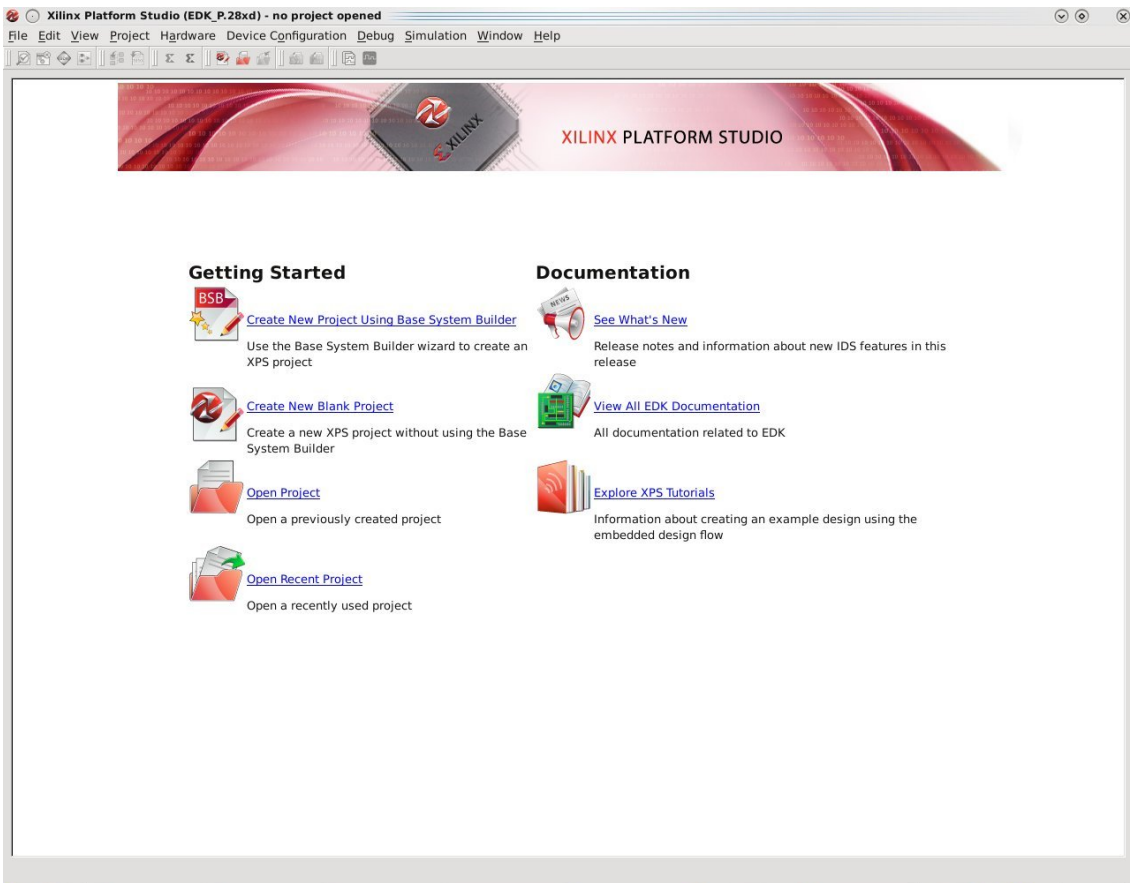

Figure 2: Create a New Blank Project

Enter a project file name, e.g. test_axi4

Select a target FPGA device, for the ADMXRC7K1 you can choose:

- Architecture : kintex7

- Device Size : xc7k325t
- Package : ffg900
- Speed Grade : -1

To use on other boards, choose the correct FPGA specification

Uncheck Auto Instatiate Clock/Reset options. These may be useful in your own future projects, but for this simple demo, we'll keep the clocking as simple as possible

Select OK

This should generate you a blank project within which you can build your system

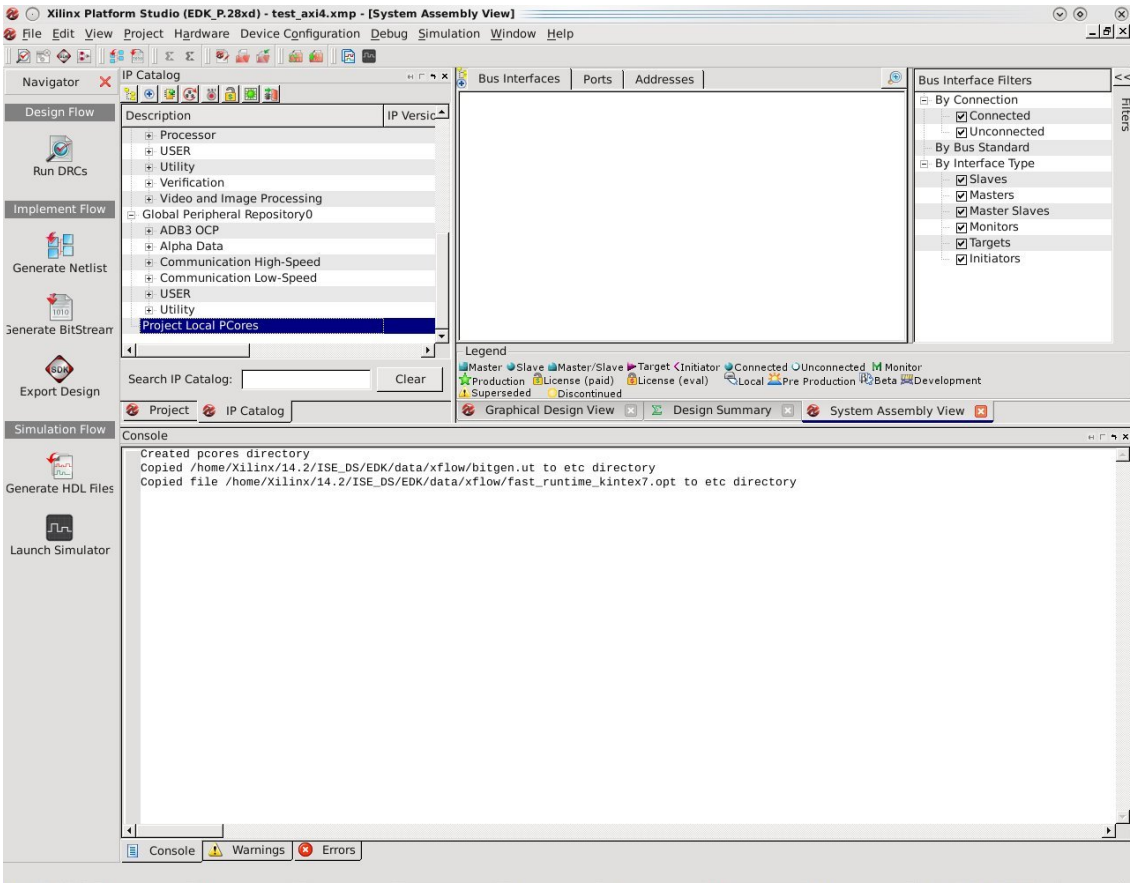

Figure 3: Blank Project

3.2 Importing the IP Cores

Experienced EDK uses will already know how to add their own IP to an individual project or to their own user repository. In this tutorial we will simply add the IP cores to the local pcores directory so that they can be accessed.

In the project you've just created, there will be a pcores directory. Using a file manager, windows explorer or other equivalent tool, copy the contents of the pcores directory in the AXI-4 MPTL Target Wrapper zip file into the projects pcores directory. Note that for the ADMXRC7K1 example you only need the subfolder mptl_target_k7x4_wrap_v1_00_a and the ibufds_clk_wrap_v1_00_a. The second IP block is a basic wrapping of an IBUFDS component to allow a differential clock on 2 pins to be easily used in the design.

If you now return to the XPS GUI and select Project->Rescan User Repoitories, you should now see the cores in the USER section of the Project Local PCores section of the IP Catalogue

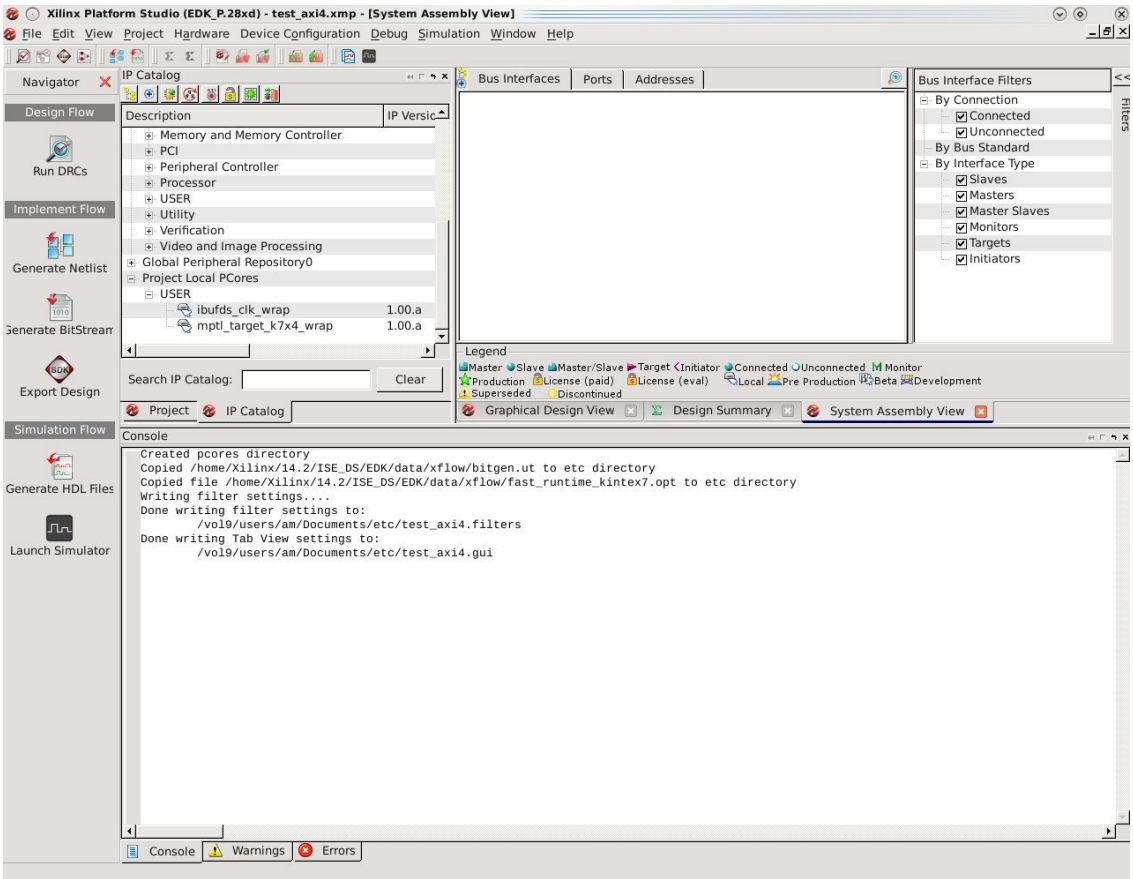

Figure 4: User PCores

3.3 Building the System

In this sub-section we will add all the IP we need for our system

Select mptl_target_k7x4_wrap and select Add IP. Click on Yes and OK to add the block.

Select ibufds_clk_wrap and select Add IP. Click on Yes and OK to add the block. This will give us a simple clock input which can be driven by the 200MHz reference clock available on the board. Other clocks can be used for both thee OCP and AXI clocks in your own designs.

Go to the EDK Install section of the IP Catalog and select and add an AXI Interconnect block from the Bus and Bridges sub-section

From the Memory and Memory Controller sub-section select and add an AXI BRAM controller. Parameterise this block by setting its Bae address to 0x00000000, its high address to 0x00003FFF. This will specify the size of the memory. You can also optionally set the AXI Data width to 128 bits which will match the MPTL blocks native width, although this is not strictly necessary.

Also from the Memory and Memory Controller sub-section select and add a Block RAM (BRAM) Block

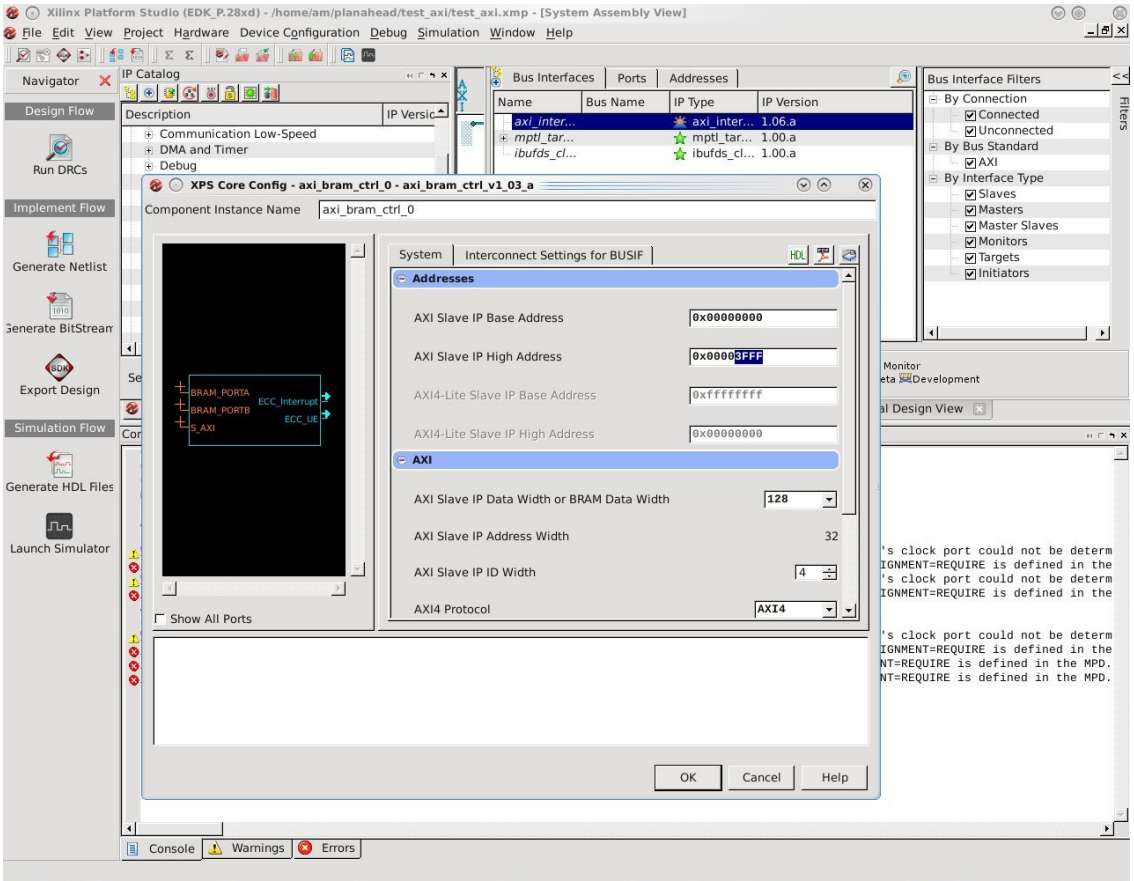

Figure 5: BRAM Configuration

3.4 Connecting the Blocks

Now the instantiated blocks need to be connected together

First connect up the Bus Interfaces in the System Assembly View. For the 5 M_AXI interfaces on the mptl_target block you should be able to connect them all up to the axi_interconnect_0 block. You can now connect the S_AXI interface on the axi_bram_ctrl_0 block up to the AXI interconnect as well. This will give a choice of masters which can access it. Tick all 5.

Finally connect up the BRAM block to the BRAM controller.

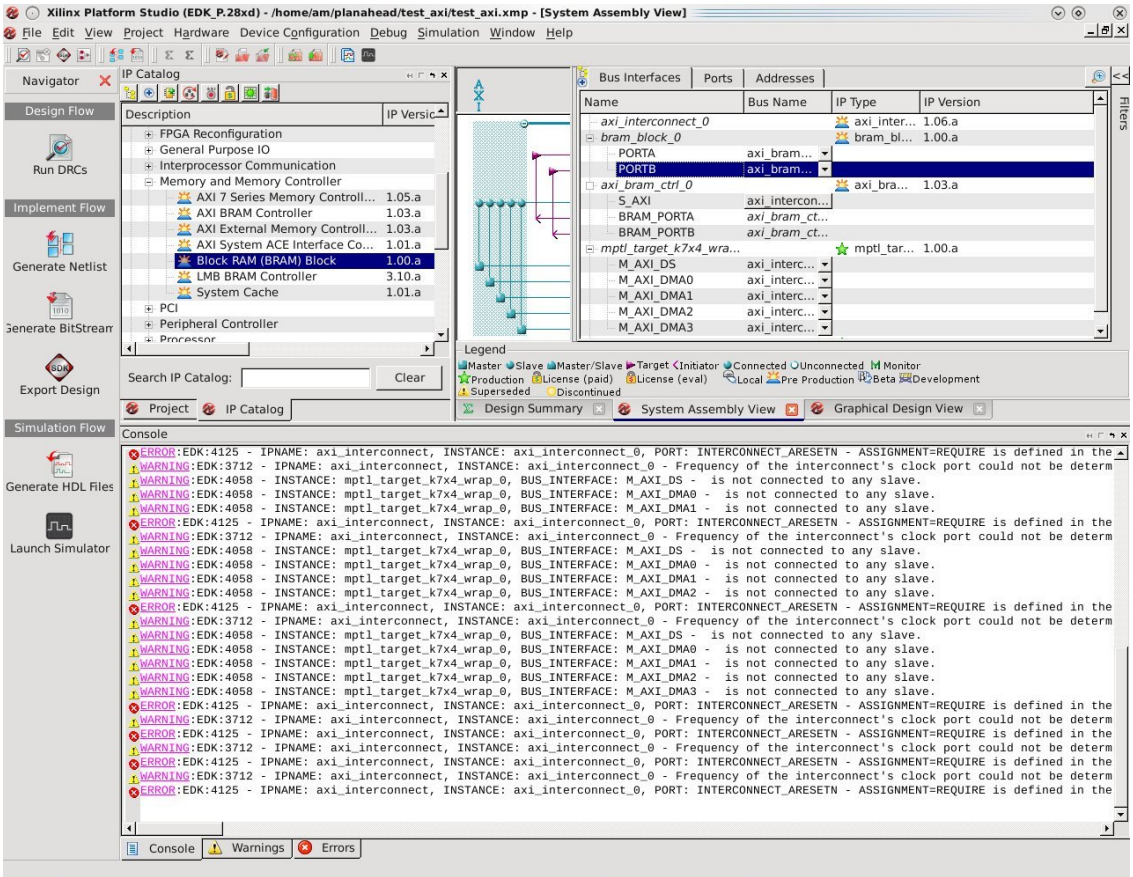

Figure 6: System Assembly

Now in the System Assembly View select the Ports tab. Expand the IP blocks to see the signals you can connect.

All the AXI4 ports need a clock. Expand the (BUS_IF) S_AXI and M_AXI ports top reveal clock ports. Connect all thes up to the OCLK output of ibufds_clk_wrap. Also connect up the INTERCONNECT_ACLK signal on the axi_interconect_0 module and ocp_clk on the mptl wrapper block to this clock.

The axi_interconnect_0 block needs an active low reset, you can connect this port up to mptl_target_k7x4_wrap_0: mtpl_resetn_out.

On the mptl_target_k7x4_wrap_0 block connect up the dma_abort signal to net_gnd

With the gpio_t2b input you can connect it up to gpio_b2t to give a simple loopback test.

A few signals (outputs from the BRAM controller and mptl_clk_out) can be left unconnected, but the rest of the signals need to be connected to external pins

On module ibufds_clk_wrap, connect iclk_p and iclk_n to external ports, renaming them ref_clk_p and ref_clk_n respectively.

On the mptl_target_k7x4_wrap_0 block the ports: mptl_tx_n, mptl_tx_p, mptl_rx_n, mptl_rx_p, mptl_clk_p, mptl_clk_n, mptl_target_configured_n, mptl_gtp_online_n and mptl_target_online_n all need connected to external ports. Shortening the names suggested by the GUI is recommended

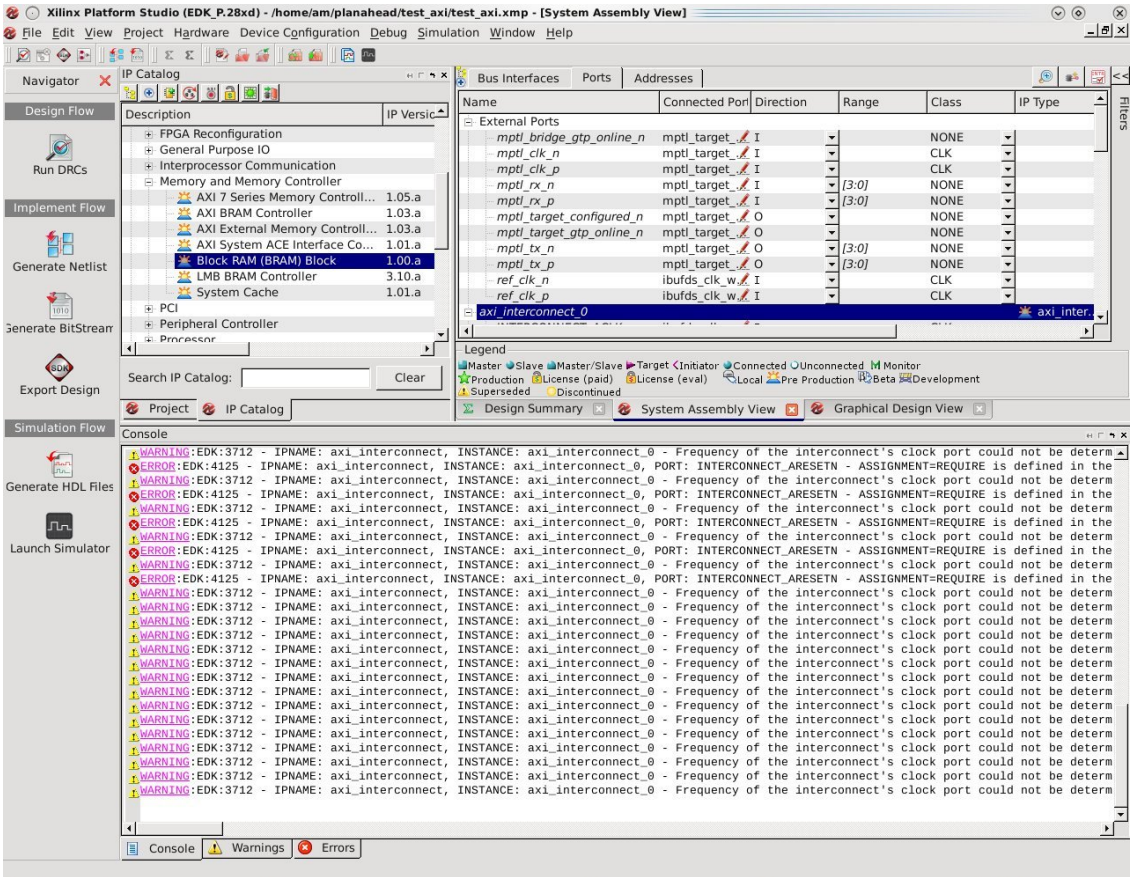

Figure 7: External Ports

You can check the connectivity of the ports using the Graphical Design View of the System. This should hopefully make it easy to spot any unconnected signals

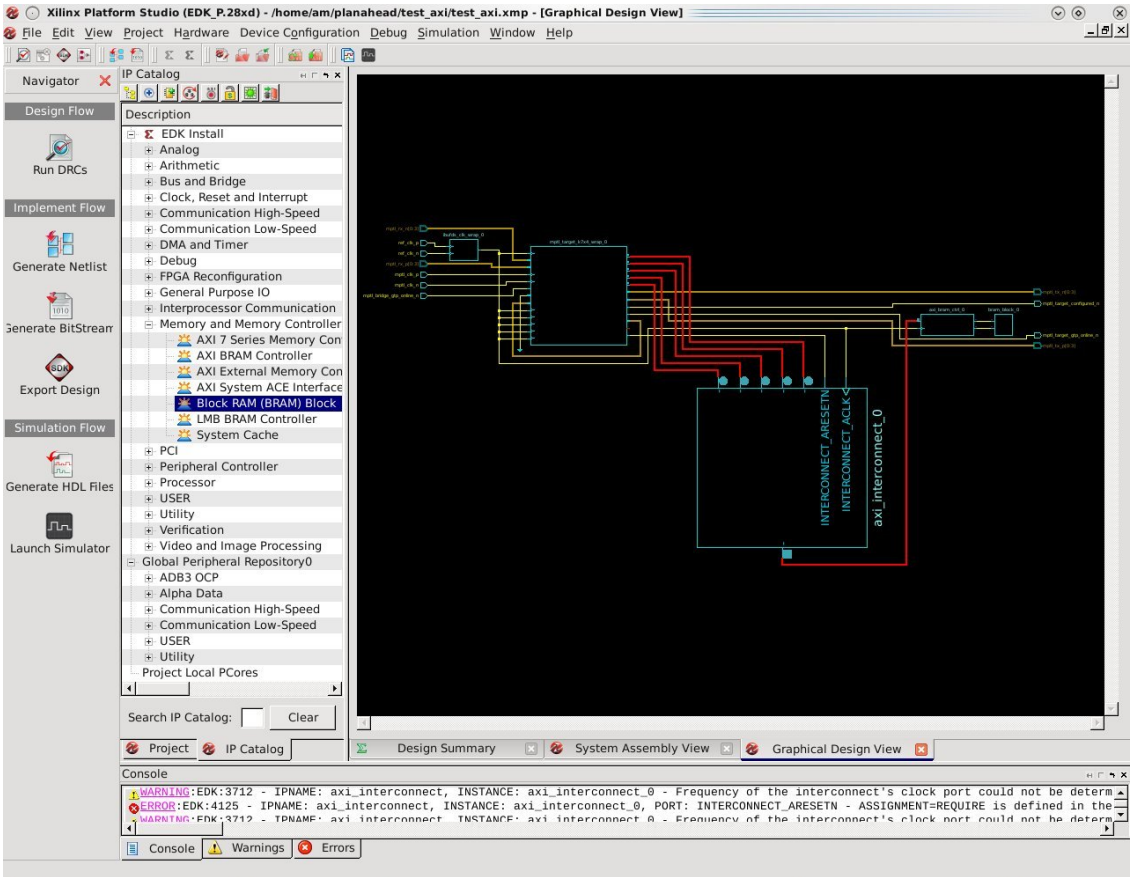

Figure 8: Graphical View of the System

3.5 Constraints

The design now needs pin and timing constraints. The easiest place to find these is in the simple reference design provided in the ADMXRC3 SDK. You can copy, paste and edit the constraints from the file in %ADMXRC3_SDK%\hdl\vhdl\examples\simple\admxrc7k1\simple-admxrc7k1.ucf into the XPS projects default constraints file. If you are targetting another board use the constraints file provided in the simple example for that board.

The pin names will need modified slightly, changing t2b to tx, b2t to rx and also making sure the sideband signal names match up.

There are many ways to constrain clocks, however the simplest way, sufficient for this example is to add a simple period constraint to mptl_clk_p (250MHz) and ref_clk_p (200MHz).

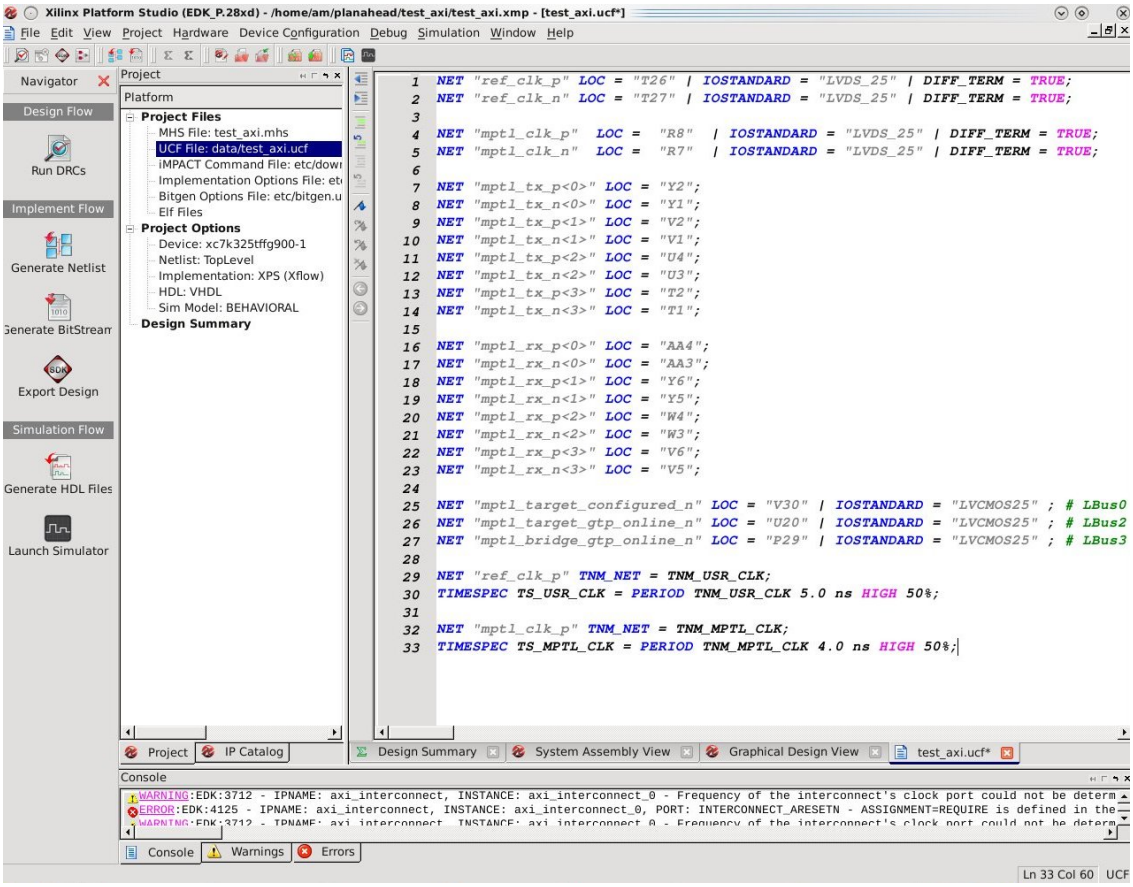

Figure 9: Constraints

3.6 Building the Bitstream

Before building the bitstream a few options need to be changed in the file etc/bitgen.ut. Without these changes the done pin will not signal that the target is configured and unused pins may be driven to unexpected values.

- In the project files, open bitgen.ut.
- Change the option -g DriveDone:No to -g DriveDone:Yes
- Add the option -g UnusedPin:PullNone

You should now be able to build the bitstream simply by selecting the Hardware->Generate Bitstream option, this will run through all the synthesis, implementation and bitstream generation stages.

Note that in some exceptional cases the default design may not meet timing. Adjusting the options in the implementation file may fix this. Adding regisiter stages to the AXI-4 BRAM Interface's AXI port may also help with meeting timing.

3.7 Testing in Hardware

You can configure your target FPGA using either a JTAG cable or the loader.exe program provided in the ADMXRC3 SDK. You can test the reading and writing of the BRAM in the target FPGA using the dump.exe program in the ADMXRC3 SDK.

```
> dump.exe wd 0 0x00 4 0x123
```

```
> dump.exe wd 0 0x14 4 0x456
> dump.exe wd 0 0x28 4 0x789
> dump.exe wd 0 0x3C 4 0xABC
> dump.exe rd 0 0x0 0x40
Window 0 offset 0x0 mapped @ 0x0x7f5c84364000
Dump of memory at 0x0x7f5c84364000 + 64(0x40) bytes:
00       04       08       0c
0x00007f5c_0x84364000: 00000123 00000000 00000000 00000000 #...............
0x00007f5c_0x84364010: 00000000 00000456 00000000 00000000 ....V...........
0x00007f5c_0x84364020: 00000000 00000000 00000789 00000000 ................
0x00007f5c_0x84364030: 00000000 00000000 00000000 00000abc ................
```

Revision History:

Date	Revision	Nature of Change
20/11/12	1.0	Initial Release

Address: 4 West Silvermills Lane,
Edinburgh, EH3 5BD, UK
Telephone: +44 131 558 2600
Fax: +44 131 558 2700
email: sales@alpha-data.com
website: http://www.alpha-data.com

Address: 3507 Ringsby Court Suite 105
Denver, CO 80216
Telephone: (303) 954 8768
Fax: (866) 820 9956 - toll free
email: sales@alpha-data.com
website: http://www.alpha-data.com