XRC Board Level Application Library v2.2a (For Simulink and System Generator)

ALPHA DATA

Copyright © 2004 Alpha Data Parallel Systems Ltd. All rights reserved.

This publication is protected by Copyright Law, with all rights reserved. No part of this publication may be reproduced, in any shape or form, without prior written consent from Alpha Data Parallel Systems Limited.

Alpha Data Parallel Systems Ltd.
4 West Silvermills Lane
Edinburgh EH3 5BD
Scotland, UK

Phone: +44 (0) 131 558 2600
Fax: +44 (0) 131 558 2700
Email: support@alphadata.co.uk

1 Introduction

The Alpha Data XRC Board Level Application Library 2.2 is a collection of embedded IP, VHDL, Simulation Models, and a top level Wrapper Builder application, all designed to ease the design of FPGA applications on Alpha Data Embedded PMC and PCI cards. These modules are designed to work with Xilinx System Generator 8.1, and Matlab/Simulink 7.1(sp3) (R14sp3) from the Mathworks.

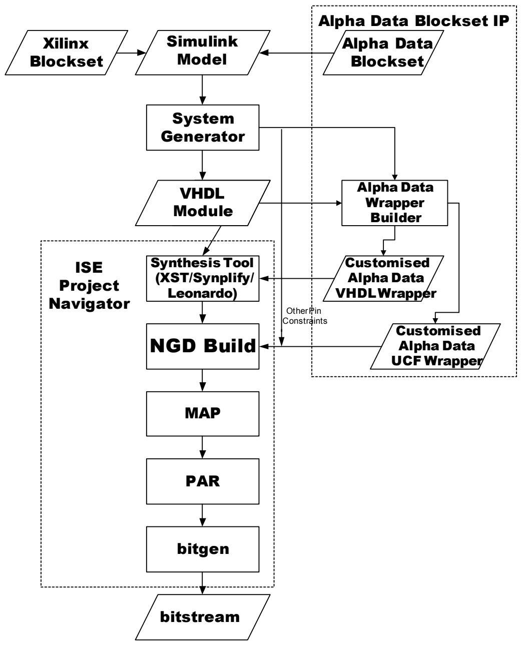
1.1 Whats New

Updates from 2.1 to 2.2.

This blockset only supports ISE/System Generator 8.1
This blockset only supports Matlab/Simulink versions 7.1(sp3).

Due to changes in the files output by System Generator, the wrapper builder now only builds VHDL and UCF files, based on the System Generator VHDL. When targeting ISE, the generated wrapper VHDL should be combined with the System Generator VHDL. Additional ports specified in System Generator are not located automatically in the ucf file. These have to be manually added from the System Generator constraints file (in 8.1 the <design>_cw.xcf file).

The design flow is now as follows:



2 Installation Instructions

The XRC Board Level Application Library is provided as a zip file and a Matlab installation script:

xrc_application_blockset.zip
setup_xrc_application.m

1) Open Matlab

2) Change Directory to where xrc_application_blockset.zip and setup_xrc_application.m have been downloaded.

3) Type setup_xrc_application

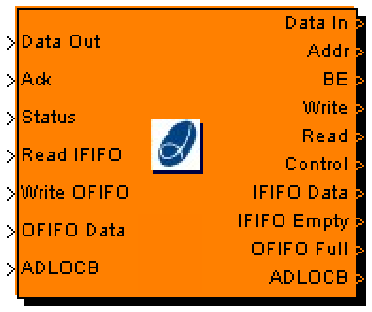
4) Quit Matlab

3 XRC Simulink Blockset

The XRC Blockset provides simulation and synthesis models to handle the most common I/O tasks in the ADM-XRC Series of cards. Communication with the host is through a Local Bus bridge and PCI. The modules provided have control and status registers (for simple control; tasks), a slow, asynchronous, memory mapped, slave interface (for more sophisticated control and small data transfers), and input and output demand mode DMA FIFO's for large high speed data transfers to and from the host. The other modules provided interface to the off-chip memory on each card (ZBT, DDR-SDRAM, DDR-II SSRAM). Other I/O is generally unidirectional and synchronous to the System Generator clock domain, and this can be implemented using the standard System Generator I/O ports. All the modules contain ADLOCB ports. This allows the components to be chained together on a single bus controlled by the host, and allow fast background access to the off-chip memory.

3.1 Local Bus Interface Modules

3.1.1 LBIF_PLX32



PLX 32 Bit Interface

This interface connects to the PLX9656 PCI bridge chip used on the ADM-XRC-II. It provides the following ports:

Control : (UFix_32_0) -- control register value – set by host at address 0x40000
Status : (Ufix_32_0) -- status register – read by host at address 0x40040

The slave memory mapped interface uses the following ports:
Data_Write: (Ufix_32_0) -- data written by host
Addr: (UFix_16_0) -- 32 bit address for read or write
BE: (Ufix_4_0) -- byte enables
Write: (Boolean) -- host wants to write data (Data_Write) to Address (Addr)
Read: (Boolean) -- host wants to read data from Address(Addr)
Data_Read: (UFix_32_0) -- data to be returned to host
Ack: (Boolean) -- indicates that Data_Read is valid – must be asserted in response to Read.

The slave memory is mapped to local bus addresses 0x000000-0x03FFFC

The Demand Mode DMA FIFOs are connected to the PLX9656 DMA channels, with writes from the host on DMA channel 0 resulting in Data Being pushed into IFIFO, and reads from the host on DMA channel 1, pops data out of OFIFO. Since Demand Mode DMA is used the host process will block if IFIFO is full or OFIFO is empty.

IFIFO Ports are:
IFIFO_Data: (UFix_32_0)
IFIFO_Empty: (Boolean)
Read_IFIFO: (Boolean)

OFIFO Ports are:
OFIFO_Data: (UFix_32_0)
OFIFO_Full: (Boolean)
Write_OFIFO: (Boolean)

The following C-Code for accessing these can be used. The Demand Mode DMA application in the ADM-XRC SDK should also be consulted to see how to set up Demand Mode DMA transfers. With this module the DMA counters at Local Bus locations 0x040080 and 0x0400C0, and these should be set before starting the DMA transfer.

```
void sendData(int offset, int size)
{
  ADMXRC2_STATUS              status;
  HANDLE                              event;

  event = CreateEvent(NULL, TRUE, FALSE, NULL);
  /*
  ** Program the PCI-to-local transfer counter.
  */
  fpgaSpace[65568] = size;
  fpgaSpace[65568];

  /*
  ** Perform the DMA transfer.
  */
  status = ADMXRC2_DoDMA(card,sendbufHandle,offset,size,0x40,
                      ADMXRC2_PCITOLOCAL,0,mode,0,NULL,event);
  CloseHandle(event);
}

void recvData(int offset, int size)
{
  ADMXRC2_STATUS              status;
  HANDLE                              event;

  event = CreateEvent(NULL, TRUE, FALSE, NULL);

  /*
  ** Program the local-to-PCI transfer counter.
  */
  fpgaSpace[65584] = size;
  fpgaSpace[65584];
  /*
  ** Perform the DMA transfer.
  */
  status = ADMXRC2_DoDMA(card,recvbufHandle,offset,size,0x80,
                         ADMXRC2_LOCALTOPCI,1,mode,0,NULL,event);
   CloseHandle(event);
}
```

Two parameters are provided for the Local Bus Interface Block, the first “Local Bus script Program” is used to simulate Local Bus transactions. The second parameter “Clock Ratio” sets the ratio used in simulation between Local Bus Clock Cycles and the System Generator System Clock e.g. if this is set to 1.5 then the Local Bus program will update its outputs every 1.5 System Clock cycles.

The Local Bus Script Program can be specified as a string or as a Matlab variable. If either of these is specified as load <filename.txt> then the file <filename.txt> will be loaded and parsed in place of the string.

The following commands are parsed:

slave_read <address>
reads the contents of a local bus address and displays the result

slave_write <address> <data> [be]
write 32 bit integer <data> to address with optional byte enable signal

dma_read <length> [matlab variable]
reads <length> words from OFIFO and either displays them or stores them in a a [matlab variable]

dma_write <length> [matlab variable]
writes <length> words into IFIFO. If a [matlab variable] is specifed then this is used as data otherwise integers 1..<length> are sent

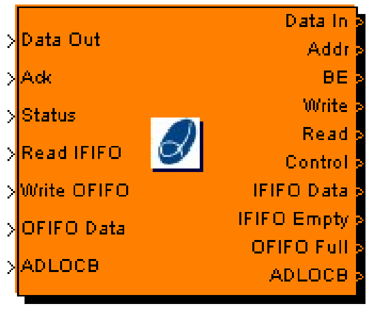
sleep <count>
this pauses the simulation program for <count> LCLK cycles

Example Program:

```
slave_write 0x40000 0xabcd1234
slave_read 0x40000
slave_read 0x40040
slave_write 0x0 0x1234abcd
dma_write 8 myfifo_in
dma_read 8 myfifo
slave_read 0x0
```

N.B. There is a 32K limit on the size of the program.

3.1.2 LBIF_XPL32

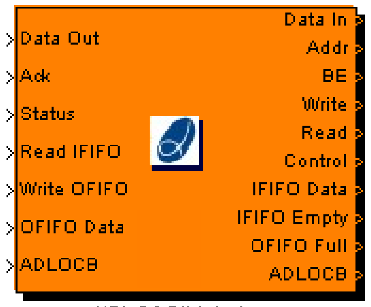


XPL 32 Bit Interface

This interface connects to the XPL PCI-Local Bus Bridge used on the ADM-XPL and ADM-XP. Functionally this module performs identically to the LBIF_PLX32 module. In implementation there are minor differences in the VHDL top level ports produced, as the data and address pins are multiplexed with the XPL bridge, unlike the PLX9656.

N.B. Due to the default clock frequency ratio between the sysgen clock domain and the Local Bus clock domain, special consideration has to be taken when generating Data Out. The asynchronous transfer register used to sample Data Out, will sample Data Out between ½ and 1 Local Bus clock cycles after Ack is asserted. Since the Sysgen clock frequency is by default twice the Local Bus clock frequency, Data Out should be held for 1 clock cycle after Ack is asserted to ensure correct transfer.

3.1.3 LBIF_XPL64



XPL 64 Bit Interface

This module is similar to LBIF_PLX32 and LBIF_XPL32, however the data widths are 64 bits wide. The ports therefore are

Control : (UFix_64_0)
Status : (Ufix_64_0)
Data_Write: (Ufix_64_0)

Addr: (UFix_16_0)
BE: (Ufix_8_0)
Write: (Boolean)
Read: (Boolean)
Data_Read: (UFix_64_0)
Ack: (Boolean)
IFIFO_Data: (UFix_64_0)
IFIFO_Empty: (Boolean)
Read_IFIFO: (Boolean)
OFIFO_Data: (UFix_64_0)
OFIFO_Full: (Boolean)
Write_OFIFO: (Boolean)

The host application should set the Local Bus Space up to allow 64 bit access. The DMA Mode bit for 64 bit access should also be set.

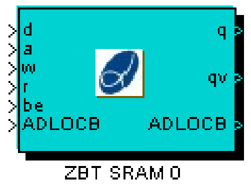
A similar format is used for the Local Bus Script Program but with one minor changes to deal with 64 bit values:
slave_write <address> <data0> [data1] [be]
will write data0 to the lower 32 bits and data1 to the upper 32 bits if specified

N.B. the address input does not change and is still refers to a 32 bit word address. So bit 0 may be ignored if 64 bits transfers are being used.

3.2 Memory Modules

Each card has a number of memory interface modules to allow access to the off-chip memory. With the SRAM modules, the memory is accessed directly. With the DRAM modules, assess is via a cache. For simulation, it is possible to initialise the memory from Matlab by specifying the contents of the variables: *sram0, sram1, sram2* etc. for the SRAM banks, and *dram0, dram1* for the DRAM banks. Each element is converted into a 32 bit integer. For 64 bit memories, 2 elements are written to each location with the lower 32 bits written first. When the simulation finishes, the memory contents are written back into the Matlab variables, however the size of the variable is used to determine how much data is transferred back to Matlab.

3.2.1 ZBT SRAM (32 bit)

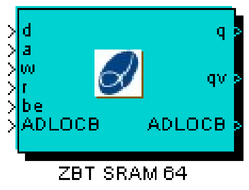


These modules implement the 32bit ZBT SRAM on the ADM-XRC-II.
Ports are:
d : UFix_32_0
a : UFix_20_0
w : Boolean
r : Boolean
be : UFix_4_0
q : UFix_32_0
qv : Boolean

If "w" is asserted then data of "d" is stored in the SRAM at address "a".
If "r" is asserted then SRAM address "a" is read and the output appear on "q" 4 clock cycles later, with "qv" going high to indicate that it is valid.

The modules for the 32 bit ZBT SRAM on the ADM-XRC-4LX and ADM-XRC-4SX are almost identical. The only minor difference is that the address input 'a' is 21 bits wide rather than 20.

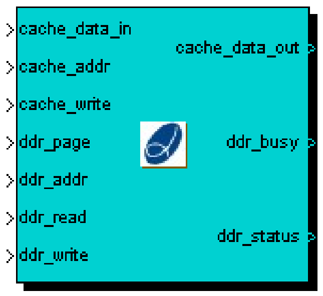
3.2.2 ZBT SRAM (64 bit)



This module implement the 64bit ZBT SRAM on the ADM-XPL.
Ports are:
d : UFix_64_0
a : UFix_20_0
w : Boolean
r : Boolean
be : UFix_8_0
q : UFix_64_0
qv : Boolean

If "w" is asserted then data of "d" is stored in the SRAM at address "a".
If "r" is asserted then SRAM address "a" is read and the output appear on "q" 4 clock cycles later, with "qv" going high to indicate that it is valid.

3.2.3 XRM-DDR SDRAM (64 bit)



XRM DDR SDRAM

This module implements and interface to the XRM-DDR SDRAM on the XRM-DDR module. The data interface to the System Generator module is via an 8K cache memory block RAM, with a 1 clock cycle latency.

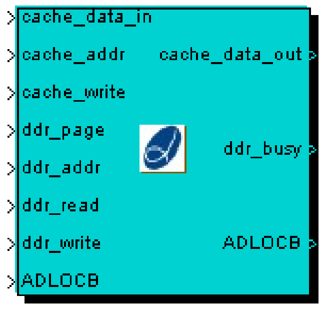
Ports are:
cache_data_in : UFix_32_0
cache_data_out : UFix_32_0
cache_addr: UFix_11_0
cache_write: Boolean

Control signals are used to burst the data from the 8K cache to the DDR
ddr_page : UFix_14_0 is used to select an 8K page in the DDR
ddr_addr : UFix_4_0 Selects a 512 byte block within the cache
ddr_read : Boolean – when asserted the 512 byte block is copied from DDR to Cache
ddr_write : Boolean – when asserted the 512 byte block is copied from Cache to DDR
ddr_busy : Boolean -- Indicates whether a burst is in operation or not. While ddr_busy is high, ddr_read and ddr_write will be ignored.

The time for a burst will vary as it may have to wait for a refresh to finish, however each burst should take between 36 (usual) and 43 (worst case) clock cycles.

N.B. The XRM DDR SDRAM module does not currently support ADLOCB connection.

3.2.4 DDR SDRAM (32 bit)



DDR SDRAM

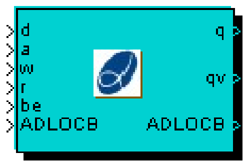
These modules implement the interface to the DDR SDRAM on the ADM-XPL and ADM-XP. The data interface to the System Generator module is via a 4K cache memory block RAM, with a 1 clock cycle latency.

Ports are:
cache_data_in : UFix_32_0
cache_data_out : UFix_32_0
cache_addr: UFix_10_0
cache_write: Boolean

Control signals are used to burst the data from the 4K cache to the DDR
ddr_page : UFix_15_0 is used to select an 4K page in the DDR
ddr_addr : UFix_4_0 Selects a 256 byte block within the cache
ddr_read : Boolean – when asserted the 256 byte block is copied from DDR to Cache
ddr_write : Boolean – when asserted the 256 byte block is copied from Cache to DDR
ddr_busy : Boolean -- Indicates whether a burst is in operation or not. While ddr_busy is high, ddr_read and ddr_write will be ignored.

The time for a burst will vary as it may have to wait for a refresh to finish, however each burst should take between 36 (usual) and 43 (worst case) clock cycles.

3.2.5 DDR-II SSRAM



DDRII SRAM 0

These modules implement the interface to the DDR-II SSRAM on the ADM-XP. The 32 bit DDR ports on the devices are mapped to single clock 64 bit wide ports. The SSRAM devices have a burst length of 4, and therefore the minimum transfer to the

RAM will take 2 clock cycles. This creates some operational limitations of the use of these devices.

Ports are:
d : UFix_64_0
a : UFix_21_0
w : Boolean
r : Boolean
be : UFix_8_0
q : UFix_64_0
qv : Boolean

For writes, the DDR-II controller waits for a second write "w" before bursting both data elements to the SRAM, in successive locations. The addresses used will be "a" when the first write is asserted and "a"+1. The byte enables can be specified separately for each write. Write bursts can be continuous, but should contain an even number of writes. "a" should be incremented every write.

Read "r" should not be asserted after an odd number of writes. It should also not be asserted one clock cycle after the last write.

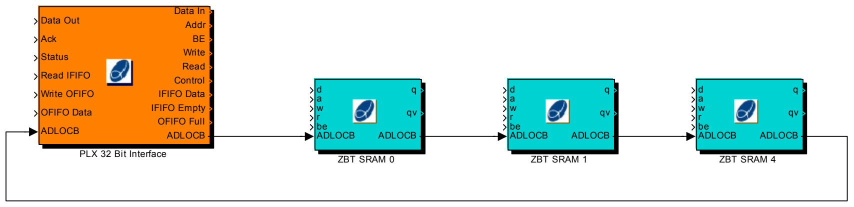
Reads to an even address "a(0)"= 0 will generate two reads 3 and 4 clock cycles after "r" is asserted. "qv" will indicate that the data in "q" is valid. For reads to an odd address "a(0)" =1, one read will be generated and the data will be valid after 4 clock cycles.

Read bursts can be continuous, for as long as "r" is asserted, and "a" is incremented every clock cycle.

For correct operation the DDR-II-SSRAM must be clocked at over 75MHz and less than 133MHz.

3.3 ADLOCB

The Alpha Data Local to On Chip Bus interface provides a fast, simple to use mechanism for high speed data transfers between the host and off-chip memory with minimal impact on the System Generator design logic.



The ADLOCB requires a Local Bus Interface module to act as the master. RAM modules are then connected in a ring as slaves. Finally the last RAM output should be connected back to the Local Bus Interface to provide a data path for reading data.

From the host, the ADLOCB is mapped into the FPGA into a 2MB window at addresses 0x200000 to 0x3FFFFF. A page/device register is provided at address 0x040100. The lower 16 bits are used to identify the off chip memory device. The SRAM devices are indexed first with the DRAM device numbers depending on the number of SRAM devices on the board. On the XRC-II, ZBT SRAM banks 0 to 5 have device numbers 0 to 5. On the XPL, the ZBT SRAM is device 0 and the DDR SDRAM is device 1. On the XP, the DDR SRAM banks have device numbers 0 to 3 and the DDR DRAM banks have device numbers 4 and 5.

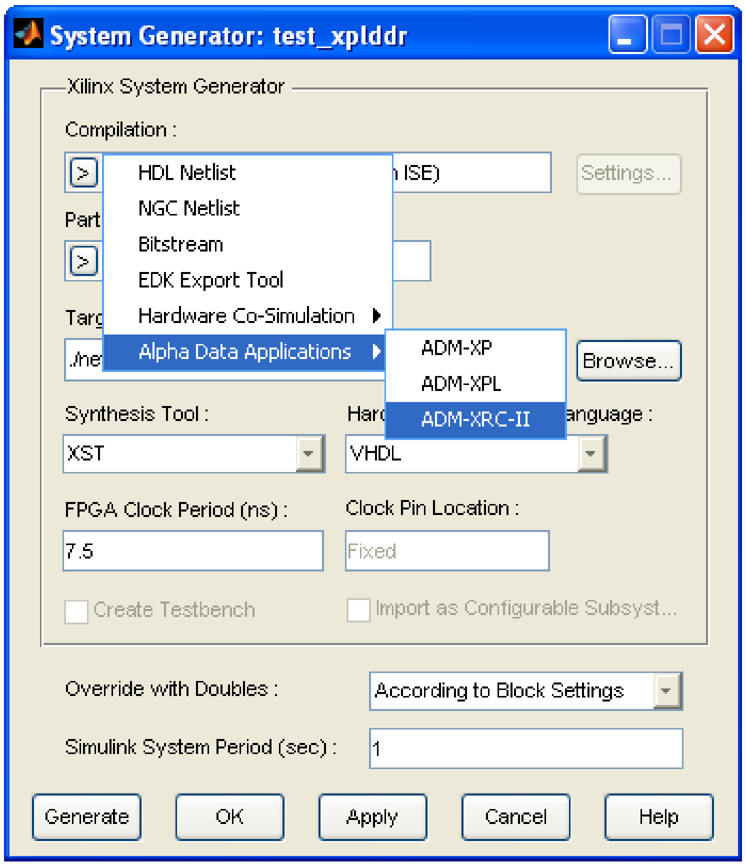
The upper 16 bits (0x040102) of the register are used to select the upper address bits of the memory if the RAM bank is larger than 2MB.

The ADLOCB system does not include any mutual exclusion logic. Host accesses will have higher priority than System Generator Logic accesses, however simultaneous access to any memory may produce unexpected results. The application should implement the appropriate mutual exclusion logic, using the status and control registers to indicate when it is safe for the host and System Generator to access memories.

The Cache in the SDRAM also introduces a twist in the data on a 16 bit boundary. 32 bit elements D0 and D1 at addresses A0, A0+1, appear on the Local bus in the order:
LA(0) : LD(31:0) = [D1(15:0) D0(15:0)]
LA(1) : LD(31:0) = [D1(31:16) D0(31:16)]

4 Wrapper File Generation

The VHDL produced by System Generator cannot handle all the interfacing requirements for the ADM-XRC series cards. Therefore the VHDL module produced must be wrapped before synthesizing and generating a bitstream.



Synthesis Tool : XST

Hard... Language : VHDL

FPGA Clock Period (ns) : 7.5

Clock Pin Location : Fixed

Create Testbench

Import as Configurable Subsyst...

Override with Doubles : According to Block Settings

Simulink System Period (sec) : 1

A new compilation option has therefore been added to the System Generator Token. When Generate is run with this option selected this will run through the System Generation process to produce VHDL. The wrapper builder will the run and create a vhdl file, a ucf. (<design>_top.vhd, <design>_top.ucf).

This wrapper builder will identify all the external ports associated with the embedded IP modules. It will then connect up the appropriate clocking circuits and tri-state buffers. It will however not identify any user defined ports (in Gateway blocks). The pin constraints for these will appear in the System Generator output constraint file (the .xcf file with version 8.1). The top level .ucf constraints and the user defined LOC constraints should be combined into a single UCF file using a text editor before running ISE Project Navigator.

If the wrapper builder detects that the wrapper files have already been generated it will provide the option to not rebuild them. It is only necessary to rebuild them if the ports in the System Generator design have changed. (i.e. after the addition or deletion of any PCI, RAM or Gateway blocks.) Selecting Yes will rebuild the wrapper files, and will also back up the old files; selecting No, will leave the wrapper files unchanged.

The wrapper building stage can be avoided completely by changing the Compilation Option back to HDL netlist, which will limit the changes to the System Generator VHDL.

In many cases the default generated wrapper VHDL and UCF might not quite match the users requirements. However these can be modified, (e.g. to change the system clock pin, or add in timing or area group constraints) with any text editor or within the Project Navigator environment. In these cases after each generation the user should select not to rebuild the wrapper files, or change the Compilation Option back to HDL netlist.

N.B. for Synplify/Synplify Pro Users, the UCF file generated uses <> as bus delimiter symbols, however by default Synplify and Synplify Pro use () . Therefore before running Synthesize, a Synplify Constraints file (.sdc) containing the following specification:

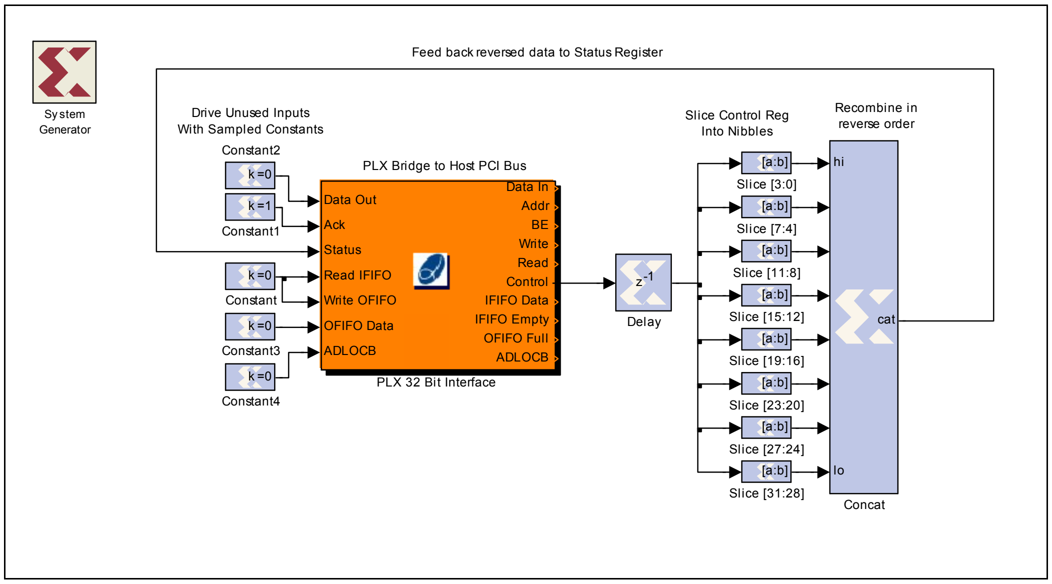
```
define_global_attribute syn_edif_bit_format {%n<%i>}
```

must be added to the project. Select the "Properties…" option from the Synthesize Process menu, and select the Constraint File Options Tab which allows the specification of a file. A suitable file can be found in %ADMXRC_SDK4%\fpga\vhdl\common\synpro_bus.sdc if the ADM SDK is installed.

5 Example Applications

Three simple applications are provided for each board. These applications demonstrate basic connections to the Local Bus Interface, basic I/O and a simple memory application. Separate applications for each board are provided in examples/admxrc2, examples/admxpl, examples/admxp. Matlab scripts simple.m and memory.m are provided in the examples directory to run these applications from within Matlab.

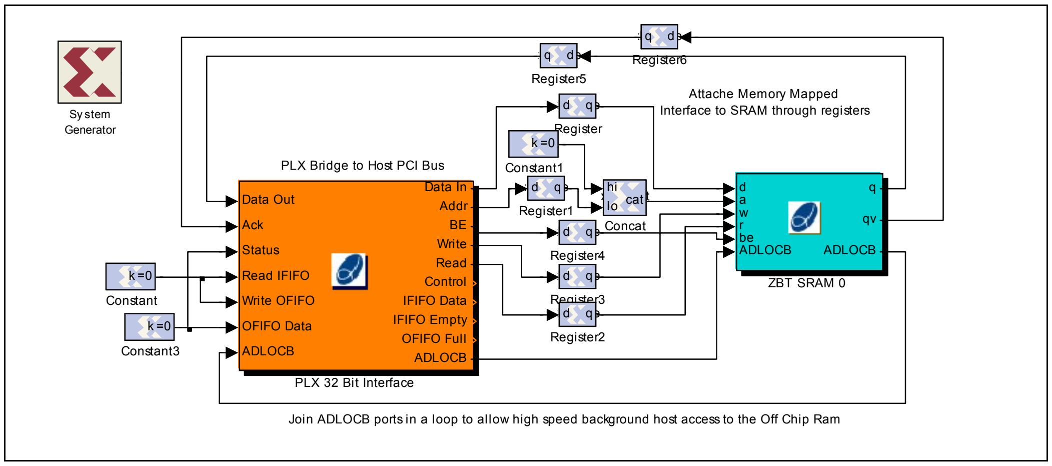
5.1 Simple



This example here shows the simplest possible interface using the PLX/XPL Interface Module. Data from the control register is nibble reversed and then output to the status register.

This example connects up pins on a front panel XRM I/O to the control and status registers. The pin definitions for frontio_in and frontio_out can be included in the Gateway Blocks.

5.2 Memory



This example demonstrates how to connect up a simple memory mapped interface. It also shows how to connect up the ADLOCB ports in the modules.